**SELMATE**: A Test Automation Tool based on Selenium

AVIJIT BASAK

DEBAYAN DAS

ARUP DEY

ARGHYADIP GHOSH

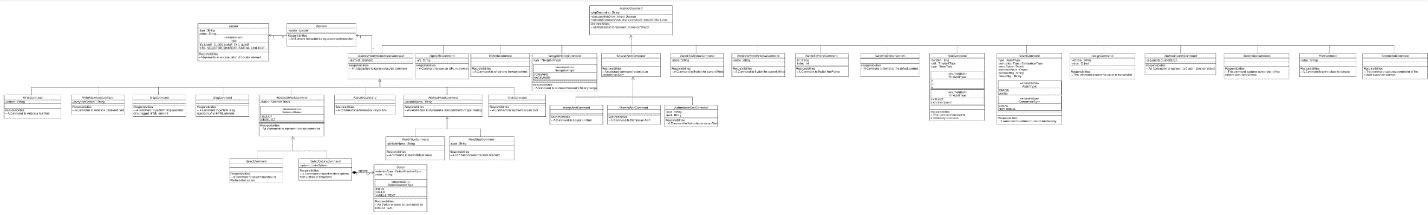
**Introduction:** Automation of Regression testing has become very common practice in any enterprise application development using Agile methodology. Selmate is developed as an extensible and customizable tool to automate regression testing for Web applications. Selmate provides its own scripting feature based on spreadsheet/xml and relevant API in Java for external communication. Selenium WebDriver is used internally by Selmate for all browser communication.

In current era, most of the software applications are web based due to better accessibility and portability. With the increasing popularity of Agile methodology in application development the requirement of test automation for regression testing is growing day by day. Selenium has been a very popular tool for test automation of Web application due to its user-friendly programming interfaces in multiple languages like C#, Groovy, Java, Perl, PHP, Python, Ruby and Scala. But automation using selenium requires good amount of programming expertise in any supported language. Selmate is an endeavor to minimize the programming effort required for test automation using Selenium. It provides a user-friendly way of script writing using spreadsheet and XML where each user interaction with browser is represented as command. A programmable API for automation using dynamic data is also provided. Spreadsheet would be used as an input for automation of regression testing and the Selmate API would be used for automation of dynamic process like administrative jobs using dynamic data.

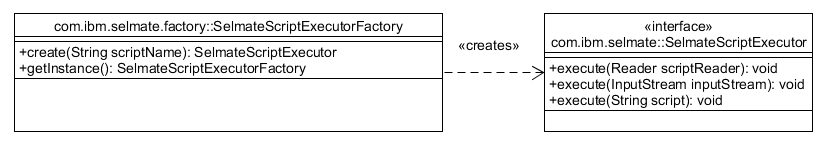
**Architecture:** Architecturally Selmate can be divided into two principal components, core component and the spreadsheet/XLS adapter. The core component accepts input as xml of a predefined schema. The schema contains a description of commands and the corresponding input and is based on a predefined command hierarchy. The core component uses the input xml to generate set of internal Selmate commands and execute them sequentially. XLS adapter component provides a spreadsheet based interface which converts the spreadsheet to input XML for communicating with the core component. The spreadsheet should contain a script of multiple commands in a predefined template.

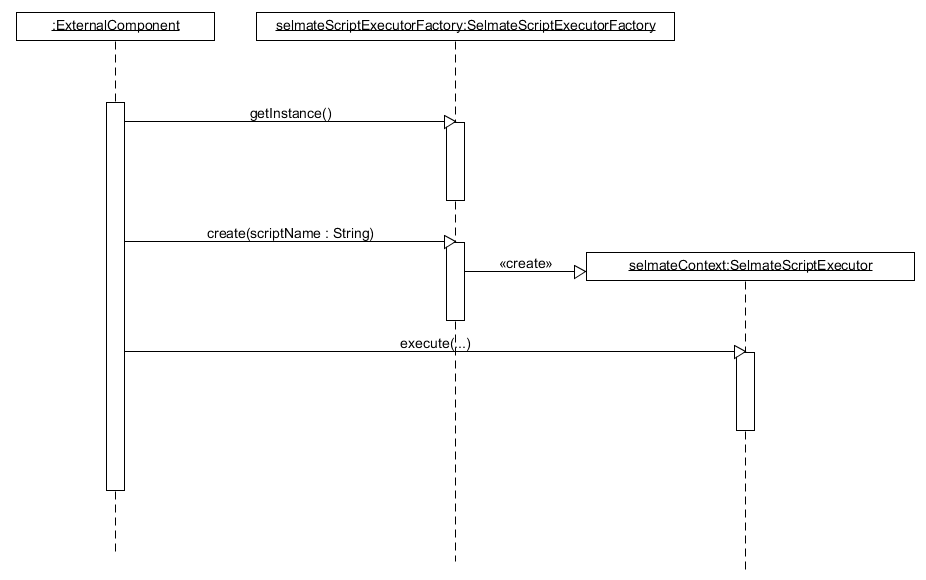
**Basics of Commands:** Selmate is based on a command driven architecture. Each user interaction with web browser is represented as a command and each command is atomic in nature. Commands also accept one or more inputs. The commands are of different types. Some need to interact with the web elements in a web page while some does not. Commands like OpenURLCommand, RefreshCommand do not need to interact with HTML elements while WriteCommand, ReadCommand etc interacts with HTML elements in web page. The relevant web elements are usually located by either of ID, XPATH, CLASSNAME, TAGNAME or CSS\_SELECTOR. In addition to that link element can also be located by LINKTEXT or PARTLINKTEXT. The diagram below depicts the conceptual command hierarchy in Selmate.





