Selenium IDE

Give Selenium a chance

Selenese Concepts

Command -> Target -> Value

Selenese Concepts: Commands

- ***Actions*** – manipulate the state of application
- ***Accessors*** – examine the state of application and store it in variables
- ***Assertions*** (verify, assert, waitFor) – verify that the state of application conforms to what is expected

Selenese Concepts: Actions

- **open, openWindow**
- **assignId**
- **captureEntirePageScreenshot**
- **check, removeSelection, removeAllSelections, select, uncheck**
- **click, keyDown, keyPress, keyUp, mouseMove, type, typeKeys**
- **createCookie, deleteCookie**
- **dragAndDrop**
- **focus**
- **goBack, refresh**
- **pause, setSpeed**
- **runScript**
- **store**
- **submit**
- **waitForCondition, waitForPageToLoad**
- …

Selenese Concepts: Accessors/Assertions

- Different prefixes:
  - **store, assert, assertNot, verify, verifyNot, waitFor, waitForNot**
- Possible suffixes:
  - **Selected, SelectedId, SelectedIndexes, SelectedIndex, SelectedLabel, SelectedLabels, SelectedValue, SelectedValues, SelectOptions, Checked, SomethingSelected**
  - **AllButtons, AllFields, AllLinks**
  - **Attribute, Value, Text, Editable, TextPresent**
  - **BodyText, HtmlSource, Location, Title**
  - **Cookie, CookieByName, CookiePresent**
  - **ElementIndex, XpathCount, Ordered**
  - **Eval, Expression**
  - **Table**
  - **ElementPresent, Visible**

Selenese Concepts: Variables

- store… can be used to store variable
  - storeValue->userName->name
- All variables are stored in JavaScript map *storedVars*
- Variable may be accessed by *${name}* or *storedVars['name']*
  - type->userName->${name}
  - type->userName->storedVars['name']
- Variable can be accessed in *javascript{}* block
  - type->userName->javascript{storedVars['name'].toUpperCase()}

Selenese Concepts: JavaScript

- Usage with script parameter
  - assertEval, storeEval, verifyEval, waitForEval
  - First only parameter (Target field)
- Other usages with special syntax
  - javascript{}

Selenese Concepts: Locators

- **identifier** – select element by @id attribute or first by name attribute, default mode
- **id=*id*** – select element by @id attribute
- **name=*name*** – select element by @name attribute
  - name=continue type=button – filtering by attribute
  - name=continue Clear – filtering by value
- **dom=*javascriptExpression*** – find element by evaluating JavaScript expression
  - dom=document.forms['myForm'].myDropdown
- **xpath=*xpathExpression*** – find element by XPath expression
  - xpath=//input[@name='name2' and @value='yes']

