Document History

Date	Course Version No.	Software Version No.	Developer / SME	Reviewer(s)	Approver	Change Record Remarks
27 Jan, 2016	1.0	NA	Ritika Verma – (Automation CoE)	Shubhasmit Gupta (Automation CoE)	Shubhasmit Gupta (Automation CoE)	New contents created
06 Jan, 2017	1.1	NA	Ritika Verma – (Automation CoE)	Rizvan Saiyed (Automation CoE) Shubhasmit Gupta (Automation CoE) Sridatri Panda	Rizvan Saiyed (Automation CoE)	Post-integration changes made to the training material.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Course Goals and Non Goals

- Course Goals
  - At the end of this program, participants gain an understanding of how to automate test cases using Selenium testing API of a web application.
- Course Non Goals
  - This course does not cover the topics other than mentioned in course goals

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Pre-requisites

- Web Designing & Development Technologies like HTML5, CSS 3, JavaScript & XML
- Testing concepts
- Requirement Validation & Functional Decomposition
- Use case
- Defect Reporting
- Java 8 with JAXB
- Development Tools

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

Intended Audience

- Test Engineers, Software Engineers and Senior Software Engineers.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Day Wise Schedule

- Day 1 (Half Day)
  - Lesson 1: Introduction to Automation
  - Lesson 2: Introduction to Selenium
- Day 2
  - Lesson 3: Working With Selenium IDE
- Day 3
  - Lesson 3 (Contd.): Working With Selenium IDE
  - Lesson 4: Selenium 2.0 – Web Driver
- Day 4
  - Lesson 5: Testing Web Applications Using Web Driver API
- Day 5
  - Lesson 5 (Contd.): Testing Web Applications Using Web Driver API

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

Day Wise Schedule

- Day 6
  - Lesson 5 (Contd.): Testing Web Applications Using Web Driver API
- Day 7
  - Lesson 5(Contd.): Testing Web Applications Using Web Driver API
- Day 8
  - Lesson 6 Web Driver Test with Xunit
- Day 9
  - Lesson 7 (Contd.): Selenium Web Driver – Advance
- Day 10
  - Lesson 7 (Contd.): Selenium Web Driver – Advance
- Day 11
  - Lesson 8: Selenium Frameworks

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

Table of Contents

- Lesson 1: Introduction to automation
  - 1.1. Automation vs Manual
  - 1.2. What is automation
  - 1.3. What is test automation
  - 1.4. WHY and WHEN?
  - 1.5. Example Of Test Automation
- Lesson 2: Introduction to Selenium
  - 2.1. Introduction to Selenium
  - 2.2. What it is and what it is not
  - 2.3. Landscape and Usage
    - 2.3.1. Selenium IDE
    - 2.3.2. Selenium Remote Control (Selenium 1.0)
    - 2.3.3. Selenium WebDriver (Selenium 2.0)
    - 2.3.4. Selenium Grid

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

Table of Contents

- Lesson 3: Working With Selenium IDE
  - 3.1. Selenium IDE – An Introduction
  - 3.2. Installation of Selenium IDE
  - 3.3. Components of Selenium IDE
  - 3.4. Introduction to Selenium Commands – “Selenese”
  - 3.5. Understanding Element Locators in Selenium IDE
    - 3.5.1. ID
    - 3.5.2. Name
    - 3.5.3. Link Text
    - 3.5.4. CSS Selector
      - 3.5.4.1. Tag and ID
      - 3.5.4.2. Tag and class
      - 3.5.4.3. Tag and attribute
      - 3.5.4.4. Tag, class, and attribute
      - 3.5.4.5. Inner text

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Table of Contents

- 3.5.5. DOM (Document Object Model)
  - 3.5.5.1. getElementById
  - 3.5.5.2. getElementsByName
  - 3.5.5.3. dom:name
  - 3.5.5.4. dom: index
- 3.5.6. XPath
- 3.6. Working with Alerts
- 3.7. Creating Test Script using Selenium IDE
- 3.8. Creating & Executing Test Suits
- 3.9. Exporting scripts to multiple languages and Formats

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

Table of Contents

- Lesson 4: Selenium 2.0 – Web Driver
  - 4.1. Introduction To Web Driver
  - 4.2. Web Driver Vs Selenium RC Vs Selenium IDE
  - 4.3. Benefits of Web Driver over Selenium IDE and RC
  - 4.4. Limitations of Web Driver
- Lesson 5: Testing Web Applications Using Web Driver API
  - 5.1. Writing first Web Driver Test
  - 5.2. Locating UI Elements-Developers Tools
  - 5.3. Navigation API
  - 5.3.1. get
  - 5.3.2. navigate

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Table of Contents

- 5.4. Interrogation API
  - 5.4.1. getTitle
  - 5.4.2. getCurrentUrl
  - 5.4.3. getPageSource
- 5.5. WebElement API
  - 5.5.1. findElement & findElements
  - 5.5.2. By
    - 5.5.2.1. id
    - 5.5.2.2. xpath
    - 5.5.2.3. cssSelector
    - 5.5.2.4. className
    - 5.5.2.5. linkText
    - 5.5.2.6. name
    - 5.5.2.7. tagName
    - 5.5.2.8. partialLinkText

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

Table of Contents

- Lesson 5: Testing Web Applications Using Web Driver API (Cont.)
  - 5.5. WebElement API
    - 5.5.3. click
    - 5.5.4. clear
    - 5.5.5. sendKeys
    - 5.5.6. submit
    - 5.5.7. Select – selectByVisibleText etc.
    - 5.5.8. getText
    - 5.5.9. getAttribute
  - 5.6. Handling Popup Dialogs and Alerts
  - 5.7. Windows
  - 5.7.1. getWindowHandle and getWindowHandles
  - 5.7.2. switchTo
  - 5.7.3. manage

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

Table of Contents

- 5.8. Alerts
  - 5.8.1. switchTo
  - 5.8.2. dismiss
  - 5.8.3. accept
- 5.9. Synchronization
- 5.10. Why synchronization is important

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

Table of Contents

- Lesson 5: Testing Web Applications Using Web Driver API
  - 5.11. Using Explicit & Implicit Wait
    - 5.11.1. ExpectedCondition & ExpectedConditions
  - 5.11.2. WebDriverWait
  - 5.11.3. ImplicitlyWait
  - 5.11.4. pageLoadTimeout
  - 5.12. JavaScriptExecutor
- Lesson 6: Web Driver Test with Xunit
  - 6.1 Introduction to Xunit and Junit
  - 6.2 Junit Annotations
  - 6.3. Assertions/Verifications –JUnit or TestNG
    - 6.1.1. Webdriver testcases with JUnit or TestNG
    - 6.1.2. Test Suite Lesson

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Table of Contents

- Lesson 7: Selenium Web Driver – Advance
  - 7.1. Selenium: How it works
  - 7.2. Different drivers
    - 7.2.1. Firefox
    - 7.2.2. Chrome
    - 7.2.3. Internet Explorer
    - 7.2.4. Headless Browser
      - 7.2.4.1. Ghost Driver and Phantom JS
    - 7.2.5. Mobile Browsers
      - 7.2.5.1. Selendriod & Appium

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Table of Contents

- Lesson 7: Selenium Web Driver – Advance
  - 7.3. Remote Web Driver
  - 7.4. Capabilities
  - 7.5. Profile setting
  - 7.6. Selenium Grid
- Lesson 8: Selenium Frameworks
  - 8.1 Framework Overview
  - 8.2. Data Driven (Excel, Databases)
  - 8.3. Keyword Driven
  - 8.4. Component based (Sprintest®/CBF)
  - 8.5. Reports (Excel, PDF)
  - 8.6. TDD (Junit, TestNG)
  - 8.7. BDD (Cucumber, SpecFlow)
  - 8.8. ATDD (Fitnesse)
  - 8.9. CI Tools (Jenkins etc)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

References

- NA

Next Step Courses (if applicable)

- NA

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Other Parallel Technology Areas

- NA

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

Test Automation & Advanced Selenium

Lesson 1: Introduction to Automation

Lesson Objectives

- Automation vs. Manual
- What is Automation
- What is Test Automation
- Why And When
- Example Of Test Automation

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

1.1: Introduction to Automation

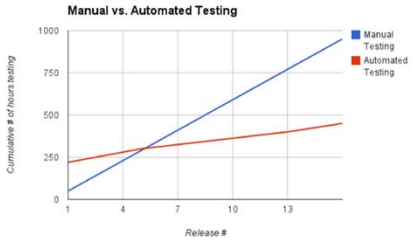
Automation vs. Manual

MANUAL TESTING

- Testing is time consuming and tedious
- Delay the ability in thoroughly testing an application
- Critical bugs escape undetected
- What happens when multiple platforms involved

AUTOMATED TESTING

- Higher efficiency
- Higher product quality
- Easy to focus on all possible workflows
- Delivers: Reusability, Consistency and Productivity



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

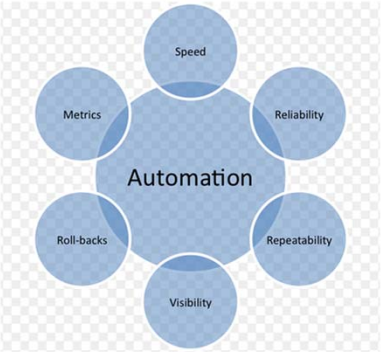
 3

1.1: Introduction To Automation

What is Automation

- "The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is automation applied to an inefficient operation will magnify the inefficiency."

-Bill Gates



Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

 4

Automation is the linking of disparate systems and software in such a way that they become self-acting or self-regulating.

1.1: Introduction To Automation

What is Test Automation

- The method which/that takes automation tool's support to execute the test cases is known as Automation Testing.
- It is a method which
  - uses automation tools to run tests that repeat predefined actions
  - matches the developed program's probable and real results
- If the project prospects and results align, your project is behaving as it should, and you are likely bug free. If the two don't align, still, there is a problem that requires to be addressed. You'll have to take a look at your code, alter it, and continue to run tests until the actual and expected outcomes align.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Test Automation and its Benefits:

Why automate Testing?
In today´s fast moving world, it is a challenge for any company to continuously maintain and improve the quality and efficiency of software systems development. In many software projects, testing is neglected because of time or cost constraints. This leads to a lack of product quality, followed by customer dissatisfaction and ultimately to increased overall quality costs.

The main reasons for these added costs are primarily:
1.poor test strategy
2.underestimated effort of test case generation
3.delay in testing
4.subsequent test maintenance

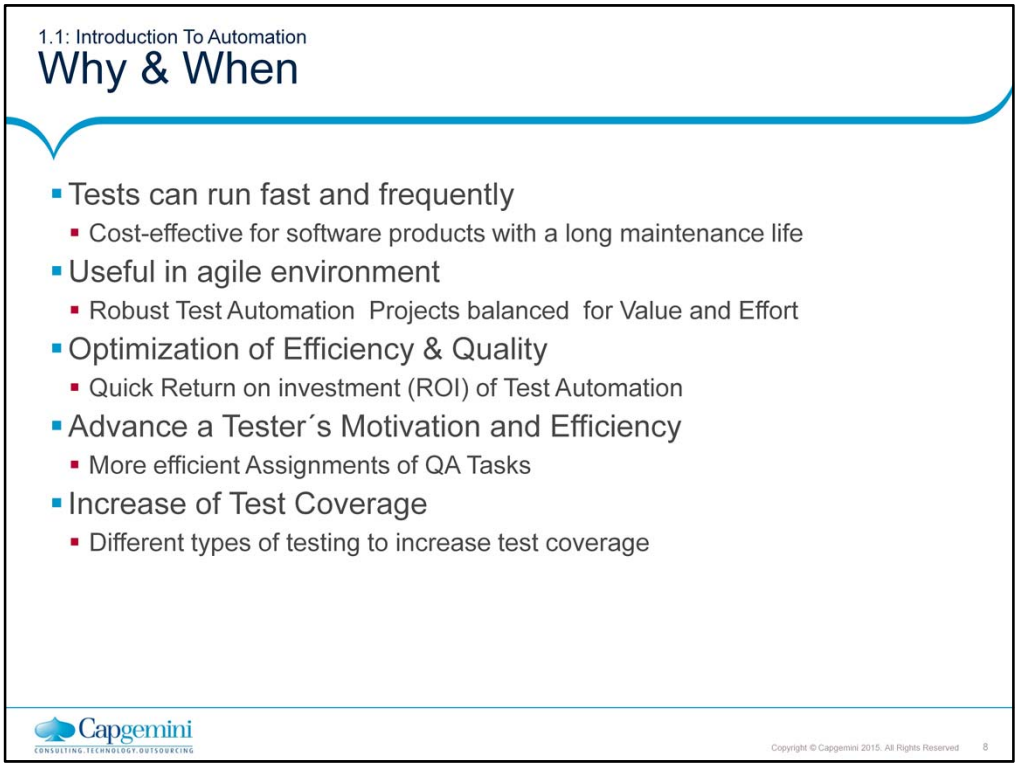
Test automation can improve the development process of a software product in many cases. The automation of tests is initially associated with increased effort, but the related benefits will quickly pay off. Automated tests can run fast and frequently, which is cost-effective for software products with a long maintenance life. When testing in an agile environment, the ability to quickly react to ever-changing software systems and requirements is necessary. New test cases are generated continuously and can be added to existing automation in parallel to the development of the software itself.
In both manual and automated testing environments test cases need to be modified for extended periods of time as the software project progresses. It is important to be aware that complete coverage of all tests using test automation is unrealistic. When deciding what tests to automate first, their value vs. the effort to create them needs to be considered. Test cases with high value and low effort should be automated first. Subsequently test cases with frequent use, changes, and past errors; as well as test cases with low to moderate effort in setting up the test environment and developing the automation project are best suited for automation.

Optimization of Speed, Efficiency, Quality and the Decrease of Costs:
The main goal in software development processes is a timely release. Automated tests run fast and frequently, due to reused modules within different tests. Automated regression tests which ensure the continuous system stability and functionality after changes to the software were made lead to shorter development cycles combined with better quality software and thus the benefits of automated testing quickly outgain the initial costs.

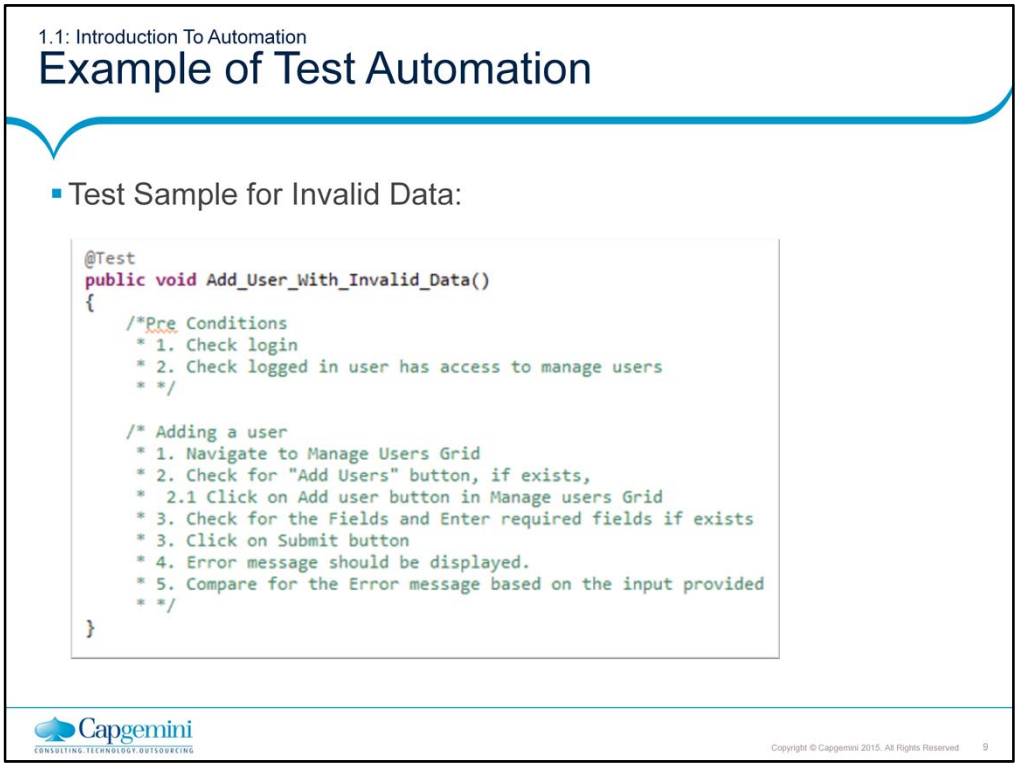
Advance a Tester´s Motivation and Efficiency:
Manual testing can be mundane, error-prone and therefore become exasperating. Test automation alleviates testers' frustrations and allows the test execution without user interaction while guaranteeing repeatability and accuracy. Instead testers can now concentrate on more difficult test scenarios.

Increase of Test Coverage:
Sufficient test coverage of software projects is often achieved only with great effort. Frequent repetition of the same or similar test cases is laborious and time consuming to perform manually.

Some examples are:
Regression test after debugging or further development of software
Testing of software on different platforms or with different configurations
Data-driven testing (creation of tests using the same actions but with many different inputs)
Test automation allows performing different types of testing efficiently and effectively.

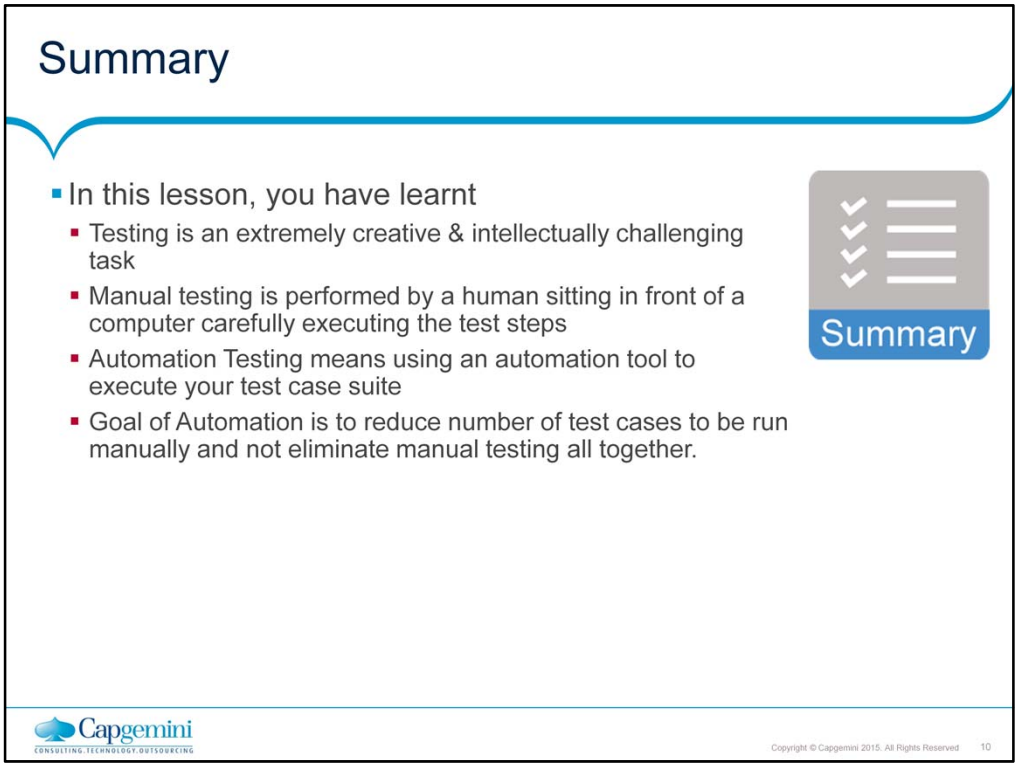


Add the notes here.



In Above example “Add Users With Valid Data”

To execute the test case, we need to login to the application. We also need to check if the user is already logged in or not. And the other we need to check is if the logged in person is having access to “Add Users” or Not. If the above Two Conditions are passed then we should execute the rest Else we should return the test as failed.



Add the notes here.

Review Question

Question 1

Why would you want to automate a test? Is it to:

- Increase test coverage?
- Improve quality?
- Save time for exploratory testing?
- Find more bugs?
- Replace manual testers?

Question 2: True/False

- Automation Testing uses automation tools to run tests that repeat predefined actions.

Question 3: Fill in the Blanks

- Automation Testing delivers _____, consistency and productivity.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

Test Automation & Advanced Selenium

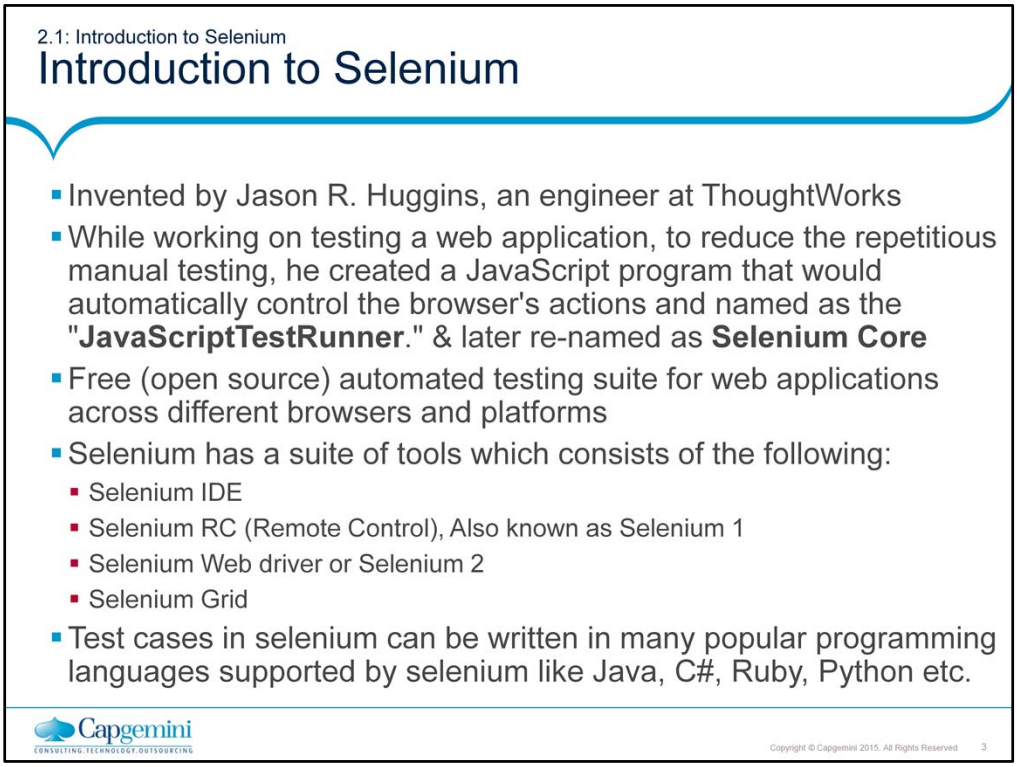
Lesson 2: Introduction to Selenium

Lesson Objectives

- Introduction to Selenium
- What it is and what it is not
- Landscape and Usage
  - Selenium IDE
  - Selenium Remote Control (Selenium 1.0)
  - Selenium Web Driver (Selenium 2.0)
  - Selenium Grid

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2


2.1: Introduction to Selenium
Introduction to Selenium
Invented by Jason R. Huggins, an engineer at ThoughtWorks
While working on testing a web application, to reduce the repetitious manual testing, he created a JavaScript program that would automatically control the browser's actions and named as the "JavaScriptTestRunner." & later re-named as Selenium Core
Free (open source) automated testing suite for web applications across different browsers and platforms
Selenium has a suite of tools which consists of the following:
Selenium IDE
Selenium RC (Remote Control), Also known as Selenium 1
Selenium Web driver or Selenium 2
Selenium Grid
Test cases in selenium can be written in many popular programming languages supported by selenium like Java, C#, Ruby, Python etc.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
3

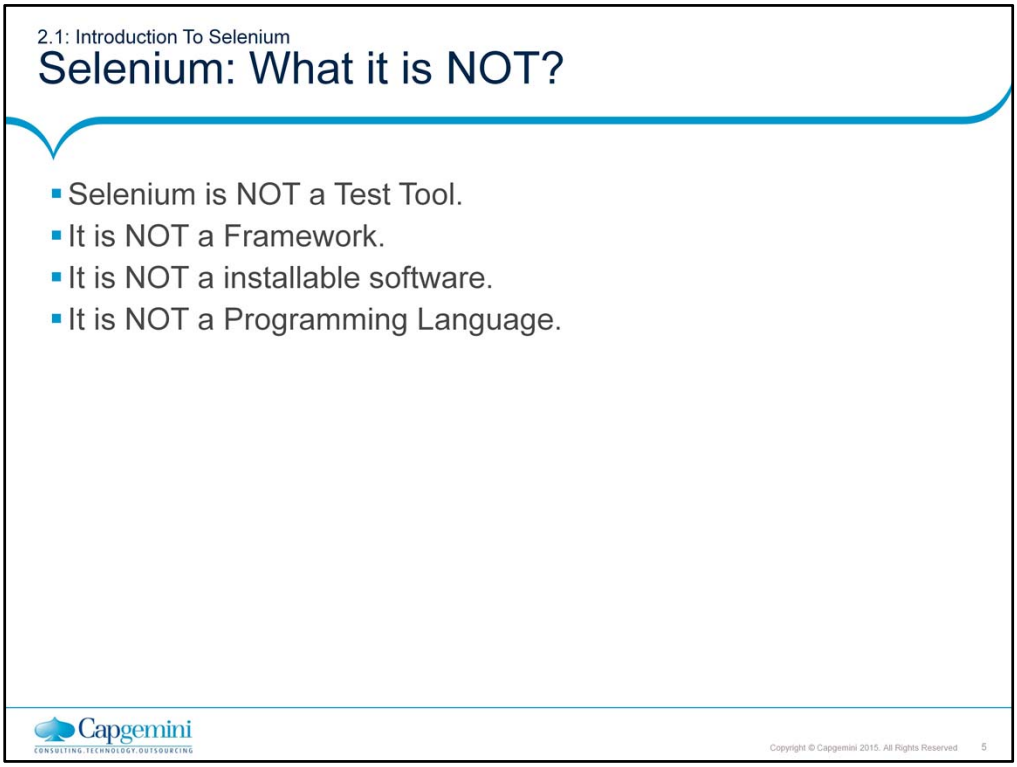
2.1: Introduction To Selenium

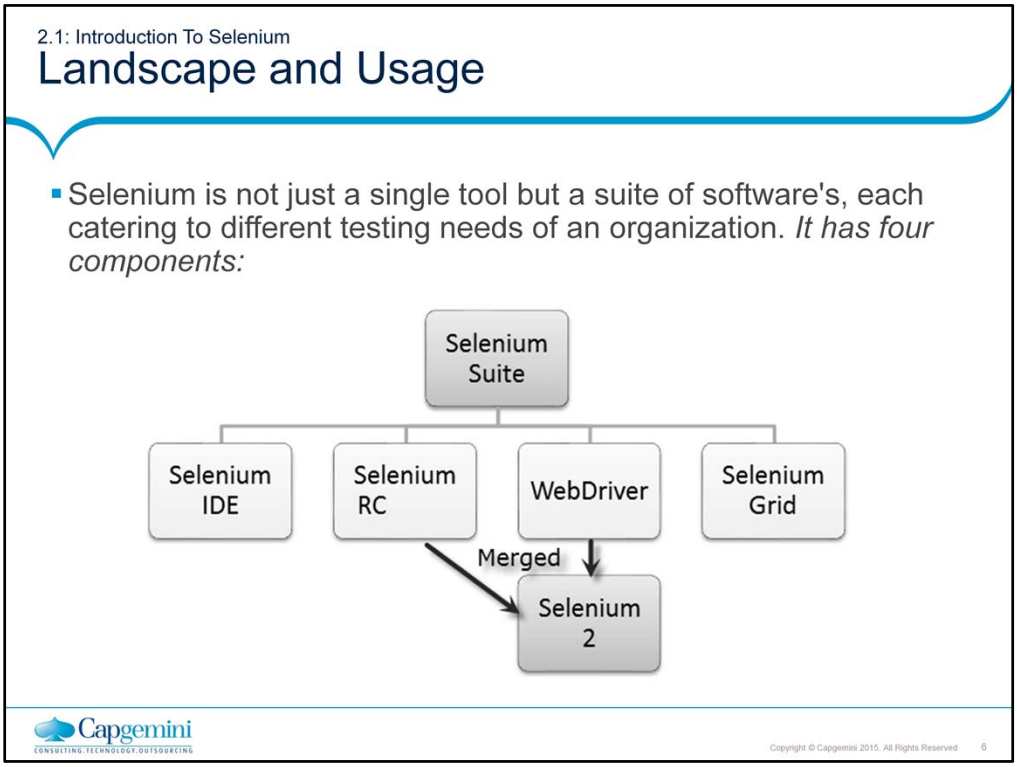
Selenium: What it is?

- Selenium automates browsers. That's it!
- Selenium is a portable testing API for web applications.
- Web Application: Pure HTML & JavaScript (No Flash, ActiveX, Silverlight etc.)
- Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that.
- It is Open Source and Freeware.
- Supports multiple browsers & OS.
  - Browsers: Firefox, Chrome, IE, Safari, Opera, PhantomJS
  - OS: Windows, MAC, Linux, Android, iOS
- In process Headless execution.

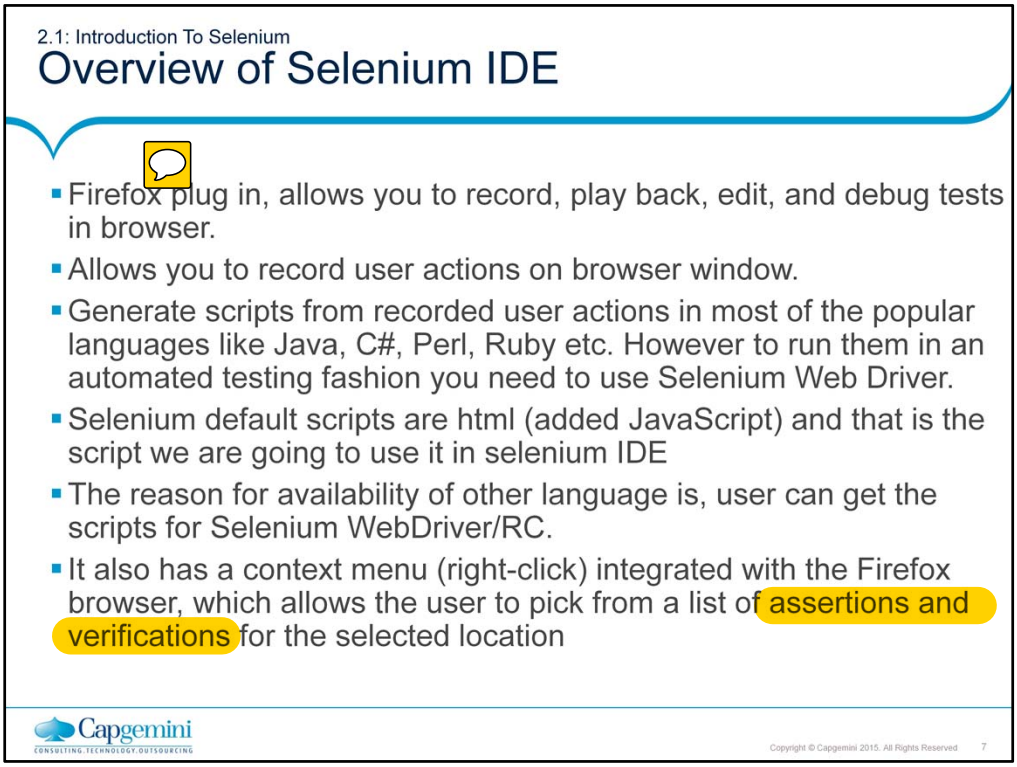
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4


2.1: Introduction To Selenium
Selenium: What it is NOT?
Selenium is NOT a Test Tool.
It is NOT a Framework.
It is NOT a installable software.
It is NOT a Programming Language.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
5



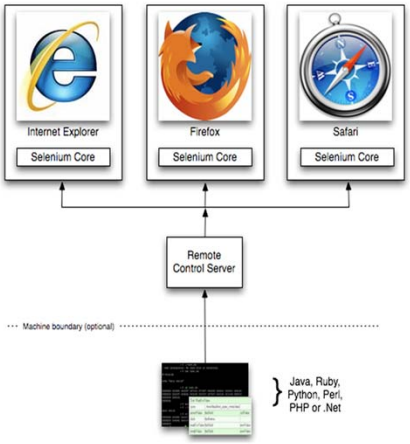
Add the notes here.


