WSL Specification

v0.4

Abstract

WSL is the acronym for Web Scripting Language, that's a web-oriented language that allow to automatize the common web operations, such as sending emails, sending sms and all the other possible ones. It could be thought as a code generator for KMS, that is, for a code-driven web browser.

Scanner-related specification

Identifiers

An identifier is defined in the following way:

identifier ({literal}|_)({literal}|{digit}|_)*

Where literal and digit are generated by the following regular expressions:

digit [[:digit:]]
literal [[:upper:][:lower:]]

The identifiers' handling is similar to the c++ one, that is, an identifier is an alpha-numeric word without spaces or other simbols, beginning with a literal or underscore, and with a length of, al least one character.

Strings

A quoted string is defined in the following way:

quotedString \"([^"\\]|\\\\|\\\")*\"

The quoted strings are defined as a quote followed by a set of 0 or more of every character except \ , and then followed by another quote .
The following escape sequences are allowed inside the quoted string:

\\ interpreted as \
\ interpreted as

Comments

The single-line and multi-line comment are defined in the following way:

single-line comment //.*$
multi-line comment /* ... */

The comments' handling is similar to the c++ one.
Could be used both single-line then multi-line comments.

E.g.

```
// This is a single-line comment

/*
 * In this multi-line comment I say you that...
 * ...WSL compiler is most powerful then gcc!
 */
```

Keywords

The used keywords are the following.
Some keywords are lowercased (with java-like conventions) becouse they are used in conjunction with the uppercased ones.

SERVICE	PROCEDURE	RETURN	IF
MATCHES	ELSE	ENDIF	CYCLE
BREAK	CONTINUE		
INPUT	content	type	regex
name	description	defaultValue	
icon	version	author	
REQUEST	SHOW	WAIT	EXIT
onTrue	onTrueOutput	onFalse	onFalseOutput
+	-	*	/
%	=	==	!=

Parser-related specification

Grammar

The parser used for WSL is an LALR(1), with the following grammar:

```
start                   : newlines service newlines inputs_or_procedures
                        ;
newlines                :
                        | NEWLINE newlines
                        ;
service                 : SERVICE IDENTIFIER NEWLINE service_attributes
                        ;
service_attributes      :
                        | NAME ASSIGNMENT_OP QUOTED_STRING NEWLINE service_attributes
                        | ICON ASSIGNMENT_OP QUOTED_STRING NEWLINE service_attributes
                        | DESCRIPTION ASSIGNMENT_OP QUOTED_STRING NEWLINE service_attributes
                        | VERSION ASSIGNMENT_OP QUOTED_STRING NEWLINE service_attributes
                        | AUTHOR ASSIGNMENT_OP QUOTED_STRING NEWLINE service_attributes
                        ;
inputs_or_procedures    :
                        | input newlines inputs_or_procedures
                        | procedure newlines inputs_or_procedures
                        ;
input                   : input_token attribute NEWLINE
                        ;
input_token             : INPUT IDENTIFIER NEWLINE
                        ;
attribute               :
                        | CONTENT ASSIGNMENT_OP QUOTED_STRING NEWLINE attribute
                        | TYPE ASSIGNMENT_OP QUOTED_STRING NEWLINE attribute
                        | REGEX ASSIGNMENT_OP QUOTED_STRING NEWLINE attribute
                        | NAME ASSIGNMENT_OP QUOTED_STRING NEWLINE attribute
                        | DESCRIPTION ASSIGNMENT_OP QUOTED_STRING NEWLINE attribute
                        | DEFAULT_VALUE ASSIGNMENT_OP QUOTED_STRING NEWLINE attribute
                        ;
procedure               : procedure_token block RETURN NEWLINE
                        ;
procedure_token         : PROCEDURE IDENTIFIER NEWLINE
                        ;
block                   : newlines
                        | assignment newlines block
                        | command newlines block
                        | matches newlines block
                        | cycle newlines block
                        ;
assignment              : IDENTIFIER ASSIGNMENT_OP stuff NEWLINE
                        ;
stuff                   : identifier
                        | identifier PLUS_OP stuff
                        ;
identifier              : IDENTIFIER
                        | QUOTED_STRING
                        | INTEGER
                        ;
command                 : REQUEST_CMD text NEWLINE
                        | REQUEST_CMD text text NEWLINE
                        | IDENTIFIER ASSIGNMENT_OP REQUEST_CMD text NEWLINE
                        | IDENTIFIER ASSIGNMENT_OP REQUEST_CMD text text NEWLINE
```

```
                        | EXIT_CMD NEWLINE
                        ;
matches                 : matches_token event
                        ;
matches_token           : MATCHES IDENTIFIER text NEWLINE
                        ;
event                   :
                        | OT  ASSIGNMENT_OP BREAK NEWLINE event
                        | OT  ASSIGNMENT_OP CONTINUE NEWLINE event
                        | OTO ASSIGNMENT_OP QUOTED_STRING NEWLINE event
                        | OF  ASSIGNMENT_OP BREAK NEWLINE event
                        | OF  ASSIGNMENT_OP CONTINUE NEWLINE event
                        | OFO ASSIGNMENT_OP QUOTED_STRING NEWLINE event
                        ;
cycle                   : cycle_token newlines block BREAK NEWLINE
                        ;
cycle_token             : CYCLE int_expression NEWLINE
                        | CYCLE NEWLINE
                        ;
text                    : string
                        | string PLUS_OP text
                        ;
string                  : QUOTED_STRING
                        | IDENTIFIER
                        ;
int_expression          : number
                        | int_expression PLUS_OP int_expression
                        | int_expression MINUS_OP int_expression
                        | int_expression DIVIDE_OP int_expression
                        | int_expression MULTIPLICATION_OP int_expression
                        | int_expression PERCENT_OP int_expression
                        ;
number                  : INTEGER
                        | IDENTIFIER
                        ;
```