Selenese Concepts: Locators

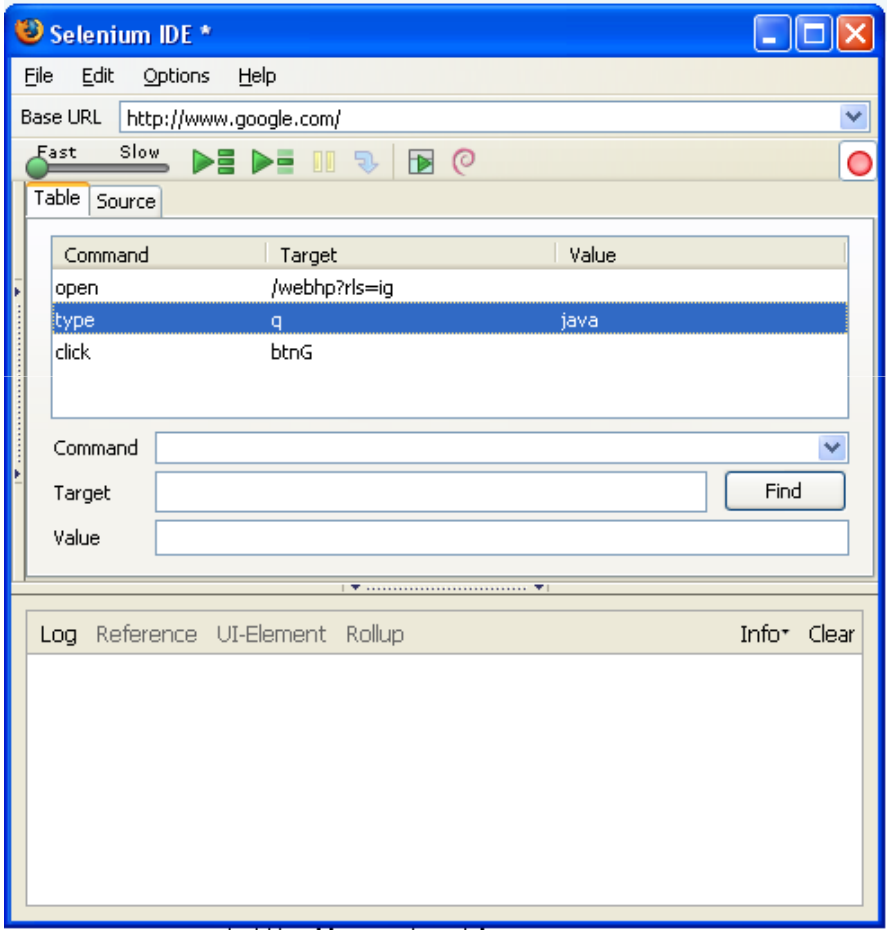
- **link=*textPattern*** – select link with text matching specified pattern
  - link=Home
- **css=*cssSelectorSyntax*** – select element using CSS selector (supports almost all css1, css2, css3 selectors)
  - css=a[href="#id3"]
- **ui=*uiSpecifierString*** – select element by resolving UI specifier to element locator and evaluating it
  - ui=loginPages::loginButton()

Selenese Concepts: Patterns

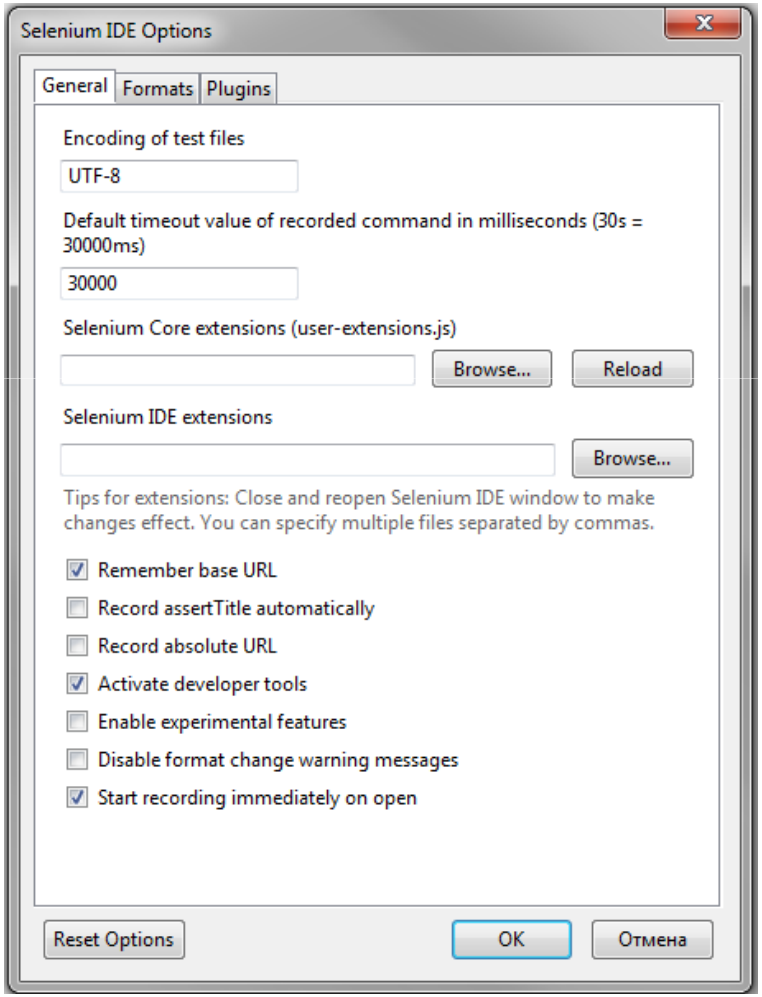
- **glob:*pattern*** – match a string against a "glob" (aka "wildmat") pattern, default
- **regexp:*regexp*** – match a string using a regular-expression
- **regexpi:*regexpi*** – match a string using a case-insensitive regular-expression
- **exact:*string*** – match a string exactly