2.1: Introduction To Selenium
Overview of Selenium IDE
Firefox plug in, allows you to record, play back, edit, and debug tests in browser.
Allows you to record user actions on browser window.
Generate scripts from recorded user actions in most of the popular languages like Java, C#, Perl, Ruby etc. However to run them in an automated testing fashion you need to use Selenium Web Driver.
Selenium default scripts are html (added JavaScript) and that is the script we are going to use it in selenium IDE
The reason for availability of other language is, user can get the scripts for Selenium WebDriver/RC.
It also has a context menu (right-click) integrated with the Firefox browser, which allows the user to pick from a list of assertions and verifications for the selected location
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
7

2.1: Introduction To Selenium

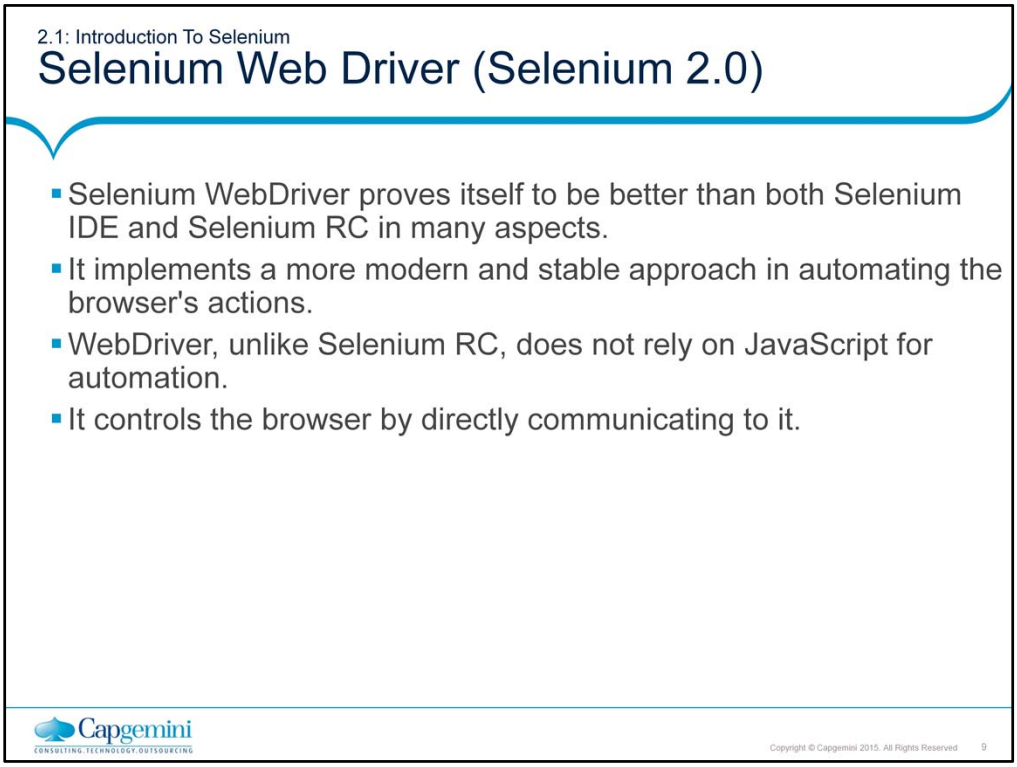
Selenium Remote Control (Selenium 1.0)

- Selenium Remote Control (RC) is a test tool that allows you to write automated web application UI tests in any programming language against any HTTP website using any mainstream JavaScript-enabled browser.
- Selenium RC comes in two parts:
  - A server which automatically launches and kills browsers, and acts as a HTTP proxy for web requests from them.
  - Client libraries for your favorite computer language.



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

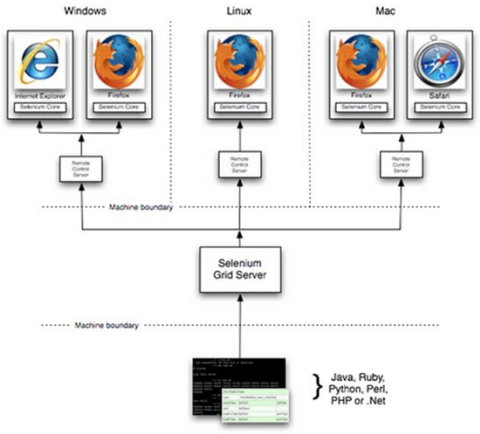
 8


2.1: Introduction To Selenium
Selenium Web Driver (Selenium 2.0)
Selenium WebDriver proves itself to be better than both Selenium IDE and Selenium RC in many aspects.
It implements a more modern and stable approach in automating the browser's actions.
WebDriver, unlike Selenium RC, does not rely on JavaScript for automation.
It controls the browser by directly communicating to it.
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
9

2.1: Introduction To Selenium

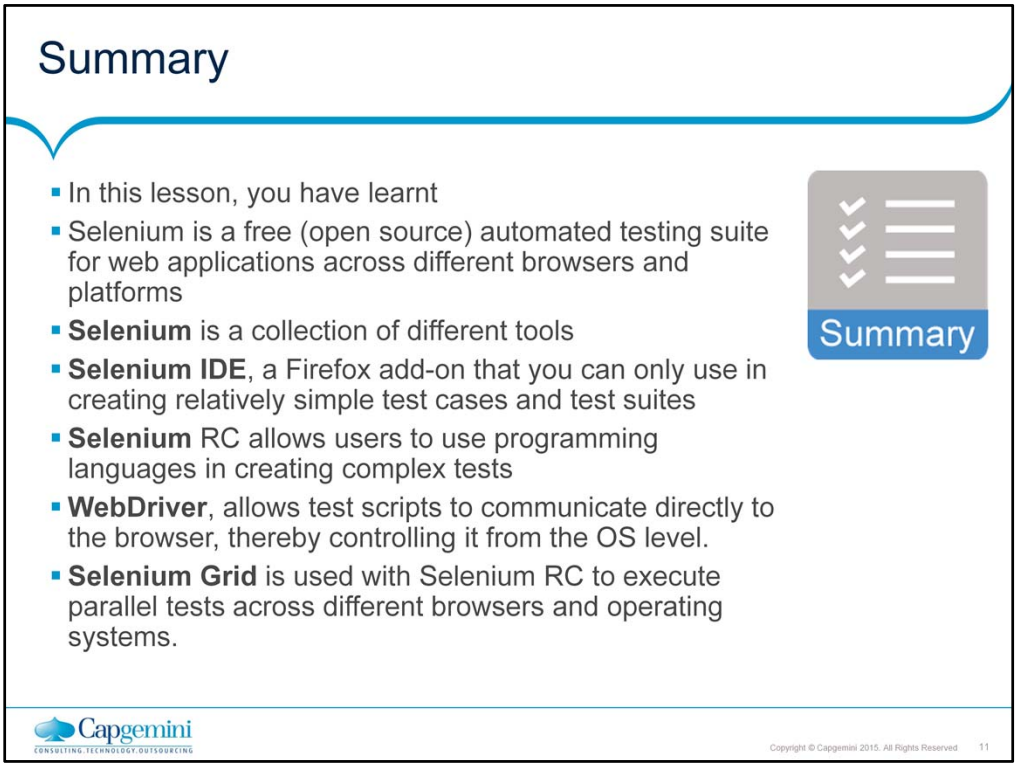
Selenium Grid

- Runs multiple tests on different machines against different browsers and operating systems in parallel
- Support s distributed test execution.
- Run tests in a distributed test execution environment



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10



Add the notes here.

Review Question

Question 1

- Select the component which is NOT part of Selenium suite
  - Selenium IDE
  - Selenium RC
  - Selenium Grid
  - Selenium Web

Question 2: True/False

- Selenium is a set of tools that supports rapid development of test automation scripts for web based applications.

Question 3: Fill in the Blanks

- WebDriver, unlike Selenium RC, does not rely on _____ for automation.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Test Automation & Advanced Selenium

Lesson 3: Working With Selenium IDE

Lesson Objectives

- Selenium IDE – An Introduction
- Installation of Selenium IDE
- Components of Selenium IDE
- Introduction to Selenium Commands – “Selenese”
- Understanding Element Locators in Selenium IDE
  - ID
  - Name
  - Link Text
  - CSS Selector
    - Tag and ID
    - Tag and class
    - Tag and attribute
    - Tag, class, and attribute
    - Inner text

Lesson Objectives

- DOM (Document Object Model)
  - getElementById
  - getElementsByName
  - dom:name
  - dom:index
- Working with Alerts
- Creating Test Script using Selenium IDE
- Creating & Executing Test Suits
- Exporting scripts to multiple languages and Formats

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

3.1: Working With Selenium IDE

Selenium IDE – An Introduction

- Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE).
- Selenium IDE, fully-featured Integrated Development Environment (IDE)
- Installs as a plugin in Mozilla Firefox
- Enables developers to test their web applications through Selenium
- Records user interactions with the web browser and play back to test for errors
- Effortless to install and easy to learn

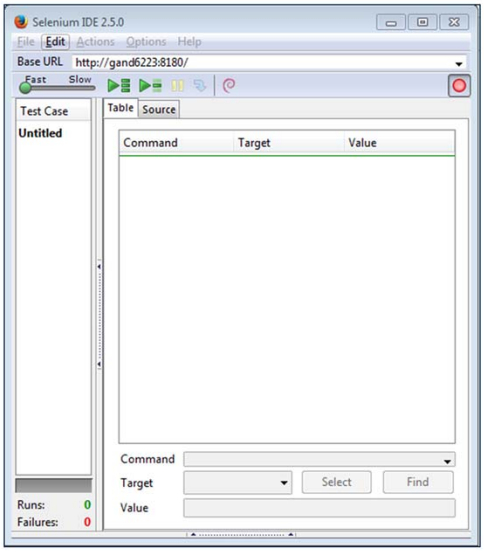
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

3.1: Working With Selenium IDE

Installation of Selenium IDE

- Download Selenium IDE
- Copy Selenium IDE file to extensions folder of Mozilla Firefox.
- To facilitate opening of .xpi file for selenium you can drag and drop the .xpi file on the Mozilla browser to install it and get it in the add-ons of the browser

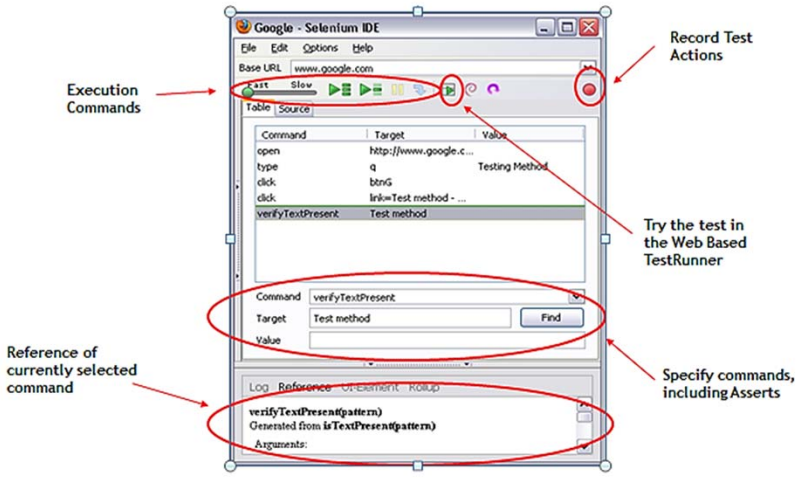


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

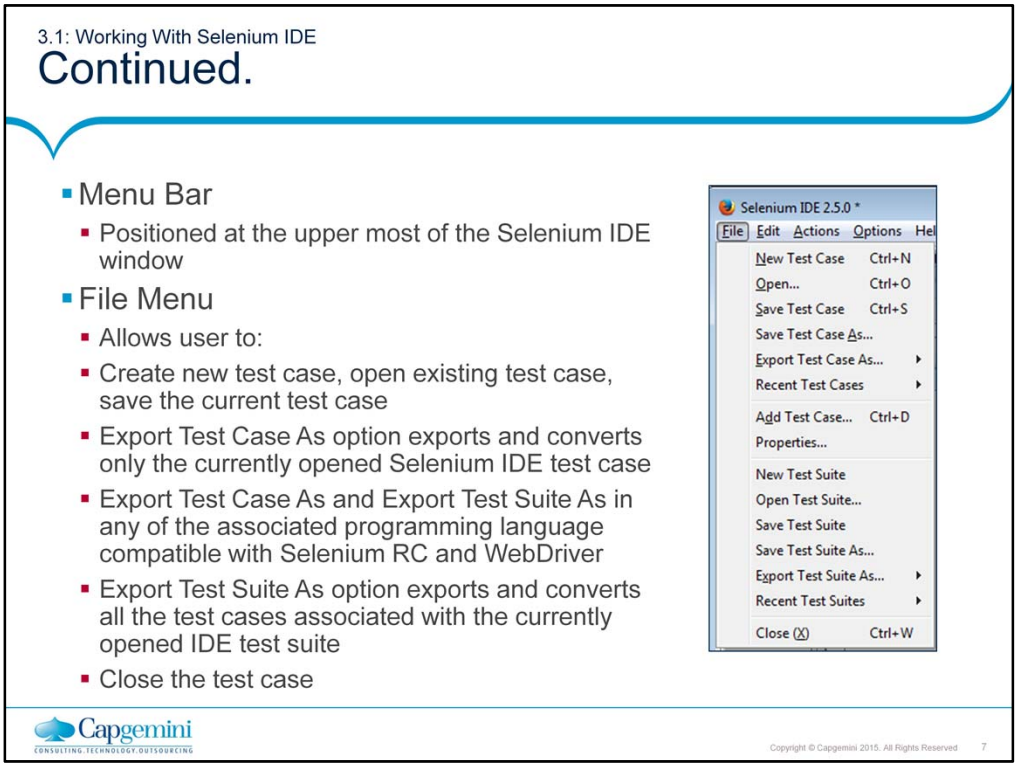
3.1: Working With Selenium IDE

Components of Selenium IDE



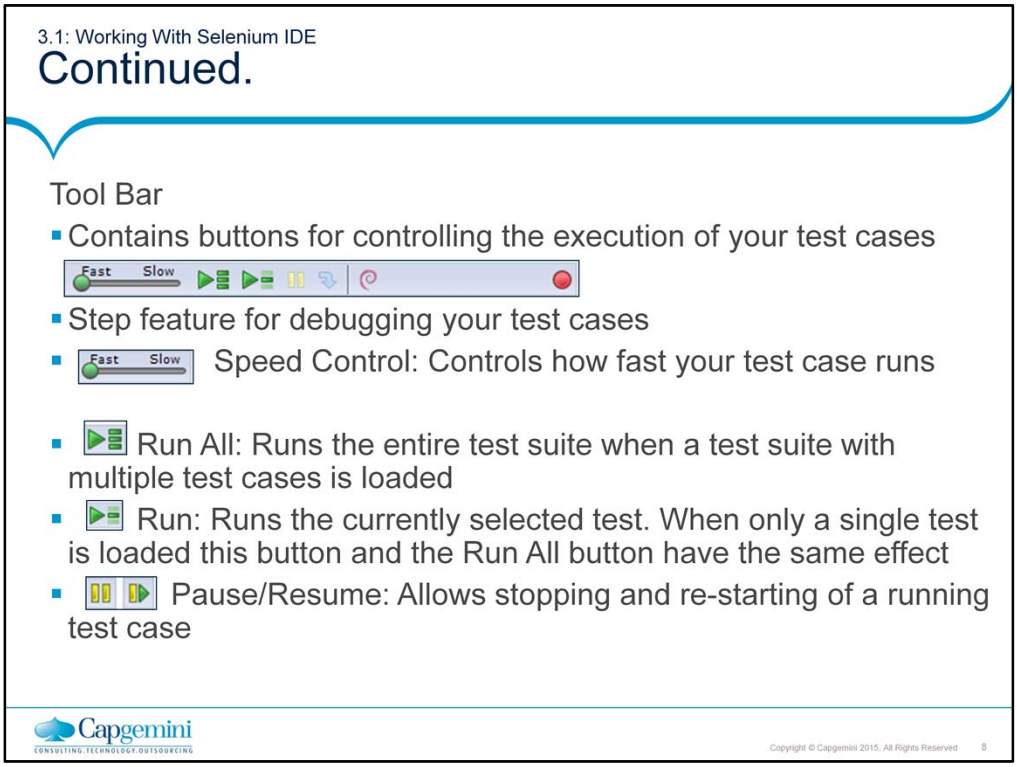
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

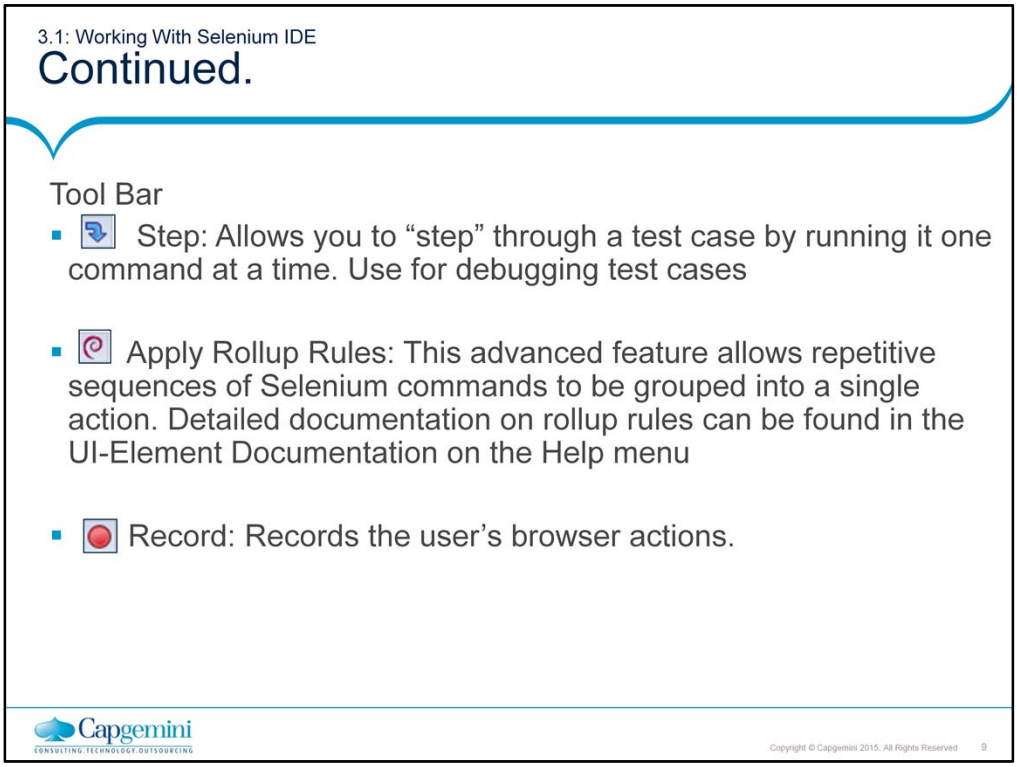


1.File Menu

It is very much analogous to the file menu belonging to any other application. Export Test Case As and Export Test Suite give the liberty to the user to prefer amid the available unit testing frameworks like jUnit, TestNG etc.
Thus an IDE test case can be exported for a chosen union of programming language, unit testing framework and tool from the selenium package.



The Selenium IDE test cases can be saved into following format:
HTML format
The Selenium IDE test cases can be exported into following formats/programming languages.
java (IDE exported in Java)
rb (IDE exported in Ruby)
py (IDE exported in Python)
cs (IDE exported in C#)



The Selenium IDE test cases can be saved into following format:
HTML format
The Selenium IDE test cases can be exported into following formats/programming languages.
java (IDE exported in Java)
rb (IDE exported in Ruby)
py (IDE exported in Python)
cs (IDE exported in C#)

3.1: Working With Selenium IDE

Continued.

- Test Case Pane
- Script is displayed in the test case pane
- It has two tabs:
  - Displays the command and their parameters in a readable “Table” format
  - “Source” displays the test case in the native format in which the file will be stored
    - By default, this is HTML although it can be changed to a programming language such as Java or C#, or a scripting language like Python

Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

3.1: Working With Selenium IDE

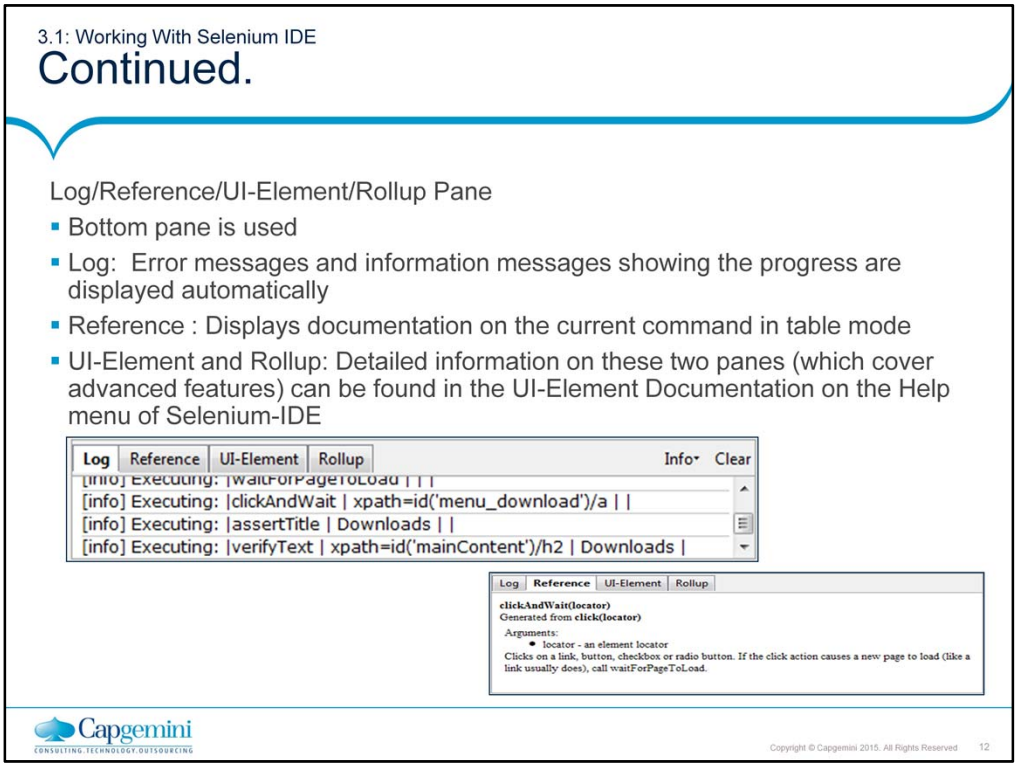
Continued.

Command, Target and Value

- Command displays the currently selected command along with its parameters
- Target displays unique tag value for the selected field
- Value contains the data in case of text box fields

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

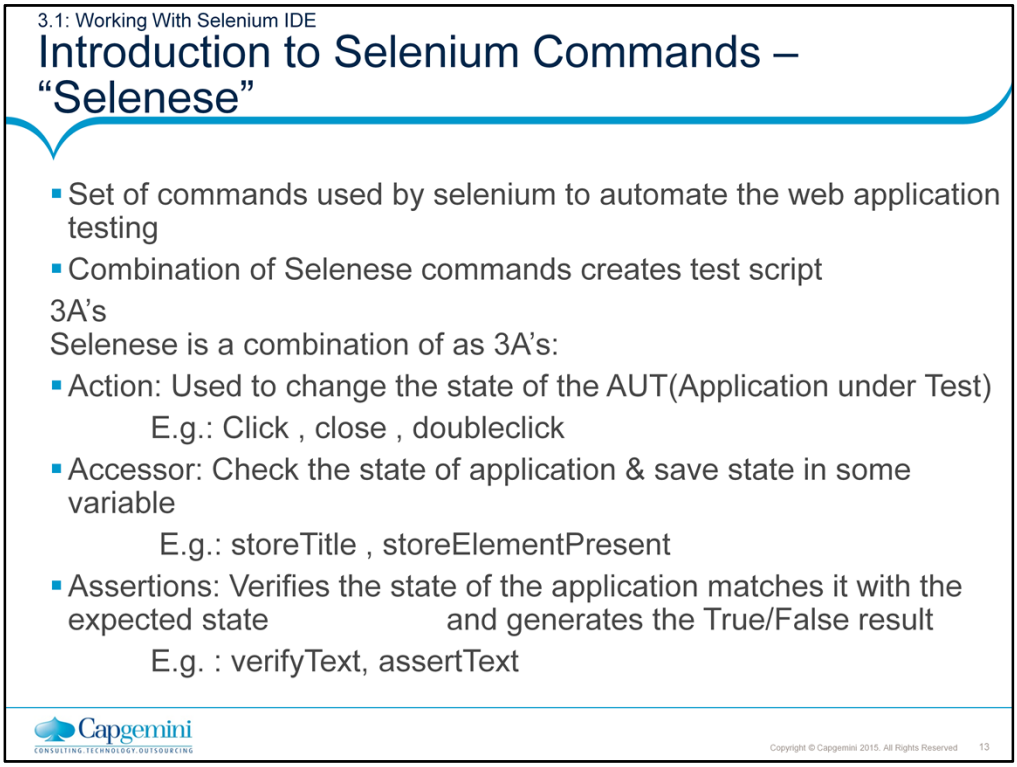


Log/Reference/UI-Element/Rollup Pane
The bottom pane is used for four different functions–Log, Reference, UI-Element, and Rollup–depending on which tab is selected.

Log
When you run your test case, error messages and information messages showing the progress are displayed in this pane automatically, even if you do not first select the Log tab. These messages are often useful for test case debugging. Notice the Clear button for clearing the Log. Also notice the Info button is a drop-down allowing selection of different levels of information to log.

Reference
The Reference tab is the default selection whenever you are entering or modifying Selenese commands and parameters in Table mode. In Table mode, the Reference pane will display documentation on the current command. When entering or modifying commands, whether from Table or Source mode, it is critically important to ensure that the parameters specified in the Target and Value fields match those specified in the parameter list in the Reference pane. The number of parameters provided must match the number specified, the order of parameters provided must match the order specified, and the type of parameters provided must match the type specified. If there is a mismatch in any of these three areas, the command will not run correctly.

While the Reference tab is invaluable as a quick reference, it is still often necessary to consult the Selenium Reference document.



Action:
Used to change the state of the AUT(Application under Test)like click on some link, type a value in edit box, select some options on the page, select a value from drop down etc.
When action is performed on AUT the test will fail if the action is not achieved.

Accessor:
This command check the state of the application and save the application state in some variable. It can be used for automatic generation of assertions.

Assertions:
Are very similar to checkpoint in UFT/QTP. Assertion verifies the state of the application matches it with the expected state and generates the True/False result.

