Tata Programming Language Manual

By:

Buko Obele (aso22@columbia.edu)
Rafael Pelossof (rp2056@columbia.edu)
Artem Litvinovich (al867@columbia.edu)
Isaac Krieger (ibk2002@columbia.edu)

Contents

[1] INTRODUCTION

Tata is a simple, robust, web-services oriented, fourth generation, base functional, highly expressive, portable, very-dynamic language.

1.1 Motivations

Tata is geared towards developers who are implementing next generation web services in more traditional business languages (Java, C++, C#) but are in need of a powerful scripting language that can test, integrate, and weave together these new web services into a new breed of networked applications.

1.2 Goals

Tata's chief goals are simplicity and expressiveness. With only a few short and simple lines of code, developers can create complex programs that work with XML, HTTP, and SOAP and other emerging technologies. Tata programs are designed to be very readable and require almost no documentation beyond the code itself.

To achieve such lofty goals, Tata borrows heavily from existing language paradigms while striking out into bold new territory. The best features from declarative languages like SQL and XPath, object-oriented languages like Java and C#, very high level scripting languages like Python and Ruby, domain specific languages like SML and XSLT, and functional languages like Lisp and OCaml have been woven together into a simple, powerful and elegant new paradigm.

1.3 Major Features

· *Base Functional*

In terms of sheer expressive power, few languages can approach purely functional languages such as Lisp. These languages derive their expressiveness from a highly unified computation model: everything is either a function or a list that functions can operate upon. Tata borrows heavily from the functional model. In Tata, everything is a function. Variables are functions, function parameters are functions, function results are functions, identifiers are functions, and, of course, functions are functions. Using this highly unified computation model, Tata can achieve feats such as closures and self-modifying programs. Functional languages don't enjoy much popularity because they introduce a certain amount of artificial complexity when solving problems. The computation model of the language can intrude and confuse developers. Tata solves this problem by leaving out many of the more esoteric, arcane, and complex constructs of traditional functional languages. The result is a simple computational model known as "base functional."

· *Easy to Read*

Another problem with many functional languages is they are often quite difficult to write and read. Tata solves this problem by removing all of the cruft commonly found in traditional languages—things like semicolons, parentheses, and even equal signs—all symbols that the compiler forces the programmer to place though the programmer gains nothing. This makes Tata extremely easy to read. It's pretty much all words!

· *Highly Extensible*

Tata purposefully avoids the more traditional approach of defining a "general purpose" language that can be used to solve every other problem. Instead, Tata acknowledges that it can't solve every problem and invites developers to extend it. Using its carefully designed extension framework, developers can easily add new data types, instructions, and even evaluation semantics to the Tata interpreter. Entire "mini-languages" (eg Zawk, or TSDL (see below)) can be simulated within Tata. Tata achieves this high extensibility by removing the line between a language and its API. Libraries in Tata may add new language constructs, functions and even interpretation semantics.

· *Domain Specific*

Domain-specific solutions are easier to understand and implement compared to "general purpose" solutions. A major aspect of Tata's design philosophy is that the domain specific solution is preferred. When trying to solve problems within a specific knowledge domain, the basic entities and processes of that model domain should be evident within the script. For example, when working with XML, XML documents, XSL transformations, and XPath expressions should all be first class variables. When constructing web services, request processing logic should *be* the scripting logic.

· *Declarative*

It's always easier to tell a computer "what" to do rather than "how" to do something, Tata aims to be declarative whenever possible. Relying on rich instruction sets including "smart iterators", data types, and its functional ancestry, Tata encourages developers to think carefully about data, relationships between data, and how to transform data from one process so that it can be passed to another process.

· *Very Dynamic Language*

Current very high-level (VHL) languages increase programmer productivity by removing the burdens of garbage collection, strong typing, and interface specification. Tata also does this--and more. A comprehensive type-conversion framework ensures that programmers almost never have to worry about typing and interface defintiion. The interpreter also includes a garbage collector so programmers can ignore arcane allocation details. Beyond this, Tata also offers developers greater *expressive* flexibility. For example, variable names can be defined dynamically at run time and functions can be built "lego-style" from other functions to produce some whole new methods for solving problems.

· *Strongly Structured*

One of the problems of very high level languages is that they can quickly become bug-ridden because their looseness introduces a certain level of opacity into the program flow. Scenarios can occur where developers aren't sure what type of variable they're dealing with or what function is actually being invoked. Tata avoids this by imposing a stict program structure and as a result it's always clear exactly what the interpreter will do when it encounters a given block of code.

1.4 Simple Examples

Here is the classic "Hello World" program in Tata:

```
print  "Hello World"
```

Here a string is being passed to the *print* function. Of course, "Hello World" is also a function: it's the String Identity Function that returns itself.

Here is a program that calculates the sum of 10 to 1000.

```
set “total” 0
print
        foreach “num” num_range 10 1000
        {
                set “total” add get “total” get “x”
        }
```

Here the *print* function is being invoked with the *foreach* function as an argument. The *foreach* function iterates over a range of numbers and continually executes its third argument—a block function because blocks, defined by '{}', are indeed a function in the language. Even the numbers 10 and 1000 are functions—they are Number Identity Functions--functions that always return some constant numerical value. This unified computation model allows tata to perform such neat tricks as chaining functions and even currying.

Here's an example of the flexibility of Tata. Here the names of variables will be dynamically bound.

```
set "people"
        (
                "Jimmy James"
                "Johnny Jones"
                "Mary Mono"
                "Bobby Bee"
                "Sam Summers"
        )

set "ages" ( 23 29 24 19 41 )

for_each “x”  get “people”
{
        set get “x” index get “loop.cnt” get “ages”
}
```

Here the *set* function takes an even number of parameters: each odd parameter must be a string and each even parameter is an object. Two collections are defined: a list called "people" and a list called "ages". The *foreach* function iterates over its first parameter, the list of people's names, and then executes its (not-quite) third parameter, the *set* function. Here the *set* function binds a variable defined by the current value of the loop. The value of this parameter is a person's name. The variable that set defines is the corresponding age of the person in the "ages" list which is retrieved by indexing into the "ages" list based on the current index into the "people" collection. After having evaluated this function the following code:

```
print get “Bobby Bee”
```

will print '19'. This works because Tata enforces no formal definition of an 'identifier'—instead, even the variable names you choose can be dynamically constructed.

The real goal of Tata is to work with web services and XML on both the client and server. Here's a small Tata program that's a fully functioning Newsreader. RSS ("Really Simple Syndication") is an XML vocabulary for describing a set of headlines on a current website. Many web sites, particularly weblogs and news sites, will offer an RSS service so programs called "newsreaders" can download and display their current headlines. Users use newsreaders to quickly skim and aggregrate the headlines of hundreds of sites. Here is a Tata newsreader:

```
# a simple newsreader of RSS 2.0 weblogs. a great example of Do More
With Not Quite So Much
# DMWNQSM!

# set the weblogs we're going to read
set "blogroll"
         (
        url "http://radio.weblogs.com/0106046/rss.xml" # weakliem
        url "http://www.scripting.com/rss.xml" # winer
        url "http://nyt.weblogs.com/rss.xml" # new york times
frontpage
        url "http://partners.userland.com/nytRss/technology.xml" #
nyt technology
        url "http://partners.userland.com/nytRss/arts.xml" # nyt
arts
        )

# set the xslt we use to do rss2txt
set "rss2txt" xml.xslt io.file "test/rss2txt.xsl"

# create the file we're going to append the docs to
set "news.txt" io.file "news.txt"
io.create get "news.txt"

# tmp file to store each one
set "tmp.txt" io.file "tmp.txt"
io.create get "tmp.txt"

foreach "feed" get "blogroll"
{
          xml.transform get "rss2txt" xml.doc get "feed" get
"tmp.txt"
          io.append get "tmp.txt" get "news.txt"
}
```

This script might not make much sense unless you understand XML technologies like XPath and XSLT but see if you can figure out what it does!

Summation

Tata is a powerful tool for dealing with many essential problems of the network era. It allows developers to quickly and easily weave together business components and integrate information systems.

[2] TUTORIAL

2.1 What is Tata?

tata is a new scripting language created to serve as a testing harness for developers implementing XML web services. Essentially tata provides a very simple mechanism for working with today's cutting edge technologies like XML, HTTP, and SOAP in a way that is both intuitive, elegant and powerful. For developers and administrators working with these technologies tata provides a solution to several problems dealing with invoking, testing, and integrating web services to rapidly construct new solutions.

2.2 Hello World

Moving right along, here's the perennial Hello World program in tata:

```
# an old favorite, the 'hello world' program
println “hello world”
```

To run this program just cut and past the code into a file like 'hello-world.tata.' Then invoke your tata interpreter on the script, then sit back and enjoy the show!

2.3 Function Junction

Pretty much everything you need to know can be learned just from examining the 'hello world' program.

Note the line that begins with '#': this line is a comment that's ignored by the interpreter. The '#'-sign warns the interpreter to ignore it and all text following it until the end of the line.

Next, we encounter the 'println' function. As you might've guessed, this command prints out some text to the system console followed by a newline.

What's actually going on here is a bit more complex. println is a function whose argument is any object that can be converted to a String. Because tata is a weakly typed language println can be used print out almost anything including strings, numbers, and even HTTP responses. Once println evaluates it returns the string that it printed out.

Although println prints strings there are no strings in the tata language. In fact, there are no user-visible types in the tata language at all. In fact, the only type of 'object' that exists in the language are functions. Functions work just how you might expect: they take some arguments and return some result. In the 'hello world' program above the character sequence “hello world” is in fact a function known as the 'string literal' function. When this function is evaluated it returns a string as defined by the parser responsible for compiling the script. Usually the string returned is just whatever text you type in between the double quotes though there are some special characters that can change this.

The previous point is particularly important so we'll repeat it: *everything in tata is a function*. A tata program is defined to be a sequence of functions placed on a stack and its execution is merely the process of popping each of these functions off the stack and evaluating each function. Because everything is a function this provides the language with a *highly unified computation model*; this fancy term simply means it's possible to do some neat things in tata that you can usually only do in purely functional languages. (tata is not a purely functional language nor does it aim to be. Functional languages, while tremendously powerful, are also often very confusing. tata instead tries to borrow as much as possible from functional languages while adhering to its core principle of simplicity).

To drive this point home, let's rewrite the 'hello world' function. Consider:

```
# a rewrite of the 'hello world' program
println concat “hello” “ world”
```

Here, it looks like we're passing the concat function to the println function. We're not; we're passing the result of the concat function. As you might guess, the concat function simply takes two strings and appends the second one to the first and returns the resultant concatenated string.

2.4 Types and Conversions

Before we go any further, let's just say a quick something about types and variables in tata. As you already know, everything in tata is a function. There really is no concept of types as you might encounter in most other languages (partially because there is no real concept of variables or identifiers). Most of the time, this illusion of typelessness is preserved. The programmer will never explicitly assign a type nor will he ever have to perform any kind of 'type checking' or 'type casting'. For example, the following script works just fine!

```
# example of weak typing
println add “1.0” 1.0
```

The only time the programmer has to worry about types is during type conversions. We say “only time” because tata strives to provide every reasonable type conversion you might think of. Strings, for example, can be transparently converted to files, URLs, XML documents, numbers and even HTTP connections. The only time tata won't provide a type conversion for you is when it'd make the program confusing (there is such a thing as scripts that are *too succinct*. Trust us). In practice, you'll likely never get type errors unless your program is functionally incorrect because you're using the wrong function or forgot some function.

2.5 Variables & Scope

If there's nothing but functions in tata then you might be wondering if the language supports variables (as just about every other language in the world does). The answer is *yes* ... and *no*. There is no concept of variables or identifiers or any of that silly stuff. Instead, there are two special functions that allow you to store objects and later retrieve them. The set function does the storing and the get function does the

retrieving. Here's an example:

```
# an example of using get and set
set “str” concat “hello” “world”
println get “str”
```

What's going on here might be a little confusing at first but you can probably figure it out. The set function takes two arguments: the first must be functionally equivalent to a string while the second can be anything. When it's evaluated the set function stores its second object in the current scope under its first argument. The get function simply takes a string and retrieves the object from scope.

The concept of scope is the same as you might encounter it in other languages but it's enforced more strictly. In tata, every function, when it's evaluated, is assigned a scope. This scope is “chained” to the parent function which caused the function's evaluation. For example, in the previous example, the set function causes the concat function to evaluate its argument and return a result. So any objects in the scope of the set function will also be visible to the concat function.

Knowing what you do now about scopes, you might be confused why the get function is able to retrieve the “str” variable since it doesn't appear to share a scope with the set function. In fact, the get function does share a scope—it's the special “script scope.” There's actually really nothing special about the script scope—it's simply the scope assigned to the “implicit script function”. The implicit script function is the function that causes execution of scripts: it simply iterates over all of its arguments (which must be functions) and evaluates each one. When you write tata programs what you're really doing is passing arguments to the implicit script function! This means that both the get and set functions do share a common parent function which is why the variable set by the previous is visible to the former.

2.6 Conditional Logic

The language provides the core foundation evaluating conditions. Generally, some objects can be interpreted as the boolean state-variables true or false. For example, a number is considered true if it's non-zero. This interpretation is actually nothing but a conversion. Most of the time, you can feel free to plug any object where a boolean is expected and expect it to work.

2.7 Collections

One of the core abstractions of the language is the concept of a 'collection'. A collection is just a bunch of things grouped together into a single object. Collections in tata are usually defined using the '( )' list function. For example, consider this program:

```
foreach “string” ( “hello” “world” “isn't” “this” “cool?” )
{
          print get “string”
          print ” “
}
```

It's probably pretty clear to you what's going on here. The foreach function is one of the many collection functions that perform some on a collection. (Really, it iterates

over a collection and executes its body. *Actually*, you might be interested to know there's no such thing as the 'body' of a foreach function. Instead the foreach function takes three arguments: a string, a collection, and a function. This is known as *function currying*. In tata '{ }' is a special function known as the *block function* which executes every function within it and returns the value of the last function). The previous example is of course equivalent to:

```
set “coll” ( “hello” “world” “isn't” “this” “cool?”)
foreach “string” get “coll”
{
          print get “string”
          print ” “
}
```

This second example simply emphasizes the point that '( )' the list function is its own function and not part of the foreach function. You can use it anywhere you need a collection.

2.8 Where To Go From Here

Now that you get the basics of the language (it's an extremely simple language!) the best thing to do is look at the language reference manual. There you'll find detailed explanations of what each function does, tons of examples, and this, combined with some basic intuition, should put you well on your way.

[3] TATA LANGUAGE REFERENCE

3.1 Introduction

Tata is a language inspired by domain specific, declarative languages (SQL, XPath), scripting languages (Python, Ruby), and functional languages (Lisp, OCaml). Tata allows developers working with next generating web services to weave together simple but powerful solutions while being highly extensible.

3.2 Lexical and Parsing Conventions

3.2.1 Lexical Definition of a Script

Lexically, the fundamental unit of a script is a token. A script is sequence of tokens. Each token is a set of one or more identifiers that can be unambiguously mapped to a corresponding function. Identifiers correspond to logical entities within the script such as functions or namespaces. When an identifier appears within a script the corresponding function is said to be *declared.*

3.2.2 Lexical Constraints

Identifiers are an arbitrarily sized sequence of alphanumeric characters. Identifiers MUST NOT contain any white space characters (except for the special string constants, see Constraint Exceptions). Identifiers MUST NOT contain any punctuation characters, particularly '@', ':', '"', ''', or '.'. Identifiers MUST start with a letter.

3.2.3 Constraint Exceptions

The following exceptions to the lexical constraints exist: the declaration of string constants, number constants, the composition of code blocks, and the definition of lists. These exceptions exist primarily to make scripts easier to read and write; they are syntactic sugar. Identifiers that do NOT obey the lexical constraints of identifiers declare these four special functions.

3.2.4 Translation Model

Translating a script involves mapping identifiers to the logical entities (usually functions) they represent and constructing the equivalent execution model. The following phases occur during script translation:
Tokenization: Translation of a script begins by separating each sequence of characters (a 'word') separated by white space. Comments can also be made by adding a '#' before the comment and thus ending a line.

Token Normalization: At this point, Constraint Exceptions are enforced. "String Value" function declarations, which may consist of multiple tokens, are collapsed into a single token. Similarly, block composition and list definition function declarations may be expanded or collapsed at this time.

Function Resolution: Each token is unambiguously mapped to a corresponding

function. During function resolution, library loading also takes place. If a declared function is located in an external library that library will be loaded within the script execution environment and initialized at this time.

Value Stack Construction: Functions are pushed onto the value stack in the reverse order that they are declared. For example, this script:

echo “Hello World”
results in the following Value Stack:
[EchoFunction][StringValueFunction(“Hello World”)].

3.2.5 External Libraries

External libraries containing additional related functions may be dynamically loaded during script translation. Each external library consists of a set of functions which will be made available upon library loading, a set of conversion rules for defining conversions between proper objects, and lifecycle logic that may be executed each time the library is loaded, a script referencing that library is executed, and the library is unloaded.

In order to utilize external libraries the script must contain special *import commands* that will be passed directly to the translator and interpreter. Import commands must be the first statements in a script; they may be preceded only by white space. Import commands consist of three separate parts: the special keyword ‘import’ followed by a library qualified name followed by the word ‘as’ followed by a library alias. For example, ‘import tata.io as io’ causes the ‘tata.io’ library to be imported through the alias ‘io’. Exactly how library qualified names are resolved and loaded is undefined and may vary between interpreters.

Once imported, functions within the external library can be referenced through the namespace resolution operator ‘:’. The library alias must appear before the namespace resolution operator and the name of the function must appear immediately after the resolution operator. For example, ‘io:copy’ references the ‘copy’ function within the tata.io library.

3.3 Execution Model

3.3.1 The Value Stack

A script is defined to be a stack known as the Value Stack. The state of the Value Stack defines the state of the script while it’s being executed. The Value Stack contains one or more functions.

3.3.2 Execution of a Script

A script, once it’s been translated into a Value Stack, is defined to be a special implicit (or undeclared) function. This undeclared function is known as the Implicit Stack Function (ISF). A script is executed by invoking the ISF.

The ISF works as follows: if the Value Stack is empty, the special Null Value is returned otherwise the first function is popped off the top of the Value Stack, invoked,

and its return value is saved. If the stack is empty at this point then the return value of the previous function is returned as the value of the ISF. Otherwise the next function is popped off the stack, invoked, and its return value is saved. The process now repeats until the stack is empty.

For example, consider the example script:
echo “Hello World”
This script is translated into the Value Stack:
[EchoFunction][StringValueFunction(“Hello World”)]

When the ISF is invoked it pops the ‘echo’ function off the top of the Value Stack and invokes it. When the ‘echo’ function is invoked it pops the String Value Function off the top of the Value Stack, invokes the String Value Function and prints its return value, “Hello World”, to the console. At this point the Value Stack is empty so script execution is halted and the return value of the ISF is the return value of the ‘echo’ function which is the String that was printed, in this case “Hello World.”

3.3.3 Definition of a Function

A function is a singular unit of work. Functions are invoked—they are not called and parameters are never explicitly passed. Instead, a Function usually retrieves parameters by popping other functions off the Value Stack and evaluating these parameter functions by invoking them and converting their return values to types that the function can understand. When invoked, Functions MUST return a value even if this value is the Null Value. The return value of a Function may be another Function.

3.3.4 Lifecycle of Function Invocation

When a Function is invoked it proceeds through three distinct phases. First, the *Initialization Phase* occurs. During this first phase, unless the function explicitly forbids it, the function’s context will be constructed and any other initialization logic will be executed. Next, *Evaluation Phase* occurs during which the function’s main execution logic will be executed and its return value will be calculated. Finally, *Cleanup Phase* occurs during which the Function may release any resources such as TCP/IP sockets or database connections that it acquired during initialization. At the end of clean up, the return value of the function is returned to the function that invoked it or, if the function is the Implicit Stack Function, to the program that executed the script.

3.3.5 Function Idempotentence

Function execution is always idempotent. This means that there must be no script-level difference between invoking a function one time and invoking it an unnamed number of times. The primary consequence of the idempotent constraint is that functions must be totally stateless. Regardless of how many times they are invoked or their invocation context they should still perform the same application logic.

3.3.6 Function Scopes, Contexts, Variable Resolution

Every Function that is invoked is invoked within its own Function Context. The primary purpose of the Function Context is to provide a scope for all variables that are

read or written using the core Get and Set functions. When one Function invokes another Function the invoked function *inherits* the invoker's context. All variables visible within the parent Function will then become visible within the child Function's context. Note that the Implicit Stack Function does possess a Context, which, because it is always the first function executed, acts as a kind of global context.

3.4 Core Functions

3.4.1 Core Function Definition

Technically, core functions are not integral to the language. They are not treated differently from non-core functions or granted any special privileges. The only truly "core" function is the Implicit Stack Function, which is defined and invoked by the interpreter. Core functions are necessary, and defined within the language reference, because they provide basic functionality necessary to make the language usable in the majority of scenarios. But, for optimization or specialization purposes, the interpreter in certain contexts might in fact remove core functions. Still, one of the primary goals of Tata is provide a foundation on which smaller 'languages' can be built through external libraries and this goal is achieved by limiting the core functions to the smallest number possible.

3.4.2 Value Functions

Value Functions are a special class of functions that always return some constant value such as a string or a number whenever they're invoked. In a sense, value functions can be understood as "data types" in traditional, strongly typed languages but this is misleading because value functions are far more flexible than basic data types and no type-checking is ever enforced by the interpreter. Because value functions are not integrated into the core language, library developers are encouraged to add new data types through the external library mechanism.

3.4.3 Conversions and Proper Objects

The result of invoking a Value Function (or, in fact, any other type of Function) is often not necessarily a function but a special value known as a 'Proper Object'. All Functions are also 'Proper Objects' but the reverse is not true. This is somewhat confusing but it's an important point: a String Value Function is NOT a String, instead it is a function that returns a 'proper object' that may be treated like a String.

The interpreter provides facilities to convert any proper object and any other proper object. A Conversion is defined to be a 3-tuple consisting of the following elements: a source type, a destination type, and a "Converter" which implements the conversion logic necessary to convert objects of the former into instances of the latter. Library developers may add new conversions to the language. Multiple conversions may be defined for the same type transition and core conversions, e.g. String to Number, can in fact be replaced or extended. Conversions are not part of the language since they can be dynamically replaced, extended or redefined.

3.4.4 Core Function Reference

String Value Function: The String Value Function is a Value Function that returns some constant String. String Value Functions are declared by surrounding a sequence of characters (including whitespace) with double quotes (“). For example, “Hello World” corresponds to a String Value Function that always returns the proper String “Hello World”. Strings may also contain the special character sequences \n, \t, \r, and \\ which corresponds respectively to the new-line character, the tab character, the line feed character and the forward slash character.

Number Value Function: The Number Value Function is a Value Function that returns some constant numerical value. Number Value Functions come in two general types: Long Value Functions and Double Value Functions. Long Value Functions always return a 64-bit integer value in the range -9223372036854775808 to +9223372036854775808. Double Value Functions always return a 64-bit IEEE 754 floating-point value with the range ±4.9E-324 to ±1.7976931348623157E+308. Number Value Functions are declared by writing a sequence of numbers. Generally, if the sequence of numbers contains a decimal point it will be mapped to a Double Value Function otherwise it’ll be mapped to a Long Value Function.

Null Value Function

The Null Value Function is a special function that always returns the special Null value. The null value function is declared by the token ‘null’.

True Value Function

The True Value Function returns the logical truth-value, *true*. It is declared by the token ‘true’.

False Value Function

The False Value Function returns the logical truth-value, *false*. It is declared by the token ‘false’.

URL Value Function

The URL Value Function returns a Uniform Resource Locator when invoked. It is declared by writing the token ‘url’. When the function is invoked it will pop the next function off the stack, invoke it, and convert the return value to a String and finally convert the resultant string into a URL. Usually, the URL function is simply followed by a String Value Function e.g.: url “http://www.yahoo.com”.

Set Function

The Set Function is responsible for storing variables within the context of the function that invoked it. When invoked, the Set Function will pop two functions off the Value Stack. The first Function will be evaluated to a String (it will be invoked and, if necessary, its return value will be converted to a String) and the second function will be invoked and its return value saved. Next the return value of the second function will be bound in the context of the parent function under the String of the first function. Finally, the return value of the second function is returned. For example, set “myName” “Johnny” will store the String “Johnny” under the name “myName”.

Get Function

The Get Function is responsible for retrieving variables bound within the

context of the function that invoked it. When invoked, the Get Function will pop a function off the Value Stack and evaluate it to a String. It will then attempt to retrieve the variable bound under that String in the context of the parent function and return that value otherwise it will return the special 'null' value.

Echo Function

The Echo Function is responsible for printing strings to the console. When invoked, the Echo Function will pop a function off the Value Stack and evaluate it to a String. It will then print this String to console and return it to the caller. For example, echo "Hello World" will both print "Hello World" and return the proper object ["Hello World"].

Foreach Function

The Foreach Function is used for looping over collections and processing each item in a collection. When it is invoked it pops three more functions off the Value Stack. The first function is evaluated to a String. This String is the name of the "loop variable"—the Foreach Function will loop over the collection and for each item in the collection, bind it to its context under the given string. The second function is evaluated to yield a collection. The final function is the "loop function". For each item in the collection, this loop function will be evaluated. The Foreach Function returns the collection that was iterated over.

If Function

The If Function is used to implement basic branching logic. When it is invoked, it iterates down the stack, looking for the last function that can play a role in the If function. Once that's defined, it pops at least two but potentially more functions off the Value Stack. The first function is evaluated to a Boolean value. If it's true, then the second function will be executed. If it's not true, the If function may peek at the top of the stack and look for an else function and execute that—otherwise the If function simply returns 'true' or 'false' and control is passed back to the Implicit Stack Function.

Integral Boolean Functions

Several integral Boolean functions are allowed which generally only apply to Strings (or Objects that can be converted to Strings) and Numbers (or Objects that can be converted to Numbers). The Boolean predicate functions are followed by two arguments which are evaluated to Strings or Numbers and can be tested for equality (eq), inequality (neq), greater than (gt), less than (lt), greater than equals (gte), less than equals (lte), and function (and) and or function (or). The urinary Not Function expects to be followed by another function, which will be evaluated to a Boolean truth-value and then returns the negation of this truth-value.

Block Functions

Two functions are used for defining 'blocks'. Blocks are used to compose related groups of functions into a single function that possesses the same context. A 'Begin Block' Function can be declared by writing an open curly bracket ('}') and an 'End Block' Function is declared by writing a closing curly bracket ('}').

When the ‘Begin Block’ function is executed it enters into an infinite loop. During an iteration of the loop the function at the top of the Value Stack is popped off and invoked and its return value saved. When the ‘End Block’ Function is executed it halts the loop and forces the ‘Begin Block’ Function to return the result value of the last function executed before the ‘Begin Block’ Function. Blocks used in this manner may nest infinitely. It’s also interesting to note that the interpreter need not be aware of blocks—they are a construct not integral to any script and instead introduced through the ‘Begin Block’ and ‘End Block’ functions.

List Functions

Often times it’s necessary to construct a collection from several variables. Two functions, the ‘Begin List’ Function and the ‘End List’ Function allow this. The Begin List Function is declared by writing an open square brackets (‘[‘) and the End List Function is declared by writing a close square brackets (‘]’). When the Begin List Function is executed it enters into an infinite loop. During an iteration of the loop, the function at the top of the Value Stack is popped off, invoked, and its result value is added to a collection. When the End List Function is executed it forces the Begin List Function to break out of the loop and return the collection it has built. For example, [“John Doe” 23 “Suzy Seraph” 20 “Max Moore” 21] constructs a collection of the elements String[“John Doe”], Long[23], String[“Suzy Seraph”, Long[20], String[“Max Moore”, and Long[21].

3.5 Core Functions Listing

The core functions are those that are guaranteed to be provided by every tata interpreter. They are as follows:

String literal function

The string literal function is invoked by surrounding some sequence of characters with double quotes eg “hello world”. When evaluated it returns the characters within the quotes.

Number literal function

The number literal function is invoked by typing some sequence of characters that's understood to be a number eg .12, 3, 4.5. Numbers can either be decimal or integer. For example, consider the script:

an example of the number literal function

print 12.5

This will print the string “12.5” after converting the number to a string.

[string] println [string]

The println function prints some string to the system console followed by a newline. The println function expects a single argument that must be functionally equivalent to a string. The function returns whatever it prints out.

[string] print [string]

The print function prints some string “as is” to the system console. The print function expects a single argument that must be functionally equivalent to a string. The function returns whatever string it printed to the console.

true

The true function returns the logical constant boolean value 'true'.

false

The false function returns the logical constant boolean value 'false'.

null

The null function returns the logical value 'null'. This value is understood to be the absence of an object. Null is functionally equivalent to the boolean value 'false' and the number zero.

[object] set [string] [object]

The set function binds a variable into the current scope. It takes two arguments: the first must be functionally equivalent to a string and the second may be anything. When executed, the object will be bound into the current scope under the label assigned by the first argument. The function returns whatever was bound into the scope.

[object] get [string]

The get function retrieves an object that's been stored in the current scope or in any parent scope of the current scope. It takes a single argument: a string and it returns the object bound under the label defined by the string in the closest scope. If no object is bound under the label in scope of the function then null is returned.

[object] unset [string]

Certain objects may be expensive to keep in memory for long periods of time (eg XML documents). The unset function allows an object to be removed from the current scope so that it may be garbage collected. It takes a single argument, which must be a string and returns the object bound in the scope under the label. It also removes this object from the scope. It also returns this object.

[object] assign [string] [object]

The assign function assigns new object values to an existing label that's bound somewhere in scope. Assign never creates a new variable (like set): instead it searches up through its current scope until it finds a variable bound under the label define by its first argument. It then unbinds the old value and returns it and binds the new object value defined by its second argument.

[URL] url [string]

The url function expects a single argument, a string, and returns a URL representation of the string.

[Collection] ([f1] [f2] ... [fn])

The list function takes an arbitrary number of function arguments and evaluates each one. The result of each function is then stored in a collection which is returned. For example:

example of the list-collection function

set “var” (1 2 3 4 5)

println get “var”

The previous script will print '(1 2 3 4 5)'. Here the list function evaluates each number literal function and stores the resulting number in a collection that is bound under the label “var”.

[object] { [f1] [f2] ... [fn] }

The block function takes an arbitrary number of function arguments and evaluates each one. It then returns the result of the last function it evaluated. For example:

```
# example of the block function

set “10”

{

set “var” 5

assign “var” add “5” get “var”

}

println get “var”
```

This script will print '10' since the result of the assign function is the result of the add function which is the number ten.

[string] concat [string] [string]

The concat function takes two arguments that must be strings and returns the string created by appending the second argument to the first. See if you can figure out what this script does:

an example of the concat function

println concat concat “tata” “says” concat “hello” “world!”

[boolean] eq [object] [object]

The eq function takes two arguments and returns true if they are equal else false. Currently this function only really works for strings and numbers—all other logical comparisons are iffy.

[boolean] not [boolean]

The not function takes a single argument that must be functionally equivalent to a boolean. It then returns the negation of this boolean.

[boolean] gt [object] [object]

The gt function takes two arguments and returns true if the first is greater than the second else false. Currently this function only really works for strings and numbers—all other comparisons are iffy.

[boolean] gte [object] [object]

The gte function takes two arguments and returns true if the first greater than or equals to the second argument, else it returns false.

[boolean] lt [object] [object]

The lt function takes two arguments and returns true if the first is strictly less than the second else false.

[boolean] lte [object] [object]

The lte function takes two arguments and returns true if the first is less than or equal to the second else it returns false.

[object] if [boolean] [f1] (else [f2]) endif

The if...endif function takes two arguments: the first must be functionally equivalent to a Boolean and the second is a function. If the Boolean argument is true then the function f1 will be evaluated and the if...endif function will return the Boolean value true. If the Boolean argument is false and en else is present then f2 will be evaluated; the if...endif function will return the Boolean value false.

[object] when [boolean] [f1]

The when function takes two arguments: the first must be functionally equivalent to a Boolean and the second is a function. If the boolean is true then the function f1 will be evaluated and the when function will return the Boolean value true. Otherwise the when function returns the Boolean value false.

[object] unless [boolean] [f1]

The unless function takes two arguments: the first must be functionally equivalent to a Boolean the second is a function. If the Boolean argument is false then f1 will be evaluated and the unless function will return the Boolean value false. If the Boolean argument is true then f1 will not be evaluated and the unless function will return the Boolean value true.

[boolean] and [boolean] [boolean]

The and function takes two Boolean arguments and performs the logical AND operation. The Boolean value of this operation is then returned.

[boolean] or [boolean] [boolean]

The or function takes two Boolean arguments and performs the logical OR operation. The Boolean value of this operation is then returned.

[number] add [number] [number]

The add function takes two numeric arguments and sums them and returns their sum as a number.

[number] sub [number] [number]

The sum function takes two numeric arguments and subtracts the second from the first and returns this difference as number.

[number] mul [number] [number]

The mul function takes two numeric arguments and multiples the first by the second and returns this product as a number.

[number] div [number] [number]

The div function takes two numeric arguments and divides the first by the second and returns this quotient as a number.

[number] mod [number] [number]

The mod function takes two numeric arguments and returns the remainder of dividing the first by the second.

[object] foreach [string] [collection] [f1]

The foreach function takes three arguments. The first is a string argument, which is known as the index label. The second is a collection of objects. The third is a function argument. When evaluated the foreach function iterates over the collection and binds the current object in the collection into the context using the string argument a label. It then evaluates [f1]. On the final iteration, the result of [f1] is returned. For example:

example of the foreach function

set “total” 0

foreach “num” (1 2 3 4 5 6 7 8 9)

{

assign “total” add get “total” get “num”

}

println “total”

This script prints the sum of all the numbers in the collection.

[object] find [string] [collection] [f1]

The find function takes three arguments: a string that is used as an index label, a collection and a function f1. It iterates over the collection binds the object in the collection into the context under the given string label and then evaluates f1. If the result of f1 is the boolean value true then the find function returns the current object in the collection. For example:

```
# example of the find function
set “result”
find “x” ( 1 2 3 4 5 )
{
eq get “x” 4
}
println get “result”
```

This script prints “4”.

[object] find_last [string] [collection] [f1]

The find_last function is identical to the find function except it iterates over the collection backwards thereby returning the last object in the collection that fulfills the criteria f1.

[collection] filter [string] [collection] [f1]

The filter function takes three arguments: a string that's used as the index label a collection and a function f1. It then iterates over the collection and binds the current object in the collection under the label string and then evaluates f1. If f1 returns true then the current object is added to the result collection. The filter function then returns the 'result' collection. For example:

```
# an example of the filter function
set “odds”
filter “x” ( 1 2 3 4 5 6 )
{
eq mod get “x” 2 1
}
println get “odds”
```

This script prints “(1 3 5)”. Here, only this numbers that are divisible by two are stored in the result of the filter function.

[collection] filter_out [string] [collection] [f1]

The filter_out function is equivalent to the filter function except it only adds those objects in the collection for which the function f1 evaluates to false (indicating they don't meet some criteria).

[collection] transform [string] [collection] [f1]

The transform function takes three arguments: a string, a collection, and a function f1. It then iterates over the collection each time binding the current object under the label defined by the first argument. It then evaluates the function f1 and the result of this function is added to the result collection. It then returns the result collection. For example:

```
# an example of the transform function
set “doubles”
transform “x” ( 1 2 3 4 5 )
{
mul get “x” 2
}
println get “doubles”
```

This script will print out “(2 4 6 8 10)”.

[number] count [string] [collection] [f1]

The count function is logically equivalent to the filter function except it merely returns the number of items in the collection that fulfill some criteria defined by the function f1.

[object] replace [number] [collection] [object]

The replace function takes three arguments: a number, a collection and an object. It then replaces the object at the specified position (as defined by the first argument) in the collection passed as the second argument with the third argument. It returns whatever object was previously at that position in the collection.

[number] size [collection]

The size function returns the number of objects in a collection.

[object] index [number] [collection]

The index function returns the object at a given position within the collection.

[object] rem [number] [collection]

The rem function removes the object at the specified position (as defined by the first argument, a number) in the collection and returns the object previously in the collection.

[collection] subrange [number] [number] [collection]

The subrange function takes two numbers and a collection. The first number defines the bottom of the range and the second number defines the top of the range. It then retrieves all objects in the range from the given collection and returns the resultant collection.

[collection] union [collection] [collection]

The union function takes two collections and returns the result of inserting every object in the second collection into the first.

[collection] overlap [collection] [collection]

The overlap function takes two collections and returns a collection containing every object in both collections.

[collection] diff [collection] [collection]

The diff function takes two collections and returns a collection containing every object in the first that's not also in the second.

[object] while [f1] [f2]

The while function implements the traditional while-loop paradigm. It takes two arguments, both functions. The first function, when evaluated, should yield a Boolean value. The second function should may yield any object; its return value will be the return value of the while function. (If a while function never executes it simply returns null). Note that the two arguments of the while functions should almost always be grouped in blocks (except for the degenerate cases of 'while true' or 'while false'). Here's an example of using the while function:

```
# an example of using the while function
set "x" 0.0

# note that you gotta put the conditional inside a block!
while { gt get "x" 0.0 }
{
println concat "x is: " get "x"
assign "x" sub get "x" 1
}
```

3.6 Object Oriented Functions\

[function] oo.function [block]

The oo.function function constructs a function object that is defined to be a reusable unit of code that can be executed an arbitrary number of times.

[object] oo.call [function]

Call a function. The oo.call method calls some function which forces the function to be evaluated in the caller's context. Here is an example of using it:

```
# an example of using the oo.call function
set “doGreet”
oo.function
{
println get “greeting”
}

set “greeting” “Hello World!”

oo.call get “doGreet”
```

This script will print 'Hello World'. What's important to realize is that the doGreet function finds the “greeting” variable because it is defined in the script context.

[object] oo.object (inherits [object]) ... oo.endobject

The oo.object function constructs a new object. When this function is evaluated this causes every function up until the oo.endobject function to be evaluated in a special persistent scope known as the “object scope.” Usually, when doing object oriented you bind certain variables—known as properies—that define an object's state and certain functions—known as methods—that define its behavior. When one object inherits another object it inherits all properties and functions bound into the object's context. This function returns the newly created object type.

[object] oo.send [object] [string]

The oo.send function can be used to send methods to an object. This is used to cause the object to do something by executing some function defined in the object's scope. This function returns whatever the object's function returns. Here is an example of using oo.send:

```
# an example of using oo.send
set “greeter”
oo.object
set “greeting” “Hello World!”
set “doGreet”
oo.function
{
println get “greeting”
}
oo.endobject

oo.send get “greeter” “doGreet”
```

This script will also print out “Hello World”.

[object] oo.getp [object] [string]

The oo.getp function retrieves some property (or even some function) that has been

bound into some object instance. It takes an object argument and a string indicating the name of the variable to retrieve—it then returns this variable. For example, given the script defined above then:

example of using oo.getp

```
println oo.getp get “greeter” “greeting”
```

Would also print out 'Hello World.

[object] oo.setp [object] [string] [object]

The oo.setp function can be used to add new properties or even functions to an object. It takes three arguments, the object to alter, the name of the property or function to assign, and the object to bind into the object. For example consider the script:

```
# example of using oo.setp
set “greeter”
oo.object
set “sayHello”
oo.function
{
println “Goodbye world!”
}
oo.endobject

# oops! reassign the function
oo.setp get “greeter” “sayHello”
oo.function
{
println “Hello World!”
}

oo.send get “greeter” “sayHello”
```

[object] oo.assignp [object] [string] [object]

The oo.assignp function can be used to change the properties and functions bound in an object. Unlike the oo.setp function the oo.assignp function is guaranteed to never create a new variable, only reassign an existing variable. It takes the object to alter, the name of the variable, and the new value of the variable. It returns this new variable after assigning the object.

[object] oo.new [object]

The oo.new function can be used to create new objects from existing objects. The function takes a single function that must be an existing object and returns a clone which has all the same properties and functions as the original object.

3.7 I/O Functions

[file] io.file [string]

The io.file function takes a string representing a path on the local hard disk and returns a value representing the File identified by the path.

[dir] io.dir [string]

The io.dir function takes a string representing a path on the local hard disk and returns a directory object identified by that path.

[file] io.append [stream] [file]

The io.append function takes a stream of data and a file and appends the stream of data onto the file and returns the file that it appended data too.

[object] io.copy [stream] [stream]

The io.copy function takes two streams and copies the first stream into the second stream. It returns the second stream of data (really it returns the stream's 'placeholder' object).

[boolean] io.del [file|directory]

The io.del function deletes a directory or a file and returns the boolean value which is true if the path was deleted else false.

[boolean] io.create [file]

The io.create function takes a file or directory object and actually attempts to create it on the disk (if it doesn't already exist). It then returns true if the creation was successful else false.

[boolean] io.mkdir [dir]

The io.mkdir function takes a directory object and actually attempts to create it on the hard disk (if it doesn't already exist). It returns the boolean value true if the directory was created, else false.

[file|directory] io.fcopy [file|directory] [file|directory]

The io.copy function takes a file or a directory, known as the source, and another file or directory known as the destination. It then performs a copying function: it either copies a file (source) into a directory (destination), a directory (source) into another directory (destination), or a file (source) into another file (destination). It then returns the destination object.

