Selenium Integration

Choose your way of using Selenium

WebDriver/Selenium RC Tests Issues

- Verbose
  - “browser.”, “driver.” or “selenium.” is everywhere
  - Locators everywhere
- Not expressive
  - You can’t see how UI looks like
  - You can’t tell what is the UI element by locator
- Coupling
  - UI locators, actions and assertions are coupled together
- Fragile
  - Hard to understand XPath
  - Related to page structure

DSL Approach

- Question: After some time you have following issues?
  - Hard to understand the intent of automated tests
  - Duplication of Selenium test code
  - Hard to reuse
  - Test the wrong thing
  - Because of complexity, fewer scenarios are created
  - Overtesting cause performance problem
  - New team members find it hard to write tests
  - QAs take a long time to write automated tests
- Answer: Yes ☹
- Solution: You need DSL!

DSL Approach: Impact

- DSL makes your tests
  - ***High-level*** – they are in the language of product management, at the user level
  - ***Readable*** – product management can understand them
  - ***Writable*** – easy for developer or QA to write new test
  - ***Extensible*** – tests base is able to grow with DSL

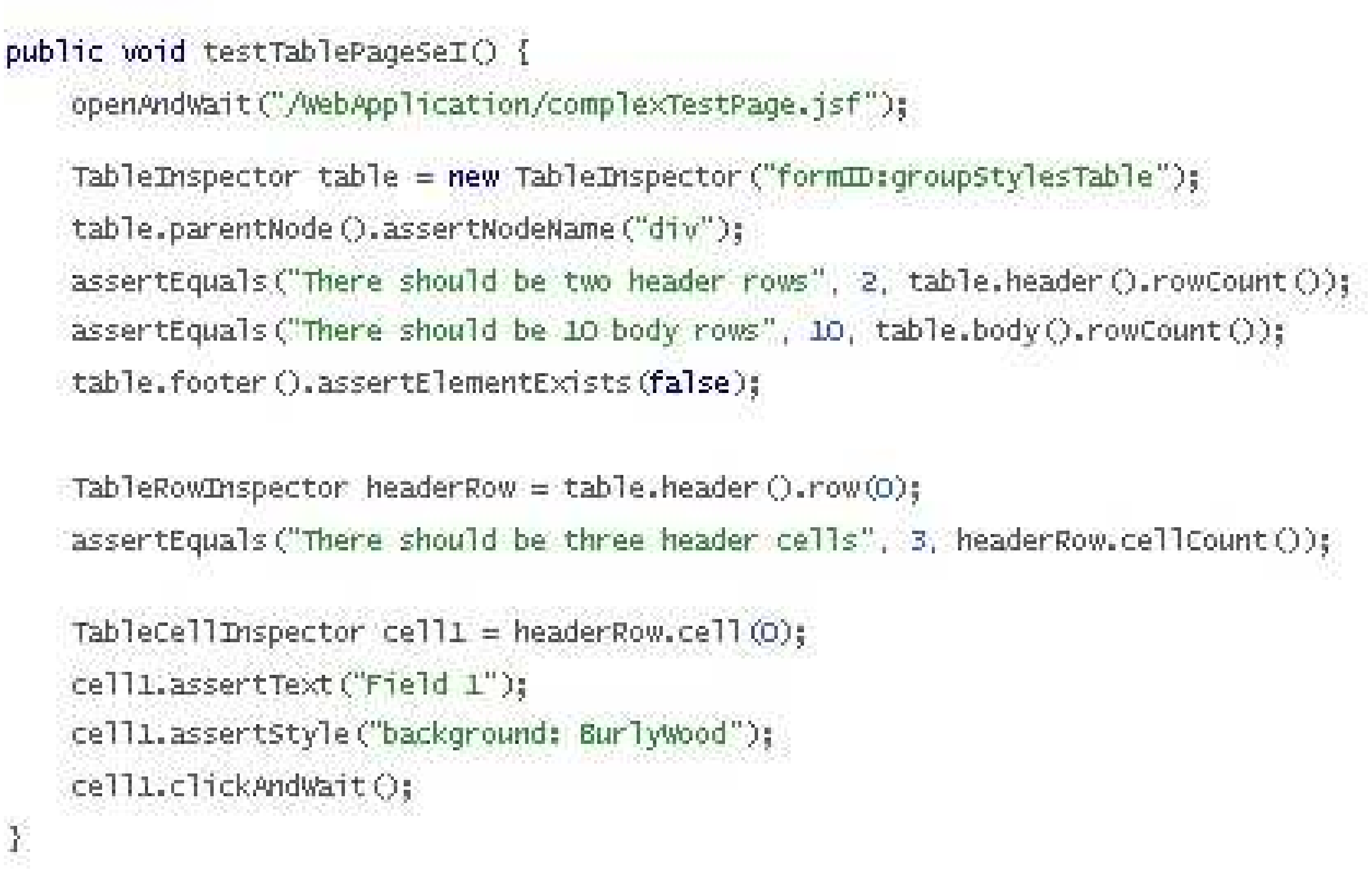
DSL Approach: Steps

- For each page object type create class with general methods
    - Checkbox: click, shouldBeSelected, shouldNotBeSelected, …
    - TextBox: type, shouldBeReadOnly, …
    - Table, SelectBox, Label, …
    - Complex objects like Panel, ListOfItems
- Tests become clear and self-documented
    - googlePage.searchTextbox().type(“Selenium”);
    - googlePage.searchButton().click();
- Extend assertions to make them clearer
    - hasValue, containsText, …
- Create basic hierarchy to allow chain invocations
    - has, assertThat, with, …
- Create helpers with reusable functionality
    - Database helper, navigation helper, …

Selenium Inspector

- Higher level API based on Selenium
- Simplify testing for some technologies like JSF
- Introduce parent-child relationships
- Allow testing components not only primitive elements
- Provide a lot of assertion methods

Selenium Inspector: Usage

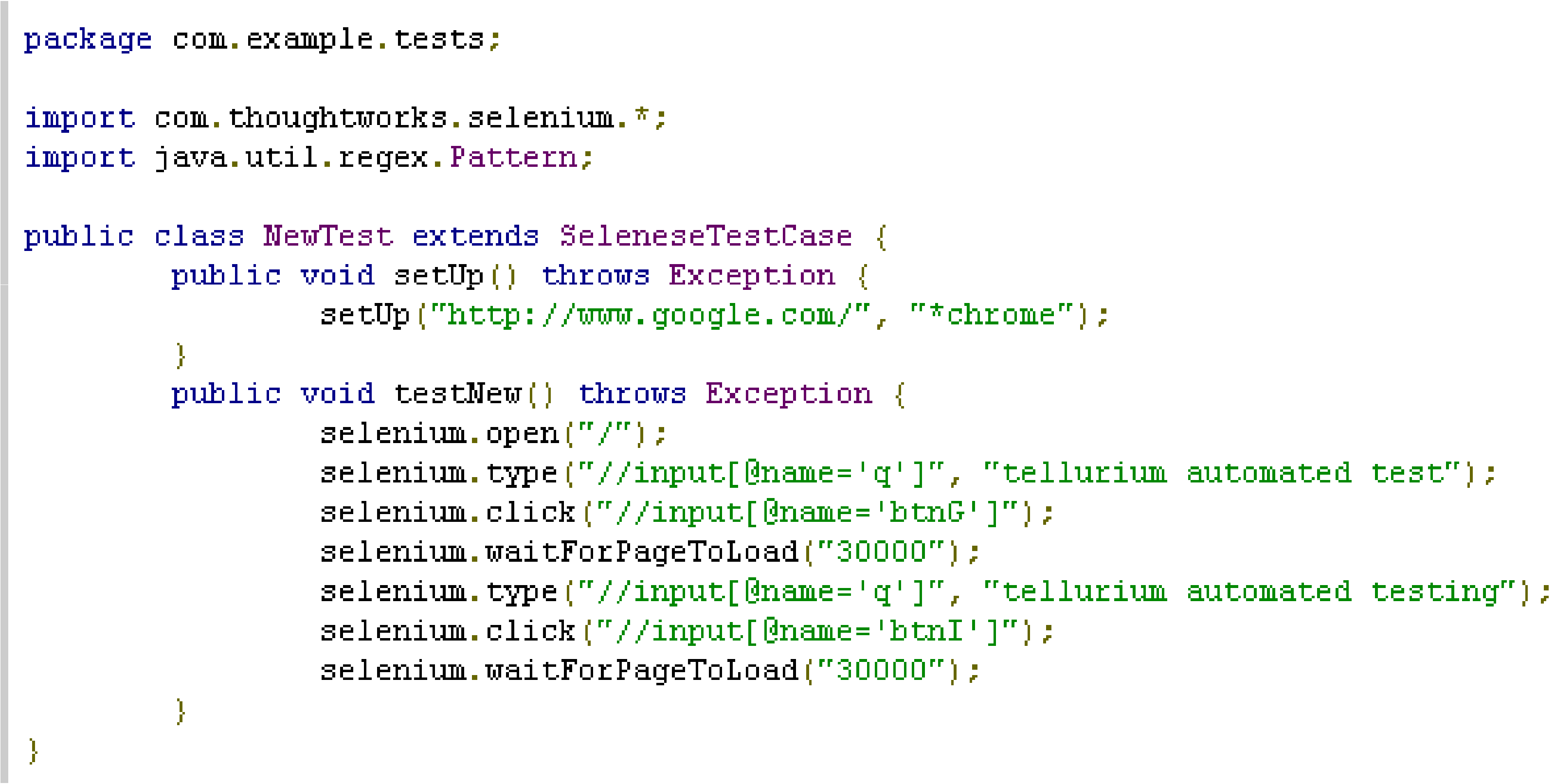


```
public void testTablePageSeI() {
    openAndWait("/WebApplication/complexTestPage.jsf");

    TableInspector table = new TableInspector("formID:groupStylesTable");
    table.parentNode().assertNodeName("div");
    assertEquals("There should be two header rows", 2, table.header().rowCount());
    assertEquals("There should be 10 body rows", 10, table.body().rowCount());
    table.footer().assertElementExists(false);

    TableRowInspector headerRow = table.header().row(0);
    assertEquals("There should be three header cells", 3, headerRow.cellCount());

    TableCellInspector cell1 = headerRow.cell(0);
    cell1.assertText("Field 1");
    cell1.assertStyle("background: BurlyWood");
    cell1.clickAndWait();
}
```

Tellurium

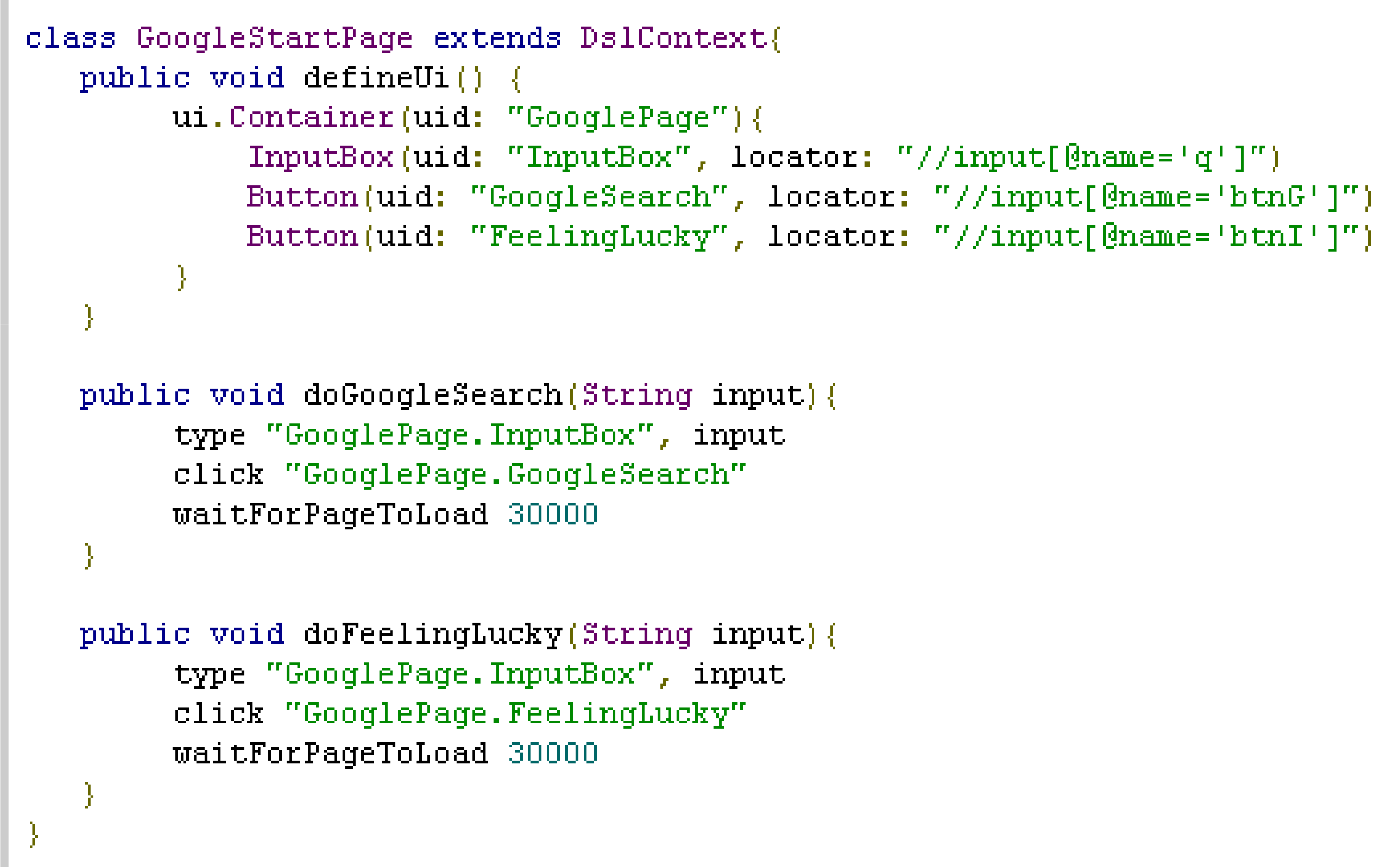
- Built on top of Selenium at the current stage
- Implemented in Groovy and Java
- Object to Locator Mapping – define object by attributes at runtime
- Composite Locators
- Group Locating Concept – group elements to simplify their finding
- Domain Specific Language (DSL)
- UI templates – represent many identical UI elements or dynamic size elements at runtime
- Data Driven Testing
- Tellurium UI Model Plugin (TrUMP) – Firefox plug-in to automatically create UI modules from pages

