

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 (Fittesse)
- CI Tools (Jenkins etc)



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.

8.1: Selenium Frameworks

Data Driven

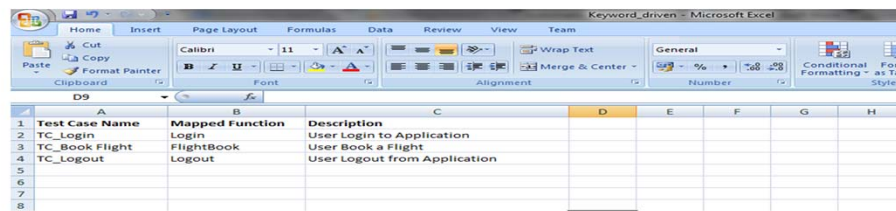
- Data driven is the design of possible inputs what may given by the end user. This would cover maximum probabilities of an input data.
- It can be Spread sheet or sql. We have to connect and pass the values to the respective field or element.
- In this type of framework Data is not hardcoded with script ,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	passwd		
3	TC_YahooMail	www.yahoo.com	abc@yahoo.com	passwd	Good Morning	
4	TC_Facebook	www.facebook.com	abc@somemail.com	passwd		abc def
5						
6						
7						
8	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.		
9						
10						

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



	A	B	C	D	E	F	G	H
1	Test Case Name	Mapped Function	Description					
2	TC_Login	Login	User Login to Application					
3	TC_Book Flight	FlightBook	User Book a Flight					
4	TC_Logout	Logout	User Logout from Application					
5								
6								
7								
8								



Copyright © Capgemini 2015. All Rights Reserved 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 too.

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.

9.1: Selenium Frameworks

Component Based (Sprintest®/CBF)

- Component Based is an automatic test which is made up of components.
- It allows you to create automatic tests.
- Capgemini Test Automation Framework CBF is Component Based Test Automation solution, aimed at optimizing test automation for business critical applications.
- CBF is designed to help test analysts and engineers build and automate tests using such abstraction, thus fine-tuning the test engineering process to the application under test (AUT) and minimizing the tedium of detailing each test step in each test one by one.

Framework Architecture

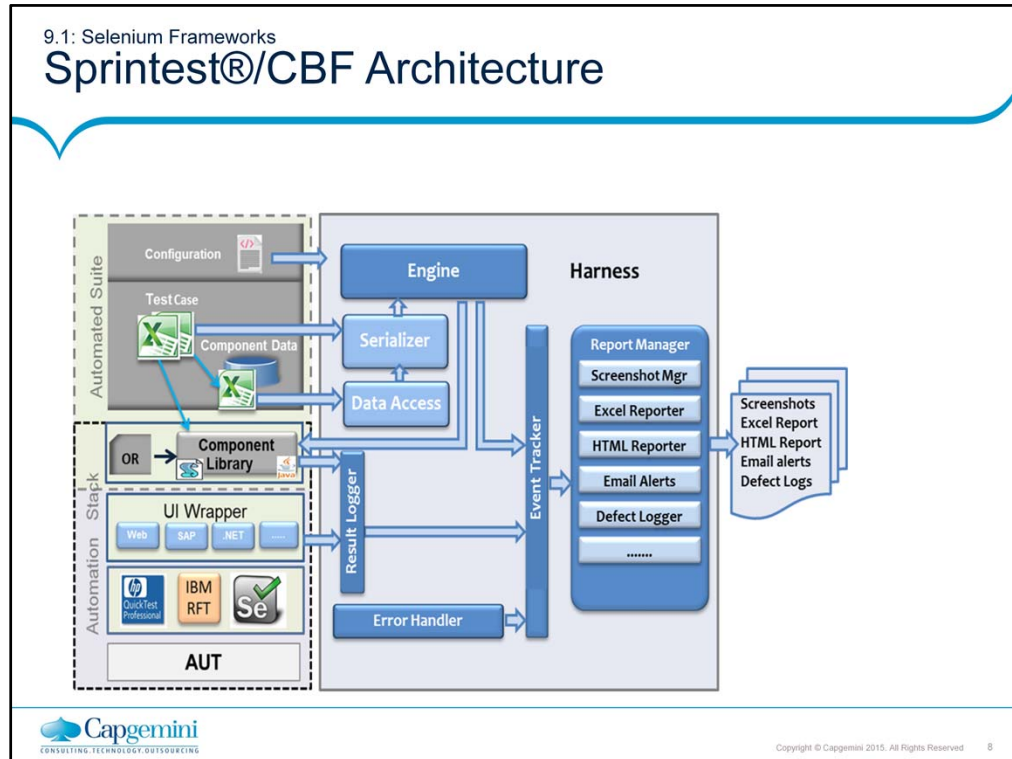
In consideration to the richness and diversity of UI of the applications and nature of test cases in scope, we are using a Modular Based Hybrid Automation Framework for the Automation project.



Copyright © Capgemini 2015. All Rights Reserved 7

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.

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

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.