[file] io.mkchild [file] [directory]

The io.mkchild function takes a file, which can be located anywhere, and and a directory. It then returns a file that has the same name as the first argument but is located in the directory defined by the second argument.

3.8 XML Functions

[xml] xml.doc [stream]

The xml.doc function takes a stream of data and returns an XML document defined by parsing the stream of data. For example:

```
# an example of the xml.doc function
set “xml” xml.doc
“<?xml version=”1.0”?><root><elem>An example</elem>< root>”
println get “xml”
```

The previous example will print out the document defined in the string passed to the xml.doc function after first parsing and normalizing it as XML.

[xslt] xml.xslt [stream]

The xml.xslt function takes a stream of data and compiles it down into an XSLT transformation. This compiled transformation can then be used to transform other XML documents. It returns the transformation.

[xml] xml.xtransform [xslt] [xml]

The xml.xtransform function is used for transforming an XML document with a transformation when you know the result will also be valid XML. It takes two arguments: a transformation and an XML document against which the transformation should be applied. It returns an XML document that is the result of the transformation.

[boolean] xml.transform [xslt] [xml] [stream]

The xml.transform function applies a transformation to a given XML document and writes the transformation result into its third argument that must be some sort of output stream. It then returns true if the transformation was successful else false is returned.

[xpath] xml.xpath [string]

The xml.xpath function accepts a string as its first argument and returns a compiled XPath expression that can be used to select against XML documents.

[collection] xml.select [xpath] [node]

The xml.select function takes two arguments. The first is an XPath expression indicating some selection criteria. The second is an XML document (though it may in fact be any node) against which the XPath expression is applied. The function returns a collection of nodes in the document that match the XPath selection criteria.

[object] xml.valueof [xpath] [node]

The xml.valueof function takes two arguments, first an XPath expression that will be applied to the second an XML document (though it may be any type of XML node) and it will apply the XPath expression against the document and return the result of the valueof operation. This function allows the user to retrieve arbitrary strings, numbers, or boolean variables from an XML document.

3.9 HTTP Functions

[collection] http.nvcoll [collection]

The http.nvcoll constructs a collection of NV pairs (Name-Value pairs) that can be passed in HTTP queries, forms, or as HTTP headers. This function takes a single argument, a collection of strings. These strings are intepreted as pairs: each string n is

interpreted as a 'name' and each following string n+1 is interpreted as a value of that name. It returns a collection of NV pairs constructed from these strings.

[credentials] http.creds [“basic”|”digest”|] realm user pass

The http.creds function constructs a credentials object that can be used to authenticate HTTP requests using either HTTP basic or digest authentication. It takes four strings as arguments: the first must be either “basic” or “digest” indicating the type of authorization, the second is the URL realm of resources being requested, and the third and fourth are a username and password respectively. It returns a credentials object.

[connection] http.conn [string] [number]

The http.conn constructs a HTTP connection object that can be used to connect to web servers. It takes two arguments. The first is a string indicating the host of the web server (either by domain name or IP). The second is a number indicating the port the HTTPd process is listening on. It returns a connection object.

[response] http.get [string] [connection] [collection] [collection]

The http.get function executes a HTTP GET request. The first argument is a string indicating the path to the resource to be requested. (This usually must start with a '/' eg '/index.html'). The second argument is a connection object representing a web server to be requested. The third argument is a collection of NV pairs that will be passed on the query string. The fourth argument is a collection of NV pairs that will be passed as request headers. It returns a http response object. For example:

```
# example of the http.get function
set "yahoo" http.conn "www.yahoo.com" 80
set "response" http.get "/index.html" get "yahoo" null null
println get "response"
```

This script will print the current contents of the front page of yahoo.com.

[response] http.put [string] [connection] [stream] [collection]

The http.put function executes a HTTP PUT request. The first argument is a string indicating the path of the resource to be requested. The second argument is a HTTP connection to the web server to be requested. The third argument is a stream of data that will be read as the body of the request. The fourth argument is a collection of NV pairs that will be passed as request headers. It returns a response object.

[response] http.delete [string] [connection] [collection]

The http.delete function executes a HTTP DELETE request. The first argument is a string indicating the path of the resource to be deleted. The second argument is a connection to a web server. The third argument is a collection of NV pairs that will be passed as the request headers.

[response] http.post [string] [connection] [stream] [collection]

The http.post function executes a HTTP POST request. The first argument is a string indicating the path of the resource to be posted to. The second argument is a connection to a web server. The third argument is a stream of data that will form the contents of the request. The fourth argument is a collection of NV pairs that will be passed as request headers. It returns a response object.

[response] http.trace [string] [connection] [collection]

The http.trace function executes a HTTP TRACE request. The first argument is a string indicating the path of the resource to be traced. The second argument is a connection to a web server. The third argument is a collection of NV pairs that will be passed as request headers. It returns a response object.

[response] http.head [string] [connection] [collection] [collection]

The http.head function executes a HTTP HEAD request. The first argument is a string indicating the path of the resource. The second argument is a connection object. The third argument is a collection of NV pairs that will be passed as query parameters. The fourth argument is a collection of NV pairs that will be passed as request headers. It returns a response object.

[response] http.options [string] [connection] [stream] [collection]

The http.options function executes a HTTP OPTIONS request. The first argument is a path to the resource. The second argument is a connection object. The third argument is a stream of data that will be sent as the body of the request. The fourth argument is a collection of NV pairs to be sent as request headers. It returns a response object.

[string] http.header [string] [response]

The http.header function retrieves a header from a response. The first argument is a string indicating the name of the header. The second argument is the response itself. It returns a string indicating the value of the header or null.

[date] http.dateheader [string] [response]

The http.dateheader function retrieves a header that's a date value from a response. It returns a date value after parsing the header value identified by the first argument according to the rules of RFC 1023.

[number] http.status [response]

The http.status function returns the status code of the given response.

[stream] http.body [response]

The http.body function returns a stream of data that's the contents of a response.

[4] PROJECT PLAN

4.1 Task Breakdown

Buko- design, project structure and parser implementation, and documentation
Rafi- core function implementation, testing and documentation
Artem- xml and http function implementation, testing, and documentation
Isaac- io function implementation, testing and documentation

Many tasks were shared throughout the development of the project as all members participated in testing and documentation.

4.2 Project Timeline and Log

2-18-03	Whitepaper, first group meeting
3-25-03	Start language reference manual, begin implementation
3-26-03	Language reference manual finished, Isaak joins group
4-15-03	Parser using stack implemented
4-27-03	Functions in writing process
5-02-03	"Hello World"
5-03-03	Continue working on functions till deadline
5-13-03	Class presentation
5-17-03	Final project

4.3 Software Development

Absolutely everything from the Parser to the Lexer to the functions to the testing were made using JDK 1.4 on Windows XP, 2000, or even Linux, depending on whose operating system it is being worked on (Artem and Rafi are linux geeks).

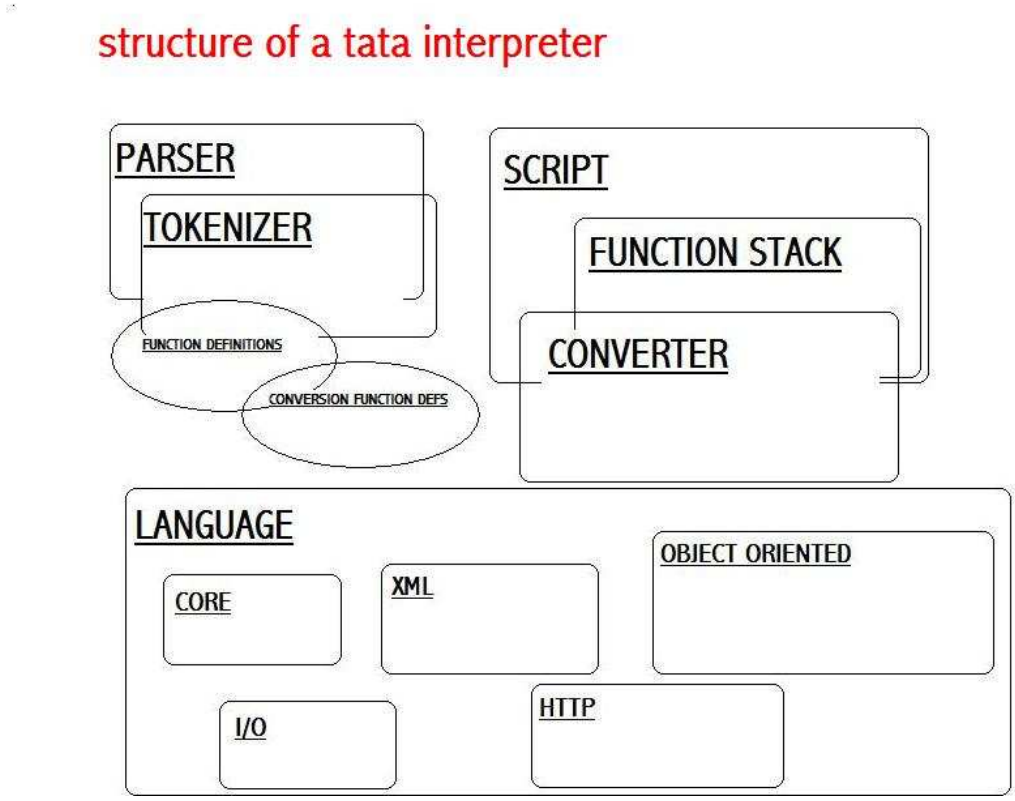
[5] ARCHITECTURAL DESIGN

5.1 Introduction

This document provides a high-level overview of the architecture of the tata interpreter. This document fulfills requirement #5 “Architectural Design” of the language submission.

5.2 Block Diagram

The following diagram provides a visual diagram of the tata interpreter.



5.3 Architecture Overview

One of our goals in developing the tata scripting language was to challenge the traditional model of language construction. We wanted to both provide an alternative and extend the common view that a language is a set of keywords, some additional syntax, and some semantics and that a program is essentially of combination keywords and API calls of these three that never changes. Instead we feel that there is no meaningful distinction between a language and libraries developed in a language. Further we feel that a lot of the extra tokens programmers write in traditional programs, things like parentheses and semicolons and the like are often unnecessary—they are simply extra complexity that is forced upon the programmer so that the language compiler and interpreter can be made simpler. So with tata we

wanted to describe a language that was as simple as possible but not necessarily simple in the traditional sense but rather simple in the sense that the language should provide a simple, powerful foundation that the programmer can easily extend and employ once he uses it. You might call this ‘elegant’ but we adopted the less pretentious acronym DMWNQSM as our motto: Do More With Not Quite So Much.

5.4 Architecture Analysis

The architectural goals of tata had great influence on the design of the interpreter. Because tata programs had to be extremely dynamic and because the lines between language, API, and syntax were intentionally blurred or destroyed (e.g. there is no formal notion of an identifier, instead the set function binds objects to strings which can themselves be dynamically constructed) we didn’t use traditional language construction tools like ANTLR.

The tata parser works by accepting some input data which is understood to be a sequence of characters encoded in some given character set. This sequence is decoded and recoded as a sequence of Unicode characters. Next the parser uses a tokenizer to break the characters up into tokens. A token isn’t defined by simply word boundaries (see the LRM for more info) rather a few simply logical rules (e.g. ignoring comments, whitespace handling) are used to turn the stream of characters into a stream of tokens. Next the tata parser examines each token and attempts to resolve it into a function. In order to resolve the token the parser tries to match it, as a pattern, against a special file known as the function definition base. This is just a plain text file that matches token patterns against function definitions and in fact it’s trivial to add new functions to the language by editing this file. If the parser can resolve the function it pushes the function into a data structure known as a function stack. If the parser can’t resolve the token then (if extension mode is enabled) it simply pushes the token sequence onto the stack. (This strange behavior exists because the designers wanted to allow for the possibility of using a special tata IDE to create ‘mixed documents’—that is tata programs that are mixed with other content like HTML, other languages like Java or even complex objects like Microsoft OLE components (eg an Excel spreadsheet). This functionality was not implemented but the architecture allows for it). Because tata is a weakly typed language there must be a mechanism to perform arbitrary conversions between different object types. Instead of hard-coding such conversions into the interpreter conversions are defined as special implicit functions. These functions are defined in a separate file, the conversion definition base. The parser also parses this file to construct a component known as the Converter.

Once a function stack has been constructed the script function is executed. This function is a bootstrap function that simply pops each function off the stack and evaluates it in the global context.

Functions are defined in modules that usually perform some related group of work. The standard modules are the Core module which contains 40+ functions for working with basic elements like strings and numbers, the I/O module for working with files and directories, the XML module for working with XML technologies, and the HTTP module for working with web technologies. Each module is aware only of the basic interfaces defined by the tata interpreter such as the Function interface, the Context interface that enforces scoping constraints, and the Script interface that defines a

program. Because tata is a weakly typed language it must also be able to perform arbitrary conversions. These conversions are performed by a script-global conversion engine known as the 'Converter' (somewhat confusingly, different tata scripts can define different conversions) that functions use in order to convert their arguments into the expected types. Beyond this modules are allowed to do whatever they want—they can introduce new language constructs or even modify the behavior of the interpreter. As long as they respect the basic design model of the interpreter—that is a set of functions to be evaluated on a stack—modules are allowed total freedom. Very few semantics are imposed upon module developers allowing for the possibility of using tata as a 'language engine' to construct new mini-languages (see the whitepaper for more about this, for example the http.* functions are in many ways a way of controlling a web user agent). For example, it's conceivable that other languages like *awk* which have relatively simple computation models (e.g. pattern:work matching) could be simulated in tata in a "true-to-form" manner so that the there'd be little difference between the awk program and the tata program (besides minor syntax issues). More research is being done on this.

5.5 Work Distribution

The design of the language was originally Buko's and he made many of the early, crucial design decisions though later he had much input from Rafi and Artem (Isaac didn't join the group until after the LRM had been submitted). In terms of implementation, the parser was implemented by Buko. The core package was implemented by Rafi with help from Buko. The XML and HTTP modules were implemented by Artem and later, Isaac worked from this code to produce the I/O module. The object oriented module was implemented by Buko.

[6] TEST PLAN

6.1 Overview

The tata interpreter is a complex software program consisting of over 40 language functions, 210 source files, and several files of documentation and specification. Implementing such a program "from scratch" by first constructing a high level design (affectionately known as BDUF "Big Design Up Front") would be fantastically difficult and, in particular, would require a high degree of coordination among the developers—something the tata developers, who at best could meet once or twice a week, did not have. Thus in the development of the interpreter the traditional programming model of Design-Implement-Test was abandoned. Instead, the principles of new methodologies such as Extreme Programming[1] (XP) and Test Driven Development[2] (TDD) were embraced.

In both XP and TDD testing, particularly unit testing, is integral to the development process and so testing was integral to the development of the tata interpreter. Generally, development proceeded by breaking down the program components into the smallest possible units and then incrementally creating and testing each unit. For example, the implementation of the parser was first broken down into implementation of a tokenizer and an analyzer. Then each of these would actually further be broken down into smaller units. Each of these units was then subjected to unit testing (in Java) that explicitly verified that the code worked and also tested boundary conditions and the like. The implementation of the actual language proceeded in a similar fashion. First a simple string function was implemented and tested. Next a print function, and so on and so forth. There was no formal integration phase—instead each new component of functionality was incrementally added to a common code-base. The benefits of this methodology were many.

6.2 Benefits

· *Extremely Low Bug Count*

Generally, we experienced an extremely low rate of bugs. Once development was complete we discovered only 4 major bugs in over 230+ source files.

· *Self-Documenting*

Because each function of the language got its own set of test files it was always very easy to verify and check the behavior of a function. Writing documents like the Language Reference Manual and the Tutorial became simply a matter of documenting the assumptions made by our tests.

· *Controlled Complexity*

Initially, we attempted to construct a high level design of the language. The course structure encouraged this by first requiring a whitepaper and an LRM before development was complete. But by starting with the code and "going backwards" we were able to ensure that most everything we were specifying was actually doable. Sometimes this "code before design" strategy failed—for example,

[1] http://www.extremeprogramming.org
[2] http://www.objectmentor.com/writeUps/TestDrivenDevelopment

the weak-typing/auto-conversion framework had to be rewritten from scratch twice and it is still a source of some bugs in the interpreter.

· *Refactoring Confidence*

It was (and still is) very easy to add new functionality to the interpreter. This is because the behavior of the interpreter was always completely specified by a series of tests therefore new functionality could be introduced and by simply rerunning test suits we could be confident that old functionality had not been broken. This allowed us to make drastic changes to the code-base up to the very last minute and still be confident we had a working product.

· *Modularization*

Because each component needed to be both heavily tested and incrementally constructed we were forced to ruthlessly modularize the architecture of the language.

· Three Levels of Testing

Generally, there were three levels of testing that we used.

· *Code Unit Tests*

Unit tests were developed to guarantee that code adhered to its interface contract and to test explicit assumptions made by the developer of the code. Unit tests were constructed for most every ‘aspect’ of the code-base where an aspect is usually a method though might also be a class. Our unit tests were developed using the popular Java library JUnit[3]. For example, here is a code unit test:

```
public class TokenStackTest extends TataTestCase

{
public void testTokenize() throws Exception
{
String s = "this string has five tokens";
TokenStack ts = new TokenStack(s);
assertEquals(5, ts.size());
}
public void testState() throws Exception
{
String s = "welcome to the terror dome";
TokenStack ts = new TokenStack(s);
assertEquals(ts.peek(), "welcome");
ts.pop();
ts.pop();
String the = ts.pop();
assertEquals("the", the);
ts.push("a");
assertEquals("a", ts.pop());
assertEquals("terror", ts.pop());
assertEquals("dome", ts.pop());
}
}
```

[3] http://www.junit.org

This specific test verifies the functionality of the very simple tokenizing code. This test verifies both that the code works and also tests boundary conditions.

· *Language Unit Tests*

The next level of testing was testing specific functions of the language. This was done by writing several (over 60) very small tests that verified a single function or aspect of the interpreter. Here are some example language unit tests:

```
# from div-test.tata
# test of the division function

println div 49 7
println div 48 7
```

Here is another example language level unit test that tests the foreach function:

```
# test of the foreach function

foreach "str" ( "Hello" "World" "Who's" "A" "Bad" "Man?" )
{
        print get "str"
        print "_"
}

set "total" 0
foreach "num" ( 1 2 3 4 5 6 7 8 9 )
{
        set "total" get "total"
        print get "num"
}
println "this is the end"
println get "total"

# now check this, we loop over an empty collection!
# BEWARE THE EL GREGOR
foreach "num" ( )
        { println "if you see this, you're already dead" }
```

6.3 Blackbox Testing

Most of the testing was done using whitebox unit tests. These tests were written by the developer to test explicit functionality and assertions in the code. But it’s impossible to write unit tests for all possible cases so another level of testing was implemented. Another level of testing was needed to verify that the language actually adhered to the constraints and functionality defined by the formal Language Reference. This blackbox testing was often written by developers for code they didn’t work on and so couldn’t make any assumptions about. Also these blackbox tests had to be very thorough. Here is an example blackbox test:

```
# advanced test of the division function
println div 49 7
#println div -49 7 # error neg numbers, should use “sub 0 49”
```

```
#println div 49 -7
println div 49 0
println div 0 7
println div 1 7
println div 7 7
println div 7 7.1
println div 7.1 7
println div 1.7 1.7
println div 1.7 7
println div 1.7 0
println div 0 1
println div 0 1.1
println div 2.2 1.1
#println div 0 "1" #all commented lines from here cause the interpreter to throw
#println div 0 "a" #an exception. This is expected and wanted.
#println div 0 ( )
#println div 0 ( "1" )
#println div 0 ( 1 )
#println div 0 ( )
println div true 0
println div 0 true
#println div 0 null
#println div null 0
#println div null ( )
#println div null "a"
#println div "a" null
#println div "b" "a"
#println div ( ) ( )
#println div ( null ) ( null )
#println div ( 1 2 3 ) ( 4 5 "6" )
#println div ( 1 2 3 ) ( 1 2 3 )
#println div ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println div ( 1 2 3 ) ( 4.2 -5.3 "6" 3 2 )
```

It was impossible to automate these blackbox tests simply because when we wrote error assertion tests (that is tests designed to fail by causing an error in the interpreter) an exception was generated and the Java Virtual Machine would fail fast and need to be restarted. Instead a few simple bash scripts were thrown together to, for example, run all tata scripts in a directory and print their output to screen. Then the output of each script had to be manually verified. These files containing the results of several tests were known in the group as “master files” and proved to be very helpful. All the errors were listed in a master error file for all to see and were addressed one by one. This enabled the group to continue testing for advanced errors concurrently and not wait for each error to be handled serially. The master file also provided all the team members knowledge about errors in other tests that might affect different correlated tests. When a lazy developer refused to or was slow to address errors in the master file then peer pressure could be applied or, failing that, insults about the female members of his family.

6.4 Testing Task Breakdown

Most of the code level and language level unit testing was done by the

person who wrote the code. For example, parser testing was done by Buko while the core functions testing was done by Rafi. The IO testing was done by Isaac and the XML/HTTP testing by Artem. For the blackbox testing this was “switched” so that Rafi did parser testing and Isaac and Artem wrote tests for the other’s code.

[7] LESSONS LEARNED

7.1 Lessons by group members

Buko – No matter how hard you try things will end up differently than first anticipated, originally we had tried to make the language functional using SOAP but we had to scrap that idea.

Isaac – Testing has to be very thorough many factors may affect the tests especially those that aren't considered

Rafi – A functional language is definitely the way to go it was pretty simple to implement and came out great

Artem – In development, modularity is key to succeeding in a limited time frame. Differences in IO on varios operating systems platforms can cause errors, even in a platform-independent language such as java. Many languages such as perl have good reference on implemeting XML and HTTP related functionality. The XML related domain is vast but easily broken down to subsections such as parsing, manipulation, transformation, and query.

7.2 Advice for future language developers

- Start early, the earlier you start the better off you will be.

- Make sure you have fun when you make your language. Be creative with your syntax and function names.

- Watch out for relative paths, lack of internet connections, and OS differences.

APPENDIX A

General Source Code (with location paths)

- Rafi is responsible for majority of code under the “core” subdirectory.
- Artem is responsible for majority of code under the “http” and “xml” subdirectories.
- Isaac is responsible for majority of code under the “io” subdirectory.
- Buko is responsible for oo and parse subdirectories, glue code, and anything not falling into one of the subdirectories listed above.

A full distribution of Tata with source may be found at:
http://www.artemz1.com/other/tata/

```
[Contents of: ./src/conversion.properties]

# File defines all the possible conversions
# Last Modified: 5/5/03

# core conversions

conversion.double2int=net.concedere.tata.convert.core.DoubleIntegerC
onversion
double2int.from=java.lang.Double
double2int.to=java.lang.Integer

conversion.int2double=net.concedere.tata.convert.core.IntegerDoubleC
onversion
int2double.from=java.lang.Integer
int2double.to=java.lang.Double

conversion.int2string=net.concedere.tata.convert.core.IntegerStringC
onversion
int2string.from=java.lang.Integer
int2string.to=java.lang.String

conversion.double2string=net.concedere.tata.convert.core.DoubleStrin
gConversion
double2string.from=java.lang.Double
double2string.to=java.lang.String

conversion.string2int=net.concedere.tata.convert.core.StringIntegerC
onversion
string2int.from=java.lang.String
string2int.to=java.lang.Integer

conversion.boolean2string=net.concedere.tata.convert.core.BooleanStr
ingConversion
boolean2string.from=java.lang.Boolean
boolean2string.to=java.lang.String

conversion.url2string=net.concedere.tata.convert.core.UrlStringConve
rsion
url2string.from=java.net.URL
url2string.to=java.lang.String

conversion.string2url=net.concedere.tata.convert.core.StringUrlConve
rsion
string2url.from=java.lang.String
string2url.to=java.net.URL

conversion.coll2string=net.concedere.tata.convert.core.CollectionStr
ingConversion
coll2string.from=java.util.Collection
coll2string.to=java.lang.String
```

```
conversion.int2bool=net.concedere.tata.convert.core.NumberBooleanCon
version
int2bool.from=java.lang.Integer
int2bool.to=java.lang.Boolean

conversion.double2bool=net.concedere.tata.convert.core.NumberBoolean
Conversion
double2bool.from=java.lang.Double
double2bool.to=java.lang.Boolean

conversion.bool2int=net.concedere.tata.convert.core.BooleanIntegerCo
nversion
bool2int.from=java.lang.Boolean
bool2int.to=java.lang.Integer

conversion.bool2double=net.concedere.tata.convert.core.BooleanDouble
Conversion
bool2double.from=java.lang.Boolean
bool2double.to=java.lang.Double

conversion.coll2int=net.concedere.tata.convert.core.CollectionIntege
rConversion
coll2int.from=java.util.Collection
coll2int.to=java.lang.Integer

# io conversions

conversion.file2string=net.concedere.tata.convert.io.FileStringConve
rsion
file2string.from=java.io.File
file2string.to=java.lang.String

conversion.string2file=net.concedere.tata.convert.io.StringFileConve
rsion
string2file.from=java.lang.String
string2file.to=java.io.File

conversion.file2inputstream=net.concedere.tata.convert.io.FileInputS
treamConversion
file2inputstream.from=java.io.File
file2inputstream.to=java.io.InputStream

conversion.file2outputstream=net.concedere.tata.convert.io.FileOutpu
tStreamConversion
file2outputstream.from=java.io.File
file2outputstream.to=java.io.OutputStream

conversion.string2inputstream=net.concedere.tata.convert.io.StringIn
putStreamConversion
string2inputstream.from=java.lang.String
string2inputstream.to=java.io.InputStream

conversion.url2inputstream=net.concedere.tata.convert.io.UrlInputStr
eamConversion
url2inputstream.from=java.net.URL
url2inputstream.to=java.io.InputStream


# xml conversions

conversion.doc2string=net.concedere.tata.convert.xml.DocumentStringC
onversion
doc2string.from=org.dom4j.Document
doc2string.to=java.lang.String

conversion.string2doc=net.concedere.tata.convert.xml.StringDocumentC
onversion
string2doc.from=java.lang.String
string2doc.to=org.dom4j.Document
```

```
conversion.string2xslt=net.concedere.tata.convert.xml.StringXsltConv
ersion
string2xslt.from=java.lang.String
string2xslt.to=javax.xml.transform.Transformer

conversion.doc2inputstream=net.concedere.tata.convert.xml.DocumentIn
putStreamConversion
doc2inputstream.from=org.dom4j.Document
doc2inputstream.to=java.io.InputStream

conversion.inputstream2doc=net.concedere.tata.convert.xml.InputStrea
mDocumentConversion
inputstream2doc.from=java.io.InputStream
inputstream2doc.to=org.dom4j.Document

conversion.xpath2string=net.concedere.tata.convert.xml.XpathStringCo
nversion
xpath2string.from=org.dom4j.XPath
xpath2string.to=java.lang.String

conversion.node2string=net.concedere.tata.convert.xml.NodeStringConv
ersion
node2string.from=org.dom4j.Node
node2string.to=java.lang.String

# http conversions

conversion.nvpair2string=net.concedere.tata.convert.http.NvPairStrin
gConversion
nvpair2string.from=HTTPClient.NVPair
nvpair2string.to=java.lang.String

conversion.credentials2string=net.concedere.tata.convert.http.HttpCr
edentialsStringConversion
credentials2string.from=net.concedere.tata.http.HttpCredentials
credentials2string.to=java.lang.String

conversion.httpconnection2string=net.concedere.tata.convert.http.Htt
pConnectionStringConversion
httpconnection2string.from=HTTPClient.HTTPConnection
httpconnection2string.to=java.lang.String

conversion.httpconnection2url=net.concedere.tata.convert.http.HttpCo
nnectionUrlConversion
httpconnection2url.from=HTTPClient.HTTPConnection
httpconnection2url.to=java.net.URL

conversion.url2httpconnection=net.concedere.tata.convert.http.UrlHtt
pConnectionConversion
url2httpconnection.from=java.net.URL
url2httpconnection.to=HTTPClient.HTTPConnection

conversion.response2string=net.concedere.tata.convert.http.HttpRespo
nseStringConversion
response2string.from=net.concedere.tata.http.HttpResponse
response2string.to=java.lang.String

 [End of: ./src/conversion.properties]


[Contents of: ./src/function-defs.properties]


# defines the token->function definitions

# Core Functions

println=net.concedere.tata.core.PrintlnFunction
print=net.concedere.tata.core.PrintFunction
```

```
true=net.concedere.tata.core.TrueValueFunction
false=net.concedere.tata.core.FalseValueFunction
null=net.concedere.tata.core.NullValueFunction

set=net.concedere.tata.core.SetFunction
>>=net.concedere.tata.core.SetFunction
get=net.concedere.tata.core.GetFunction
<<=net.concedere.tata.core.GetFunction
unset=net.concedere.tata.core.UnsetFunction
assign=net.concedere.tata.core.AssignFunction

url=net.concedere.tata.core.UrlFunction

(=net.concedere.tata.core.BeginListFunction
)=net.concedere.tata.core.EndListFunction
{=net.concedere.tata.core.BeginBlockFunction
}=net.concedere.tata.core.EndBlockFunction

&=net.concedere.tata.core.ConcatenationFunction
concat=net.concedere.tata.core.ConcatenationFunction

# Boolean Logic

eq=net.concedere.tata.core.EqualsFunction
not=net.concedere.tata.core.NotFunction
gt=net.concedere.tata.core.GreaterThanFunction
gte=net.concedere.tata.core.GreaterThanEqualsFunction
lt=net.concedere.tata.core.LessThanFunction
lte=net.concedere.tata.core.LessThanEqualsFunction

# Conditional Logic

if=net.concedere.tata.core.IfFunction
else=net.concedere.tata.core.ElseFunction
endif=net.concedere.tata.core.EndIfFunction
when=net.concedere.tata.core.WhenFunction
unless=net.concedere.tata.core.UnlessFunction
and=net.concedere.tata.core.AndFunction
or=net.concedere.tata.core.OrFunction

# Math Functions

add=net.concedere.tata.core.AddFunction
sub=net.concedere.tata.core.SubtractFunction
mul=net.concedere.tata.core.MultiplyFunction
div=net.concedere.tata.core.DivideFunction
mod=net.concedere.tata.core.ModulusFunction

# Collection Manip Functions

foreach=net.concedere.tata.core.ForEachFunction
find=net.concedere.tata.core.FindFunction
find_last=net.concedere.tata.core.FindLastFunction
filter=net.concedere.tata.core.FilterFunction
filter_out=net.concedere.tata.core.FilterOutFunction
transform=net.concedere.tata.core.TransformFunction
count=net.concedere.tata.core.CountFunction
replace=net.concedere.tata.core.ReplaceFunction
size=net.concedere.tata.core.SizeFunction
index=net.concedere.tata.core.IndexFunction
rem=net.concedere.tata.core.RemoveFunction
subrange=net.concedere.tata.core.SubRangeFunction
union=net.concedere.tata.core.UnionFunction
overlap=net.concedere.tata.core.OverlapFunction
diff=net.concedere.tata.core.DiffFunction

# pure loops
while=net.concedere.tata.core.WhileFunction

# Object Oriented Functions
```

```
oo.object=net.concedere.tata.oo.BeginObjectFunction
oo.endobject=net.concedere.tata.oo.EndObjectFunction
oo.function=net.concedere.tata.oo.CreateFunctionFunction
oo.send=net.concedere.tata.oo.SendFunction
oo.call=net.concedere.tata.oo.CallFunction
oo.getp=net.concedere.tata.oo.GetPropertyFunction
oo.setp=net.concedere.tata.oo.SetPropertyFunction
oo.assignp=net.concedere.tata.oo.AssignPropertyFunction
oo.new=net.concedere.tata.oo.NewFunction
oo.inherits=net.concedere.tata.oo.InheritsFunction


# I/O functions
,
io.file=net.concedere.tata.io.FileFunction
io.dir=net.concedere.tata.io.DirectoryFunction
io.append=net.concedere.tata.io.AppendFunction
io.copy=net.concedere.tata.io.CopyFunction
io.del=net.concedere.tata.io.DeleteFunction
io.create=net.concedere.tata.io.CreateFunction
io.mkdir=net.concedere.tata.io.MakeDirFunction
io.fcopy=net.concedere.tata.io.FileCopyFunction
io.mkchild=net.concedere.tata.io.MakeChildFunction

# XML functions

xml.doc=net.concedere.tata.xml.DocumentFunction
xml.xslt=net.concedere.tata.xml.XsltFunction
xml.xtransform=net.concedere.tata.xml.XmlTransformFunction
xml.transform=net.concedere.tata.xml.TransformFunction
xml.xpath=net.concedere.tata.xml.XpathFunction
xml.select=net.concedere.tata.xml.SelectNodesFunction
xml.valueof=net.concedere.tata.xml.ValueOfFunction

# HTTP functions

http.nvcoll=net.concedere.tata.http.NvCollectionFunction
http.creds=net.concedere.tata.http.CredentialsFunction
http.conn=net.concedere.tata.http.ConnectionFunction
http.get=net.concedere.tata.http.GetFunction
http.put=net.concedere.tata.http.PutFunction
http.delete=net.concedere.tata.http.DeleteFunction
http.post=net.concedere.tata.http.PostFunction
http.trace=net.concedere.tata.http.TraceFunction
http.head=net.concedere.tata.http.HeadFunction
http.options=net.concedere.tata.http.OptionsFunction
http.header=net.concedere.tata.http.HeaderFunction
http.dateheader=net.concedere.tata.http.DateHeaderFunction
http.status=net.concedere.tata.http.StatusCodeFunction
http.body=net.concedere.tata.http.HttpBodyFunction



[End of: ./src/function-defs.properties]


[Contents of: ./src/net/concedere/tata/AbstractFunction.java]


package net.concedere.tata;

/**
 * Simple implementation of the Function interface.
 *
 * @version 0.1
 **/
public abstract class AbstractFunction implements Function
{
```

```
}
```

[End of: ./src/net/concedere/tata/AbstractFunction.java]

[Contents of: ./src/net/concedere/tata/BlockFunction.java]

```
package net.concedere.tata;

/**
 * A Function is a BlockFunction if and only if it has a matching
function
 * which represents the end of its logical block.
 *
 * @version 0.1
 **/
public interface BlockFunction
{
          /**
           * Get the class of the end function that ends this block
function.
           *
           * @return class of the end function that ends this block
           */
          public Class getEndFunctionClass();
}
```

[End of: ./src/net/concedere/tata/BlockFunction.java]

[Contents of: ./src/net/concedere/tata/Context.java]

```
package net.concedere.tata;

import java.util.Map;
import java.util.Collections;
import java.util.HashMap;

/**
 * Represents a scope in which a Function can execute.
 *
 * @version 0.1
 **/
public class Context implements Cloneable
{
    private Context parentCtx;
    private Map varMap = new HashMap();

          /**
           * Construct a new Context with the given parent.
           *
           * @param parent parent of the given context
           */
          public Context(Context parent)
          {
                    this.parentCtx = parent;
          }

          /**
           * Get the parent context of this context.
           *
           * @return parent context of the context
           */
          public Context getParent()
          {
                    return parentCtx;
```

```
            }

            /**
             * Bind a variable into the context.
             *
             * @param id  id to store the variable under
             * @param var variable to store in the context
             */
            public void set(String id, Object var)
            {
        if (id == null)
                        {
                                String msg = "id can't be null";
                                throw new
IllegalArgumentException(msg);
                        }

                        varMap.put(id, var);
            }

            /**
             * Get a variable out of the context. If the current
context doesn't have
             * the variable it's parent is searched.
             *
             * @param id id of the variable to retrieve
             * @return object bound to <code>id</code> or
<code>null</code>
             */
    public Object get(String id)
            {
        if (id == null)
                        {
                                String msg = "id can't be null";
                                throw new
IllegalArgumentException(msg);
                        }

        if (varMap.containsKey(id))
                        {
                                return varMap.get(id);
                        }
                        else
                        {
             return (parentCtx == null ? null : parentCtx.get(id));
                        }
            }

            /**
             * Unset a variable from the context allowing it to be
garbage collected.
             *
             * @param id id of the variable to unset
             */
    public void unset(String id)
            {
                        if (id == null)
                        {
                                String msg = "id can't be null";
                                throw new
IllegalArgumentException(msg);
                        }

                        varMap.remove(id);
            }

            /**
             * Determine if a given variable exists and is bound in
the context. The
             * parent context will be searched if possible.
```

```
            *
            * @param id id of the variable to check for
            * @return <code>true</code> iff the context contains the
variable id
            */
    public boolean exists(String id)
            {
        if (id == null)
                    {
                                String msg = "id can't be null";
                                throw new
IllegalArgumentException(msg);
                    }

                    if (varMap.containsKey(id))
                    {
                                return true;
                    }
                    else
                    {
             return (parentCtx == null ? false :
parentCtx.exists(id));
                    }
          }

          /**
           * Find the first context that actually that actually
contains the
           * given variable. This is useful if you want to modify a
variable instead
           * of simply creating a new one.
           *
           * @param id id of the variable to find
           * @return Context that actually directly contains the
variable
           */
          public Context findContaining(String id)
          {
        if (id == null)
                    {
                                String msg = "id can't be null";
                                throw new
IllegalArgumentException(msg);
                    }

        if (varMap.containsKey(id))
                    {
                                return this;
                    }
                    else
                    {
                                return (parentCtx == null ? null :
parentCtx.findContaining(id));
                    }
          }

          public Object clone()
          {
                    Context ctx = new Context(parentCtx);
                    ctx.varMap = new HashMap(varMap);
                    return ctx;
          }

          /**
           * Destroy the context and any data it might contain.
           */
          public void destroy()
          {
                    varMap.clear();
                    varMap = null;
```

```
          }
}
```

[End of: ./src/net/concedere/tata/Context.java]

[Contents of: ./src/net/concedere/tata/convert/Conversion.java]

```
package net.concedere.tata.convert;

import net.concedere.tata.Script;

/**
 * A Conversion defines a conversion between two data types within
the script.
 *
 * @version 0.1
 **/
public interface Conversion
{
          /**
           * Convert the given object to the target class and return
the result.
           *
           * @param o       object to perform the conversion on
           * @param toClass Class to convert the object to
           * @param script  Script the conversion is happening in
           * @return the result of the conversion
           * @throws ConversionException
           *         if something goes wrong
           */
    public Object convert(Object o, Class toClass, Script script)
                                throws ConversionException;
}
```

[End of: ./src/net/concedere/tata/convert/Conversion.java]