Tellurium: Initial Selenium Test



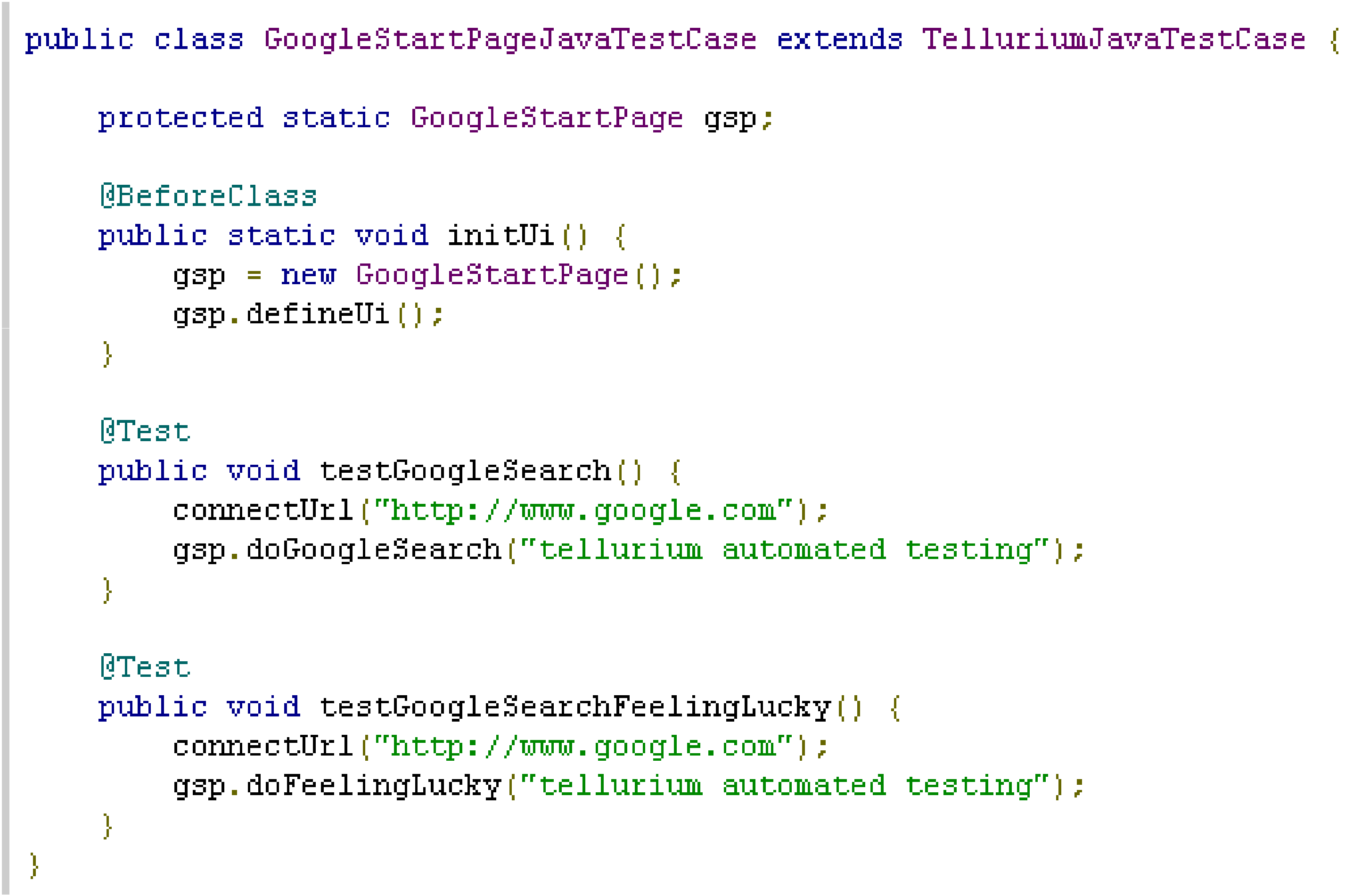
```java
package com.example.tests;

import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class NewTest extends SeleneseTestCase {
        public void setUp() throws Exception {
                setUp("http://www.google.com/", "*chrome");
        }
        public void testNew() throws Exception {
                selenium.open("/");
                selenium.type("//input[@name='q']", "tellurium automated test");
                selenium.click("//input[@name='btnG']");
                selenium.waitForPageToLoad("30000");
                selenium.type("//input[@name='q']", "tellurium automated testing");
                selenium.click("//input[@name='btnI']");
                selenium.waitForPageToLoad("30000");
        }
}
```

Tellurium: UI Module



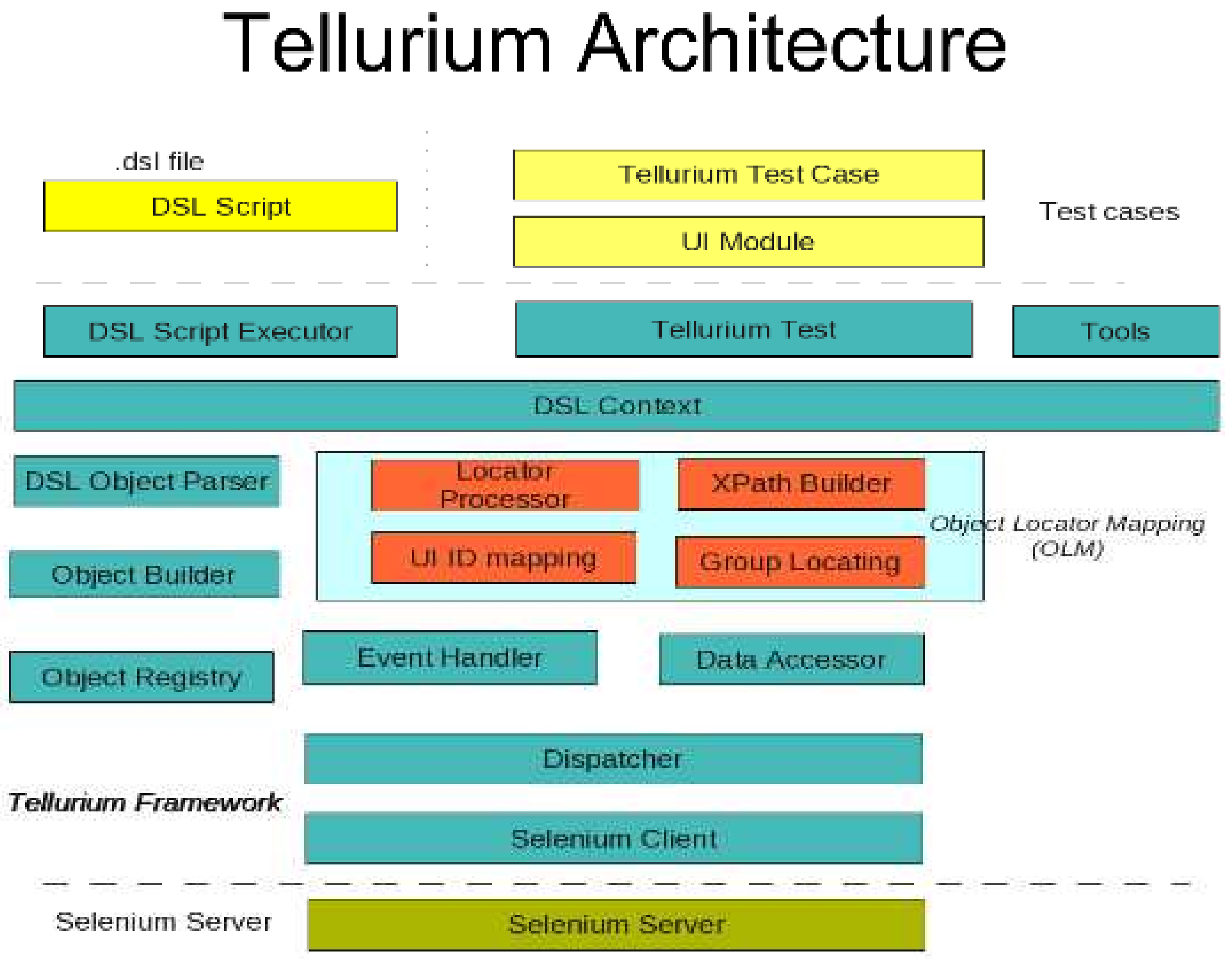
```
class GoogleStartPage extends DslContext{
   public void defineUi() {
        ui.Container(uid: "GooglePage"){
            InputBox(uid: "InputBox", locator: "//input[@name='q']")
            Button(uid: "GoogleSearch", locator: "//input[@name='btnG']")
            Button(uid: "FeelingLucky", locator: "//input[@name='btnI']")
        }
   }

   public void doGoogleSearch(String input){
        type "GooglePage.InputBox", input
        click "GooglePage.GoogleSearch"
        waitForPageToLoad 30000
   }

   public void doFeelingLucky(String input){
        type "GooglePage.InputBox", input
        click "GooglePage.FeelingLucky"
        waitForPageToLoad 30000
   }
}
```