**Selmate-core:** This component provides the core functionality of Selmate i.e. communicating with the browser through proper interfaces and commands. It exposes basic interfaces for communication for the external components. The principal interface exposed for communication is SelmateScriptExecutor. This interface declares three overloaded operations for executing Selmate script. A factory class SelmateScriptExecutorFactory is also provided to create a SelmateScriptExecutor. The basic component interfaces, their relations and way to interact with them is illustrated below:





The core component also provides a SelmateScriptExecutorClient class which implements the mentioned functionality. This class exposes a main method for users. The command to invoke this class would be like this:

java -D<DRIVER VM ARG NAME>="<<driver file path>>" -jar selmate-core<<version>>.jar com.ibm.selmate.client.SelmateScriptExecutorClient --file <<input xml file path>>

The input xml should follow the schema attached below:



**Selmate Spreadsheet/XLS Adapter:** This component works as an adapter to Selmate-core component. Users using the spreadsheet format of the Selmate script should be using this adapter. This component accepts a script in predefined spreadsheet template, transforms it into a script of xml format which is suitable for core component and invoke the SelmateScriptExecutor class for further processing. This component provides a class XLSScriptExecutor containing main method. The required command to invoke this component would be like this:

java -D<DRIVER VM ARG NAME>="<<driver file path>>" -jar selmate-xls<<version>>.jar --file <<input csv file path>>

The template to write the Selmate script is attached below. The description and purpose of each of the columns are mentioned below w.r.t headers.



COMMAND: This column contains the predefined command name. The value in this column is mandatory and should be one of the values as defined inside the template.

COMMENTS: This column contains the relevant comments which would be ignored by Selmate. The value of this column is optional.

NARRATION: This column contains the description of the user operation performed by the command. This value would be present in the generated report as step description. The content of this column is optional.

VAR NAME: This column contains variable name if required for any command.

LOCATOR TYPE: This column contains type of locator for the corresponding command element. The set of allowable locator types for each command are mentioned in the Template spreadsheet. The content in this column is mandatory if the command interacts with the HTML elements in a web page.

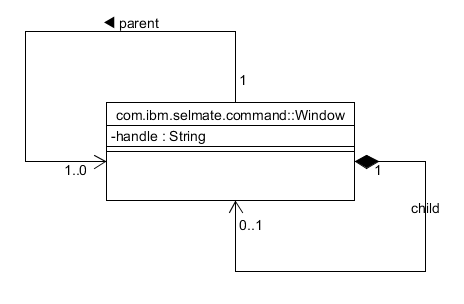
LOCATOR VALUE: This column contains the value of locator for the required element. This value should be in correlation with LOCATOR TYPE column content. The value in this column is mandatory for commands interacting with HTML elements in a web page.

INPUT VALUE: This column contains the input value for the corresponding command. There could be any no of columns containing the INPUT VALUE.