[Contents of:
./src/net/concedere/tata/convert/ConversionDescriptor.java]

```
package net.concedere.tata.convert;

/**
 * Describes a possible conversion that the converter can make.
 *
 * @version 0.1
 **/
class ConversionDescriptor
{
          private Class toClass;
          private Class fromClass;

    /**
           * Construct a new conversion descriptor to describe the
conversion from
           * one class to another class.
           *
           * @param fromClass class the conversion goes from
           * @param toClass   class the conversion goes to
           */
          public ConversionDescriptor(Class fromClass, Class
toClass)
          {
                    if ((fromClass == null) || (toClass == null))
                    {
```

```
                                String msg = "no parameter can be
null";
                                throw new
IllegalArgumentException(msg);
                        }

                        this.toClass = toClass;
                        this.fromClass = fromClass;
            }

            /**
             * Get the class the conversion goes from.
             *
             * @return Class the conversion goes from
             */
            public Class getFromClass()
            {
                        return fromClass;
            }

            /**
             * Class the conversion goes to.
             *
             * @return class the conversion goes to
             */
            public Class getToClass()
            {
                        return toClass;
            }

    public boolean equals(Object o)
            {
        if (o == null || !(o instanceof ConversionDescriptor))
return false;

                        ConversionDescriptor cd =
(ConversionDescriptor) o;
        return cd.toClass.equals(toClass) &&
cd.fromClass.equals(fromClass);
            }

    public int hashCode()
            {
        return fromClass.hashCode() ^ toClass.hashCode();
            }
}


[End of: ./src/net/concedere/tata/convert/ConversionDescriptor.java]


[Contents of:
./src/net/concedere/tata/convert/ConversionException.java]


package net.concedere.tata.convert;

import net.concedere.tata.ScriptException;

/**
 * A general error that occured during conversion.
 *
 * @version 0.1
 **/
public class ConversionException extends ScriptException
{
            public ConversionException()
            {
                        super();
            }
```

```
	public ConversionException(String s)
	{
		super(s);
	}

	public ConversionException(String s, Throwable t)
	{
		super(s, t);
	}

	public ConversionException(Throwable t)
	{
		super(t);
	}
}

[End of: ./src/net/concedere/tata/convert/ConversionException.java]


[Contents of: ./src/net/concedere/tata/convert/Converter.java]


package net.concedere.tata.convert;

import org.apache.log4j.Logger;
import org.dom4j.XPath;
import org.dom4j.Document;
import org.dom4j.Node;

import java.util.*;

import net.concedere.tata.Script;

/**
 * Object responsible for performing conversions in the script.
 *
 * @version 0.1
 **/
public class Converter
{
	private Map conversionMap =
Collections.synchronizedMap(new HashMap());

	private static final Logger log =
Logger.getLogger(Converter.class);

	/**
	 * Perform a conversion.
	 *
	 * @param o       object to convert
	 * @param toClass class to convert it to
	 * @param script  Script the conversion is happening in
	 * @return result of the most appropriate conversion or
<code>null</code>
	 *         if the conversion is impossible
	 * @throws ConversionException
	 *         if something goes wrong
	 */
	public Object convert(Object o, Class toClass, Script
script)
				throws ConversionException
	{
		if ((toClass == null))
		{
			String msg = "toClass cannot be
null";
			throw new
IllegalArgumentException(msg);
```

```
                    }

                    if (o == null) return null;

                    // check for the short-circuit case
                    if (o.getClass() == toClass)
                    {
                              return o;
                    }

                    // check for the case of ArrayList.class ->
Collection.class or
                    // org.dom4j.Document->org.dom4j.Node
                    if (toClass.isInstance(o))
                    {
                              return o;
                    }

                    // HACK: we'll watch out for collections
implementations specifically
                    ConversionDescriptor cd  = null;
                    if (o instanceof Collection)
                    {
                              // look for the collection mapping if
no list was found
                              cd = new
ConversionDescriptor(Collection.class, toClass);
                    }
                    else if (o instanceof Document)
                    {
                              // look for a Document conversion
                              cd = new
ConversionDescriptor(Document.class, toClass);
                    }
                    else if (o instanceof XPath)
                    {
                              // look for an XPath conversion
                              cd = new
ConversionDescriptor(XPath.class, toClass);
                    }
                    else if (o instanceof Node)
                    {
                              cd = new
ConversionDescriptor(Node.class, toClass);
                    }
                    else
                    {
                              cd = new
ConversionDescriptor(o.getClass(), toClass);
                    }

        if (conversionMap.containsKey(cd))
                    {
                              Conversion c = (Conversion)
conversionMap.get(cd);
            return c.convert(o, toClass, script);
                    }
                    else
                    {
                              log.warn("Asked to perform impossible
conversion, object " + o
                                                  + " of type" +
o.getClass() + " to " + toClass);
                              return null;
                    }
          }

          /**
           * Register a possible Conversion with the Converter.
           *
```

```
            * @param fromClass    source class of the conversion
            * @param toClass      destination class of the conversion
            * @param conversion   class of the Conversion that
implements it
            * @throws ConversionException
            *         if the conversion can't be registered
            */
    public void register(Class fromClass, Class toClass, Class
conversion)
                    throws ConversionException
          {
        if ((fromClass == null) || (toClass == null) || (conversion
== null))
                    {
                              String msg = "no parameter can be
null";
                              throw new
IllegalArgumentException(msg);
                    }

        ConversionDescriptor cd = new
ConversionDescriptor(fromClass, toClass);

                    // instantiate the Conversion class
        Conversion c = null;
                    try
                    {
                              c =  (Conversion)
conversion.newInstance();
                    }
                    catch (Exception e)
                    {
                              String msg = "couldn't instantiate
Conversion from class "
                                                  + conversion;
                              throw new ConversionException(msg,
e);
                    }

        conversionMap.put(cd, c);
          }
}


[End of: ./src/net/concedere/tata/convert/Converter.java]


[Contents of:
./src/net/concedere/tata/convert/core/BooleanDoubleConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Convert a boolean to a double.
 *
 * @version 0.1
 **/
public class BooleanDoubleConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    if (((Boolean) o).booleanValue())
                    {
```

```
                                return new Double(1);
                        }
                        else
                        {
                                return new Double(0);
                        }
                }
}


[End of:
./src/net/concedere/tata/convert/core/BooleanDoubleConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/BooleanIntegerConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Convert a boolean to a number.
 *
 * @version 0.1
 **/
public class BooleanIntegerConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script
script)
                                throws ConversionException
        {
                if (((Boolean) o).booleanValue())
                {
                                return new Integer(1);
                }
                else
                {
                                return new Integer(0);
                }
        }
}


[End of:
./src/net/concedere/tata/convert/core/BooleanIntegerConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/BooleanStringConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Boolean to String conversion.
 *
 * @version 0.1
 **/
public class BooleanStringConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script
script)
```

```
                                throws ConversionException
        {
                return o.toString();
        }
}


[End of:
./src/net/concedere/tata/convert/core/BooleanStringConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/CollectionDoubleConversion.jav
a]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.util.Collection;

/**
 * Convert a collection to a double.
 *
 * @version 0.1
 **/
public class CollectionDoubleConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script
script)
                                throws ConversionException
        {
                return new Double(((Collection) o).size());
        }
}


[End of:
./src/net/concedere/tata/convert/core/CollectionDoubleConversion.jav
a]


[Contents of:
./src/net/concedere/tata/convert/core/CollectionIntegerConversion.ja
va]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.util.Collection;

/**
 * Convert a collection to a number.
 *
 * @version 0.1
 **/
public class CollectionIntegerConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script
script)
                                throws ConversionException
        {
                return new Integer(((Collection) o).size());
```

```
        }
}


[End of:
./src/net/concedere/tata/convert/core/CollectionIntegerConversion.ja
va]


[Contents of:
./src/net/concedere/tata/convert/core/CollectionStringConversion.jav
a]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.convert.Converter;
import net.concedere.tata.Script;

import java.util.Collection;
import java.util.Iterator;

/**
 * Converts a collection to a string.
 *
 * @version 0.1
 **/
public class CollectionStringConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script
script)
                        throws ConversionException
        {
                Converter convert = script.getConverter();
                StringBuffer buf = new StringBuffer("(");

                Collection list = (Collection) o;
                for (Iterator i = list.iterator(); i.hasNext();
)
                {
                        buf.append(' ');
                        Object elem = i.next();
                        buf.append((String)
convert.convert(elem, String.class, script));
          buf.append(' ');
                }
                buf.append(')');

                return buf.toString();
        }
}


[End of:
./src/net/concedere/tata/convert/core/CollectionStringConversion.jav
a]


[Contents of:
./src/net/concedere/tata/convert/core/DoubleIntegerConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
```

```
/**
 * Convert an integer to a double.
 *
 * @version 0.1
 **/
public class DoubleIntegerConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    return new Integer(((Double) o).intValue());
          }
}


[End of:
./src/net/concedere/tata/convert/core/DoubleIntegerConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/DoubleStringConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Converts doubles to strings.
 *
 * @version 0.1
 **/
public class DoubleStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script s)
throws ConversionException
          {
                    return o.toString();
          }
}


[End of:
./src/net/concedere/tata/convert/core/DoubleStringConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/IntegerDoubleConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Convert an integer to a double.
 *
 * @version 0.1
 **/
public class IntegerDoubleConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
```

```
                return new Double(((Integer) o).doubleValue());
        }
}


[End of:
./src/net/concedere/tata/convert/core/IntegerDoubleConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/IntegerStringConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Defines the conversion from an integer to a String.
 *
 * @version 0.1
 **/
public class IntegerStringConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script s)
throws ConversionException
        {
                return o.toString();
        }
}


[End of:
./src/net/concedere/tata/convert/core/IntegerStringConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/NumberBooleanConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Convert a number to a boolean. True if it's non-zero, false if it
 * is.
 *
 * @version 0.1
 **/
public class NumberBooleanConversion implements Conversion
{
        public Object convert(Object o, Class toClass, Script
script)
                        throws ConversionException
        {
                if (((Number) o).doubleValue() != 0)
                {
                        return Boolean.TRUE;
                }
                else
                {
                        return Boolean.FALSE;
                }
        }
}
```

[End of:
./src/net/concedere/tata/convert/core/NumberBooleanConversion.java]

[Contents of:
./src/net/concedere/tata/convert/core/StringIntegerConversion.java]

```
package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Defines conversion from a String to an integer.
 *
 * @version 0.1
 **/
public class StringIntegerConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script s)
throws ConversionException
          {
                    return new Integer((String) o);
          }
}
```

[End of:
./src/net/concedere/tata/convert/core/StringIntegerConversion.java]

[Contents of:
./src/net/concedere/tata/convert/core/StringUrlConversion.java]

```
package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.net.URL;
import java.net.MalformedURLException;

/**
 * Convert a String to a URL.
 *
 * @version 0.1
 **/
public class StringUrlConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script s)
throws ConversionException
          {
                    try
                    {
                              return new URL((String) o);
                    }
                    catch (MalformedURLException mue)
                    {
                              String msg = "couldn't convert string
" + o + " to url";
                              throw new ConversionException(msg,
mue);
                    }
          }
```

```
}


[End of:
./src/net/concedere/tata/convert/core/StringUrlConversion.java]


[Contents of:
./src/net/concedere/tata/convert/core/UrlStringConversion.java]


package net.concedere.tata.convert.core;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

/**
 * Convert a URL to a String.
 *
 * @version 0.1
 **/
public class UrlStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script s)
throws ConversionException
          {
                    return o.toString();
          }
}


[End of:
./src/net/concedere/tata/convert/core/UrlStringConversion.java]


[Contents of:
./src/net/concedere/tata/convert/http/HttpConnectionStringConversion
.java]


package net.concedere.tata.convert.http;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import HTTPClient.HTTPConnection;

import java.net.URL;
import java.net.MalformedURLException;

/**
 * Convert a HTTP connection to a String. This is the URL
representing the
 * web server that the connection is representing.
 *
 * @version 0.1
 **/
public class HttpConnectionStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    HTTPConnection conn = (HTTPConnection) o;

                    try
                    {
                              return new URL(conn.getProtocol(),
conn.getHost(),
```

```
conn.getPort(), "").toExternalForm();
                    }
                    catch (MalformedURLException mue)
                    {
                              String msg = "failed to construct a
url";
                              throw new ConversionException(msg,
mue);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/http/HttpConnectionStringConversion
.java]


[Contents of:
./src/net/concedere/tata/convert/http/HttpConnectionUrlConversion.ja
va]


package net.concedere.tata.convert.http;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import HTTPClient.HTTPConnection;

import java.net.URL;
import java.net.MalformedURLException;

/**
 * Transparently convert an HTTP Connection to a URL.
 *
 * @version 0.1
 **/
public class HttpConnectionUrlConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    HTTPConnection conn = (HTTPConnection) o;

                    try
                    {
                              return new URL(conn.getProtocol(),
conn.getHost(),

conn.getPort(), "");
                    }
                    catch (MalformedURLException mue)
                    {
                              String msg = "failed to construct a
url";
                              throw new ConversionException(msg,

mue);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/http/HttpConnectionUrlConversion.ja
va]
```

[Contents of:
./src/net/concedere/tata/convert/http/HttpCredentialsStringConversio
n.java]

```
package net.concedere.tata.convert.http;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import net.concedere.tata.http.HttpCredentials;

/**
 * Converts HTTP credentials to a string.
 *
 * @version 0.1
 **/
public class HttpCredentialsStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    HttpCredentials creds = (HttpCredentials) o;

                    StringBuffer buf = new StringBuffer("http-
credentials[");
        buf.append("type=").append(creds.getTypeDesc());
                    buf.append(", realm=").append(creds.getRealm());
                    buf.append(", user=").append(creds.getUser());
                    buf.append(",
pass=").append(creds.getPass()).append("]");
                    return buf.toString();
          }
}
```

[End of:
./src/net/concedere/tata/convert/http/HttpCredentialsStringConversio
n.java]

[Contents of:
./src/net/concedere/tata/convert/http/HttpResponseStringConversion.j
ava]

```
package net.concedere.tata.convert.http;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import net.concedere.tata.http.HttpResponse;
import net.concedere.tata.http.HttpException;

/**
 * Convert a HTTP response to a String by printing out the response
data as a
 * string.
 *
 * @version 0.1
 **/
public class HttpResponseStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    try
                    {
```

```
                                return ((HttpResponse)
o).getString();
                        }
                        catch (HttpException he)
                        {
                                String msg = "couldn't print out
response string";
                                throw new ConversionException(msg,
he);
                        }
                }
}


[End of:
./src/net/concedere/tata/convert/http/HttpResponseStringConversion.j
ava]


[Contents of:
./src/net/concedere/tata/convert/http/NvPairStringConversion.java]


package net.concedere.tata.convert.http;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import HTTPClient.NVPair;

/**
 * Convert an NvPair into a String.
 *
 * @version 0.1
 **/
public class NvPairStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                                throws ConversionException
          {
                        return ((NVPair) o).getName() + "=" + ((NVPair)
o).getValue();
          }
}


[End of:
./src/net/concedere/tata/convert/http/NvPairStringConversion.java]


[Contents of:
./src/net/concedere/tata/convert/http/UrlHttpConnectionConversion.ja
va]


package net.concedere.tata.convert.http;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.net.URL;

import HTTPClient.HTTPConnection;
import HTTPClient.ProtocolNotSuppException;

/**
 * Transparently convert a URL to a HTTP connection.
 *
```

```
 * @version 0.1
 **/
public class UrlHttpConnectionConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    URL url = (URL) o;

                    try
                    {
                              return new HTTPConnection(url);
                    }
                    catch (ProtocolNotSuppException pnse)
                    {
                              String msg = "couldn't convert url to
http connection";
                              throw new ConversionException(msg,
pnse);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/http/UrlHttpConnectionConversion.ja
va]


[Contents of:
./src/net/concedere/tata/convert/io/FileInputStreamConversion.java]


package net.concedere.tata.convert.io;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.io.*;

/**
 * Converts a file to an inputstream.
 *
 * @version 0.1
 **/
public class FileInputStreamConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    File f = (File) o;
                    if (f.isDirectory())
                    {
                              String msg = f + " is a directory!";
                              throw new ConversionException(msg);
                    }

                    try
                    {
                              return new BufferedInputStream(new
FileInputStream(f));
                    }
                    catch (IOException ioe)
                    {
                              String msg = "failed to convert
object " + o + " to file";
                              throw new ConversionException(msg,
```

```
ioe);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/io/FileInputStreamConversion.java]


[Contents of:
./src/net/concedere/tata/convert/io/FileOutputStreamConversion.java]


package net.concedere.tata.convert.io;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.io.File;
import java.io.FileOutputStream;
import java.io.BufferedOutputStream;
import java.io.IOException;

/**
 *
 * @version 0.1
 **/
public class FileOutputStreamConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    File f = (File) o;
                    if (f.isDirectory())
                    {
                              String msg = f + " is a directory!";
                              throw new ConversionException(msg);
                    }

        try
                    {
                              return new BufferedOutputStream(new
FileOutputStream(f));
                    }
                    catch (IOException ioe)
                    {
                              String msg = "couldn't convert " + f
+ " to outputstream";
                              throw new ConversionException(msg,
ioe);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/io/FileOutputStreamConversion.java]


[Contents of:
./src/net/concedere/tata/convert/io/FileStringConversion.java]


package net.concedere.tata.convert.io;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
```

```
import net.concedere.tata.Script;

import java.io.File;
import java.io.IOException;

/**
 * Converts a file to a string.
 *
 * @version 0.1
 **/
public class FileStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    try
                    {
                              return ((File) o).getCanonicalPath();
                    }
                    catch (IOException ioe)
                    {
                              String msg = "failed to convert
file";
                              throw new ConversionException(msg,
ioe);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/io/FileStringConversion.java]


[Contents of:
./src/net/concedere/tata/convert/io/StringFileConversion.java]


package net.concedere.tata.convert.io;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.io.File;

/**
 * Convert a string to a file if the String represents a path.
 *
 * @version 0.1
 **/
public class StringFileConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    return new File((String) o);
          }
}


[End of:
./src/net/concedere/tata/convert/io/StringFileConversion.java]


[Contents of:
./src/net/concedere/tata/convert/io/StringInputStreamConversion.java
]
```

```
package net.concedere.tata.convert.io;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.io.ByteArrayInputStream;

/**
 *
 * @version 0.1
 **/
public class StringInputStreamConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    return new ByteArrayInputStream(((String)
o).getBytes());
          }
}

[End of:
./src/net/concedere/tata/convert/io/StringInputStreamConversion.java
]


[Contents of:
./src/net/concedere/tata/convert/io/UrlInputStreamConversion.java]


package net.concedere.tata.convert.io;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import java.net.URL;
import java.io.IOException;

/**
 * Convert a URL into an InputStream.
 *
 * @version 0.1
 **/
public class UrlInputStreamConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    try
                    {
                              return ((URL) o).openStream();
                    }
                    catch (IOException ioe)
                    {
                              String msg = "couldn't open stream to
" + o;
                              throw new ConversionException(msg,
ioe);
                    }
          }
}


[End of:
```

./src/net/concedere/tata/convert/io/UrlInputStreamConversion.java]

[Contents of:
./src/net/concedere/tata/convert/xml/DocumentInputStreamConversion.j
ava]

```
package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import org.dom4j.Document;

import java.io.StringReader;
import java.io.InputStreamReader;
import java.io.ByteArrayInputStream;

/**
 * Converts a document into an InputStream for other reading.
 *
 * @version 0.1
 **/
public class DocumentInputStreamConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    Document doc = (Document) o;
                    return new
ByteArrayInputStream(doc.asXML().getBytes());

          }
}
```

[End of:
./src/net/concedere/tata/convert/xml/DocumentInputStreamConversion.j
ava]

[Contents of:
./src/net/concedere/tata/convert/xml/DocumentStringConversion.java]

```
package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import org.dom4j.Document;

/**
 * Converts an XML document to a string.
 *
 * @version 0.1
 **/
public class DocumentStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    return ((Document) o).asXML();
          }
}
```

[End of:
./src/net/concedere/tata/convert/xml/DocumentStringConversion.java]

[Contents of:
./src/net/concedere/tata/convert/xml/InputStreamDocumentConversion.j
ava]

```
package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import net.concedere.tata.ScriptException;
import org.dom4j.io.SAXReader;
import org.dom4j.DocumentException;

import java.io.InputStream;

/**
 * Transparently convert an InputStream to a document.
 *
 * @version 0.1
 **/
public class InputStreamDocumentConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    try
                    {
                              SAXReader reader = new SAXReader();
                              return reader.read((InputStream) o);
                    }
                    catch (DocumentException de)
                    {
                              String msg = "couldn't parse xml
stream";
                              throw new ConversionException(msg,
de);
                    }
          }
}
```

[End of:
./src/net/concedere/tata/convert/xml/InputStreamDocumentConversion.j
ava]

[Contents of:
./src/net/concedere/tata/convert/xml/NodeStringConversion.java]

```
package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import org.dom4j.Node;

/**
 * Converts a generic DOM Node into a string.
 *
 * @version 0.1
 **/
public class NodeStringConversion implements Conversion
{
```

```
	public Object convert(Object o, Class toClass, Script
script)
				throws ConversionException
	{
		return ((Node) o).getText();
	}
}
```

[End of:
./src/net/concedere/tata/convert/xml/NodeStringConversion.java]

[Contents of:
./src/net/concedere/tata/convert/xml/StringDocumentConversion.java]

```
package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import org.dom4j.io.SAXReader;
import org.dom4j.DocumentException;

/**
 * Convert a string to a document transparently.
 *
 * @version 0.1
 **/
public class StringDocumentConversion implements Conversion
{
	public Object convert(Object o, Class toClass, Script
script)
				throws ConversionException
	{
		try
		{
				return new SAXReader().read((String)
o);
		}
		catch (DocumentException de)
		{
				String msg = "couldn't convert
document from string " + o;
				throw new ConversionException(msg,
de);
		}
	}
}
```

[End of:
./src/net/concedere/tata/convert/xml/StringDocumentConversion.java]

[Contents of:
./src/net/concedere/tata/convert/xml/StringXsltConversion.java]

```
package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;

import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.stream.StreamSource;
import java.io.StringReader;
```

```
/**
 * Convert a string directly into a transformer.
 *
 * @version 0.1
 **/
public class StringXsltConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    try
                    {
                              TransformerFactory factory =
TransformerFactory.newInstance();
                              return factory.newTransformer(new
StreamSource(
                                                  new
StringReader((String )o)));
                    }
                    catch (TransformerConfigurationException tce)
                    {
                              String msg = "couldn't convert string
to transformer";
                              throw new ConversionException(msg,
tce);
                    }
          }
}


[End of:
./src/net/concedere/tata/convert/xml/StringXsltConversion.java]


[Contents of:
./src/net/concedere/tata/convert/xml/XpathStringConversion.java]


package net.concedere.tata.convert.xml;

import net.concedere.tata.convert.Conversion;
import net.concedere.tata.convert.ConversionException;
import net.concedere.tata.Script;
import org.dom4j.XPath;

/**
 * Convert an XPath object into a String.
 *
 * @version 0.1
 **/
public class XpathStringConversion implements Conversion
{
          public Object convert(Object o, Class toClass, Script
script)
                              throws ConversionException
          {
                    return ((XPath) o).getText();
          }
}


[End of:
./src/net/concedere/tata/convert/xml/XpathStringConversion.java]


[Contents of: ./src/net/concedere/tata/core/AddFunction.java]
```

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Implements basic addition. Like awk, all numbers are floats.
 *
 * @version 0.1
 **/
public class AddFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object o1 = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);

        Double d1 = (Double) script.getConverter()
                                          .convert(o1, Double.class,
script);
                    Double d2 = (Double) script.getConverter()
                                          .convert(o2, Double.class,
script);

        return new Double(d1.doubleValue() + d2.doubleValue());
          }
}


[End of: ./src/net/concedere/tata/core/AddFunction.java]


[Contents of: ./src/net/concedere/tata/core/AndFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Returns true if both its boolean arguments are true. This method
never
 * short-circuits.
 *
 * @version 0.1
 **/
public class AndFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object a1 = f1.eval(script, context);
                    Boolean b1 = (Boolean) script.getConverter()
                                          .convert(a1, Boolean.class,
script);

        Function f2 = script.getFunctionStack().pop();
                    Object a2= f2.eval(script, context);
                    Boolean b2 = (Boolean) script.getConverter()
                                          .convert(a2, Boolean.class,
script);
```

```
        return new Boolean(b1.booleanValue() && b2.booleanValue());
          }
}


[End of: ./src/net/concedere/tata/core/AndFunction.java]


[Contents of: ./src/net/concedere/tata/core/AssignFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Rather than simply setting a new variable in the current context,
as the
 * set function always does, the assign function searches for a
previously
 * defined value and replaces its value.
 *
 * @version 0.1
 **/
public class AssignFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object idobj = f1.eval(script, context);
                    String id = (String) script.getConverter()
                                        .convert(idobj,
String.class, script);

        // now find the function
                    Context containing = ctx.findContaining(id);
                    if (containing == null)
                    {
                              String msg = "no such variable as " +
id;
                              throw new ScriptException(msg);
                    }

        // now, we figure out what to replace it with
                    Function valFunc =
script.getFunctionStack().pop();
                    Object valObj = valFunc.eval(script, context);
                    containing.set(id, valObj);

                    context.destroy();
                    return valObj;
          }
}


[End of: ./src/net/concedere/tata/core/AssignFunction.java]


[Contents of: ./src/net/concedere/tata/core/BeginBlockFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Executes the body of a block.
```

```
 *
 * @version 0.1
 **/
public class BeginBlockFunction extends AbstractFunction
                    implements BlockFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        // get every function but the EndBlock function
        Object result = null;
                    while (true)
                    {
                              Function f1 =
script.getFunctionStack().pop();

                              if (f1.getClass() ==
EndBlockFunction.class) break;
          result = f1.eval(script, context);
                    }

                    context.destroy();
                    return result;
          }

          public Class getEndFunctionClass()
          {
                    return EndBlockFunction.class;
          }
}


[End of: ./src/net/concedere/tata/core/BeginBlockFunction.java]


[Contents of: ./src/net/concedere/tata/core/BeginListFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.List;
import java.util.ArrayList;

/**
 * Function that begins a list.
 *
 * @version 0.1
 **/
public class BeginListFunction extends AbstractFunction
                    implements BlockFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    List list = new ArrayList();

                    Context context = new Context(ctx);

                    while (true)
                    {
                              Function f1 =
script.getFunctionStack().pop();
                              if (f1.getClass() ==
EndListFunction.class) break;

                              list.add(f1.eval(script, context));
```

```
                        }

                        context.destroy();
                        return list;
            }

            public Class getEndFunctionClass()
            {
                        return EndListFunction.class;
            }
}


[End of: ./src/net/concedere/tata/core/BeginListFunction.java]


[Contents of:
./src/net/concedere/tata/core/ConcatenationFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Concatenates two strings.
 *
 * @version 0.1
 **/
public class ConcatenationFunction extends AbstractFunction
{
            public Object eval(Script script, Context ctx)
                                    throws ScriptException
            {
                        Context context = new Context(ctx);

                        Function f1 = script.getFunctionStack().pop();
                        Object o1 = f1.eval(script, context);
                        String s1 = (String) script.getConverter()
                                                .convert(o1, String.class,
script);

                        Function f2 = script.getFunctionStack().pop();
                        Object o2 = f2.eval(script, context);
                        String s2 = (String) script.getConverter()
                                                .convert(o2, String.class,
script);

                        if (s1 == null) return s2;
                        if (s2 == null) return s1;

                        return s1 + s2;
            }
}


[End of: ./src/net/concedere/tata/core/ConcatenationFunction.java]


[Contents of: ./src/net/concedere/tata/core/CountFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.ArrayList;
import java.util.Iterator;
```

```
/**
 * Counts how many items fulfill some criteria in a collection.
 *
 * @version 0.1
 **/
public class CountFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the id to bind
        Function idFun = script.getFunctionStack().pop();
                    Object sobj = idFun.eval(script, context);
                    String id = (String) script.getConverter()
                                                .convert(sobj,
String.class, script);

                    // if id is null, it's an error
                    if (id == null)
                    {
                              String msg = "foreach requires non-
null id string for first arg";
                              throw new ScriptException(msg);
                    }

        // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                    // if col is null, short-circuit out
                    if (col == null) return null;

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                    Script blockScript = new Script(body);
                    blockScript.setConverter(script.getConverter());

                    int result = 0;
                    for (Iterator iter = col.iterator();
iter.hasNext(); )
                    {
            // bind the variable in the context
                              Object val = iter.next();
                              context.set(id, val);

                              // mark the stack if this isn't the
last iteration
                              if (iter.hasNext())
                              {
                                        blockScript.getFunctionStac
k().mark();
                              }

                              // execute the function block
                              Object bo =
body.pop().eval(blockScript, context);
                              Boolean b = (Boolean) blockScript
                                                  .getConverter().
convert(bo, Boolean.class, blockScript);
            if (b.booleanValue())
                              {
                                        result += 1;
                              }
```

```
                                // restore the stack if this isn't
the last iteration!
                                if (iter.hasNext())
                                {
                                        blockScript.getFunctionStac
k().unmark();
                                }
                        }

                        context.destroy();
                        return new Integer(result);
                }
}


[End of: ./src/net/concedere/tata/core/CountFunction.java]


[Contents of: ./src/net/concedere/tata/core/DiffFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.ArrayList;

/**
 * One collection minus another.
 *
 * @version 0.1
 **/
public class DiffFunction extends AbstractFunction
{
        public Object eval(Script script, Context ctx)
                                throws ScriptException
        {
                        Context context = new Context(ctx);

                        Function colF1 =
script.getFunctionStack().pop();
                Object cobj1 = colF1.eval(script, context);
                        Collection co1 = (Collection)
script.getConverter()
                                                .convert(cobj1,
Collection.class, script);

                Function colF2 = script.getFunctionStack().pop();
                        Object cobj2 = colF2.eval(script, context);
                        Collection co2 = (Collection)
script.getConverter()
                                                .convert(cobj2,
Collection.class, script);

                if (co1 == null)
                        {
                                return null;
                        }

                        if (co2 == null) return co1;


                        ArrayList list = new ArrayList(co1);
                        list.removeAll(co2);
                        return list;
        }
}
```

[End of: ./src/net/concedere/tata/core/DiffFunction.java]

[Contents of: ./src/net/concedere/tata/core/DivideFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Divide two numbers.
 *
 * @version 0.1
 **/
public class DivideFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object o1 = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);

        Double d1 = (Double) script.getConverter()
                                          .convert(o1, Double.class,
script);
                    Double d2 = (Double) script.getConverter()
                                          .convert(o2, Double.class,
script);

        return new Double(d1.doubleValue() / d2.doubleValue());
          }
}
```

[End of: ./src/net/concedere/tata/core/DivideFunction.java]

[Contents of:
./src/net/concedere/tata/core/DoubleValueFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Represents a constant double.
 *
 * @version 0.1
 **/
public class DoubleValueFunction extends AbstractFunction
{
          private Double d;

          /**
           * Construct a new double value function.
           *
           * @param d double to be returned
           */
          public DoubleValueFunction(Double d)
          {
                    if (d == null)
                    {
```

```
                                String msg = "d can't be null";
                                throw new
IllegalArgumentException(msg);
                        }

                        this.d = d;
                }

                public Object eval(Script script, Context ctx)
                                throws ScriptException
                {
                        return d;
                }
}
```

[End of: ./src/net/concedere/tata/core/DoubleValueFunction.java]

[Contents of: ./src/net/concedere/tata/core/ElseFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * A marker function used to structure if statements.
 *
 * @version 0.1
 **/
public class ElseFunction extends AbstractFunction
{
                public Object eval(Script script, Context ctx)
                                throws ScriptException
                {
                        return null;
                }
}
```

[End of: ./src/net/concedere/tata/core/ElseFunction.java]

[Contents of: ./src/net/concedere/tata/core/EndBlockFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * A marker function used to indicate the end of blocks.
 *
 * @version 0.1
 **/
public class EndBlockFunction extends AbstractFunction
{
                public Object eval(Script script, Context ctx)
                                throws ScriptException
                {
                        return null;
                }
}
```

[End of: ./src/net/concedere/tata/core/EndBlockFunction.java]

[Contents of: ./src/net/concedere/tata/core/EndIfFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * Marker function that marks the unconditional end of the if
function.
 *
 * @version 0.1
 **/
public class EndIfFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return null;
          }
}
```

[End of: ./src/net/concedere/tata/core/EndIfFunction.java]

[Contents of: ./src/net/concedere/tata/core/EndListFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * A marker function used by the BeginListFunction to determine the
end of
 * lists.
 *
 * @version 0.1
 **/
public class EndListFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return null;
          }
}
```

[End of: ./src/net/concedere/tata/core/EndListFunction.java]

[Contents of: ./src/net/concedere/tata/core/EqualsFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;
```

```
/**
 * Function that determines whether its arguments are true.
 *
 * @version 0.1
 **/
public class EqualsFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object a1 = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object a2 = f2.eval(script, context);

                    // have to check for null
                    context.destroy();
                    if (a1 == null)
                    {
                              if (a2 == null)
                              {
                                        return Boolean.TRUE;
                              }
                              else
                              {
                                        return Boolean.FALSE;
                              }
                    }

                    // before we can convert it, we must convert a2
to a1's type
        a2 = script.getConverter().convert(a2, a1.getClass(),
script);

                    // otherwise do the normal test
                    if (a1.equals(a2))
                    {
                              return Boolean.TRUE;
                    }
                    else
                    {
                              return Boolean.FALSE;
                    }
          }
}


[End of: ./src/net/concedere/tata/core/EqualsFunction.java]


[Contents of: ./src/net/concedere/tata/core/FalseValueFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * Function that always returns false.
 *
 * @version 0.1
 **/
public class FalseValueFunction extends AbstractFunction
```

```
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return Boolean.FALSE;
          }
}


[End of: ./src/net/concedere/tata/core/FalseValueFunction.java]


[Contents of: ./src/net/concedere/tata/core/FilterFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.Iterator;
import java.util.ArrayList;

/**
 * Implements the higher order filter function.
 *
 * @version 0.1
 **/
public class FilterFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {

        Context context = new Context(ctx);

                    // get the id to bind
        Function idFun = script.getFunctionStack().pop();
                    Object sobj = idFun.eval(script, context);
                    String id = (String) script.getConverter()
                                                .convert(sobj,
String.class, script);

                    // if id is null, it's an error
                    if (id == null)
                    {
                              String msg = "foreach requires non-
null id string for first arg";
                              throw new ScriptException(msg);
                    }

        // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                    // if col is null, short-circuit out
                    if (col == null) return null;

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                    Script blockScript = new Script(body);
                    blockScript.setConverter(script.getConverter());

                    Collection result = new ArrayList();
                    for (Iterator iter = col.iterator();
```

```
iter.hasNext(); )
                        {
            // bind the variable in the context
                                    Object val = iter.next();
                                    context.set(id, val);

                                    // mark the stack if this isn't the
last iteration
                                    if (iter.hasNext())
                                    {
                                              blockScript.getFunctionStac
k().mark();
                                    }

                                    // execute the function block
                                    Object bo =
body.pop().eval(blockScript, context);
                                    Boolean b = (Boolean) blockScript
                                                        .getConverter().
convert(bo, Boolean.class, blockScript);
            if (b.booleanValue())
                                    {
                                              result.add(val);
                                    }

                                    // restore the stack if this isn't
the last iteration!
                                    if (iter.hasNext())
                                    {
                                              blockScript.getFunctionStac
k().unmark();
                                    }
                        }

                        context.destroy();
                        return result;
          }
}


[End of: ./src/net/concedere/tata/core/FilterFunction.java]


[Contents of: ./src/net/concedere/tata/core/FilterOutFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * Remove all elements from a collection which return false.
 *
 * @version 0.1
 **/
public class FilterOutFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                    throws ScriptException
          {
                       Context context = new Context(ctx);

                       // get the id to bind
         Function idFun = script.getFunctionStack().pop();
                       Object sobj = idFun.eval(script, context);
                       String id = (String) script.getConverter()
```

```
                                        .convert(sobj,
String.class, script);

                    // if id is null, it's an error
                    if (id == null)
                    {
                            String msg = "foreach requires non-
null id string for first arg";
                            throw new ScriptException(msg);
                    }

        // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                        .convert(cobj,
Collection.class, script);

                    // if col is null, short-circuit out
                    if (col == null) return null;

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                    Script blockScript = new Script(body);
                    blockScript.setConverter(script.getConverter());

                    Collection result = new ArrayList();
                    for (Iterator iter = col.iterator();
iter.hasNext(); )
                    {
            // bind the variable in the context
                            Object val = iter.next();
                            context.set(id, val);

                            // mark the stack if this isn't the
last iteration
                            if (iter.hasNext())
                            {
                                    blockScript.getFunctionStac
k().mark();
                            }

                            // execute the function block
                            Object bo =
body.pop().eval(blockScript, context);
                            Boolean b = (Boolean) blockScript
                                                .getConverter().
convert(bo, Boolean.class, blockScript);
            if (!b.booleanValue())
                            {
                                    result.add(val);
                            }

                            // restore the stack if this isn't
the last iteration!
                            if (iter.hasNext())
                            {
                                    blockScript.getFunctionStac
k().unmark();
                            }
                    }

                    context.destroy();
                    return result;
          }
}
```

[End of: ./src/net/concedere/tata/core/FilterOutFunction.java]

[Contents of: ./src/net/concedere/tata/core/FindFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.Iterator;

/**
 * Find the first instance of some object in a collection.
 *
 * @version 0.1
 **/
public class FindFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the id to bind
        Function idFun = script.getFunctionStack().pop();
                    Object sobj = idFun.eval(script, context);
                    String id = (String) script.getConverter()
                                        .convert(sobj,
String.class, script);

                    // if id is null, it's an error
                    if (id == null)
                    {
                              String msg = "foreach requires non-
null id string for first arg";
                              throw new ScriptException(msg);
                    }

        // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                        .convert(cobj,
Collection.class, script);

                    // if col is null, short-circuit out
                    if (col == null) return null;

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                    Script blockScript = new Script(body);
                    blockScript.setConverter(script.getConverter());

                    Object result = null;
                    for (Iterator iter = col.iterator();
iter.hasNext(); )
                    {
            // bind the variable in the context
                              Object val = iter.next();
                              context.set(id, val);

                              // mark the stack if this isn't the
last iteration
                              if (iter.hasNext())
                              {
                                        blockScript.getFunctionStac
```

```
k().mark();
                                }

                                // execute the function block
                                Object bo =
body.pop().eval(blockScript, context);
                                Boolean b = (Boolean) blockScript
                                                .getConverter().
convert(bo, Boolean.class, blockScript);
            if (b.booleanValue())
                                {
                                        result = val;
                                        break;
                                }

                                // restore the stack if this isn't
the last iteration!
                                if (iter.hasNext())
                                {
                                        blockScript.getFunctionStac
k().unmark();
                                }
                        }

                        context.destroy();
                        return result;
                }
}


[End of: ./src/net/concedere/tata/core/FindFunction.java]


[Contents of: ./src/net/concedere/tata/core/FindLastFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.*;

/**
 * Same as the find function but it iterates over the collection
backwards.
 *
 * @version 0.1
 **/
public class FindLastFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                        Context context = new Context(ctx);

                        // get the id to bind
                Function idFun = script.getFunctionStack().pop();
                        Object sobj = idFun.eval(script, context);
                        String id = (String) script.getConverter()
                                                .convert(sobj,
String.class, script);

                        // if id is null, it's an error
                        if (id == null)
                        {
                                String msg = "foreach requires non-
null id string for first arg";
                                throw new ScriptException(msg);
                        }
```

```
        // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                    // if col is null, short-circuit out
                    if (col == null) return null;

                    // reverse the collection
                    List revList = Arrays.asList(col.toArray(new
Object[col.size()]));
        Collections.reverse(revList);

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                    Script blockScript = new Script(body);
                    blockScript.setConverter(script.getConverter());

                    Object result = null;
                    for (Iterator iter = revList.iterator();
iter.hasNext(); )
                    {
            // bind the variable in the context
                                Object val = iter.next();
                                context.set(id, val);

                                // mark the stack if this isn't the
last iteration
                                if (iter.hasNext())
                                {
                                          blockScript.getFunctionStac
k().mark();
                                }

                                // execute the function block
                                Object bo =
body.pop().eval(blockScript, context);
                                Boolean b = (Boolean) blockScript
                                                    .getConverter().
convert(bo, Boolean.class, blockScript);
            if (b.booleanValue())
                                {
                                          result = val;
                                          break;
                                }

                                // restore the stack if this isn't
the last iteration!
                                if (iter.hasNext())
                                {
                                          blockScript.getFunctionStac
k().unmark();
                                }
                    }

                    context.destroy();
                    return result;
          }
}


[End of: ./src/net/concedere/tata/core/FindLastFunction.java]


[Contents of: ./src/net/concedere/tata/core/ForEachFunction.java]
```

```
package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.Iterator;

/**
 * The standard looping mechanism.
 *
 * @version 0.1
 **/
public class ForEachFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    // get the id to bind
        Function idFun = script.getFunctionStack().pop();
                    Object sobj = idFun.eval(script, context);
                    String id = (String) script.getConverter()
                                                .convert(sobj,
String.class, script);

                    // if id is null, it's an error
                    if (id == null)
                    {
                              String msg = "foreach requires non-
null id string for first arg";
                              throw new ScriptException(msg);
                    }

        // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                    // if col is null, short-circuit out
                    if (col == null) return null;

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                    Script blockScript = new Script(body);
                    blockScript.setConverter(script.getConverter());

                    Object result = null;
                    for (Iterator iter = col.iterator();
iter.hasNext(); )
                    {
            // bind the variable in the context
                              context.set(id, iter.next());

                              // mark the stack if this isn't the
last iteration
                              if (iter.hasNext())
                              {
                                        blockScript.getFunctionStac
k().mark();
                              }

                              // execute the function block
                              result = body.pop().eval(blockScript,
context);
```

```
                                // restore the stack if this isn't
the last iteration!
                                if (iter.hasNext())
                                {
                                        blockScript.getFunctionStac
k().unmark();
                                }
                        }

                        context.destroy();
                        return result;
                }
}


[End of: ./src/net/concedere/tata/core/ForEachFunction.java]


[Contents of: ./src/net/concedere/tata/core/GetFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Retrieve a variable from the context.
 *
 * @version 0.1
 **/
public class GetFunction extends AbstractFunction
{
        public Object eval(Script script, Context ctx)
                                throws ScriptException
        {
                Context context = new Context(ctx);

      Function f1 = script.getFunctionStack().pop();
                Object sobj = f1.eval(script, context);
                String id = (String) script.getConverter()
                                        .convert(sobj,
String.class, script);

                context.destroy();
                return ctx.get(id);
        }
}


[End of: ./src/net/concedere/tata/core/GetFunction.java]


[Contents of:
./src/net/concedere/tata/core/GreaterThanEqualsFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Determine if two objects are greater than or equal to another
object.
 *
 * @version 0.1
 **/
public class GreaterThanEqualsFunction extends AbstractFunction
{
        public Object eval(Script script, Context ctx) throws
```

```
ScriptException
          {
          Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);

                    context.destroy();

                    // null is never greater than anything
                    if (o1 == null)
                    {
                              if (o2 == null)
                              {
                                        return Boolean.TRUE;
                              }
                              else
                              {
                                        return Boolean.FALSE;
                              }
                    }


                    // if comparable, use the natural ordering
                    if (o1 instanceof Comparable)
                    {
                              return Boolean.valueOf(
(((Comparable) o1).compareTo(o2) >= 0) );
                    }

                    // otherwise, let's convert both to strings
                    String s1 = (String) script.getConverter()
                                        .convert(o1, String.class,
script);
        String s2 = (String) script.getConverter()
                                        .convert(o2, String.class,
script);

                    if (s1 != null)
                    {
                              return
Boolean.valueOf(s1.compareTo(s2) >= 0);
                    }
                    else
                    {
                              return Boolean.FALSE;
                    }
          }
}


[End of:
./src/net/concedere/tata/core/GreaterThanEqualsFunction.java]