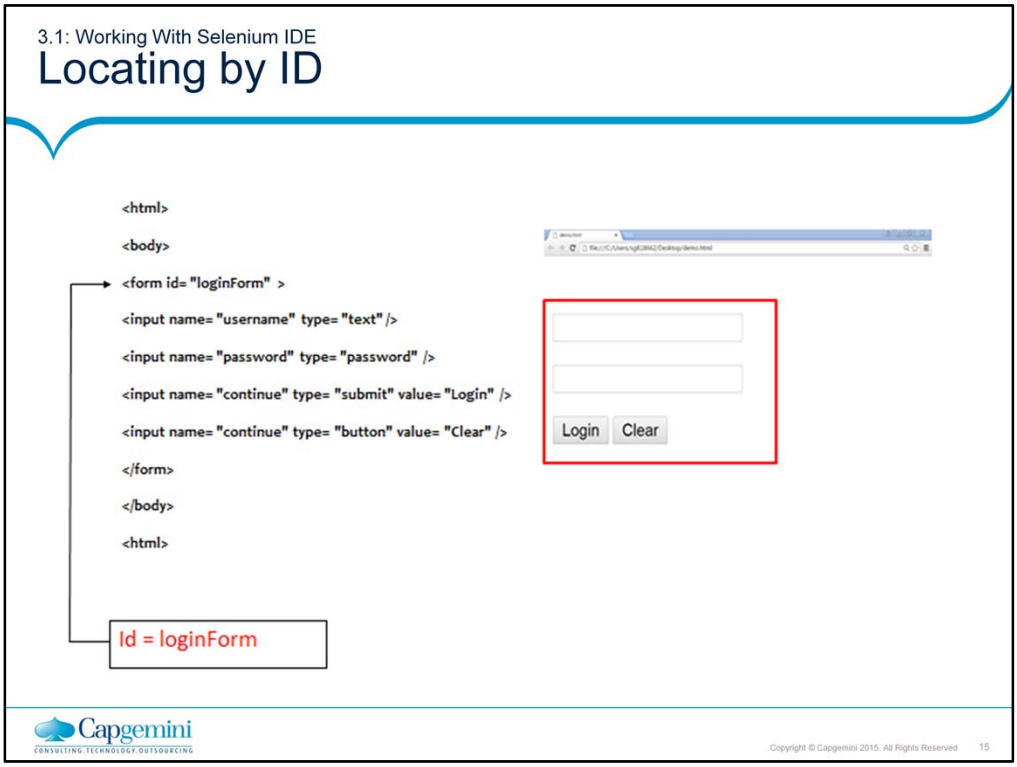
3.1: Working With Selenium IDE

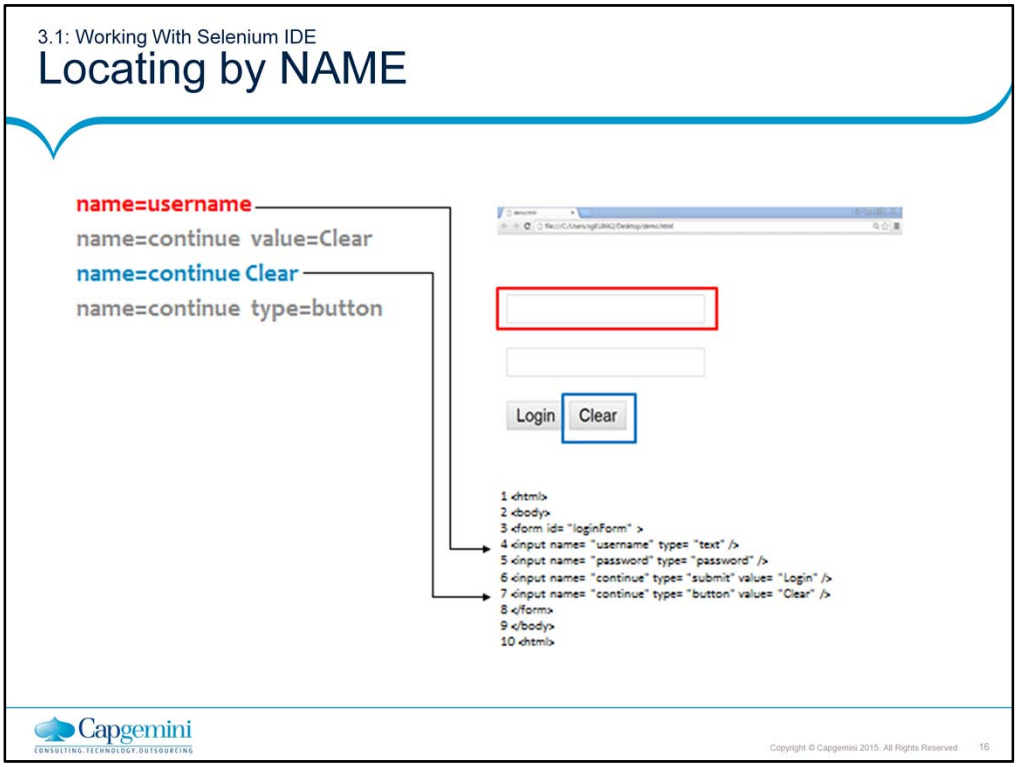
Understanding Element Locators in Selenium IDE

- Types of Element Locators
- ID
- Name
- Link Text
- CSS Selector
  - Tag and ID
  - Tag and class
  - Tag and attribute
  - Tag, class, and attribute
  - Inner text
- DOM (Document Object Model)
  - getElementById
  - getElementsByName
  - dom:name
  - dom: index
- XPath

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14


3.1: Working With Selenium IDE
Locating by ID
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
Login
Clear
Id = loginForm
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
15


3.1: Working With Selenium IDE
Locating by NAME
name=username
name=continue value=Clear
name=continue Clear
name=continue type=button
Login
Clear
1 <html>
2 <body>
3 <form id= "loginForm" >
4 <input name= "username" type= "text" />
5 <input name= "password" type= "password" />
6 <input name= "continue" type= "submit" value= "Login" />
7 <input name= "continue" type= "button" value= "Clear" />
8 </form>
9 </body>
10 <html>
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
16

3.1: Working With Selenium IDE

Locating by Link Text

link=Link1

link=Link3


```
1 <html>
2 <body>
3 <a href="#">Link1</a>
4 <a href="#">Link2</a>
5 <a href="#">Link3</a>
6 <form id= "loginForm" >
7 <input name= "username" type= "text" />
8 <input name= "password" type= "password" />
9 <input name= "continue" type= "submit" value= "Login" />
10 <input name= "continue" type= "button" value= "Clear" />
11 </form>
12 </body>
13 <html>
```

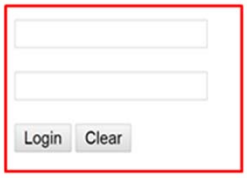
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

3.1: Working With Selenium IDE

Locating by CSS Selector - Tag and ID

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text" />
<input name= "password" type= "password" />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



css = form# loginForm

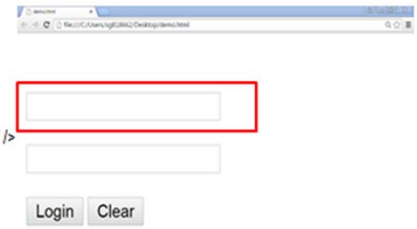
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

3.1: Working With Selenium IDE

Locating by CSS Selector - Tag and Class

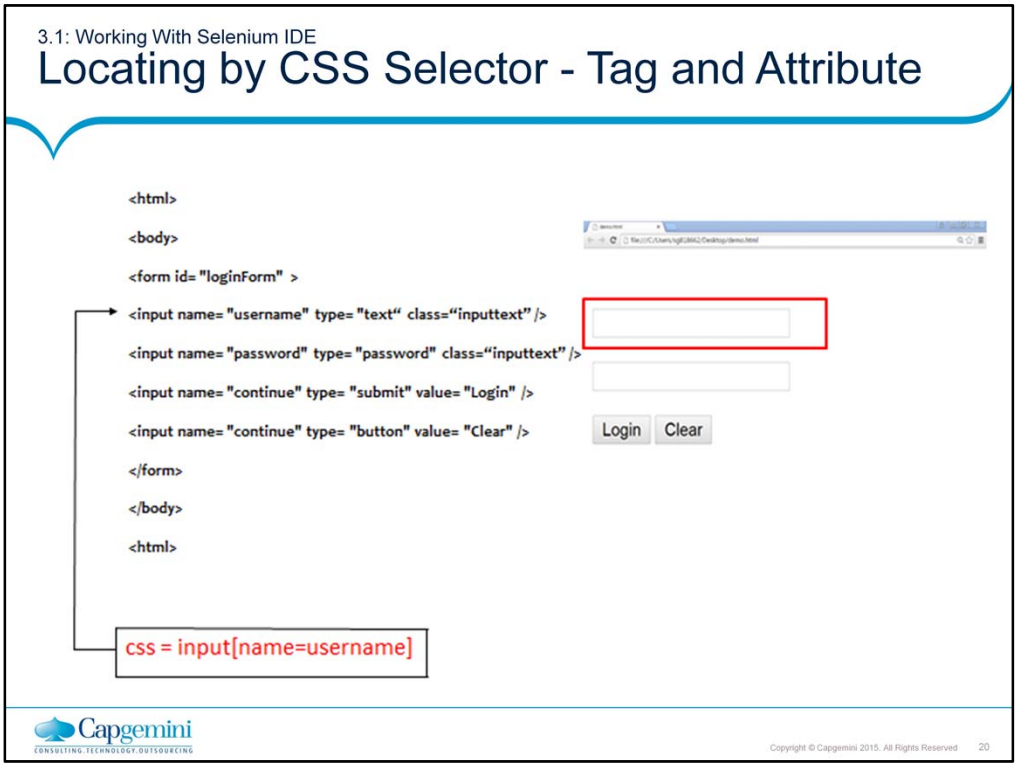
```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



css = input.inputtext

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19


3.1: Working With Selenium IDE
Locating by CSS Selector - Tag and Attribute
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
Login
Clear
css = input[name=username]
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
20

3.1: Working With Selenium IDE

Locating by CSS Selector – Tag, Class and Attribute

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```

Login Clear

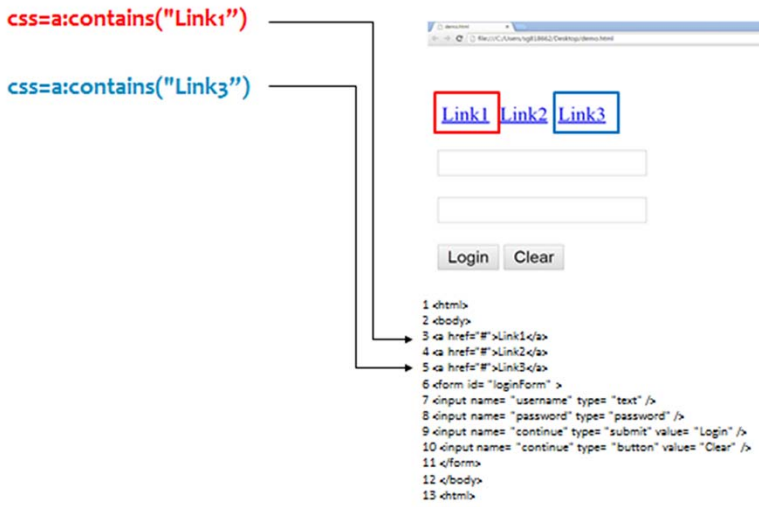
css = input.inputtext[name=username]

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

3.1: Working With Selenium IDE

Locating by CSS Selector – Inner text



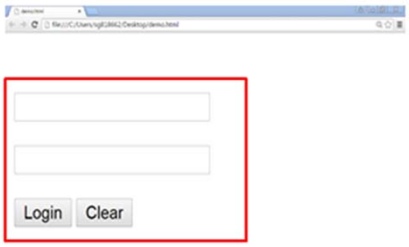
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

3.1: Working With Selenium IDE

Locating by DOM – getElementsById

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```




```
document.getElementById(" loginForm ")
```

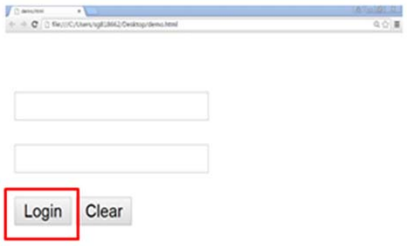
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

3.1: Working With Selenium IDE

Locating by DOM – getElementsByName

```
<html>
<body>
<form id= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



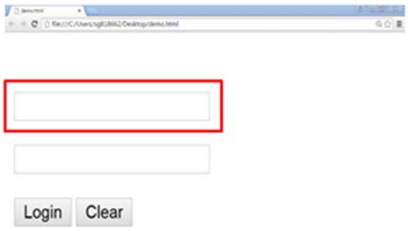

```
document.getElementsByName(" continue ")[1]
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

3.1: Working With Selenium IDE

Locating by DOM – DOM:Name

```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



document.forms[" loginForm "].elements[" username "]

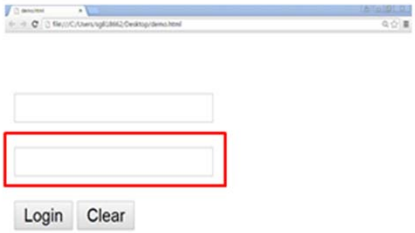
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25

3.1: Working With Selenium IDE

Locating by DOM – DOM:Index

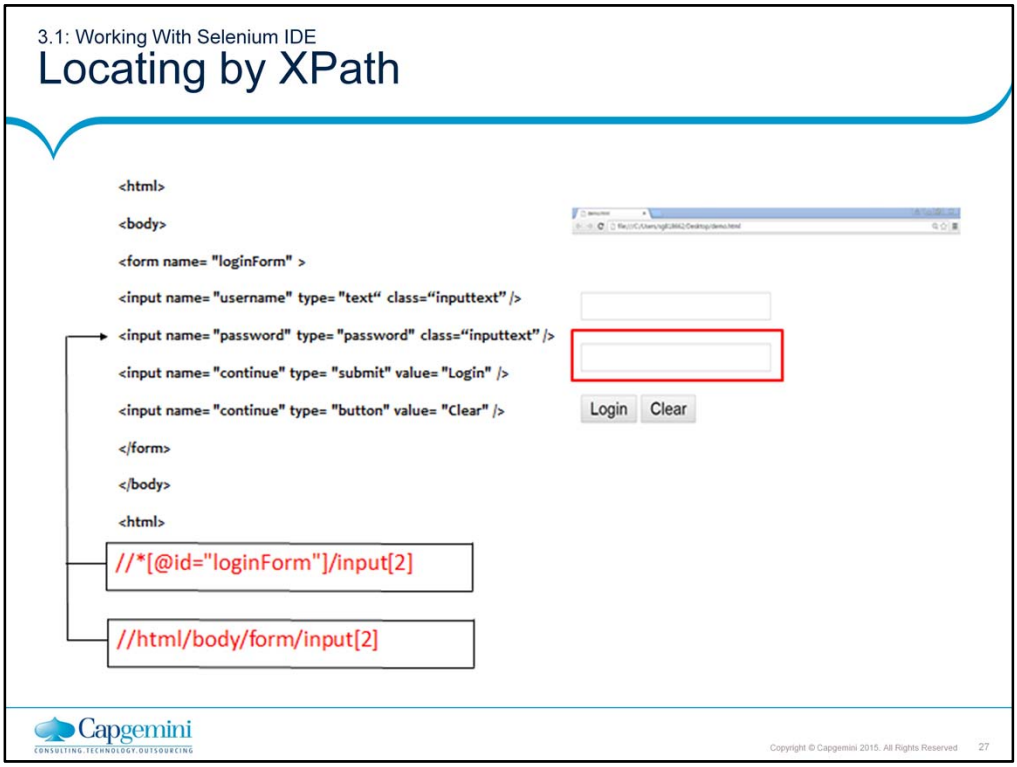
```
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
```



document.forms[1].elements[2]

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26


3.1: Working With Selenium IDE
Locating by XPath
<html>
<body>
<form name= "loginForm" >
<input name= "username" type= "text“ class=“inputtext” />
<input name= "password" type= "password" class=“inputtext” />
<input name= "continue" type= "submit" value= "Login" />
<input name= "continue" type= "button" value= "Clear" />
</form>
</body>
<html>
Login
Clear
//*[@id="loginForm"]/input[2]
//html/body/form/input[2]
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
27

3.1: Working With Selenium IDE

Working With Alerts

- Simplest form of pop-up windows
- Most common Selenium IDE commands used in handling alerts are the following :

assertAlert assertNotAlert	retrieves the message of the alert and asserts it to a string value that you specified
assertAlertPresent assertAlertNotPresent	asserts if an Alert is present or not
storeAlert	retrieves the alert message and stores it in a variable that you will specify
storeAlertPresent	returns TRUE if an alert is present; FALSE if otherwise
verifyAlert verifyNotAlert	retrieves the message of the alert and verifies if it is equal to the string value that you specified
verifyAlertPresent verifyAlertNotPresent	verifies if an Alert is present or not

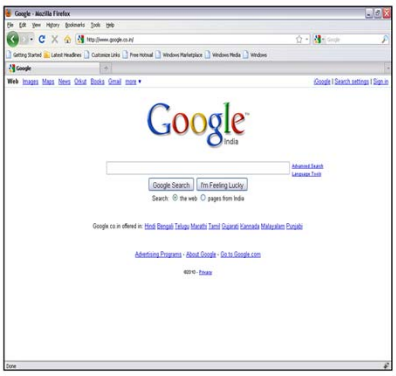
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

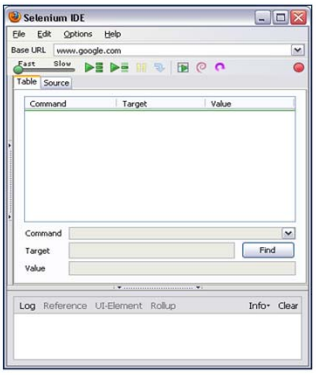
3.1: Working With Selenium IDE

Creating Test Scripts using Selenium IDE

- Go to the Web Page for which you want to carry out the test



- Hit the record button on IDE



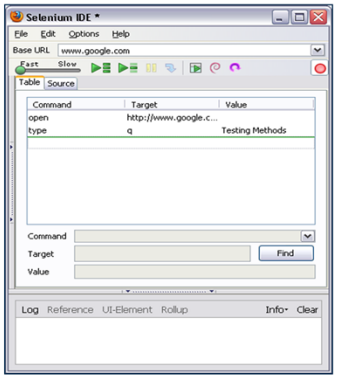
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

3.1: Working With Selenium IDE

Creating Test Scripts using Selenium IDE(Contd.)

- Enter the text on Web Page and submit

- IDE should be updated, stop the recorder and add the assertions

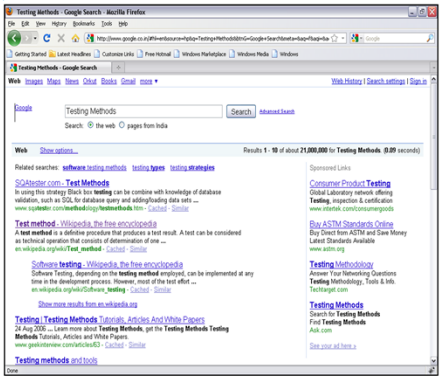


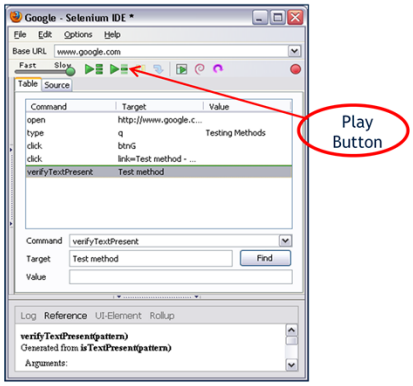
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

3.1: Working With Selenium IDE

Creating Test Scripts using Selenium IDE(Contd.)





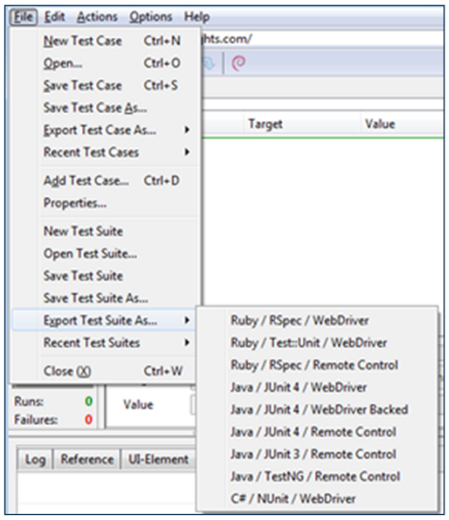
Hit the play button to play the recorded scripts

3.1: Working With Selenium IDE

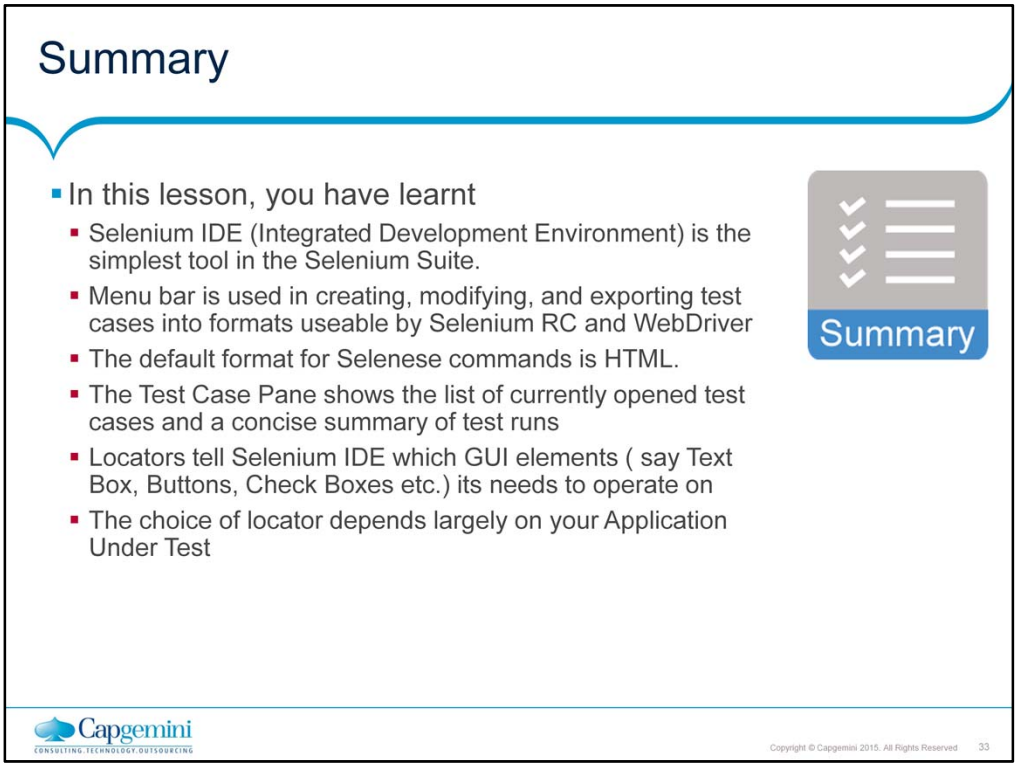
Exporting scripts to multiple languages and Formats

- Test cases can be exported only to the following formats:

- .cs (C# source code)
- .java (Java source code)
- .py (Python source code)
- .rb (Ruby source code)



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING



Add the notes here.

Review Question

- Question 1
  - Select the Locator which is NOT part of Selenium IDE
  - CSS Selector
  - XPath
  - getElementsById
  - getElementsByXpath
- Question 2: True/False
  - The choice of locator depends largely on your Application Under Test.
- Question 3: Fill in the Blanks
  - Error messages and information messages showing the progress are displayed automatically in _____ pane.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 34

Test Automation & Advanced Selenium

Lesson 4: Selenium 2.0 – Web Driver

Lesson Objectives

- Introduction To Web Driver
- Web Driver Vs Selenium RC Vs Selenium IDE
- Benefits of Web Driver over Selenium IDE and RC
- Limitations of Web Driver

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

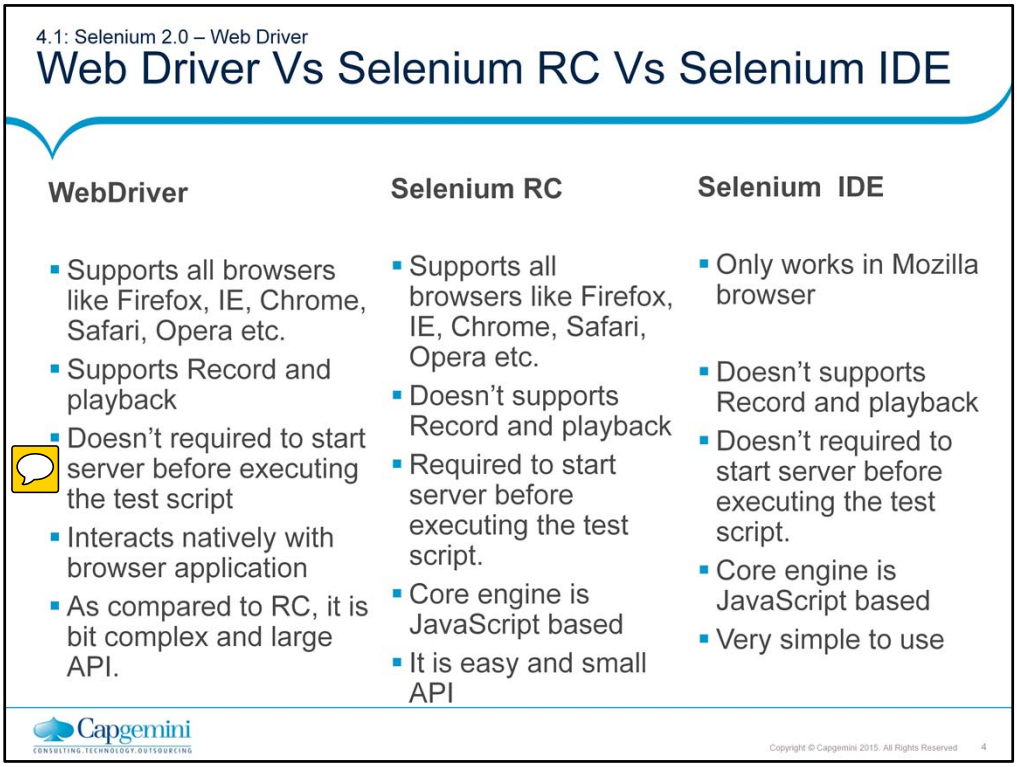
4.1: Selenium 2.0 – Web Driver

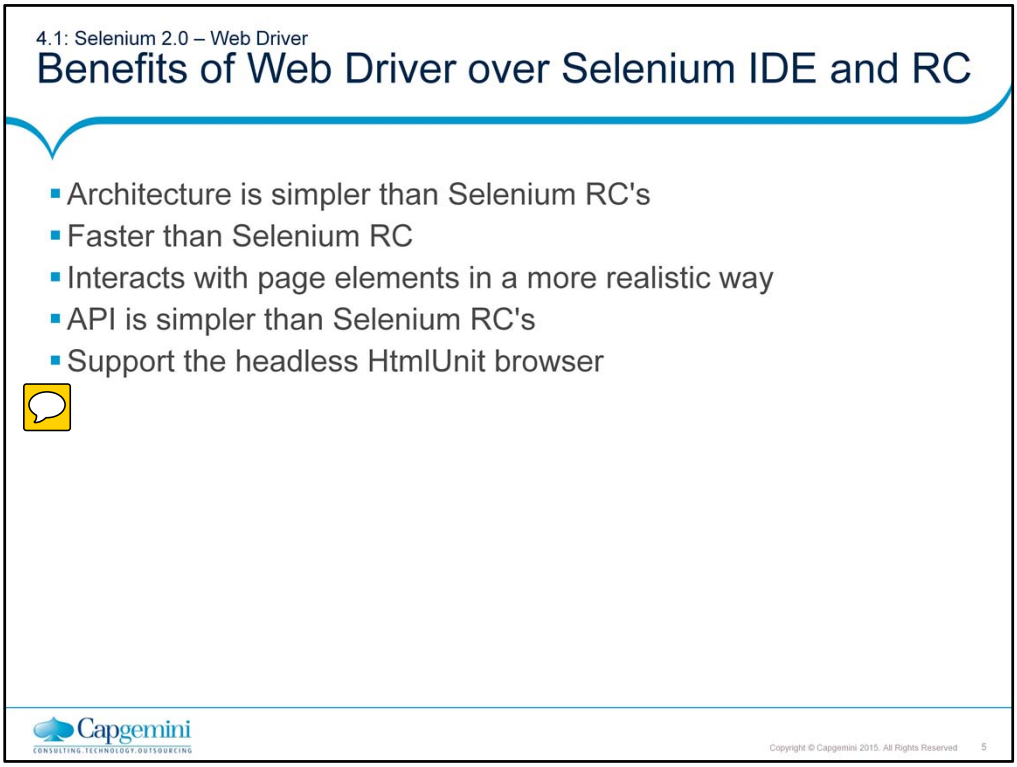
Introduction To Web Driver

- Web automation framework that allows you to execute your tests against different browsers, not just Firefox (unlike Selenium IDE).
- Provides a simpler, more concise programming interface
- Selenium-WebDriver was developed to better support dynamic web pages where elements of a page may change without the page itself being reloaded
- Supply a well-designed object-oriented API that provides improved support for modern advanced web-app testing problems.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3


4.1: Selenium 2.0 – Web Driver
Web Driver Vs Selenium RC Vs Selenium IDE
WebDriver
Supports all browsers like Firefox, IE, Chrome, Safari, Opera etc.
Supports Record and playback
Doesn't required to start server before executing the test script
Interacts natively with browser application
As compared to RC, it is bit complex and large API.
Selenium RC
Supports all browsers like Firefox, IE, Chrome, Safari, Opera etc.
Doesn't supports Record and playback
Required to start server before executing the test script.
Core engine is JavaScript based
It is easy and small API
Selenium IDE
Only works in Mozilla browser
Doesn't supports Record and playback
Doesn't required to start server before executing the test script.
Core engine is JavaScript based
Very simple to use
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
4



1. Architecture
WebDriver's architecture is simpler than Selenium RC's.

It controls the browser from the OS level
All you need are your programming language's IDE (which contains your Selenium commands) and a browser.
You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing
The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser
When you begin testing, Selenium RC Server "injects" a Javascript program called Selenium Core into the browser.
Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.
When the instructions are received, Selenium Core will execute them as Javascript commands.
The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.
The RC Server will receive the response of the browser and then display the results to you.
RC Server will fetch the next instruction from your test script to repeat the whole cycle.

2. Speed
WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

- Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.

3. Real-life Interaction
WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

- Selenium Core, just like other Javascript codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests. Differences in API

4. API
Selenium RC's API is more matured but contains redundancies and often confusing commands.

- For example, most of the time, testers are confused whether to use type or typeKeys; or whether to use click, mouseDown, or mouseDownAt. Worse, different browsers interpret each of these commands in different ways tool
- WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.

5. Browser Support
WebDriver can support the headless HtmlUnit browser

- HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.
- It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.
- Since it is invisible to the user, it can only be controlled through automated means.
- Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.

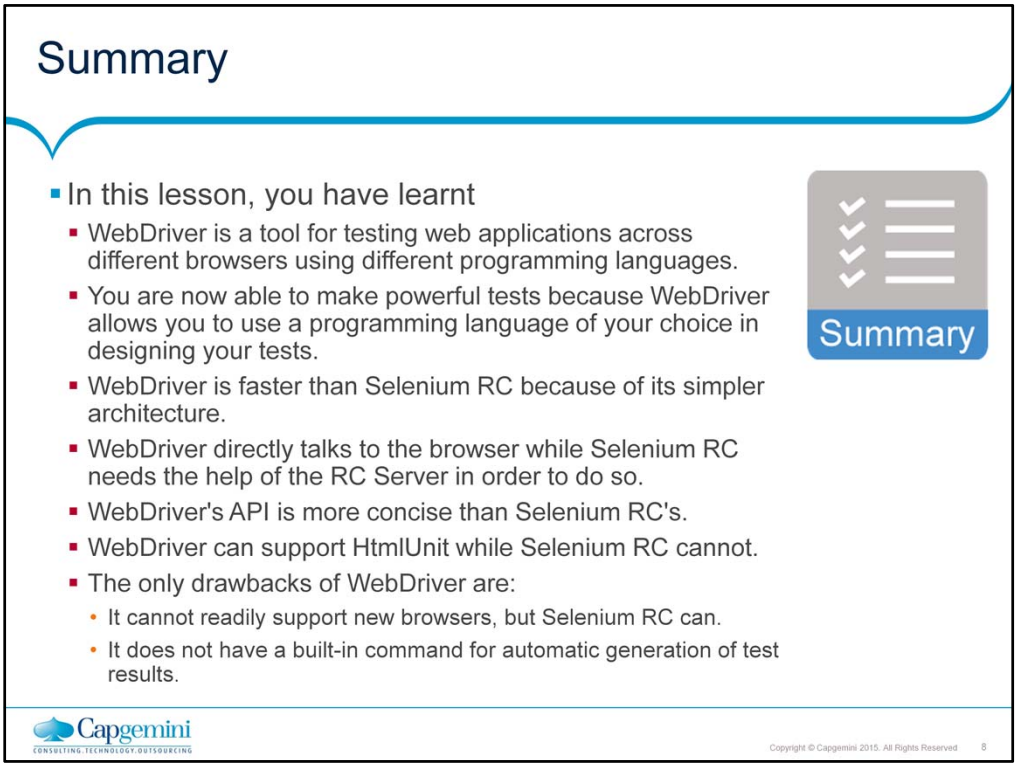
4.1: Selenium 2.0 – Web Driver

Limitations of Web Driver

- Cannot Readily Support New Browsers
  - Since, WebDriver operates on the OS level
  - Different browsers communicate with the OS in different ways.
  - If a new browser comes out, it may have a different process of communicating with the OS as compared to other browsers
- Has no built-in command that automatically generates a Test Results File
  - Have to rely on your IDE's output window, or design the report yourself using the capabilities of your programming language and store it as text, html, etc.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7



Add the notes here.