**Command Descriptions:** The descriptions and usage for all the commands are described below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Command Name** | **Spread Sheet Command** | **XSD Command Type Name** | **Functionality** |
| OpenURLCommand | OPEN | OpenURLCommandType | This command would be used to open a specified URL in a browser window. The URL is passed as an input argument. The supported schemes for URL are http/https and file. |
| CloseCommand | CLOSE | CloseCommandType | This command would be used to close the currently active window. |
| RefreshCommand | REFRESH | RefreshCommandType | This command refreshes the current webpage in a browser window. |
| NavigateHistoryCommand {type = FORWARD} | HIST\_FWD | NavigationHistoryCommandType {type = FORWARD} | This command should be used for navigating to next page according to browsing history in a browser window. |
| NavigateHistoryCommand {type = BACKWARD} | HIST\_BKWD | NavigationHistoryCommandType {type = BACKWARD} | This command should be used for navigating to previous page according to browsing history in a browser window. |
| SwitchToWindowCommand | SWITCH\_WINDOW | SwitchToWindowCommandType | This command is responsible for switching the execution from one browser window to another. This command should be used to change control from parent window to child window. Selmate window handling is described later. |
| SwitchToParentWindow | SWITCH\_PARENT\_WINDOW | SwitchToParentWindowCommandType | This command switches the control from current window to the parent window. The detail of Selmate window handling is described later. |
| SwitchToFrameCommand | SWITCH\_FRAME | SwitchToFrameCommandType | This command is responsible for switching the execution from one iFrame to another. Index or id identifies a frame. The identifier type is mentioned in column LOCATOR TYPE and the corresponding value is mentioned in LOCATOR VALUE in spreadsheet format. |
| SwitchToDefaultContent | SWITCH\_TO\_DEFAULT  \_CONTENT | SwitchToDefaultContentCommandType | This command is used while dealing with multiple frames in a page. It switches the control to the default content of the application. |
| AcceptAlertCommand | ACCEPT\_ALERT | AcceptAlertCommandType | This command is used for accepting an ACCEPT/REJECT alert in a browser window. |
| DismissAlertCommand | DISMISS\_ALERT | DismissAlertCommandType | This command is responsible for dismissing an alert in a browser window. |
| AuthenticateAlertCommand | AUTHENTICATE | AuthenticateAlertCommandType | This command is responsible for authenticating an alert using proper username and password in a browser window. |
| AssignCommand | ASSIGN | AssignCommandType | This command is responsible for assigning a specified value to a variable. The name of the variable and the corresponding value are passed as arguments. The acceptable pattern for a variable name is “[a-zA-Z\_0-9]+”. Once a variable is defined it can be referred by other commands to use its value. |
| WaitCommand {type = INTERMEDIATE} | DELAY | WaitCommandType {type = INTERMEDIATE} | This command is responsible for introducing a DELAY between two commands. The amount of delay is passed as an argument. The default unit of duration is second. By default, Semate imposes a delay of one second between two successive commands. This command should be used when the default delay is not sufficient. |
| WaitCommand {type = INTERMEDIATE} | WAIT | WaitCommandType {type = INTERMEDIATE} | This command is responsible for introducing an implicit WAIT for each of the Selmate commands. The amount of delay is passed as an argument. The default unit of duration is second. This WAIT time should be effective only for commands interacting with web elements. By default, each command waits 120 secs for an element to appear in browser. Once a new WAIT time is defined it would be applicable for all subsequent commands. |
| AssertCommand {type = ERROR, comparisonType = EQUAL} | ASSERT\_EQ | AssertCommandType {type = ERROR, comparisonType = EQUAL} | This command compares actual and expected value. Both are passed as input arguments. Optionally a failure message can also be passed as an argument. If the values are equal assert would return true otherwise false. If the assert fails it would stop the execution of the script. |
| AssertCommand {type = ERROR, comparisonType = NOT\_EQUAL} | ASSERT\_NEQ | AssertCommandType {type = ERROR, comparisonType = NOT\_EQUAL} | This command compares the expected and actual values. It would return true if the values are **not** equal otherwise false. If the assert returns false, the script execution would be stopped. |
| AssertCommand {type = ERROR, comparisonType = EQUAL} | ASSERT\_EQ\_WARN | AssertCommandType {type = ERROR, comparisonType = EQUAL} | This command compares actual and expected value. Both are passed as input arguments. If the values are equal assert would return true otherwise false. If the assert fails it would create a warning message and script execution would continue. |
| AssertCommand {type = WARN, comparisonType = NOT\_EQUAL} | ASSERT\_NEQ\_WARN | AssertCommandType {type = WARN, comparisonType = NOT\_EQUAL} | This command compares the inequality of the expected and actual values. It would return true if the values are not equal otherwise false. If the assert returns false, it would create a warning message and script execution would continue. |
| WriteCommand | WRITE | WriteTextCommandType | This command is responsible for writing text in an editable text field in a browser window. The content of text and the element locator are passed as arguments. |
| WritePasswordCommand | WRITE\_PWD | WritePasswordCommandType | This command is responsible for writing password in an editable password field in a browser window. Written characters will be displayed as ‘\*’. This command requires the encrypted password and html field identifier as arguments. Encryption should be done using SelmatePwdHandler as described later. |
| DragCommand | DRAG\_FROM | DragCommandType | This command is used to drag a WebElement. The element would be identified by element identifier as passed by input argument. The element instance is stored as a Selmate context variable named “DRAG\_DROP\_VARIABLE\_NAME”. |
| DropCommand | DROP\_TO | DropCommandType | This command is used to drop an element to a html tree structure. The element is fetched from the variable named “DRAG\_DROP\_VARIABLE\_NAME”. |
| SelectCommand {status = SELECT} | SELECT | SelectCommandType {status = SELECT} | This command is responsible for selecting one option for Radio button, Check box etc. in a browser window. The element to select is specified as input argument with proper identifier. |
| SelectCommand {status = DESELECT} | DESELECT | SelectCommandType {status = DESELECT} | This command is responsible for de-selecting one option for Radio button, Check box etc. in a browser window. The element to de-select is specified as input argument with proper identifier. |
| SelectOptionsCommand {status = SELECT} | SELECT | SelectOptionsCommandType {status = SELECT} | This command is responsible for selecting one or more options for list box/dropdown in a web page. The element to select is identified by proper identifier passed as an argument. The option would be identified by index, visible text or value. The spreadsheet adapter supports option selection by visible text only. |
| SelectOptionsCommand {status = DESELECT} | DESELECT | SelectOptionsCommandType {status = DESELECT} | This command is responsible for de-selecting one or more options for list box/dropdown in a web page. The element to de-select is identified by proper identifier passed as an argument. The option would be identified by index, visible text or value. The spreadsheet adapter supports option selection by visible text only. |
| ActivateCommand | ACTIVATE | ActivateCommandType | This command is used to click or activate a hyperlink. The hyperlink is identified by proper element identifier as mentioned in the spreadsheet template. |
| ReadValueCommand | READVALUE | ReadValueCommandType | This command reads value from an html element. The element is identified by proper identifier. If XPATH is used as element identifier, then the related element text would be read as value. If any other type of identifiers is used, then the corresponding attribute name is to be specified, whose value is to be fetched. |
| ReadStateCommand | READSTATE | ReadStateCommandType | This command is used to read state of an element. The element is identified by proper identifier. Three kinds of states are usually read i.e. ENABLED, SELECTED and DISPLAYED. |
| ClickCommand | CLICK | ClickCommandType | This command represents a user click. This can be applied on any html element identified by proper identifier. |
| ScreenshotCommand | SCREENSHOT | ScreenshotCommandType | This command is responsible for taking a screenshot of the current browser window. |
| PrintCommand | PRINT | PrintCommandType | This command prints the input value in console as well as in generated report. The value can refer to variables previously defined. This should be mostly used for debugging purpose. |
| AbstractCustomCommand | CUSTOM\_ACTION | CustomCommandType | This command represents a custom user activity which are not covered by Selmate basic commands. User must implement a custom command. The name of the command class which implement this feature is passed as an argument to this command as described in the template. Other input arguments required for this custom activity are also passed through this command. |