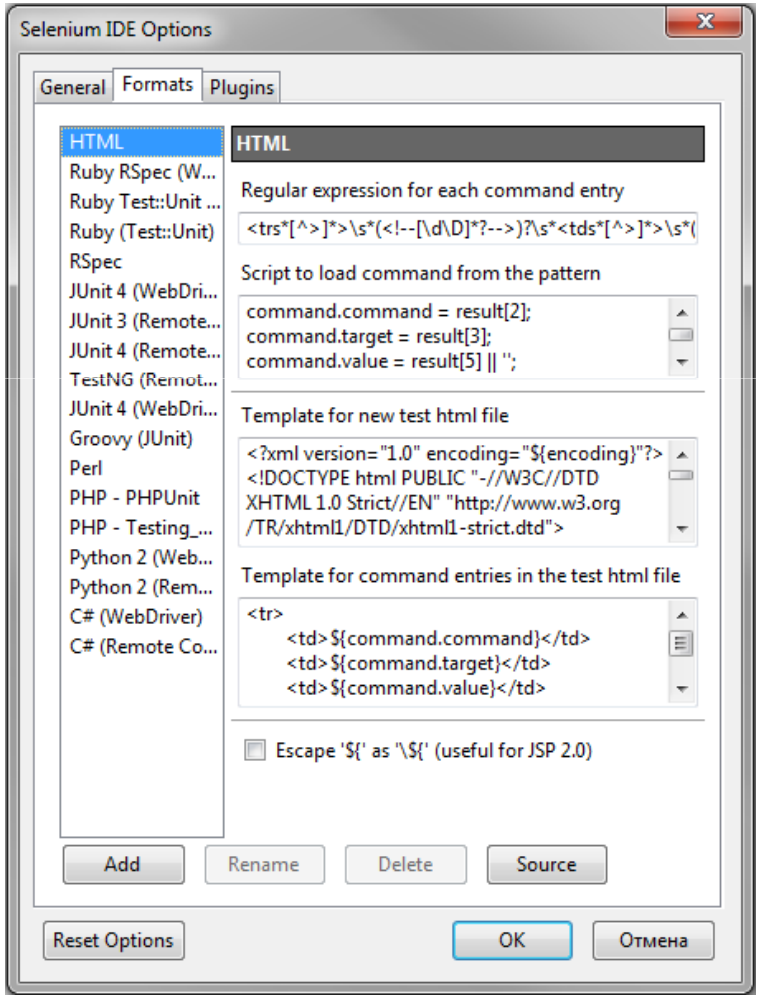
Selenium IDE

- Plug-in for Firefox
- Record and play tests
- Test editor and debugger
- Support JavaScript extensions
- Transform test to different languages
- Auto complete Selenium commands
- Intelligent location selection
- Find element on page or source code
- Inline documentation
- Manage tests and suites



Selenium IDE: Settings





Selenium IDE: Formats

- Support a lot of formats (HTML, Java, C#, Perl, PHP, Python, Ruby, Wiki, FitNesse, CSV)
- Steps to add custom format
  - Open formats settings dialog
  - Add new format and set its name
  - Implement JavaScript methods
    - parse – parse string to create test case
    - format – convert test case to string
    - formatCommands – convert some commands to string
  - Add configuration options