Tellurium: Converted Test

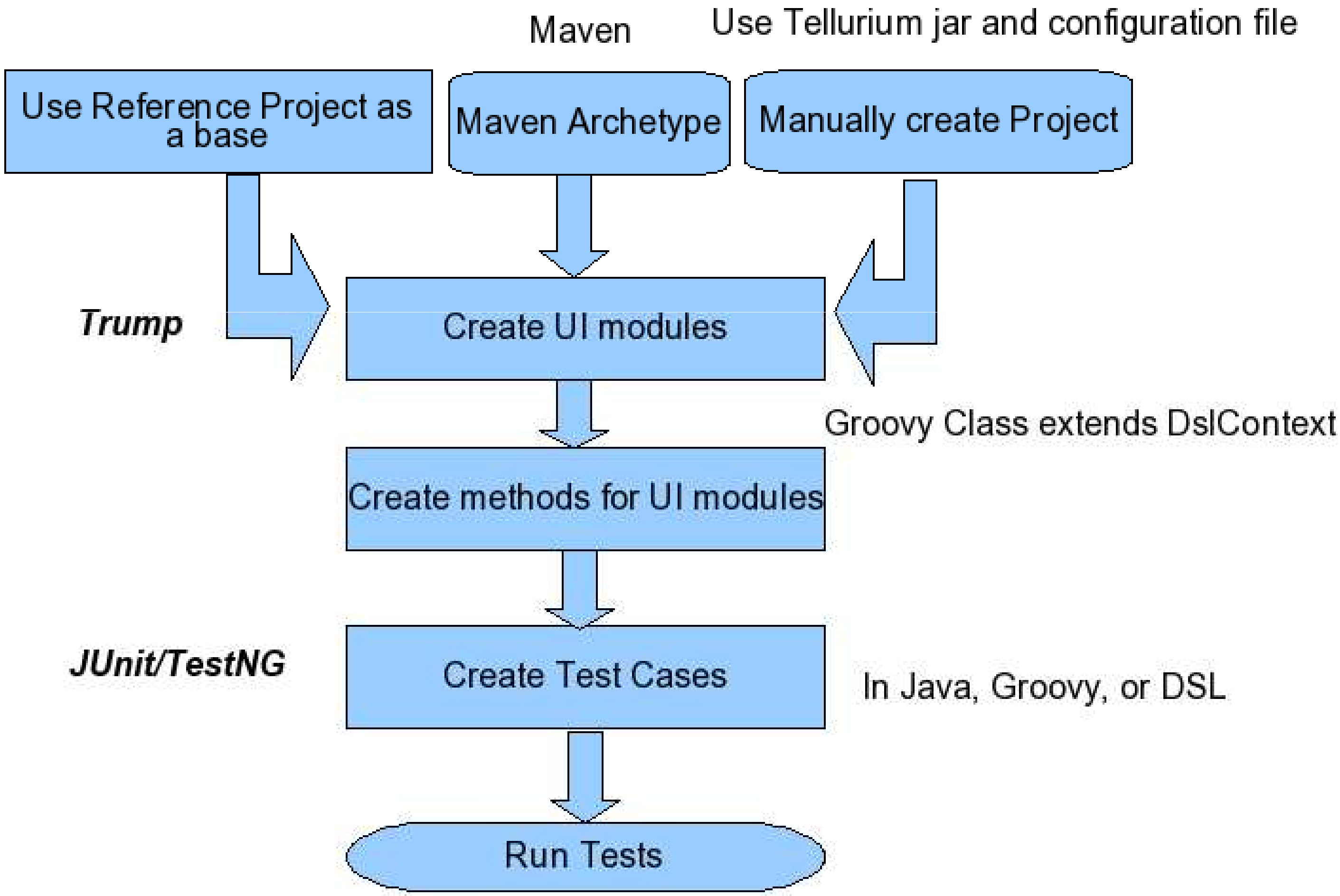


```
public class GoogleStartPageJavaTestCase extends TelluriumJavaTestCase {

    protected static GoogleStartPage gsp;

    @BeforeClass
    public static void initUi() {
        gsp = new GoogleStartPage();
        gsp.defineUi();
    }

    @Test
    public void testGoogleSearch() {
        connectUrl("http://www.google.com");
        gsp.doGoogleSearch("tellurium automated testing");
    }

    @Test
    public void testGoogleSearchFeelingLucky() {
        connectUrl("http://www.google.com");
        gsp.doFeelingLucky("tellurium automated testing");
    }
}
```

Tellurium: Architecture



Tellurium: Test Creation



Maven

Use Tellurium jar and configuration file

Use Reference Project as a base

Maven Archetype

Manually create Project

Trump

Create UI modules

Groovy Class extends DslContext

Create methods for UI modules

JUnit/TestNG

Create Test Cases

In Java, Groovy, or DSL

Run Tests

Tellurium: UI Module

- UI module – collection of UI elements grouped together
  - Separate Groovy file
  - Contains all elements configuration
  - Contains all actions available with the elements
  - Build XPath in runtime
  - Express page structure
  - If one element changed all other will automatically affected
  - Make composite objects reusable
  - Access elements by UID

Tellurium: UI Object

- UI object includes
  - ***uid*** – unique in UI module
  - ***namespace*** – for future extension
  - ***locator*** – locator of the UI object
    - A base locator (relative XPath)
    - A composite locator (object attributes)
  - ***group*** – applied to some kinds of UI objects to switch on grouping locating
  - ***respond*** – define JavaScript events the UI object could respond to
  - ***some basic methods*** – isElementPresent, isVisible, isDisabled, waitForElementPresent, getText, mouseOver, mouseOut, getAttribute

Tellurium: UI Object Types

- Button, SubmitButton, Icon
  - Additional methods: click, clickAt, doubleClick
- CheckBox, RadioButton
  - Additional methods: check, isChecked, uncheck
- Image
  - Additional methods: getImageSource, getImageAlt, getImageTitle
- TextBox
  - Additional methods: waitForText
- InputBox
  - Additional methods: type, keyType, typeAndReturn, clearText, isEditable, getValue
- UrlLink
  - Additional methods: getLink, click, doubleClick, clickAt
- Selector
  - Additional methods: selectByLabel, selectByValue, getSelectedOptions, getSelectedLabel, getSelectedValue, getSelectedIndex, getSelectedId, isSomethingSelected

Tellurium: UI Object Types

- Container
- Form
  - Additional methods: submit
- List
- Table
- Frame
  - Additional methods: selectFrame, getWhetherThisFrameMatchFrameExpression, waitForFrameToLoad
- Window
- Option
- Widget

Tellurium: TrUMP

- Automatically generates UI modules for pages
- Just click on the elements to add them
- Doesn’t generate tests

Tellurium: Tellurium IDE

- Records test scenarios in browser
- Builds UI map for used elements
- Allows to change UI map with specialized forms
- Different types of exporting

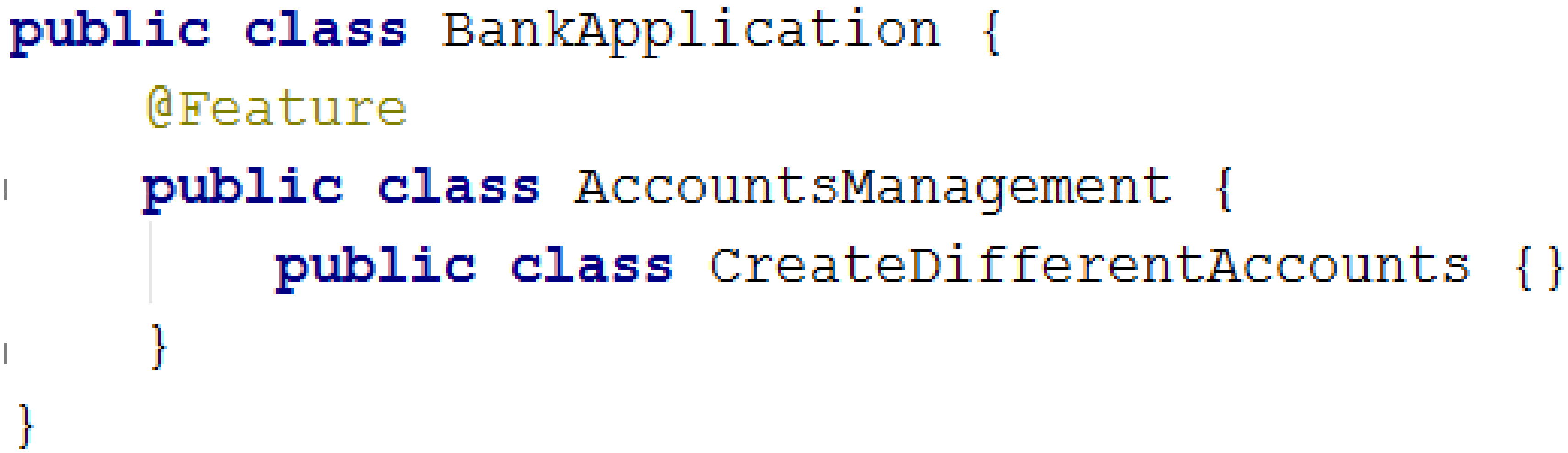
Tellurium: Benefits

- UI and tests are separated
- UI is in separate Groovy file
- Composite locator generates XPath in runtime
- Group locating makes element independent from external elements
- Easy to refactor
- Reusable
- Expressive

Thucydides

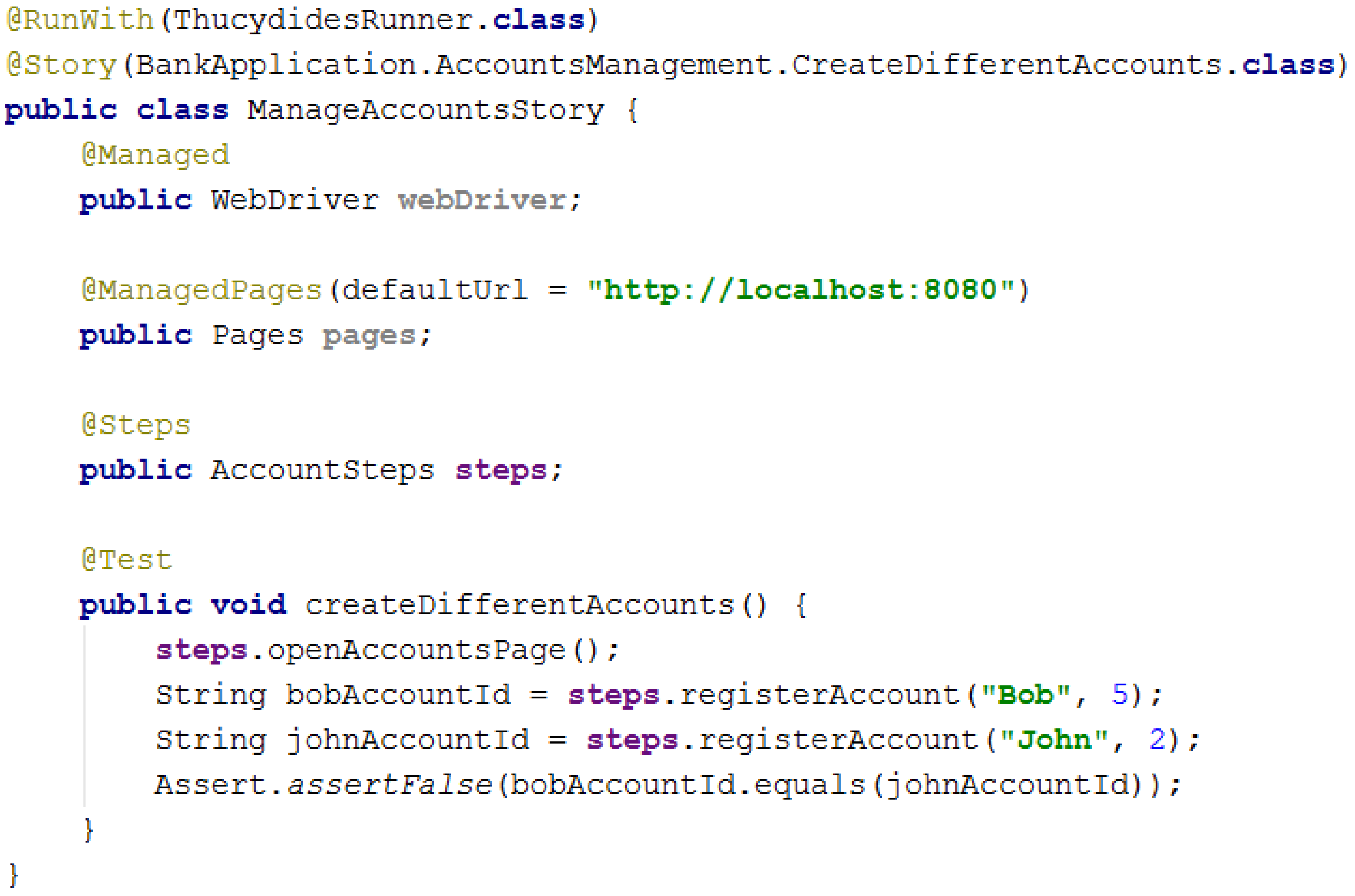
- Acceptance testing tool, based on WebDriver
- Clear separation of concepts
- Easy to understand and manage tests
- Lots of additions to WebDriver
- Completely in Java with Maven integration
- Amazing reporting with different views and screenshots