Examples

Sending an MMS using vodafone web site

There, a simple example:

```
/*
 * www.vodafone.it-mms.wsl
 */

/*
 * SERVICE defines a service
 *
 * SERVICE <The identifier of this service>
 *          name = <The friendly name this appears to the user, used by the user interface>
 *          icon = <The icon associated to this service, used by the user interface>
 *          description = <The description of this service>
 *          version = <The version of this service>
 *          author = <The author of this service>
 */
SERVICE www_vodafone_it_mms
```

```
	name = "www.vodafone.it (MMS)"
	icon = "./icons/www.vodafone.it-mms.svg"
	description = "Allow to send infinite MMSs to vodafone phone numbers"
	version = "0.1"
	author = "scrockman"

/*
 * INPUT defines an input
 *
 * INPUT <The identifier of this input>
 *		content = <The content of this input (see kms specification for more details)>
 *		type = <The type of this input (see kms specification for more details)>
 *		regex = <The regex to check the correctness of the input, used by the user interface>
 *		name = <The friendly name of this service, used by the user interface>
 *		description = <The description of this service, used by the user interface>
 *		defaultValue = <The default value of that input, used by the user interface>
 */
INPUT subjectInput
	content = "string"
	type = "variable"
	regex = ".{16}"
	name = "Subject"
	description = "The subject of the message"
	defaultValue = "Da_"

INPUT recipientInput
	content = "string"
	type = "variable"
	regex = "+39\\d{10}"
	name = "Recipient"
	description  = "The recipient of the message"
	defaultValue = "+39"

INPUT textInput
	content = "string"
	type = "variable"
	regex = ".{8000}"
	name = "Text"
	description = "The text of the message"
	defaultValue = ""

/*
 * PROCEDURE defines a procedure
 *
 * PROCEDURE <The identifier of this procedure>
 */
PROCEDURE sendProcedure

	/* Integer assignments */
	int1 = 10
	int2 = 5 + int1

		/*
		 * CYCLE repeats a code-block a specified-value times
		 *
		 * CYCLE <integer>
		 */
		CYCLE 5 + int2

			port = 8080 // An integer assignment
			page = "/WebComposer/web/elaborapop.jsp" // A string assignment
			url1 = "https://mmsviaweb.net.vodafoneomnitel.it:" + port + page
			url2 = "https://mmsviaweb.net.vodafoneomnitel.it" + page
			data = "subjecttosend="+subjectInput+"&TextName="+textInpu +"&recipient="+recipientInput
			+"&SmilName=&ImageName=&AudioName="
```

```
                    /*
                     * REQUEST does a web request
                     *
                     * REQUEST <uri> [<post_data>]
                     * Note that the request method (HTTP/HTTPS is determined throught the <uri>)
                     */
                    REQUEST url1
                    uri1 = REQUEST "https://mmsviaweb.net.vodafoneomnitel.it/WebComposer/web/elaborapop.jsp"
                    "subjecttosend="+subjectInput+"&TextName="+textInput+"&recipient="+recipientInput
                    +"&SmilName=&ImageName=&AudioName="

                    uri2 = REQUEST "https://mmsviaweb.net.vodafoneomnitel.it/WebComposer/web/elaborapop.jsp"
                    regex1 = "Il tuo messaggio è stato inviato"

                    /*
                     * MATCHES checks if a uri matches a given regex
                     *
                     * MATCHES <uri> <regex>
                     *          onTrue = CONTINUE | BREAK
                     *          onFalse = CONTINUE | BREAK
                     *          onTrueOutput = <quotedString>
                     *          onFalseOutput = <quotedString>
                     */
                    MATCHES uri1 regex1
                              onTrue = BREAK
                              onFalse = CONTINUE
                              onTrueOutput = "Message sent with success"
                              onFalseOutput = "Message not sent"

          /* BREAK exits from the current cycle */
          BREAK

/* RETURN exits from the current procedure */
RETURN
```