Review Question

- Question 1
  - Select which is NOT a feature of Web Driver
  - Architecture is simpler
  - Faster
  - API is complex
  - Supports all the browsers

- Question 2: True/False
  - WebDriver can Readily Support New Browsers.
- Question 3: Fill in the Blanks
  - WebDriver can support _____while Selenium RC cannot.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

Test Automation & Advanced Selenium

Lesson 5: Testing Web Applications Using Web Driver API

Lesson Objectives

- Writing first Web Driver Test
- Locating UI Elements-Developers Tools
- Navigation API
  - get
  - Navigate
- Interrogation API
  - getTitle
  - getCurrentUrl
  - getPageSource

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

Lesson Objectives

- WebElement API
  - findElement & findElements
  - By
    - id
    - xpath
    - cssSelector
    - className
    - linkText
    - name
    - tagName
    - partialLinkText

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

Lesson Objectives (Contd.)

- WebElement API
  - click
  - clear
  - sendKeys
  - submit
  - Select – selectByVisibleText etc.
  - getText
  - getAttribute
- Handling Popup Dialogs and Alerts

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4

Lesson Objectives (Contd.)

- Windows
  - getWindowHandle and getWindowHandles
  - switchTo
  - Manage
- Alerts
  - switchTo
  - dismiss
  - Accept
- Why synchronization is important

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

Lesson Objectives (Contd.)

- Using Explicit & Implicit Wait
  - Expected Condition & Expected Conditions
  - WebDriverWait
  - ImplicitlyWait
  - pageLoadTimeout
- JavaScript Executor

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

5.1: Testing Web Applications Using Web Driver API

Writing first Web Driver Test(Java)

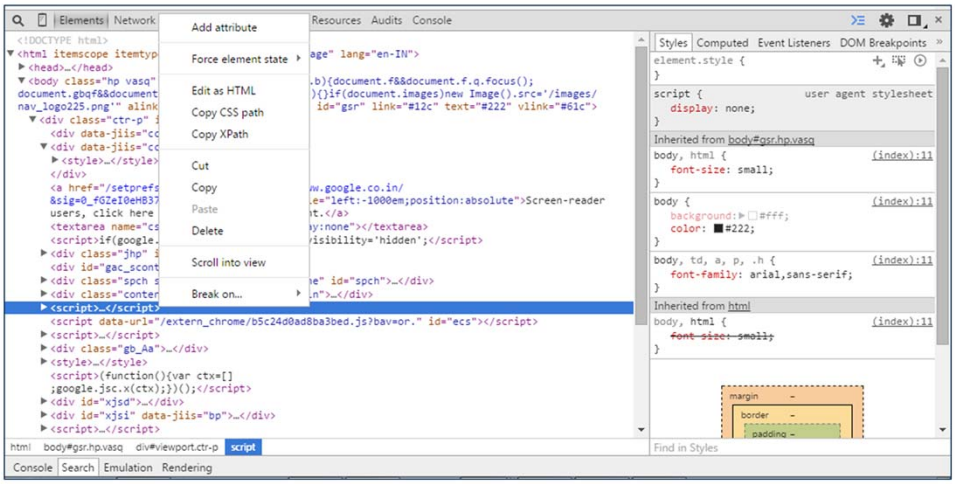
```
package org.openqa.selenium.example;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class Selenium2Example  {
    public static void main(String[] args) {
        // Create a new instance of the Firefox driver
        WebDriver driver = new FirefoxDriver();
        // And now use this to visit Google
        driver.get("http://www.google.com");
        // Find the text input element by its name
        WebElement element = driver.findElement(By.name("q"));
        // Enter something to search for
        element.sendKeys("Cheese!");
        // Now submit the form. WebDriver will find the form for us from the element
        element.submit();
        // Check the title of the page
        System.out.println("Page title is: " + driver.getTitle());
        //Close the browser
        driver.quit();
    }
}
```

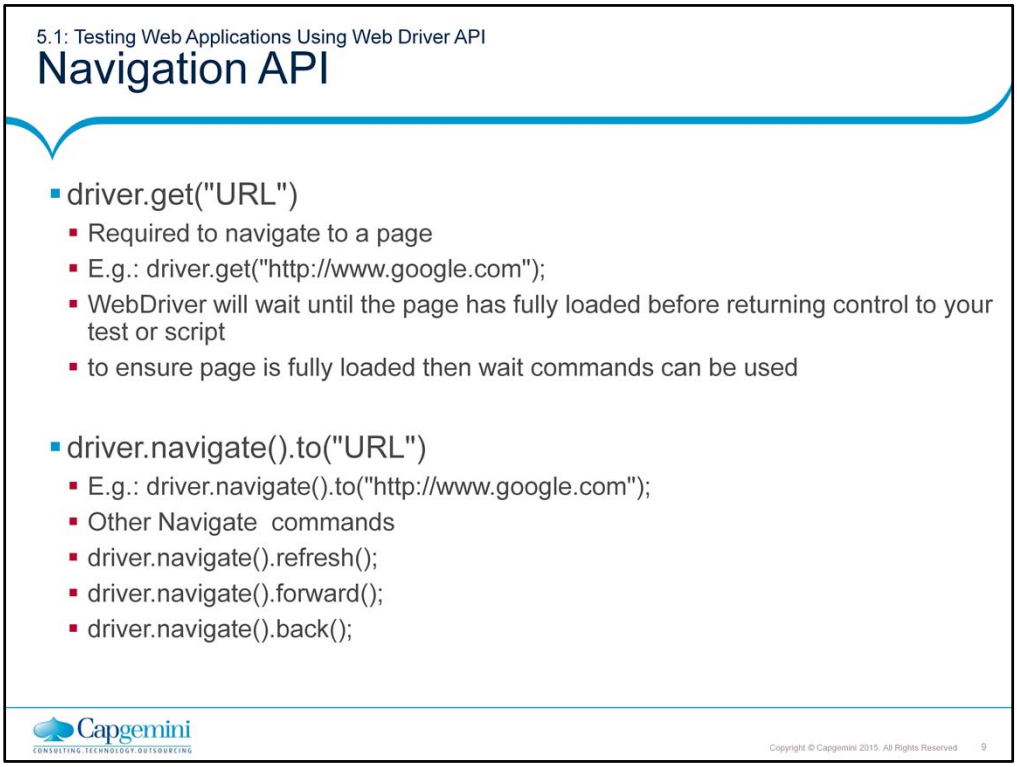
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

5.1: Testing Web Applications Using Web Driver API

Locating UI Elements-Developers Tools





Single-Page Applications are an exception to this.
The difference between these two methods comes not from their behavior, but from the behavior in the way the application works and how browser deal with it.
navigate().to() navigates to the page by changing the URL like doing forward/backward navigation.
Whereas, get() refreshes the page to changing the URL.
So, in cases where application domain changes, both the method behaves similarly. That is, page is refreshed in both the cases. But, in single-page applications, while navigate().to() do not refreshes the page, get() do.
Moreover, this is the reason browser history is getting lost when get() is used due to application being refreshed.

5.1: Testing Web Applications Using Web Driver API

Interrogation API

- driver.getTitle()
  - Get the title of the current page
- driver. getCurrentUrl()
  - Get the current URL of the browser
- driver.getPageSource()
  - Get the source code of the page

- Syntax:

```
public void testTitleReliability() {
driver.get("https://www.google.com");
        boolean title = driver.getTitle().contains("Google");
        if(title)
                String currentURL = driver.getCurrentUrl();
                (If you want to verify a particular text is present or not on the page,do as below)
                boolean b = driver.getPageSource().contains("your text");
                System.out.println("Expected title is present ");
else if(!title)
System.out.println(" Expected title is not present"); ");
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

5.1: Testing Web Applications Using Web Driver API

WebElement API

- findElement:
  - Used to locate single element and return WebElement object of first occurrences element on web page
  - If element not found, throw s exception NoSuchElementException
  - Syntax: findElement(By by)
- Example:

```
WebElement element = driver.findElement(By.id("Home"));
element.click();
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

5.1: Testing Web Applications Using Web Driver API

WebElement API

- findElements:
  - Used to find multiple element on webpage, e.g.: count total number of row in table
  - Returns List of WebElement object of all occurrences of element
  - If element not found, returns empty List of WebElement object
  - Syntax: List element = findElements(By by)
- Example:

```
List {WebElement} element = driver.findElement(By.xpath("//table/tr"));
```

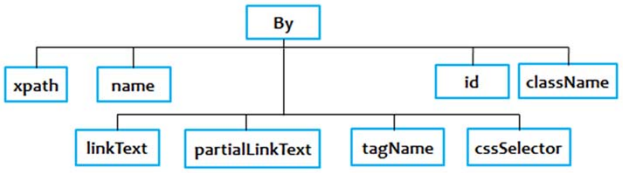
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

5.1: Testing Web Applications Using Web Driver API

WebElement API (Cont..)

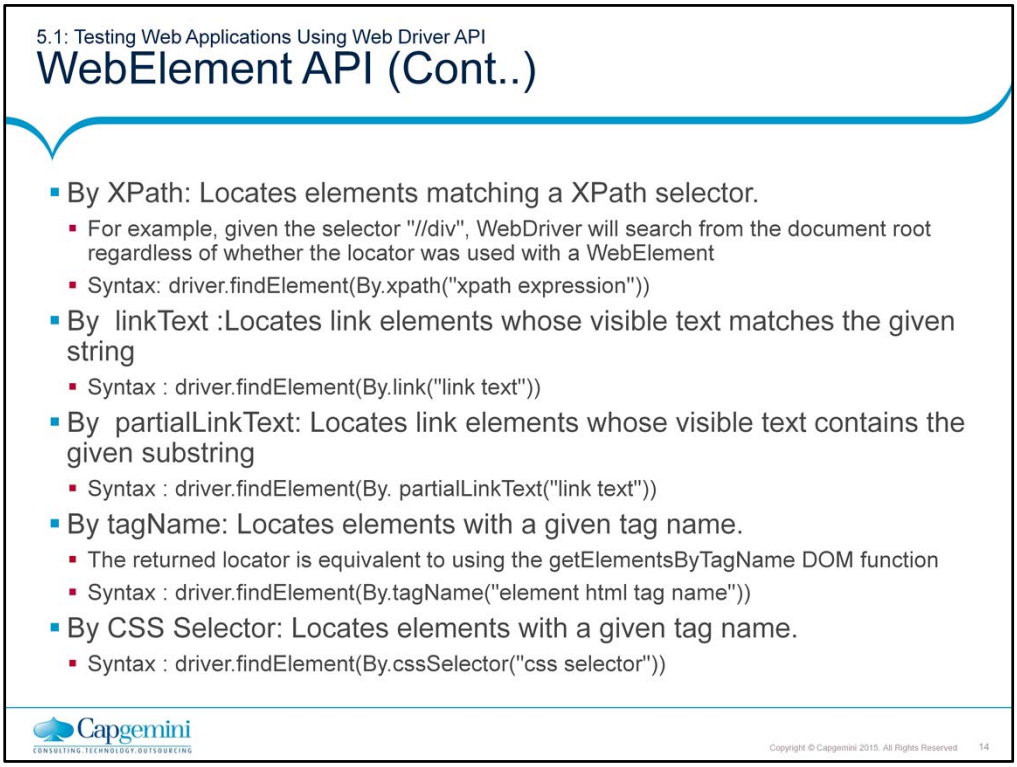
- By:
  - A collection of factory functions for creating webdriver.Locator instances



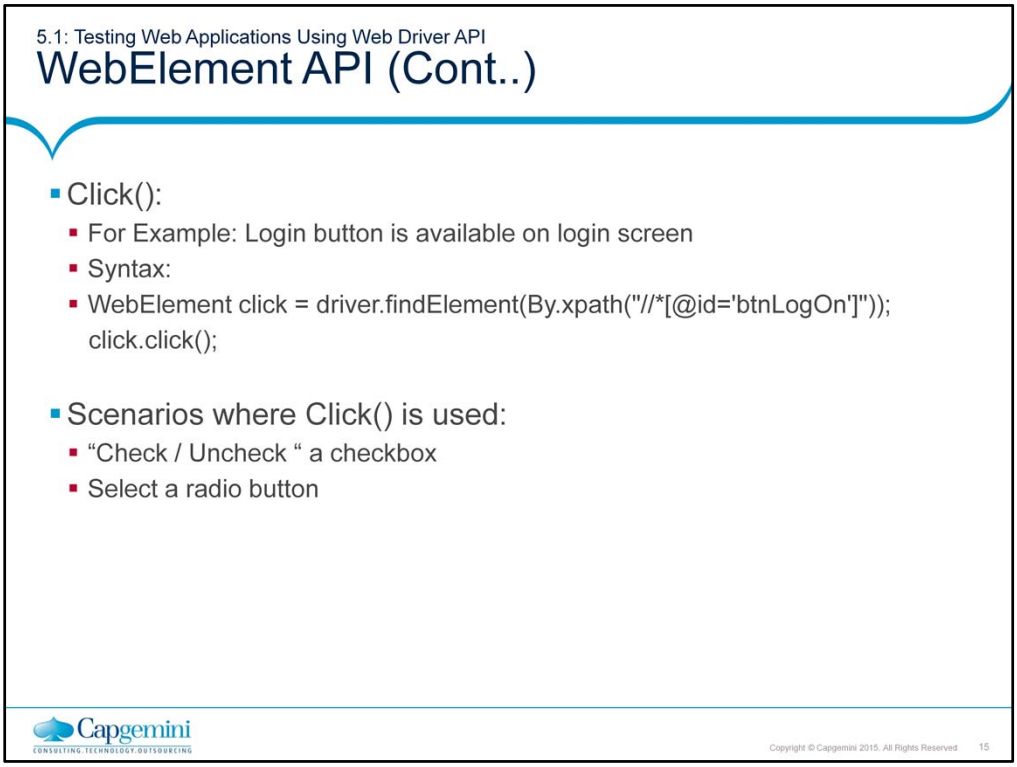
- By id: Locates an element by its ID
  - Syntax: driver.findElement(By.id("element id"))
- By className: Locates elements that have a specific class name
  - Syntax : driver.findElement(By.className("element class"))
- By name: Locates elements whose name attribute has the given value
  - Syntax : driver.findElement(By.name("element name"))

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13



CSS selectors for Selenium with example
When we don't have an option to choose Id or Name, we should prefer using CSS locators as the best alternative.
CSS is "Cascading Style Sheets" and it is defined to display HTML in structured and colorful styles are applied to webpage.
Selectors are patterns that match against elements in a tree, and as such form one of several technologies that can be used to select nodes in an XML document. Visit to know more W3.Org Css selectors
CSS has more Advantage than Xpath
CSS is much more faster and simpler than the Xpath.
In IE Xpath works very slow, where as Css works faster when compared to Xpath.

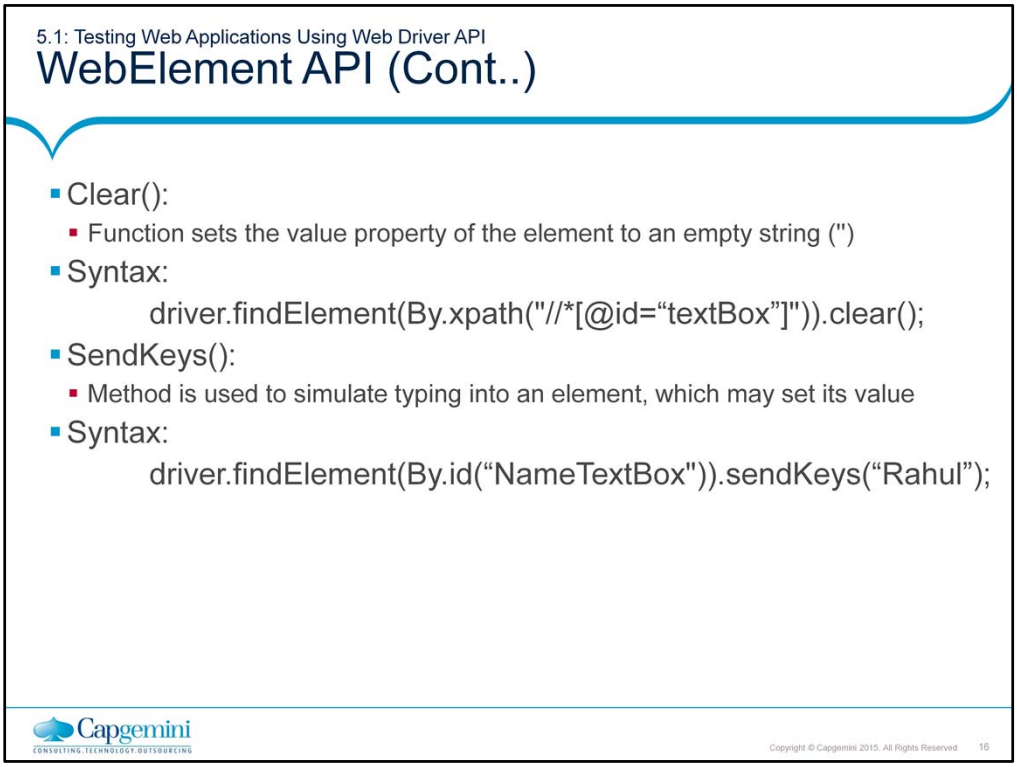


Click():
Click a link / button:
To click on an object through webdriver first we need to find out which locator we are going to use. Is it ID, name, xpath, or css? For this purpose we can utilize firebug / xpath checker to find out is there any id / name exists for the object we are going to perform action upon. Then write the code as below:
driver.findElement(By.xpath("//a[@href='CustomerInfo.htm']")).click();
In the above line of code "driver" could be FirefoxDriver, InternetExplorerDriver, ChromeDriver, HtmlUnitDriver, etc. On one of these browsers we are going to find an element and then click as per the code.
findElement is an API provided by the webdriver which requires argument "By.xpath". The "xpath" can be replaced by one of the below methods if we need to identify the element with any other attributes such as css, name, classname, etc.

"Check / Uncheck " a checkbox
To "Check / Uncheck" a checkbox, the object needs to be identified using findElement method and then just click. To find out whether the checkbox is checked or not utilize the method – element.isSelected()

```
WebElement kancheck = driver.findElement(By.name("kannada"));
kancheck.click();
System.out.println(kancheck.isSelected());
```

Above code snippet will first click the checkbox named kannada and then verifies whether it is clicked or not.



Click():
Click a link / button:
To click on an object through webdriver first we need to find out which locator we are going to use. Is it ID, name, xpath, or css? For this purpose we can utilize firebug / xpath checker to find out is there any id / name exists for the object we are going to perform action upon. Then write the code as below:
driver.findElement(By.xpath("//a[@href='CustomerInfo.htm']")).click();
In the above line of code "driver" could be FirefoxDriver, InternetExplorerDriver, ChromeDriver, HtmlUnitDriver, etc. On one of these browsers we are going to find an element and then click as per the code.
findElement is an API provided by the webdriver which requires argument "By.xpath". The "xpath" can be replaced by one of the below methods if we need to identify the element with any other attributes such as css, name, classname, etc.

"Check / Uncheck " a checkbox
To "Check / Uncheck" a checkbox, the object needs to be identified using findElement method and then just click. To find out whether the checkbox is checked or not utilize the method – element.isSelected()

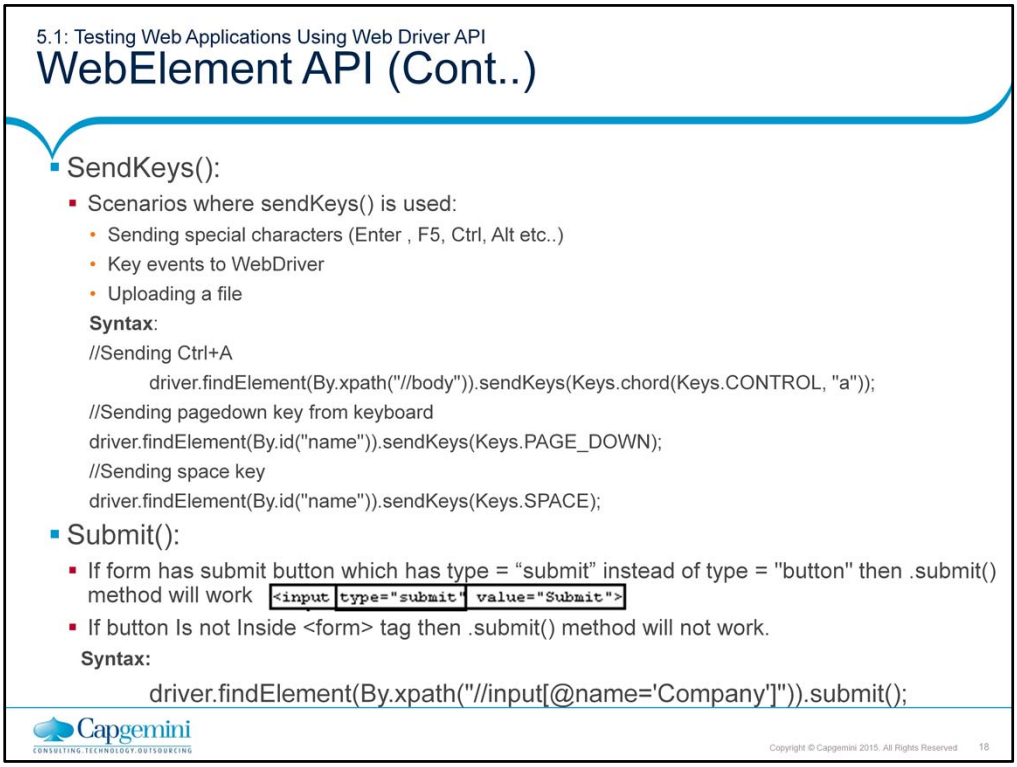
```
WebElement kancheck = driver.findElement(By.name("kannada"));
kancheck.click();
System.out.println(kancheck.isSelected());
```

Above code snippet will first click the checkbox named kannada and then verifies whether it is clicked or not.

Select a radio button
Follow the same steps which are used in Checkbox to select a radio button and then verify the status using isSelected() method.

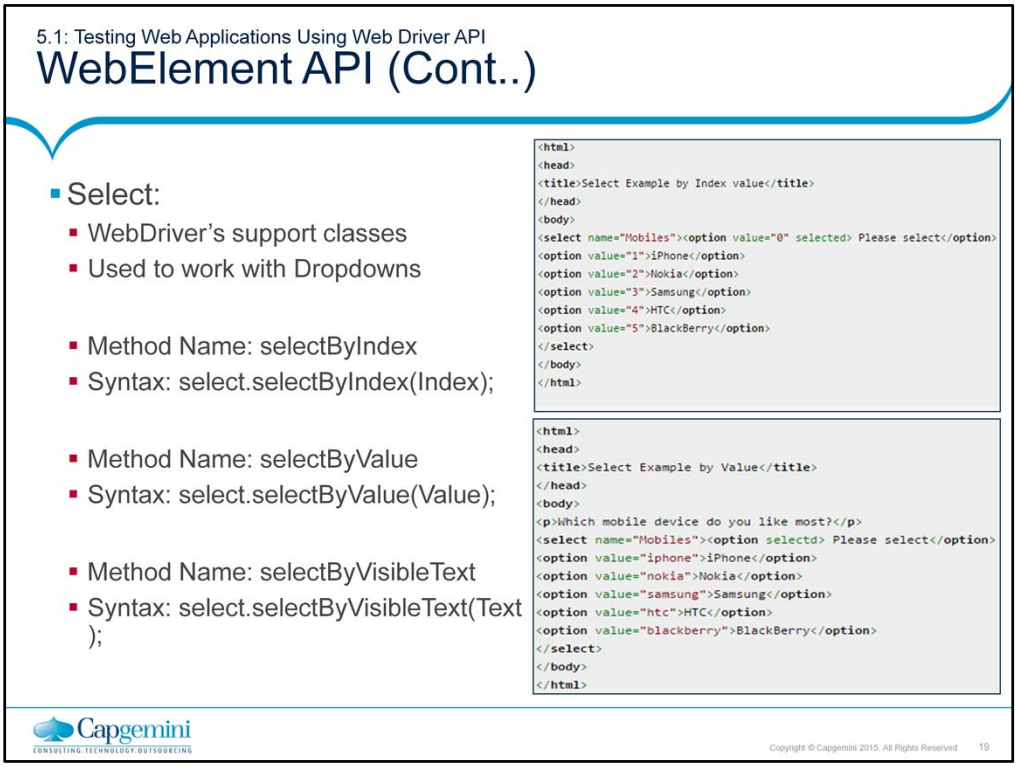
```
WebElement gender =
driver.findElement(By.xpath("//input[@name='male']"));
gender.click();
System.out.println(gender.isSelected());
```

Clear();
The clear() method executes an "Automation Atom", which is a JavaScript function intended to provide the smallest basic unit of automation functionality for a browser. In the case of clear(), that function sets the value property of the element to an empty string (''), then fires the onchange event on the element. The atoms function you're interested in is bot.action.clear()
click
clear
sendKeys
submit
Select – selectByVisibleText etc.
getText
getAttribute



When to use .click() method
You can use .click() method to click on any button of software web application. Means element's type = "button" or type = "submit", .click() method will works for both.

When to use .submit() method
If you will look at firebug view for any form's submit button then always It's type will be "submit". In this case, .submit() method Is very good alternative of .click() method.



Method Name: selectByIndex
Purpose: To Select the option based on the index given by the user.
There is an attribute called "values" which will have the index values.
Example:

```
WebElement element=driver.findElement(By.name("Mobiles"));
Select se=new Select(element);
se.selectByIndex(1);
```

Method Name: selectByValue
Purpose: To Select the options that have a value matching with the given argument by the user.
Example:

```
WebElement element=driver.findElement(By.name("Mobiles"));
Select se=new Select(element);
se.selectByValue("nokia");
```

Method Name: selectByVisibleText
Purpose: To Select all options that display text matching the given argument. It will not look for any index or value, it will try to match the VisibleText (which will display in dropdown)
Example:

```
WebElement element=driver.findElement(By.name("Mobiles"));
Select se=new Select(element);
se.selectByVisibleText("HTC");
```

Method Name: deselectByIndex
Purpose: To Deselect the option at the given index. The user has to provide the value of index.
Example:

```
se.deselectByIndex(1);
```

Method Name: deselectByValue
Purpose: To Deselect all options that have a value matching the given argument.
Example:

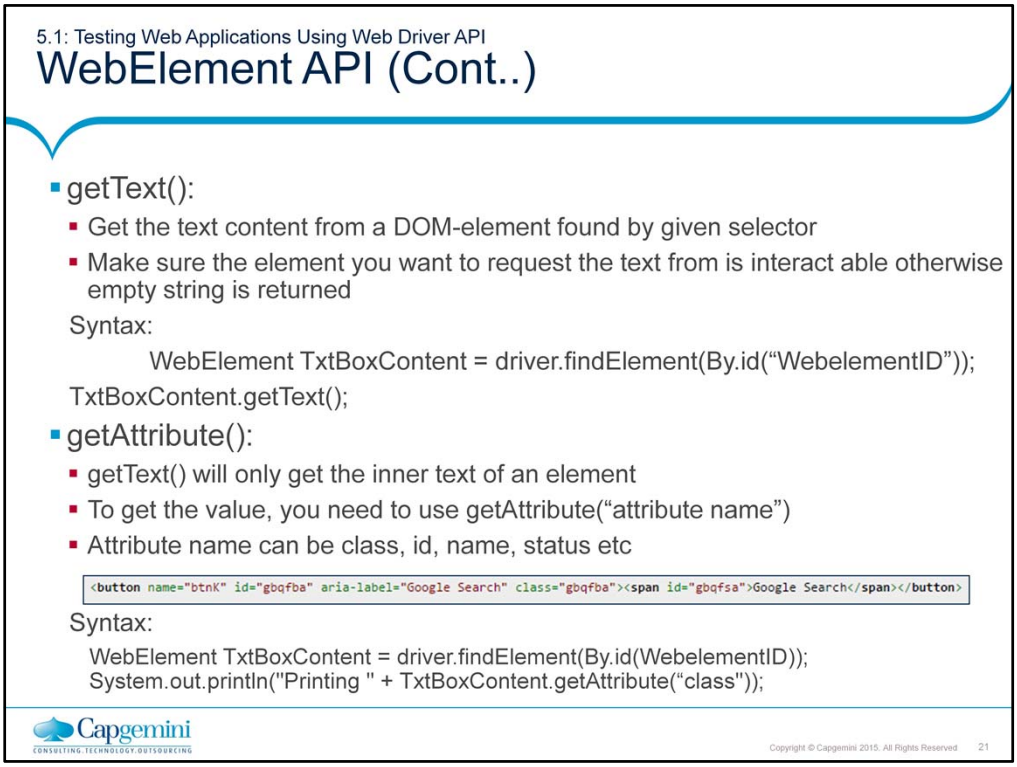
```
se.deselectByValue("nokia");
```

Method Name: deselectByVisibleText
Purpose: To Deselect all options that display text matching the given argument.
Example:

```
se.deselectByVisibleText("HTC");
```

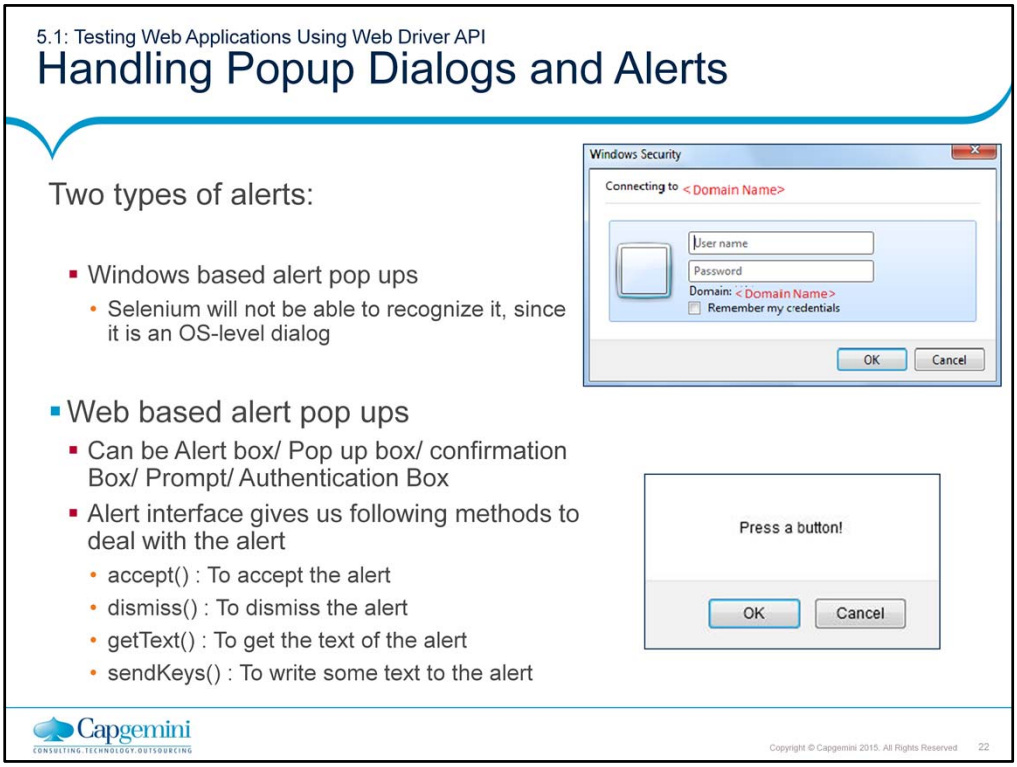
Method Name: deselectAll
Purpose: To Clear all selected entries. This works only when the SELECT supports multiple selections. It throws NotImplemented eError if the "SELECT" does not support multiple selections. In select it mandatory to have an attribute multiple="multiple“
Example:

```
se.deselectAll();
```



When to use .click() method
You can use .click() method to click on any button of software web application. Means element's type = "button" or type = "submit", .click() method will works for both.

