Easysurvey

--- an online survey generate language

PLT Project Language Reference Manual

Team Members:

Taotao Li	tl2453
Luyao Shi	ls2899
Zaisheng Dai	zd2140
Yifan Zhang	yz 2365

Professor:

Stephen A. Edwards

TABLE OF CONTENTS

1.Language Description

Online survey can find its importance in areas such as Market Research, Event Planning, Customer Feadback, Product Planning and Education & Training . *EasySurvey* is a light weight language which is designed to make the onlie survey gennereation process easy and fun. *EasySurvey* defines a framework that can allow developer to define basic components of the online survey. User can use "*EasySurvey language*" to define "Page" to represent the every survey page. And each page can have several "*QuestionSet*", each "*QuestionSet*" can have some numbers of "*Question*". Then each question contains its own properties, such as "*Title*" and "*Type*" etc.

2. Input and output

EasySurvey will accept the files with .as extension as the input file; the EasySurvey complier will compile the input file, which will be analyzed in lexically and grammatically. If everything is correct, the compiler will generate the flex page source code **(.mxml).** Then the output source code can be run and tested under Adobe Flash Builder. Then developer can export the application as a package, which can be run in any flash-support environment.

3. Lexicon

3.1 Token separator

White space " ", New line '\n', Carriage return '\r' and Horizontal tab '\t' are the token separators**.**

3.2 Comments

Comments begins with "/*" and end with "*/", including everything between them.

3.3 Constants

Integer: a sequence of one or more digits.

String: a string starts with a double quote " followed by zero or more characters, ended by a double quote

3.4 Identifier

An identifier is defined as a combination of alphanumeric characters [a-z][AZ][0-9] and must start with a alphabet character. Length of an identifier cannot exceed 32.

This language is case sensitive.

3.5 Keywords

if

else

int

new

Main

Question

Title

SingleSelection

MultipleSelcetion

DropdownList

TextField

UserInput

QuestionSet

Display

AddQuestion

setVisible

Image:

JumpButton:

3.6 Separators

3.6.1 ';'

';' is used to separate statements.

For example:

Question question1;

QuestionSet qs1= *new QuestionSet*();

3.6.2 '{' and '}'

'{' and **'}'** are used to separate sets like function declarations , gather the statements in if block or else block, or gathering parameters when assigning the content of the question.

For example:

(1) Questionlogo (*QuestionSet* qs1, *Image* logo)
{ }
Main()
{ } (separating function declarations)

(2) if(qs1.q1.*UserInput*>18)
{ }
else { } (gather statements in if block or else block)

(3) *Question* Address;
Address.*Title*="Where do you live?";
Address.*SingleSelection*={"Bronx","New Jersy","China"}; (assigning the content of the question)

3.6.2 ','

',' is used to separate the parameters and identifier.

For example: The third case above.

3.6.2 '(' and ')'

'(' and **')'** are used to indicate the token before '(' is a function and gather the function parameters. Also, they can be used to contain the condition sentence of **if.**

3.7 Operators

3.7.1 ASSIGN

'=' is used to represent assignment, usually used when constructing a *new QuestionSet*, setting the *Title* of the question or the content of the question.

For example:

question1.*Title*="question1" ;

question1.*SingleSelection*= {"1", "2"};

QuestionSet questionset1=*new Questionset*( );

3.7.2 Dot

'.' is usually behind a union type like *QuestionSet* or *Question*, following the elements of that union type.

For example:

"question1.*Title*", "questionset1.quesstion1.*UserInput*"

3.7.3 Comparison Operator

The comparison operator in EasySurvey includes equal, less than, greater than, less equal and greater equal wich are presented as <, >,<=,>=,== associating from left to right.

Constraint: the left hand side of the operator must be userinput. 7.5 show the definition of comparison operator.

3.8 Attributes

***Title* :**

Title is one of the attribute of the question. It is a string which will show the informant what the questions are.

***SingleSelection*:**

the informant can only choose one of the answers.

***MultipleSelcetion*:**

The informant can choose multiple answers.

***DropdownList*:**

The answers will represent in the form of dropdown list.

TextField:

The informant should type his/her answer in text field.

UserInput :

represent the value user input, every question has one UserInput

3.9 Data Types

***int*:**

An integer is a sequence of digits. All the integer constants are views as decimal.

Sample:

int y;

***Question*:**

Question is used to define the survey questions. It consists of *Title* and *Type* which might be one of *RadioBox, DropdownList*, *MutipleBox*, and *TextArea*.