**Selmate Window Handling:** Browser windows are managed by Selmate with respect to the generated window handle. Selmate uses a class Window to represent a browser window including main window and popups. It maintains a chain of windows opened hierarchically in parent child relation. Each window maintains a reference of its parent window as well as it’s child window which is opened from this window. The main browser window doesn’t have any reference to parent window. Window can refer to one and only one child window at one time. The script must be managed carefully so that from one window (e.g. WinA) only one child window (e.g. WinB) should be opened at a time to perform all the necessary operations and should be closed after that. As part of its operation WinB can open another window (e.g. WinC) and that one can open another one (e.g. WinD) as per the requirement. There is no limit to the no of window descendants but there cannot be any siblings for any of them in open state. Selmate provides two commands to switch focus between multiple windows. Once a new child window is opened SwitchToWindowCommand should be used to shift the control to new window. Then once the user operations are finished in this new window and the window is closed SwitchToParentWindowCommand should be used to move focus/control to the parent window. The UML diagram below illustrates the hierarchy concept of window. Users can access these window instances using SelmateContext interface as described later.



**Selmate Report Structure:** Selenium generates two kinds of reports, static HTML report and video recording of execution. The base report file is an xml file names report.xml. Selmate also generates a predefined stylesheet report.xsl which is referred by the xml file to format the same. The report.xml file can also refer to some images taken as screenshot during script execution. report.xml file can be opened in the browser to display the report as an HTML file. The stylesheet can be customized to format the report in any way. A video recording file named RECORDING.mov is also generated which portrays the entire process execution.

**Example Usage of Selmate:** Selmate can be used in two ways i.e. either by XLS adapter for automation of regression testing or by selmate-core programming interface.

**Selmate XLS Adapter:** Some examples are listed below using XLS adapter for different types of user interaction using Selmate XLS adapter. XPATH is used as element locator. The generated report is also added for reference.

**Browser Interaction:** The script presented in this section demonstrates use of Selmate commands while performing basic navigation of web pages in browser. The script first opens the webpage <http://www.calculator.net/> using the command OPEN. Second command navigates to another page for Interest Calculation by clicking on the respective hyperlink “Interest Calculator”. We can navigate to the previous page i.e. home page by using the command HIST\_BKWD. The command HIST\_FWD is used again to come back to the Interest Calculator page.