Thucydides: Declare Features



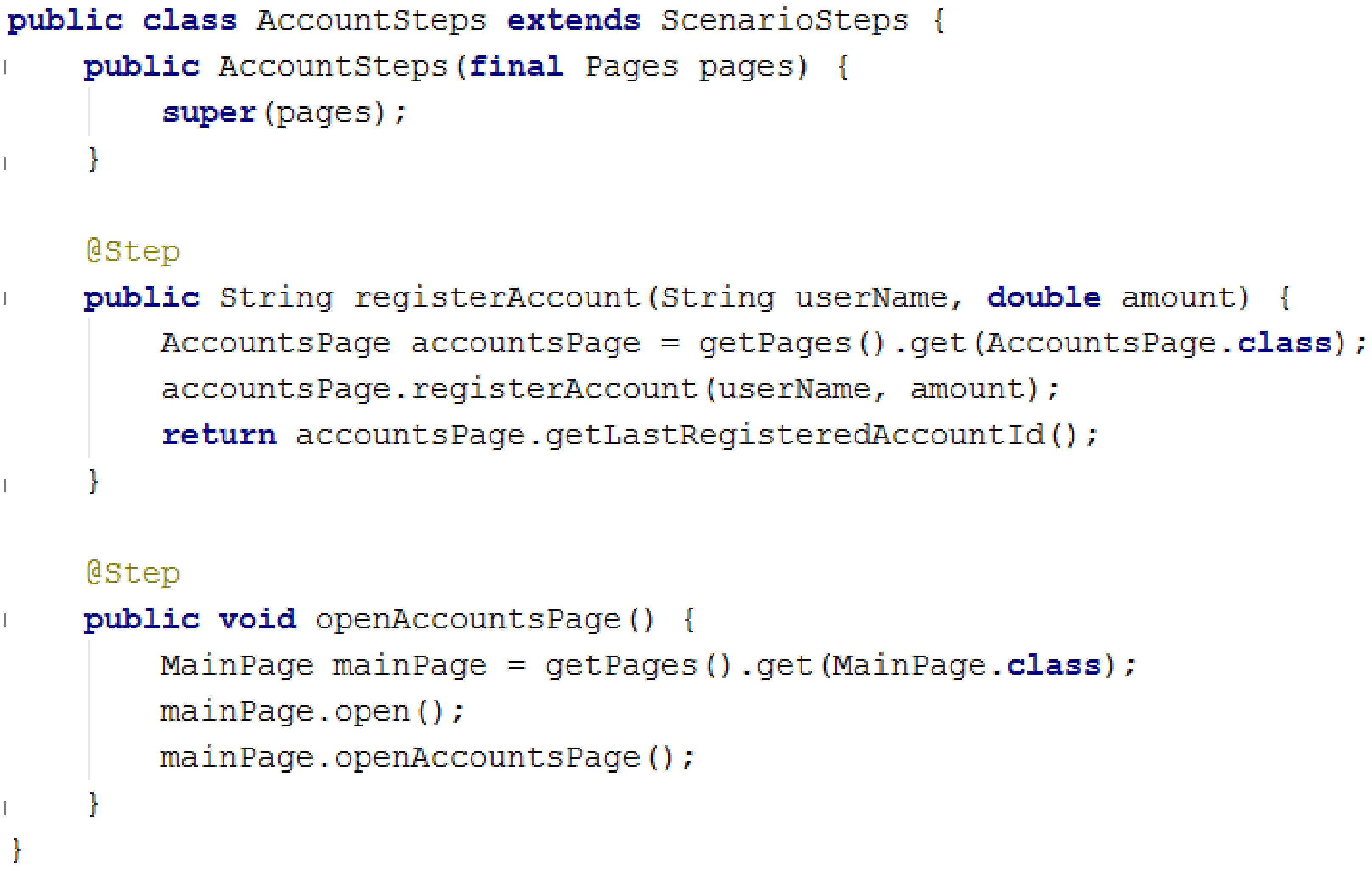
```
public class BankApplication {
    @Feature
    public class AccountsManagement {
        public class CreateDifferentAccounts {}
    }
}
```

Thucydides: Create Test



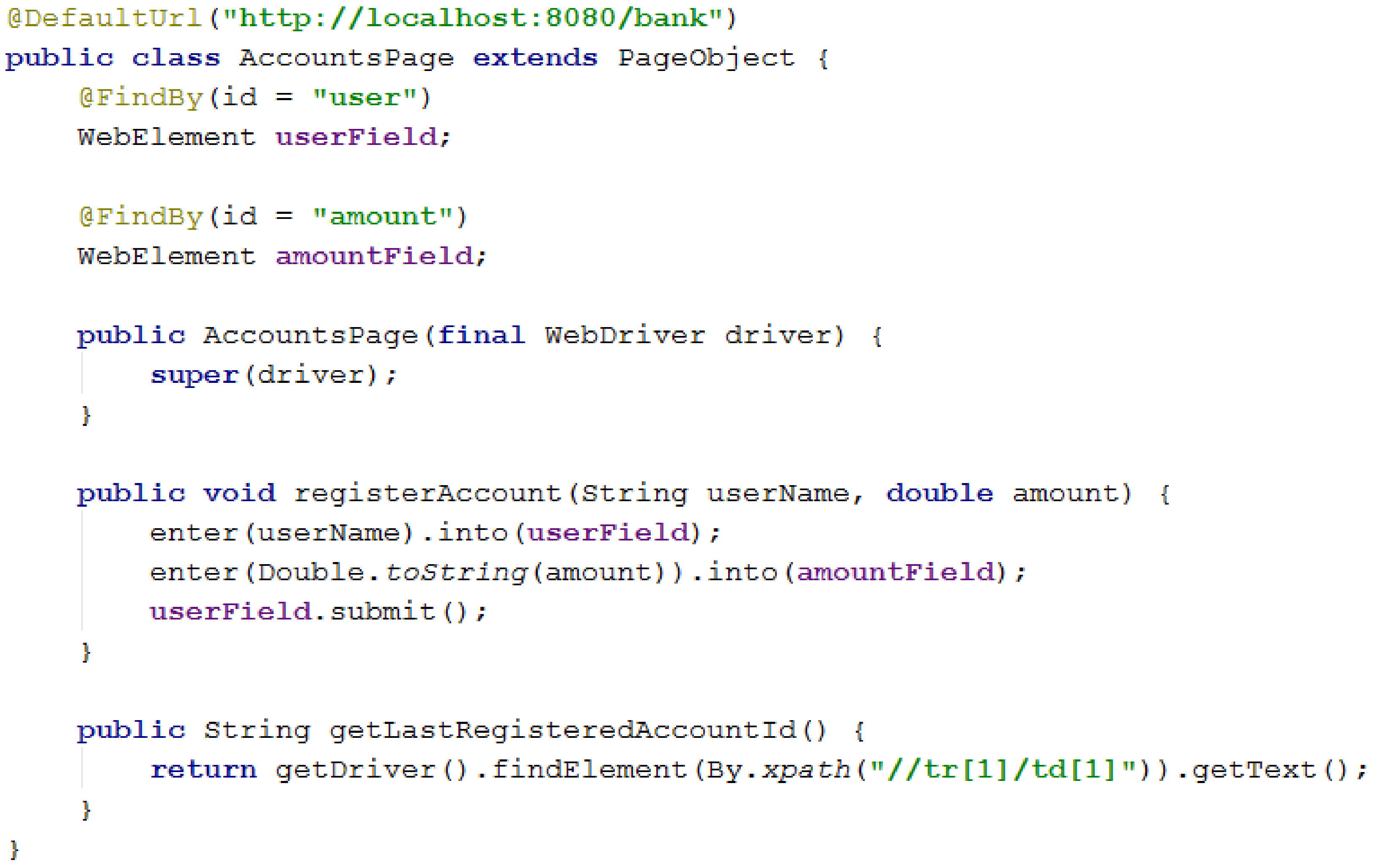
```
@RunWith(ThucydidesRunner.class)
@Story(BankApplication.AccountsManagement.CreateDifferentAccounts.class)
public class ManageAccountsStory {
    @Managed
    public WebDriver webDriver;

    @ManagedPages(defaultUrl = "http://localhost:8080")
    public Pages pages;

    @Steps
    public AccountSteps steps;

    @Test
    public void createDifferentAccounts() {
        steps.openAccountsPage();
        String bobAccountId = steps.registerAccount("Bob", 5);
        String johnAccountId = steps.registerAccount("John", 2);
        Assert.assertFalse(bobAccountId.equals(johnAccountId));
    }
}
```

Thucydides: Implement Steps



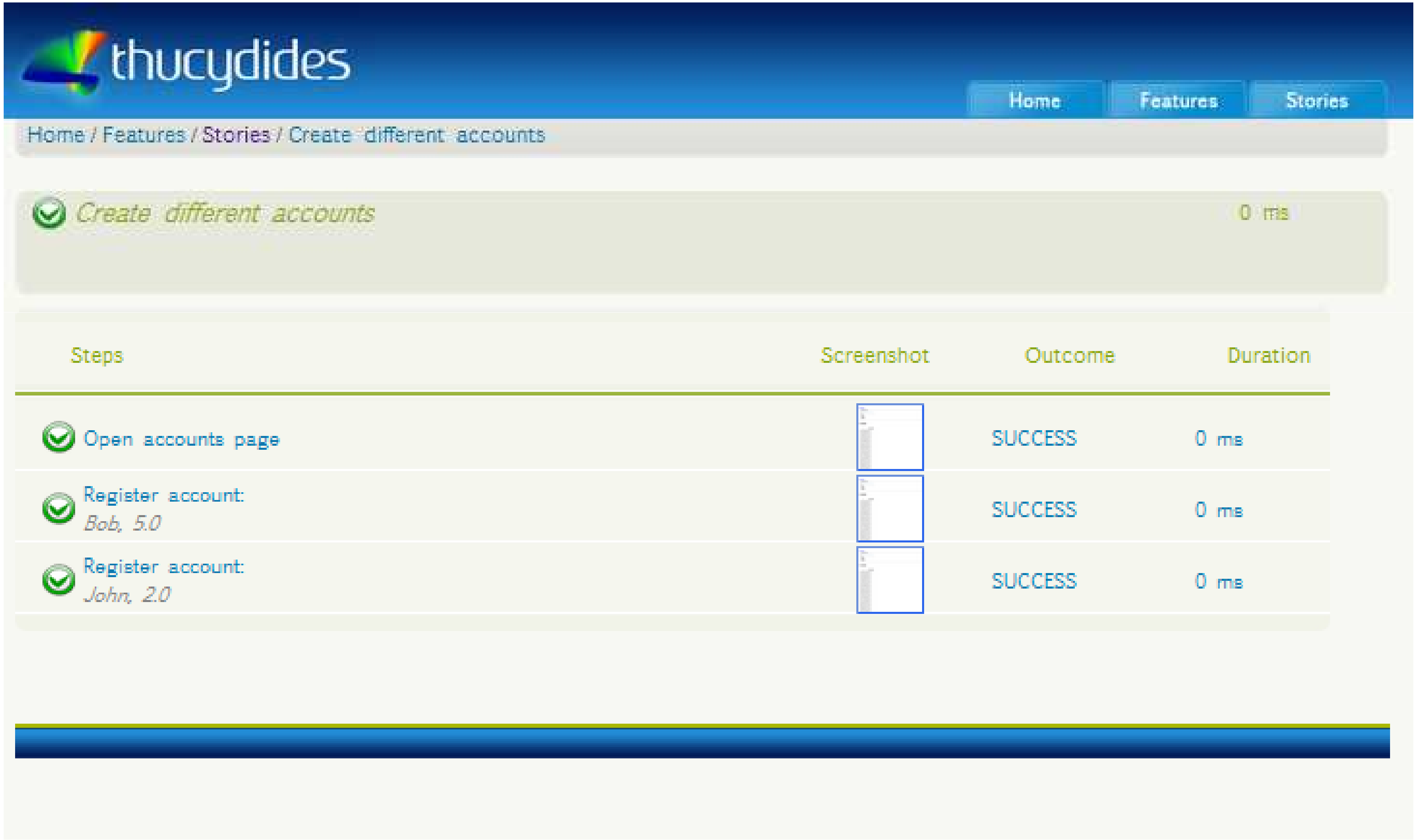
```java
public class AccountSteps extends ScenarioSteps {
    public AccountSteps(final Pages pages) {
        super(pages);
    }

    @Step
    public String registerAccount(String userName, double amount) {
        AccountsPage accountsPage = getPages().get(AccountsPage.class);
        accountsPage.registerAccount(userName, amount);
        return accountsPage.getLastRegisteredAccountId();
    }

    @Step
    public void openAccountsPage() {
        MainPage mainPage = getPages().get(MainPage.class);
        mainPage.open();
        mainPage.openAccountsPage();
    }
}
```

Thucydides: Create Pages



```java
@DefaultUrl("http://localhost:8080/bank")
public class AccountsPage extends PageObject {
    @FindBy(id = "user")
    WebElement userField;

    @FindBy(id = "amount")
    WebElement amountField;

    public AccountsPage(final WebDriver driver) {
        super(driver);
    }

    public void registerAccount(String userName, double amount) {
        enter(userName).into(userField);
        enter(Double.toString(amount)).into(amountField);
        userField.submit();
    }

    public String getLastRegisteredAccountId() {
        return getDriver().findElement(By.xpath("//tr[1]/td[1]")).getText();
    }
}
```