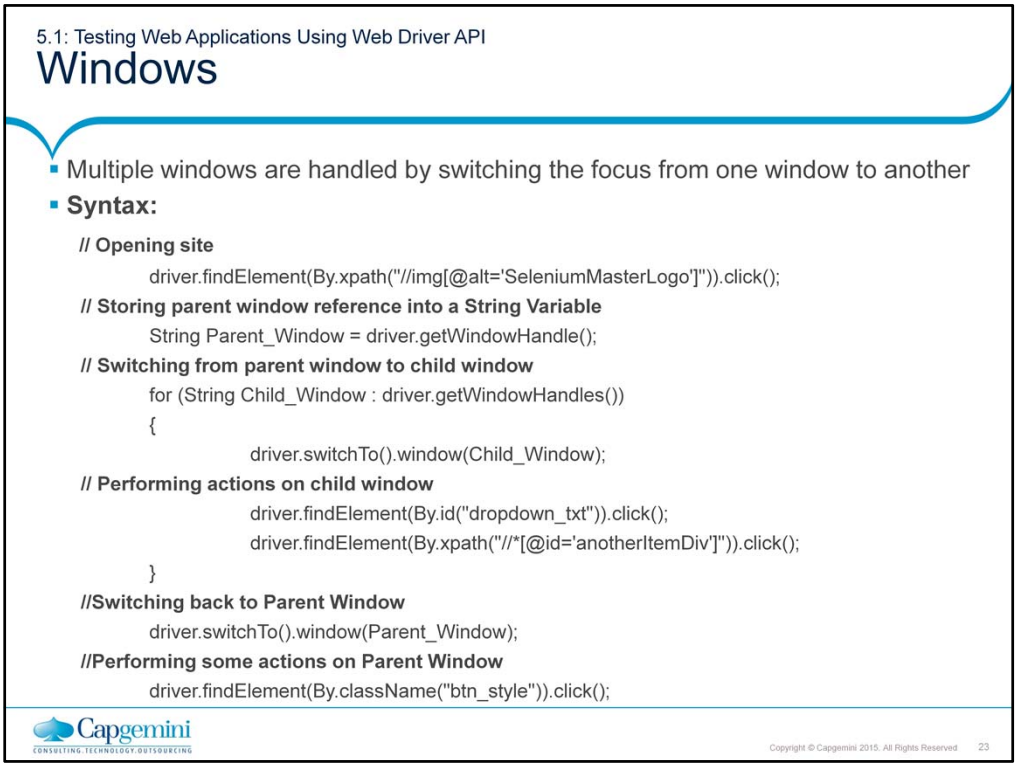
When to use .submit() method
If you will look at firebug view for any form's submit button then always It's type will be "submit". In this case, .submit() method Is very good alternative of .click() method.



Simple alert
Simple alerts just have a OK button on them. They are mainly used to display some information to the user.

Confirmation Alert
This alert comes with an option to accept or dismiss the alert. To accept the alert you can use Alert.accept() and to dismiss you can use the Alert.dismiss().

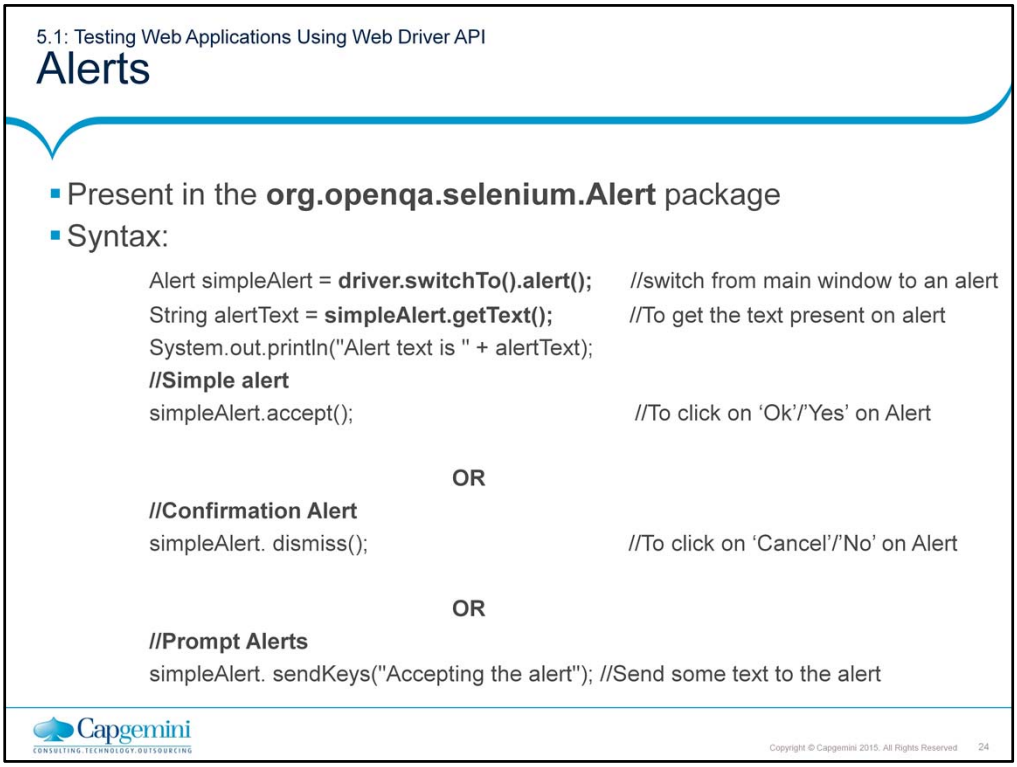
Prompt Alerts
In prompt alerts you get an option to add text to the alert box. This is specifically used when some input is required from the user. We will use
the sendKeys() method to type something in the Prompt alert box.



Steps for understanding window handling:

Window A has a link "Link1" and we need to click on the link (click event).
Window B displays and we perform some actions.

The entire process can be fundamentally segregated into following steps:
Step 1 : Clicking on Link1 on Window A
A new Window B is opened.
Step 2 : Save reference for Window A
Step 3 : Create reference for Window B
Step 4 : Move Focus from Window A to Window B
Window B is active now
Step 5 : Perform Actions on Window B
Complete the entire set of Actions
Step 6 : Move Focus from Window B to Window A
Window A is active now



Alerts are different from regular windows. The main difference is that alerts are blocking in nature. They will not allow any action on the underlying webpage if they are present. So if an alert is present on the webpage and you try to access any of the element in the underlying page you will get following exception: UnhandledAlertException: Modal dialog present

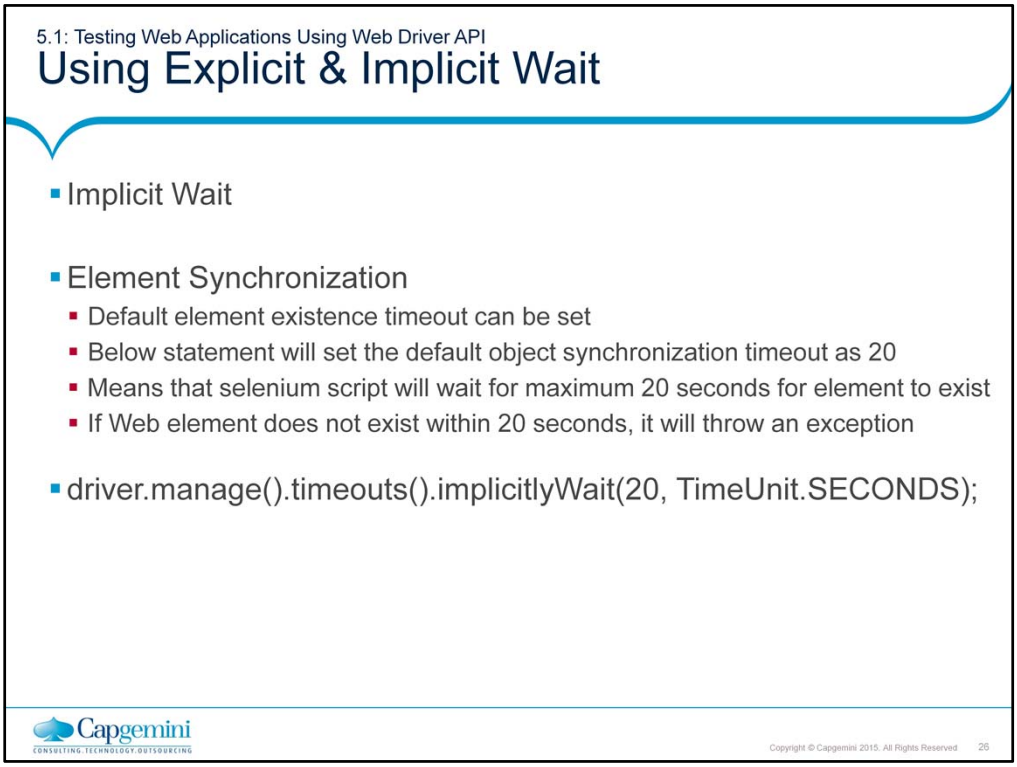
5.1: Testing Web Applications Using Web Driver API

Why synchronization is important

- "Mechanism which involves more than one components to work parallel with each other"
- Every time user performs an operation on the browser, one of the following happens:
  - The request goes all the way to server and entire DOM is refreshed when response comes back
  - The request hits the server and only partial DOM gets refreshed (Ajax requests or asynchronous JavaScript calls)
  - The request is processed on the client side itself by JavaScript functions
- So if we think about the overall workflow, there is a need of certain synchronization that happens between the client(aka. browser) and the server (the url)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 25



Synchronization can be classified into two categories:

1. Unconditional
2. Conditional Synchronization

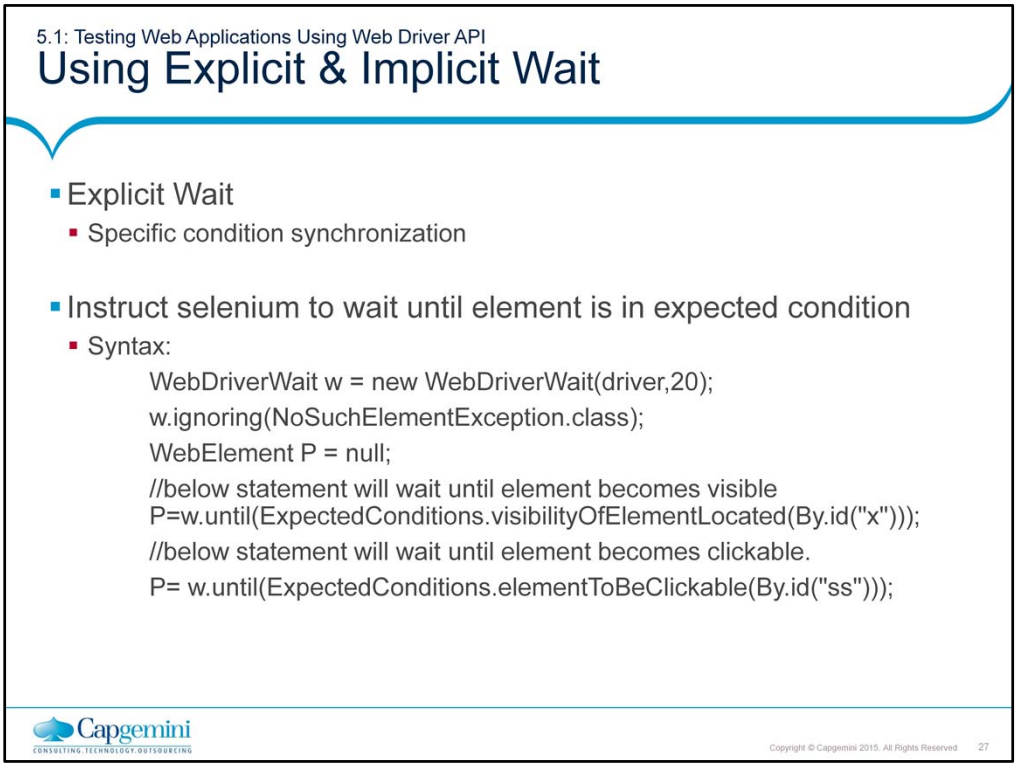
Unconditional :
In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.
Examples: Wait() and Thread.Sleep();

The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.

The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

Conditional Synchronization:
We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.
It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.

In Selenium we have implicit Wait and Explicit Wait conditional statements



Page Load Synchronization:

Default page navigation timeout can be set.
Below statement will set the navigation timeout as 50
Means that selenium script will wait for maximum 50 seconds for page to load
If page does not load within 50 seconds, it will throw an exception
Syntax
driver.manage().timeouts().pageLoadTimeout(50,TimeUnit.SECONDS);

5.1: Testing Web Applications Using Web Driver API

JavaScript Executor

- An interface which provides mechanism to execute Javascript through selenium driver
- Provides “executescript” & "executeAsyncScript" methods, to run JavaScript in the context of the currently selected frame or window
- Used to enhance the capabilities of the existing scripts by performing Javascript injection into our application under test
- Package
  import org.openqa.selenium.JavascriptExecutor;

  Syntax

  ```
  JavascriptExecutor js = (JavascriptExecutor) driver;
  js.executeScript(Script,Arguments);
  ```

script - The JavaScript to execute
Arguments - The arguments to the script.(Optional)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

5.1: Testing Web Applications Using Web Driver API

JavaScript Executor(Scenarios)

- How to generate Alert Pop window in selenium?
- Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver
Js.executeScript("alert('hello world');");
```

- How to click a button in Selenium WebDriver using JavaScript?
  Code: JavascriptExecutor js = (JavascriptExecutor)driver;

```
js.executeScript("arguments[0].click();", element);
```

- How to refresh browser window using Javascript ?
- Code:

```
JavascriptExecutor js = (JavascriptExecutor)driver;
driver.executeScript("history.go(0)");
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

5.1: Testing Web Applications Using Web Driver API

JavaScript Executor(Scenarios)

- How to get innertext of the entire webpage in Selenium?
  Code:
  JavascriptExecutor js = (JavascriptExecutor)driver;
  string sText = js.executeScript("return document.documentElement.innerText;").toString();

- How to get the Title of our webpage ?
  Code:
  JavascriptExecutor js = (JavascriptExecutor)driver;
  string sText = js.executeScript("return document.title;").toString();

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30

5.1: Testing Web Applications Using Web Driver API

JavaScript Executor(Scenarios)

- How to perform Scroll on application using Selenium?
  Code:
  JavascriptExecutor js = (JavascriptExecutor)driver; //Vertical scroll - down by 50 pixels js.executeScript("window.scrollBy(0,50)");

- Note: for scrolling till the bottom of the page we can use the code:

  js.executeScript("window.scrollBy(0,document.body.scrollHeight)");

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 31

5.1: Testing Web Applications Using Web Driver API

JavaScript Executor(Scenarios)

- How to click on a Sub Menu which is only visible on mouse hover on Menu?
  Code:

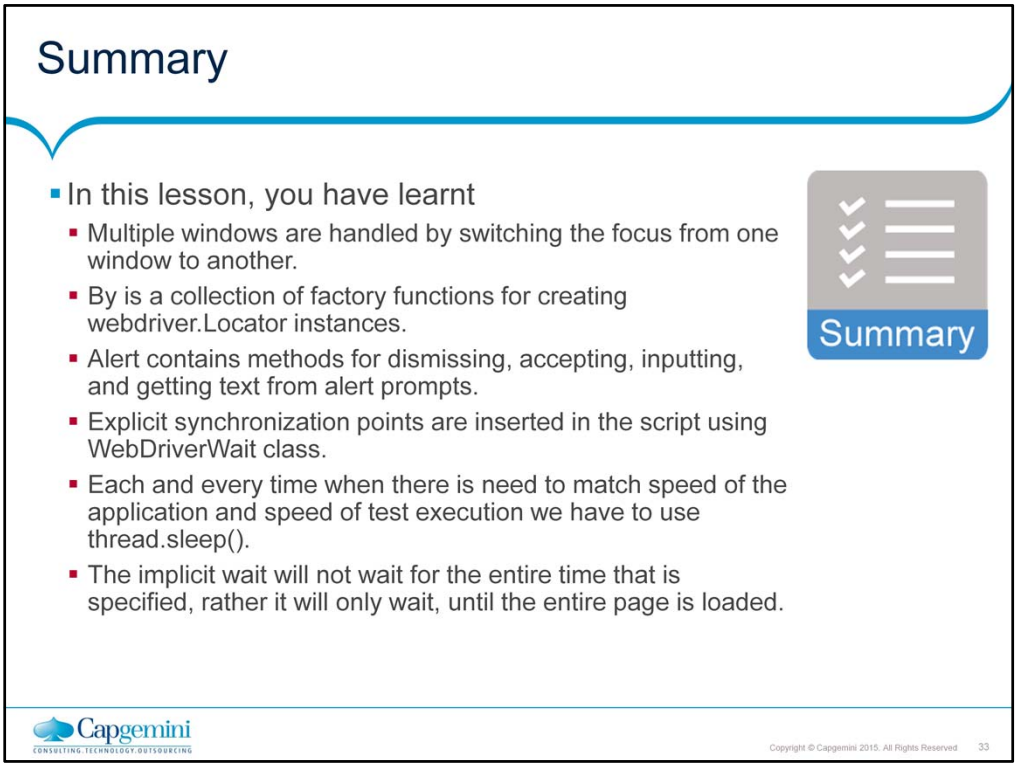
```
JavascriptExecutor js = (JavascriptExecutor)driver;
  //Hover on Automation Menu on the Menu Bar
   js.executeScript("$('ul.menus.menu-secondary.sf-js-enabled.sub-menu li').hover()");
```

- How to navigate to different page using Javascript?
  Code:

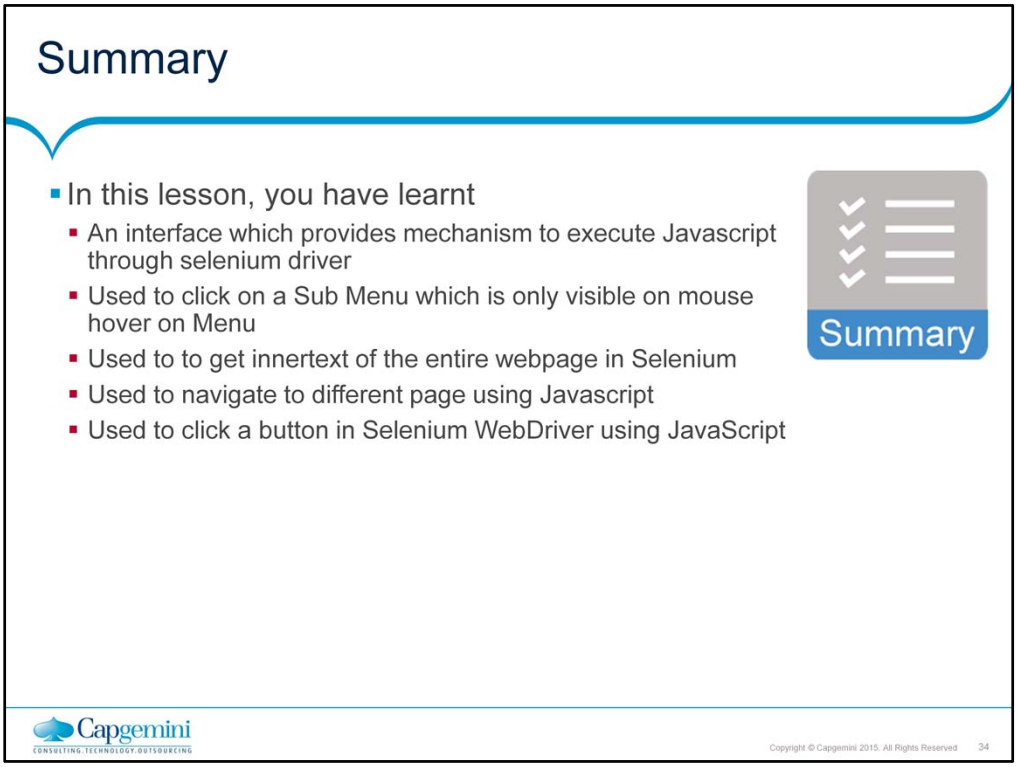
```
JavascriptExecutor js = (JavascriptExecutor)driver;   //Navigate to new Page js.executeScript("window.location = 'https://www.facebook.com/uftHelp'");
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 32



Add the notes here.



Add the notes here.

Review Question

- Question 1
  - Select which is NOT an Explicit Wait
    - VisibilityOfElementLocated
    - ElementToBeClickable
    - PageLoadTimeout
    - None of the above

- Question 2: True/False
  - The syntax is correct:
  - Syntax : driver.findElement(By. PartialLinkText("link text"));
- Question 3: Fill in the Blanks
  - findElements is used to find _____ element on webpage

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 35

Review Question

- Question 4: True/False
  - The syntax is correct:

  Syntax :

  JavascriptExecutor js = (JavascriptExecutor)driver;

- Question 5: Fill in the Blanks
  - An interface which provides mechanism to execute _____ through selenium driver

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 36

Answer 1:
True

Answer 2:
Javascript

Test Automation & Advanced Selenium

Lesson 6: Web Driver Test with Xunit

Lesson Objectives

- Introduction to Xunit and Junit
- Junit Annotations
- Assertions/Verifications with Junit or TestNG
- Web Driver Test cases with Junit or TestNG
- Test Suite

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

6.1: Selenium 2.0 – Web Driver Test with Xunit

Web Driver Test with Xunit

- Is something not missing? We do Testing, where are Test Cases , Test Suite, Verifications, Results , Reports….
- Because, Selenium web driver is automation API not testing API
- So, combine Selenium automation with testing frameworks, like Xunit

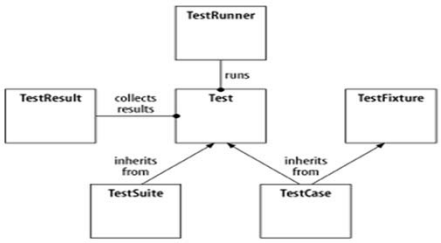
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

6.1: Selenium 2.0 – Web Driver Test with Xunit

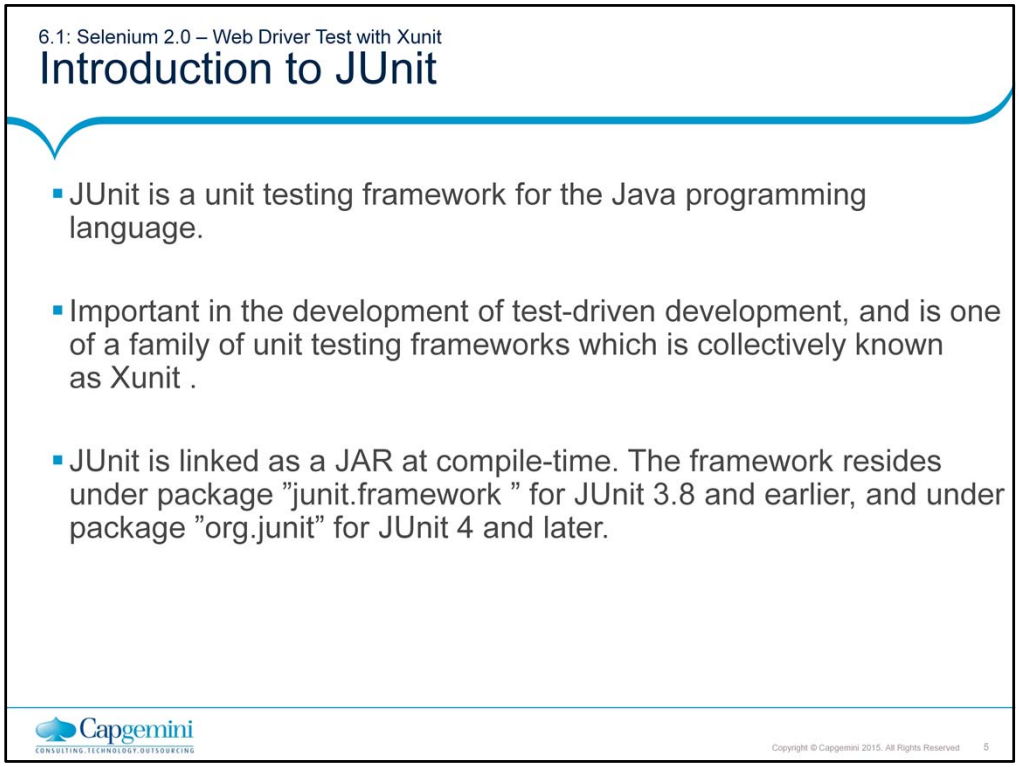
Introduction To XUNIT

- Xunit is the collective name for several unit testing frameworks that derive their structure and functionality from Smalltalk's SUnit.
- The names of many of these frameworks are a variation on "SUnit", usually substituting the "S" for the first letter (or letters) in the name of their intended language ("JUnit" for Java, "RUnit" for R etc.).
- These frameworks and their common architecture are collectively known as "Xunit".
- All Xunit frameworks share the following basic component architecture:



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 4



A research survey performed in 2013 across 10,000 Java projects hosted on GitHub found that JUnit, (in a tie with slf4j-api), was the most commonly included external library. Each library was used by 30.7% of projects.

6.1: Selenium 2.0 – Web Driver Test with Xunit

Junit – Annotations

@Test:

- The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method. Any exceptions thrown by the test will be reported by JUnit as a failure. If no exceptions are thrown, the test is assumed to have succeeded.

```
1  public class MyTestClass {
2      @Test
3      public void myTestMethod() {
4          /**
5           * Use Assert methods to call your methods to be tested.
6           * A simple test to check whether the given list is empty or not.
7           */
8          org.junit.Assert.assertTrue( new ArrayList().isEmpty() );
9      }
10 }
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 6

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@Before:

- When writing tests, it is common to find that several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before the Test method. The @Before methods of super classes will be run before those of the current class.

```
1  public class MyTestClass {
2      List<String> testList;
3      @Before
4      public void initialize() {
5          testList = new ArrayList<String>();
6      }
7      @Test
8      public void myTestMethod() {
9          /**
10          * Use Assert methods to call your methods to be tested.
11          * A simple test to check whether the given list is empty or not.
12          */
13         org.junit.Assert.assertTrue( testList.isEmpty() );
14     }
15 }
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 7

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@After:

- If you allocate external resources in a Before method you need to release them after the test runs. Annotating a public void method with @After causes that method to be run after the Test method. All @After methods are guaranteed to run even if a Before or Test method throws an exception. The @After methods declared in super classes will be run after those of the current class.

```
1  public class MyTestClass {
2      OutputStream stream;
3      @Before
4      public void initialize() {
5          /**
6           * Open OutputStream, and use this stream for tests.
7           */
8          stream = new FileOutputStream(...);
9      }
10     @Test
11     public void myTestMethod() {
12         /**
13          * Now use OutputStream object to perform tests
14          */
15         ...
16         ...
17     }
18     @After
19     public void closeOutputStream() {
20         /**
21          * Close output stream here
22          */
23         try{
24           if(stream != null) stream.close();
25         } catch(Exception ex){
26         }
27     }
28 }
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 8

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@BeforeClass:

- Annotating a public static void no-arg method with @BeforeClass causes it to be run once before any of the test methods in the class. The @BeforeClass methods of superclasses will be run before those the current class.
- The annotations @BeforeClass and @Before are same in functionality. The only difference is the method annotated with @BeforeClass will be called once per test class based, and the method annotated with @Before will be called once per test based.

```
1  public class MyTestClass {
2      @BeforeClass
3      public void initGlobalResources() {
4          /**
5           * This method will be called only once per test class.
6           */
7      }
8      @Before
9      public void initializeResources() {
10         /**
11          * This method will be called before calling every test.
12          */
13     }
14     @Test
15     public void myTestMethod1() {
16         /**
17          * initializeResources() method will be called before calling this method
18          */
19     }
20     @Test
21     public void myTestMethod2() {
22         /**
23          * initializeResources() method will be called before calling this method
24          */
25     }
26 }
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@AfterClass:

- If you allocate expensive external resources in a BeforeClass method you need to release them after all the tests in the class have run. Annotating a public static void method with @AfterClass causes that method to be run after all the tests in the class have been run. All @AfterClass methods are guaranteed to run even if a BeforeClass method throws an exception. The @AfterClass methods declared in superclasses will be run after those of the current class.