[Contents of:
./src/net/concedere/tata/core/GreaterThanFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Determines if one object is greater than another object.
 *
 * @version 0.1
 **/
```

```
public class GreaterThanFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);

                    context.destroy();

                    // null is never greater than anything
                    if (o1 == null)
                    {
                              return Boolean.FALSE;
                    }

                    // if comparable, use the natural ordering
                    if (o1 instanceof Comparable)
                    {
                              return Boolean.valueOf(
(((Comparable) o1).compareTo(o2) > 0) );
                    }

                    // otherwise, let's convert both to strings
                    String s1 = (String) script.getConverter()
                                                .convert(o1, String.class,
script);
        String s2 = (String) script.getConverter()
                                                .convert(o2, String.class,
script);

                    if (s1 != null)
                    {
                              return
Boolean.valueOf(s1.compareTo(s2) > 0);
                    }
                    else
                    {
                              return Boolean.FALSE;
                    }
          }
}


[End of: ./src/net/concedere/tata/core/GreaterThanFunction.java]


[Contents of: ./src/net/concedere/tata/core/IfFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Allows basic conditional logic in scripts.
 *
 * @version 0.1
 **/
public class IfFunction extends AbstractFunction implements
BlockFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);
```

```
                        Function boolFunc =
script.getFunctionStack().pop();
                        // now evaluate the boolean function to
collapse the stack to the next
                        // function which is what to execute on true!
                        Object boj = boolFunc.eval(script, context);
                        Boolean b = (Boolean)
                                                script.getConverter().conve
rt(boj, Boolean.class, script);

                FunctionStack trueBlock =
script.getFunctionStack().popBlock();

                        // peek and see if there's an else function
                        Function elseFun =
script.getFunctionStack().pop();
                        FunctionStack falseBlock = null;
                        if (elseFun.getClass() == ElseFunction.class)
                        {
                                falseBlock =
script.getFunctionStack().popBlock();

                                // now pop the ending if function
                                script.getFunctionStack().pop();
                        }
                        else if (elseFun.getClass() ==
EndIfFunction.class)
                        {
                                ; // do nothing
                        }
                        else
                        {
                                String msg = "if must be followed by
else or endif";
                                throw new ScriptException(msg);
                        }

                // create a new Scrpt object!
                        if (b != null && b.booleanValue())
                        {
                                Script trueScript = new
Script(trueBlock);
                                trueScript.setConverter(script.getCon
verter());
                                trueBlock.pop().eval(trueScript,
context);
                        }
                        else if (falseBlock != null)
                        {
                                Script falseScript = new
Script(falseBlock);
                                falseScript.setConverter(script.getCo
nverter());
                                falseBlock.pop().eval(falseScript,
context);
                        }

                        context.destroy();
                        return b;
                }

                public Class getEndFunctionClass()
                {
                        return EndIfFunction.class;
                }
}


[End of: ./src/net/concedere/tata/core/IfFunction.java]
```

[Contents of: ./src/net/concedere/tata/core/IndexFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.List;
import java.util.Iterator;

/**
 * Retrieve the specified index in the collection.
 *
 * @version 0.1
 **/
public class IndexFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function numFun =
script.getFunctionStack().pop();
                    Object nob = numFun.eval(script, context);
                    Integer i = (Integer) script.getConverter()
                                        .convert(nob,
Integer.class, script);


                    // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                        .convert(cobj,
Collection.class, script);

                    if (col == null)
                    {
                              return null;
                    }

        if (col instanceof List)
                    {
                              return ((List)
col).get(i.intValue());
                    }
                    else
                    {
            // loop until we get to the value
                              int counter = i.intValue() + 1;
                              for (Iterator it = col.iterator();
it.hasNext(); )
                              {
                                        Object var = it.next();
                counter -= 1;
                                        if (counter == 0) return
var;
                              }

                              String msg = "index is bad";
                              throw new
IndexOutOfBoundsException(msg);
                    }
          }
```

```
}
```

[End of: ./src/net/concedere/tata/core/IndexFunction.java]

[Contents of:
./src/net/concedere/tata/core/IntegerValueFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Represents a constant integer.
 *
 * @version 0.1
 **/
public class IntegerValueFunction extends AbstractFunction
{
          private Integer i;

          /**
           * Construct a new integer function to represent the given
integer.
           *
           * @param i integer this function will evaluate to
           */
          public IntegerValueFunction(Integer i)
          {
                    if (i == null)
                    {
                              String msg = "i can't be null";
                              throw new
IllegalArgumentException(msg);
                    }

                    this.i = i;
          }

          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return i;
          }
}
```

[End of: ./src/net/concedere/tata/core/IntegerValueFunction.java]

[Contents of:
./src/net/concedere/tata/core/LessThanEqualsFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Determine if two objects are less than or equals to two
functions.
 *
 * @version 0.1
 **/
public class LessThanEqualsFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
```

```
	{
		Context context = new Context(ctx);

		Function f1 = script.getFunctionStack().pop();
		Object o1 = f1.eval(script, context);
		Function f2 = script.getFunctionStack().pop();

		Object o2 = f2.eval(script, context);

		context.destroy();

		// null is always less than anything
		if (o1 == null)
		{
			return Boolean.TRUE;
		}


		// if comparable, use the natural ordering
		if (o1 instanceof Comparable)
		{
			return Boolean.valueOf(
(((Comparable) o1).compareTo(o2) <= 0) );
		}

		// otherwise, let's convert both to strings
		String s1 = (String) script.getConverter()
					.convert(o1, String.class,
script);
	String s2 = (String) script.getConverter()
					.convert(o2, String.class,
script);

		if (s1 != null)
		{
			return
Boolean.valueOf(s1.compareTo(s2) <= 0);
		}
		else
		{
			return Boolean.TRUE;
		}
	}
}


[End of: ./src/net/concedere/tata/core/LessThanEqualsFunction.java]


[Contents of: ./src/net/concedere/tata/core/LessThanFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Determine if two functions are less than one another.
 *
 * @version 0.1
 **/
public class LessThanFunction extends AbstractFunction
{
	public Object eval(Script script, Context ctx)
			throws ScriptException
	{
		Context context = new Context(ctx);

		Function f1 = script.getFunctionStack().pop();
		Object o1 = f1.eval(script, context);
```

```
                        Function f2 = script.getFunctionStack().pop();

                        Object o2 = f2.eval(script, context);

                        context.destroy();

                        // null is always less than anything
                        if (o1 == null)
                        {
                                  return Boolean.TRUE;
                        }

                        // if comparable, use the natural ordering
                        if (o1 instanceof Comparable)
                        {
                                  return Boolean.valueOf(
(((Comparable) o1).compareTo(o2) < 0) );
                        }

                        // otherwise, let's convert both to strings
                        String s1 = (String) script.getConverter()
                                                  .convert(o1, String.class,
script);
        String s2 = (String) script.getConverter()
                                                  .convert(o2, String.class,
script);

                        if (s1 != null)
                        {
                                  return
Boolean.valueOf(s1.compareTo(s2) < 0);
                        }
                        else
                        {
                                  return Boolean.TRUE;
                        }
          }
}


[End of: ./src/net/concedere/tata/core/LessThanFunction.java]


[Contents of: ./src/net/concedere/tata/core/ModulusFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Implements the modulus function.
 *
 * @version 0.1
 **/
public class ModulusFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                  throws ScriptException
          {
                        Context context = new Context(ctx);

                        Function f1 = script.getFunctionStack().pop();
        Object o1 = f1.eval(script, context);

                        Function f2 = script.getFunctionStack().pop();
                        Object o2 = f2.eval(script, context);

        Double d1 = (Double) script.getConverter()
                                                  .convert(o1, Double.class,
```

```
script);
                    Double d2 = (Double) script.getConverter()
                                                .convert(o2, Double.class,
script);

        return new Double(d1.doubleValue() % d2.doubleValue());
          }
}


[End of: ./src/net/concedere/tata/core/ModulusFunction.java]


[Contents of: ./src/net/concedere/tata/core/MultiplyFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Multiply two doubles.
 *
 * @version 0.1
 **/
public class MultiplyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object o1 = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);

        Double d1 = (Double) script.getConverter()
                                                .convert(o1, Double.class,
script);
                    Double d2 = (Double) script.getConverter()
                                                .convert(o2, Double.class,
script);

        return new Double(d1.doubleValue() * d2.doubleValue());
          }
}


[End of: ./src/net/concedere/tata/core/MultiplyFunction.java]


[Contents of: ./src/net/concedere/tata/core/NotFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Returns the inversion of a boolean.
 *
 * @version 0.1
 **/
public class NotFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
```

```
                        Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
        Object boj = f1.eval(script, context);

                        Boolean b = (Boolean) script.getConverter()
                                                .convert(boj,
Boolean.class, script);
                        context.destroy();

        if (b.booleanValue())
                        {
                                return Boolean.FALSE;
                        }
                        else
                        {
                                return Boolean.TRUE;
                        }
          }
}


[End of: ./src/net/concedere/tata/core/NotFunction.java]


[Contents of: ./src/net/concedere/tata/core/NullValueFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * Function that always returns null.
 *
 * @version 0.1
 **/
public class NullValueFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                        return null;
          }
}


[End of: ./src/net/concedere/tata/core/NullValueFunction.java]


[Contents of: ./src/net/concedere/tata/core/OrFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Or conditional function.
 *
 * @version 0.1
 **/
public class OrFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
```

```
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object a1 = f1.eval(script, context);
                    Boolean b1 = (Boolean) script.getConverter()
                                        .convert(a1, Boolean.class,
script);

        Function f2 = script.getFunctionStack().pop();
                    Object a2= f2.eval(script, context);
                    Boolean b2 = (Boolean) script.getConverter()
                                        .convert(a2, Boolean.class,
script);

        return new Boolean(b1.booleanValue() || b2.booleanValue());
          }
}


[End of: ./src/net/concedere/tata/core/OrFunction.java]


[Contents of: ./src/net/concedere/tata/core/OverlapFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * Creates the intersection of two collections.
 *
 * @version 0.1
 **/
public class OverlapFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function colF1 =
script.getFunctionStack().pop();
        Object cobj1 = colF1.eval(script, context);
                    Collection co1 = (Collection)
script.getConverter()
                                        .convert(cobj1,
Collection.class, script);

        Function colF2 = script.getFunctionStack().pop();
                    Object cobj2 = colF2.eval(script, context);
                    Collection co2 = (Collection)
script.getConverter()
                                        .convert(cobj2,
Collection.class, script);

        if ((co1 == null) || (co2 == null))
                    {
                              return null;
                    }


                    ArrayList list = new ArrayList(co1);
                    list.retainAll(co2);
                    return list;
          }
```

```
}
```

[End of: ./src/net/concedere/tata/core/OverlapFunction.java]

[Contents of: ./src/net/concedere/tata/core/PrintFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Prints something to the console without printing a trailing
newline.
 *
 * @version 0.1
 **/
public class PrintFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    // construct our own context
          Context context = new Context(ctx);

          // pop the first function off the stack, evalute it in its
own child
                    // context, then print out its result to the
screen
        Function arg = script.getFunctionStack().pop();
                    Object o = arg.eval(script, context);
                    String s = (String) script.getConverter()
                                        .convert(o, String.class,
script);
                    System.out.print(s);

                    context.destroy();
                    // return whatever we printed out
                    return s;
          }
}
```

[End of: ./src/net/concedere/tata/core/PrintFunction.java]

[Contents of: ./src/net/concedere/tata/core/PrintlnFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Prints something out to the system console.
 *
 * @version 0.1
 **/
public class PrintlnFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    // construct our own context
                    Context context = new Context(ctx);

                    // pop the first function off the stack,
evalute it in its own child
```

```
                        // context, then print out its result to the
screen
        Function arg = script.getFunctionStack().pop();
                        Object o = arg.eval(script, context);
                        String s = (String) script.getConverter()
                                                .convert(o, String.class,
script);
                        System.out.println(s);

                        context.destroy();
                        // return whatever we printed out
                        return s;
          }
}


[End of: ./src/net/concedere/tata/core/PrintlnFunction.java]


[Contents of: ./src/net/concedere/tata/core/RemoveFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.List;

/**
 * Remove an element from a function.
 *
 * @version 0.1
 **/
public class RemoveFunction extends AbstractFunction
{
           public Object eval(Script script, Context ctx)
                                throws ScriptException
           {
        Context context = new Context(ctx);

                        Function numFun =
script.getFunctionStack().pop();
                        Object nob = numFun.eval(script, context);
                        Integer i = (Integer) script.getConverter()
                                                .convert(nob,
Integer.class, script);


                        // get the collection to loop over
                        Function colFun =
script.getFunctionStack().pop();
                        Object cobj = colFun.eval(script, context);
                        Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                        if (col == null)
                        {
                                return null;
                        }

        if (col instanceof List)
                        {
                                return ((List)
col).remove(i.intValue());
                        }
                        else
                        {
```

```
                                String msg = "collection is not a
list, remove unpredictable!";
                                throw new ClassCastException(msg);
                        }
                }
}


[End of: ./src/net/concedere/tata/core/RemoveFunction.java]


[Contents of: ./src/net/concedere/tata/core/ReplaceFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.List;
import java.util.Iterator;

/**
 *
 * @version 0.1
 **/
public class ReplaceFunction extends AbstractFunction
{
public Object eval(Script script, Context ctx)
                                throws ScriptException
                {
                        Context context = new Context(ctx);

                        Function numFun =
script.getFunctionStack().pop();
                        Object nob = numFun.eval(script, context);
                        Integer i = (Integer) script.getConverter()
                                                .convert(nob,
Integer.class, script);


                        // get the collection to loop over
                        Function colFun =
script.getFunctionStack().pop();
                        Object cobj = colFun.eval(script, context);
                        Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                        // get the value
                        Function valFun =
script.getFunctionStack().pop();
                        Object val = valFun.eval(script, context);

                        if (col == null)
                        {
                                return null;
                        }

                if (col instanceof List)
                        {
                                return ((List)
col).set((i.intValue()), val);
                        }
                        else
                        {
                        // collection isn't s a list
                                String msg = "not a list";
                                throw new ClassCastException(msg);
```

```
                    }
          }
}

[End of: ./src/net/concedere/tata/core/ReplaceFunction.java]

[Contents of: ./src/net/concedere/tata/core/SetFunction.java]

package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Stores a variable in the context.
 *
 * @version 0.1
 **/
public class SetFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // pop the first argument off the stack, make
it a String
        Function f1 = script.getFunctionStack().pop();
                    Object sobj = f1.eval(script, context);
        String id = (String) script.getConverter()
                                             .convert(sobj,
String.class, script);

        // pop the second argument off the stack, eval it
        Function f2 = script.getFunctionStack().pop();
                    Object var = f2.eval(script, context);

                    // store it in the parent context
                    ctx.set(id, var);

                    context.destroy();
                    // return the value bound
                    return var;
          }
}

[End of: ./src/net/concedere/tata/core/SetFunction.java]

[Contents of: ./src/net/concedere/tata/core/SizeFunction.java]

package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;

/**
 * Returns the number of items in a collection.
 *
 * @version 0.1
 **/
public class SizeFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
```

```
        {
                Context context = new Context(ctx);

                // get the collection to loop over
                Function colFun =
script.getFunctionStack().pop();
                Object cobj = colFun.eval(script, context);
                Collection col = (Collection)
script.getConverter()
                                        .convert(cobj,
Collection.class, script);

                if (col == null)
                {
                        return new Integer(0);
                }
                else
                {
                        return new Integer(col.size());
                }
        }
}


[End of: ./src/net/concedere/tata/core/SizeFunction.java]


[Contents of:
./src/net/concedere/tata/core/StringValueFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Represents a constant String value.
 *
 * @version 0.1
 **/
public class StringValueFunction extends AbstractFunction
{
    private String string;

        /**
         * Construct a new string value function to hold the given
string value.
         *
         * @param s string value to hold
         */
        public StringValueFunction(String s)
        {
                if (s == null)
                {
                        String msg = "s can't be null";
                        throw new
IllegalArgumentException(msg);
                }

        this.string = s;
          }

          public Object eval(Script script, Context ctx) throws
ScriptException
          {
        return string;
          }
}
```

[End of: ./src/net/concedere/tata/core/StringValueFunction.java]

[Contents of: ./src/net/concedere/tata/core/SubRangeFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.List;
import java.util.Iterator;

/**
 * Retrieves a subrange of a collection.
 *
 * @version 0.1
 **/
public class SubRangeFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function numFun =
script.getFunctionStack().pop();
                    Object nob = numFun.eval(script, context);
                    Integer i = (Integer) script.getConverter()
                                        .convert(nob,
Integer.class, script);

                    Function numFun2 =
script.getFunctionStack().pop();
                    Object nob2 = numFun2.eval(script, context);
                    Integer i2 = (Integer) script.getConverter()
                                        .convert(nob2,
Integer.class, script);


                    // get the collection to loop over
                    Function colFun =
script.getFunctionStack().pop();
                    Object cobj = colFun.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                        .convert(cobj,
Collection.class, script);

                    if (col == null)
                    {
                              return null;
                    }

        if (col instanceof List)
                    {
                              return ((List)
col).subList(i.intValue(), i2.intValue());
                    }
                    else
                    {
                              String msg = "collection isn't a
list, subrange doesn't " +
                                                  "make sense!";
                              throw new
IndexOutOfBoundsException(msg);
                    }
          }
}
```

```
[End of: ./src/net/concedere/tata/core/SubRangeFunction.java]


[Contents of: ./src/net/concedere/tata/core/SubtractFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Subtract two numbers.
 *
 * @version 0.1
 **/
public class SubtractFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object o1 = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);

        Double d1 = (Double) script.getConverter()
                                          .convert(o1, Double.class,
script);
                    Double d2 = (Double) script.getConverter()
                                          .convert(o2, Double.class,
script);

        return new Double(d1.doubleValue() - d2.doubleValue());
          }
}


[End of: ./src/net/concedere/tata/core/SubtractFunction.java]


[Contents of: ./src/net/concedere/tata/core/TransformFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * Applies a function to every object in a collection and returns
the
 * result.
 *
 * @version 0.1
 **/
public class TransformFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);
```

```
                        // get the id to bind
        Function idFun = script.getFunctionStack().pop();
                        Object sobj = idFun.eval(script, context);
                        String id = (String) script.getConverter()
                                                .convert(sobj,
String.class, script);

                        // if id is null, it's an error
                        if (id == null)
                        {
                                String msg = "foreach requires non-
null id string for first arg";
                                throw new ScriptException(msg);
                        }

        // get the collection to loop over
                        Function colFun =
script.getFunctionStack().pop();
                        Object cobj = colFun.eval(script, context);
                        Collection col = (Collection)
script.getConverter()
                                                .convert(cobj,
Collection.class, script);

                        // if col is null, short-circuit out
                        if (col == null) return null;

        // get the function block to evaluate!
        FunctionStack body= script.getFunctionStack().popBlock();
                        Script blockScript = new Script(body);
                        blockScript.setConverter(script.getConverter());

                        Collection result = new ArrayList();
                        for (Iterator iter = col.iterator();
iter.hasNext(); )
                        {
            // bind the variable in the context
                                Object val = iter.next();
                                context.set(id, val);

                                // mark the stack if this isn't the
last iteration
                                if (iter.hasNext())
                                {
                                        blockScript.getFunctionStac
k().mark();
                                }

                                // execute the function block
                                Object bo =
body.pop().eval(blockScript, context);
                                result.add(bo);

                                // restore the stack if this isn't
the last iteration!
                                if (iter.hasNext())
                                {
                                        blockScript.getFunctionStac
k().unmark();
                                }
                        }

                        context.destroy();
                        return result;
          }
}


[End of: ./src/net/concedere/tata/core/TransformFunction.java]
```

[Contents of: ./src/net/concedere/tata/core/TrueValueFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * Function that always returns true.
 *
 * @version 0.1
 **/
public class TrueValueFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return Boolean.TRUE;
          }
}
```

[End of: ./src/net/concedere/tata/core/TrueValueFunction.java]

[Contents of: ./src/net/concedere/tata/core/UnionFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * Returns the union of two collections.
 *
 * @version 0.1
 **/
public class UnionFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function colF1 =
script.getFunctionStack().pop();
        Object cobj1 = colF1.eval(script, context);
                    Collection co1 = (Collection)
script.getConverter()
                                                .convert(cobj1,
Collection.class, script);

        Function colF2 = script.getFunctionStack().pop();
                    Object cobj2 = colF2.eval(script, context);
                    Collection co2 = (Collection)
script.getConverter()
                                                .convert(cobj2,
Collection.class, script);

        if (co1 == null)
                    {
                              return co2;
```

```
                    }

                    if (co2 == null)
                    {
                              return co1;
                    }

                    ArrayList list = new ArrayList(co1);
                    for (Iterator i = co2.iterator(); i.hasNext(); )
                    {
                              list.add(i.next());
                    }

                    return list;
          }
}


[End of: ./src/net/concedere/tata/core/UnionFunction.java]


[Contents of: ./src/net/concedere/tata/core/UnlessFunction.java]


package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * An optimized function for representing if statements that only
have an
 * else.
 *
 * @version 0.1
 **/
public class UnlessFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // pop and evaluate the conditional function
                    Function condFunc =
script.getFunctionStack().pop();
                    Object boj = condFunc.eval(script, context);
                    Boolean b = (Boolean)
                                        script.getConverter().conve
rt(boj, Boolean.class, script);

                    FunctionStack condBlock =
script.getFunctionStack().popBlock();
                    Script blockScript = new Script(condBlock);
                    blockScript.setConverter(script.getConverter());

                    // if the boolean is null or it's false, we
execute
                    if (b == null || !b.booleanValue())
                    {
                              condBlock.pop().eval(blockScript,
context);
                    }

                    return b;
          }
}


[End of: ./src/net/concedere/tata/core/UnlessFunction.java]
```

[Contents of: ./src/net/concedere/tata/core/UnsetFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * Remove a variable from the context.
 *
 * @version 0.1
 **/
public class UnsetFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object sobj = f1.eval(script, context);
                    String id = (String) script.getConverter()
                                        .convert(sobj,
String.class, script);

                    Object ret = ctx.get(id);
                    ctx.unset(id);
                    context.destroy();
                    return ret;
          }
}
```

[End of: ./src/net/concedere/tata/core/UnsetFunction.java]

[Contents of: ./src/net/concedere/tata/core/UrlFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

import java.net.URL;
import java.net.MalformedURLException;

/**
 * Returns the value of a URL from a String.
 *
 * @version 0.1
 **/
public class UrlFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object sobj = f1.eval(script, context);
                    String url = (String) script.getConverter()
                                        .convert(sobj,
String.class, script);

                    try
                    {
                              return new URL(url);
                    }
                    catch (MalformedURLException mue)
```

```
                    {
                              String msg = "encountered invalue url
" + url;
                              throw new ScriptException(msg, mue);
                    }
          }
}
```

[End of: ./src/net/concedere/tata/core/UrlFunction.java]

[Contents of: ./src/net/concedere/tata/core/WhenFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
 * WhenFunction is a highly optimized version of the if function
that doesn't
 * have an 'else' block.
 *
 * TODO: you might be curious why there's an if...endif but not so
for
 * when and unless... who the fuck knows?
 *
 * @version 0.1
 **/
public class WhenFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    // pop and evaluate the conditional function
        Function condFunc = script.getFunctionStack().pop();
                    Object boj = condFunc.eval(script, context);
                    Boolean b = (Boolean)
                                        script.getConverter().conve
rt(boj, Boolean.class, script);

                    FunctionStack condBlock =
script.getFunctionStack().popBlock();
        Script blockScript = new Script(condBlock);
                    blockScript.setConverter(script.getConverter());

        if (b != null && b.booleanValue())
                    {
            condBlock.pop().eval(blockScript, context);
                    }

        return b;
          }
}
```

[End of: ./src/net/concedere/tata/core/WhenFunction.java]

[Contents of: ./src/net/concedere/tata/core/WhileFunction.java]

```
package net.concedere.tata.core;

import net.concedere.tata.*;

/**
```

```
 * An implementation of the the while loop in tata. The while
function takes
 * the form: while [condition] [f1] where [condition] is a block
that must
 * be executed continuously and if it evaluates true then [f1] will
be
 * executed.
 *
 * @version 0.1
 **/
public class WhileFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the conditional block
                    FunctionStack conditional =
script.getFunctionStack().popBlock();
                    Script condScript = new Script(conditional);
                    condScript.setConverter(script.getConverter());

                    // get the loop block
                    FunctionStack loop =
script.getFunctionStack().popBlock();
                    Script loopScript = new Script(loop);
                    loopScript.setConverter(script.getConverter());

                    boolean keepRunning = true;

                    // evaluate the conditional
                    condScript.getFunctionStack().mark();
                    Object result =
condScript.getFunctionStack().pop()
                                        .eval(condScript, context);
                    Boolean b = (Boolean) script.getConverter()
                                        .convert(result,
Boolean.class, script);
                    keepRunning = b.booleanValue();
                    condScript.getFunctionStack().unmark();

                    Object retVal = null;
                    while (keepRunning)
                    {
                              // execute the loop
                              loopScript.getFunctionStack().mark();
                              retVal =
loopScript.getFunctionStack().pop()
                                                  .eval(loopScript
, context);
                              loopScript.getFunctionStack().unmark(
);

                              // now we gotta evaluate the
conditional again
                              condScript.getFunctionStack().mark();
                              result =
condScript.getFunctionStack().pop()
                                                  .eval(condScript
, context);
                              b = (Boolean) script.getConverter()
                                                  .convert(result,
Boolean.class, script);
                              keepRunning = b.booleanValue();
                              condScript.getFunctionStack().unmark(
);
                    }

                    context.destroy();
```

```
                        return retVal;
                }
}


[End of: ./src/net/concedere/tata/core/WhileFunction.java]


[Contents of: ./src/net/concedere/tata/Function.java]


package net.concedere.tata;

import java.io.Serializable;

/**
 * Represents the fundamental unit of work in a script.
 *
 * @version 0.1
 **/
public interface Function extends Serializable
{
          /**
           * Ask a function to evaluate itself.
           *
           * @param script script representing the current script
           * @param ctx    context of the parent function!
           * @return value of evaluating the function
           * @throws ScriptException
           *          if something goes wrong
           */
    public Object eval(Script script, Context ctx)
                                throws ScriptException;
}


[End of: ./src/net/concedere/tata/Function.java]


[Contents of: ./src/net/concedere/tata/FunctionStack.java]


package net.concedere.tata;

import java.util.LinkedList;
import java.util.Iterator;

/**
 * A FunctionStack represents a script as it's being executed.
 *
 * @version 0.1
 **/
public class FunctionStack
{
          private LinkedList stack = new LinkedList();
          private LinkedList history = new LinkedList();
          private boolean historyOn = false;

          /**
           * Push a function onto the stack.
           *
           * @param fun Function to push onto the stack
           */
          public void push(Function fun)
          {
                    if (fun == null)
                    {
                              String msg = "fun can't be null";
                              throw new
IllegalArgumentException(msg);
```

```
                }

                stack.addFirst(fun);
        }

        /**
         * Add a function to the bottom of the stack. This is only
used by the
         * parser when constructing function stacks.
         *
         * @param fun Function to add to the bottom of the stack
         */
        public void addLast(Function fun)
        {
                if (fun == null)
                {
                        String msg = "fun can't be null";
                        throw new
IllegalArgumentException(msg);
                }

                stack.addLast(fun);
        }

        /**
         * Pop a Function off the stack.
         *
         * @return Function at the top of the stack
         */
        public Function pop()
        {
                if (stack.isEmpty())
                {
                        String msg = "FunctionStack is
empty!";
                        throw new IllegalStateException(msg);
                }

                Function f = (Function) stack.removeFirst();


                if (historyOn)
                {
                        history.addFirst(f);
                }

                return f;
        }

        /**
         * Retrieve the Function at the top of the stack but don't
remove it.
         *
         * @return Function at the top of the stack
         */
        public Function peek()
        {
                if (stack.isEmpty())
                {
                        String msg = "FunctionStack is
empty";
                        throw new IllegalStateException(msg);
                }
                else
                {
                        return (Function) stack.getFirst();
                }
        }

        /**
```

```
            * Mark the stack. When the stack is reset it's history
will be restored.
            */
           public void mark()
           {
                     historyOn = true;
           }

           /**
            * Unmark the stack by restoring its history.
            */
           public void unmark()
           {
                     // TODO: test this method!
                     for (Iterator iter = history.listIterator();
iter.hasNext(); )
                     {
                               stack.addFirst(iter.next());
                     }
                     history.clear();
                     historyOn = false;
           }

           /**
            * Test if the function stack is empty.
            *
            * @return <code>true</code> iff the stack is empty
            */
           public boolean isEmpty()
           {
                     return stack.isEmpty();
           }

           /**
            * Pop the next logical block off the stack.
            *
            * @return FunctionStack representing the next logical
block
            */
           public FunctionStack popBlock()
           {
                     if (stack.isEmpty())
                     {
                               String msg = "stack is empty";
                               throw new IllegalStateException(msg);
                     }

         FunctionStack retStack = new FunctionStack();

                     // look at the top of the stack, if it's not a
BlockingFunction, just
                     // shove it on and return it
                     if (!(stack.getFirst() instanceof
BlockFunction))
                     {
                               retStack.addLast((Function)
stack.removeFirst());
                               return retStack;
                     }

         LinkedList endStack = new LinkedList();
         Function startBlock = (Function) stack.removeFirst();
         endStack.addFirst(((BlockFunction)
startBlock).getEndFunctionClass());
                     retStack.addLast(startBlock);

         while (!endStack.isEmpty())
                     {
                               Function next = (Function)
stack.removeFirst();
```

```
            retStack.addLast(next);

                                // if this is the end block of our
current block
                                if (next.getClass() ==
endStack.getFirst())
                                {
                                        endStack.removeFirst();
                                }

                                // if this is a blocking function,
add its end stack to the list
                                if (next instanceof BlockFunction)
                                {
                                        endStack.addFirst(
                                                                ((Bloc
kFunction) next).getEndFunctionClass());
                                }
                        }

        return retStack;
          }
}


[End of: ./src/net/concedere/tata/FunctionStack.java]


[Contents of: ./src/net/concedere/tata/http/AuthFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;

/**
 * The AuthFunction authorizes a connection by applying some
credentials to
 * the connection after which requests using that connection will be
 * silently authorized. The first argument should be a
HttpCredentials object
 * while the second argument should be a HttpConnection. It returns
the
 * newly authorized HttpConnection object.
 *
 * @version 0.1
 **/
public class AuthFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
        HttpCredentials creds = (HttpCredentials)
script.getConverter()
                                        .convert(o1,
HttpCredentials.class, script);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(o2,
HTTPConnection.class, script);
        context.destroy();
```

```
                    switch (creds.getType())
                    {
                              case HttpCredentials.BasicAuth:
                                        conn.addBasicAuthorization(
creds.getRealm(),

                                           creds.getUser(),
creds.getPass());
                                        break;
                              case HttpCredentials.DigestAuth:
                                        conn.addDigestAuthorization
(creds.getRealm(),

                                                  creds.getUser(),
creds.getPass());
                                        break;
                              default:
                                        String msg = "detected
unexpected auth type";
                                        throw new
IllegalStateException(msg);
                    }

          return conn;
          }
}


[End of: ./src/net/concedere/tata/http/AuthFunction.java]


[Contents of:
./src/net/concedere/tata/http/CloseConnectionFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;

/**
 * Abort all requests made on a connection and close all sockets
made on
 * connection. It takes a single argument which is the
HTTPConnection to close
 * and returns true if the close is successful else false.
 *
 * TODO: test this function, but how?
 *
 * @version 0.1
 **/
public class CloseConnectionFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object cob = f1.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(cob,
HTTPConnection.class, script);
                    context.destroy();

                    conn.stop();

                    return Boolean.TRUE;
          }
```

```
}
```

[End of: ./src/net/concedere/tata/http/CloseConnectionFunction.java]

[Contents of: ./src/net/concedere/tata/http/ConnectionFunction.java]

```
package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;

/**
 * ConnectionFunction creates an HTTPConnection object from a
hostname and a
 * port. The first argument must be a string indicating the host of
the web
 * server and the second argument must be a port indicating the port
of the
 * server.
 *
 * @version 0.1
 **/
public class ConnectionFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String host = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    Function f2 = script.getFunctionStack().pop();
          Object o2 = f2.eval(script, context);
                    Integer port = (Integer) script.getConverter()
                                        .convert(o2, Integer.class,
script);
                    context.destroy();

                    return new HTTPConnection(host,
port.intValue());

          }
}
```

[End of: ./src/net/concedere/tata/http/ConnectionFunction.java]

[Contents of:
./src/net/concedere/tata/http/CredentialsFunction.java]

```
package net.concedere.tata.http;

import net.concedere.tata.*;

/**
 * The CredentialsFunction creates an HTTP credentials object that
can then be
 * used to authorize requests to a specific resource. It supports
both basic
 * and digest authorization. The function takes three arguments, all
strings.
```

```
 * The first argument must be either "digest" or "basic". The next
three
 * arguments are: realm, username, and password.
 *
 * @version 0.1
 **/
public class CredentialsFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String type = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    String realm = (String) script.getConverter()
                                        .convert(o2, String.class,
script);

        Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    String user = (String) script.getConverter()
                                        .convert(o3, String.class,
script);

                    Function f4 = script.getFunctionStack().pop();
                    Object o4 = f4.eval(script, context);
                    String pass = (String) script.getConverter()
                                        .convert(o4, String.class,
script);

                    int credType = -1;
                    if (type.equalsIgnoreCase("digest"))
                    {
                              credType =
HttpCredentials.DigestAuth;
                    }
                    else if (type.equalsIgnoreCase("basic"))
                    {
                              credType = HttpCredentials.BasicAuth;
                    }
                    else
                    {
                              String msg = "credentials type must
be either 'basic' or 'digest'";
                              throw new ScriptException(msg);
                    }

                    return new HttpCredentials(credType, realm,
user, pass);
          }

}


[End of: ./src/net/concedere/tata/http/CredentialsFunction.java]


[Contents of: ./src/net/concedere/tata/http/DateHeaderFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
```

```
/**
 * This function extracts a header from a HTTP response and returns
it as a
 * date. (Dates must be specially formatted in HTTP in accordance
with
 * RFC 1023). The first argument is a String identifying the name of
the header
 * function while the second argument is a HTTP response object. The
result
 * is a Date object or null.
 *
 * TODO: test this function
 *
 * @version 0.1
 **/
public class DateHeaderFunction extends AbstractFunction
{
    public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object sob = f1.eval(script, context);
                    String header = (String) script.getConverter()
                                          .convert(sob, String.class,
script);

                    Function f2 = script.getFunctionStack().pop();
                    Object rob = f2.eval(script, context);
                    HttpResponse response = (HttpResponse)
script.getConverter()
                                          .convert(rob,
HttpResponse.class, script);
                    context.destroy();

                    return response.getDateHeader(header);
          }
}


[End of: ./src/net/concedere/tata/http/DateHeaderFunction.java]


[Contents of: ./src/net/concedere/tata/http/DeleteFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;

import java.util.Collection;

/**
 * Executes a HTTP DELETE request. The first argument is the path to
the
 * resource to be deleted. The second argument is the HTTPConnection
to use.
 * The third argument is a collection of headers for the request.
The result
 * is a HttpResponse.
 *
 * TODO: test this function
 *
 * @version 0.1
 **/
public class DeleteFunction extends AbstractFunction
```

```
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    // get the path to the resource
                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    // get the HTTPConnection
                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(o2,
HTTPConnection.class, script);

                    // get the headers for the request
        // get the headers for the request
                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    Collection headerParams = (Collection)
script.getConverter()
                                        .convert(o3,
Collection.class, script);
                    NVPair[] headers = null;
                    if (headerParams != null)
                    {
                              headers = (NVPair[]) headerParams
                                                  .toArray(new
NVPair[headerParams.size()]);
                    }

                    context.destroy();

        // execute the delete request
                    try
                    {
                              return new
HttpResponse(conn.Delete(path, headers));
                    }
                    catch (Exception e)
                    {
                              String msg = "delete request failed";
                              throw new ScriptException(msg, e);
                    }
          }
}


[End of: ./src/net/concedere/tata/http/DeleteFunction.java]


[Contents of: ./src/net/concedere/tata/http/GetFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;

import java.util.Collection;

/**
 * Executes a HTTP GET request and returns the response. The first
```

```
argument is
 * a string that represents the path to some resource. The second
argument
 * is a HTTPConnection. The third argument is a an array of NVPair
objects
 * that will be passed on the query string. The last argument is an
array
 * of NVPair objects that will be passed as headers of the request.
The result
 * of the function is a HTTPResponse object.
 *
 * @version 0.1
 **/
public class GetFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(o2,
HTTPConnection.class, script);

                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    Collection queryParams = (Collection)
script.getConverter()
                                        .convert(o3,
Collection.class, script);
                    NVPair[] query = null;
                    if (queryParams != null)
                    {
                              query = (NVPair[]) queryParams
                                                  .toArray(new
NVPair[queryParams.size()]);
                    }

                    Function f4 = script.getFunctionStack().pop();
                    Object o4 = f4.eval(script, context);
                    Collection headerParams = (Collection)
script.getConverter()
                                        .convert(o4,
Collection.class, script);
                    NVPair[] headers = null;
                    if (headerParams != null)
                    {
                              headers = (NVPair[]) headerParams
                                                  .toArray(new
NVPair[headerParams.size()]);
                    }

                    context.destroy();

                    try
                    {
                              return new
HttpResponse(conn.Get(path, query, headers));
                    }
                    catch (Exception e)
                    {
```

```
                                String msg = "get request failed";
                                throw new HttpException(msg, e);
                        }
                }
}


[End of: ./src/net/concedere/tata/http/GetFunction.java]


[Contents of: ./src/net/concedere/tata/http/HeaderFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;

/**
 * HeaderFunction retrieves a HTTP header from a HttpResponse. The
first
 * argument is a String identifying the name of the header. The
second
 * argument is a HttpResponse object returned by some previous
request. This
 * function returns a String.
 *
 * TODO: test this function
 *
 * @version 0.1
 **/
public class HeaderFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object sob = f1.eval(script, context);
                    String header = (String) script.getConverter()
                                                .convert(sob, String.class,
script);

                    Function f2 = script.getFunctionStack().pop();
                    Object rob = f2.eval(script, context);
                    HttpResponse response = (HttpResponse)
script.getConverter()
                                                .convert(rob,
HttpResponse.class, script);
                    context.destroy();

                    return response.getHeader(header);
          }
}


[End of: ./src/net/concedere/tata/http/HeaderFunction.java]


[Contents of: ./src/net/concedere/tata/http/HeadFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;

import java.util.Collection;
```

```
/**
 * Executes a HTTP HEAD function. This function takes three
parameters. The
 * first is a String identifying the path to the resource that will
be
 * targetted. The second is a HTTPConnection. The third is a
collection of
 * NVPairs to passed as the query string. The fourth is a collection
of
 * NVPairs to be passed as headers. The result is a HTTP response.
 *
 * TODO: test this function
 *
 * @version 0.1
 **/
public class HeadFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(o2,
HTTPConnection.class, script);

                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    Collection queryParams = (Collection)
script.getConverter()
                                        .convert(o3,
Collection.class, script);
                    NVPair[] query = null;
                    if (queryParams != null)
                    {
                              query = (NVPair[]) queryParams
                                                  .toArray(new
NVPair[queryParams.size()]);
                    }

                    Function f4 = script.getFunctionStack().pop();
                    Object o4 = f4.eval(script, context);
                    Collection headerParams = (Collection)
script.getConverter()
                                        .convert(o4,
Collection.class, script);
                    NVPair[] headers = null;
                    if (headerParams != null)
                    {
                              headers = (NVPair[]) headerParams
                                                  .toArray(new
NVPair[headerParams.size()]);
                    }

                    context.destroy();

                    try
                    {
                              return new
HttpResponse(conn.Head(path, query, headers));
                    }
```

```
                    catch (Exception e)
                    {
                              String msg = "get request failed";
                              throw new HttpException(msg, e);
                    }
          }
}
```

[End of: ./src/net/concedere/tata/http/HeadFunction.java]

[Contents of: ./src/net/concedere/tata/http/HttpBodyFunction.java]

```
package net.concedere.tata.http;

import net.concedere.tata.*;

/**
 * This function extracts the request data from a HTTP response as
an
 * InputStream. It takes a single argument which is the HTTP
response and
 * returns the InputStream of the request body.
 *
 * @version 0.1
 **/
public class HttpBodyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object rob = f1.eval(script, context);
                    HttpResponse resp = (HttpResponse)
script.getConverter()
                                                  .convert(rob,
HttpResponse.class, script);
                    context.destroy();

                    return resp.getInputStream();
          }
}
```

[End of: ./src/net/concedere/tata/http/HttpBodyFunction.java]