Thucydides: Check Reports

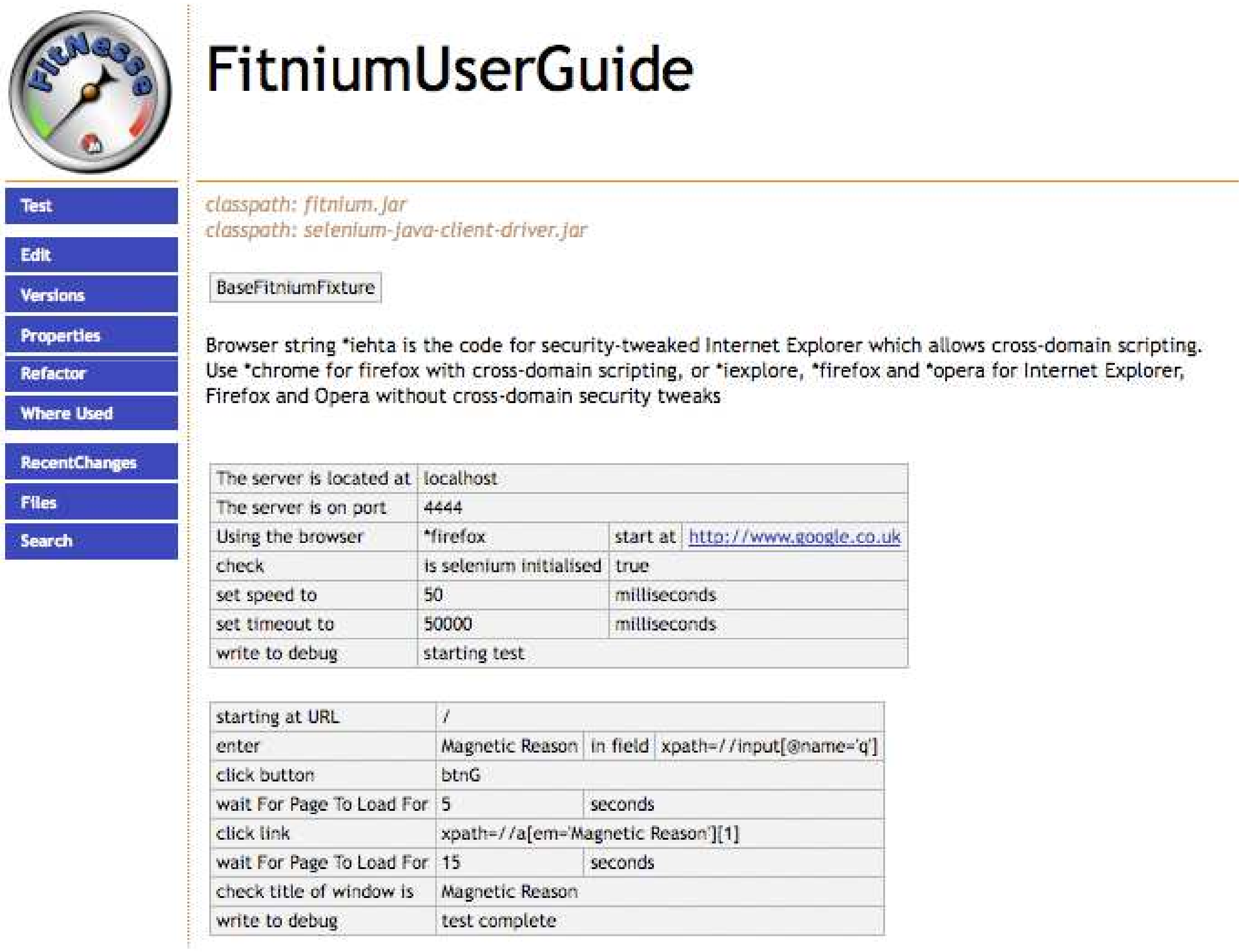


Home / Features / Stories / Create different accounts

✓ *Create different accounts* 0 ms

Steps	Screenshot	Outcome	Duration
✓ Open accounts page		SUCCESS	0 ms
✓ Register account: *Bob, 5.0*		SUCCESS	0 ms
✓ Register account: *John, 2.0*		SUCCESS	0 ms

FitNesse Integration: Fitnium



FitniumUserGuide

classpath: fitnium.jar
classpath: selenium-java-client-driver.jar

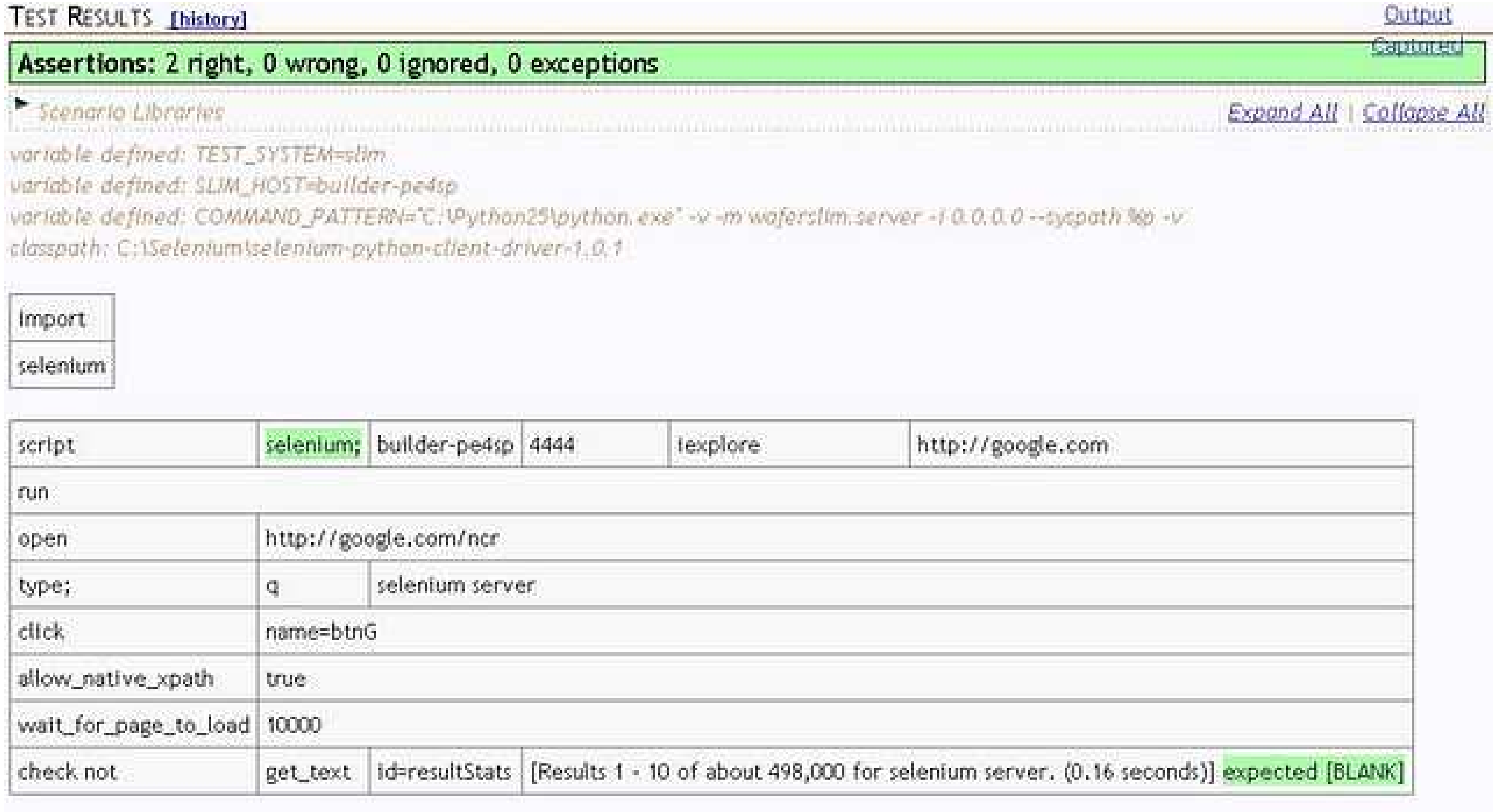
BaseFitniumFixture

Browser string *iehta is the code for security-tweaked Internet Explorer which allows cross-domain scripting. Use *chrome for firefox with cross-domain scripting, or *iexplore, *firefox and *opera for Internet Explorer, Firefox and Opera without cross-domain security tweaks

The server is located at	localhost		
The server is on port	4444		
Using the browser	*firefox	start at	http://www.google.co.uk
check	is selenium initialised	true	
set speed to	50	milliseconds	
set timeout to	50000	milliseconds	
write to debug	starting test		

starting at URL	/		
enter	Magnetic Reason	in field	xpath=//input[@name='q']
click button	btnG		
wait For Page To Load For	5	seconds	
click link	xpath=//a[em='Magnetic Reason'][1]		
wait For Page To Load For	15	seconds	
check title of window is	Magnetic Reason		
write to debug	test complete		

FitNesse Integration: Selenesse



TEST RESULTS [history]

Output Captured

Assertions: 2 right, 0 wrong, 0 ignored, 0 exceptions

► *Scenario Libraries*

Expand All | Collapse All

variable defined: TEST_SYSTEM=slim

variable defined: SLIM_HOST=builder-pe4sp

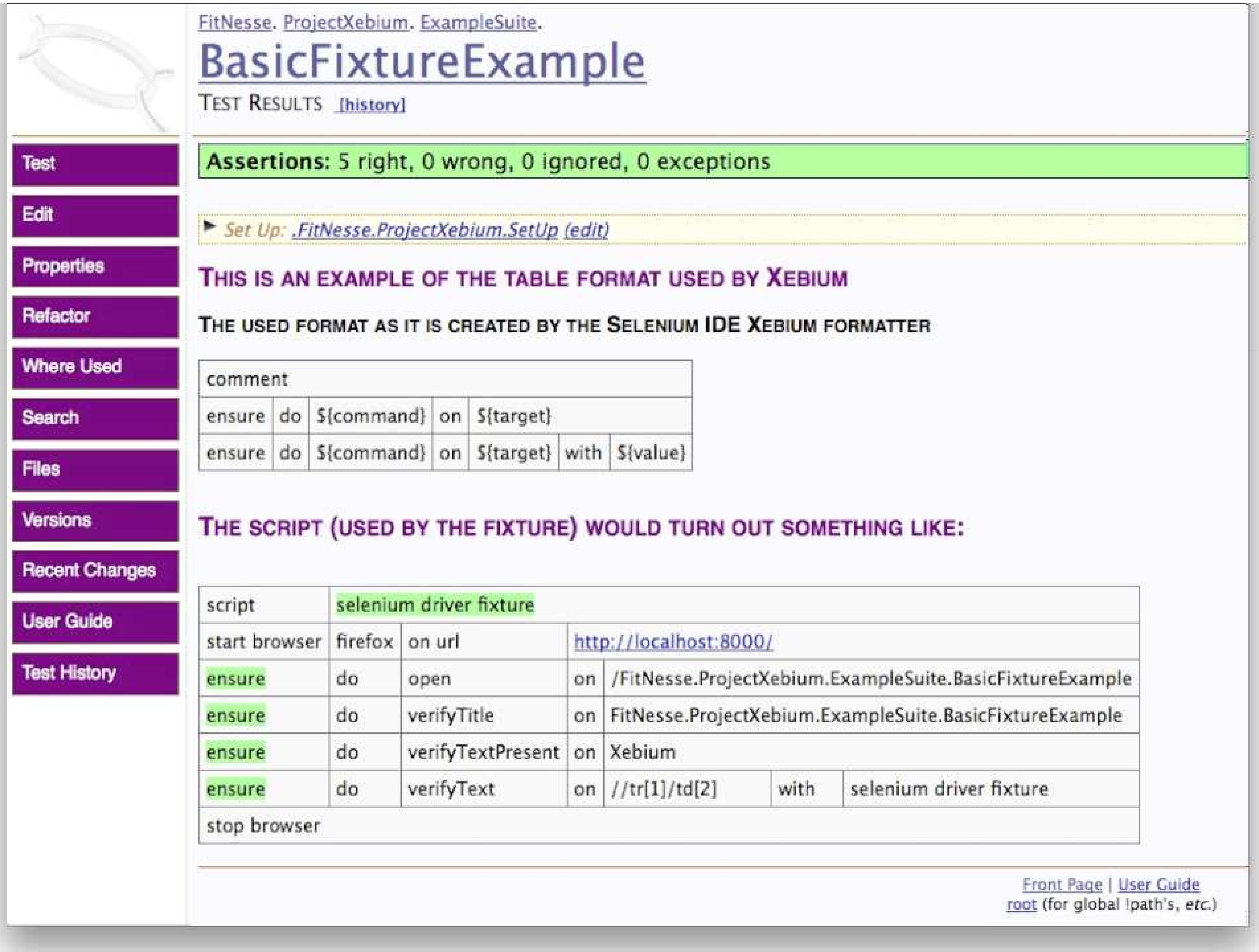
variable defined: COMMAND_PATTERN="C:\Python25\python.exe" -v -m waferslim.server -i 0.0.0.0 --syspath %p -v

classpath: C:\Selenium\selenium-python-client-driver-1.0.1

import
selenium

script	selenium;	builder-pe4sp	4444	iexplore	http://google.com
run					
open	http://google.com/ncr				
type;	q	selenium server			
click	name=btnG				
allow_native_xpath	true				
wait_for_page_to_load	10000				
check not	get_text	id=resultStats	[Results 1 - 10 of about 498,000 for selenium server. (0.16 seconds)] expected [BLANK]		

FitNesse Integration: Xebium



FitNesse. ProjectXebium. ExampleSuite.