Sample:

Question Name;

Name.*Title*="What's your name?";

Name.*TextArea*={10}; /* the TextArea occupy 10 characters*

QuestionSet:

QuestionSet is used to define a set of related question and display them as a whole part.

Sample:

QuestionSet qs1 = *new QuestionSet*();

***Image*:**

Claim an image for the use of displaying it.

***JumpButtorn*:**

Jump to next or another page.

3.10 Embeded Action

***Display*:**

Display a series of *QuestionSet* or *Image*.

Sample:

Display Image1; Display Qs1;

***AddQuestion*( ):**

Add a *Question to* a *QuestionSe*t

Sample:

qs1.*AddQuestion*(Name)

***setVisible*( ):**

set the visibility of a *QuestionSe*t. The parameter should be integer, while 0 represent that the question would be hidden, the other integer would show the questionset.

Sample:

Qs1.setVisible(1) **/*Qs1 would now be visible*/**

4. Declaration

decl:

QUESTIONSET ID ASSIGN NEW QUESTIONSET LPAREN RPAREN SEMI
| *datatype var_decl* SEMI

4.1 QuestionSet Declaration

The declaration of *QuestionSet* has the form

QUESTIONSET ID ASSIGN NEW QUESTIONSET LPAREN RPAREN SEMI

For example:

QuestionSet ID =*new QuestionSet*( );

4.2 Other datatype declaration

Other datatype declaration has the form

datatype var_decl SEMI

The *var_decl* contains a list of variable names. The *datatype* has the form

datatype:

QUESTION |IMAGE |JUMPBUTTORN |INT

For example: *Question* q1, q2, q3;

5. Function definition

The function definition has the form

fun_def :

ID LPARAN *args_list* RPARAN LBRACE *decl_list stmt_list* RBRACE

For example:

QuesionLogo(*QuestionSet* qs1, *Image* logo)

{ …..}

The *args_list* is a list of arguments.

For example: int a, Question q1, QuestionSet qs1

The *decl_list* is a list of declarations.

The *stmt_list* is a list of statements.

6. Statements

6.1 Common statements

The common statements are composed with expression and a semicolon.

6.2 Block statements

The block statements have the form

stmt:

LBRACE *stmt_list* RBRACE

stmt_list is a list of statements.

6.3 If statements

If statements have the form

stmt:

IF LPAREN *expr* RPAREN *stmt* %prec NOELSE
|IF LPAREN *expr* RPAREN *stmt* ELSE *stmt*

7. Expressions

7.1 Simple expressions

The simple expressions include int constant or string constant and return their value.

7.2 Assignment expressions

The assignment expressions have the form

expr:

ID POINT MULTIPLE ASSIGN LBRACE *string_opt* RBRACE

(Assign the content of the question ID, noting the type of the question is *MutipleSelection*)

|ID POINT SINGLE ASSIGN LBRACE *string_opt* RBRACE
(Assign the content of the question ID, noting the type of the question is *SingleSelection*)

|ID POINT DROP ASSIGN LBRACE *string_opt* RBRACE
(Assign the content of the question ID, noting the type of the question is *DropdownBox*)

|ID POINT TEXT ASSIGN LBRACE INT_LITERAL RBRACE
(Assign the width of the text question)

7.3 Special expressions

Adding a *Question* to a *QuestionSet* is a special expression. It has the form

expr:

ID POINT ADDQUESTION LPARENT *var_decl* RPARENT

The *var_decl* is a list of variables(ID).

Displaying a series of *QuestionSet* is a special expression. It has the form

expr:

DISPLAY *var_decl*

Setting the visibility of a QuestionSet is a special expression. It has the form

expr:

ID POINT SETVISABLE LPARENT INT_LITERAL RPARENT

7.4 Call expressions

They have the form

expr:

ID LPAREN *var_opt* RPAREN

7.5 Condition expressions

They have the form

expr:
ID POINT ID POINT USERINPUT EQ *expr*
|ID POINT ID POINT USERINPUT NEQ *expr*
|ID POINT ID POINT USERINPUT LT *expr*
|ID POINT ID POINT USERINPUT LEQ *expr*
|ID POINT ID POINT USERINPUT GT *expr*
|ID POINT ID POINT USERINPUT GEQ *expr*

Normally, to get a *UserInput*, the developer should provide the *UserInput* of which *Question* in which *QuestionSet*. That's why the form has two "ID POINT".

8. Scope

Global variables have a global scope. The scope of other variables is in their function declaration block.