[Contents of: ./src/net/concedere/tata/http/HttpCredentials.java]

```
package net.concedere.tata.http;

/**
 * Represents credentials needed to authorize requests to a
protected
 * resources.
 *
 * @version 0.1
 **/
public class HttpCredentials
{
          /**
           * Constant indicating basic authorization will be used.
           */
    public static final int BasicAuth = 0;
```

```
	/**
	 * Constant indicating that digest authorization will be
used.
	 */
	public static final int DigestAuth = 1;

	private int type;
	private String realm;
	private String user;
	private String pass;

	/**
	 * Construct a new credentials object representing
credentials needed to
	 * access some resource.
	 *
	 * @param type   constant indicating whether basic or
digest auth will be
	 *               used
	 * @param realm  realm that the resource belongs to
	 * @param user   name of the user generating the request
	 * @param pass   password of the user
	 */
	public HttpCredentials(int type, String realm, String
user, String pass)
	{
		if ((realm == null) || (user == null) || (pass
== null))
		{
			String msg = "no parameter can be
null";
			throw new
IllegalArgumentException(msg);
		}

		this.realm = realm;
		this.user = user;
		this.pass = pass;

		if ((type != BasicAuth) & (type != DigestAuth))
		{
	String msg = "type must be one of the publicly defined
constants";
			throw new
IllegalArgumentException(msg);
		}

		this.type = type;
	}

	/**
	 * Get the password of the user.
	 *
	 * @return password of the user
	 */
	public String getPass()
	{
		return pass;
	}

	/**
	 * Get the realm the credentials apply to.
	 *
	 * @return realm that the credentials apply to
	 */
	public String getRealm()
	{
		return realm;
	}
```

```
	/**
	 * Get the type of authentication that should be used.
	 *
	 * @return type of authentication that the credentials
apply to
	 */
	public int getType()
	{
		return type;
	}

	/**
	 * Get a textual description of the authentication method.
	 *
	 * @return textual description of the authentication
method
	 */
	public String getTypeDesc()
	{
		switch (type)
		{
			case BasicAuth:
				return "basic";
			case DigestAuth:
				return "digest";
			default:
				String msg = "invalid type
detected";
				throw new
IllegalStateException(msg);
		}
	}

	/**
	 * Get the name of the user for the credentials.
	 *
	 * @return name of the user for the credentials
	 */
	public String getUser()
	{
		return user;
	}
}

[End of: ./src/net/concedere/tata/http/HttpCredentials.java]

[Contents of: ./src/net/concedere/tata/http/HttpException.java]

package net.concedere.tata.http;

import net.concedere.tata.ScriptException;

/**
 * Represents an exception that occurred within an HTTP function.
 *
 * @version 0.1
 **/
public class HttpException extends ScriptException
{
	public HttpException()
	{
		super();
	}

	public HttpException(String s)
	{
		super(s);
```

```
	}

	public HttpException(String s, Throwable t)
	{
		super(s, t);
	}

	public HttpException(Throwable t)
	{
		super(t);
	}
}


[End of: ./src/net/concedere/tata/http/HttpException.java]


[Contents of: ./src/net/concedere/tata/http/HttpResponse.java]


package net.concedere.tata.http;

import HTTPClient.HTTPResponse;
import HTTPClient.URI;

import java.util.Date;
import java.io.InputStream;
import java.io.ByteArrayInputStream;

/**
 * Represents a response to an HTTP request.
 *
 * @version 0.1
 **/
public class HttpResponse
{
	private HTTPResponse response;

	/**
	 * Construct a new response from the underlying response.
	 *
	 * @param response response containing data
	 */
	public HttpResponse(HTTPResponse response) throws
HttpException
	{
		if (response == null)
		{
			String msg = "response can't be
null";
			throw new
IllegalArgumentException(msg);
		}

		this.response = response;

	// NOTE: here we gotta read in all the response data at once
and save
		// it in memory, this allows the response data
to be read
		// multiple times!
	try
		{
			response.getData();
		}
		catch (Exception e)
		{
			String msg = "response failed";
			throw new HttpException(msg, e);
		}
```

```
	}
public int getStatusCode() throws HttpException
	{
    try
		{
			return response.getStatusCode();
		}
		catch (Exception e)
		{
			String msg = "response failed";
			throw new HttpException(msg, e);
		}
	}

public String getReasonLine() throws HttpException
	{
		try
		{
			return response.getReasonLine();
		}
		catch (Exception e)
		{
			String msg = "response failed";
			throw new HttpException(msg, e);
		}
	}

public String getProtocolVersion() throws HttpException
	{
		try
		{
			return response.getVersion();
		}
		catch (Exception e)
		{
			String msg = "response failed";
			throw new HttpException(msg, e);
		}
	}

public URI getEffectiveUri() throws HttpException
	{
		try
		{
			return response.getEffectiveURI();
		}
		catch (Exception e)
		{
			String msg = "response failed";
			throw new HttpException(msg, e);
		}
	}

	public URI getOriginalUri() throws HttpException
	{
		try
		{
			return response.getOriginalURI();
		}
		catch (Exception e)
		{
			String msg = "response failed";
			throw new HttpException(msg, e);
		}
	}

public String getHeader(String hdr) throws HttpException
	{
    try
```

```
                {
            return response.getHeader(hdr);
                }
                catch (Exception e)
                {
                        String msg = "response failed";
                        throw new HttpException(msg, e);
                }
        }

        public Date getDateHeader(String hdr) throws HttpException
        {
      try
                {
            return response.getHeaderAsDate(hdr);
                }
                catch (Exception e)
                {
                        String msg = "response failed";
                        throw new HttpException(msg, e);
                }
        }

        public InputStream getInputStream() throws HttpException
        {
      try
                {
                        InputStream in =
response.getInputStream();

                        if (in != null)
                        {
                                return in;
                        }
                        else
                        {
                                return new
ByteArrayInputStream(new byte[0]);
                        }
                }
                catch (Exception e)
                {
                        String msg = "response failed";
                        throw new HttpException(msg, e);
                }
        }

    public byte[] getData() throws HttpException
        {
                try
                {
                        return response.getData();
                }
                catch (Exception e)
                {
                        String msg = "response failed";
                        throw new HttpException(msg, e);
                }
        }

    public String getString() throws HttpException
        {
      try
                {
                        return response.getText();
                }
                catch (NullPointerException e)
                {
        // if there is no body then getText throws an NPE so we
gotta catch
```

```
                                // it here
                                return "";
                        }
                        catch (Exception e)
                        {
                                String msg = "response failed";
                                throw new HttpException(msg, e);
                        }
                }
}


[End of: ./src/net/concedere/tata/http/HttpResponse.java]


[Contents of:
./src/net/concedere/tata/http/NvCollectionFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;

import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.ArrayList;

import HTTPClient.NVPair;

/**
 * Takes a list of strings and generations a collection of NvPair
objects that
 * can be used in HTTP requests.
 *
 * @version 0.1
 **/
public class NvCollectionFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object lobj = f1.eval(script, context);
                    Collection col = (Collection)
script.getConverter()
                                                .convert(lobj,
Collection.class, script);

                    List nvPairs = new ArrayList();
        for (Iterator iter = col.iterator(); iter.hasNext(); )
                    {
                              String key = (String) iter.next();
                              String val = (String) iter.next();

                              NVPair nvpair = new NVPair(key, val);
                              nvPairs.add(nvpair);
                    }

                    return nvPairs;
          }
}


[End of: ./src/net/concedere/tata/http/NvCollectionFunction.java]


[Contents of: ./src/net/concedere/tata/http/OptionsFunction.java]
```

```
package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;

import java.io.InputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Collection;

/**
 * Execute an HTTP Options request. The first argument is a String
that
 * identifies the path to the resource. The second argument is a
HTTP
 * Connection. The third argument is an InputStream that should be
passed as
 * the body of the request. The fourth argument is a collection of
headers that
 * should be passed with the request. The result is a HTTP Response.
 *
 * TODO: test this method
 *
 * @version 0.1
 **/
public class OptionsFunction extends AbstractFunction
{
    public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the path to the resource
                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                          .convert(o1, String.class,
script);

                    // get the HTTPConnection
                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                          .convert(o2,
HTTPConnection.class, script);

                    // get the request data
                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    InputStream data = (InputStream)
script.getConverter()
                                          .convert(o3,
InputStream.class, script);

                    // get the headers for the request
                    Function f4 = script.getFunctionStack().pop();
                    Object o4 = f4.eval(script, context);
                    Collection headerParams = (Collection)
script.getConverter()
                                          .convert(o4,
Collection.class, script);
                    NVPair[] headers = null;
                    if (headerParams != null)
                    {
                              headers = (NVPair[]) headerParams
                                                   .toArray(new
```

```
NVPair[headerParams.size()]);
                    }

                    context.destroy();

                    // read the InputStream into a byte array
                    byte[] requestData = null;
                    try
                    {
                              if (data != null)
                              {
                                        ByteArrayOutputStream baos
= new ByteArrayOutputStream();
                                        byte[] buf = new byte[512];
                                        while (true)
                                        {
                                                  int bytesRead =
data.read(buf);
                                                  if (bytesRead ==
-1) break;
                                                  baos.write(buf,
0, bytesRead);
                                        }

                                        requestData =
baos.toByteArray();
                              }
                    }
                    catch (IOException ioe)
                    {
                              String msg = "couldn't buffer request
inputstream";
                              throw new ScriptException(msg, ioe);
                    }

                    // put the request and return the response
                    try
                    {
                              return new
HttpResponse(conn.Options(path, headers, requestData));
                    }
                    catch (Exception e)
                    {
                              String msg = "couldn't execute put
request";
                              throw new ScriptException(msg, e);
                    }
          }
}


[End of: ./src/net/concedere/tata/http/OptionsFunction.java]


[Contents of: ./src/net/concedere/tata/http/PostFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;

import java.io.InputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Collection;

/**
 * Execute a HTTP POST request. The first argument is a String
```

```
identifying the
 * resource to be posted to. The second argument is the
HTTPConnection to use.
 * The third argument is an InputStream to the body of the post. The
fourth
 * argument is a collection of headers. The result is a HTTP
Response.
 *
 * TODO: test this function
 *
 * @version 0.1
 **/
public class PostFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the path to the resource
                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    // get the HTTPConnection
                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(o2,
HTTPConnection.class, script);

                    // get the request data
                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    InputStream data = (InputStream)
script.getConverter()
                                        .convert(o3,
InputStream.class, script);

                    // get the headers for the request
                    Function f4 = script.getFunctionStack().pop();
                    Object o4 = f4.eval(script, context);
                    Collection headerParams = (Collection)
script.getConverter()
                                        .convert(o4,
Collection.class, script);
                    NVPair[] headers = null;
                    if (headerParams != null)
                    {
                              headers = (NVPair[]) headerParams
                                                  .toArray(new
NVPair[headerParams.size()]);
                    }

                    context.destroy();

                    // read the InputStream into a byte array
                    byte[] requestData = null;
                    try
                    {
                              if (data != null)
                              {
                                        ByteArrayOutputStream baos
= new ByteArrayOutputStream();
                                        byte[] buf = new byte[512];
                                        while (true)
                                        {
```

```
                                                int bytesRead =
data.read(buf);
                                                if (bytesRead ==
-1) break;
                                                baos.write(buf,
0, bytesRead);
                                        }
                                        requestData =
baos.toByteArray();
                                }
                        }
                        catch (IOException ioe)
                        {
                                String msg = "couldn't buffer request
inputstream";
                                throw new ScriptException(msg, ioe);
                        }

                        // put the request and return the response
                        try
                        {
                                return new
HttpResponse(conn.Post(path, requestData, headers));
                        }
                        catch (Exception e)
                        {
                                String msg = "couldn't execute put
request";
                                throw new ScriptException(msg, e);
                        }
                }
}


[End of: ./src/net/concedere/tata/http/PostFunction.java]


[Contents of: ./src/net/concedere/tata/http/PutFunction.java]


package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;

import java.io.InputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Collection;

/**
 * Executes a HTTP PUT request. This function takes four arguments.
The first
 * is a String identifying the resource to be put. The second is a
 * HTTPConnection. The third is an InputStream that'll serve as the
body of
 * this request. The fourth is a collection of headers. The result
is a
 * HTTP Response.
 *
 * TODO: test this function!
 *
 * @version 0.1
 **/
public class PutFunction extends AbstractFunction
{
        public Object eval(Script script, Context ctx)
                                throws ScriptException
```

```
        {
                Context context = new Context(ctx);

                // get the path to the resource
                Function f1 = script.getFunctionStack().pop();
                Object o1 = f1.eval(script, context);
                String path = (String) script.getConverter()
                                .convert(o1, String.class,
script);

                // get the HTTPConnection
                Function f2 = script.getFunctionStack().pop();
                Object o2 = f2.eval(script, context);
                HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                .convert(o2,
HTTPConnection.class, script);

                // get the request data
                Function f3 = script.getFunctionStack().pop();
                Object o3 = f3.eval(script, context);
                InputStream data = (InputStream)
script.getConverter()
                                .convert(o3,
InputStream.class, script);

                // get the headers for the request
                Function f4 = script.getFunctionStack().pop();
                Object o4 = f4.eval(script, context);
                Collection headerParams = (Collection)
script.getConverter()
                                .convert(o4,
Collection.class, script);
                NVPair[] headers = null;
                if (headerParams != null)
                {
                        headers = (NVPair[]) headerParams
                                        .toArray(new
NVPair[headerParams.size()]);
                }

                context.destroy();

                // read the InputStream into a byte array
                byte[] requestData = null;
                try
                {
                        if (data != null)
                        {
                                ByteArrayOutputStream baos
= new ByteArrayOutputStream();
                                byte[] buf = new byte[512];
                                while (true)
                                {
                                        int bytesRead =
data.read(buf);
                                        if (bytesRead ==
-1) break;
                                        baos.write(buf,
0, bytesRead);
                                }

                                requestData =
baos.toByteArray();
                        }
                }
                catch (IOException ioe)
                {
                        String msg = "couldn't buffer request
inputstream";
```

```
                                throw new ScriptException(msg, ioe);
                        }

                        // put the request and return the response
                        try
                        {
                                return new
HttpResponse(conn.Put(path, requestData, headers));
                        }
                        catch (Exception e)
                        {
                                String msg = "couldn't execute put
request";
                                throw new ScriptException(msg, e);
                        }
                }
}
```

[End of: ./src/net/concedere/tata/http/PutFunction.java]

[Contents of: ./src/net/concedere/tata/http/StatusCodeFunction.java]

```
package net.concedere.tata.http;

import net.concedere.tata.*;

/**
 * This function extracts the response status code. This function
takes
 * one argument, an HttpResponse and the result is an Integer
indicating te
 * response's status code.
 *
 * @version 0.1
 **/
public class StatusCodeFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object rob = f1.eval(script, context);
                    HttpResponse resp = (HttpResponse)
script.getConverter()
                                                  .convert(rob,
HttpResponse.class, script);

                    context.destroy();

                    return new Integer(resp.getStatusCode());
          }
}
```

[End of: ./src/net/concedere/tata/http/StatusCodeFunction.java]

[Contents of: ./src/net/concedere/tata/http/TraceFunction.java]

```
package net.concedere.tata.http;

import net.concedere.tata.*;
import HTTPClient.HTTPConnection;
import HTTPClient.NVPair;
```

```
import java.util.Collection;

/**
 * Execute an HTTP Trace request. The first argument is a String
identifying
 * the resource to be traced. The second argument is the
HTTPConnection to
 * be used. The third argument is the collection of headers to pass
along
 * with the request.
 *
 * TODO: test this function
 *
 * @version 0.1
 **/
public class TraceFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the path to the resource
                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                        .convert(o1, String.class,
script);

                    // get the HTTPConnection
                    Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    HTTPConnection conn = (HTTPConnection)
script.getConverter()
                                        .convert(o2,
HTTPConnection.class, script);

                    // get the headers for the request
        // get the headers for the request
                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);
                    Collection headerParams = (Collection)
script.getConverter()
                                        .convert(o3,
Collection.class, script);
                    NVPair[] headers = null;
                    if (headerParams != null)
                    {
                              headers = (NVPair[]) headerParams
                                                  .toArray(new
NVPair[headerParams.size()]);
                    }

                    context.destroy();

        // execute the delete request
                    try
                    {
                              return new
HttpResponse(conn.Trace(path, headers));
                    }
                    catch (Exception e)
                    {
                              String msg = "delete request failed";
                              throw new ScriptException(msg, e);
                    }
          }
}
```

[End of: ./src/net/concedere/tata/http/TraceFunction.java]

[Contents of: ./src/net/concedere/tata/io/AppendFunction.java]

```
package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.*;

/**
 * Append data onto the end of the file.
 *
 * @version 0.1
 **/
public class AppendFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object iob = f1.eval(script, context);
                    InputStream in = (InputStream)
script.getConverter()
                                        .convert(iob,
InputStream.class, script);

                    Function f2 = script.getFunctionStack().pop();
                    Object fob = f2.eval(script, context);
                    File file = (File) script.getConverter()
                                        .convert(fob, File.class,
script);
        OutputStream out = null;

                    context.destroy();

                    try
                    {
                              out = new BufferedOutputStream(new
FileOutputStream(file, true));
            IoUtilities.copy(in, out);
                              out.flush();
            return file;
                    }
                    catch (IOException ioe)
                    {
                              String msg = "couldn't copy " + in +
" to " + file;
                              throw new ScriptException(msg, ioe);
                    }
                    finally
                    {
                              IoUtilities.close(in);
                              IoUtilities.close(out);
                    }

          }
}
```

[End of: ./src/net/concedere/tata/io/AppendFunction.java]

[Contents of: ./src/net/concedere/tata/io/CopyFunction.java]

```
package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;

/**
 * Copy an arbitary InputStream into an arbitrary OutputStream.
 *
 * @version 0.1
 **/
public class CopyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object iobj = f1.eval(script, context);
          InputStream in = (InputStream) script.getConverter()
                                                .convert(iobj,
InputStream.class, script);

                    Function f2 = script.getFunctionStack().pop();
                    Object oobj = f2.eval(script, context);
                    OutputStream out = (OutputStream)
script.getConverter()
                                           .convert(oobj,
OutputStream.class, script);
        context.destroy();

        try
                    {
                              IoUtilities.copy(in, out);
                              return oobj;
                    }
                    catch (IOException ioe)
                    {
                              String msg = "failed to copy source
to destination";
                              throw new ScriptException(msg, ioe);
                    }
                    finally
                    {
                              IoUtilities.close(in);
                              IoUtilities.close(out);
                    }
          }
}


[End of: ./src/net/concedere/tata/io/CopyFunction.java]


[Contents of: ./src/net/concedere/tata/io/CreateFunction.java]


package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.File;
import java.io.IOException;

/**
 * Create a file or directory on the local disk.
```

```
 *
 * @version 0.1
 **/
public class CreateFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object fob = f1.eval(script, context);
                    context.destroy();
                    File file = (File) script.getConverter()
                                        .convert(fob, File.class,
script);

                    try
                    {
                              return
Boolean.valueOf(file.createNewFile());
                    }
                    catch (IOException ioe)
                    {
                              String msg = "couldn't create file "
+ file;
                              throw new ScriptException(msg, ioe);
                    }
          }
}


[End of: ./src/net/concedere/tata/io/CreateFunction.java]


[Contents of: ./src/net/concedere/tata/io/DeleteFunction.java]


package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.File;

/**
 * Delete a file or directory on the local disk.
 *
 * @version 0.1
 **/
public class DeleteFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
          Object fob = f1.eval(script, context);
                    context.destroy();

                    File file = (File) script.getConverter()
                                        .convert(fob, File.class,
script);

        return Boolean.valueOf(file.delete());
          }
}


[End of: ./src/net/concedere/tata/io/DeleteFunction.java]
```

[Contents of: ./src/net/concedere/tata/io/DirectoryFunction.java]

```
package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.File;

/**
 * Constructs a file that is a directory.
 *
 * @version 0.1
 **/
public class DirectoryFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object fo = f1.eval(script, context);
                    String path = (String) script.getConverter()
                                        .convert(fo, String.class,
script);
                    context.destroy();
                    File file = new File(path);

                    if (!file.isDirectory())
                    {
                              String msg = "path " + path + " isn't
a directory";
                              throw new
IllegalArgumentException(msg);
                    }

                    return file;
          }
}
```

[End of: ./src/net/concedere/tata/io/DirectoryFunction.java]

[Contents of: ./src/net/concedere/tata/io/FileCopyFunction.java]

```
package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.File;
import java.io.IOException;

/**
 * Copy files and directories around on the local disk. There are
three basic
 * possible situations. If the source is a directory and the
destination is a
 * directory the source directory is copied into the destination
directory
 * as a subdirectory. If the source is a file and the destination is
a
 * directory the source file is copied into the destination. If the
source
 * is a file and the destination is a file than the destination file
is
```

```
 * overwritten.
 *
 * TODO: test this function more thoroughly
 *
 * @version 0.1
 **/
public class FileCopyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object sobj = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object dobj = f2.eval(script, context);

                    context.destroy();

                    File src = (File) script.getConverter()
                                        .convert(sobj, File.class,
script);
                    File dest = (File) script.getConverter()
                                        .convert(dobj, File.class,
script);

                    if (src.isDirectory())
                    {
                              // make sure dest is a directory
                              if (!dest.isDirectory())
                              {
                                        String msg = "if source is
a directory, dest must be a" +
                                                            "
directory";
                                        throw new
ScriptException(msg);
                              }

                              try
                              {
                                        IoUtilities.copyDirToDir(sr
c, dest);
                                        return dest;
                              }
                              catch (IOException ioe)
                              {
                                        String emsg = "couldn't
copy srcDir=" + src + " to " +
                                                            "destD
ir=" + dest;
                                        throw new
ScriptException(emsg, ioe);
                              }
                    }


                    if (dest.isDirectory())
                    {
                              try
                              {
                                        IoUtilities.copyFileToDir(s
rc, dest);
                                        return dest;
                              }
                              catch (IOException ioe)
                              {
                                        String msg = "couldn't copy
```

```
src file=" + src + " to dest dir="
                                                                    +
dest;
                                        throw new
ScriptException(msg, ioe);
                        }
                    }
                    else
                    {
                        try
                        {
                                        IoUtilities.copyFileToFile(
src, dest);
                                        return dest;
                        }
                        catch (IOException ioe)
                        {
                                        String msg = "couldn't copy
source file " + src + " to dest " +
                                                                    "file
" + dest;
                                        throw new
ScriptException(msg, ioe);
                        }
                    }
        }
}

[End of: ./src/net/concedere/tata/io/FileCopyFunction.java]

[Contents of: ./src/net/concedere/tata/io/FileFunction.java]

package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.File;

/**
 * Represents a file on disk.
 *
 * @version 0.1
 **/
public class FileFunction extends AbstractFunction
{
        public Object eval(Script script, Context ctx)
                        throws ScriptException
        {
                Context context = new Context(ctx);

                Function f1 = script.getFunctionStack().pop();
                Object fo = f1.eval(script, context);
                String path = (String) script.getConverter()
                                        .convert(fo, String.class,
script);
                context.destroy();
                File file = new File(path);

                if (file.isDirectory())
                {
                        String msg = "file " + path + " is a
directory!";
                        throw new ScriptException(msg);
                }

                return file;
        }
```

```
}

[End of: ./src/net/concedere/tata/io/FileFunction.java]

[Contents of: ./src/net/concedere/tata/io/IoUtilities.java]

package net.concedere.tata.io;

import java.io.*;

/**
 * Contains several utility functions for working with IO.
 *
 * TODO: test this class
 *
 * @version 0.1
 **/
public class IoUtilities
{
	/**
	 * Unconditionally close an inputstream.
	 *
	 * @param input InpustStream to close
	 */
    public static void close(InputStream input)
	{
		if (input != null)
		{
			try
			{
				input.close();
			}
			catch (Exception e)
			{
				; // squash it
			}
		}
	}

	/**
	 * Unconditionally close an output stream.
	 *
	 * @param output OutputStream to close
	 */
	public static void close(OutputStream output)
	{
		if (output != null)
		{
			try
			{
				output.close();
			}
			catch (Exception e)
			{
				; // squash it
			}
		}
	}

	/**
	 * Copy an InputStream to an OutputStream. This method
does <em>not</em>
	 * close the inputstreams.
	 *
	 * @param input  InputStream to copy
	 * @param output OutputStream to copy to
	 * @throws IOException
```

```
     *          if the copy fails
     */
    public static void copy(InputStream input, OutputStream
output)
                        throws IOException
    {
                if ((input == null) || (output == null))
                {
                        String msg = "no parameter can be
null";
                        throw new
IllegalArgumentException(msg);
                }

                byte[] buffer = new byte[512];
                while (true)
                {
                        int bytesRead = input.read(buffer);
                        if (bytesRead == -1) break;
                        output.write(buffer, 0, bytesRead);
                }
    }

    /**
     * Copy one directory to another directory. The source
directory will be
     * created in the destination directory (or overwritten if
it already
     * exists) and every file in the source directory will be
copied to the
     * new destination directory.
     *
     * @param srcDir  File representing the srcDir
     * @param destDir File representing the destination
directory
     * @throws IOException
     *          if something goes wrong
     */
    public static void copyDirToDir(File srcDir, File destDir)
                        throws IOException
    {
                // check for null
                if ((srcDir == null) || (destDir == null))
                {
                        String msg = "srcdir, destDir cannot
be null";
                        throw new
IllegalArgumentException(msg);
                }

                // make sure they exist
                if (!srcDir.exists() || !destDir.exists())
                {
                        String msg = "both srcDir and destDir
must exist";
                        throw new
IllegalArgumentException(msg);
                }

                // make sure they're directories
        if (!srcDir.isDirectory() || !destDir.isDirectory())
                {
                        String msg = "both srcDir and destDir
must be directories";
                        throw new
IllegalArgumentException(msg);
                }

                // create a new subdirectory in destDir named
after srcDir
```

```
                        File subDir = new File(destDir,
srcDir.getName());

                        // if the directory exists delete it
                        if (subDir.exists()) subDir.delete();

                        // now create the new directory
                        subDir.mkdir();

                        // now loop over every file in the src
directory and copy it to the
                // destination directory
                        File[] files = srcDir.listFiles();
                        File destFile = null;
                        for (int i = 0; i < files.length; ++i)
                        {
                                copyFileToDir(files[i], subDir);
                        }
                }

                /**
                 * Copy a file into a directory. If a file with the same
name already
                 * exists in the target directory, it's deleted.
                 *
                 * @param srcFile file containing data to copy
                 * @param destDir directory to copy the data to
                 * @throws IOException
                 *          if something goes wrong
                 */
                public static void copyFileToDir(File srcFile, File
destDir)
                        throws IOException
                {
                        // check null
                        if (srcFile == null || destDir == null)
                        {
                                String msg = "srcFile, destDir can't
be null";
                                throw new
IllegalArgumentException(msg);
                        }

                        // make sure both exist
                        if (!srcFile.isFile() || !destDir.isDirectory())
                        {
                                String msg = "srcFile, destDir must
both exist and be a file and"
                                                + " directory
respectively";
                                throw new
IllegalArgumentException(msg);
                        }

                        // create a new file in the sub directory
                        File fileCopy = new File(destDir,
srcFile.getName());

                        // if that file exists, delete it
                        if (fileCopy.exists()) fileCopy.delete();

                        // create the file
                        fileCopy.createNewFile();

                        // now copy the srcFile to the new file
                        copyFileToFile(srcFile, fileCopy);
                }

                /**
                 * Copy one file into another file. If the destination
```

```
file doesn't exist
            * then create it.
            *
            * @param srcFile  file containing data to copy
            * @param destFile file to copy the data to
            * @throws IOException
            *         if the copying fails
            */
           public static void copyFileToFile(File srcFile, File
destFile)
                     throws IOException
           {
                     if (srcFile == null || destFile == null)
                     {
                               String msg = "srcFile, destFile can't
be null";
                               throw new
IllegalArgumentException(msg);
                     }

         // make sure srcFile exists and is a file
                     if (!srcFile.isFile())
                     {
                               String msg = "srcFile doesn't exist
or is not a file";
                               throw new
IllegalArgumentException(msg);
                     }

                     // if the destFile exists, make sure it's a file
                     if (destFile.exists() & !destFile.isFile())
                     {
                               String msg = "destFile exists but
isn't a file";
                               throw new
IllegalArgumentException(msg);
                     }

                     // if the destFile doesn't exist, create it
                     if (!destFile.exists())
destFile.createNewFile();

         // copy one file to the other
                     InputStream input = null;
                     OutputStream output = null;
                     try
                     {
                               input = new BufferedInputStream(new
FileInputStream(srcFile));
                               output = new BufferedOutputStream(new
FileOutputStream(destFile));
                               copy(input, output);
                     }
                     finally
                     {
                               close(input);
                               close(output);
                     }
           }
}


[End of: ./src/net/concedere/tata/io/IoUtilities.java]


[Contents of: ./src/net/concedere/tata/io/MakeChildFunction.java]


package net.concedere.tata.io;
```

```
import net.concedere.tata.*;

import java.io.File;

/**
 * Make a file a child of another directory.
 *
 * @version 0.1
 **/
public class MakeChildFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

         Function f1 = script.getFunctionStack().pop();
                    Object fob = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object dob = f2.eval(script, context);

                    context.destroy();

                    File file = (File) script.getConverter()
                                        .convert(fob, File.class,
script);

         File dir = (File) script.getConverter()
                                        .convert(dob, File.class,
script);

                    return new File(dir, file.getName());
          }
}


[End of: ./src/net/concedere/tata/io/MakeChildFunction.java]


[Contents of: ./src/net/concedere/tata/io/MakeDirFunction.java]


package net.concedere.tata.io;

import net.concedere.tata.*;

import java.io.File;
import java.io.IOException;

/**
 * Construct a new directory on the local disk.
 *
 * @version 0.1
 **/
public class MakeDirFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object fob = f1.eval(script, context);
                    context.destroy();
                    File file = (File) script.getConverter()
                                        .convert(fob, File.class,
script);

                    return Boolean.valueOf(file.mkdirs());
```

```
          }
}


[End of: ./src/net/concedere/tata/io/MakeDirFunction.java]


[Contents of:
./src/net/concedere/tata/oo/AssignPropertyFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The assign property function can be used to alter an existing
object
 * rather than attempting to create a new property on an object. It
takes
 * the form: oo.assignp [object] [name] [value].
 *
 * @version 0.1
 **/
public class AssignPropertyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // get the object
                    Function f0 = script.getFunctionStack().pop();
                    Object tobj = f0.eval(script, context);
                    TataObject tob = (TataObject)
script.getConverter()
                                        .convert(tobj,
TataObject.class, script);

                    // get the id
                    Function f1 = script.getFunctionStack().pop();
                    Object idobj = f1.eval(script, context);
                    String id = (String) script.getConverter()
                                        .convert(idobj,
String.class, script);

        // now, we figure out what to replace it with
                    Function valFunc =
script.getFunctionStack().pop();
                    Object valObj = valFunc.eval(script, context);


        // now find the function
                    Context containing =
tob.getContext().findContaining(id);
                    if (containing == null)
                    {
                              String msg = "no such variable as " +
id;
                              throw new ScriptException(msg);
                    }
                    containing.set(id, valObj);


                    context.destroy();
                    return valObj;
          }
}
```

[End of: ./src/net/concedere/tata/oo/AssignPropertyFunction.java]

[Contents of: ./src/net/concedere/tata/oo/BeginObjectFunction.java]

```
package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The BeginObjectFunction is used to create objects. It executes
every
 * function on the stack until it reaches and EndObject function.
 *
 * @version 0.1
 **/
public class BeginObjectFunction extends AbstractFunction
                    implements BlockFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    TataObject object = null;

                    // peek at the function stack and see if its
the 'inherits' function
                    Function inherits =
script.getFunctionStack().peek();

                    // get the object we're inheriting
                    if (inherits.getClass() ==
InheritsFunction.class)
                    {
                              script.getFunctionStack().pop();
                              Function f1 =
script.getFunctionStack().pop();

            Object o1 = f1.eval(script, ctx);
                              TataObject parent = (TataObject)
script.getConverter()
                                                       .convert(o1,
TataObject.class, script);
                              object = new
TataObject(parent.getContext());
                    }
                    else
                    {
                              object = new TataObject();
                    }

                    Context context = object.getContext();


        while (true)
                    {
                              Function f1 =
script.getFunctionStack().pop();
            if (f1.getClass() == EndObjectFunction.class) break;

                              f1.eval(script, context);
                    }

        return object;


          }

          public Class getEndFunctionClass()
          {
```

```
                    return EndObjectFunction.class;
          }
}


[End of: ./src/net/concedere/tata/oo/BeginObjectFunction.java]


[Contents of: ./src/net/concedere/tata/oo/CallFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The CallFunction is used for calling functions that aren't part
of an
 * object. When a function is called (rather than sending an object
some
 * method) it is executed in the context of the caller rather than
the context
 * of the object it potentially belongs to.
 *
 *
 * @version 0.1
 **/
public class CallFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
        Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
        TataFunctionObject tfo = (TataFunctionObject)
script.getConverter()
                                          .convert(o1,
TataFunctionObject.class, script);



                    Object retVal = tfo.evaluate(context);
                    context.destroy();

                    return retVal;

          }
}


[End of: ./src/net/concedere/tata/oo/CallFunction.java]


[Contents of:
./src/net/concedere/tata/oo/CreateFunctionFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * CreateFunctionFunction constructs a TFO object and returns it. It
takes the
 * form: oo.function [block] where [block] is some block of code.
 * It returns a TataFunctionObject object.
 *
```

```
 * @version 0.1
 **/
public class CreateFunctionFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    FunctionStack stack =
script.getFunctionStack().popBlock();
                    return new TataFunctionObject(stack, script);
          }
}


[End of: ./src/net/concedere/tata/oo/CreateFunctionFunction.java]


[Contents of: ./src/net/concedere/tata/oo/EndObjectFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.Function;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * A placeholder function used to mark the end of objects.
 *
 * @version 0.1
 **/
public class EndObjectFunction implements Function
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return null;
          }
}


[End of: ./src/net/concedere/tata/oo/EndObjectFunction.java]


[Contents of: ./src/net/concedere/tata/oo/GetPropertyFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The GetPropertyFunction can be used to retrieve properties from
an object's
 * context. It is of the form: oo.getp [object] [name] where
[object] is an
 * object and [name] is a String that's the name of the property to
retrieve.
 *
 * @version 0.1
 **/
public class GetPropertyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
```

```
                    Object ob = f1.eval(script, context);
                    TataObject tob = (TataObject)
script.getConverter()
                                        .convert(ob,
TataObject.class, script);

                    Function f2 = script.getFunctionStack().pop();
                    Object sob = f2.eval(script, context);
                    String name = (String) script.getConverter()
                                        .convert(sob, String.class,
script);

                    context.destroy();
                    return tob.getContext().get(name);
          }
}
```

[End of: ./src/net/concedere/tata/oo/GetPropertyFunction.java]

[Contents of: ./src/net/concedere/tata/oo/InheritsFunction.java]

```
package net.concedere.tata.oo;

import net.concedere.tata.AbstractFunction;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * A marker function for implementing object inheritance.
 *
 * @version 0.1
 **/
public class InheritsFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    return null;
          }
}
```

[End of: ./src/net/concedere/tata/oo/InheritsFunction.java]

[Contents of: ./src/net/concedere/tata/oo/NewFunction.java]

```
package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The NewFunction allows you to construct new objects from existing
objects
 * by cloning them. It is of the form: oo.new [object] and it
returns a copy
 * of the object.
 *
 * @version 0.1
 **/
public class NewFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
```

```
                    Context context = new Context(ctx);

        Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    TataObject tob = (TataObject)
script.getConverter()
                                            .convert(o1,
TataObject.class, script);

        // now copy the context
                    Context newCtx = (Context)
tob.getContext().clone();
                    context.destroy();

                    return new TataObject(newCtx);
          }
}


[End of: ./src/net/concedere/tata/oo/NewFunction.java]


[Contents of: ./src/net/concedere/tata/oo/SendFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The SendFunction is used to send messages to objects that can
cause
 * functions to be executed on that object. It is of the form:
oo.send [object]
 * [methodName] where [object] is a object and [methodName] is a
string
 * indicating the name of the method to execute. It returns whatever
the
 * execution of that method returns. If the method doesn't exist an
error
 * is thrown.
 * <p />
 * This function guarantees that methods will be executed inside the
context
 * of the object they belong to.
 *
 * @version 0.1
 **/
public class SendFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object o1 = f1.eval(script, context);
                    TataObject tob = (TataObject)
script.getConverter()
                                            .convert(o1,
TataObject.class, script);

        Function f2 = script.getFunctionStack().pop();
                    Object o2 = f2.eval(script, context);
                    String method = (String) script.getConverter()
                                            .convert(o2, String.class,
script);

        // look for the method
                    Object ob = tob.getContext().get(method);
```

```
        if (ob == null)
                        {
                                String msg = "method " + method + "
not defined on object!";
                                throw new ScriptException(msg);
                        }

        if (ob instanceof TataFunctionObject)
                        {
                                // eval the tata method with the
script of the TataFunctionObject and
                                // inside the context of the method
it belongs to
             return ((TataFunctionObject)
ob).evaluate(tob.getContext());
                        }
                        else
                        {
                                String msg = "object " + ob + " isn't
a method!";
                                throw new ScriptException(msg);
                        }


          }
}


[End of: ./src/net/concedere/tata/oo/SendFunction.java]


[Contents of: ./src/net/concedere/tata/oo/SetPropertyFunction.java]


package net.concedere.tata.oo;

import net.concedere.tata.*;

/**
 * The SetPropertyFunction allows you to set variables on an
object's context.
 * It allows you to store objects in an object's context (and even
add
 * functions to the object)! It takes the form: oo.setp [object]
[name] [var].
 * It returns the object set on the object's context.
 *
 * @version 0.1
 **/
public class SetPropertyFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object ob1 = f1.eval(script, context);
                    TataObject tobj = (TataObject)
script.getConverter()
                                        .convert(ob1,
TataObject.class, script);

                    Function f2 = script.getFunctionStack().pop();
                    Object ob2 = f2.eval(script, context);
                    String name = (String) script.getConverter()
                                        .convert(ob2, String.class,
script);
```

```
                    Function f3 = script.getFunctionStack().pop();
                    Object o3 = f3.eval(script, context);

          tobj.getContext().set(name, o3);
                    context.destroy();

                    return o3;
          }
}


[End of: ./src/net/concedere/tata/oo/SetPropertyFunction.java]


[Contents of: ./src/net/concedere/tata/oo/TataFunctionObject.java]


package net.concedere.tata.oo;

import net.concedere.tata.FunctionStack;
import net.concedere.tata.Script;
import net.concedere.tata.Context;
import net.concedere.tata.ScriptException;

/**
 * A method represents some block of code that may be called
multiple times.
 *
 * @version 0.1
 **/
public class TataFunctionObject
{
          private Script script;

          /**
           * Construct a new TataFunctionObject.
           *
           * @param fstack  FunctionStack that represents the method
body
           * @param pScript Parent script of the script creating
this script
           */
          public TataFunctionObject(FunctionStack fstack, Script
pScript)
          {
                    if ((fstack == null) || (pScript == null))
                    {
                              String msg = "no parameter can be
null";
                              throw new
IllegalArgumentException(msg);
                    }

                    this.script = new Script(fstack);
                    this.script.setConverter(pScript.getConverter()
);
          }

          /**
           * Get the sub Script for this TataFunctionObject.
           *
           * @return function stack for this method
           */
          public Script getScript()
          {
                    return script;
          }


          public Object evaluate(Context context) throws
```

```
ScriptException
          {
                    if (context == null)
                    {
                              String msg = "context can't be null";
                              throw new
IllegalArgumentException(msg);
                    }

        // mark the function stack, execute the stack, then unmark
it
                    script.getFunctionStack().mark();
                    Object retVal =
script.getFunctionStack().pop().eval(script, context);
        script.getFunctionStack().unmark();

                    return retVal;
          }
}


[End of: ./src/net/concedere/tata/oo/TataFunctionObject.java]


[Contents of: ./src/net/concedere/tata/oo/TataObject.java]


package net.concedere.tata.oo;

import net.concedere.tata.Context;

/**
 * An TataObject represents some collection of code that is
persistent and may
 * be executed multiple times. An TataObject has its own Context and
methods
 * that can be called within the context.
 *
 * @version 0.1
 **/
public class TataObject
{
          private Context context;

    /**
           * Construct a new object.
           */
          public TataObject()
          {
                    // by default, object contexts are roots
                    context = new Context(null);
          }

          /**
           * Construct a TataObject from an existing context.
           *
           * @param ctx existing context that'll define the state of
this context
           */
          public TataObject(Context ctx)
          {
                    this.context = ctx;
          }

          /**
           * Get the context of the TataObject.
           *
           * @return object of the context
           */
          public Context getContext()
```

```
        {
                return context;
        }
}


[End of: ./src/net/concedere/tata/oo/TataObject.java]


[Contents of: ./src/net/concedere/tata/parse/ConversionParser.java]


package net.concedere.tata.parse;

import net.concedere.tata.convert.Converter;
import net.concedere.tata.util.Utilities;
import net.concedere.tata.ScriptException;

import java.util.Properties;
import java.util.Iterator;

/**
 * Class responsible for parsing the .properties file defining all
the
 * possible conversions.
 *
 * @version 0.1
 **/
public class ConversionParser
{
        /**
         * Parse a properties file defining script conversion and
register each
         * conversion with the given Converter.
         *
         * @param prop      properties file defining the
conversions
         * @param converter converter to register the conversions
with
         * @throws net.concedere.tata.ScriptException
         *          if something goes wrong
         */
        public void parse(Properties prop, Converter converter)
                        throws ScriptException
        {
      if ((prop == null) || (converter == null))
                {
                        String msg = "no parameter can be
null";
                        throw new
IllegalArgumentException(msg);
                }

      // process all the properties
                for (Iterator i = prop.keySet().iterator();
i.hasNext(); )
                {
                        String key = (String) i.next();
                        if (key.startsWith("conversion."))
                        {
                                String conversionClassName
= prop.getProperty(key);
                                // chop off the
'conversion.' part
                                key = key.substring(11);
            String fromClassName = prop.getProperty(key +
".from");
                                String toClassName =
prop.getProperty(key + ".to");
```

```
                                        if (fromClassName == null
|| toClassName == null)
                                        {
                                                String msg =
"conversion " + key + " doesn't define "
                + " from or to field";
                                                throw new
ParserException(msg);
                                        }

                // load the three classes
                                        Class conversionClass =
null;
                                        Class fromClass = null;
                                        Class toClass = null;
                                        try
                                        {
                                                conversionClass
= Utilities.loadClass(conversionClassName);
                                                fromClass =
Utilities.loadClass(fromClassName);
                                                toClass =
Utilities.loadClass(toClassName);
                                        }
                                        catch (Exception e)
                                        {
                                                String msg =
"couldn't load a class";
                                                throw new
ParserException(msg, e);
                                        }

                        // register them
                                        converter.register(fromClas
s, toClass, conversionClass);
                                }
                        }
                }
        }
}
```