```
public class MyTestClass {
    @BeforeClass
    public void initGlobalResources() {
        /**
         * This method will be called only once per test class. It will be called
         * before executing test.
         */
    }
    @Test
    public void myTestMethod1() {
        // write your test code here...
        ...
        ...
    }
    @AfterClass
    public void closeGlobalResources() {
        /**
         * This method will be called only once per test class. It will be called
         * after executing test.
         */
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@Ignore:

- Sometimes you want to temporarily disable a test or a group of tests. Methods annotated with Test that are also annotated with @Ignore will not be executed as tests. Also, you can annotate a class containing test methods with @Ignore and none of the containing tests will be executed. Native JUnit 4 test runners should report the number of ignored tests along with the number of tests that ran and the number of tests that failed.

```
1  public class MyTestClass {
2      @Ignore
3      @Test
4      public void myTestMethod() {
5          /**
6           * This test will be ignored.
7           */
8          org.junit.Assert.assertTrue( new ArrayList().isEmpty() );
9      }
10 }
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

6.1: Selenium 2.0 – Web Driver Test with Xunit

Junit - Assertion

- JUnit provides overloaded assertion methods for all primitive types and Objects and arrays.
- The parameter order is expected value followed by actual value
- Some of the important methods of Assert class are:
  - void assertEquals(boolean expected, boolean actual)
    Check that two primitives/Objects are equal
  - void assertTrue(boolean expected, boolean actual)
    Check that a condition is true
  - void assertFalse(boolean condition)
    Check that a condition is false
  - void assertNotNull(Object object)
    Check that an object isn't null
  - void assertNull(Object object)
    Check that an object is null

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

- void assertSame(boolean condition)
  The assertSame() methods tests if two object references point to the same object

- void assertNotSame(boolean condition)
  The assertNotSame() methods tests if two object references not point to the same object

- void assertArrayEquals(expectedArray, resultArray);
  The assertArrayEquals() method will test whether two arrays are equal to each other

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

6.1: Selenium 2.0 – Web Driver Test with Xunit

Junit – Assertion Example

```
1  public class TestAssertions {
2      @Test
3      public void testAssertions() {
4          //test data
5          String str1 = new String ("abc");
6          String str2 = new String ("abc");
7          String str3 = null;
8          String str4 = "abc";
9          String str5 = "abc";
10         int val1 = 5;
11         int val2 = 6;
12         String[] expectedArray = {"one", "two", "three"};
13         String[] resultArray =  {"one", "two", "three"};
14         //Check that two objects are equal
15         assertEquals(str1, str2);
16         //Check that a condition is true
17         assertTrue (val1 < val2);
18         //Check that a condition is false
19         assertFalse(val1 > val2);
20         //Check that an object isn't null
21         assertNotNull(str1);
22         //Check that an object is null
23         assertNull(str3);
24         //Check if two object references point to the same object
25         assertSame(str4,str5);
26         //Check if two object references not point to the same object
27         assertNotSame(str1,str3);
28         //Check whether two arrays are equal to each other.
29         assertArrayEquals(expectedArray, resultArray);
30     }
31 }
```

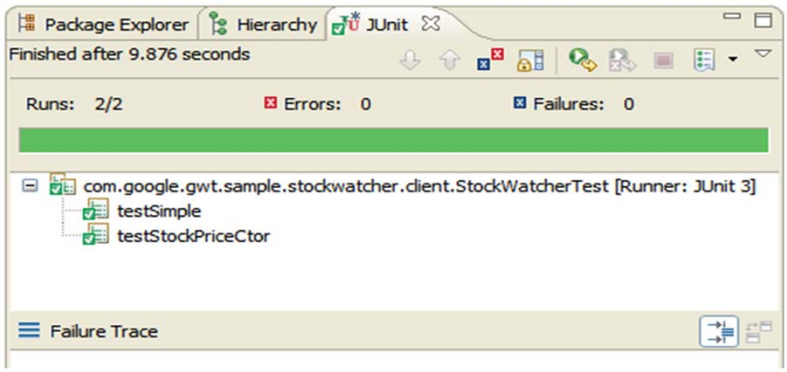
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

6.1: Selenium 2.0 – Web Driver Test with Xunit

Junit – Reports

- Junit report collects individual XML files
- Merge the individual XML files generated by the JUnit task and eventually apply a stylesheet on the resulting merged document to provide a browsable report of the test cases results



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

6.1: Selenium 2.0 – Web Driver Test with Xunit

Web Driver Test cases with TestNG

- Open source Java testing framework, not limited to unit tests
- Designed to be better than JUnit, especially when testing integrated classes
- Supports parameterized tests out-of-the-box (in much more convenient way than JUnit does)
- Facilitates running multi-threaded tests
- Allows to express dependencies between test methods
- Integrates very well with the build tools: Ant, Maven and Gradle
- Supported by all major IDEs
- Can be used with different JVM languages (e.g. Java, Groovy, Scala) and cooperates with many quality and testing tools (e.g. code coverage tools, mocking libraries, matchers libraries)
- Some popular solutions - e.g. Spring Framework - provide means to facilitate testing with TestNG

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

Writing a test is typically a three-step process:

- Write the business logic of your test and insert TestNG annotations in your code.
- Add the information about your test (e.g. the class name, the groups you wish to run, etc...) in a testng.xml file or in build.xml.
- Run TestNG

```
import org.testng.annotations.Test;
import static org.testng.Assert.assertEquals;

public class TestNGSimpleTest {
        @Test
        public void testAdd() {
                String str = "TestNG is working fine";
                assertEquals("TestNG is working fine", str);
        }
}
```

Test Case Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite1">
  <test name="test1">
    <classes>
       <class name="TestNGSimpleTest"/>
    </classes>
  </test>
</suite>
```

testng.xml File

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

6.1: Selenium 2.0 – Web Driver Test with Xunit

TestNG Annotations

Annotation	Description
@Test	The annotation notifies the system that the method annotated as @Test is a test method
@BeforeSuite	The annotation notifies the system that the method annotated as @BeforeSuite must be executed before executing the tests in the entire suite
@AfterSuite	The annotation notifies the system that the method annotated as @AfterSuite must be executed after executing the tests in the entire suite
@BeforeTest	The annotation notifies the system that the method annotated as @BeforeTest must be executed before executing any test method within the same test class

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@AfterTest	The annotation notifies the system that the method annotated as @AfterTest must be executed after executing any test method within the same test class
@BeforeClass	The annotation notifies the system that the method annotated as @BeforeClass must be executed before executing the first test method within the same test class
@AfterClass	The annotation notifies the system that the method annotated as @AfterClass must be executed after executing the last test method within the same test class
@BeforeMethod	The annotation notifies the system that the method annotated as @BeforeMethod must be executed before executing any and every test method within the same test class

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

6.1: Selenium 2.0 – Web Driver Test with Xunit

Continued.

@AfterMethod	The annotation notifies the system that the method annotated as @AfterMethod must be executed after executing any and every test method within the same test class
@BeforeGroups	The annotation notifies the system that the method annotated as @BeforeGroups is a configuration method that enlists a group and that must be executed before executing the first test method of the group
@AfterGroups	The annotation notifies the system that the method annotated as @AfterGroups is a configuration method that enlists a group and that must be executed after executing the last test method of the group

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

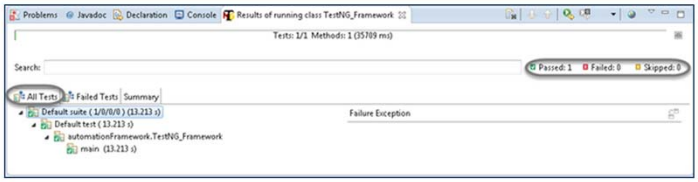
6.1: Selenium 2.0 – Web Driver Test with Xunit

TestNG Result

- TestNG result is displayed into two windows as shown below:
  - Console Window
  - Testing Result Window

a

b



 21

6.1: Selenium 2.0 – Web Driver Test with Xunit

TestNG Reports

- Generates a different type of report for test execution
- Whenever TestNG is run, HTML and XML reports are generated by default in the directory
- For implementing a reporting class, the class has to implement an org.testng.IReporter interface
- Has its own reporter objects which are called when whole suite run ends

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

6.1: Selenium 2.0 – Web Driver Test with Xunit

TestNG Reports

- Object containing the information of the whole test run is passed on to the report implementations
- Default implementations are:
  - Main
  - Failed Reporter
  - XML Reporter
  - EmailableReporter2
  - JUnitReport Reporter
  - SuiteHTML Reporter

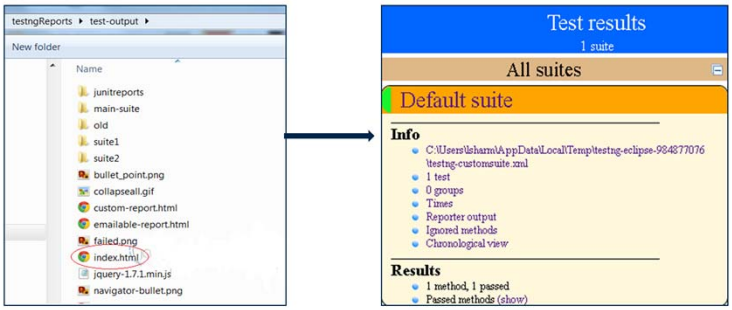
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

6.1: Selenium 2.0 – Web Driver Test with Xunit

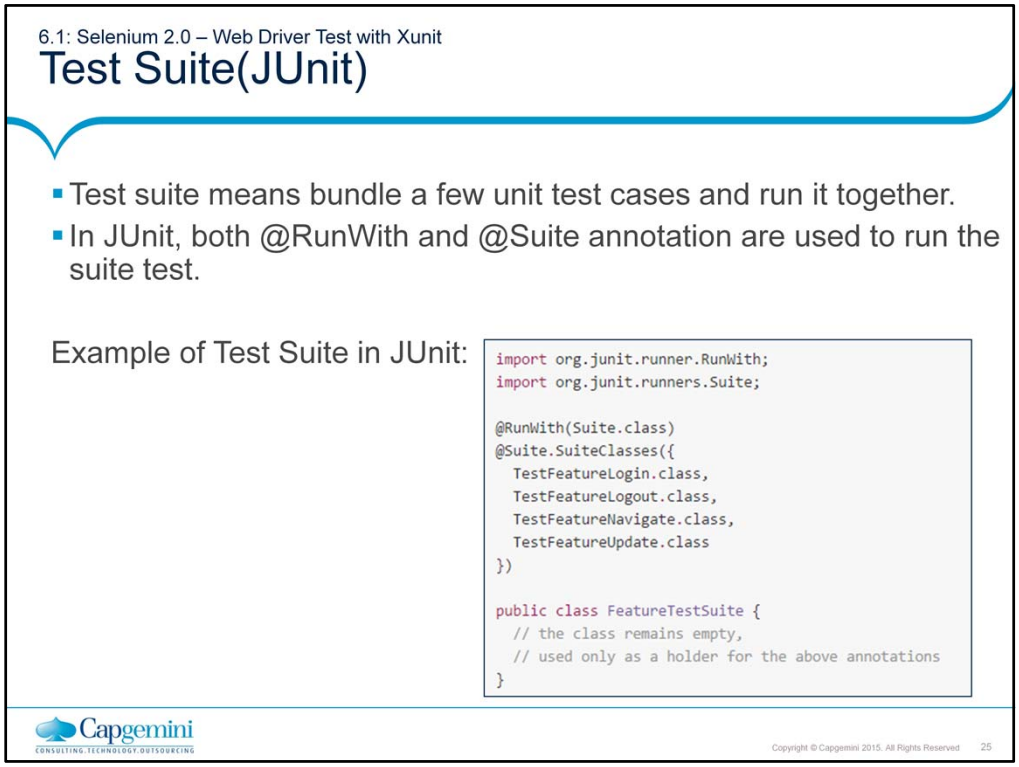
TestNG Reports

- In Main report layout ,test-output directory contains HTML reports like an index.html file, that is the entry point to the TestNG HTML report.
- The top-level report gives us a list of all the suites that were just run, along with an individual and compound total for each passed, failed, and skipped test.



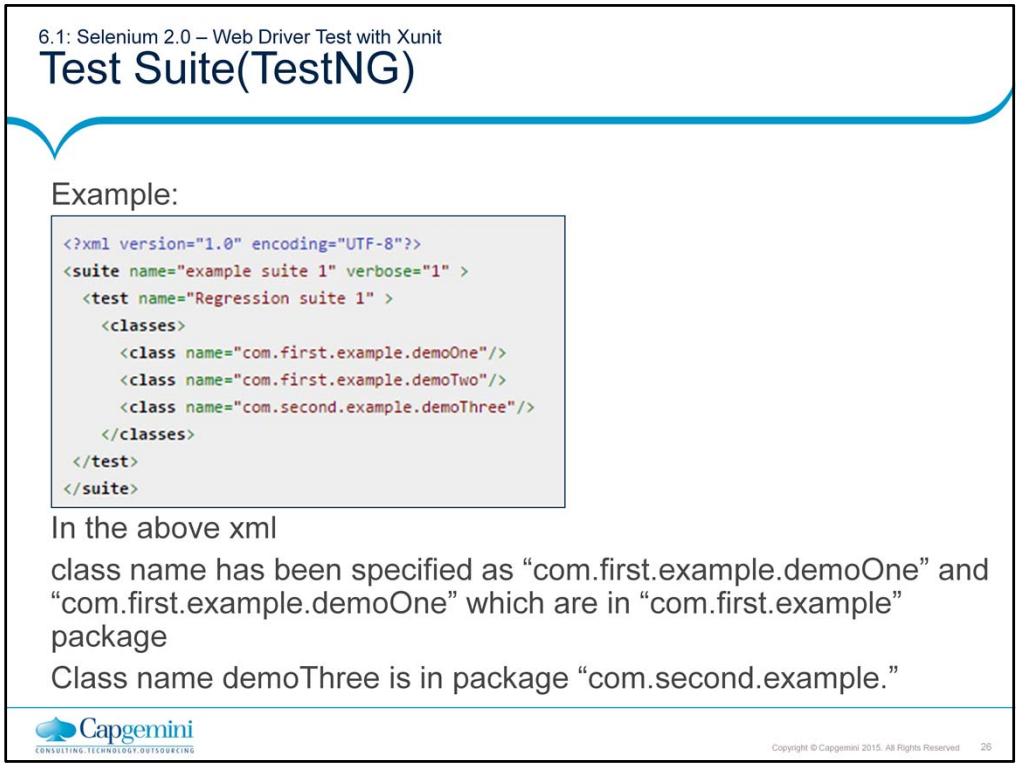
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

 24

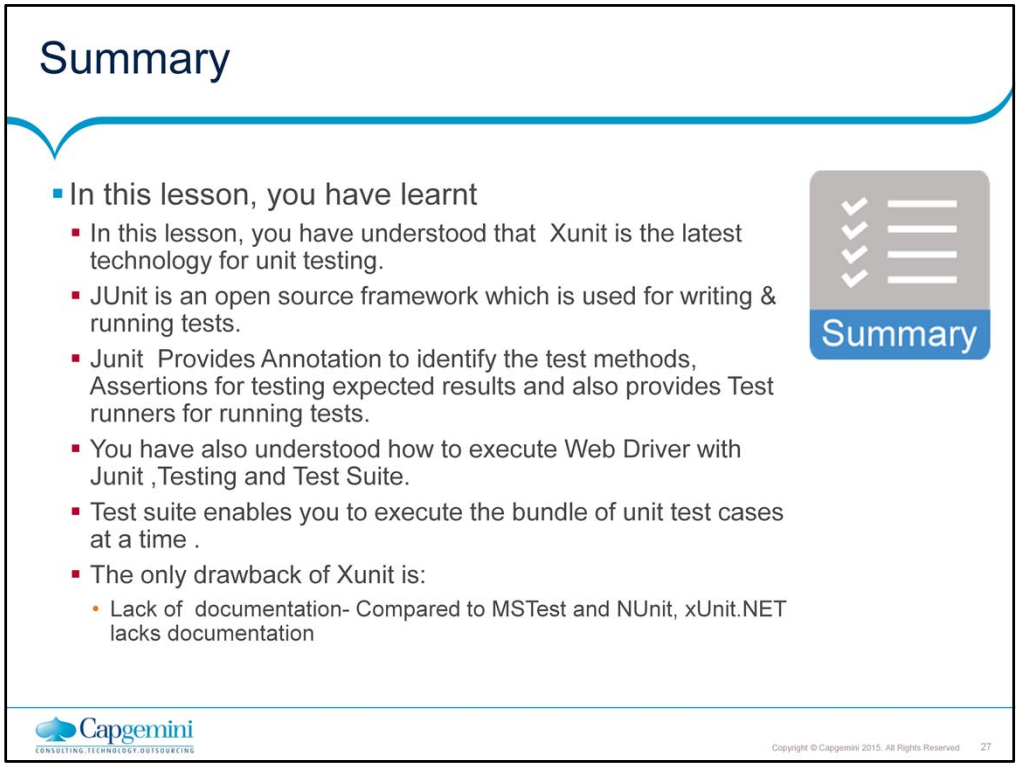


Aggregating tests in suites:
Using Suite as a runner allows you to manually build a suite containing tests from many classes. It is the JUnit 4 equivalent of the JUnit 3.8.x static Test suite() method. To use it, annotate a class with @RunWith(Suite.class) and @SuiteClasses(TestClass1.class, ...). When you run this class, it will run all the tests in all the suite classes.

Example:
The class above is a placeholder for the suite annotations, no other implementation is required.
Note : @RunWith annotation, which specifies that the JUnit 4 test runner to use is org.junit.runners.Suite for running this particular test class. This works in conjunction with the @Suite annotation, which tells the Suite runner which test classes to include in this suite and in which order.



We need to specify the class names along with packages in between the classes tags.
All the classes specified in the xml will get executes which have TestNG annotations.



Add the notes here.

Review Question

- Question 1
  - Select the Annotation which is NOT part of JUnit Annotations
  - @After
  - @After or Before
  - @Before
  - @AfterClass

- Question 2: True/False
  - The Selenium web driver is automation API not testing AP
- Question 3: Fill in the Blanks
  - The assertSame() methods tests if two object references point to the _____ .

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 28

Test Automation & Advanced Selenium

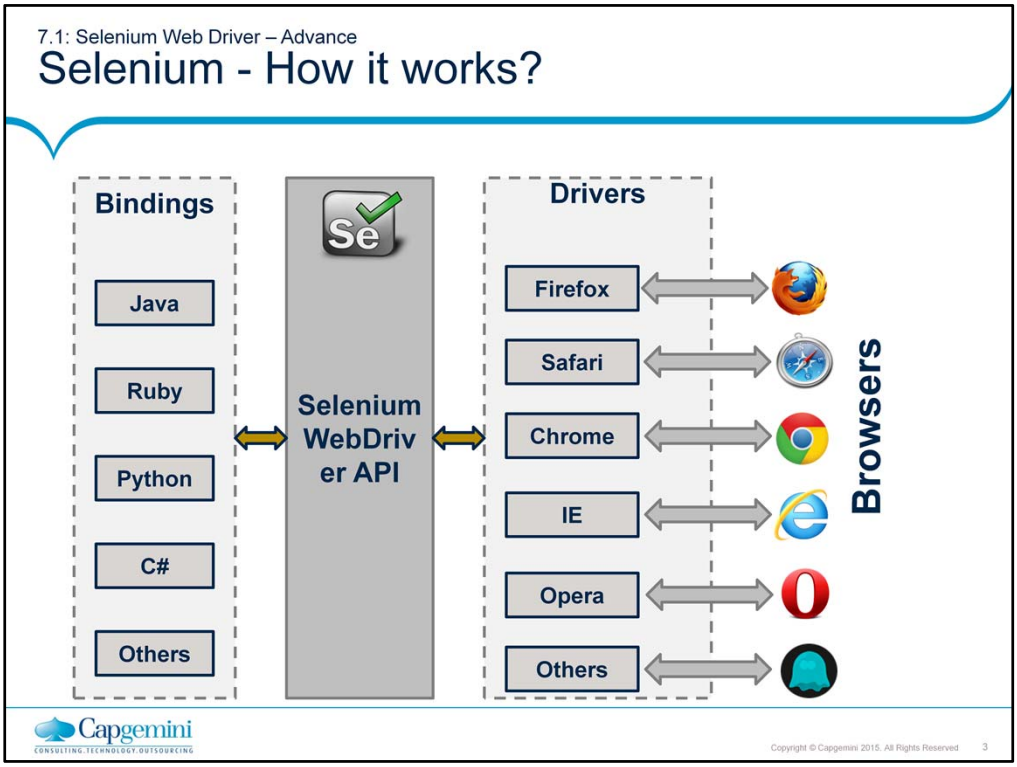
Lesson 7: Selenium Web Driver – Advance

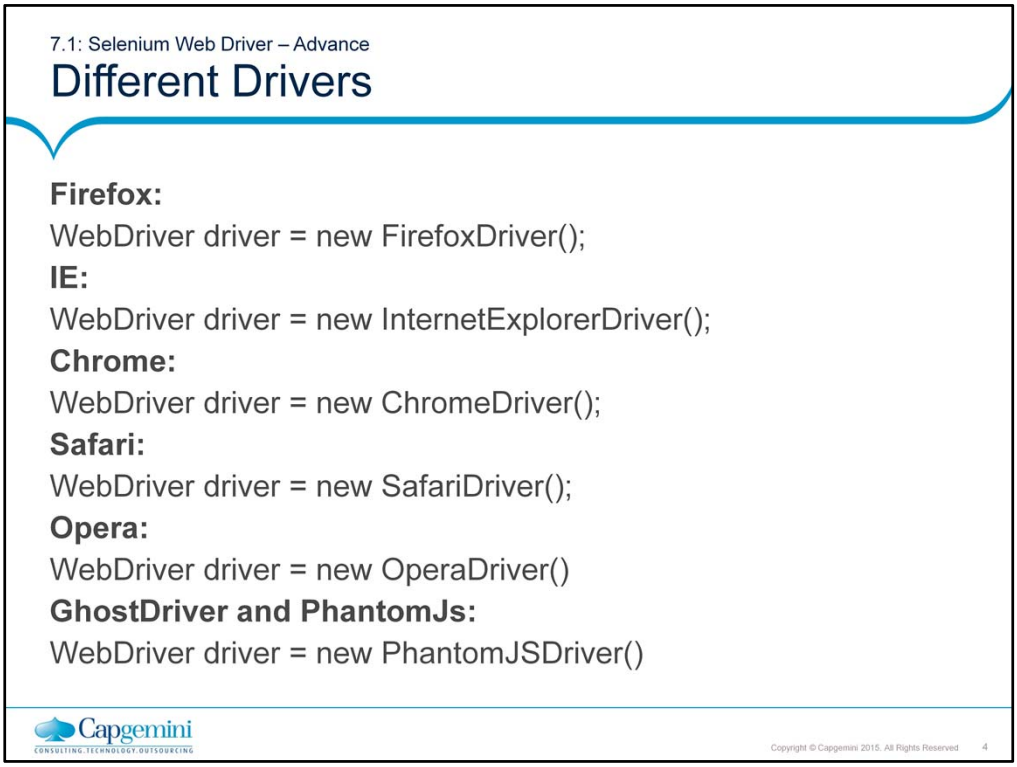
Lesson Objectives

- Selenium: How it works
- Different drivers
  - Chrome
  - Firefox
  - Internet Explorer
  - Headless Browser
    - Ghost Driver and Phantom JS
  - Mobile Browsers
    - Selendriod & Appium
  - Remote Web Driver
  - Capabilities
  - Profile setting
  - Selenium Grid

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2


7.1: Selenium Web Driver – Advance
Selenium - How it works?
Bindings
Java
Ruby
Python
C#
Others
Selenium WebDriver API
Drivers
Firefox
Safari
Chrome
IE
Opera
Others
Browsers
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved
3



1. Architecture
WebDriver's architecture is simpler than Selenium RC's.

It controls the browser from the OS level
All you need are your programming language's IDE (which contains your Selenium commands) and a browser.
You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing
The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser
When you begin testing, Selenium RC Server "injects" a Javascript program called Selenium Core into the browser.
Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.
When the instructions are received, Selenium Core will execute them as Javascript commands.
The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.
The RC Server will receive the response of the browser and then display the results to you.
RC Server will fetch the next instruction from your test script to repeat the whole cycle.

2. Speed
WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

- Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.

3. Real-life Interaction
WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

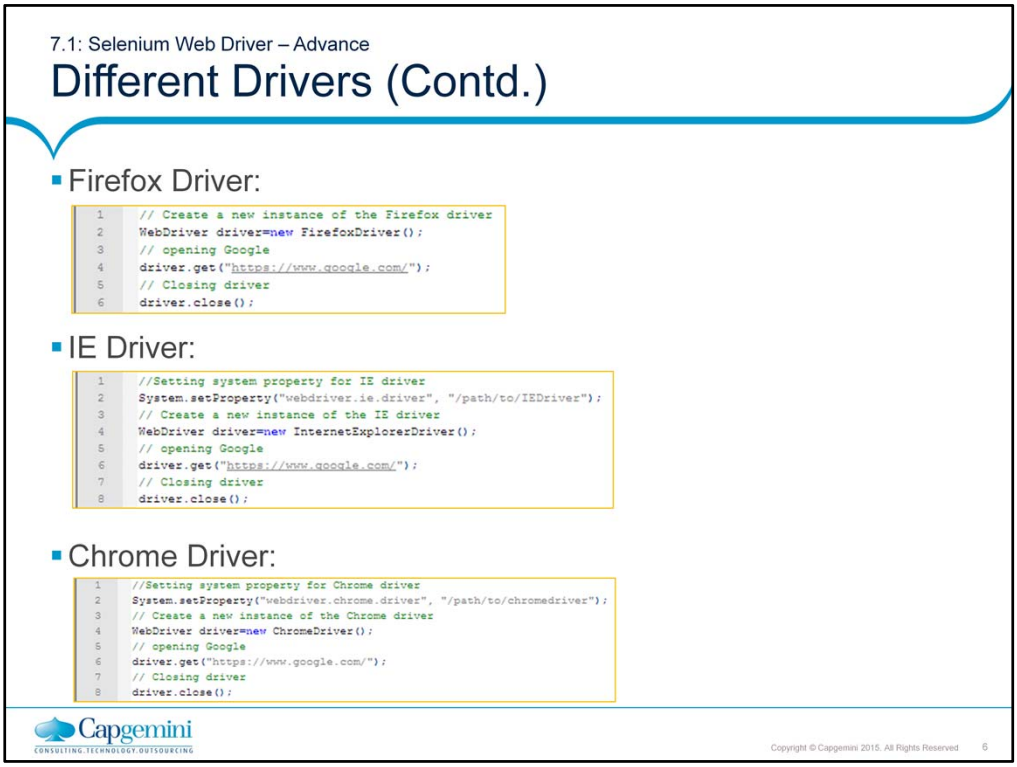
- Selenium Core, just like other Javascript codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests. Differences in API

4. API
Selenium RC's API is more matured but contains redundancies and often confusing commands.

- For example, most of the time, testers are confused whether to use type or typeKeys; or whether to use click, mouseDown, or mouseDownAt. Worse, different browsers interpret each of these commands in different ways tool
- WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.

5. Browser Support
WebDriver can support the headless HtmlUnit browser

- HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.
- It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.
- Since it is invisible to the user, it can only be controlled through automated means.
- Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.



1. Architecture
WebDriver's architecture is simpler than Selenium RC's.

It controls the browser from the OS level
All you need are your programming language's IDE (which contains your Selenium commands) and a browser.
You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing
The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser
When you begin testing, Selenium RC Server "injects" a Javascript program called Selenium Core into the browser.
Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.
When the instructions are received, Selenium Core will execute them as Javascript commands.
The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.
The RC Server will receive the response of the browser and then display the results to you.
RC Server will fetch the next instruction from your test script to repeat the whole cycle.

2. Speed
WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

- Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.

3. Real-life Interaction
WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

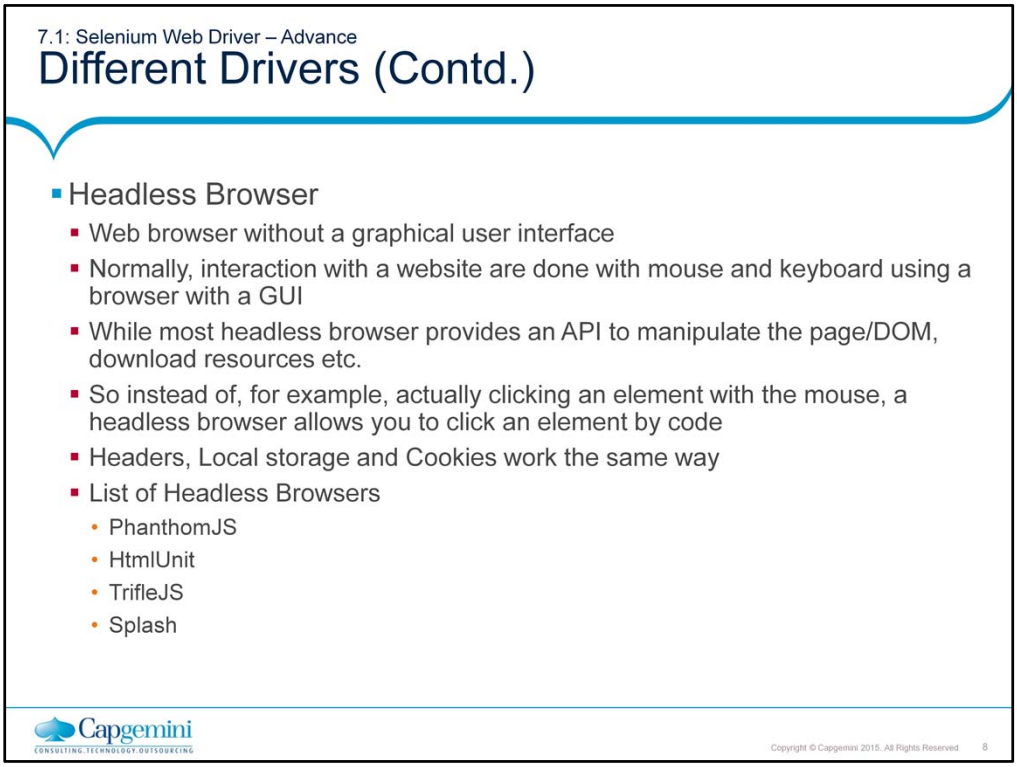
- Selenium Core, just like other Javascript codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests. Differences in API

4. API
Selenium RC's API is more matured but contains redundancies and often confusing commands.

- For example, most of the time, testers are confused whether to use type or typeKeys; or whether to use click, mouseDown, or mouseDownAt. Worse, different browsers interpret each of these commands in different ways tool
- WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.

5. Browser Support
WebDriver can support the headless HtmlUnit browser

- HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.
- It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.
- Since it is invisible to the user, it can only be controlled through automated means.
- Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.



Headless browsers are typically used in following situations:
1. You have a central build tool which does not have any browser installed on it. So to do the basic level of sanity tests after every build you may use the headless browser to run your tests.
2. You want to write a crawler program that goes through different pages and collects data, headless browser will be your choice. Because you really don't care about opening a browser. All you need is to access the webpages.
3. You would like to simulate multiple browser versions on the same machine. In that case you would want to use a headless browser, because most of them support simulation of different versions of browsers. We will come to this point soon.

Things to pay attention to before using headless browser
Headless browsers are simulation programs, they are not your real browsers. Most of these headless browsers have evolved enough to simulate, to a pretty close approximation, like a real browser. Still you would not want to run all your tests in a headless browser. JavaScript is one area where you would want to be really careful before using a Headless browser. JavaScript are implemented differently by different browsers. Although JavaScript is a standard but each browser has its own little differences in the way that they have implemented JavaScript. This is also true in case of headless browsers also. For example HtmlUnit headless browser uses the Rihno JavaScript engine which not being used by any other browser.