**Script:**



**Report:**



**Text Box Interaction:** The script mentioned below is used to write a number into a text field. The script opens the online calculator using the OPEN command. Then it navigates to the page for percentage calculation by clicking on a hyperlink. The number 100 and 25 is entered to the respective textboxes. Then the Calculate button is clicked. The result is read from the display and assigned to the variable “VALUE”. The value of the variable is shown in the report by PRINT command. The same is also checked for equality using the Assert command. The respective screenshot is taken as proof of execution. The relevant report is also attached.

**Script:**



**Report:**



**Radio Button Interaction:** The script mentioned below is used to demonstrate radio button and text box interaction using Selmate. First the calculator website is opened in browser using OPEN command. The it navigates to Mortgage Payoff Calculator page. Out of repayment options the “Payback altogether” option is selected using the SELECT command. Then the Original Loan amount, Original Loan Terms and Remaining Terms are entered as 200000, 20 and 15 respectively. The Calculate button is clicked using the ACTIVATE command. The relevant screenshots are also captured.

**Script:**



**Report:**



**Checkbox Interaction:** The script mentioned below is used to test checkbox interaction. First the calculator website is opened in browser. The “Mortgage Calculator” page is visited. The checkbox “Include Options below” is unchecked. The “Home Price” is entered as 200000. Then the Calculate button is clicked. The relevant screenshots are captured.

**Script:**



**Report:**



**Dropdown Interaction:** The following script demonstrates the use of Selmate commands for using drop down. The calculator website is first opened in browser. Then the browser window navigates to “Interest Calculator” page. The dropdown “Compound” is set as “quarterly” and Calculate button is clicked. The corresponding report is also added here.

**Script:**



**Report:**



**Drag & Drop Interaction:** The script presented below demonstrates the use of Selmate commands for drag & drop type of user interaction. Initially the website page is opened in browser. The element “Disabled Node” under “Contextual Menu with Drag & Drop” is to be moved up in tree structure by one level. So, the element is first dragged by DRAG\_FROM command and stored in a variable named ELEMENT. The same element is placed as child of “Parent Node” using the DROP\_TO command. The variable named ELEMENT is referred in the DROP\_TO command as input to refer to the dragged WebElement.

**Script:**



**Report:**



**ListBox Interaction:** The script presented in this section shows a listbox interaction with multiple selection options. A sample html file from local file system is used for showing this behavior. The local html file is opened in browser by OPEN command. The URL scheme should be “file”. The command SELECT\_OPTION is used to select two separate options in the listbox. The script and the report are mentioned here.

**HTML File:**



**Script:**



**Report:**



**Read Status of HTML field:** This script presented here demonstrates the process of reading HTML element states. A sample HTML file from local file system is used to show the command behavior. The file is opened by OPEN command with local URI. The command READSTATE is used to read states of different html elements for all three supported states SELECTED, DISPLAYED and ENABLED.

**HTML File:**



**Script:**



**Report:**



**Interaction with Browser Alert:** The script presented below shows the basic user interaction i.e. accept and reject with html Alert. A sample html file is opened from local file system which displays the Alert window on load, which can be accepted or rejected. On acceptance it shows message True while on rejection it shows the message False. The generated report is also attached.

**HTML Input:**



**Script:**



**Report:**



**Assertion of Values:** The ASSERT commands are used to perform comparison of expected and actual values. The expected value is usually assigned to a variable by ASSIGN command. The actual value is usually read from the web page using READVALUE command and assigned to another variable. These two values are compared with each other using either ASSERT\_EQ for equality or ASSERT\_NEQ for inequality commands. If the assert fails it would stop further execution of the script. Selmate also provides assert command of type warning which does not stop further processing of script and only produces a warning message. A sample script is provided to demonstrate the use of ASSERT commands.

**Script:**



**Report:**



**Interaction with Child Window:** Selmate can be used to work with one or many child windows present in a web page. Windows are handled in a hierarchical way. A sample script is provided to demonstrate the opening and closing of child window. Once a child window is opened the control needs to be switched to the newly opened window using SWITCH\_WINDOW command. Then any command executed should be applicable on this child window. Once the operations are done, the child window can be closed by CLOSE command. Once the window is closed, the control needs to be switched to the parent window using SWITCH\_PARENT\_WINDOW as described in the script below.