[End of: ./src/net/concedere/tata/parse/ConversionParser.java]

[Contents of: ./src/net/concedere/tata/parse/FunctionFactory.java]

```
package net.concedere.tata.parse;

import net.concedere.tata.Function;

/**
 * Object responsible for constructing function instances from
tokens.
 *
 * @version 0.1
 **/
public interface FunctionFactory
{
        /**
         * Construct a new function instance from the given token.
         *
         * @param token token describing the function
         * @return most approrpriate Function instance
         * @throws ParserException
         *          if something goes wrong
         */
        public Function create(String token) throws
```

```
ParserException;
}


[End of: ./src/net/concedere/tata/parse/FunctionFactory.java]


[Contents of: ./src/net/concedere/tata/parse/Parser.java]


package net.concedere.tata.parse;

import net.concedere.tata.FunctionStack;
import net.concedere.tata.Function;
import net.concedere.tata.core.IntegerValueFunction;
import net.concedere.tata.core.DoubleValueFunction;
import net.concedere.tata.core.StringValueFunction;

import java.io.Reader;
import java.io.LineNumberReader;
import java.io.IOException;
import java.util.StringTokenizer;

/**
 * Parser that parses a script and returns a FunctionStack
representing the
 * script.
 *
 * @version 0.1
 **/
public class Parser
{
          /**
           * FunctionFactory used to create functions.
           */
          private FunctionFactory funcFactory;


          /**
           * Construct a new Parser used for parsing a set of
scripts.
           *
           * @param fac factory used to create functions
           */
          public Parser(FunctionFactory fac)
          {
                    if (fac == null)
                    {
                              String msg = "no parameter can be
null";
                              throw new
IllegalArgumentException(msg);
                    }

                    this.funcFactory = fac;
          }

          /**
           * Parse a script and return the FunctionStack
representing the script.
           *
           * @param in Reader containing the script data
           * @return FunctionStack representing the script
           * @throws ParserException
           *          if something goes wrong
           */
          public FunctionStack parse(Reader in) throws
ParserException
          {
        if (in == null)
```

```
                    {
                              String msg = "in can't be null";
                              throw new
IllegalArgumentException(msg);
                    }

                    FunctionStack stack = new FunctionStack();
        LineNumberReader lnr = new LineNumberReader(in);

                    processScript(lnr, stack);
                    return stack;
          }

          /**
           * Process the actual script text.
           *
           * @param lnr   LineNumberReader containing the script
text
           * @param stack FunctionStack being built
           * @throws ParserException
           *          if something goes wrong
           */
    private void processScript(LineNumberReader lnr, FunctionStack
stack)
                    throws ParserException
          {
                    String line = null;
                    String token = null;
                    Function func = null;

nextLine:
                    while (true)
                    {

            try
                              {
                                        line = lnr.readLine();
                              }
                              catch (IOException ioe)
                              {
                                        String msg = "couldn't read
input at line # "
                                                  +
lnr.getLineNumber();
                                        throw new
ParserException(msg, ioe);
                              }

                              // if the end of input break out of
the loop
                              if (line == null) break;

                              // get rid of any whitespace at the
front and back of the line
                              line.trim();

                              // if empty or the first character is
a '#' then skip the line
                              if (line.length() == 0 ||
line.charAt(0) == '#') continue;

            // tokenize the line
                              TokenStack tokenStack = new
TokenStack(line);
                              while (!tokenStack.isEmpty())
                              {
                                        // try to parse the current
token
                                        try
                                        {
```

```
                                                token =
tokenStack.pop();

                                                // if the token
starts with '#' skip the rest of the line
                                                if
(token.length() > 0 && token.charAt(0) == '#')
                                                {
                                                        contin
ue nextLine;
                                                }

                                                if
(isStringLiteral(token))
                                                {
                                // transform the token into a
StringValueFunction and
                                                        //
push it onto the stack
                                func = parseStringLiteral(token,
tokenStack);
                                                }
                                                else if
(isNumberLiteral(token))
                                                {
                                                // transform the
token into a NumberValueFunction and
                                                        //
push it onto the stack
                                func = parseNumberLiteral(token,
tokenStack);
                                                }
                                                else
                                                {
                                                        func =
parseToken(token);
                                                }
                                        }
                                        catch (Exception e)
                                        {
                                                String msg =
"couldn't parse token " + token + " on " +
          "line " + line + " @ " + lnr.getLineNumber();
                                                throw new
ParserException(msg, e);
                                        }

                // push whatever function was built onto the stack
                                        stack.addLast(func);
                                }
                        }
          }

          /**
           * Parse a token that is a number literal. If the token
contains a
           * '.' it is mapped to a DoubleValueFunction otherwise
it's mapped to an
           * IntegerValueFunction.
           *
           * @param token token to test
           * @param stack represents the line the token was found on
           * @return IntegerValueFunction or DoubleValueFunction
depending on the
           *         token text
           */
          private Function parseNumberLiteral(String token,
TokenStack stack)
          {
```

```
                    if (token.indexOf('.') < 0)
                    {
                    Integer i = Integer.valueOf(token);
                              return new IntegerValueFunction(i);
                    }
                    else
                    {
                              Double d = Double.valueOf(token);
                              return new DoubleValueFunction(d);
                    }
          }

          /**
           * Determine if a token is a number literal. This is only
true if the
           * first character in the token is a number or a '.'
period.
           *
           * @param token token to test for number literality
           * @return <code>true</code> iff the token is a number
literal
           **/
          private boolean isNumberLiteral(String token)
          {
                    return Character.isDigit(token.charAt(0)) ||
                                        token.charAt(0) == '.';
          }

          /**
           * Parse a token representing a String literal into a
StringValueFunction.
           * This is done by removing the first and last characters
which're
           * understood to be '"' (double quotation) characters.
           *
           * @param token token to parse
           * @param stack represents the line the token was found on
           * @return StringValueFunction corresponding to
<code>token</code>
           * @throws ParserException
           *         if the string is malformed
           */
          private Function parseStringLiteral(String token,
TokenStack stack)
                    throws ParserException
          {
                    // check for the short-circuit case, a single
word string "world"
        if (token.charAt(0) == '"' & token.charAt(token.length() -
1) == '"')
                    {
                              // check for the space character!
                              if (token.length() == 2) return new
StringValueFunction(" ");

                              token = token.substring(1,
token.length() - 1);
                              return new
StringValueFunction(token);
                    }

                    // otherwise, chip off the leading '"' and look
for the token that
                    // ends it!
        token = token.substring(1, token.length());
        StringBuffer buf = new StringBuffer(token);
                    buf.append(' ');

                    // now we need to loop until we find a token
```

```
that ends with '"
        while (!stack.isEmpty())
                        {
                                token = stack.pop();
                                if (token.charAt(token.length() - 1)
== '"')
                                {
                                        // chop off the last '"'
                                        token = token.substring(0,
token.length() - 1);
                 buf.append(token);
                                        return new
StringValueFunction(buf.toString());
                                }
                                else
                                {
                                        buf.append(token).append('
');
                                }
                        }

        // if we got here, we exhausted the stack looking for the
end of the
                        // string
                        String msg = "found string literal with no end,
" + token;
                        throw new ParserException(msg);
          }

          /**
           * Determine if a token is a String literal. This is true
if the token's
           * first character is a '"' (double quote) character.
           *
           * @param token token to test for string literality
           * @return  <code>true</code> iff the token is a string
literal
           */
          private boolean isStringLiteral(String token)
          {
                    return token.charAt(0) == '"';
          }

          /**
           * Parse a regular token that's potentially prefixed.
           *
           * @param token token to be parsed
           * @return Function corresponding to <code>token</code>
           * @throws ParserException
           *          if something goes wrong
           */
          private Function parseToken(String token) throws
ParserException
          {
                    return funcFactory.create(token);
          }



}


[End of: ./src/net/concedere/tata/parse/Parser.java]


[Contents of: ./src/net/concedere/tata/parse/ParserException.java]


package net.concedere.tata.parse;
```

```
import net.concedere.tata.ScriptException;

/**
 * Represents a generic exception within the parser.
 *
 * @version 0.1
 **/
public class ParserException extends ScriptException
{
          public ParserException()
          {
                    super();
          }

          public ParserException(String s)
          {
                    super(s);
          }

          public ParserException(String s, Throwable t)
          {
                    super(s, t);
          }

          public ParserException(Throwable t)
          {
                    super(t);
          }
}


[End of: ./src/net/concedere/tata/parse/ParserException.java]


[Contents of:
./src/net/concedere/tata/parse/PropertiesFunctionFactory.java]


package net.concedere.tata.parse;

import net.concedere.tata.Function;
import net.concedere.tata.util.Utilities;

import java.util.Properties;

/**
 * Basic implementation of the FunctionFactory interface backed by a
 * properties file.
 *
 * @version 0.1
 **/
public class PropertiesFunctionFactory implements FunctionFactory
{
          private Properties prop;

          /**
           * Construct a new PropertiesFunctionFactory from the
given properties
           * object.
           *
           * @param prop Properties object to convert
           */
          public PropertiesFunctionFactory(Properties prop)
          {
                    if (prop == null)
                    {
                              String msg = "prop can't be null";
                              throw new
IllegalArgumentException(msg);
                    }
```

```
                    this.prop = prop;
          }

          public Function create(String token) throws
ParserException
          {
                    if (!prop.containsKey(token))
                    {
                              String msg = "no such token
definition as " + token;
                              throw new ParserException(msg);
                    }

                    String clazz = prop.getProperty(token);
                    try
                    {
                              return (Function)
Utilities.newInstance(clazz);
                    }
                    catch (Exception e)
                    {
                              String msg = "couldn't instantiate
Function class " + clazz;
                              throw new ParserException(msg, e);
                    }
          }
}


[End of:
./src/net/concedere/tata/parse/PropertiesFunctionFactory.java]


[Contents of: ./src/net/concedere/tata/parse/TokenStack.java]


package net.concedere.tata.parse;

import java.util.*;

/**
 * Utility class used by the parser to represent a line of a script.
 *
 * @version 0.1
 **/
class TokenStack
{
          private LinkedList list = new LinkedList();

          /**
           * Construct a new TokenStack to represent the given
string.
           *
           * @param s that must stack-ized
           */
          public TokenStack(String s)
          {
                    if (s == null)
                    {
                              String msg = "s can't be null";
                              throw new
IllegalArgumentException(msg);
                    }

                    // todo: this doesn't always work! it fails for
" ", it'll treat the
                    // string as two different tokens, '"', and '"'
                    String[] tokens = s.split(" ");
                    String token = null;
```

```
                    for (int i = 0 ; i < tokens.length; ++i)
                    {
                              token = tokens[i].trim();
                              if (token.length() > 0)
list.addLast(token);
                    }
          }

          /**
           * Push a token back onto the stack.
           *
           * @param s string representing the token
           */
          public void push(String s)
          {
        if (s == null)
                    {
                              String msg = "s can't be null";
                              throw new
IllegalArgumentException(msg);
                    }

        list.addFirst(s);
          }

          /**
           * Pop a value off the top of the stack.
           *
           * @return  String token at the top of the stack
           */
          public String pop()
          {
                    if (list.isEmpty())
                    {
                              String msg = "stack is empty";
                              throw new IllegalStateException(msg);
                    }

                    return  (String) list.removeFirst();
          }

          /**
           * Peek at the string at the top of the stack but don't
remove it
           *
           * @return String at the top of the stack or
<code>null</code> if the stack
           *         is empty
           */
    public String peek()
          {
                    if (list.isEmpty())
                    {
                              return null;
                    }
                    else
                    {
                              return (String) list.getFirst();
                    }
          }

          /**
           * Return the number of tokens in the stack.
           *
           * @return number of tokens in the stack
           */
          public int size()
          {
                    return list.size();
          }
```

```
            /**
             * Determine if the stack is emtpy.
             *
             * @return <code>true</code> iff the stack is empty
             */
            public boolean isEmpty()
            {
                        return list.isEmpty();
            }
}


[End of: ./src/net/concedere/tata/parse/TokenStack.java]


[Contents of: ./src/net/concedere/tata/Script.java]


package net.concedere.tata;

import net.concedere.tata.convert.Converter;

/**
 * Represents a script environment in which functions are being
executed.
 *
 * @version 0.1
 **/
public class Script
{
            private Converter converter = new Converter();
            private FunctionStack functionStack;
    private Context rootCtx = new Context(null);

            /**
             * Construct a new Script based on the given Function
stack.
             *
             * @param stack stack representing the function stack
             */
            public Script(FunctionStack stack)
            {
                        if (stack == null)
                        {
                                    String msg = "stack can't be null";
                                    throw new
IllegalArgumentException(msg);
                        }

                        this.functionStack = stack;
            }

            /**
             * Evaluate the script. A script is evaluated by popping
every function
             * off the stack and evaluating it.
             *
             * @return the result of the last function in the script
             * @throws ScriptException
             *         if something goes wrong
             */
            public Object evaluate() throws ScriptException
            {
                        Object result = null;
        while (!functionStack.isEmpty())
                        {
                                    result =
functionStack.pop().eval(this, rootCtx);
                        }
```

```
                return result;
        }

        /**
         * Get the Converter used by this script.
         *
         * @return converter used by this script
         */
        public Converter getConverter()
        {
                return converter;
        }

        /**
         * Set the converter to be used by this script.
         *
         * @param converter converter to be used by this script
         */
        public void setConverter(Converter converter)
        {
                this.converter = converter;
        }

        /**
         * Get the current function stack of this script.
         *
         * @return function stack of the script
         */
        public FunctionStack getFunctionStack()
        {
                return functionStack;
        }

}

[End of: ./src/net/concedere/tata/Script.java]

[Contents of: ./src/net/concedere/tata/ScriptException.java]

package net.concedere.tata;

/**
 * General exception in tata.
 *
 * @version 0.1
 **/
public class ScriptException extends Exception
{
        public ScriptException()
        {
                super();
        }

        public ScriptException(String s)
        {
                super(s);
        }

        public ScriptException(String s, Throwable t)
        {
                super(s, t);
        }

        public ScriptException(Throwable t)
        {
                super(t);
```

```
        }
}


[End of: ./src/net/concedere/tata/ScriptException.java]


[Contents of: ./src/net/concedere/tata/ScriptLauncher.java]


package net.concedere.tata;

import net.concedere.tata.parse.Parser;
import net.concedere.tata.parse.FunctionFactory;
import net.concedere.tata.parse.PropertiesFunctionFactory;
import net.concedere.tata.parse.ConversionParser;

import java.io.Reader;
import java.io.IOException;
import java.io.FileReader;
import java.util.Properties;

/**
 * Utility class for starting scripts.
 *
 * @version 0.1
 **/
public class ScriptLauncher
{
        public static Script create(Reader r) throws
ScriptException
        {
                Properties props = new Properties();
                try
                {
                        props.load(ScriptLauncher.class

.getResourceAsStream("/function-defs.properties"));
                }
                catch (IOException ioe)
                {
                        String msg = "couldn't load function
definitions";
                        throw new ScriptException(msg, ioe);
                }

                FunctionFactory factory = new
PropertiesFunctionFactory(props);
                Parser p = new Parser(factory);

                FunctionStack stack = p.parse(r);
                Script script = new Script(stack);

                // now parse the soncersions
                ConversionParser conversionParser = new
ConversionParser();
                Properties conversionProps = new Properties();
                try
                {
                        conversionProps
                                                .load(ScriptLaun
cher.class

.getResourceAsStream("/conversion.properties"));
                }
                catch (IOException e)
                {
                        String msg = "couldn't load
conversion properties";
                        throw new ScriptException(msg, e);
```

```
                    }
                    conversionParser.parse(conversionProps,
script.getConverter());
                    return script;
          }

          public static void main(String[] args) throws Exception
          {
                    if (args.length < 1)
                    {
                              String msg = "please use 'tata
<script>'";
                              System.out.println(msg);
                              System.exit(-1);
                    }

                    String testFile = args[0];

                    Script script = create(new
FileReader(testFile));
        Object o = script.evaluate();
          }
}


[End of: ./src/net/concedere/tata/ScriptLauncher.java]


[Contents of: ./src/net/concedere/tata/util/Utilities.java]


package net.concedere.tata.util;

import java.io.InputStream;
import java.io.OutputStream;
import java.io.Reader;

/**
 * Common utility methods.
 *
 * @version 0.1
 **/
public class Utilities
{
          /**
           * Close an InputStream and ignore any potential
exceptions.
           *
           * @param in InputStream to close.
           */
          public static void close(InputStream in)
          {
                    if (in != null)
                    {
                              try
                              {
                                        in.close();
                              }
                              catch (Exception e)
                              {
                                        ; // squash it
                              }
                    }
          }

          /**
           * Close a Reader and ignore any potential exceptions.
           *
           * @param r Reader to close.
           */
```

```
	public static void close(Reader r)
	{
		if (r != null)
		{
			try
			{
				r.close();
			}
			catch (Exception e)
			{
				; // squash it
			}
		}
	}

    /**
	 * Close an OutputStream and ignore any potential
exceptions.
	 *
	 * @param out OutputStream to close
	 */
	public static void close(OutputStream out)
	{
		if (out != null)
		{
			try
			{
				out.close();
			}
			catch (Exception e)
			{
				; // squash it
			}
		}
	}

    /**
	 * Construct a new instance of an object from it's fully
qualified
	 * classname. The Class must expose a default constructor
for this to
	 * work.
	 *
	 * @param className fully qualified class name
	 * @return newly constructed instance of the Class
	 * @throws ClassNotFoundException
	 * @throws IllegalAccessException
	 * @throws InstantiationException
	 */
	public static Object newInstance(String className)
			throws ClassNotFoundException,
IllegalAccessException,
				InstantiationException
	{
		Class clazz = null;
		ClassLoader classLoader = null;

		try
		{
			classLoader =
Thread.currentThread().getContextClassLoader();
		}
		catch (Exception e)
		{
			; // squash it
		}
		if (classLoader == null)
		{
			classLoader =
Utilities.class.getClassLoader();
```

```
                    }

                    clazz = classLoader.loadClass(className);
                    return clazz.newInstance();
          }

          /**
           * Load a class from the classpath.
           *
           * @param className
           * @return
           * @throws ClassNotFoundException
           * @throws IllegalAccessException
           * @throws InstantiationException
           */
          public static Class loadClass(String className)
                              throws ClassNotFoundException,
IllegalAccessException,
                              InstantiationException
          {
          Class clazz = null;
                    ClassLoader classLoader = null;

                    try
                    {
                              classLoader =
Thread.currentThread().getContextClassLoader();
                    }
                    catch (Exception e)
                    {
                              ; // squash it
                    }
                    if (classLoader == null)
                    {
                              classLoader =
Utilities.class.getClassLoader();
                    }

                    clazz = classLoader.loadClass(className);
                    return clazz;
          }
}


[End of: ./src/net/concedere/tata/util/Utilities.java]


[Contents of: ./src/net/concedere/tata/xml/DocumentFunction.java]


package net.concedere.tata.xml;

import net.concedere.tata.*;

import java.io.InputStream;

import org.dom4j.io.SAXReader;
import org.dom4j.DocumentException;

/**
 * Parses an inputstream into an XML document.
 *
 * @version 0.1
 **/
public class DocumentFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);
```

```
                    Function f1 = script.getFunctionStack().pop();
                    Object iob = f1.eval(script, context);
                    InputStream in = (InputStream)
script.getConverter()
                                        .convert(iob,
InputStream.class, script);
        context.destroy();

        try
                    {
                              SAXReader reader = new SAXReader();
                              return reader.read(in);
                    }
                    catch (DocumentException de)
                    {
                              String msg = "couldn't parse xml
stream";
                              throw new ScriptException(msg, de);
                    }
          }
}


[End of: ./src/net/concedere/tata/xml/DocumentFunction.java]


[Contents of: ./src/net/concedere/tata/xml/SelectNodesFunction.java]


package net.concedere.tata.xml;

import net.concedere.tata.*;
import org.dom4j.Node;
import org.dom4j.XPath;

/**
 * Applies an XPath selection function to a node and returns a
collection
 * of nodes that are selected. The select function takes two
arguments,
 * an XPath object defining the selection criteria and a Node. The
result is a
 * Collection of nodes that all meet the XPATH criteria.
 *
 * @version 0.1
 **/
public class SelectNodesFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object xobj = f1.eval(script, context);
                    XPath xpath = (XPath) script.getConverter()
                                        .convert(xobj, XPath.class,
script);

        Function f2 = script.getFunctionStack().pop();
                    Object nodeObj = f2.eval(script, context);
                    Node node = (Node) script.getConverter()
                                        .convert(nodeObj,
Node.class, script);
                    context.destroy();

                    return xpath.selectNodes(node);
          }
}
```

[End of: ./src/net/concedere/tata/xml/SelectNodesFunction.java]

[Contents of: ./src/net/concedere/tata/xml/TransformFunction.java]

```
package net.concedere.tata.xml;

import net.concedere.tata.*;

import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.stream.StreamResult;

import org.dom4j.Document;
import org.dom4j.io.DocumentSource;
import java.io.ByteArrayOutputStream;
import java.io.OutputStream;

/**
 * Transform an XML document using an XSLT into an arbitrary
InputStream
 * representing the result of the transformation.
 *
 * @version 0.1
 **/
public class TransformFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    // first argument, a transformer
                    Function f1 = script.getFunctionStack().pop();
                    Object tob = f1.eval(script, context);

                    // second argument, an xml document
                    Function f2 = script.getFunctionStack().pop();
                    Object dob = f2.eval(script, context);

                    // third argument, an outputstream
                    Function f3 = script.getFunctionStack().pop();
                    Object oob = f3.eval(script, context);

                    context.destroy();

                    Transformer tx = (Transformer)
script.getConverter()
                                        .convert(tob,
Transformer.class, script);
                    Document doc = (Document) script.getConverter()
                                        .convert(dob,
Document.class, script);
        OutputStream out = (OutputStream) script.getConverter()
                                        .convert(oob,
OutputStream.class, script);

        DocumentSource source = new DocumentSource(doc);
                    StreamResult result = new StreamResult(out);

                    try
                    {
          tx.transform(source, result);
                              return Boolean.TRUE;
                    }
                    catch (TransformerException te)
                    {
```

```
                                String msg = "couldn't perform
transformation";
                                throw new ScriptException(msg, te);
                        }
                }
}


[End of: ./src/net/concedere/tata/xml/TransformFunction.java]


[Contents of: ./src/net/concedere/tata/xml/ValueOfFunction.java]


package net.concedere.tata.xml;

import net.concedere.tata.*;
import org.dom4j.XPath;
import org.dom4j.Node;

/**
 * Applies an XPATH expression against a Node and returns the value
of the
 * expression as defined by the XPATH spec. The function takes two
arguments;
 * the first is an XPath object that defines the selection criteria
and the
 * second is a Node to which the XPath object is applied. It returns
the value
 * of the XPath expression which is usually either a Number, String
or a
 * Boolean.
 *
 * @version 0.1
 **/
public class ValueOfFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                                throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
        Object xobj = f1.eval(script, context);
                    XPath xpath = (XPath) script.getConverter()
                                        .convert(xobj, XPath.class,
script);

        Function f2 = script.getFunctionStack().pop();
                    Object nodeObj = f2.eval(script, context);
                    Node node = (Node) script.getConverter()
                                        .convert(nodeObj,
Node.class, script);
                    context.destroy();

                    return xpath.valueOf(node);
          }
}


[End of: ./src/net/concedere/tata/xml/ValueOfFunction.java]


[Contents of:
./src/net/concedere/tata/xml/XmlTransformFunction.java]


package net.concedere.tata.xml;

import net.concedere.tata.*;
```

```
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;

import org.dom4j.Document;
import org.dom4j.io.DocumentSource;
import org.dom4j.io.DocumentResult;

/**
 * Perform an XML transformation on a Dom4j Document using a given
Transformer.
 * This function can only be used when the result of the
transformation is
 * also an XML document.
 *
 * @version 0.1
 **/
public class XmlTransformFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object tob = f1.eval(script, context);

                    Function f2 = script.getFunctionStack().pop();
                    Object dob = f2.eval(script, context);

                    context.destroy();

                    Transformer tx = (Transformer)
script.getConverter()
                                        .convert(tob,
Transformer.class, script);
                    Document doc = (Document) script.getConverter()
                                        .convert(dob,
Document.class, script);

        DocumentSource source = new DocumentSource(doc);
                    DocumentResult result = new DocumentResult();

                    try
                    {
          tx.transform(source, result);
                              return result.getDocument();
                    }
                    catch (TransformerException te)
                    {
                              String msg = "couldn't perform
transformation";
                              throw new ScriptException(msg, te);
                    }
          }
}


[End of: ./src/net/concedere/tata/xml/XmlTransformFunction.java]


[Contents of: ./src/net/concedere/tata/xml/XpathFunction.java]


package net.concedere.tata.xml;

import net.concedere.tata.*;
import org.dom4j.DocumentFactory;
import org.dom4j.XPath;
```

```
/**
 * An XpathFunction creates an XPath object that can be used to
select nodes
 * and values from some XML node. An XPath function takes two
arguments, an
 * XPATH expression as a string and a Map containing prefix->URI
bindings for
 * the namespace.
 *
 * TODO: currently this object doesn't handle prefix->URI bindings.
in the
 * future, when Maps are implemented in the core language, this will
be
 * implemented.
 *
 * @version 0.1
 **/
public class XpathFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
          {
                    Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object sobj = f1.eval(script, context);
                    String xpath = (String) script.getConverter()
                                        .convert(sobj,
String.class, script);

                    // todo: get a map containing prefix->URI
namespace bindings

        context.destroy();

                    // create the xpath and return it
                    XPath ret =
DocumentFactory.getInstance().createXPath(xpath);

                    return ret;
          }
}


[End of: ./src/net/concedere/tata/xml/XpathFunction.java]


[Contents of: ./src/net/concedere/tata/xml/XsltFunction.java]


package net.concedere.tata.xml;

import net.concedere.tata.*;

import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.stream.StreamSource;
import java.io.InputStream;

/**
 * Converts an InputStream into a Transformer that can be used to
transform
 * XML documents.
 *
 * @version 0.1
 **/
public class XsltFunction extends AbstractFunction
{
          public Object eval(Script script, Context ctx)
                              throws ScriptException
```

```
          {
        Context context = new Context(ctx);

                    Function f1 = script.getFunctionStack().pop();
                    Object iob = f1.eval(script, context);
                    InputStream in = (InputStream)
script.getConverter()
                                          .convert(iob,
InputStream.class, script);
        context.destroy();

                    try
                    {
            TransformerFactory factory =
TransformerFactory.newInstance();
            return factory.newTransformer(new StreamSource(in));
                    }
                    catch (TransformerConfigurationException tce)
                    {
                              String msg = "couldn't convert
inputstream to transformer";
                              throw new ScriptException(msg, tce);
                    }
          }
}


[End of: ./src/net/concedere/tata/xml/XsltFunction.java]
```

APPENDIX B

Testing Source Code

```
[Contents of: ./test/core/add-test.tata]


# Modified:
# 4.21.2003

# This is a test of the add function

println add 1 1

set "total" 0
foreach "num" ( 1 2 3 4 5 6 )
{
          assign "total" add get "total" get "num"
}
println "total is: "
print get "total"

[End of: ./test/core/add-test.tata]


[Contents of: ./test/core/and-test.tata]


# Modified:.
# 4.22.2003


# test of the and function

if and true true
{
          println "who is EL GREGOR?"
}
endif

if and true false
{
          println "EL GREGOR doesn't exist...he is only a myth"
}
endif

[End of: ./test/core/and-test.tata]


[Contents of: ./test/core/assign-test.tata]


# Modified:.
# 4.21.2003

# test of the assign function
set "test" 0

{
          assign "test" 1
}

print "test was assigned to "
println get "test"

[End of: ./test/core/assign-test.tata]
```

```
[Contents of: ./test/core/block-test.tata]


# Modified:.
# 4.20.2003

# test of the block function

set "result"
          {
                    println "Hello World!"
                    println "Goodbye Cruel World!"
                    println "Stand aside if you fear the eternal
darkness!"
                    7
          }
print get "result"

[End of: ./test/core/block-test.tata]


[Contents of: ./test/core/boolean-test.tata]


# Modified:.
# 4.20.2003

# test of the boolean functions

println true

println false

[End of: ./test/core/boolean-test.tata]


[Contents of: ./test/core/concat-test.tata]


# Modified:.
# 4.22.2003

# test of the concatenation function

println & "I " & "Ain't " & "Dying " "Today "

println ( 1 2 3 4 5 6 )

[End of: ./test/core/concat-test.tata]


[Contents of: ./test/core/count-test.tata]


# Modified:.
# 4.22.2003

# test of the count function

set "result"
          count "x" ( 1 2 3 4 5 6 7 )
          {
                    eq mod get "x" 2 0
          }

println get "result"

# to get the number of elements,in a collection just always return
true
```

```
set "size"
          count "x" ( 1 2 3 4 5 6 7 8 9 0 ) true

println get "size"

[End of: ./test/core/count-test.tata]


[Contents of: ./test/core/diff-test.tata]


# Modified:.
# 4.22.2003

# test of the diff function

set "nums1" ( 1 2 3 4 5 6 7 8 9 0 )

set "nums2" ( 3 4 5 )

println diff get "nums1" get "nums2"

[End of: ./test/core/diff-test.tata]


[Contents of: ./test/core/div-test.tata]


# Modified:.
# 4.21.2003

# test of the division function

println div 49 7

[End of: ./test/core/div-test.tata]


[Contents of: ./test/core/eq-test.tata]


# Modified:.
# 4.20.2003

# Test of the equals function

set "t" eq 7 7
println get "t"

[End of: ./test/core/eq-test.tata]


[Contents of: ./test/core/filter_out-test.tata]


# Modified:
# 4.22.2003

# test of the filter_out function

set "odds"
          filter_out "x" ( 1 2 3 4 5 6 )
          {
                    eq mod get "x" 2 0
          }

println get "odds"

[End of: ./test/core/filter_out-test.tata]
```

[Contents of: ./test/core/filter-test.tata]

```
# Modified:.
# 4.21.2003


# test of the filter function


set "1" 1
set "1" add get "1" 2
println concat "1 is " get "1"

set concat "ste" "vens"
          filter "x" ( 1 2 3 4 5 6 )
          {
                    eq mod get "x" 2 1
          }

set "listOfList" ( ( 1 2 3 ) ( 4 5 6 ) )

foreach "list" get "listOfList"
{
          transform "x" get "list"
          {
                    if eq "x" 2
                    {
                              mul get "x
                    }
          }

}

println get "listOfList"

# println get "stevens"
```

[End of: ./test/core/filter-test.tata]

[Contents of: ./test/core/find_last-test.tata]

```
# Modified:.
# 4.22.2003

# test of the find_last function

set "result"
          find_last "x" ( 1 8 2 3 10 5 6 7 )
          {
                    gt get "x" 7
          }

println get "result"
```

[End of: ./test/core/find_last-test.tata]

[Contents of: ./test/core/find-test.tata]

```
# Modified:.
# 4.21.2003

# Test of the find function

set "result"
```

```
          find "x" ( 1 2 3 4 5 )
          {
                    eq get "x" 4
          }

println get "result"
```

[End of: ./test/core/find-test.tata]

[Contents of: ./test/core/foreach-test.tata]

```
# Modified:
# 4.20.2003

# test of the foreach function

foreach "str" ( "Hello" "World" "Who's" "A" "Bad" "Man?" )
{
          print get "str"
          print "_"
}

set "total" 0
foreach "num" ( 1 2 3 4 5 6 7 8 9 )
{
          set "total" get "total"
          print get "num"
}
println "this is the end"
println get "total"

# now check this, we loop over an empty collection!
# BEWARE THE EL GREGOR
foreach "num" ( )
          { println "if you see this, you're already dead" }
```

[End of: ./test/core/foreach-test.tata]

[Contents of: ./test/core/get-test.tata]

```
# Modified:.
# 4.20.2003

# test of the get function

set "num" 7

# get from the root context
get "num"

# get from the parent context
println get "num"
```

[End of: ./test/core/get-test.tata]

[Contents of: ./test/core/gte-test.tata]

```
# Modified:.
# 4.20.2003

# Test of the gte function

println gte 7 8
```

```
println gte 7 7

println gte 8 7

[End of: ./test/core/gte-test.tata]


[Contents of: ./test/core/gt-test.tata]


# Modified:
# 4.20.2003


# test of the gt function
println gt 7 7

println gt 7 8

println gt 8 7

[End of: ./test/core/gt-test.tata]


[Contents of: ./test/core/if-test.tata]


# Modified:.
# 4.21.2003

# test of the 'if' function

# basic if else
if true
{
          println "conditional was true"
}
else
{
          println "conditional was false"
}
endif

# looky here, an if without an else and a complex conditional!
if not false
{
          println "conditional was true"
}
endif

# pay attention to what is going on here! the body of an if or else
statement
# must be a block! a block must either be a SINGLE function or a
blocking
# function. 'println "this won't get printed"' is TWO functions!
TWO! IT MUST
# BE BLOCKED. This is a common mistake, so learn it!
# BEWARE THE EL GREGOR

if true
{
          if false
                    { println "this won't get printed" }
          else
                    { println "looky here, nested if statements!
WHO IS THE EL GREGOR?!" }
          endif
}
else
```

```
{
          println "if you see this, you are fucked"
}
endif

# now we gotta do the if ... else if, note the double endif
statements!
if false
{
          println "this won't show"
}
else
          if true
          {
                    println "EL GREGOR, please print this"
          }
          endif
endif
```

[End of: ./test/core/if-test.tata]

[Contents of: ./test/core/index-test.tata]

```
# Modified:.
# 4.22.2003


# test of the index function

set "3"
          index 2 ( 1 2 3 )

println get "3"
```

[End of: ./test/core/index-test.tata]

[Contents of: ./test/core/list-test.tata]

```
# Modified:
# 4.20.2003

# test of the list function

set "numList" ( 1 2 3 4 5 6 7 )
print get "numList"
```

[End of: ./test/core/list-test.tata]

[Contents of: ./test/core/lte-test.tata]

```
# Modified:.
# 4.20.2003

# Test of the less than equals function

println lte 7 7 # true
println lte 7 8 # true
println lte 8 7 # false
```

[End of: ./test/core/lte-test.tata]

[Contents of: ./test/core/lt-test.tata]

```
# Modified:.
# 4.20.2003

# Test of the less than function

println lt 7 7  # false

println lt 7 8  # true

println lt 8 7  # false
```

[End of: ./test/core/lt-test.tata]

[Contents of: ./test/core/mod-test.tata]

```
# Modified:.
# 4.21.2003


# test of the modulus function

println mod 12 5
```

[End of: ./test/core/mod-test.tata]

[Contents of: ./test/core/mul-test.tata]

```
# Modified:.
# 4.21.2003

# test of the multiply function

println mul mul 4 3 mul 6 2
```

[End of: ./test/core/mul-test.tata]

[Contents of: ./test/core/not-test.tata]

```
# Modified:
# 4.20.2003

# Test of the not function
println not true
```

[End of: ./test/core/not-test.tata]

[Contents of: ./test/core/null-test.tata]

```
# Modified:
# 4.20.2003

# test of the null function

set "test" null
println get "test"
println null
```

[End of: ./test/core/null-test.tata]

[Contents of: ./test/core/or-test.tata]

```
# Modified:.
# 4.22.2003

# test of the or function

if or true true
{
          println "where is EL GREGOR?"
}
endif

if or false true
{
          println "he is behind you"
}
endif

if or false false
{
          println "shit."
}
endif
```

[End of: ./test/core/or-test.tata]

[Contents of: ./test/core/overlap-test.tata]

```
# Modified:.
# 4.22.2003


# test of the overlap function

set "nums1" ( 1 2 3 4 5 6 7 )

set "nums2" ( 4 5 8 )

println overlap get "nums1" get "nums2"
```

[End of: ./test/core/overlap-test.tata]

[Contents of: ./test/core/println-test.tata]

```
# Modified:.
# 4.19.2003

# Test of the println function

# print a string
println "Hello World!"

# print an integer
println 7

# print a double
println 12.5
```

[End of: ./test/core/println-test.tata]

[Contents of: ./test/core/print-test.tata]

```
# Modified:.
```

```
# 4.19.2003

# Test of the print function

# print a string
print "Hello World!"

# print an integer
print 7

# print a double
print 12.5

[End of: ./test/core/print-test.tata]


[Contents of: ./test/core/rem-test.tata]


# Modified:
# 4.22.2003


# test of the rem function

set "nums" ( 1 2 3 4 5 6 )

set "3"
          rem 2 get "nums"

println  get "3"

println get "nums"

[End of: ./test/core/rem-test.tata]


[Contents of: ./test/core/replace-test.tata]


# Modified:.
# 4.23.2003


# test of the replace function

set "nums" ( 1 2 3 4 5 6 )

println get "nums"

replace 1 get "nums" 3

println get "nums"

set "rlist" ( ( ( 3 ) ( 4 ) ) 5 6 )

println get "rlist"

set "newList" ( 1 1 )

set "var" replace 0 index 1 index 0 get "rlist" get "newList"

println get "rlist"




[End of: ./test/core/replace-test.tata]
```

[Contents of: ./test/core/set-test.tata]

```
# Modified:.
# 4.20.03

# test of the set function

print set "true" true
```

[End of: ./test/core/set-test.tata]

[Contents of: ./test/core/size-test.tata]

```
# Modified:.
# 4.22.2003

# test of the size function

set "10"
          size ( 1 2 3 4 5 6 7 8 9 10 )

println get "10"

set "0"
          size ( )

println get "0"
```

[End of: ./test/core/size-test.tata]

[Contents of: ./test/core/subrange-test.tata]

```
# Modified:.
# 4.22.2003

# test of the subrange function
set "nums" ( 1 2 3 4 5 6 7 8 9 0 )

println subrange 1 4 get "nums"  # should print 2-4
```

[End of: ./test/core/subrange-test.tata]

[Contents of: ./test/core/sub-test.tata]

```
# Modified:.
# 4.21.2003

# test of the subtract function

set "alot" 100
set "alittle" 1

println sub get "alot" get "alittle"
```

[End of: ./test/core/sub-test.tata]

[Contents of: ./test/core/transform-test.tata]

```
# Modified:.
# 4.21.2003

# test of the transform function
set "doubles"
          transform "x" ( 1 2 3 4 5 )
          {
                    mul get "x" 2
          }

println get "doubles"
```

[End of: ./test/core/transform-test.tata]

[Contents of: ./test/core/union-test.tata]

```
# Modified:.
# 4.22.2003

# test of the union function

set "nums" ( 1 2 3 4 5 )
set "nums2" ( 6 7 8 9 0 )

println union get "nums" get "nums2"
```

[End of: ./test/core/union-test.tata]

[Contents of: ./test/core/unless-test.tata]

```
# Modified:.
# 4.20.2003

# test of the unless function


unless false
{
          println "i am the terror dome"
}

unless true
{
          println "the EL GREGOR got you fool!"
}
```

[End of: ./test/core/unless-test.tata]