BasicFixtureExample

TEST RESULTS [history]

Assertions: 5 right, 0 wrong, 0 ignored, 0 exceptions

▶ *Set Up: .FitNesse.ProjectXebium.SetUp (edit)*

THIS IS AN EXAMPLE OF THE TABLE FORMAT USED BY XEBIUM

THE USED FORMAT AS IT IS CREATED BY THE SELENIUM IDE XEBIUM FORMATTER

comment						
ensure	do	${command}	on	${target}		
ensure	do	${command}	on	${target}	with	${value}

THE SCRIPT (USED BY THE FIXTURE) WOULD TURN OUT SOMETHING LIKE:

script	selenium driver fixture					
start browser	firefox	on url	http://localhost:8000/			
ensure	do	open	on	/FitNesse.ProjectXebium.ExampleSuite.BasicFixtureExample		
ensure	do	verifyTitle	on	FitNesse.ProjectXebium.ExampleSuite.BasicFixtureExample		
ensure	do	verifyTextPresent	on	Xebium		
ensure	do	verifyText	on	//tr[1]/td[2]	with	selenium driver fixture
stop browser						

Front Page | User Guide
root (for global !path's, *etc.*)

FitNesse Integration: Benefits

- Integration tools provide fixtures for integration with WebDriver/Selenium RC
  - Connection to the server using driver
  - Many Selenium commands
- FitNesse provide test management Wiki server
  - Selenium tests may be mixed with acceptance tests
  - Acceptance tests may use Selenium commands
- WebDriver/Selenium RC executes all tests
  - Remote server is started separately

Molybdenum: Test Execution

- Recording, executing and report generation without a line of code
- Runs tests with a defined delay or step by step or failed tests only
- Allows breakpoints to be set and stored along with the test

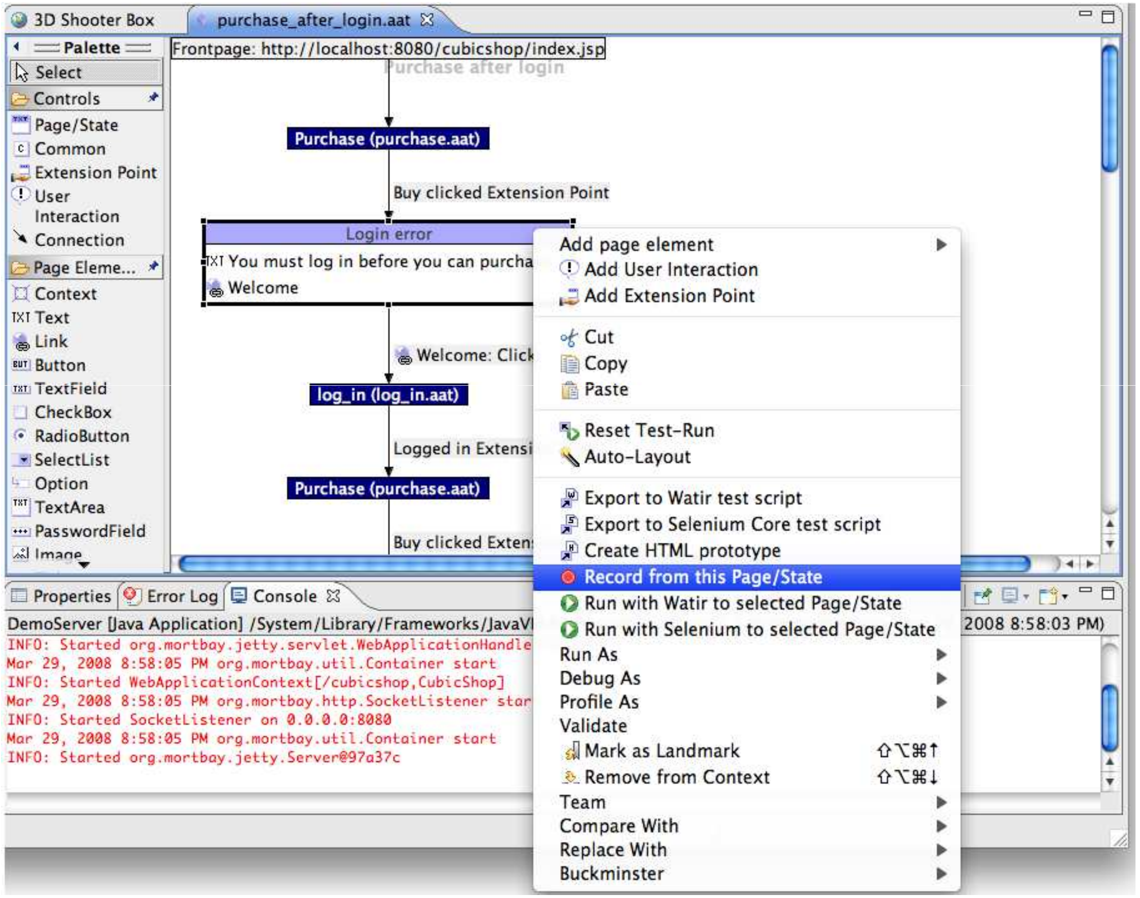
Molybdenum: Reporting

- HTML reports
- One click test reruns
- Attached HTML source to broken commands
- Screenshots in the report in case of errors

Molybdenum: Test Creation

- Capture and replay mode
- Ajax recording
- Imports Selenese tests

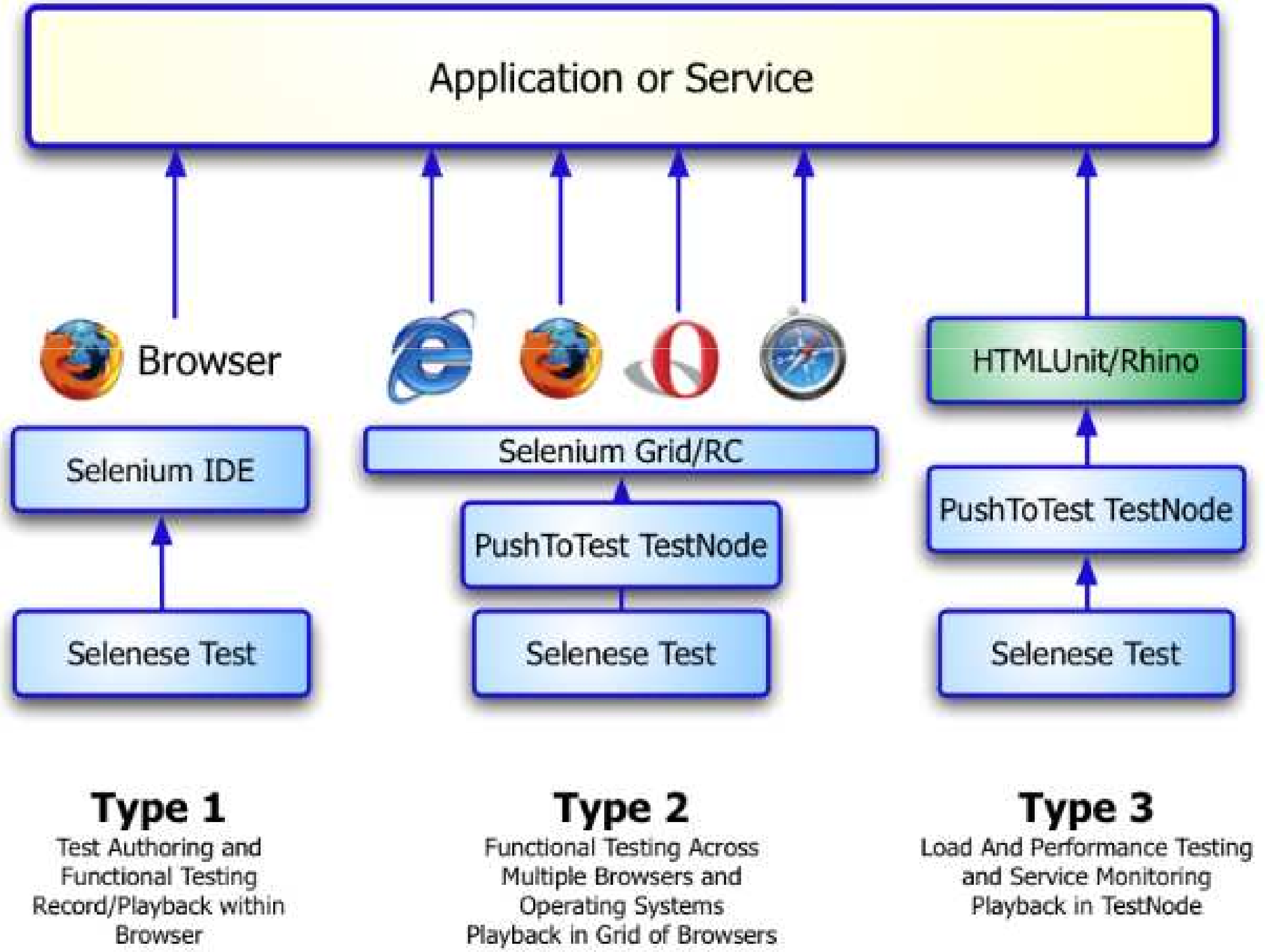
Selenium: CubicTest



```
DemoServer [Java Application] /System/Library/Frameworks/JavaVl
INFO: Started org.mortbay.jetty.servlet.WebApplicationHandle
Mar 29, 2008 8:58:05 PM org.mortbay.util.Container start
INFO: Started WebApplicationContext[/cubicshop,CubicShop]
Mar 29, 2008 8:58:05 PM org.mortbay.http.SocketListener star
INFO: Started SocketListener on 0.0.0.0:8080
Mar 29, 2008 8:58:05 PM org.mortbay.util.Container start
INFO: Started org.mortbay.jetty.Server@97a37c
```

- Graphical Eclipse plug-in
- Graphical test editor for Selenium and Watir
- Centered around pages/states and transitions
- Test recorder and runner

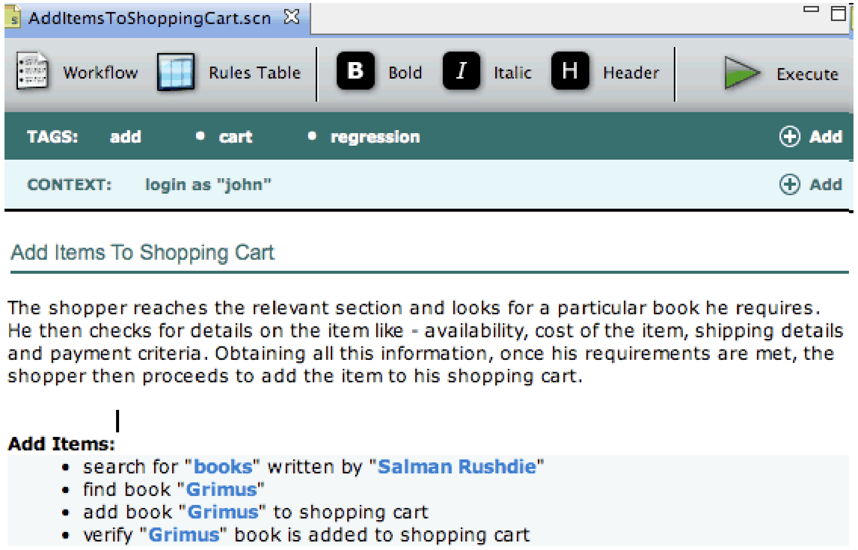
PushToTest Integration



Acceptance Testing: Twist

- Acceptance testing tool integrated with WebDriver/Selenium
- Created by ThoughtWorks
- Tests are in plain English language
- Organize test suites in various ways
- Powerful editor features
- Use DSL for test creation
- Run and analyze test suites