**HTML Input:**



**Script:**



**Report:**

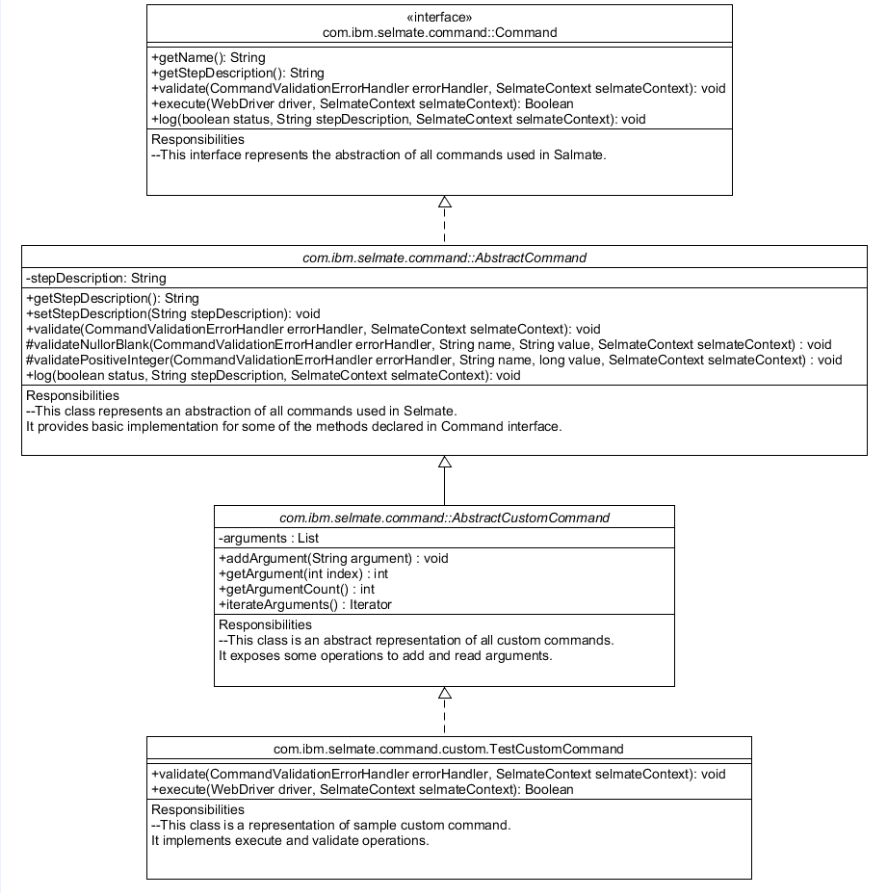


**Selmate Core Component Usage:** The program attached below demonstrates the use of selmate core component for interacting with web browser. We have simulated the example of textbox interaction using core component interface. To work with core component user needs to create a java project with suitable name. Incorporate selmate-core.jar as library. Then create and write the java file as attached below.



**Customization of Selmate:** Selmate core component is customizable as per application requirements. Two kinds of customizations are possible in Selmate, creation of custom command and customization of default WebDriver.

**Custom Command Development:** The most common form of customization would be writing Custom commands for specific application. Selmate provides commands for most common user actions required in general web applications. Sometimes some special user actions are required in certain applications. The core component provides required abstractions to implement them. The abstract class to extend for writing a custom command is AbstractCustomCommand which internally extends AbstractCommand class as shown in the diagram below. The other relevant classes related to customization are CommandValidationErrorHandler and SelmateContext.



**Command:** This interface represents the abstraction of all Selmate commands. The description of declared operations are mentioned below.

|  |  |
| --- | --- |
| **Operation Name** | **Functionality** |
| getName() | Returns the name of the command implementation. |
| getStepDescription() | Returns the description of the relevant step. |
| validate(CommandValidationErrorHandler errorHandler, SelmateContext selmateContext) | Validates the command input and add error messages to the CommandValidationErrorHandler class. |
| execute(WebDriver driver, SelmateContext selmateContext) | Execute the command with the WebDriver instance passed as an argument. |
| log(boolean status, String stepDescription, SelmateContext selmateContext) | Logs the commands of current command execution with proper step description. |

**AbstractCommand:** This class represents an abstraction of all Selmate commands and implements the Command interface. This class provides default implementation of some of the operations as described below.

|  |  |
| --- | --- |
| **Operation Name** | **Functionality** |
| getStepDescription() | Returns the description of the relevant step. |
| setStepDescription(String stepDescription) | Set description of the relevant step. |
| validate(CommandValidationErrorHandler errorHandler, SelmateContext selmateContext) | Provides a blank implementation of command validation. Implementation of validation logic is optional for all the command implementation classes. |
| validateNullorBlank(CommandValidationErrorHandler errorHandler, String name, String value, SelmateContext selmateContext) | Checks if the value passed is Null or Blank. |
| validatePositiveInteger(CommandValidationErrorHandler errorHandler, String name, long value, SelmateContext selmateContext) | Checks if the value passed is a positive integer. |
| log(boolean status, String stepDescription, SelmateContext selmateContext) | Logs the status of current command execution. This is a default implementation of logging. Individual command class can overwrite this if required. |

**AbstractCustomCommand:** This class is the basic abstraction for all custom commands in Selmate. It provides an encapsulation of all arguments passed as input. Any custom command class should extend this class and implement the getName() and execute() operation. Implementation of validate() method is optional for any of the custom command class.