9. Conditionals

Conditionals are used to determine the order of questions shown to readers. They are defined in the following form.
If (expression of ***UserInput***) statement **else** statement
The else part is necessary, because it will decide which *questionset* to be *display*.

For example:

```
if ( qs1.IsOnCampus.UserInput == "OnCampus")
{
qs2.setVisible(1);
}

else {
qs3.setVistible(1);
}
```

10. Code snippet:

```
/**********function declaration*********/

QuestionLogo(QuestionSet Qs1, Image imag1)
{
  Display Qs1,
  Display image1;
}

/**********main function<***********/

Main()
{
    /*variable declaration*/

     Question Name;
     Question Address;
     Question IsOnCampus;
     Question Fruit;
     Question OnCampus;
    Question OffCampus:


     Image logo;

    /********Assign value to variation********/

     logo ="C:\EasySury\image1x.jpg";

     Name.Title="What's your name?";
     Name.TextField=(10);

     IsOnCampus.Title="Are you living on Campus?";
     IsOnCampus. SingleSelection={"off campus","on cmpus"};

     Address.Title="Where do you live?";
     Address.SingleSelection{"Bronx","New Jersy","China"};

     Fruit.Title="Which is/are your favorite fruit?";
     Fruit. MultipleSelection={"Apple", "Pear", "Banana"};
```

```
    OnCampus.Title="Which area?";
    OnCampus. SingleSelection{"MorningSide","WestWood"};

    OffCampus.Title="Which district?";
    Offcampus.SingleSelection{"Manhattan", "Queens", "Bronx", "Brooklyn"}


    /********questionsets declaration and assignment*******/


    QuesttionSet qs1=new Questionset();
    qsl.AddQuestion(Name,Address,Fruit);
    qs1.AddQuestion(IsOnCampus);

  /*******display the questionset********/
    Diplay qs1 ;


 /*********function call************/

   QuestionLogo(qs1,logo) ;


    QuestionSet qs2=QuestionSet();
    qs2.AddQuestion(OnCampus);
    QuestionSet qs3=QuestionSet(  );
    qs3.AddQuestion(OffCampus);

    /*Condition example, we will base on the users' input to show the next input.
      If the informant live on campus, we will display the questionset related to live on campus.
  */

if ( qs1.IsOnCampus.UserInput == "OnCampus"){

qs2.setVisible(1);
}

else {
qs3.setVistible(1);

}

}
```

11. Syntax Summary

program:
| *program decl*
| *program fun_def*

fun_def : ID LPARAN *args_list* RPARAN LBRACE *decl_list stmt_list* RBRACE

args_list :
| def_args
| args_list COMMA def_args

def_args: *datatype* ID | QUESTIONSET ID

decl_list:
| *decl_list* *decl*

decl:
QUESTIONSET ID ASSIGN NEW QUESTIONSET LPAREN RPAREN
| *datatype var_decl* SEMI

var_opt:
|*var_decl*
var_decl :
|*var*
|*var_decl* COMMA *var*
var : ID

datatype:
QUESTION
|IMAGE
|JUMPBUTTORN
|INT

stmt:
expr SEMI
|LBRACE *stmt_list* RBRACE\
|IF LPAREN *expr* RPAREN *stmt* %prec NOELSE
|IF LPAREN *expr* RPAREN *stmt* ELSE *stmt*

expr:
INT_LITERAL
| STRING_LITERAL |ID POINT MULTIPLE ASSIGN LBRACE *string_opt* RBRACE
| ID POINT SINGLE ASSIGN LBRACE *string_opt* RBRACE
| ID POINT DROP ASSIGN LBRACE *string_opt* RBRACE
| ID POINT TEXT ASSIGN LPAREN INT_LITERAL RPAREN

| ID POINT TITLE ASSIGN STRING_LITERAL
| ID ASSIGN STRING_LITERAL
| ID POINT ADDQUESTION LPARENT *var_decl* RPARENT
| ID POINT SETVISABLE LPARENT INT_LITERAL RPARENT
| DISPLAY *var_decl*
| ID LPAREN *var_opt* RPAREN
| ID POINT ID POINT USERINPUT EQ *expr*
| ID POINT ID POINT USERINPUT NEQ *expr*
| ID POINT ID POINT USERINPUT LT *expr*
| ID POINT ID POINT USERINPUT LEQ *expr*
| ID POINT ID POINT USERINPUT GT *expr*
| ID POINT ID POINT USERINPUT GEQ *expr*

string_list:
STRING_LITERAL
| string_list COMMA STRING_LITERAL
string_opt:
|*string_list*

stmt_list:
|*stmt_list stmt*