[Contents of: ./test/core/unset-test.tata]

```
# Modified:.
# 4.20.2003

# test of the unset function

set "num" 7

println get "num"

unset "num"

println get "num"
```

```
[End of: ./test/core/unset-test.tata]


[Contents of: ./test/core/url-test.tata]


# Modified:.
# 4.20.2003

# test of the url function

set "yahoo.com" url "http://www.yahoo.com"

println get "yahoo.com"

[End of: ./test/core/url-test.tata]


[Contents of: ./test/core/when-test.tata]


# Modified:.
# 4.21.2003

# Test of the when function

when true
{
          println "he is the shadow of death"
}

when false
{
          println "if you see this, you're already dead"
}

when true
{
          when true
          {
                    println "check me, check this, check me check
this"
          }
}

[End of: ./test/core/when-test.tata]


[Contents of: ./test/core/while-test.tata]


# Modified:.
# 5.7.03

# test of the while function

set "x" 0.0

# note that you gotta put the conditional inside a block!
while { gt get "x" 0.0 }
{
          println concat "x is: " get "x"
          assign "x" sub get "x" 1
}

[End of: ./test/core/while-test.tata]
```

```
[Contents of: ./test/http/body-test.tata]


# Modified:.
# 5.7.03

# test of the http body function

set "yahoo.com" http.body
          http.get "/index.html" http.conn "www.yahoo.com" 80 null
null

io.copy get "yahoo.com" io.file "yahoo-data.txt"



[End of: ./test/http/body-test.tata]

[Contents of: ./test/http/conn-test.tata]


# Modified:
# 5.6.03

# test of the http.conn function

set "yahooConn" http.conn "www.yahoo.com" 80

println get "yahooConn"

[End of: ./test/http/conn-test.tata]

[Contents of: ./test/http/creds-test.tata]


# Modified:.
# 5/6/03

# test of the creds function

set "creds" http.creds "digest" "nixerverse" "buko" "buko"

println get "creds"

[End of: ./test/http/creds-test.tata]

[Contents of: ./test/http/get-test.tata]


# Modified:
# 5.7.03

# test of the http.get function

set "yahoo" http.conn "www.yahoo.com" 80

set "response" http.get "/index.html" get "yahoo" null null

println get "response"

[End of: ./test/http/get-test.tata]

[Contents of: ./test/http/nvcoll-test.tata]
```

```
# Modified:
# 5/6/03

# test of the nvcoll function

set "nvs" http.nvcoll ( "key1" "val1" "key2" "val2" )

println get "nvs"
```

[End of: ./test/http/nvcoll-test.tata]

[Contents of: ./test/http/statuscode-test.tata]

```
# Modified:
# 5.7.03

# test of the http.statuscode function

println http.status http.get "/index.html" http.conn "www.yahoo.com"
80 null null
```

[End of: ./test/http/statuscode-test.tata]

[Contents of: ./test/io/append-test.tata]

```
# Modified:.
# 4.24.2003

# test of the append function

io.append "Hello World" io.file "test/io/append-test.tata"
```

[End of: ./test/io/append-test.tata]

[Contents of: ./test/io/copy-test.tata]

```
# Modified:.
# 4.24.2003

# test of the io.copy function

set "out" io.file "new-copy.tata"

io.copy io.file "test/io/copy-test.tata" get "out"
```

[End of: ./test/io/copy-test.tata]

[Contents of: ./test/io/create-test.tata]

```
# Modified:.
# 4.24.2003

# test of the io.create function

if io.create "create-test.tata"
{
          println "created the file!"
}
endif
```

[End of: ./test/io/create-test.tata]

[Contents of: ./test/io/delete-test.tata]

```
# Modified:.
# 4.24.2003

# test of the delete function

set "file" io.file "new-copy.tata"

if io.del get "file"
{
          println concat "successfully deleted " get "file"
}
endif
```

[End of: ./test/io/delete-test.tata]

[Contents of: ./test/io/dir-test.tata]

```
# Modified:.
# 4.24.2003

# test of the directory function

set "cDir" io.dir "C:\"

println get "cDir"
```

[End of: ./test/io/dir-test.tata]

[Contents of: ./test/io/fcopy-test.tata]

```
# Modified:.
# 4.24.2003

# test of the fcopy function

io.fcopy
          {
                    io.create "test.txt"
                    io.file "test.txt"
          }
          {
                    io.mkdir "test-out"
                    io.dir "test-out"
          }
```

[End of: ./test/io/fcopy-test.tata]

[Contents of: ./test/io/file-test.tata]

```
# Modified:.
# 4.23.2003

# test of the file function
```

```
set "myFile" io.file "C:\ditd\java\tata\test\io\file-test.tata"

println get "myFile"


[End of: ./test/io/file-test.tata]


[Contents of: ./test/io/mkchild-test.tata]


# Modified:.
# 4.24.2003

# test of the mkchild function

set "file" io.file "test-file"

println io.mkchild get "file"
          {
                    io.mkdir "test-dir"
                    io.dir "test-dir"
          }


[End of: ./test/io/mkchild-test.tata]


[Contents of: ./test/io/mkdir-test.tata]


# Modified:.
# 4.24.2003

# test of the make dir function

if io.mkdir "mkdir-test"
{
          println "created the directory!"
}
endif

[End of: ./test/io/mkdir-test.tata]


[Contents of: ./test/newsreader.tata]


# Modified:
# 5.7.03

# a simple newsreader of RSS 2.0 weblogs

# set the weblogs we're going to read
set "blogroll"
          (
                    url "http://radio.weblogs.com/0106046/rss.xml"
# weakliem
        url "http://www.scripting.com/rss.xml" # winer
        url "http://nyt.weblogs.com/rss.xml" # new york times
frontpage
        url "http://partners.userland.com/nytRss/technology.xml" #
nyt technology
        url "http://partners.userland.com/nytRss/arts.xml" # nyt
arts
          )

# set the xslt we use to do rss2txt
set "rss2txt" xml.xslt io.file "test/rss2txt.xsl"
```

```
# create the file we're going to append the docs to
set "news.txt" io.file "news.txt"
io.create get "news.txt"

# tmp file to store each one
set "tmp.txt" io.file "tmp.txt"
io.create get "tmp.txt"

foreach "feed" get "blogroll"
{
          xml.transform get "rss2txt" xml.doc get "feed" get
"tmp.txt"
          io.append get "tmp.txt" get "news.txt"
}


[End of: ./test/newsreader.tata]


[Contents of: ./test/oo/assignp-test.tata]


# Modified:
# 5.7.03

# test of the assignp function

set "obj"
          oo.object
                    set "var" "var1"
          oo.endobject

oo.assignp get "obj" "var" "var1-OWNZORED"

println oo.getp get "obj" "var"


[End of: ./test/oo/assignp-test.tata]


[Contents of: ./test/oo/call-test.tata]


# Modified:.
# 5.7.03

# test of the call function

set "helloFunc"
          oo.function
          {
                    println get "greeting"
          }

set "greeting" "WHERE IS EL GREGOR?!"

oo.call get "helloFunc"

[End of: ./test/oo/call-test.tata]


[Contents of: ./test/oo/getp-test.tata]


# Modified:.
# 5.7.03

# test of the getp function
set "myObj"
```

```
          oo.object
                    set "var" "helloWorld"
          oo.endobject

println oo.getp get "myObj" "var"

[End of: ./test/oo/getp-test.tata]


[Contents of: ./test/oo/inherits-test.tata]


# Modified:.
# 5.7.03

# test of object inheritance

set "greeter"
          oo.object
                    set "greeting" "Hello! How are you today?"

                    set "doGreet"
                              oo.function
                              {
                                        println get "greeting"
                              }
          oo.endobject

set "policeman"
          oo.object oo.inherits get "greeter"
                    set "freeze" "Freeze right there cocksucker!"

                    set "doFreeze"
                              oo.function
                              {
                                        println get "freeze"
                              }
          oo.endobject

oo.send get "policeman" "doFreeze"
oo.send get "policeman" "doGreet"


[End of: ./test/oo/inherits-test.tata]


[Contents of: ./test/oo/new-test.tata]


# Modified:.
# 5.7.03

# test of the new function

set "greeter"
          oo.object
                    set "greeting" "hello from greeter"

                    set "doGreet"
                              oo.function
                              {
                                        println get "greeting"
                              }
          oo.endobject

oo.send get "greeter" "doGreet"

# now make a new greeter and change stuff around
set "greeter2" oo.new get "greeter"
```

```
oo.setp get "greeter2" "greeting" "hello from greeter2"

oo.send get "greeter2" "doGreet"

# now print the greeter
oo.send get "greeter" "doGreet"



[End of: ./test/oo/new-test.tata]


[Contents of: ./test/oo/send-test.tata]


# Modified:.
# 5.7.03

# test of the send function

set "person"
          oo.object
                    set "greeting" "Hello!!"

                    set "sayGreeting" oo.function
                              {
                                        println get "greeting"
                              }
          oo.endobject

oo.send get "person" "sayGreeting"

[End of: ./test/oo/send-test.tata]


[Contents of: ./test/oo/setp-test.tata]


# Modified:
# 5.7.03

# test of the setp function

set "obj"
          oo.object
                    set "greeting" "this shouldn't be printed"

                    set "helloWorld"
                              oo.function
                              {
                                        println get "greeting"
                              }
          oo.endobject

oo.setp get "obj" "greeting" "hello world!"

oo.send get "obj" "helloWorld"

[End of: ./test/oo/setp-test.tata]


[Contents of: ./test/pointer-test.tata]


# Modified:

# test of pointers

set "a" 3
set "b" get "a"
```

```
assign "b" 4

println concat "a is " get "a"

[End of: ./test/pointer-test.tata]


[Contents of: ./test/rss2txt.xsl]


<?xml version="1.0" ?>
<xsl:stylesheet
          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
          xmlns:dc="http://purl.org/dc/elements/1.1/" version="1.0">

  <xsl:output method="text"/>

  <xsl:template match="*[local-name()='title']">
    <xsl:text>title: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="*[local-name()='link']">
    <xsl:text>link: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="*[local-name()='description']">
    <xsl:text>description: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="dc:creator">
    <xsl:text>author: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="dc:date">
    <xsl:text>date: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="language"/>  <!-- suppress -->

</xsl:stylesheet>



[End of: ./test/rss2txt.xsl]


[Contents of: ./test/xml/doc-test.tata]


# Modified:.
# 4.25.2003


# test of the xml.doc function

set "testXml" xml.doc "<?xml version="1.0"?><root><elem>This is a
test</elem></root>"
println get "testXml"

[End of: ./test/xml/doc-test.tata]


[Contents of: ./test/xml/select-test.tata]
```

```
# Modified:.
# 5/5/03

# test of the select function

set "doc" xml.doc url "http://www.scripting.com/rss.xml"

set "itemNodes" xml.select xml.xpath "//item/description" get "doc"

foreach "node" get "itemNodes"
{
          println get "node"
}
```

[End of: ./test/xml/select-test.tata]

[Contents of: ./test/xml/transform-test.tata]

```
# Modified:.
# 4.25.2003

# test of the transform function

if xml.transform
                     xml.xslt
                                io.file "test\xml-test\rss2txt.xsl"
                     xml.doc io.file "test\xml-
test\scriptingnews_rss.xml"
                     io.file "test\xml-test\scriptingnews.txt"
{
          println "boo yah baybee!"
}
endif
```

[End of: ./test/xml/transform-test.tata]

[Contents of: ./test/xml/valueof-test.tata]

```
# Modified:.
# 5/6/03

# test of the xml.valueof function

set "doc" xml.doc url "http://www.scripting.com/rss.xml"

set "value" xml.valueof xml.xpath "/rss[@version]" get "doc"

println get "value"
```

[End of: ./test/xml/valueof-test.tata]

[Contents of: ./test/xml/xpath-test.tata]

```
# Modified:
# 5/6/03

# test of the xpath function

set "xpath" xml.xpath "//item"

println concat "xpath is " get "xpath"
```

```
[End of: ./test/xml/xpath-test.tata]


[Contents of: ./test/xml/xslt-test.tata]


# Modified:.
# 4.25.2003

# test of the xml.xslt function

set "rss2txt" xml.xslt io.file "test\xml-test\rss2txt.xsl"



[End of: ./test/xml/xslt-test.tata]


[Contents of: ./test/xml/xtransform-test.tata]


# Modified:.
# 4.25.2003

# test of the xml-transform function

println xml.xtransform
          xml.xslt io.file "test\xml-test\rss2rdf.xsl"
          xml.doc io.file "test\xml-test\scriptingnews_rss.xml"

[End of: ./test/xml/xtransform-test.tata]


[Contents of: ./test/xml-test/rss2rdf.xsl]


<?xml version="1.0"?>
<xsl:transform
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:r="http://backend.userland.com/rss2"
  xmlns="http://purl.org/rss/1.0/"
  version="1.0">

<xsl:output indent="yes" cdata-section-elements="content:encoded" />


<!-- general element conversions -->

<xsl:template match="/">
  <rdf:RDF>
    <xsl:apply-templates/>
  </rdf:RDF>
</xsl:template>

<xsl:template match="*">
  <xsl:choose>
    <xsl:when test="namespace-uri()='' or namespace-
uri()='http://backend.userland.com/rss2'">
      <xsl:element name="{name()}"
namespace="http://purl.org/rss/1.0/">
        <xsl:apply-templates select="*|@*|text()" />
      </xsl:element>
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy>
        <xsl:apply-templates select="*|@*|text()" />
      </xsl:copy>
```

```
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="@*">
  <!--<xsl:copy><xsl:value-of select="." /></xsl:copy>-->
</xsl:template>


<xsl:template match="text()">
  <xsl:value-of select="normalize-space(.)" />
</xsl:template>

<xsl:template match="rss|r:rss">
  <xsl:apply-templates select="channel|r:channel" />
  <xsl:apply-templates
select="channel/item[guid|link]|r:channel/r:item[r:guid|r:link]"
mode="rdfitem" />
</xsl:template>

<xsl:template match="channel|r:channel">
  <channel rdf:about="{link|r:link}">
    <xsl:apply-templates/>
    <items>
      <rdf:Seq>
        <xsl:apply-templates select="item|r:item" mode="li" />
      </rdf:Seq>
    </items>
  </channel>
</xsl:template>


<!-- channel content conversions -->

<xsl:template match="title|r:title">
  <dc:title><xsl:value-of select="." /></dc:title>
</xsl:template>

<xsl:template match="description|r:description">
  <dc:description><xsl:value-of select="." /></dc:description>
</xsl:template>

<xsl:template match="language|r:language">
  <dc:language><xsl:value-of select="." /></dc:language>
</xsl:template>

<xsl:template match="copyright|r:copyright">
  <dc:rights><xsl:value-of select="." /></dc:rights>
</xsl:template>

<xsl:template
match="lastBuildDate|pubdate|r:lastBuildDate|r:pubdate">
  <dc:date><xsl:call-template name="date" /></dc:date>
</xsl:template>

<xsl:template match="managingEditor|r:managingEditor">
  <dc:creator><xsl:value-of select="." /></dc:creator>
</xsl:template>

<!-- elements from 0.94 not converted:
          webMaster
          category
          generator
          docs
          cloud
          ttl
          image
          textInput
          skipHours
          skipDays
```

```
-->

<!-- item content conversions -->

<xsl:template match="item/description|r:item/r:description">
  <dc:description><xsl:call-template name="removeTags"
/></dc:description>
  <xsl:if test="not(../content:encoded)">
    <content:encoded><xsl:value-of select="." /></content:encoded>
  </xsl:if>
</xsl:template>

<xsl:template match="pubDate|r:pubDate">
  <dc:date><xsl:call-template name="date" /></dc:date>
</xsl:template>

<xsl:template match="source|r:source">
  <dc:source><xsl:value-of select="@url" /></dc:source>
</xsl:template>

<xsl:template match="author|r:author">
  <dc:creator><xsl:value-of select="." /></dc:creator>
</xsl:template>

<!-- elements from 0.94 not converted:
          category
          comments
          enclosure
-->

<!-- item templates -->

<xsl:template match="item|r:item" mode="li">
  <xsl:choose>
    <xsl:when test="link|r:link">
      <rdf:li rdf:resource="{link|r:link}" />
    </xsl:when>
    <xsl:when test="guid|r:guid">
      <rdf:li rdf:resource="{guid|r:guid}" />
    </xsl:when>
    <xsl:otherwise>
      <rdf:li rdf:parseType="Resource">
        <xsl:apply-templates />
      </rdf:li>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="item[link]|r:item[r:link]" mode="rdfitem">
  <item rdf:about="{link|r:link}">
    <xsl:apply-templates/>
  </item>
</xsl:template>

<xsl:template
match="item[guid][not(link)]|r:item[r:guid][not(r:link)]"
mode="rdfitem">
  <item rdf:about="{r:guid|guid}">
    <xsl:apply-templates/>
  </item>
</xsl:template>


<!-- utility templates -->

<xsl:template match="channel/link|r:channel/r:link" />
<xsl:template match="channel/item|r:channel/r:item" />
<xsl:template match="item/guid|r:item/r:guid" />
<xsl:template match="item/link|r:item/r:link" />
```

```
<xsl:template name="date">
  <xsl:variable name="m" select="substring(., 9, 3)" />
  <xsl:value-of select="substring(., 13, 4)"
  />-<xsl:choose>
    <xsl:when test="$m='Jan'">01</xsl:when>
    <xsl:when test="$m='Feb'">02</xsl:when>
    <xsl:when test="$m='Mar'">03</xsl:when>
    <xsl:when test="$m='Apr'">04</xsl:when>
    <xsl:when test="$m='May'">05</xsl:when>
    <xsl:when test="$m='Jun'">06</xsl:when>
    <xsl:when test="$m='Jul'">07</xsl:when>
    <xsl:when test="$m='Aug'">08</xsl:when>
    <xsl:when test="$m='Sep'">09</xsl:when>
    <xsl:when test="$m='Oct'">10</xsl:when>
    <xsl:when test="$m='Nov'">11</xsl:when>
    <xsl:when test="$m='Dec'">12</xsl:when>
    <xsl:otherwise>00</xsl:otherwise>
  </xsl:choose>-<xsl:value-of select="substring(., 6, 2)"
  />T<xsl:value-of select="substring(., 18, 8)" />
</xsl:template>

<xsl:template name="removeTags">
  <xsl:param name="html" select="." />
  <xsl:choose>
    <xsl:when test="contains($html,'&lt;')">
      <xsl:call-template name="removeEntities">
        <xsl:with-param name="html" select="substring-
before($html,'&lt;')" />
      </xsl:call-template>
      <xsl:call-template name="removeTags">
        <xsl:with-param name="html" select="substring-after($html,
'&gt;')" />
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="removeEntities">
        <xsl:with-param name="html" select="$html" />
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template name="removeEntities">
  <xsl:param name="html" select="." />
  <xsl:choose>
    <xsl:when test="contains($html,'&amp;')">
      <xsl:value-of select="substring-before($html,'&amp;')" />
      <xsl:variable name="c" select="substring-before(substring-
after($html,'&amp;'),';')" />
      <xsl:choose>
        <xsl:when test="$c='nbsp'"> </xsl:when>
        <xsl:when test="$c='lt'">&lt;</xsl:when>
        <xsl:when test="$c='gt'">&gt;</xsl:when>
        <xsl:when test="$c='amp'">&amp;</xsl:when>
        <xsl:when test="$c='quot'">&quot;</xsl:when>
        <xsl:when test="$c='apos'">&apos;</xsl:when>
        <xsl:otherwise>?</xsl:otherwise>
      </xsl:choose>
      <xsl:call-template name="removeTags">
        <xsl:with-param name="html" select="substring-after($html,
';')" />
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$html" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

</xsl:transform>
```

[End of: ./test/xml-test/rss2rdf.xsl]

[Contents of: ./test/xml-test/rss2txt.xsl]

```
<?xml version="1.0" ?>
<xsl:stylesheet
          xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
          xmlns:dc="http://purl.org/dc/elements/1.1/" version="1.0">

  <xsl:output method="text"/>

  <xsl:template match="*[local-name()='title']">
    <xsl:text>title: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="*[local-name()='link']">
    <xsl:text>link: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="*[local-name()='description']">
    <xsl:text>description: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="dc:creator">
    <xsl:text>author: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="dc:date">
    <xsl:text>date: </xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="language"/>  <!-- suppress -->

</xsl:stylesheet>
```

[End of: ./test/xml-test/rss2txt.xsl]

[Contents of: ./test/xml-test/scriptingnews.txt]

```
                    title: Scripting News
                    link: http://www.scripting.com/
                    description: A weblog about scripting and stuff
like that.

                    http://radio.weblogs.com/0001015/userland/scrip
tingNewsLeftLinks.opml
                    http://radio.weblogs.com/0001015/gems/mySubscri
ptions.opml
                    Copyright 1997-2003 Dave Winer
                    Fri, 25 Apr 2003 14:00:00 GMT
                    http://backend.userland.com/rss
                    Radio UserLand v8.0.5
                    1765
                    rssUpdates
                    dave@userland.com
                    dave@userland.com
```

8
9
10
11
12
4
5
6
7

15

description: News.Com: "A US district court on Thursday ruled for a second time that Verizon Communications must give up the identity of an anonymous Internet subscriber accused of swapping music files online."
Fri, 25 Apr 2003 10:36:23 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:6:36:23AM

description: David Carter-Tod: Manila Express for News Items.
Fri, 25 Apr 2003 09:48:43 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:5:48:43AM

description: Animated demo of David's tool, above.
Fri, 25 Apr 2003 10:57:18 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:6:57:18AM

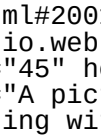
description: MacNN has upgraded to RSS 2.0.
Fri, 25 Apr 2003 11:08:13 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:7:08:13AM

description: More testing of outbound trackback in Manila. If you have a MT site for me to test with please post a comment with a pointer to the site. Thanks.
Fri, 25 Apr 2003 13:00:45 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:9:00:45AM

description: Adam Curry: "Geeks, nerds, programmers and developers often complain they feel misunderstood in corporate and other social circles. It flows both ways guys."
Fri, 25 Apr 2003 10:40:49 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:6:40:49AM

description: The Register reports that the Chronicle has fired tech columnist Henry Norr. I've known Henry for 20 years. An exceptionally intelligent and honest analyst. What a loss for the Chronicle and the tech industry.

Fri, 25 Apr 2003 10:33:09 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#When:6:33:09AM

title: Three hot topics
link: http://scriptingnews.userland.com/backissues/2003/04/25#threeHotTopics
description:
Report from last night's session. We did the usual hour's worth of software demos and then switched over to three topics that are much on my mind:
  1. How to integrate blogging with radio (not Radio). We talked about this at length. As other people talked I realized it is not about technology. There is no magic formula that will make the two worlds connect. Chris says let's hear what they think, and I say let's see. Two different senses, one visual and cerebral, and the other auditory and soulful. The answer, as it often is, is people. A roundtable of intelligent bloggers, like Washington Week in Review, or The Capital Gang, but staffed by writers who work in blogspace, and done on the radio, once a week. Chris is our anchor. I want to do this at BloggerCon, and every week, starting asap.
2. Blogging and the New Hampshire primary. Citizen bloggers covering the candidates for US president, follow it where it goes. First step -- clearly -- go to NH myself and find some candidates. Luckily I'm speaking at Dartmouth on May 9. I'll go looking for presidential hopefuls. With my camera and some questions. I'll try to explain weblogs. And here's another way to proceed. Are there any people in NH reading this site who think weblogs could make a difference? We need a citizen's committee for evangelizing the concept. Everyone who hears it goes Hmm, that might work.
3. BloggerCon, otherwise known as Weblogs in Meatspace. In October a conference in Cambridge about weblogs as writer's medium, from a historic and technological perspective. Computer industry conferences have done a great job of the latter, at best a superficial job at the former. I want librarians, lawyers, historians, executives, musicians, producers, pundits, scholars, educators, personalities, politicians, and more.
BTW, Philip Greenspun came to last night's meeting. I expect he will write about this discussion as well. In fact, during the discussion he made some notes on his blog using my keyboard and mouse. As always the discussion was informal, and the minds alive and interesting.

Fri, 25 Apr 2003 10:03:57 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#threeHotTopics

title: This pig won't fly
link: http://scriptingnews.userland.com/backissues/2003/04/25#thisPigWontFly
description:
Social Software? I've

been in the software biz for 2.5 decades, so I've seen this kind of hype over and over. Take something that exists, give it a fancy new name, and then blast at reporters and analysts about it. Every time around the loop it works less well. In the 80s it worked very well. In the early 21st Century, there aren't enough analysts with credibility to make such a pig fly.

P2P was the last gasp. I remember getting breathless invitations to keynotes where this or that luminary was going to finally tell us what it is. In the end it wasn't the technology that made a difference, but ironically, the people. Apparently the promoters of Social Software were listening.

It's wrong. We don't need this. Weblogs are about punching through the hype machine of idiot analysts and reporters who go for their BS. Social software has existed for years. What's the big news? A few people are looking for a pole to fly their flag on. Pfui!

Fri, 25 Apr 2003 10:01:53 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#thisPigWontFly

title: Poopy little wiener boys
link: http://scriptingnews.userland.com/backissues/2003/04/25#poopyLittleWienerBoys
description:

On this day three years ago Dan Gillmor was being pecked at by poopy little wiener boys up past their bedtime. "Western civilization is in jeopardy," Dan said. "And it's all my fault."

Fri, 25 Apr 2003 10:31:31 GMT
http://scriptingnews.userland.com/backissues/2003/04/25#poopyLittleWienerBoys

[End of: ./test/xml-test/scriptingnews.txt]

[Contents of: ./test/xml-test/scriptingnews_rss.xml]

<?xml version="1.0"?>
<!-- RSS generated by Radio UserLand v8.0.5 on 4/25/2003; 10:00:00 AM Eastern -->
<rss version="2.0" xmlns:blogChannel="http://backend.userland.com/blogChannelModule">
<channel>
<title>Scripting News</title>
<link>http://www.scripting.com/</link>
<description>A weblog about scripting and stuff like that.</description>
<language>en-us</language>
<blogChannel:blogRoll>http://radio.weblogs.com/0001015/userland/scriptingNewsLeftLinks.opml</blogChannel:blogRoll>
<blogChannel:mySubscriptions>http://radio.weblogs.com/0001015/gems/mySubscriptions.opml</blogChannel:mySubscriptions>
<copyright>Copyright 1997-2003 Dave Winer</copyright>
<lastBuildDate>Fri, 25 Apr 2003 14:00:00 GMT</lastBuildDate>

```
                    <docs>http://backend.userland.com/rss</docs>
                    <generator>Radio UserLand v8.0.5</generator>
                    <category domain="Syndic8">1765</category>
                    <category
domain="http://www.weblogs.com/rssUpdates/changes.xml">rssUpdates</c
ategory>
                    <managingEditor>dave@userland.com</managingEdit
or>
                    <webMaster>dave@userland.com</webMaster>
                    <skipHours><!-- hours are GMT -->
                              <hour>8</hour>
                              <hour>9</hour>
                              <hour>10</hour>
                              <hour>11</hour>
                              <hour>12</hour>
                              <hour>4</hour>
                              <hour>5</hour>
                              <hour>6</hour>
                              <hour>7</hour>
                              </skipHours>
                    <ttl>15</ttl>
                    <item>
                              <description>&lt;a
href=&quot;http://news.com.com/2100-1027-
998268.html&quot;&gt;News.Com&lt;/a&gt;: &quot;A US district court
on Thursday ruled for a second time that Verizon Communications must
give up the identity of an anonymous Internet subscriber accused of
swapping music files online.&quot;</description>
                              <pubDate>Fri, 25 Apr 2003 10:36:23
GMT</pubDate>
                              <guid>http://scriptingnews.userland.c
om/backissues/2003/04/25#When:6:36:23AM</guid>
                              </item>
                    <item>
                              <description>David Carter-Tod: &lt;a
href=&quot;http://www.wcc.vccs.edu/dtod/frontier/manilaexpressfornew
sitems.asp&quot;&gt;Manila Express for News
Items&lt;/a&gt;.</description>
                              <pubDate>Fri, 25 Apr 2003 09:48:43
GMT</pubDate>
                              <guid>http://scriptingnews.userland.c
om/backissues/2003/04/25#When:5:48:43AM</guid>
                              </item>
                    <item>
                              <description>Animated &lt;a
href=&quot;http://manilaexpress.userland.com/stories/storyReader$318
&quot;&gt;demo&lt;/a&gt; of David's tool, above.</description>
                              <pubDate>Fri, 25 Apr 2003 10:57:18
GMT</pubDate>
                              <guid>http://scriptingnews.userland.c
om/backissues/2003/04/25#When:6:57:18AM</guid>
                              </item>
                    <item>
                              <description>MacNN has &lt;a
href=&quot;http://www.macnn.com/print/19252&quot;&gt;upgraded&lt;/a&
gt; to RSS 2.0.</description>
                              <pubDate>Fri, 25 Apr 2003 11:08:13
GMT</pubDate>
                              <guid>http://scriptingnews.userland.c
om/backissues/2003/04/25#When:7:08:13AM</guid>
                              </item>
                    <item>
                              <description>More testing of &lt;a
href=&quot;http://blogs.law.harvard.edu/crimson1/2003/04/25#a134&quo
t;&gt;outbound trackback&lt;/a&gt; in Manila. If you have a MT site
for me to test with please post a comment with a pointer to the
site. Thanks.</description>
                              <pubDate>Fri, 25 Apr 2003 13:00:45
GMT</pubDate>
                              <guid>http://scriptingnews.userland.c
```

om/backissues/2003/04/25#When:9:00:45AM</guid>
</item>
<item>
<description>Adam Curry: "Geeks, nerds, programmers and developers often complain they feel misunderstood in corporate and other social circles. It flows both ways guys." </description>
<pubDate>Fri, 25 Apr 2003 10:40:49 GMT</pubDate>
<guid>http://scriptingnews.userland.com/backissues/2003/04/25#When:6:40:49AM</guid>
</item>
<item>
<description>The Register reports that the Chronicle has fired tech columnist Henry Norr. I've known Henry for 20 years. An exceptionally intelligent and honest analyst. What a loss for the Chronicle and the tech industry.</description>
<pubDate>Fri, 25 Apr 2003 10:33:09 GMT</pubDate>
<guid>http://scriptingnews.userland.com/backissues/2003/04/25#When:6:33:09AM</guid>
</item>
<item>
<title>Three hot topics</title>
<link>http://scriptingnews.userland.com/backissues/2003/04/25#threeHotTopics</link>
<description>
<p>Report from last night's session. We did the usual hour's worth of software demos and then switched over to three topics that are much on my mind: </p>
<p><img src="http://radio.weblogs.com/0001015/images/2003/04/25/heHangedHimself.jpg" width="45" height="56" border="0" align="right" hspace="15" vspace="5" alt="A picture named heHangedHimself.jpg">1. How to integrate blogging with radio (not Radio). We talked about this at length. As other people talked I realized it is not about technology. There is no magic formula that will make the two worlds connect. Chris says let's hear what they think, and I say let's see. Two different senses, one visual and cerebral, and the other auditory and soulful. The answer, as it often is, is people. A roundtable of intelligent bloggers, like Washington Week in Review, or The Capital Gang, but staffed by writers who work in blogspace, and done on the radio, once a week. Chris is our anchor. I want to do this at BloggerCon, and every week, starting asap.</p>
<p>2. Blogging and the New Hampshire primary. Citizen bloggers covering the candidates for US president, follow it where it goes. First step -- clearly -- go to NH myself and find some candidates. Luckily I'm speaking at Dartmouth on May 9. I'll go looking for presidential hopefuls. With my camera and some questions. I'll try to explain weblogs. And here's another way to proceed. Are there any people in NH reading this site who think weblogs could make a difference? We need a citizen's committee for evangelizing the concept. Everyone who hears it goes Hmm, that might work.</p>

<p>3. BloggerCon, otherwise known as Weblogs in Meatspace. In October a conference in Cambridge about weblogs as writer's medium, from a historic and technological perspective. Computer industry conferences have done a great job of the latter, at best a superficial job at the former. I want librarians, lawyers, historians, executives, musicians, producers, pundits, scholars, educators, personalities, politicians, and more.</p>
<p>BTW, Philip Greenspun came to last night's meeting. I expect he will write about this discussion as well. In fact, during the discussion he made some notes on his blog using my keyboard and mouse. As always the discussion was informal, and the minds alive and interesting. </p>
</description>
<pubDate>Fri, 25 Apr 2003 10:03:57 GMT</pubDate>
<guid>http://scriptingnews.userland.com/backissues/2003/04/25#threeHotTopics</guid>
</item>
<item>
<title>This pig won't fly</title>
<link>http://scriptingnews.userland.com/backissues/2003/04/25#thisPigWontFly</link>
<description>
<p>Social Software? I've been in the software biz for 2.5 decades, so I've seen this kind of hype over and over. Take something that exists, give it a fancy new name, and then blast at reporters and analysts about it. Every time around the loop it works <i>less</i> well. In the 80s it worked very well. In the early 21st Century, there aren't enough analysts with credibility to make such a pig fly. </p>
<p>P2P was the last gasp. I remember getting breathless invitations to keynotes where this or that luminary was going to finally tell us what it is. In the end it wasn't the technology that made a difference, but ironically, the people. Apparently the promoters of Social Software were listening. </p>
<p>It's wrong. We don't need this. Weblogs are about punching through the hype machine of idiot analysts and reporters who go for their BS. Social software has existed for years. What's the big news? A few people are looking for a pole to fly their flag on. <i>Pfui!</i></p>
</description>
<pubDate>Fri, 25 Apr 2003 10:01:53 GMT</pubDate>
<guid>http://scriptingnews.userland.com/backissues/2003/04/25#thisPigWontFly</guid>
</item>
<item>
<title>Poopy little wiener boys</title>
<link>http://scriptingnews.userland.com/backissues/2003/04/25#poopyLittleWienerBoys</link>
<description>
<p>On this day three years ago Dan Gillmor was being pecked at by poopy little wiener boys up past their bedtime. "Western civilization is in jeopardy," Dan said. "And it's all my fault."</p>
</description>
<pubDate>Fri, 25 Apr 2003 10:31:31

```
GMT</pubDate>
                                <guid>http://scriptingnews.userland.c
om/backissues/2003/04/25#poopyLittleWienerBoys</guid>
                                </item>
                        </channel>
          </rss>


[End of: ./test/xml-test/scriptingnews_rss.xml]


[Contents of: ./test/advanced/add.tata]


# This is an advanced test of the add function

println add 1 1
println add 1 1.1
println add 1.1 1
#println add "aaa" "bbbb" #no converter
#println add 1 "aaa" #no converter
#println add 1.1 "aaa" #no converter
#println add 1.a 2 #produces error
#println add 1 ( 1 2 3 ) #no converter
#println add 1 ( "a" "b" ) #no converter
#println add ( 1 2 3 ) ( 4 5 6 ) #no converter
#println add 1 "21" #no converter
#println add "1" "21" #no converter
#println add #got exception
println add 1.1 true #prints 2.1
#println add true ( 1 ) #no converter
#println add null 1 #prints error


[End of: ./test/advanced/add.tata]


[Contents of: ./test/advanced/and.tata]


# advanced test of the and function


println and true true
println and true false
println and false true
println and false false
println and 0  true
println and 0  1
println and 1  true
#println and 0  "1"
#println and 1  ( true )
#println and ( true ) true
#println and true


[End of: ./test/advanced/and.tata]


[Contents of: ./test/advanced/assign-set.tata]


# advanced test of the assign function
set "test" 0
assign "test" 1
#assign "test1" 1 #error
#assign & "test" "a" 2 #error


#ERROR !!!!!!!!!!!!!!!!
```

```
set "12" 3
assign & 1 2 4
println get "12"  #prints 4
#end of ERROR !!!!!!!!!!!!!


assign "test" "hello"
println get "test"
assign "test" ( 1 2 "hello" )
println get "test"
set "test" ( 1 "hello" ) #should report an error !!!!!!!!!!
println get "test"
assign "test" "hey"
println get "test"


#ERROR !!!!!!!!!!!!!!!!
set "true" true
assign true 0
println get "true"
#end of ERROR !!!!!!!!!!!!!



[End of: ./test/advanced/assign-set.tata]


[Contents of: ./test/advanced/block.tata]


# advanced test of the block function

set "b1"
{
          set "b2"
          {
                    set "b3"
                    {
                              7
}
}
}


println get "b1" #prints 7
println get "b2" #prints null
println get "b3" #prints null


set "a1"
{
          & { println set "a2" 3 } { println set "a3" "r" }
}

println get "a1"
println get "a2"
println get "a3"

[End of: ./test/advanced/block.tata]


[Contents of: ./test/advanced/boolean.tata]


# advanced test of the boolean functions

println & true true #acts as a string
println add true true #acts as an int
println false #string
```

```
println true #string
```

[End of: ./test/advanced/boolean.tata]

[Contents of: ./test/advanced/concat.tata]

```
# advanced test of the concatenation function

#basically casts the objects as strings
println & 0 1
println & 0 "1"
println & 0 "a"
println & 0 ( )
println & 0 ( "1" )
println & 0 ( 1 )
println & 0 ( )
println & true 0
println & 0 true
println & 0 null #ok!! returns 0
println & null 0 #ok!! returns 0
println & null ( )
println & null "a"
println & "a" null
println & null null
set "q" "q1"
println & ( get "q" ) 0
println & ( 1 2 3 ) ( 5 6 "6" )
println & ( null ) & ( ) & ( ) & ( 1 2 3 ) ( 5 6 "6" )
```

[End of: ./test/advanced/concat.tata]