Selenium IDE: User Extensions

- assertEquals
- assertNumericCompare
- assertTextPresentXML
- assertTextPresentCount
- callWebService
- collect extended info about failures
- flowControl
- include
- datadriven
- qooxdooExtension
- reloadAndWaitFor
- removeCookie
- selectFrame
- storeNamedCookie
- storeTableContent
- timer
- waitForCondition
- debug
- eval
- flexTesting
- getTableRows
- getVal
- global
- httpresource
- inputIsEditable
- isOneOfPresent
- locateByLabelText
- locateCookie
- locateElementByPartialId
- selectWindow
- storeassertXpath
- storeGetVar
- updateFeedback
- waitAndActions

http://wiki.openqa.org/display/SEL/Contributed+User-Extensions

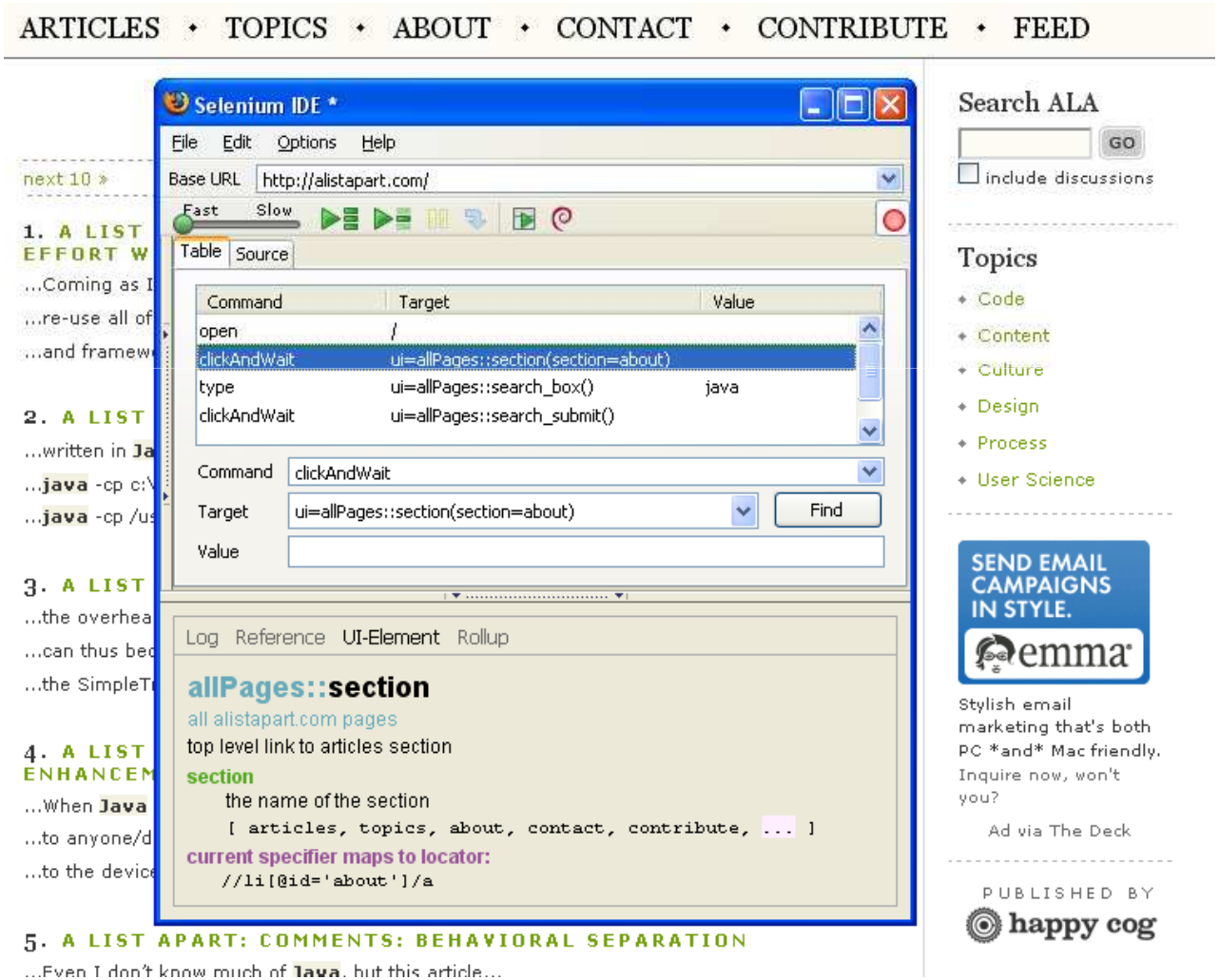
Selenium IDE: UI-Element

- ***Page*** – a unique URL, and the contents available by accessing that URL
- ***Page element*** – an element on the actual webpage
- ***Pageset*** – a set of pages that share some set of common page elements
- ***UI element*** – a mapping between a meaningful name for a page element, and the means to locate that page element's DOM node
- ***UI argument*** – an optional piece of logic that determines how the locator is generated by a UI element
- ***UI map*** – a collection of pagesets

Selenium IDE: UI-Element

- ***UI specifier string*** – a bit of text containing a pageset name, a UI element name, and optionally arguments that modify the way a locator is constructed by the UI element
- ***Rollup rule*** – logic that describes how one or more Selenium commands can be grouped into a single command, and how that single command may be expanded into its component Selenium commands
- ***Command matcher*** – matches one or more Selenium commands and optionally sets values for rollup arguments based on the matched commands
- ***Rollup argument*** – an optional piece of logic that modifies the command expansion of a rollup

Selenium IDE: UI-Element



Additional Plug-ins

- Test Suite Batch Converter
- Highlight Elements
- Stored Variables Viewer