Selenium support for headless browser
Selenium supports headless testing using its class called HtmlUnitDriver. This class internally uses HtmlUnit headless browser. HtmlUnit is a pure Java implementation. HtmlUnitWebDriver can be created as mentioned below:

```
HtmlUnitDriver unitDriver = new HtmlUnitDriver();
```

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

- PhanthomJS (Headless Browser)
  - HEADLESS WEBSITE TESTING
    - Headless Web Kit with JavaScript API
  - SCREEN CAPTURE
    - Programmatically capture web contents, including SVG and Canvas
  - PAGE AUTOMATION
    - Access and manipulate webpages with the standard DOM API, or with usual libraries like jQuery
  - Example of interacting with a page using PhantomJS:

```
page.evaluate(function() {
        //Fill in form on page
        document.getElementById('Name').value = 'John Doe';
        document.getElementById('Email').value = 'john.doe@john.doe';
        //Submit
        $('#SubmitButton').click();
    });
```

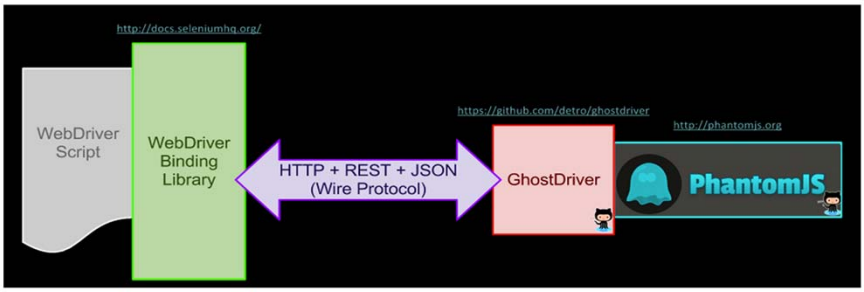
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

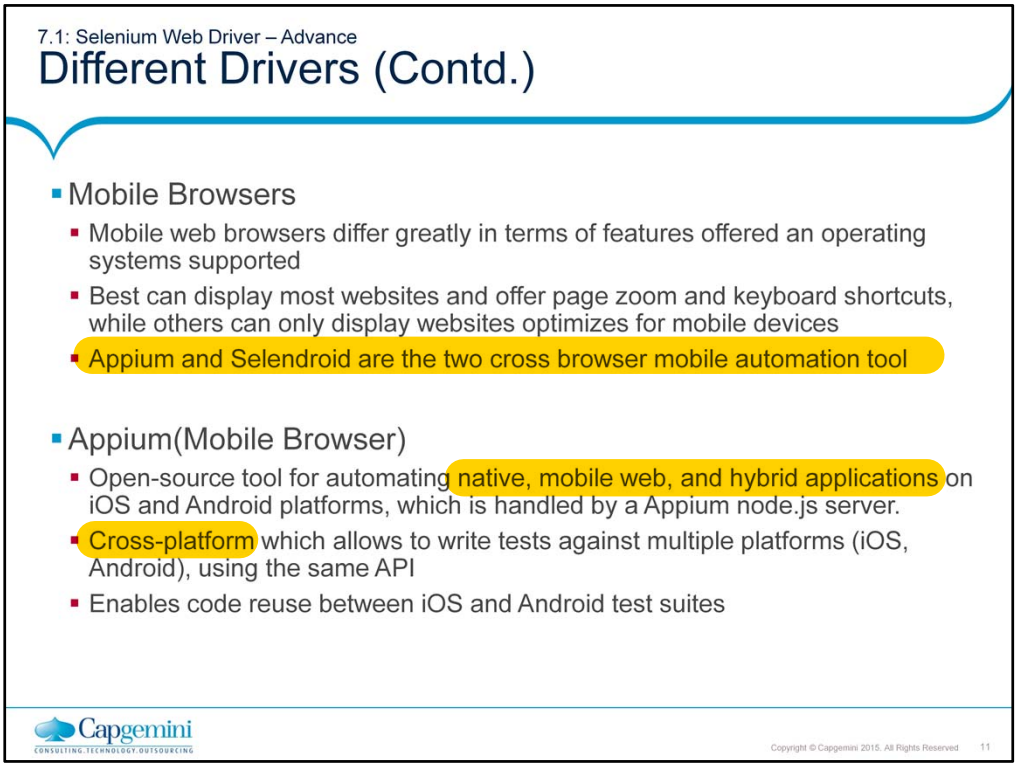
- GhostDriver(Headless Browser)
  - Pure JavaScript implementation of the WebDriver Wire Protocol for PhantomJS Remote
  - WebDriver that uses PhantomJS as back-end



Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

 10

GhostDriver is the project that provides the code that exposes the WebDriver API for use within PhantomJS.
PhantomJS bakes the GhostDriver code into itself, and ships it as part of its downloadable executable.
Thus, to use "GhostDriver" with PhantomJS, only PhantomJS is needed.



Native apps are those written using the iOS or Android SDKs.
Mobile web apps are web apps accessed using a mobile browser (Appium supports Safari on iOS and Chrome or the built-in 'Browser' app on Android). Hybrid apps have a wrapper around a "webview" -- a native control that enables interaction with web content. Projects like Phonegap, make it easy to build apps using web technologies that are then bundled into a native wrapper, creating a hybrid app.

Appium was designed to meet mobile automation needs according to a philosophy outlined by the following four tenets:

You shouldn't have to recompile your app or modify it in any way in order to automate it.
You shouldn't be locked into a specific language or framework to write and run your tests.
A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.
A mobile automation framework should be open source, in spirit and practice as well as in name.

7.1: Selenium Web Driver – Advance

Different Drivers (Contd.)

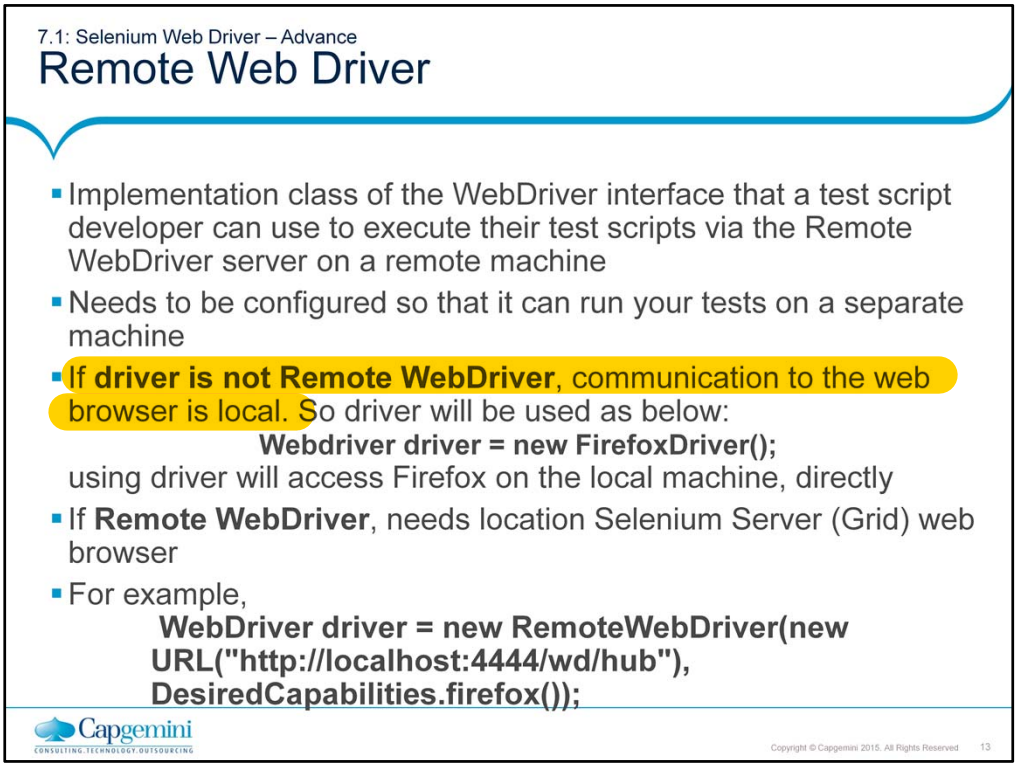
Selendroid

Test automation framework which is used Android native & hybrid applications (apps) and mobile web

- Full compatibility with the JSON Wire Protocol
- No modification of app under test required in order to automate it
- Testing the mobile web using built in Android driver webview app
- UI elements can be found by different locator types
- Selendroid can interact with multiple Android devices (emulators or hardware devices) at the same time
- Existing emulators are started automatically
- Supports hot plugging of hardware devices
- Full integration as a node into Selenium Grid for scaling and parallel testing

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12



This example assumes that Selenium Server is running on localhost with the default port of 4444. The nice thing about this is you can run Selenium Server on any machine, change the URL to point to the new machine and run the tests. For example, I run the test code from my Mac OS X computer but I run Selenium Server on a Window XP machine. This way I can launch Internet Explorer tests from my Mac OS X computer.

7.1: Selenium Web Driver – Advance

Capabilities and Profile Setting

- Capabilities describes a series of key/value pairs that encapsulate aspects of a browser
- DesiredCapabilities set properties for the WebDriver
- Profile used to create custom Firefox profile and use it with desired capabilities

- Example to use Firefox profile with desired capabilities:

```
DesiredCapabilities dc=DesiredCapabilities.firefox();
FirefoxProfile profile = new FirefoxProfile();
dc.setCapability(FirefoxDriver.PROFILE, profile);
Webdriver driver =  new FirefoxDriver(dc);
```

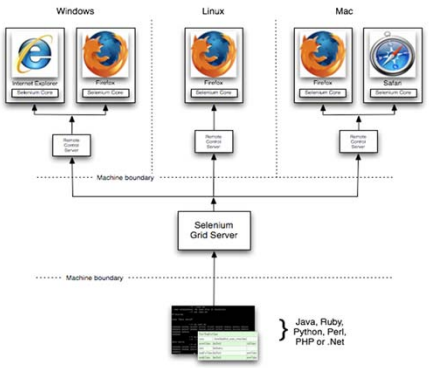
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

7.1: Selenium Web Driver – Advance

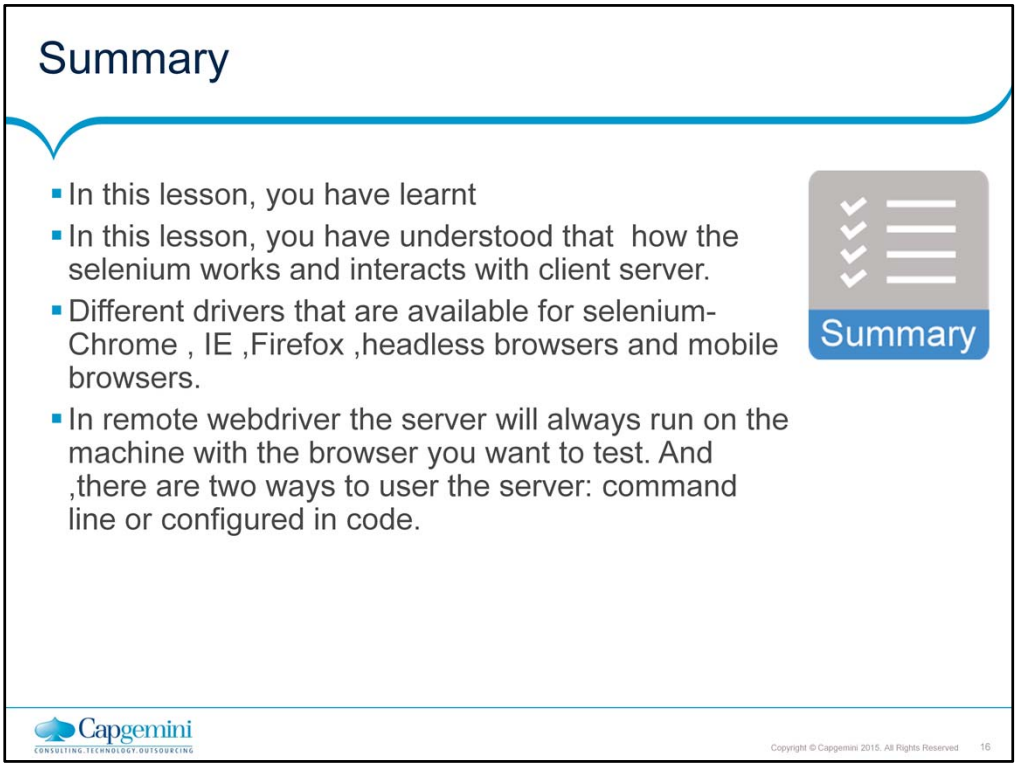
Selenium Grid

Allows you run your tests on different machines against different browsers in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems. Essentially, Selenium-Grid support distributed test execution. It allows for running your tests in a distributed test execution environment.

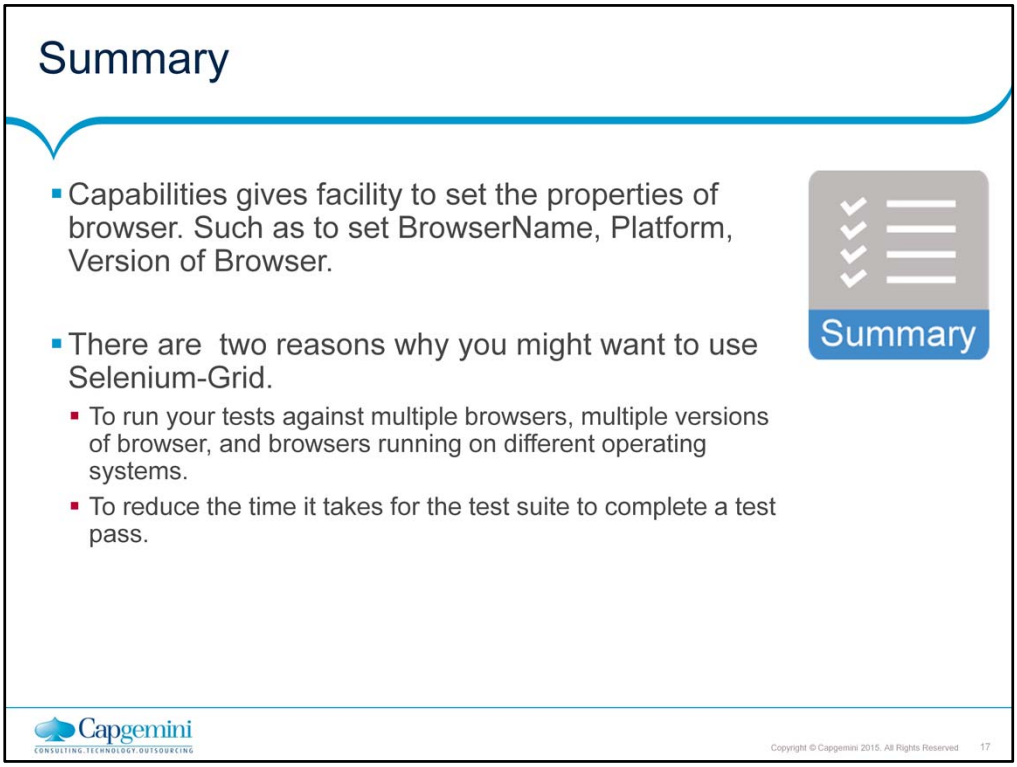


Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15



Add the notes here.



Add the notes here.

Review Question

- Question 1:
  - PhanthomJS is
  - Chrome Driver
  - Mobile Browser
  - Headless Browser
  - Firefox Driver
- Question 2: True/False
  - Firefox saves your information such as cookies and browser history in a file called your profile.
- Question 3: Fill in the Blanks
  - Client driver will send the program that is written in eclipse IDE as _____ and send it to Selenium server

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

Test Automation & Advanced Selenium

Lesson 8 : Selenium Frameworks

Lesson Objectives

- Framework Overview
- Data Driven (Excel, Databases)
- Keyword Driven
- Component based (Sprintest®/CBF)
- Reports (Excel, PDF)
- TDD (Junit, TestNG)
- BDD (Cucumber, SpecFlow)
- ATDD (Fitnesse)
- CI Tools (Jenkins etc)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2

8.1: Selenium Frameworks

Framework Overview

- Automation testing requires a well-defined approach, based on a comprehensive framework, in order to reap maximum benefits.
- A framework is a hierarchical directory that encapsulates shared resources, such as a dynamic shared library, image files, localized strings, header files, and reference documentation, in a single package. There are various frameworks available for automation, such as:
  - Data-Driven Automation Framework.
  - Keyword-Driven Automation Framework.
  - Hybrid Automation Framework.
- Component based framework(CBF) is an hybrid Automation framework.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 3

8.1: Selenium Frameworks

Data Driven

- Data driven is the design of possible inputs what may given by the end user. This would cover maximum probabilities of an input data.
- It can be Spread sheet or sql. We have to connect and pass the values to the respective field or element.
- In this type of framework Data is not hardcoded with script ,but data is provided from external source.
- When some test needs to repeat for different data set , Data driven framework gets used. In this framework, Parameters in the test case gets linked to database, excel, csv, text files from there test case run for all defined parameter in the file.
- Below figure shows example of the data driven, test case stored in excel

	A	B	C	D	E	F
1	TestCase ID	URL	UserName	Password	EmailSubject	FBSearch
2	TC_Gmail	www.gmail.com	abcd	passwrd		
3	TC_YahooMail	www.yahoomail.com	abc@yahoo.com	passwrd	Good Morning	
4	TC_Facebook	www.facebook.com	abc@somemail.com	passwrd		abc def
5						
6						
7						
8						
9						
10						

All the test cases are saved in a single excel sheet.

Few cells would be blank as they are not used by all the test cases.

TestData www.automationrepository.com

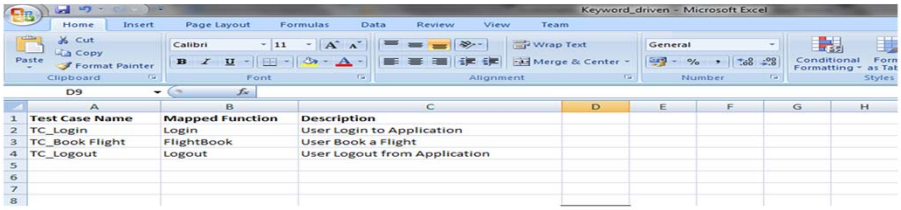
Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

 4

8.1: Selenium Frameworks

Keyword-Driven

- As Name suggest, Keyword nothing but a code which represent some action, say “login”. in this framework, we map the set of code which perform certain action with a keyword and then we use that keyword across the framework.
- The Keyword-Driven or Table-Driven framework requires the development of data tables and keywords, independent of the test automation tool used to execute them. Keyword-driven tests look very similar to manual test cases.
- The driver code "drives" the application-under-test, keyword driven test and the data. In a keyword-driven test, the functionality of the application-under-test is documented in a table like structure for e.g. Excel Sheet
- Below figure shows example of the keyword-driven, test case stored in excel



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 5

1. Architecture
WebDriver's architecture is simpler than Selenium RC's.

It controls the browser from the OS level
All you need are your programming language's IDE (which contains your Selenium commands) and a browser.
You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing
The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser
When you begin testing, Selenium RC Server "injects" a Javascript program called Selenium Core into the browser.
Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.
When the instructions are received, Selenium Core will execute them as Javascript commands.
The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.
The RC Server will receive the response of the browser and then display the results to you.
RC Server will fetch the next instruction from your test script to repeat the whole cycle.

2. Speed

WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

Selenium RC is slower since it uses a Javascript program called Selenium Core. This Selenium Core is the one that directly controls the browser, not you.

3. Real-life Interaction

WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

Selenium Core, just like other Javascript codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests.

Differences in API

4. API

Selenium RC's API is more matured but contains redundancies and often confusing commands.

For example, most of the time, testers are confused whether to use type or typeKeys; or whether to use click, mouseDown, or mouseDownAt. Worse, different browsers interpret each of these commands in different ways tool

WebDriver's API is simpler than Selenium RC's. It does not contain redundant and confusing commands.

5. Browser Support

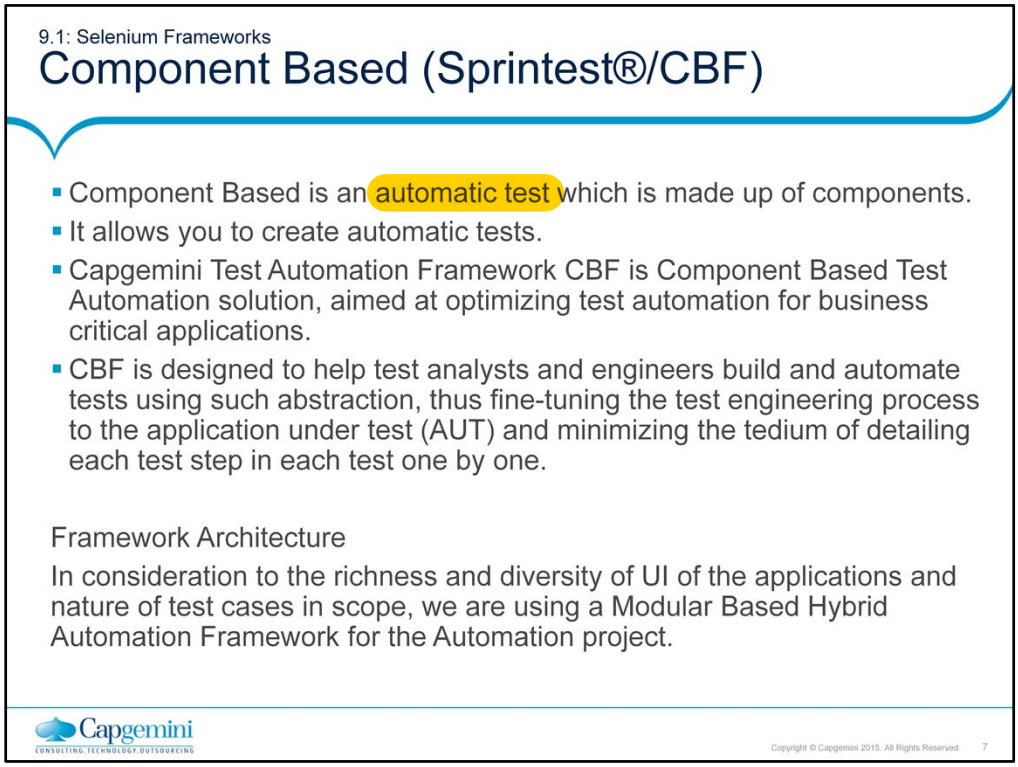
WebDriver can support the headless HtmlUnit browser

HtmlUnit is termed as "headless" because it is an invisible browser - it is GUI-less.

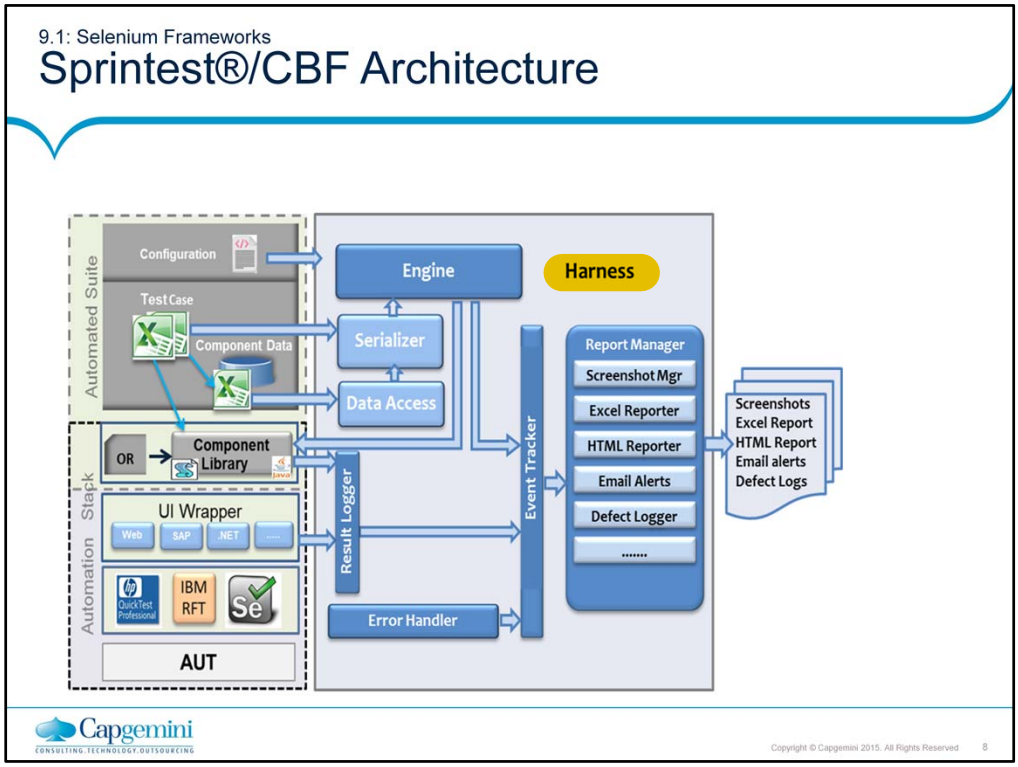
It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.

Since it is invisible to the user, it can only be controlled through automated means.

Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.



1.File Menu
It is very much analogous to the file menu belonging to any other application. Export Test Case As and Export Test Suite give the liberty to the user to prefer amid the available unit testing frameworks like jUnit, TestNG etc.
Thus an IDE test case can be exported for a chosen union of programming language, unit testing framework and tool from the selenium package.



The Selenium IDE test cases can be saved into following format:
HTML format
The Selenium IDE test cases can be exported into following formats/programming languages.
java (IDE exported in Java)
rb (IDE exported in Ruby)
py (IDE exported in Python)
cs (IDE exported in C#)

8.1: Selenium Frameworks

Continued.

- Harness: Execution gets triggered from here. Initializes the result logs, reports and initiates execution.
- Engine: General purpose runner for executing component based test cases. It supports a robust event model for reporting. Also Encapsulates runtime exception handling, test failures and recovery.
- Serializer: Deserializes the Auto_TC.xls file, resolves the references with the help of Data Access and creates a testcase object for execution.
- Data Access: Reads data from centrally maintained data files at the module level, thus helps in resolving data refs.
- Error Handler: Logs errors in a csv file, thus helps in tracking error.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 9

8.1: Selenium Frameworks

Continued.

- Result Logger: Exposes set of methods like Passed, Failed, Error, Done etc, which will be used to log results in the component libraries on verification. The logged results are made available in the selected reports.
- Report Manager: Manages different report types on user selection like ScreenshotMgr, Excel reporter, HTML Reporter, Email Alerts etc..
- Component Libraries:Collection of automation driver scripts for each action. Provision to organize functions into a small number of module driver files for maintainability. Simplifies test layer to a series of calls to this layer.
- OR:Used to store the Object Repository map file.
- Configuration: Refers to either hardware or software, or the combination of both

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 10

8.1: Selenium Frameworks

Continued.

- Configuration: Refers to either hardware or software, or the combination of both
- Component Data: Data maintained for components in the module excel workbooks.
- TestCase: Assembly of different combinations of components.
- Application Under Test: Application considered for automation.
- Automated testing tools: Execute tests, report outcomes and compare results with earlier test runs.
- UI Wrapper: Generic methods like SetValue(), Click(), GetCtrlProperty() etc.. which will be used in scripting the components.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 11

8.1: Selenium Frameworks

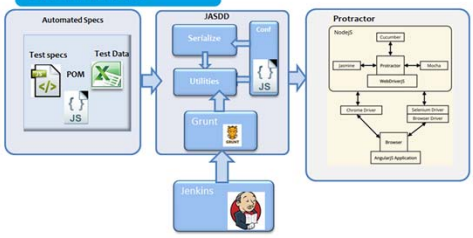
Sprintest®/Quadrant

- A Solution built over Protractor with options for Excel based test data and HTML & Excel based reports primarily for test automation of AngularJS based applications.

Highlights

- Uses Grunt for easy integration with CI tools and Command Line invocation as well
- Provides Test Data in Excel
- Provides facility to select Browser and Test Suite to executed
- Provides HTML report with Screenshots of failed steps

Architecture



Benefits

- Easy management of executions and results from a Jenkins central console
- Easily integrates with Jenkins .
- Provides to create Data Driven Test cases

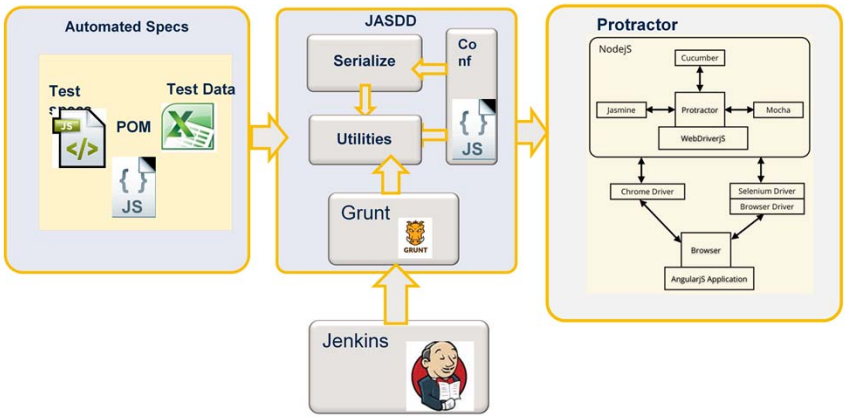
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 12

The Selenium IDE test cases can be saved into following format:
HTML format
The Selenium IDE test cases can be exported into following formats/programming languages.
java (IDE exported in Java)
rb (IDE exported in Ruby)
py (IDE exported in Python)
cs (IDE exported in C#)

8.1: Selenium Web Driver – Advance

Sprintest®/Quadrant Architecture



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 13

8.1: Selenium Frameworks

Reports

- Different applications have different reporting needs. Some require combined results for a test batch and some require individual level test report for each test case in test batch. Thus, framework should be flexible enough to generate required reports.
- Reports output in many formats such as Html/Excel/PDF Format.
- We can also capture and store all the screen shots for failed and passed scenarios
- It also store the executed data sheets with Pass/Fail status.
- It provides two types of reports
  - summary report and
  - detailed report
- The summary report provides details of execution duration, test start time and end time
- The detailed reports describe exceptional cases handled, steps passed, and steps failed.

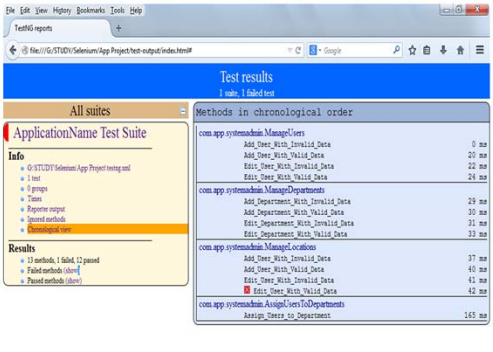
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 14

8.1: Selenium Web Driver – Advance

Continued.

- The below(fig:a) is the sample html format summary report generated by TestNG.
- The below(fig:b) is the example report to see failed test cases report in detail



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

Log/Reference/UI-Element/Rollup Pane
The bottom pane is used for four different functions–Log, Reference, UI-Element, and Rollup–depending on which tab is selected.

Log
When you run your test case, error messages and information messages showing the progress are displayed in this pane automatically, even if you do not first select the Log tab. These messages are often useful for test case debugging. Notice the Clear button for clearing the Log. Also notice the Info button is a drop-down allowing selection of different levels of information to log.

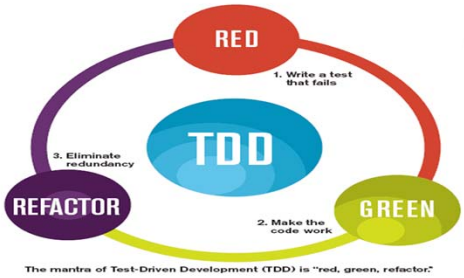
Reference
The Reference tab is the default selection whenever you are entering or modifyingSelenese commands and parameters in Table mode. In Table mode, the Reference pane will display documentation on the current command. When entering or modifying commands, whether from Table or Source mode, it is critically important to ensure that the parameters specified in the Target and Value fields match those specified in the parameter list in the Reference pane. The number of parameters provided must match the number specified, the order of parameters provided must match the order specified, and the type of parameters provided must match the type specified. If there is a mismatch in any of these three areas, the command will not run correctly.

While the Reference tab is invaluable as a quick reference, it is still often necessary to consult the Selenium Reference document.

8.1: Selenium Frameworks

TDD

- Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards.