[Contents of: ./test/advanced/count.tata]

```
# advanced test of the count function

println count "x" ( 1 2 5 5 6 ) true
####################
#all these tests expect a list and notify that.
#this is correct
println count "x" 0 1
println count "x" 0 "1"
println count "x" 0 "a"
println count "x" 0 ( )
println count "x" 0 ( "1" )
println count "x" 0 ( 1 )
println count "x" 0 ( )
println count "x" true 0
println count "x" 0 true
println count "x" 0 null
println count "x" null 0
println count "x" null ( )
println count "x" null "a"
println count "x" "a" null
println count "x" null null
####################

# lists, strings should be treated as true (ints, bools, reals are)
#println count "x" ( 1 2 3 ) ( 5 )

####################
println count "x" ( null null null ) true #returns 3, ok
```

```
set "q" "q1"
println count "x" ( get "q" ) 0 #returns 0, ok
println count "x" ( 1 2 "3" null "a" ) 1 #ret 5, ok
println count "x" ( 1 2 "3" null "a" ) 2.2 #ret 5, ok

set "lst" & & ( null ) ( ) ( 1 2 3 )
println count "x" get "lst"  3 #ret=null, ok lst isnt a list
println assign "lst" union union ( null ) ( ) ( 1 2 3 )
println count "x" get "lst"  3 #ret=4 correct

set "var" 0
println count "x" ( 1 2 3 )
{
          lt get "x" 2 #get "var"
}

println count "x" ( 1 false 3 )
{
          assign "x" get "x"
}

#produces expected error
#println count "x" ( 1 ( 1 2 3 ) 3 )
#{
#         assign "x" get "x"
#}

println count "x" ( 1 true 3 ) #ret = 3
{
          unset "x"
}

println count "x" ( 1 false 3 )  #ret = 2
{
          unset "x"
}



[End of: ./test/advanced/count.tata]


[Contents of: ./test/advanced/dfs.tata]



#a state: ((board position)(0 position: row, col)(list of moves to
get to goal))
set "goal" ( ( 1 2 3 ) ( 4 5 6 ) ( 7 8 0 ) ( 2 2 ) )
set "start" ( ( 1 2 3 ) ( 4 5 6 ) ( 0 7 8 ) ( 2 0 ) )
set "state" ( )
set "newstate" ( )
set "queue" ( )
set "zpos" ( )
set "newzpos" ( )


assign "queue" ( get "start" )

#while not eq start (head queue)
          set "state" index 0 get "queue"  #get head
          println get "state"

          rem 0 get "queue" #dequeue

          println get "queue"

#create north state
          assign "newstate" "state"
```

```
          assign "pos" index 4 get "newstate" #grab the forth list

println get "pos"

#         if gt index 0 get "pos" 0  #ensure that oper. can work
#                   assign "newpos" get "pos";
#                   replace newpos(0)  pos(0) - 1;
#                   assign newstate swap(newstate, pos, newpos)
#                   replace 4 get "newstate" get "newpos"
#                   assign "queue" ( get "queue" get
#                   set "queue" union get "queue" get "newstate"
#push newstate
#         endif
#endwhile



[End of: ./test/advanced/dfs.tata]


[Contents of: ./test/advanced/diff.tata]


# advanced test of the diff function

println diff 0 1
println diff 0 1.1
println diff 2.2 1.1
println diff 0 "1"
println diff 0 "a"
println diff 0 ( )
println diff 0 ( "1" )
println diff 0 ( 1 )
println diff 0 ( )
println diff true 0
println diff 0 true
println diff 0 null
println diff null 0
println diff null ( )
println diff null "a"
println diff "a" null
println diff "b" "a"
println diff ( ) ( )
println diff ( null ) ( null )
println diff ( 1 2 3 ) ( 4 5 "6" )
println diff ( 1 2 3 ) ( 1 2 3 )
println diff ( 1 2 3 ) ( 1 2 3 4 5 "6" )
println diff ( 1 2 3 ) ( 4.2 5.3 "6" 3 2 )




[End of: ./test/advanced/diff.tata]


[Contents of: ./test/advanced/div.tata]


# test of the division function

println div 49 7
#println div -49 7
#println div 49 -7
println div 49 0
println div 0 7
println div 1 7
println div 7 7
println div 7 7.1
println div 7.1 7
```

```
println div 1.7 1.7
println div 1.7 7
println div 1.7 0

println div 0 1
println div 0 1.1
println div 2.2 1.1
#println div 0 "1"
#println div 0 "a"
#println div 0 ( )
#println div 0 ( "1" )
#println div 0 ( 1 )
#println div 0 ( )
println div true 0
println div 0 true
#println div 0 null
#println div null 0
#println div null ( )
#println div null "a"
#println div "a" null
#println div "b" "a"
#println div ( ) ( )
#println div ( null ) ( null )
#println div ( 1 2 3 ) ( 4 5 "6" )
#println div ( 1 2 3 ) ( 1 2 3 )
#println div ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println div ( 1 2 3 ) ( 4.2 -5.3 "6" 3 2 )


[End of: ./test/advanced/div.tata]


[Contents of: ./test/advanced/eq.tata]


# Advanced Test of the equals function

println eq 0 false
println eq 1 true
println eq 49 7
println eq "a" "a"
println eq 49 0
println eq 0 7
println eq 1 7
println eq 7 7
println eq 7 7.1 #error !!!!! ret=true
println eq 7.1 7
println eq 1.7 1.7
println "*********"
println eq 1.7 7
println eq 1.7 0
println eq 1 1
println eq 0 1
println eq 0 1.1
println eq 2.2 1.1
println eq 0 "1"
println eq 0 "0" #ERROR !!!!! ret=true
println eq 0 ( ) #ERROR
println "*********"

println eq 0 ( "1" )
println eq 0 ( 1 )
println eq 0 ( )
println eq true 0
println eq 0 true
println eq 0 null
println eq null 0
println eq null ( )
println "*********"
println eq null "a"
```

```
println eq "a" null
println eq "b" "a"
println eq ( ) ( ) #works fine with all the lists!
println eq ( null ) ( null )
println eq ( 1 2 3 ) ( 4 5 "6" )
println eq ( 1 2 3 ) ( 1 2 3 )
println eq ( 1 2 3 ) ( 1 2 3 4 5 "6" )
println eq ( 1 2 3 ( ) ) ( 1 2 3 ( 1 2 ) )


[End of: ./test/advanced/eq.tata]


[Contents of: ./test/advanced/errors.txt]



#eq errors
println eq 7 7.1 #error !!!!! ret=true
println eq 0 "0" #ERROR !!!!! ret=true
println eq 0 ( ) #ERROR



#new errors
println find_last "x" ( 5 5 5 6 ) { gt get "x" 5.99 }

#this error is not caused by the "eq double int" since
#the error returns true!
println find "x" ( 0 1 2 3 4 null false true null 4.99 5.001 5 5 )
          { eq get "x" 5 }  #error !!!!! ret=true

#error!!! so what is it?? 1 or true? (obviously 1 = true)
println find "x" ( 1 true 1 )  => return=1
          { eq get "x" 1 }
println find "x" ( true 1 true 1 ) => return=true
          { eq get "x" 1 }


#shouldnt find be recursive??
println find "truex" ( ( 1 2 ( 3 ) ) 1 2 4 ) #=> output=null
          { eq get "truex" 3 }

shold print 2 3
println set "lst" union union ( null ) ( ) ( 1 2 3 )
println subrange 2 3 get "lst"



#foreach issues
#aparently the foreach function doesnt consider changes in the list.
#prints "a" and crashes
set "list" ( "a" "b" "c" )
foreach "x" get "list"
{
          println get "x"
          rem 0 get "list"
}

assign "list" ( "a" )
foreach "x" get "list"
{
          println get "x"
          assign "list" union get "list" get "list" #prints one a
}
```

```
#prints "a" "B" "c"
assign "list" ( "a" "B" "c" )
foreach "x" get "list"
{
          unset "list"   #should crash on the next itteration, but
doesnt
          println get "x"
}


#gte errors
println gte 7 7.1
println gte 0 "1"  #dont you have cases where you cast "1" to one??
println gte true 0 #ok, ret = true
println gte 0 true #error !!! gets an exception
#println gte 0 null #ok, returns exception
println gte null 0 #error !!!returns false
println gte null ( ) #error !!! returns false
println gte null "a" #error !!! false
println gte ( 1 2 3 ) ( 4 5 "6" ) #ret=false, what is the criteria?
println gte ( 1 2 3 ) ( 1 2 3 4 5 "6" ) #ret=true how
come???????????
println gte ( 1 2 3 ) ( 4.2 ( 5.3 ) "6" 3 2 ) #ret=false, again,
what is the criteria

#same errors with gt. need to check lt, lte


#ifelse errors and questions
if "1"  #sholdnt this print 1 ?
          { println 1 }
else
          { println null }
endif

#a BIG error!
set "x" 3

if get "X"
{ assign "x" 0 }  #assign doesnt change global value!!!!!!!!!!!!
endif
if get "x"
{ println "wrong!!" }
endif
println get "x" #prints 3!!!

#can also use list
if true
( println "a" println add 1 4 )
endif



#index
# a weird case
set "x" 7
println index 0 set "lst" ( get "x" assign "x" 1 ) #prints 7
println get "lst" #prints null



#mul
println mul true 0 #ret= 0
println mul 0 true



#not
#println not null #should return true ??
```

```
#overlap
#shouldnt this produce the same overlap??
println overlap ( 5 5 5 5 ) ( 5 )
println overlap ( 5 ) ( 5 5 5 5 )


#rem
set "y" ( 1 2 3 4 5 "6" )
println rem ( 1 2 3 ) get "y"  #prints 4 ??
println get "y"                          #prints ( 1 2 3 5 6 ) ??

#the rem function shoudnt work with get "y" it should be
#activated on the object itself. get "y" has the feeling of
#a return by value, where as we actually want to run the function
#on "y" itself.
# rem 3 "y" (for example)



#replace - BIG error
assign "out" ( null )
println replace 0 get "out" get "out"
println get "out"

#transform
#same as foreach
#set "list" ( "a" "b" "c" )
#transform "x" get "list"
#{
#         println get "x"
#         rem 0 get "list" #crashes!!!
#}



#unset
set "x" 3
assign "x" ( unset "x" )
println get "x"  #should print null
{ unset get "x" }
println get "x" #should print null



#when
#same error as in unless...
set "x" 3

when get "X"
{ assign "x" 0 }

when get "x"
{ println "error" } #prints error
println get "x" #prints 3


[End of: ./test/advanced/errors.txt]


[Contents of: ./test/advanced/filter.tata]


# advanced test of the filter function
# filters out x's that return true

println filter "x" ( 1 2 5 5 6 ) true
println filter "x" ( 5 5 5 5 ) { eq get "x" 5 }
####################
#all these tests expect a list and notify that.
```

```
#this is correct
println filter "x" 0 1
println filter "x" 0 "1"
println filter "x" 0 "a"
println filter "x" 0 ( )
println filter "x" 0 ( "1" )
println filter "x" 0 ( 1 )
println filter "x" 0 ( )
println filter "x" true 0
println filter "x" 0 true
println filter "x" 0 null
println filter "x" null 0
println filter "x" null ( )
println filter "x" null "a"
println filter "x" "a" null
println filter "x" null null
####################


# lists, strings should be treated as true (ints, bools, reals are)
#println filter "x" ( 1 2 3 ) ( 5 )



println "####################"
println filter "x" ( null null null ) true
set "q" "q1"
println filter "x" ( get "q" ) 0
println filter "x" ( 1 2 "3" null "a" ) 1
println filter "x" ( 1 2 "3" null "a" ) 2.2

set "lst" & & ( null ) ( ) ( 1 2 3 )
println filter "x" get "lst"  3 #ret=null, ok lst isnt a list
println assign "lst" union union ( null ) ( ) ( 1 2 3 )
println filter "x" get "lst"  3 # correct
println filter "x" get "lst"  0 # correct

set "var" 0
println filter "x" ( 1 2 3 )
{
          lt get "x" 2 #get "var"
}

println filter "x" ( 1 false 3 )
{
          assign "x" get "x"
}

#produces expected error when (1 2 3) is assigned to x
#println filter "x" ( 1 ( 1 2 3 ) 3 )
#{
#         assign "x" get "x"
#}

println filter "x" ( 1 true 3 )
{
          unset "x"
}

println filter "x" ( 1 false 3 )
{
          unset "x"
}



[End of: ./test/advanced/filter.tata]
```

[Contents of: ./test/advanced/filter_out.tata]

```
# advanced test of the filter_out function
# filters out x's that return true

println filter_out "x" ( 1 2 5 5 6 ) true
println filter_out "x" ( 5 5 5 5 ) { eq get "x" 5 }
####################
#all these tests expect a list and notify that.
#this is correct
println filter_out "x" 0 1
println filter_out "x" 0 "1"
println filter_out "x" 0 "a"
println filter_out "x" 0 ( )
println filter_out "x" 0 ( "1" )
println filter_out "x" 0 ( 1 )
println filter_out "x" 0 ( )
println filter_out "x" true 0
println filter_out "x" 0 true
println filter_out "x" 0 null
println filter_out "x" null 0
println filter_out "x" null ( )
println filter_out "x" null "a"
println filter_out "x" "a" null
println filter_out "x" null null
######################


# lists, strings should be treated as true (ints, bools, reals are)
#println filter_out "x" ( 1 2 3 ) ( 5 )



println "######################"
println filter_out "x" ( null null null ) true #returns (), ok
set "q" "q1"
println filter_out "x" ( get "q" ) 0 #returns q1, ok
println filter_out "x" ( 1 2 "3" null "a" ) 1 #ret (), ok
println filter_out "x" ( 1 2 "3" null "a" ) 2.2 #ret (), ok

set "lst" & & ( null ) ( ) ( 1 2 3 )
println filter_out "x" get "lst"  3 #ret=null, ok lst isnt a list
println assign "lst" union union ( null ) ( ) ( 1 2 3 )
println filter_out "x" get "lst"  3 #ret=() correct
println filter_out "x" get "lst"  0 #ret=() correct

set "var" 0
println filter_out "x" ( 1 2 3 )
{
          lt get "x" 2 #get "var"
}

println filter_out "x" ( 1 false 3 )
{
          assign "x" get "x"
}

#produces expected error when (1 2 3) is assigned to x
#println filter_out "x" ( 1 ( 1 2 3 ) 3 )
#{
#         assign "x" get "x"
#}

println filter_out "x" ( 1 true 3 ) #ret = 3
{
          unset "x"
}

println filter_out "x" ( 1 false 3 )  #ret = 2
```

```
{
          unset "x"
}



[End of: ./test/advanced/filter_out.tata]


[Contents of: ./test/advanced/find.tata]


# advanced test of the find function

println find "x" ( 1 2 5 5 6 ) true
println find "x" ( 5 5 5 5 ) { eq get "x" 5 }
println find "x" ( 0 1 2 3 4 null false true null 4.99 5.001 5 5 )
          { eq get "x" 5 }  #error !!!!! ret=true
#println find "x" ( 5 5 5 6 ) { gt get "x" 5.99 } #error!!!!
####################
#all these tests expect a list and notify that.
#this is correct
println find "x" 0 1
println find "x" 0 "1"
println find "x" 0 "a"
println find "x" 0 ( )
println find "x" 0 ( "1" )
println find "x" 0 ( 1 )
println find "x" 0 ( )
println find "x" true 0
println find "x" 0 true
println find "x" 0 null
println find "x" null 0
println find "x" null ( )
println find "x" null "a"
println find "x" "a" null
println find "x" null null
######################


# lists, strings should be treated as true (ints, bools, reals are)
#println find "x" ( 1 2 3 ) ( 5 )



println "######################"
println find "x" ( 1 ) false #ret = null
println find "x" ( 1 ) true #ret = null
println find "x" ( true 1 true 1 )
          { eq get "x" 1 }
#println find "x" ( "1" true ) "1" #ret = null, should return "1"

println find "truex" ( ( 1 2 ( 3 ) ) 1 2 4 )
          { eq get "truex" 3 }

println find "x" ( ) true #ret = null
println find "x" ( null null null ) true #returns null, ok
set "q" "q1"
println find "x" ( get "q" ) 0 #returns null, ok
println find "x" ( 1 2 "3" null "a" ) 1 #ret 1, ok
println find "x" ( 1 2 "3" null 1.99 ) 2.2 #ret 1, ok
println "********************"
set "lst" & & ( null ) ( ) ( 1 2 3 )
println find "x" get "lst"  3 #ret=3
println assign "lst" union union ( null ) ( ) ( 1 2 3 )
println find "x" get "lst"  3 #ret=null correct
println find "x" get "lst"  0 #ret=null correct

set "var" 0
```

```
println find "x" ( 1 2 3 )
{
          lt get "x" 2 #get "var"
}

println find "x" ( 1 false 3 )
{
          assign "x" get "x"
}

#produces expected error when (1 2 3) is assigned to x
#println find "x" ( 1 ( 1 2 3 ) 3 )
#{
#         assign "x" get "x"
#}

println find "x" ( 1 true 3 ) #ret = 3
{
          unset "x"
}

println find "x" ( false 0 1 false 3 false )  #ret = 3
{
          unset "x"
}



[End of: ./test/advanced/find.tata]


[Contents of: ./test/advanced/find_last.tata]


# advanced test of the find_last function

println find_last "x" ( 1 2 5 5 6 ) true
println find_last "x" ( 5 5 5 5 ) { eq get "x" 5 }
#println find_last "x" ( 5 5 5 6 ) { gt get "x" 5.99 } #error!!!!
####################
#all these tests expect a list and notify that.
#this is correct
println find_last "x" 0 1
println find_last "x" 0 "1"
println find_last "x" 0 "a"
println find_last "x" 0 ( )
println find_last "x" 0 ( "1" )
println find_last "x" 0 ( 1 )
println find_last "x" 0 ( )
println find_last "x" true 0
println find_last "x" 0 true
println find_last "x" 0 null
println find_last "x" null 0
println find_last "x" null ( )
println find_last "x" null "a"
println find_last "x" "a" null
println find_last "x" null null
####################


# lists, strings should be treated as true (ints, bools, reals are)
#println find_last "x" ( 1 2 3 ) ( 5 )



println "####################"
println find_last "x" ( ) true #ret = null
println find_last "x" ( null null null ) true #returns null, ok
set "q" "q1"
```

```
println find_last "x" ( get "q" ) 0 #returns null, ok
println find_last "x" ( 1 2 "3" null "a" ) 1 #ret (), ok
println find_last "x" ( 1 2 "3" null 1.99 ) 2.2 #ret (), ok

set "lst" & & ( null ) ( ) ( 1 2 3 )
println find_last "x" get "lst"  3 #ret=3
println assign "lst" union union ( null ) ( ) ( 1 2 3 )
println find_last "x" get "lst"  3 #ret=() correct
println find_last "x" get "lst"  0 #ret=() correct

set "var" 0
println find_last "x" ( 1 2 3 )
{
          lt get "x" 2 #get "var"
}

println find_last "x" ( 1 false 3 )
{
          assign "x" get "x"
}

#produces expected error when (1 2 3) is assigned to x
#println find_last "x" ( 1 ( 1 2 3 ) 3 )
#{
#         assign "x" get "x"
#}

println find_last "x" ( 1 true 3 ) #ret = 3
{
          unset "x"
}

println find_last "x" ( 1 false 3 false )  #ret = 3
{
          unset "x"
}



[End of: ./test/advanced/find_last.tata]


[Contents of: ./test/advanced/foreach.tata]


# advanced test of the foreach function

println "XXXX"
foreach "x" ( 1 2 5 5 6 )
{ println get "x" }

println "===="
foreach "x" ( 1 2.2 5.2 5 6 "4" 5 null ( "a" 1 4.4 null ( ) ) ( )
"abc" true ( false ) )
{ println get "x" }

println "XXXX"
foreach "x" ( ) { add get "x" get "x" }

println "===="
#foreach "x" null { add get "x" get "x" }

println "XXXX"
#foreach "x" ( true )

foreach "x" ( 0 1 "get x" 0 0 ) { get "x" unset "x" }

println "XXXX"
set "list" ( "a" "b" "c" )
```

```
foreach "x" get "list"
{
          println get "x"
#         rem 0 get "list" #crashes!!!
}


println "YYYY"
assign "list" ( "a" )
foreach "x" get "list"
{
          println get "x"
          assign "list" union get "list" get "list" #prints one a
}


assign "list" ( "a" "B" "c" )
foreach "x" get "list"
{
          unset "list"
          println get "x"
}



[End of: ./test/advanced/foreach.tata]


[Contents of: ./test/advanced/general.tata]


Println "test"


[End of: ./test/advanced/general.tata]


[Contents of: ./test/advanced/get.tata]


# advanced test of the get function
set "x" ( 1 2 3 )

println get "x"

{ println get "x" }
{ { println get "x" } }


println { get "x" { unset get "x" } } #ok

set "x" "get "x""
println get "x" #ok, no recursion


[End of: ./test/advanced/get.tata]


[Contents of: ./test/advanced/gt.tata]


# advanced test of the gt function

println "ints..."
println gt 49 7
println gt 49 0
println gt 0 1
println gt 0 7
println gt 1 7
```

```
println gt 7 7

println "doubles..."
#println gt 7 7.1 #error !!!! cant gt int double
#println gt 7.1 7
#println gt 1.7 1.7
#println gt 1.7 7
#println gt 1.7 0
#println gt 0 1.1
#println gt 2.2 1.1

println "nulls and bools..."
println gt true 0
#println gt 0 true #error !!! gets an exception
#println gt 0 null #exception
#println gt null 0 #error !!! returns false
#println gt null ( ) #error !!! returns false, should throw
exception
#println gt 0 ( )
println gt null "a" #error !!! returns false


println "strings and lists..."
#all exceptions
#println gt 0 "1"
#println gt 0 "a"
#println gt 0 ( )
#println gt 0 ( "1" )
#println gt 0 ( 1 )
#println gt 0 ( )
#println gt "a" null
#end of exceptions
println gt "b" "a"
#same errors as in gte!!!
#println gt ( ) ( )
#println gt ( null ) ( null )
#println gt ( 1 2 3 ) ( 4 5 "6" )
#println gt ( 1 2 3 ) ( 1 2 3 )
#println gt ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println gt ( 1 2 3 ) ( 4.2 ( 5.3 ) "6" 3 2 )


[End of: ./test/advanced/gt.tata]


[Contents of: ./test/advanced/gte.tata]


# advanced Test of the gte function


println gte 49 7
#println gte -49 7
#println gte 49 -7
println gte 49 0
println gte 0 7
println gte 1 7
println gte 7 7
#println gte 7 7.1 #error !!!! cant gte int double
#println gte 7.1 7
#println gte 1.7 1.7
#println gte 1.7 7
#println gte 1.7 0
println "XXXXXXXXXXX"
println gte 0 1

#exceptions...
#println gte 0 1.1
#println gte 2.2 1.1
#println gte 0 "1"
```

```
#println gte 0 "a"
#println gte 0 ( )
#println gte 0 ( "1" )
#println gte 0 ( 1 )
#println gte 0 ( )
println gte true 0
#println gte 0 true #error !!! gets an exception
#println gte 0 null
#println gte null 0
println gte null ( )
println ":::::::::::::"
#println gte 0 ( )
println gte null "a" #error !!! returns false
#println gte "a" null
println gte "b" "a"
println gte ( ) ( )
println gte ( null ) ( null )
println gte ( 1 2 3 ) ( 4 5 "6" )
println gte ( 1 2 3 ) ( 1 2 3 )
println gte ( 1 2 3 ) ( 1 2 3 4 5 "6" )
println gte ( 1 2 3 ) ( 4.2 ( 5.3 ) "6" 3 2 )


[End of: ./test/advanced/gte.tata]


[Contents of: ./test/advanced/if.tata]


# advanced test of the 'if' function

# basic if else
if true
{
          println "conditional was true"
}
else
{
          println "conditional was false"
}
endif

# looky here, an if without an else and a complex conditional!
if not false
{
          println "conditional was true"
}
endif

# pay attention to what is going on here! the body of an if or else
statement
# must be a block! a block must either be a SINGLE function or a
blocking
# function. 'println "this won't get printed"' is TWO functions!
TWO! IT MUST
# BE BLOCKED. This is a common mistake, so learn it!
# BEWARE THE EL GREGOR

if true
{
          if false
                    { println "this won't get printed" }
          else
                    { println "looky here, nested if statements!
WHO IS THE EL GREGOR?!" }
          endif
}
else
{
          println "if you see this, you are fucked"
```

```
}
endif

# now we gotta do the if ... else if, note the double endif
statements!
if false
{
          println "this won't show"
}
else
          if true
          {
                    println "EL GREGOR, please print this"
          }
          endif
endif


#if ( )
#         { println "true" }
#endif

if null
          { println "true" }
else
          { println null }
endif



if 0
          { println "0" }
else
          { println null }
endif


if 1
          { println "1" }
else
          { println null }
endif




if 0.0
          { println "0.0" }
else
          { println null }
endif



if 0.1
          { println "0.1" }
else
          { println null }
endif


#if "1"
#         { println 1 }
#else
#         { println null }
#endif


set "x" 3
```

```
if get "X"
{ assign "x" 0 }
endif
if get "x"
{ println "wrong!!" }
endif

println get "x"


#can also use a list
if true
( println "a" println add 1 4 )
endif









[End of: ./test/advanced/if.tata]


[Contents of: ./test/advanced/ifblock.tata]




if  {
        if true
        {
                println "hello"
        }
        endif

        }
        { println "cool" }
        endif



[End of: ./test/advanced/ifblock.tata]


[Contents of: ./test/advanced/index.tata]


# advanced test of the index function

println index 0 ( 1 2 3 )
println index 1 ( 1 2 3 )
println index 2 ( 1 2 3 )
#println index 3 ( 1 2 3 ) #exception, ok


#println index 0 ( ) #exception, ok
println index 0 ( null )
println index 0 ( false )
println index 0 ( true )


println index 0 ( "1" )
```

```
println index 0 ( null "1" )
println index 0 ( "1" false )
println index 0 ( ( 0 1 2 ) "1" )

println index 0 ( add 1 2 )
set "x" add 3 4
println index 0 set "lst" ( get "x" assign "x" 1 )
println get "lst"
```

[End of: ./test/advanced/index.tata]

[Contents of: ./test/advanced/list.tata]

```
# advanced test of the list function

println ( 1 2 3 4 5 6 7 )
println ( 1 "2" null true false ( null ) ( ) ( 1.1 1 ) 3.1 )

set "x" ( get "x" )
println get "x" #ok

assign "x" 1
assign "x" ( unset get "x" 1 2 )
println get "x" #prints null, ok



```

[End of: ./test/advanced/list.tata]

[Contents of: ./test/advanced/lt-test.tata]

```
# Buko O.
# 4.20.2003

# Test of the less than function

println lt 7 7  # false

println lt 7 8  # true

println lt 8 7  # false
```

[End of: ./test/advanced/lt-test.tata]

[Contents of: ./test/advanced/lte-test.tata]

```
# Buko O.
# 4.20.2003

# Test of the less than equals function

println lte 7 7 # true
println lte 7 8 # true
println lte 8 7 # false
```

[End of: ./test/advanced/lte-test.tata]

[Contents of: ./test/advanced/mod.tata]

```
# test of the modulus function
```

```
println mod 12 5
println mod 49 7
#println mod -49 7
#println mod 49 -7
println mod 49 0
println mod 0 7
println mod 1 7
println mod 7 7
println mod 7 7.1
println mod 7.1 7
println mod 1.7 1.7
println mod 1.7 7
println mod 1.7 0
println "%%%%%%%%%%%%%%%%"
println mod 0 1
println mod 0 1.1
println mod 2.2 1.1
#println mod 0 "1"
#println mod 0 "a"
#println mod 0 ( )
#println mod 0 ( "1" )
#println mod 0 ( 1 )
#println mod 0 ( )
println mod true 0
println mod 0 true
#println mod 0 null
#println mod null 0
#println mod null ( )
#println mod null "a"
#println mod "a" null
#println mod "b" "a"
#println mod ( ) ( )
#println mod ( null ) ( null )
#println mod ( 1 2 3 ) ( 4 5 "6" )
#println mod ( 1 2 3 ) ( 1 2 3 )
#println mod ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println mod ( 1 2 3 ) ( 4.2 -5.3 "6" 3 2 )


[End of: ./test/advanced/mod.tata]


[Contents of: ./test/advanced/mul.tata]


# advanced test of the multiply function

println mul 49 7
#println mul -49 7
#println mul 49 -7
println mul 49 0
println mul 0 7
println mul 1 7
println mul 7 7
println mul 7 7.1
println mul 7.1 7
println mul 1.7 1.7
println mul 1.7 7
println mul 1.7 0

println mul 0 1
println mul 0 1.1
println mul 2.2 1.1
#println mul 0 "1"
#println mul 0 "a"
#println mul 0 ( )
#println mul 0 ( "1" )
#println mul 0 ( 1 )
#println mul 0 ( )
println mul true 0
```

```
println mul 0 true
#println mul 0 null
#println mul null 0
#println mul null ( )
#println mul null "a"
#println mul "a" null
#println mul "b" "a"
#println mul ( ) ( )
#println mul ( null ) ( null )
#println mul ( 1 2 3 ) ( 4 5 "6" )
#println mul ( 1 2 3 ) ( 1 2 3 )
#println mul ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println mul ( 1 2 3 ) ( 4.2 -5.3 "6" 3 2 )


[End of: ./test/advanced/mul.tata]


[Contents of: ./test/advanced/not.tata]


# advanced Test of the not function

println not 49
#println not -49 7
#println not 49 -7
println not 7.1
println not 0.0
println not 0.1
println not true
println not false

#println not "true"
#println not "false"
#println not ( )
#println not ( "1" )
#println not ( 1 )
#println not null
#println not "a"
#println not ( null )
#println not ( 1 2 3 )
#println not ( 1 2 3 4 5 "6" )


[End of: ./test/advanced/not.tata]


[Contents of: ./test/advanced/null.tata]


# advanced test of the null function

println null
set "x" null
println get "X"

set "y" ( null )
println get "y"


[End of: ./test/advanced/null.tata]


[Contents of: ./test/advanced/or.tata]


# advanced test of the or function

println or true true
println or true false
```

```
println or false true
println or false false

println or 12 5
#println or -49 7
#println or 49 -7
println or 49 0
println or 0 7
println or 7 7.1
println or 7.1 7
println or 1.7 1.7
println or 1.7 0
println or 0.0 0.0
println "%%%%%%%%%%%%%%%%"
#println or 0 "1"
#println or 0 "a"
#println or 0 ( )
#println or 0 ( "1" )
#println or 0 ( 1 )
#println or 0 ( )
println or true 0
println or false 1
println or 0 true
#println or 0 null
#println or null 0
#println or null ( )
#println or null "a"
#println or "a" null
#println or "b" "a"
#println or ( ) ( )
#println or ( null ) ( null )
#println or ( 1 2 3 ) ( 4 5 "6" )
#println or ( 1 2 3 ) ( 1 2 3 )
#println or ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println or ( 1 2 3 ) ( 4.2 -5.3 "6" 3 2 )


[End of: ./test/advanced/or.tata]


[Contents of: ./test/advanced/overlap.tata]


# advanced test of the overlap function

println overlap ( 1 2 5 5 6 ) true
println overlap ( 5 5 5 5 ) ( 5 )
println overlap ( 5 ) ( 5 5 5 5 )

####################
#all these tests expect a list and notify that.
#this is correct
println overlap 0 1
println overlap 0 "1"
println overlap 0 "a"
println overlap 0 ( )
println overlap 0 ( "1" )
println overlap 0 ( 1 )
println overlap 0 ( )
println overlap true 0
println overlap 0 true
println overlap 0 null
println overlap null 0
println overlap null ( )
println overlap null "a"
println overlap "a" null
println overlap null null
######################
```

```
# lists, strings should be treated as true (ints, bools, reals are)
#println overlap ( 1 2 3 ) ( 5 )



println "####################"
println overlap ( null null null ) true
println overlap ( true null null null ) true

set "q" "q1"
println overlap ( get "q" ) 0
println overlap ( 1 2 "3" null "a" ) 1
println overlap ( 1 2 "3" null "a" ) 2.2

set "lst" & & ( null ) ( ) ( 1 2 3 )
println overlap get "lst"  3 #ret=null, ok lst isnt a list
assign "lst" union union ( null ) ( ) ( 1 2 3 )
println overlap get "lst"  3 # correct
println overlap get "lst"  0 # correct
println "%%%%%%%%%%"
set "var" 0
println overlap ( 1 2 3 ( 3 ) ) ( 1 2 3 ( 3 1 ) ( 3.1 ) ( 2.9 ) ( 3
) )
println overlap ( 1 2 3 ( 3 ) ) ( 1 2 3 ( 3 1 ) ( 3.1 ) ( 2.9 ) )
println overlap ( and true false ) ( true false )
println overlap ( true false ) ( and true false )
println overlap ( false ) ( and true false )
println overlap ( and true false ) ( false )



[End of: ./test/advanced/overlap.tata]


[Contents of: ./test/advanced/print.tata]


# advanced Test of the print function

# print a string
print "Hello World!"

# print an integer
print 7

# print a double
print 12.5

print null
# print a list
print ( ( ) )
print ( 1 2.2 "a" ( 1 2.2 "c" ( ) null ) null )
print ( and 0 1 ( null ) )
print ( true false ( true false ) print "print" )

print { "ja maan" }
print ( "ja maan" )

[End of: ./test/advanced/print.tata]


[Contents of: ./test/advanced/println.tata]


# advanced Test of the println function

# print a string
println "Hello World!"

# print an integer
```

```
println 7

# print a double
println 12.5

println null
# print a list
println ( ( ) )
println ( 1 2.2 "a" ( 1 2.2 "c" ( ) null ) null )
println ( and 0 1 ( null ) )
println ( true false ( true false ) println "println" )

println { "ja maan" }
println ( "ja maan" )

[End of: ./test/advanced/println.tata]


[Contents of: ./test/advanced/rem.tata]


# adv. test of the rem function


println rem 49 7
println rem 49 0
println rem 0 7
println rem 1 7
println rem 7 7
println rem 7 7.1
println rem 7.1 7
println rem 1.7 1.7
println rem 1.7 7
println rem 1.7 0
println rem 0 1
println rem 0 1.1
println rem 2.2 1.1
println rem true 0
println rem 0 true
println "###########"
println rem 0 "1"
println rem 0 "a"
#println rem 0 ( )
println rem 0 ( "1" )
println rem 0 ( 1 )
println rem 0 ( 1.1 )
println rem 0 null
println rem null 0
#println rem null ( )
println rem null "a"
#println rem "a" null
println rem null 4
#println rem "b" "a"
#println rem ( ) ( )
#println rem ( null ) ( null )
#println rem ( 1 2 3 ) ( 4 5 "6" )
println rem ( 1 2 3 ) ( 1 2 3 5 )
println "XXXXXXXXX"
set "y" ( 1 2 3 4 5 "6" )
println rem ( 1 2 3 ) get "y"  #error !!!
println get "y"      #error !!!

#println rem 2 "y"

set "x" null
println rem 0 get "x"
println get "x"

[End of: ./test/advanced/rem.tata]
```

[Contents of: ./test/advanced/replace.tata]

```
# adv. test of the replace function

set "nums" ( 1 2 3 4 5 6 )
replace 1 get "nums" 3
println get "nums"

set "rlist" ( ( ( 3 ) ( 4 ) ) 5 6 )
set "newList" ( 1 1 )
set "var" replace 0 index 1 index 0 get "rlist" get "newList"
println get "rlist"

set "out" 0
println replace null null null
println replace 0 ( null ) ( null )
println replace 0 null ( null )
println replace 0 ( null ) null
assign "out" ( null )
println replace 0 get "out" 1
println get "out"


assign "out" ( null )
println replace 0 get "out" null
println get "out"



assign "out" ( null )
println replace 0 get "out" ( null )
println get "out"


#error - converter stuck in recursion !!!
#assign "out" ( null )
#println replace 0 get "out" get "out"
#println get "out"


assign "out" ( null )
println replace 0.1 get "out" ( null )
println get "out"

assign "out" ( null )
println replace 0.1 get "out" ( "true" )
println get "out"

assign "out" ( 1 1.1 "1" true ( ) )
println replace 0 get "out" ( "true" )
println replace 1 get "out" ( "false" )
println replace 2 get "out" ( "( hello )" )
println replace 3 get "out" ( ( ) )
println replace 4 get "out" ( 5.5 "eight" )
println get "out"



[End of: ./test/advanced/replace.tata]


[Contents of: ./test/advanced/size.tata]


# adv. test of the size function

println size 12
println size 0
```

```
println size 7.1
println size "1"
println size "123"

println size 0 ( )
println size 0 ( "1" )
println size 0 ( 1 )
println size true
println size false
println size null
println size ( null null )
println size ( 1 null null 2 )
println size ( 1 2 3 ( 4 5 "6" ) )
println size ( 1 2 3 1 2 3 )
println size ( 1 ( 2 3 ) ( 1 2 3 ) 4 5 "6" )
```

[End of: ./test/advanced/size.tata]

[Contents of: ./test/advanced/sub.tata]

```
# advanced test of the subtract function

println sub 49 7
#println sub -49 7
#println sub 49 -7
println sub 49 0
println sub 0 7
println sub 1 7
println sub 7 7
println sub 7 7.1
println sub 7.1 7
println "************"
println sub 1.7 1.7
println sub 1.7 7
println sub 1.7 0
println sub 0 1
println sub 0 1.1
println sub 2.2 1.1
println sub true 0
println sub 0 true
println "************"

#println sub 0 "1"
#println sub 0 "a"
#println sub 0 ( )
#println sub 0 ( "1" )
#println sub 0 ( 1 )
#println sub 0 ( )
#println sub 0 null
#println sub null 0
#println sub null ( )
#println sub null "a"
#println sub "a" null
#println sub "b" "a"
#println sub ( ) ( )
#println sub ( null ) ( null )
#println sub ( 1 2 3 ) ( 4 5 "6" )
#println sub ( 1 2 3 ) ( 1 2 3 )
#println sub ( 1 2 3 ) ( 1 2 3 4 5 "6" )
#println sub ( 1 2 3 ) ( 4.2 -5.3 "6" 3 2 )
```

[End of: ./test/advanced/sub.tata]

[Contents of: ./test/advanced/subrange.tata]

```
# adv. test of the subrange function
set "nums" ( 1 2 3 4 5 6 7 8 9 0 )

println subrange 1 4 get "nums"  # should print 2-4


println subrange 0.99 1 ( 1 2 5 5 6 )
println subrange 1.9 2.1 ( 1 2 5 5 6 )
println subrange ( 5 5 5 5 ) ( 2 3 )
#println subrange 1 100 ( 5 101 0 1 2 5 0 5 6 )
println subrange 5 5 ( 1 2 5 5 6 )
println subrange 0 4 ( 1 2 null "1" 5 null )
set "0" 1
println subrange "0" "4" ( 1 2 null "1" 5 null )

println "######################"
#println subrange ( null null null ) true

println set "lst" & & ( null ) ( ) ( 1 2 3 )
println subrange 0 3 get "lst" #null
println subrange 0 3 "aaa bbb ccc ddd" #null

println set "lst" union union ( null ) ( ) ( 1 2 3 )
println subrange 2 3 get "lst"



[End of: ./test/advanced/subrange.tata]


[Contents of: ./test/advanced/template.txt]


# common test, where bin-test-func is a binary tata function
# like: and, add, div, bin-test-func ...


println bin-test-func 0 false
println bin-test-func 1 true
println bin-test-func 49 7
println bin-test-func "a" "a"
println bin-test-func 49 0
println bin-test-func 0 7
println bin-test-func 1 7
println bin-test-func 7 7
println bin-test-func 7 7.1 #error !!!!! ret=true
println bin-test-func 7.1 7
println bin-test-func 1.7 1.7
println "*********"
println bin-test-func 1.7 7
println bin-test-func 1.7 0
println bin-test-func 1 1
println bin-test-func 0 1
println bin-test-func 0 1.1
println bin-test-func 2.2 1.1
println bin-test-func 0 "1"
println bin-test-func 0 "0" #ERROR !!!!! ret=true
println bin-test-func 0 ( ) #ERROR
println "*********"
println bin-test-func 0 ( "1" )
println bin-test-func 0 ( 1 )
println bin-test-func 0 ( )
println bin-test-func true 0
println bin-test-func 0 true
println bin-test-func 0 null
println bin-test-func null 0
println bin-test-func null ( )
println "*********"
println bin-test-func null "a"
```

```
println bin-test-func "a" null
println bin-test-func "b" "a"
println bin-test-func ( ) ( ) #works fine with all the lists!
println bin-test-func ( null ) ( null )
println bin-test-func ( 1 2 3 ) ( 4 5 "6" )
println bin-test-func ( 1 2 3 ) ( 1 2 3 )
println bin-test-func ( 1 2 3 ) ( 1 2 3 4 5 "6" )
println bin-test-func ( 1 2 3 ( ) ) ( 1 2 3 ( 1 2 ) )



[End of: ./test/advanced/template.txt]


[Contents of: ./test/advanced/transform.tata]


# Buko O.
# 4.21.2003

# test of the transform function
set "doubles"
          transform "x" ( 1 2 3 4 5 )
          {
                    mul get "x" 2
          }

println get "doubles"


println transform "x" ( 1 2 3 4 ) true
println transform "x" ( 1 ( 2 ) 3.3 "4" ) { get "x" }
println transform "x" ( 1 ( 2 ) 3.3 "4" ) "x"
println transform "x" ( 1 ( 2 ) 3.3 "4" ) ( get "x" )


println "XXXX"
transform "x" ( 1 2 5 5 6 )
{ println get "x" }  #converts to a list of strings

println "===="
println transform "x" ( 1 2.2 5.2 5 6 "4" 5 null ( "a" 1 4.4 null (
) ) ( ) "abc" true ( false ) )
{ println get "x" }

println "XXXX"
println transform "x" ( ) { add get "x" get "x" }

println "===="
#transform "x" null { add get "x" get "x" }

println "XXXX"
println transform "x" ( true ) { }
println transform "x" ( true ) ( ( ) ( ) { } )

println "XXXX"
set "list" ( "a" "b" "c" )
transform "x" get "list"
{
          println get "x"
#         rem 0 get "list" #crashes!!!
}


println "YYYY"
assign "list" ( "a" )
transform "x" get "list"
{
          println get "x"
          assign "list" union get "list" get "list" #prints one a
```

```
}


assign "list" ( "a" "B" "c" )
println transform "x" get "list"
{
          unset "list"
          println get "x"
}



[End of: ./test/advanced/transform.tata]


[Contents of: ./test/advanced/union.tata]


# advanced test of the union function

println union 49 7
#println union -49 7
#println union 49 -7
println union 49 0
println union 0 7
println union 1 7
println union 7 7
println union 7 7.1
println union 7.1 7
println union 1.7 1.7
println union 1.7 7
println union 1.7 0
println union true 0
println union 0 true

println union 0 1
println union 0 1.1
println union 2.2 1.1
println union 0 "1"
println union 0 "a"
println "XXXXXXXXXXXXX"
println union 0 ( )
println union 0 ( "1" )
println union 0 ( 1 )
println union 0 ( )
println union ( ) null
println union ( "1" ) 0
println union ( 1 ) 0
println union ( ) 0

println "YYYYYYYYYYYY"
println union 0 null
println union null 0
println union null ( )
println union null "a"
println union "a" null
println union "b" "a"
println union ( ) ( )
println union ( null ) ( null )
println union ( 1 2 3 ) ( 4 5 "6" )
println union ( 1 2 3 ) ( 1 2 3 )
println union ( 1 2 3 ) ( 1 2 3 4 5 "6" )
println union ( 1 2 3 ) ( 4.2 5.3 "6" 3 2 )


[End of: ./test/advanced/union.tata]


[Contents of: ./test/advanced/unless.tata]
```

```
# advanced test of the unless function


unless false
{
          println "i am the terror dome"
}

unless true
{
          println "the EL GREGOR got you fool!"
}

unless true
{
          unless false
                    { println "this won't get printed" }
}

unless ( )
          { println "true" }

unless { } { }

unless null
          { println "true" }

unless 0
          { println "0" }

unless 1
          { println "1" }

unless 0.0
          { println "0.0" }

unless 0.1
          { println "0.1" }

#unless "1"
#         { println 1 }
#else
#         { println null }
#endunless


set "x" 3

unless get "X"
{ assign "x" 0 }

unless get "x"
{ println "ok" }

println get "x"


#can also use a list
unless true
( println "a" println add 1 4 )



[End of: ./test/advanced/unless.tata]


[Contents of: ./test/advanced/unset.tata]
```

```
# Buko O.
# 4.20.2003

# test of the unset function

set "num" 7

println get "num"

unset "num"

println get "num"


set "true" 0
println "$$$$$$$$$$$$$$$$"
unset "true"
println "$$$$$$$$$$$$$$$$"
println add 3 true


set "x" 3
assign "x" ( unset "x" )
println get "x"
{ unset get "x" }
println get "x"

println unset "true"
println unset "unset"
println unset "abc"
#println unset null
println unset ( )
```

[End of: ./test/advanced/unset.tata]

[Contents of: ./test/advanced/url.tata]

```
# advanced test of the url function

set "yahoo.com" url "http://www.yahoo.com"
println get "yahoo.com"

#println set "u1" url "http:" #this is not a valid url.
#println url ( "http:" "abc.com" )
```

[End of: ./test/advanced/url.tata]

[Contents of: ./test/advanced/when.tata]

```
# advanced Test of the when function

when false
{
          println "i am the terror dome"
}

when true
{
          println "the EL GREGOR got you fool!"
}

when true
{
```

```
          when false
                    { println "this won't get printed" }
}

when ( )
          { println "true" }

when { } { }

when null
          { println "true" }

when 0
          { println "0" }

when 1
          { println "1" }

when 0.0
          { println "0.0" }

when 0.1
          { println "0.1" }

#when "1"
#          { println 1 }
#else
#          { println null }
#endwhen


set "x" 3

when get "X"
{ assign "x" 0 }

when get "x"
{ println "error" }

println get "x"


#can also use a list
when true
( println "a" println add 1 4 )


[End of: ./test/advanced/when.tata]
```