Acceptance Testing: Twist



TAGS: add • cart • regression

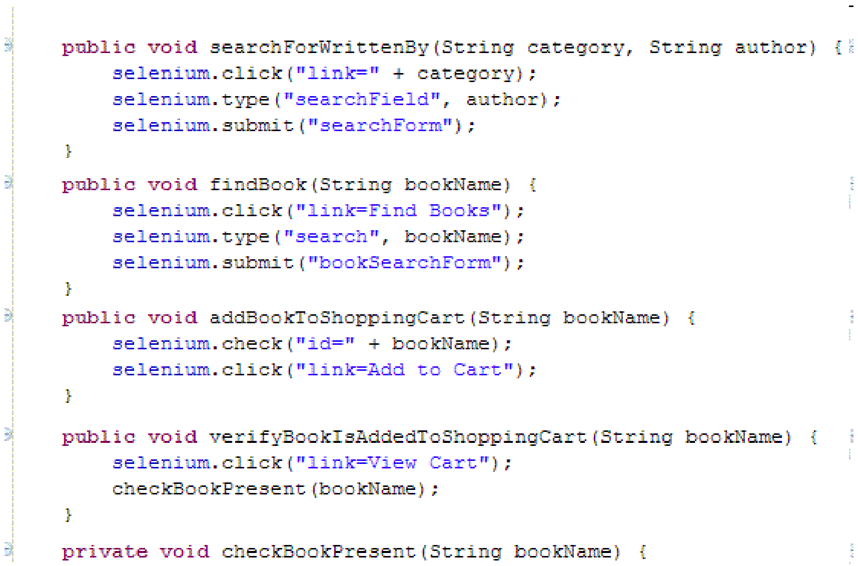
CONTEXT: login as "john"

Add Items To Shopping Cart

The shopper reaches the relevant section and looks for a particular book he requires. He then checks for details on the item like - availability, cost of the item, shipping details and payment criteria. Obtaining all this information, once his requirements are met, the shopper then proceeds to add the item to his shopping cart.

Add Items:

- search for "**books**" written by "**Salman Rushdie**"
- find book "**Grimus**"
- add book "**Grimus**" to shopping cart
- verify "**Grimus**" book is added to shopping cart

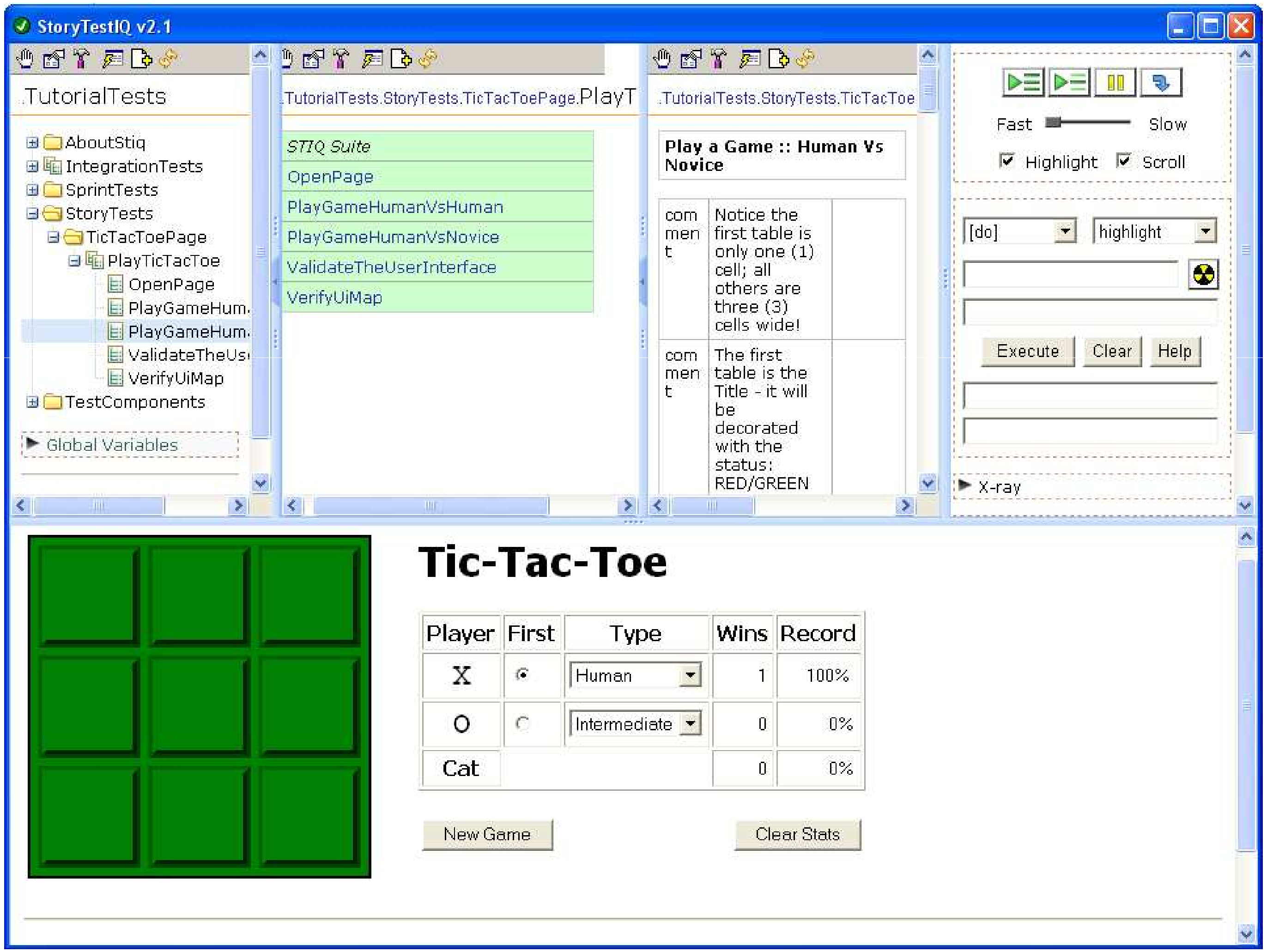


```
public void searchForWrittenBy(String category, String author) {
    selenium.click("link=" + category);
    selenium.type("searchField", author);
    selenium.submit("searchForm");
}
public void findBook(String bookName) {
    selenium.click("link=Find Books");
    selenium.type("search", bookName);
    selenium.submit("bookSearchForm");
}
public void addBookToShoppingCart(String bookName) {
    selenium.check("id=" + bookName);
    selenium.click("link=Add to Cart");
}

public void verifyBookIsAddedToShoppingCart(String bookName) {
    selenium.click("link=View Cart");
    checkBookPresent(bookName);
}

private void checkBookPresent(String bookName) {
```

StoryTestIQ: FitNesse + Selenium

- “Wiki-ized” Selenium
- Contains control widgets
- Large set of predefined actions and assertions
- Database related actions
- All features of FitNesse
- Helpful documentation
- Quick start

StoryTestIQ: User Interface



.TutorialTests

- AboutStiq
- IntegrationTests
- SprintTests
- StoryTests
  - TicTacToePage
    - PlayTicTacToe
      - OpenPage
      - PlayGameHum
      - PlayGameHum
      - ValidateTheUs
      - VerifyUiMap
- TestComponents

.TutorialTests.StoryTests.TicTacToePage.PlayT

STIQ Suite
OpenPage
PlayGameHumanVsHuman
PlayGameHumanVsNovice
ValidateTheUserInterface
VerifyUiMap

.TutorialTests.StoryTests.TicTacToe

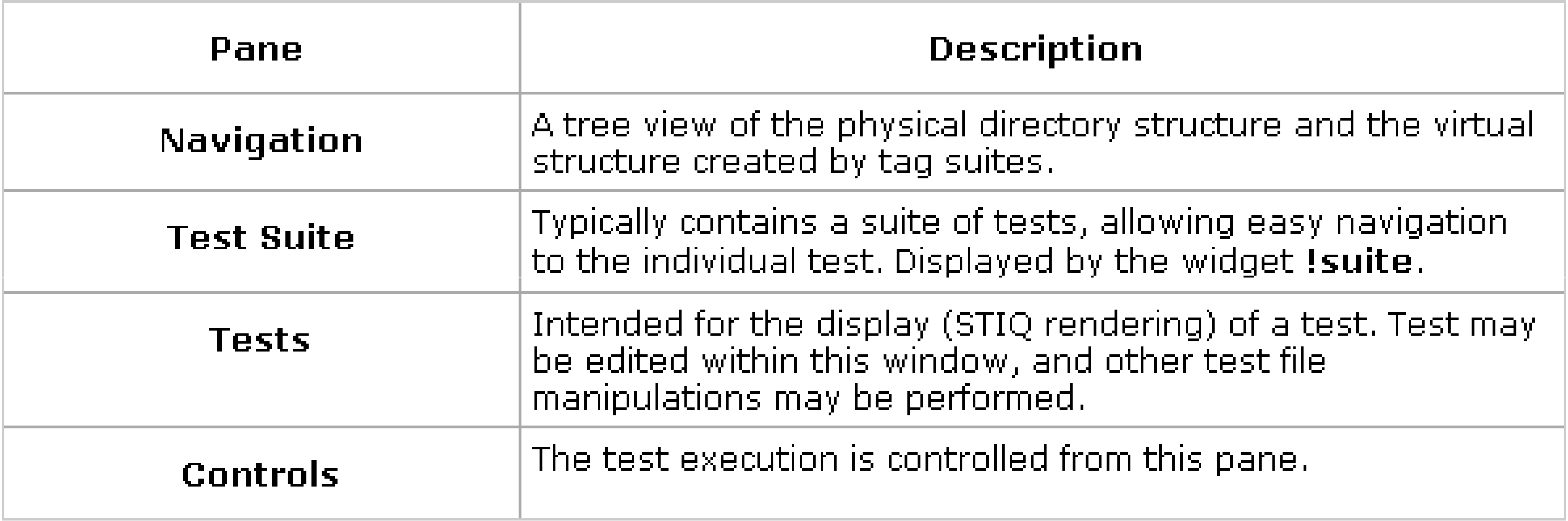
Play a Game :: Human Vs Novice

comment	Notice the first table is only one (1) cell; all others are three (3) cells wide!	
comment	The first table is the Title - it will be decorated with the status: RED/GREEN	

Tic-Tac-Toe

Player	First	Type	Wins	Record
X	●	Human	1	100%
O	○	Intermediate	0	0%
Cat			0	0%

StoryTestIQ: Panes

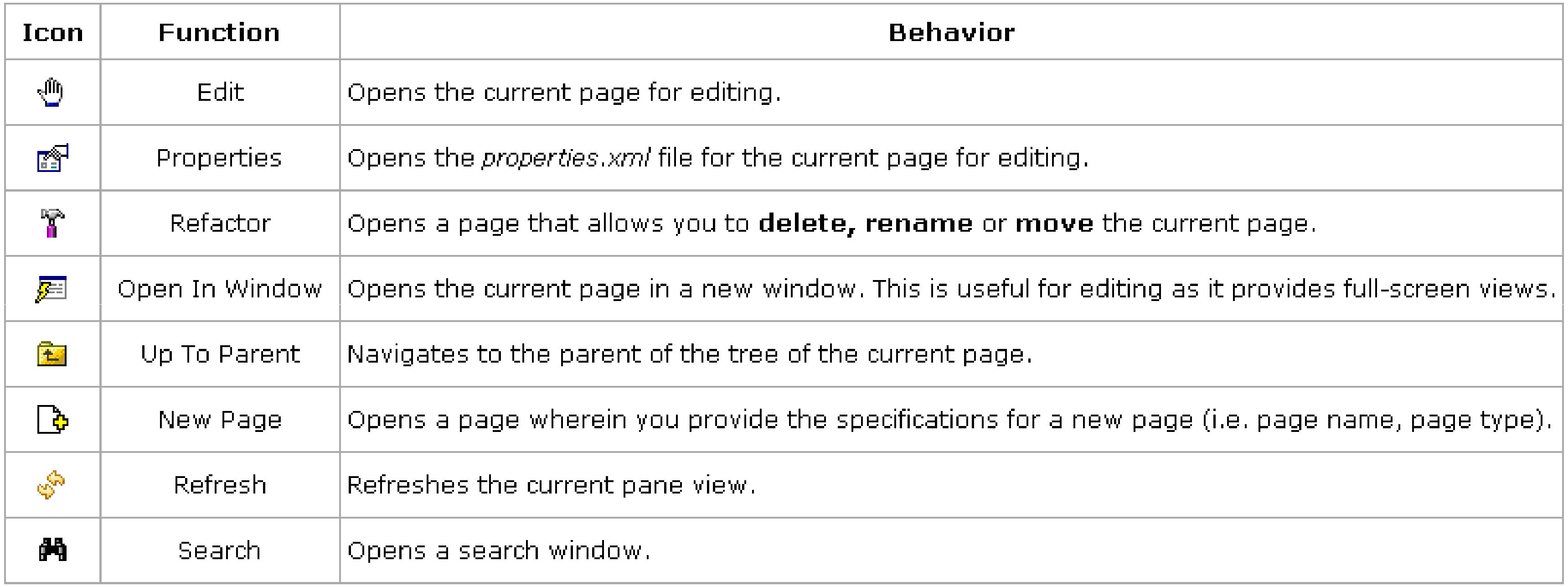


Pane	Description
Navigation	A tree view of the physical directory structure and the virtual structure created by tag suites.
Test Suite	Typically contains a suite of tests, allowing easy navigation to the individual test. Displayed by the widget **!suite**.
Tests	Intended for the display (STIQ rendering) of a test. Test may be edited within this window, and other test file manipulations may be performed.
Controls	The test execution is controlled from this pane.