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

RED – The developer writes a failing test essentially capturing the requirements in a test.
GREEN – The developer implements the business functionality, writing just enough code to pass the test
REFACTOR – The developer refines and improves the code without adding new functionality.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 16

Action: Used to change the state of the AUT(Application under Test)
like click on some link, type a value in edit box, select some options on the page, select a value from drop down etc.
When action is performed on AUT the test will fail if the action is not achieved.

Accessor:-
This command check the state of the application and save the application state in some variable. It can be used for automatic generation of assertions.

Assertions:
Are very similar to checkpoint in UFT/QTP. Assertion verifies the state of the application matches it with the expected state and generates the True/False result.

8.1: Selenium Frameworks

Acceptance TDD

- There are two levels of TDD:
  - Acceptance TDD (ATDD):It is a development methodology based on communication between the business customers, the developers, and the testers .
  - With ATDD you write a single acceptance test, or behavioral specification depending on your preferred terminology, and then just enough production functionality/code to fulfill that test.
  - The goal of ATDD is to specify detailed, executable requirements for your solution on a just in time (JIT) basis. Since, ATDD encompasses many of the same practices it is also called Behavior Driven Development (BDD)
  - Behavior Driven(BDD) :Behavior-driven development combines practices from TDD and from ATDD.
  - It includes the practice of writing tests first, but focuses on tests which describe behavior, rather than tests which test a unit of implementation. Tools such as Mspec and Specflow provide a syntax which allow non-programmers to define the behaviors which developers can then translate into automated tests.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 17

8.1: Selenium Frameworks

ATDD FrameWork

- FitNesse
  - FitNesse is one of the Acceptance TDD framework.
  - It is a web server, a wiki and an automated testing tool for software.
  - It allows users of a developed system to enter specially formatted input (its format is accessible to non-programmers).
  - The input is interpreted and tests are created automatically. These tests are then executed by the system and output is returned to the user.
  - The advantage of this approach is very fast feedback from users. The developer of the system to be tested needs to provide some support
  - It is written in Java. The program first supported only Java, but versions for several other languages have been added over time likeC++, Python, Ruby, Delphi, C#, etc.
  - Different Principles of fitNesse is described in the next page.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 18

8.1: Selenium Frameworks

Continued.

- FitNesse as a testing method
  - It was originally designed as a highly usable interface around the Fit framework. As such its intention is to support an agile style of black-box testing acceptance and regression testing. In this style of testing the functional testers in a software development project collaborate with the software developers to develop a testing suite.
- FitNesse as a testing tool
  - Tests are described in Fitnesse as some sort of coupling of inputs and expected output. These couplings are expressed as some sort of variation of a decision table. The FitNesse tool supports several of these variations, ranging from literal decision tables to tables that execute queries to tables that express testing scripts (i.e. a literal ordering of steps that must be followed to reach a result).
- FitNesse as a software tool
  - It is a tool developed in Java and shipped as a single, executable jar file. The executable includes a wiki engine, an embedded web server, a testing engine and all the resources (images, stylesheets and so on) required to create a web site in FitNesse's own style.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

8.1: Selenium Frameworks

Behavior TDD

- In software engineering, behavior-driven development is a software development process that emerged from test-driven development (TDD).
- Behavior-driven development combines the general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.
- Although BDD is principally an idea about how software development should be managed by both business interests and technical insight, the practice of BDD does assume the use of specialized software tools to support the development process.
- BDD is an agile software development technique that encourages collaboration between developers, QA, and non-technical or business participants in a software project. It's more about business specifications than about tests. You write a specification for a story and verify whether the specs work as expected.
- Some of BTDD Frameworks are:
  - Cucumber
  - Spec Flow

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 20

8.1: Selenium Frameworks

BTDD Frameworks

Cucumber

- Cucumber is a software tool that computer programmers use for testing other software. It runs automated acceptance tests written in a behavior-driven development (BDD) style.
- Cucumber is written in the Ruby programming language.
- Cucumber allows the execution of feature documentation written in business-facing text.
  - Example:
    - A feature definition, with a single scenario:
- Feature:
  - Division In order to avoid silly mistakes Cashiers must be able to calculate a fraction
- Scenario:
  - Regular numbers * I have entered 3 into the calculator * I press divide * I have entered 2 into the calculator * I press equal * The result should be 1.5 on the screen.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 21

8.1: Selenium Frameworks

Continued.

The execution of the test implicit in the feature definition above requires the definition, using the Ruby language, of a few "steps"

```
Before do @calc = Calculator.new
end
After do
End
 Given /I have entered (\d+) into the calculator/ do |n|
@calc.push n.to_i
end
When /I press (\w+)/ do |op|
@result = @calc.send op
end
Then /the result should be (.*) on the screen/ do |result|
 @result.should == result.to_f
end
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 22

8.1: Selenium Frameworks

Continued.

Specflow

- SpecFlow is a BDD library/framework for .NET that adds capabilities that are similar to Cucumber. It allows to write specification in human readable Gherkin format
- Gherkin is the language that Cucumber understands. It is a Business Readable, Domain Specific Language that lets you describe software behavior without detailing with how that behavior is implemented. It's simply a DSL for describing the required functionality for a given system. This functionality is broken down by feature, and each feature has a number of scenarios.
- It is open source and provided under a BSD license . As a part of the Cucumber family, it uses the official Gherkin parser and provides integration to the .NET framework, Silverlight, Windows Phone and Mono.
- It bridges the communication gap between domain experts and developers by binding business readable behavior specifications and examples to the underlying source code.
- It also supports the concepts of Acceptance Test Driven Development (ATDD) and Behavior Driven Development (BDD).

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 23

8.1: Selenium Frameworks

Continued.

Example:

- THE ESSENCE IN two EASY STEPS

Specify!

Describe behavior and evolve a business readable testing and specification DSL.

Answers.feature

```
Feature: Ordering answers

Scenario: The answer with the highest vote gets to the top
  Given there is a question "What's your favorite colour?" with the answers
    | Answer         | Vote |
    | Red            | 1    |
    | Cucumber green | 1    |
  When you upvote answer "Cucumber green"
  Then the answer "Cucumber green" should be on top
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 24

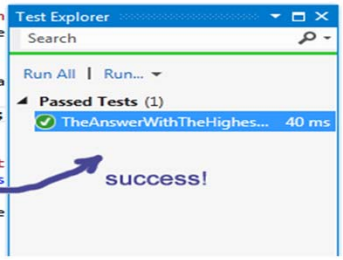
8.1: Selenium Frameworks

Continued.

Automate!

- Automate scenarios to establish a continuously validated living documentation.

```
[Binding]
public class OrderingAnswersSteps
{
    private readonly QuestionPage questionPage;

    public OrderingAnswersSteps(QuestionPage questionPage)...

    [Given(@"there is a question '(.*)' with
    public void GivenThereIsAQuestionWithThe

    [When(@"you upvote answer '(.*)'")]
    public void WhenYouUpvoteAnswer(string a
    {
        questionPage.VoteUpQuestion(answer);
    }

    [Then(@"the answer '(.*)' should be on t
    public void ThenTheAnswerShouldBeOnTop(s
    {
        Assert.AreEqual(answer, questionPage
    }
}
```



Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

8.1: Selenium Frameworks

Continuous Integration.

- Continuous integration (CI) is the practice, in software engineering, of merging all developer working copies to a shared mainline several times a day.
- The main aim of CI is to prevent integration problems, referred to as "integration hell" in early descriptions of XP. CI isn't universally accepted as an improvement over frequent integration, so it is important to distinguish between the two as there is disagreement about the virtues of each.
- In addition to automated unit tests, organizations using CI typically use a build server to implement continuous processes of applying quality control in general — small pieces of effort, applied frequently.
- Continuous integration involves integrating early and often, so as to avoid the pitfalls of "integration hell". The practice aims to reduce rework and thus reduce cost and time.
- A complementary practice to CI is that before submitting work, each programmer must do a complete build and run (and pass) all unit tests. Integration tests are usually run automatically on a CI server when it detects a new commit.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

8.1: Selenium Frameworks

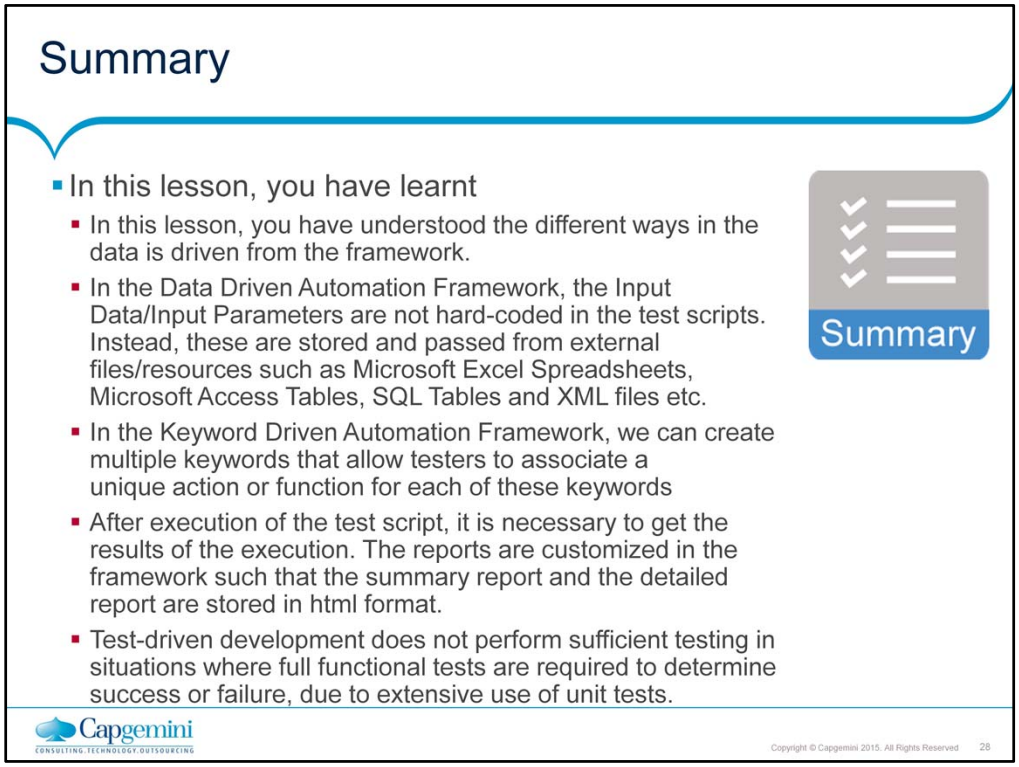
CI Tools

Jenkins

- Jenkins provides continuous integration services for software development. It is a server-based system running in a servlet container such as Apache Tomcat .
- Plugins have been released for Jenkins that extend its use to projects written in languages other than Java. Plugins are available for integrating Jenkins with most version control systems and big databases. Many build tools are supported via their respective plugins. Plugins can also change the way Jenkins looks or add new functionality.
- Builds can generate test reports in various formats supported by plugins (JUnit support is currently bundled) and Jenkins can display the reports and generate trends and render them in the GUI.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 27



Add the notes here.

Review Question

- Question 1:
  - CBF is
  - Data Driven Framework
  - Key word Driven Framework
  - Hybrid Framework
  - None of the above

- Question 2: True/False
  - The summary report provides details of execution duration, test start time and end time

- Question 3: Fill in the Blanks
  - Cucumber is written in the _____language.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 29

Test Automation & Advanced Selenium

Lab Book Version 1.0

Document Revision History

Date	Revision No.	Author	Summary of Changes
Jan 2016	1.1	Ritika Verma & Shubhasmit Gupta – Automation CoE	New contents creation

Table of Contents

Document Revision History 2
Table of Contents 3
Getting Started 4
Lab Demo 1: Example (Basic Selenium IDE Flow) 5
Lab Demo: Example (Selenium IDE) (Cont.…) 6
Lab Demo: Example (Selenium IDE) (Cont.…) 7
Lab Demo 2: Learning Selenium IDE (Modifications) 8
Lab 1. Learning Selenium IDE(Basic Flow) 9
Lab 2. Learning Selenium IDE(Performing Validations) 10
Lab 3. Create a new account (using Selenium Webdriver) 11
Lab 4. Validations in Selenium (using Selenium Webdriver) 13
Lab 5. Alert and window handling (using Selenium Webdriver) 14
Lab 6. WebDriver with JUnit/TestNG (using Selenium Webdriver with Junit and TestNG) 15
Lab 7. Advance Selenium (Chrome Driver and IE Driver) 16
Lab 8. Advance Selenium (RemoteWebDriver) 17
Appendices 18
Appendix A: Selenium Standards 18

Getting Started

Overview

This lab book is a guided tour for learning Test Automation & Advanced Selenium. It comprises solved examples and 'To Do' assignments. Follow the steps provided in the solved examples and work out the 'To Do' assignments given.

Setup Checklist for Selenium

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 512MB of RAM
- Internet Explorer 6.0 or higher
- Mozilla Firefox(Add-ons: Firebug, Firepath, Selenium IDE)

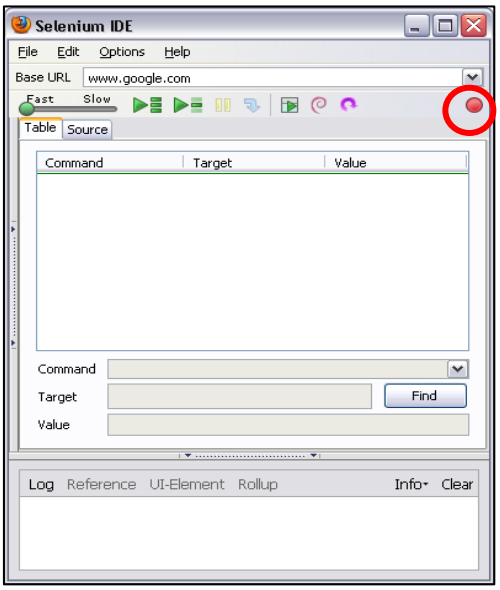
Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory Selenium_Assign. For each lab exercise create a directory as lab <lab number>.

Lab Demo 1: Example (Basic Selenium IDE Flow)

Goals	• Understand the process of automation testing of a web application on Selenium IDE • Learn to manage document spacing
Time	60 minutes

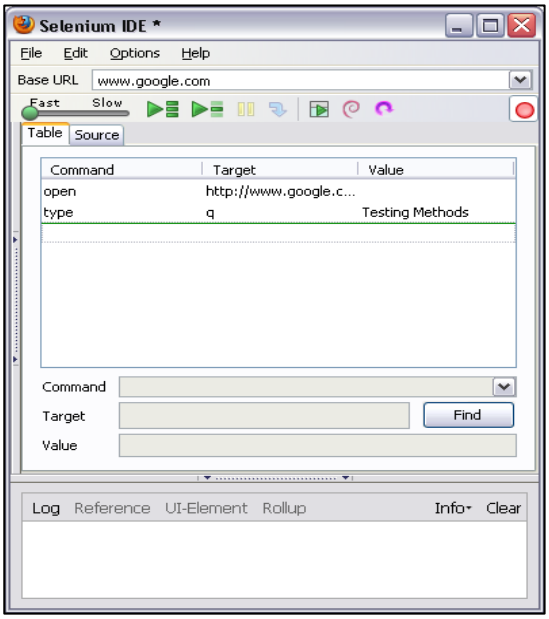
Go to the Web Page for which you want to carry out the test



Hit the record button on IDE

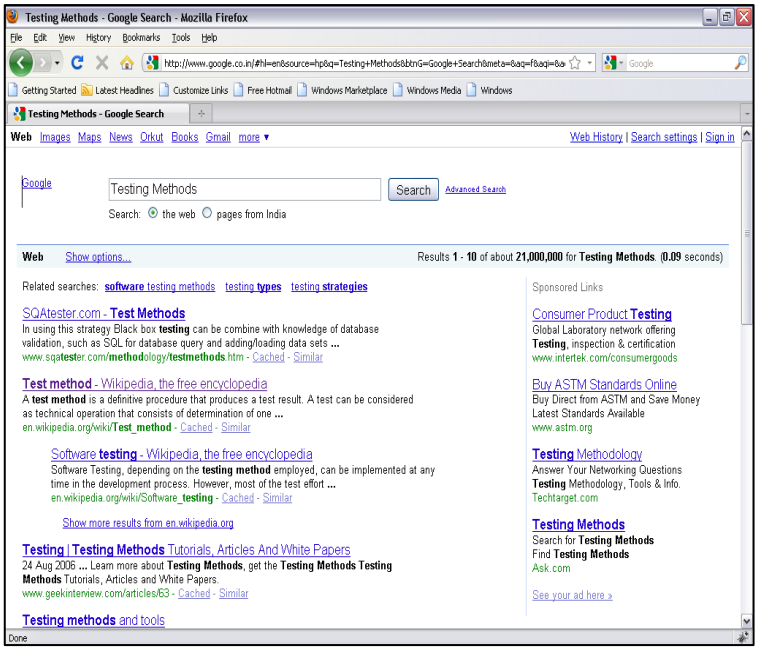
Lab Demo: Example (Selenium IDE) (Cont.)

Enter the text on Web Page and submit

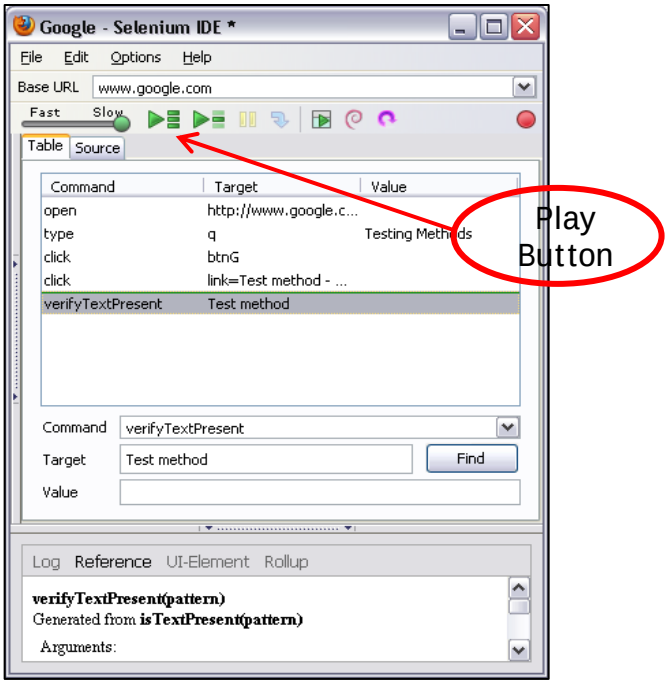


IDE should be updated, stop the recorder and add the assertions

Lab Demo: Example (Selenium IDE) (Cont.)



Perform operations on the web page



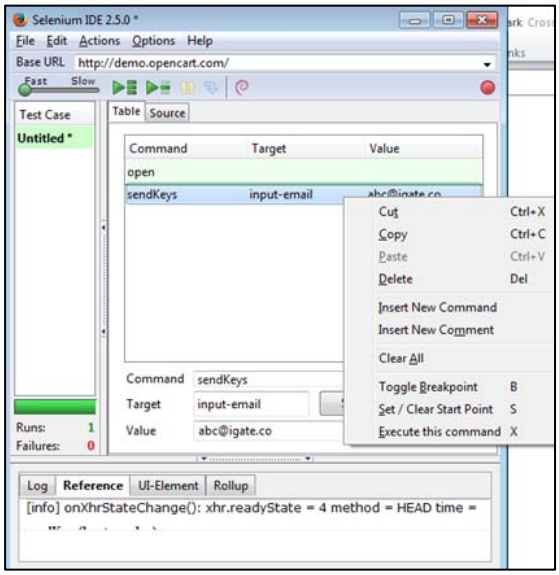
Hit the play button to play the recorded scripts

Lab Demo 2: Learning Selenium IDE (Modifications)

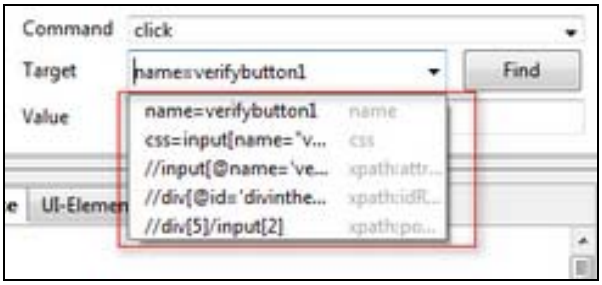
Goals	• Understand the process of further modifications and validations in automation testing of a web application on Selenium IDE • Learn to manage document spacing
Time	60 minutes

Basic URL: http://demo.opencart.com/

1. Follow the steps from Lab Demo 1
2. In order to add new command, go to the command where the new command needs to be added and then do right click , click on 'Insert New Command'



3. 'Command' dropdown provides options/ keywords to perform the operation

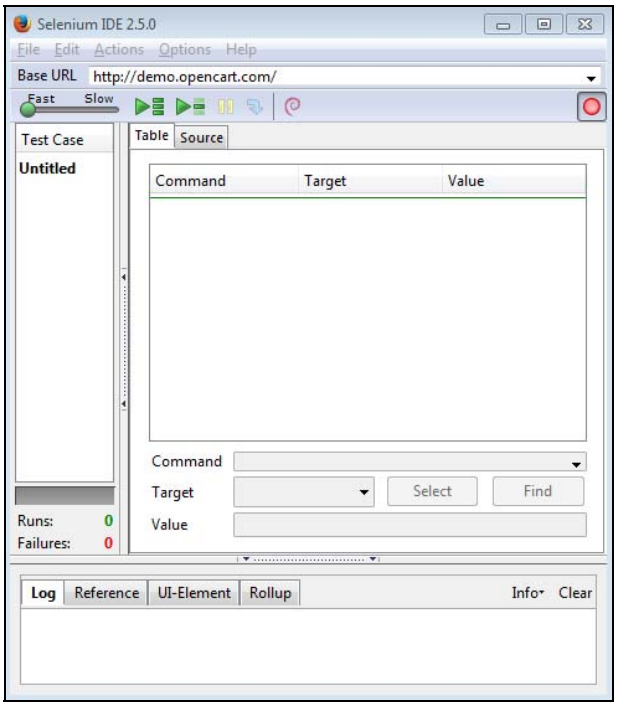


Lab 1. Learning Selenium IDE(Basic Flow)

Goals	• Understand the basic process of automation testing of a web application in Selenium IDE • Learn to manage document spacing
Time	60 minutes

Basic URL: http://demo.opencart.com/

1. Open the URL on Firefox
2. Start recording on Selenium IDE
3. Go to 'Desktops' tab
4. Click on 'Mac'
5. Select 'Name(A-Z)' from the 'Sort By' dropdown
6. Click on 'Add to Cart' button
7. Stop the recording on Selenium IDE
8. Playback the whole test case



Lab 2. Learning Selenium IDE(Performing Validations)

Goals	• Understand the process of automation testing of a web application • Learn to manage document spacing
Time	60 minutes

Basic URL: http://demo.opencart.com/

1. Open the URL on Firefox
2. Start recording on Selenium IDE
3. Verify title of the page
4. Go to 'Desktops' tab
5. Click on 'Mac'
6. Select 'Name(A-Z)' from the 'Sort By' dropdown
7. Click on 'Add to Cart' button
8. Enter ‘Mobile’ in ‘Search’ text box and click on ‘Search’ button
9. Wait for page to load
10. Clear the text from ‘Search Criteria’ text box
11. Click on ‘Search in product descriptions’ check box and click on ‘Search’ button
12. Stop the recording on Selenium IDE
13. Add the step after Step 5 where verify the 'Mac' heading
14. Change the value from ‘Mobile’ to ‘Monitors’
15. Save the test case
16. Playback the whole test case.
17. Playback set by step
18. Add the step after Click on 'Mac' where verify the 'Mac' heading
19. Save the test case
20. Export the test case as 'Java/JUnit/webDriver'
21. Run the whole Test case and check the 'Pass' Status
22. Create another test case with the same flow
23. Create the test suite for the above test cases

Lab 3. Create a new account (using Selenium Webdriver)

Goals	• Understanding the scenario from end to end and automating the same • Analyze the requirement and perform validations accordingly
Time	120 minutes

Basic URL: http://demo.opencart.com/

Please ensure that the variables have to be defined before it is being used.

Part 1: Launch Application

1. Launch the URL on Firefox
2. Verify 'Title' of the page
3. Click on 'My Account' dropdown
4. Select 'Register' from dropdown
5. 'Register Account' page will open up, verify the heading 'Register Account'
6. Click on 'Continue' button at the bottom of the page
7. Verify warning message 'Warning: You must agree to the Privacy Policy!'

Automate and validate the different sections of 'Register Account' page:

Part 2: For 'Your Personal Details'

1. Enter data in 'First Name' text box
2. Verify if 33 characters can be entered in 'First Name' text box by clicking on 'Continue' button.
3. If not, verify error message.
4. Enter data in 'Last Name' text box
5. Verify if 33 characters can be entered in 'First Name' text box by clicking on 'Continue' button.
6. If not, verify error message.
7. Enter valid 'E-mail'.
8. Enter 'Telephone' which must be between 3 and 32 characters.

Part 3: For 'Your Address'

1. Enter 'Address 1' which should contain characters between 3 and 128

2. Enter 'City' which should contain characters between 2 and 128
3. Enter 'Post Code' which should contain characters between 2 and 10
4. Select 'India' from 'Country' Dropdown
5. Select 'Region/State' from dropdown

Part 4: For 'Password'

1. Enter 'Password' which must be between 4 and 20 characters.
2. Enter 'Password Confirm'.

Part 4: For 'Newsletter'

1. Click on 'Yes' Radio button
2. Click on checkbox for 'I have read and agree to the Privacy Policy'.
3. Click on 'Continue' button
4. Verify message 'Your Account Has Been Created!'
5. Click on 'Continue'
6. Click on link 'View your order history' under 'My Orders'

Lab 4. Validations in Selenium (using Selenium Webdriver)

Goals	• Understanding the scenario from end to end and automating the same
Time	120 minutes

Basic URL: http://demo.opencart.com/

After login on the 'Open Cart', create a test script following the below mentioned steps:

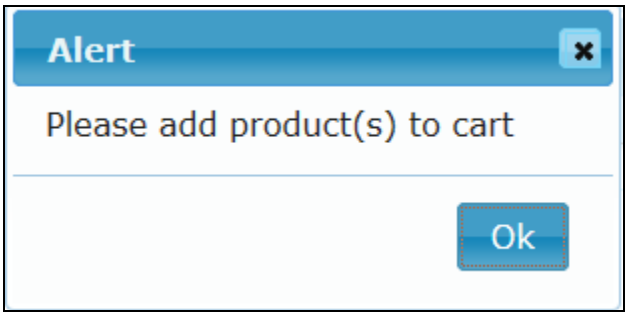
1. Login with credentials created in Lab 1
2. Go to 'Components' tab and click
3. Select 'Monitors'
4. Select 25 from 'Show' dropdown
5. Click on 'Add to cart' for the first item
6. Click on 'Specification' tab
7. Verify details present on the page
8. Click on 'Add to Wish list' button.
9. Verify message 'Success: You have added Apple Cinema 30" to your wish list!'
10. Enter 'Mobile' in ' Search' text box.
11. Click on 'Search' button
12. Click on 'Search in product descriptions' check box
13. Click on link 'HTC Touch HD' for the mobile 'HTC Touch HD'
14. Clear '1' from 'Qty' and enter '3'
15. Click on 'Add to Cart' button
16. Verify success message 'Success: You have added HTC Touch HD to your shopping cart!'
17. Click on 'View cart' button adjacent to search button
18. Verify Mobile name added to the cart
19. Click on 'Checkout' button
20. Click on 'My Account' dropdown
21. Select 'Logout' from dropdown
22. Verify 'Account Logout' heading
23. Click on 'Continue'

Lab 5. Alert and window handling (using Selenium Webdriver)

Goals	Learning alert handling and window handling basics in selenium webdriver
Time	120 minutes

Base URL: https://ispace.ig.capgemini.com/sitepages/index.aspx

1. Launch the URL
2. Go to 'Application' tab
3. Click on checkbox 'Stationery Request'
4. Verify the new title ‘Stationary’
5. On ‘Stationery’ tab, click on 'Submit to collect your Stationery >>>' link
6. Switch to alert as shown below.
7. Verify text on the alert 'Please add product(s) to cart'
8. Verify if 'Ok' button is present on the alert
9. Click on 'Ok' button
10. Click on ‘Photocopy’ tab, click on ‘Save Request’ button
11. Switch to alert and verify text ‘No changes made’
12. Verify if ‘Ok’ button is present
13. Click on ‘Ok’ button present on alert
14. Click on ‘Logout’ button
15. Close the ‘Stationery’ window



Lab 6. WebDriver with JUnit/TestNG (using Selenium Webdriver with Junit and TestNG)

Goals	Learning how to write webdriver automation testcases using Junit and TestNG
Time	120 minutes

1. Consider the flow mentioned in Lab 3 & Lab 4, complete the task using Selenium WebDriver and JUnit.
   [**NOTE:** All the verifications should be using JUnit Assertions]

2. Consider the flow mentioned in Lab 3 & Lab 4, complete the task using Selenium WebDriver and TestNG.
   [**NOTE:** All the verifications should be using TestNG Assertions. Testcases should have proper TestNG Reporter logging as well.]

3. Create one TestNG test suite for both the testcases created for question number 2 along with **testing.xml** and execute the test suite. Provide the **Reports** as well.

Lab 7. Advance Selenium (Chrome Driver and IE Driver)

Goals	Learning how to write webdriver automation testcases for Chrome and IE browser
Time	120 minutes

1. Consider the flow mentioned in Lab 3 & Lab 4, make necessary changes to execute the same flow on both Chrome and Internet Explorer browser.

2. Consider the Question number 1 and make it JUnit test case.
[**NOTE:** All the verifications should be using JUnit Assertions]

Lab 8. Advance Selenium (RemoteWebDriver)

Goals	Learning how execute Selenium scripts using RemoteWebDriver
Time	120 minutes

1. Consider the flow mentioned in Lab 3 & Lab 4, make necessary changes to execute the same flow using RemoteWebDriver and Selenium Grid on Firefox, Chrome and Internet Explorer browser.
   [NOTE: Set Platform, BrowserName and Version to DesiredCapabilities]

2. Consider the flow mentioned in Lab 4 and take screenshot after each steps. Save all the screenshots inside a folder called '**Screenshots**' in the root of the Java project.

Appendices

Appendix A: Selenium Standards

Key points to keep in mind:

Selenium standards help you reach the widest possible audience.
There are many technologies that are associated with HTML because they are used on web pages or in conjunction with HTML.

For each of the above, please follow coding conventions specified by that technology.

Sometimes you are going to have to break rules and use non-standard syntax for good reason. Try to keep this to the minimum.

How to follow Selenium standards:

See the W3C site for more information on locator types and keyword statements.

The important thing to remember is that proper keywords are essential to assist validation software in checking your document.

Use Selenium IDE to find the locators for the elements.

Refer to W3C for technical and syntax information.

Always use the comments with the codes for proper understanding.

Some simple Selenium standards:

Names of Selenium files should always end with the extension *.java*.

Correct: foo.java

Incorrect: foo.bar