|  |  |
| --- | --- |
| **Operation Name** | **Functionality** |
| addArgument(String argument) | Add an input argument to this command. |
| getArgument(int index) | Returns the argument for the mentioned index. |
| getArgumentCount() | Returns the argument count. |
| iterateArguments() | Iterates through all the arguments. |

**CommandValidationErrorHandler:** This class handles all the command validation error. It encapsulates all the validation error messages encountered in a Selmate script. The custom command implementation class should use this class for input argument validation. The description for its operations are mentioned below:

|  |  |
| --- | --- |
| **Operation Name** | **Functionality** |
| addErrorMessage(int step, String fieldName, String errorMsg) | This operation adds error message to the list of encapsulated error messages. |
| isErrorPresent() | Returns if any validation error occurred or not. |
| getMessages() | Returns the list of error messages. |

**SelmateContext:** This interface represents the execution context of Selmate. It exposes several operations which are essential for fetching current execution specific information. The relevant operations are described below.

|  |  |
| --- | --- |
| **Operation Name** | **Functionality** |
| getScriptName() | Returns the name of the script. |
| getImplicitWaitTime() | Returns the implicit wait time for an WebElement to be available. |
| getPollingInterval() | Returns the polling interval for looking up the WebElement. |
| getCurrentStep() | Returns the current step no. |
| getVariableValue(String name) | Returns the value of the variable. |
| isVariableNameReference(String name) | Returns if the specified name referrers any variable. |
| storeVariable(String name, Object value) | Stores the value w.r.t. a variable name in current execution context. If the variable pre-exists its value would be overwritten. |
| removeVariable(String name) | Removes a variable from context. |
| setImplicitWaitTime(long delayTime) | Sets the implicit wait time. |
| getMainWindow() | Returns the instance of main window. |
| getCurrentWindow() | Returns the current window in focus. |
| isExistingWindowHandle(String handle) | Returns if the passed window handle name is the currently focused window. |
| evaluateVariables(String content) | Replaces all the variable references with their actual value and returns the modified value. |

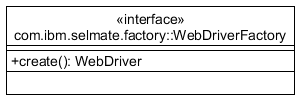
A sample custom command is developed as an example. The role of this command is to upload a xml file to an online xml file validator. The command class should be packaged in a jar file (e.g. selmate-custom.jar) which needs to be present in classpath. The relevant class and the script is attached herewith.

** **

The command to execute the above script including the mentioned command is given below:

java -D<DRIVER VM ARG NAME>="<<driver file path>>" -cp selmate-core.jar;selmate-xls-adapter.jar; selmate-custom.jar com.ibm.selmate.adapter.xls.XLSScriptExecutor --file CUSTOM\_CMD.xlsx

**Customization of WebDriver:** The default WebDriver provided by Selmate can be customized. Selmate provides supports for multiple WebDrivers e.g. FireFox, Internet Explorer and Chrome by default. It exposes an interface WebDriverFactory which represents the basic abstraction for any WebDriver factory implementation. Any concrete WebDriverFactory should implement this interface and the declared create() method. The full qualified class name for the custom WebDriverFactory implementation should be passed by a VM argument “selmate.webdriver.factory” as -Dselmate.webdriver.factory=<<full qualified class name of WebDriverFactory implementation>>. The interface WebDriverFactory is described below.



|  |  |
| --- | --- |
| **Operation Name** | **Functionality** |
| create() | This operation returns an instance of WebDriver. |

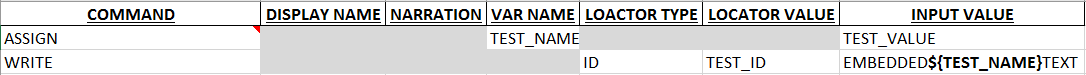
**Encryption of Password for Selmate:** Selmate provides features to encrypt a password. The same encrypted password should be used while using the WritePasswordCommand. The encryption key is internal to Selmate and should not be exposed outside. The command to encrypt a password is given below:

java -cp selmate-core.jar com.ibm.selmate.client.SelmatePasswordEncryptor <<password>>

**Use of Variables in Selmate:** Selmate provides features of using variables. Variables are declared and assigned value by AssignCommand. The same variables are referred using the declared variable name. For example, a variable can be defined like this using Selmate xls script



The input value “TEST\_VALUE” should be assigned to the variable named “TEST\_NAME”. The same variable can be referred by other command e.g. WRITE. A defined variable can be used as ${VARIABLENAME} or $VARIABLENAME. Referred variable can be embedded in another text values too using ${VARIABLENAME} as shown below.



The commands listed below supports use of variables.