- File Logging
- Log Search Bar
- Suppress Security Warning
- Selenium IDE Buttons
- Screenshot on Fail
- Test Results
- Selenium Expert
- Flow Control
- Power Debugger
- Selenium Formatter
- Flex Pilot X / FlexMonkium
- Favorites
- Remember Certificate Exception
- Page Coverage
- Implicit Wait

https://addons.mozilla.org/en-US/firefox/search?q=Selenium

Flow Control

- label
- goto, gotoAndWait, gotoIf, gotoIfAndWait
- gotolabel, gotolabelAndWait
- while, whileAndWait
- endWhile, endWhileAndWait

Detailed samples and links here: http://automated-testing.info/knowledgebase/article/selenium-ide-flowcontrol-primery-ispolzovaniya

Selenium IDE: Demo

- Test recording
- Element location on page and source code
- Code generation in different formats
- Work with custom formats
- Use extensions
- Use UI-Element
- Manage tests and suites
- Plug-ins usage

Selenium IDE: Evaluation

- Pros
  - Quick starting point
  - No same domain policy
  - Very helpful documentation and locator suggestions
- Cons
  - Generated code is garbage
  - Very dependent on page structure
  - Locators are not optimized
  - Can’t reuse test parts

Selenium IDE: Best Practices

- Create special URLs for setUp/tearDown
- Add documentation to tests and suites
- Use common HTML structure for tests
- Show test table as example
- Create test suites hierarchy
- First ‘assert’ and then ‘verify’ strategy

Selenium IDE: Other Usages

- Create regression tests for refactoring of existing application
- Record template scenarios from other sites
- Communication tool for team members
- Retrieve data from pages
- Make quick demo presentations
- Describe bugs
- Investigate site structure and testability
- Record test cases for other testing frameworks
- Operations in social networks ☺

Selenium IDE: Future is Se-Builder

- Started in 2011
- HTML + JavaScript application for Selenium tests management
- Looks ugly
- At the moment available as Firefox plug-in
- Will work in future for all browsers

Questions & Answers