StoryTestIQ: Components

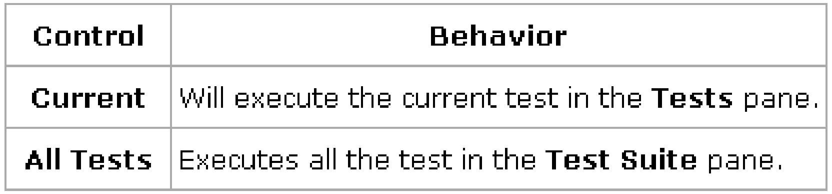
- Test
- Test Suite
- Component
- Component Suite
- Container
- Tag Suite

StoryTestIQ: Actions Toolbar

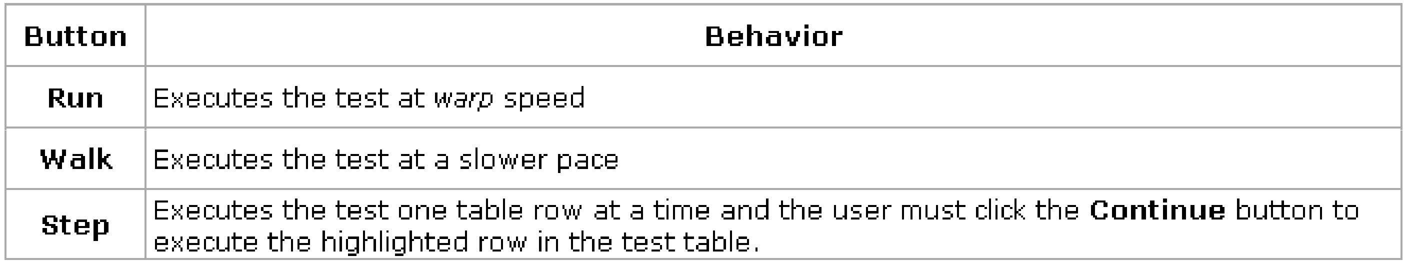


Icon	Function	Behavior
✋	Edit	Opens the current page for editing.
🗂	Properties	Opens the *properties.xml* file for the current page for editing.
🔨	Refactor	Opens a page that allows you to **delete, rename** or **move** the current page.
🗔	Open In Window	Opens the current page in a new window. This is useful for editing as it provides full-screen views.
📁	Up To Parent	Navigates to the parent of the tree of the current page.
🗋	New Page	Opens a page wherein you provide the specifications for a new page (i.e. page name, page type).
🔄	Refresh	Refreshes the current pane view.
🔭	Search	Opens a search window.

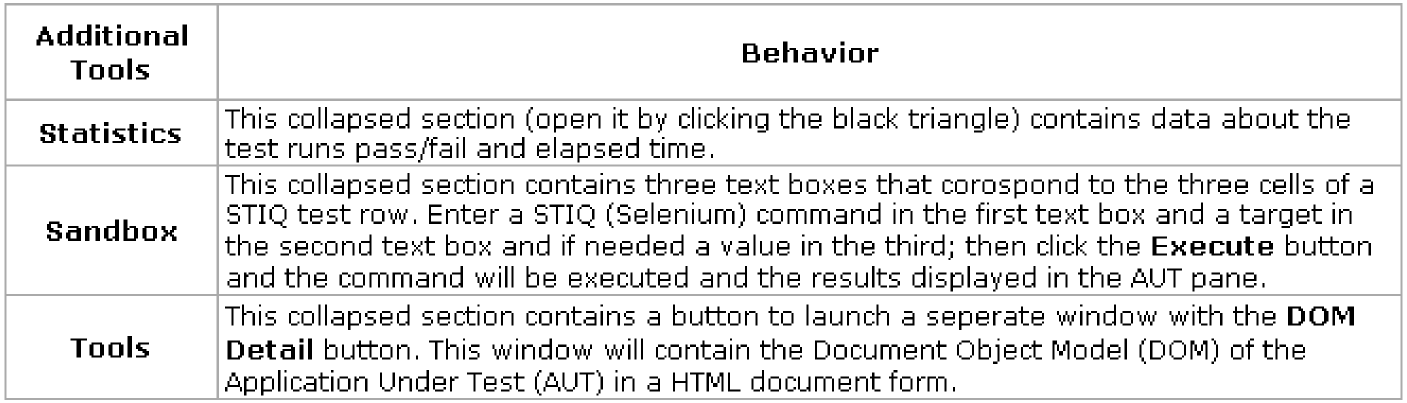
StoryTestIQ: Controls



Control	Behavior
Current	Will execute the current test in the **Tests** pane.
All Tests	Executes all the test in the **Test Suite** pane.

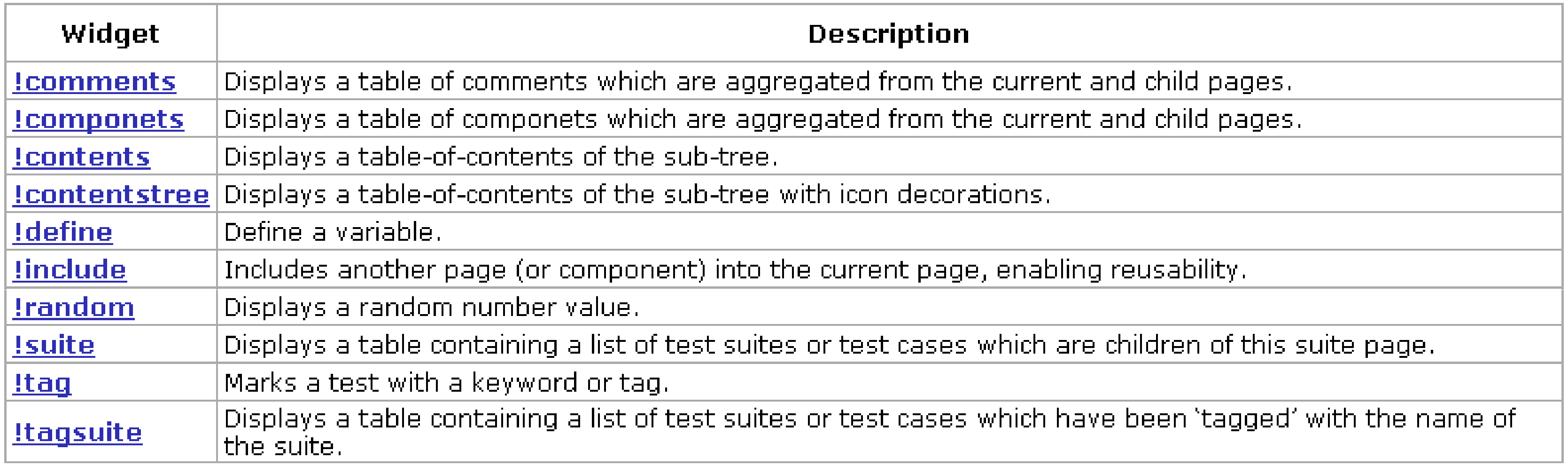


Button	Behavior
Run	Executes the test at *warp* speed
Walk	Executes the test at a slower pace
Step	Executes the test one table row at a time and the user must click the **Continue** button to execute the highlighted row in the test table.



Additional Tools	Behavior
Statistics	This collapsed section (open it by clicking the black triangle) contains data about the test runs pass/fail and elapsed time.
Sandbox	This collapsed section contains three text boxes that corospond to the three cells of a STIQ test row. Enter a STIQ (Selenium) command in the first text box and a target in the second text box and if needed a value in the third; then click the **Execute** button and the command will be executed and the results displayed in the AUT pane.
Tools	This collapsed section contains a button to launch a seperate window with the **DOM Detail** button. This window will contain the Document Object Model (DOM) of the Application Under Test (AUT) in a HTML document form.

StoryTestIQ: Essential Widgets



Widget	Description
!comments	Displays a table of comments which are aggregated from the current and child pages.
!componets	Displays a table of componets which are aggregated from the current and child pages.
!contents	Displays a table-of-contents of the sub-tree.
!contentstree	Displays a table-of-contents of the sub-tree with icon decorations.
!define	Define a variable.
!include	Includes another page (or component) into the current page, enabling reusability.
!random	Displays a random number value.
!suite	Displays a table containing a list of test suites or test cases which are children of this suite page.
!tag	Marks a test with a keyword or tag.
!tagsuite	Displays a table containing a list of test suites or test cases which have been 'tagged' with the name of the suite.

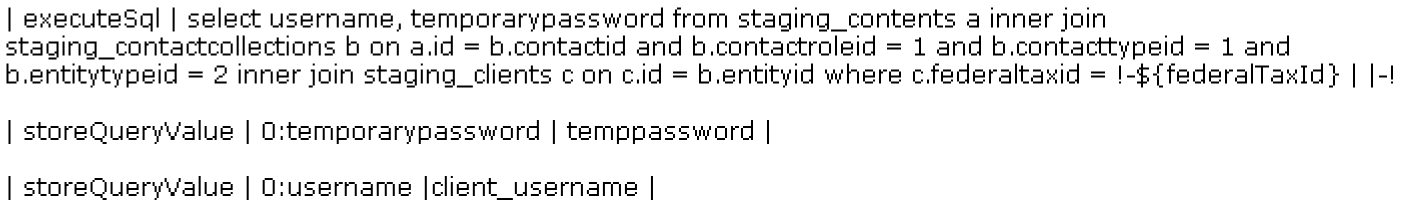
StoryTestIQ: Database Usage

- ## DB Connection

  File **storytestiq.properties** is located in the STIQ directory. Sample database settings:

```
STIQDatabaseDriver=com.microsoft.sqlserver.jdbc.SQLServerDrive
STIQDatabaseConnectionString=jdbc:sqlserver://localhost:nnnn;DatabaseName=SOMENAME
STIQDatabaseUsername=username
STIQDatabasePassword=passwor
```

- ## SQL commands
  - executeSql – executes SQL command
  - storeQueryValue – stores SQL result data from row into variable
- ## Sample



```
| executeSql | select username, temporarypassword from staging_contents a inner join
staging_contactcollections b on a.id = b.contactid and b.contactroleid = 1 and b.contacttypeid = 1 and
b.entitytypeid = 2 inner join staging_clients c on c.id = b.entityid where c.federaltaxid = !-${federalTaxId} | |-!

| storeQueryValue | 0:temporarypassword | temppassword |

| storeQueryValue | 0:username |client_username |
```

StoryTestIQ: Test Creation

- Create containers for tests structure
- Create test suite page
- Create test page
- Open test page for edit
- Create test
    - Use Selenium IDE to write Selenium test and then apply WiKi format
    - Use execute toolbox to write test step by step

StoryTestIQ: Tests structure

- **IntegrationTest** (to organize continuous integration tag-suites)
- **SprintTests** (to organize iteration tag-suites)
- **StoryTests** (to organize the tests (as opposed to tag-suites) by *feature* or *user-story* or whatever logical grouping your team comes up with)
- **TestComponents** (area for all test components (snippets of code))

StoryTestIQ: Best Practices

- Create reusable components
- Guide your HTML markup by tests
- Create tests structure for all projects
- Make tests more like specification by using comments, images, etc.
- Use execute panel to write tests

Sauce Labs

- Source IDE
  - Extend Selenium IDE
  - Allows to run tests on cloud
- Sauce Driver
- Paid OnDemand service
  - Run tests on cloud
  - Select different platforms
  - Video for each test is recorded for analysis

WebDriver/Selenium: Roadmap

- Plug-in for Internet Explorer and Safary
- Drivers for all devices and languages
- IDE improvements for WebDriver compatibility
- Fine documentation and API manuals
- Different levels of coverage

Questions & Answers