**AssignCommand:** This command accepts variable reference as part of variable’s value.

**WriteCommand:** This command accepts variable reference as part of input text.

**SelectOptionsCommand:** This command accepts variable reference as option value.

**AssertCommand:** This command accepts variable reference as part of expected value, actual value and failure message.

The character ‘$’ is treated as a special character in Selmate. In order to use ‘$’ in input of any of the commands mentioned above it should be escaped by backslash (‘\’). If the WRITE command mentions input value as “EMBEDDED\${TEST\_NAME}TEXT” value of “TEST\_NAME” won’t be replaced by “TEST\_VALUE”. If backslash (‘\’) is to be used as a constant following a ‘$’ it would be escaped by another backslash (‘\’).

**Load Testing:** Load testing can be performed in headless mode by passing the directory path for input scripts. Each script is run in separate thread. The argument ‘–bulk’ is passed as an argument to initiate load testing. Internet explorer does not support headless execution.

**Software Artifacts:** Required artifacts to use Selmate are mentioned below.

**Jar files:**  Users need two jar files (placed in jars directory) to use Selmate, one for core component and another for xls adapter. selmate-core.jar file comprises all the dependent jars including jar files required for selenium, log4j logging and jaxb. Selmate-xls-adapter.jar file comprises the jar file Apache POI required for xls parsing.

**Dependent Libraries used:**

1. Selenium Webdriver 3.141.59
2. commons-validator 1.4.0
3. Apache POI 3.9
4. Apache Log4j 1.x
5. jdom2 2.0.6
6. ATUTestRecorder 2.1
7. Testng 6.5.2

**Selmate Setup:** Setup of Selmate requires JRE 1.8 to be installed in system. Users need to use selmate-core.jar and selmate-xls.jar file. Selmate is most likely to be used with the spreadsheet adapter. The supported browsers are Chrome, Internet Explorer and Firefox. Users need to develop windows/linux script for using Selmate. The command format to be used in script for different browsers are shown below:

**Chrome:** java -Dwebdriver.chrome.driver="<<driver file path>>\chromedriver.exe" –jar selmate-xls.jar --file <<input script path>>

**Firefox:** java -Dwebdriver.gecko.driver="<<driver file path>>\geckodriver.exe" –jar selmate-xls.jar --file <<input script path>>

**Internet Explorer:** java -Dwebdriver.ie.driver="<<driver file path>>\IEDriverServer.exe" –jar selmate-xls.jar --file <<input script path>>

**Advantages of using Selmate over Selenium:** Selmate provides multiple advantages over Selenium.

Advantages for Regression testing are listed below:

|  |  |
| --- | --- |
| **Selenium** | **Selmate** |
| Requires programming knowledge in either of the languages like C#, Groovy, Java, Perl, PHP, Python, Ruby or Scala. | Does not require any programming knowledge as regression script is written in a predefined spreadsheet template. |
| Need to write program for each of the execution flow. | No need to write any program. Entire execution flow can be defined as sequence of commands in the predefined template. |
| Requires good amount of learning time. | Learning time is much less. |
| People having no programming skills can only contribute in data preparation as defining execution flow requires programming skills. | People having no programming skills can contribute in development of execution flow as well as data preparation. |
| Developers need to program tasks like browser interaction, driver management etc. for any application. | Users need to use appropriate commands to perform specific type of interactions with the browser. The tool internally manages the driver. The same is also customizable by users. |
| Requires good amount of development time as well as cost for automating regression testing using Selenium. | Requires quite less development time as programming effort is minimized to a great extent. This also minimizes the cost of automation for any web application. |

Advantages for automation of Dynamic Process are listed below:

|  |  |
| --- | --- |
| **Selenium** | **Selmate** |
| Requires explicit knowledge of Selenium API for automating any dynamic process. | Selmate provides an interface in Java to interact with it. The input should be an xml document of a defined schema. Anything specific to Selenium is managed by Selmate internally. |
| Users need to write proper code for browser interaction using Selenium API and need to handle driver management and other system specific tasks. | Users need to generate a xml script which encapsulates sequence of commands to interact with the browser. No Selenium specific program needs to be written by user. |
| Requires good amount of learning time. | As the script is xml based it requires only knowledge in xml and java as programming language. |
| Requires good amount of coding effort in Selenium. | Coding effort is less. |

**Conclusion and Future Prospect:** This tool is an endeavor for minimizing programming efforts required for automation of web application testing. This would be mostly beneficial for automation of regression testing. As regression testing automation can be managed entirely by spreadsheet the need for programming is virtually eliminated. Selmate provides most of the common commands required for interacting with browser. It replaces the programming complexity of Selenium by its command based scripting facility using spreadsheet. Users are also provided proper interfaces for writing custom commands if required for any application. Minimization of programming requirement as well as customizability would make this tool immensely helpful for enterprise application testing automation.