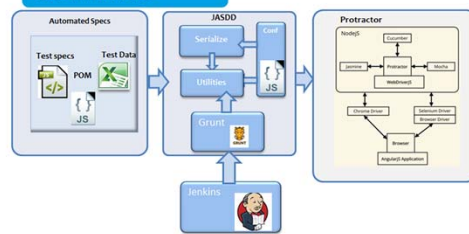
8.1: Selenium Frameworks

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



Copyright © Capgemini 2015. All Rights Reserved 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.

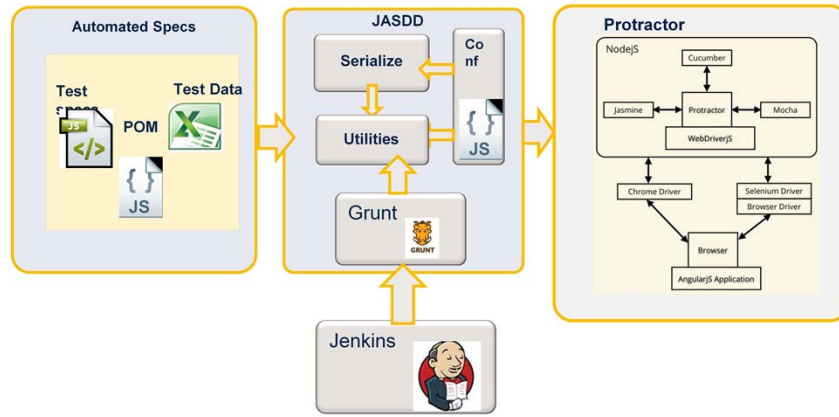
java (IDE exported in Java)

rb (IDE exported in Ruby)

py (IDE exported in Python)

cs (IDE exported in C#)

8.1: Selenium Web Driver – Advance

Sprintest®/Quadrant Architecture

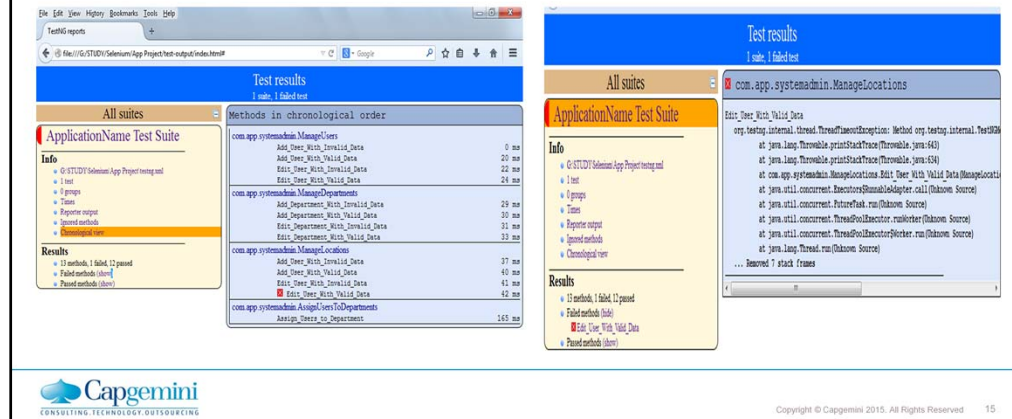
8.1: Selenium Frameworks

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.

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



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

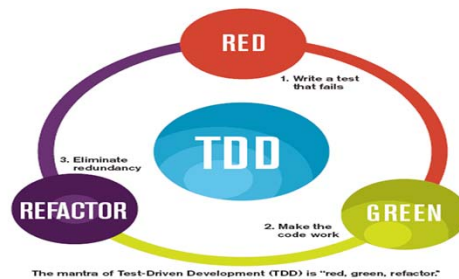
The Reference tab is the default selection whenever you are entering or modifying [Selenese](#) commands and parameters in Table mode. In Table mode, the Reference pane will display documentation on the current command. When entering or modifying commands, whether from Table or Source mode, it is critically 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.

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.



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.

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.

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 like C++, Python, Ruby, Delphi, C#, etc.
- Different Principles of fitNesse is described in the next page.

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.

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

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.

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
```

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).

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 -p X
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
```


8.1: Selenium Frameworks

Continued.

Automate!

- Automate scenarios to establish a continuously validated living documentation.

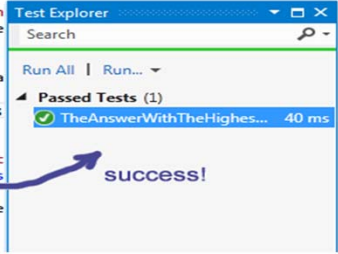
```
[Binding]
public class OrderingAnswersSteps
{
    private readonly QuestionPage questionPage;

    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
}
}
```



Test Explorer

Search

Run All | Run...

Passed Tests (1)

✓ TheAnswerWithTheHighes... 40 ms

success!

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.

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.

Summary

- In this lesson, you have learnt

- In this lesson, you have understood the different ways in the data is driven from the framework.
- In the Data Driven Automation Framework, the Input Data/Input Parameters are not hard-coded in the test scripts. Instead, these are stored and passed from external files/resources such as Microsoft Excel Spreadsheets, Microsoft Access Tables, SQL Tables and XML files etc.
- In the Keyword Driven Automation Framework, we can create multiple keywords that allow testers to associate a unique action or function for each of these keywords
- After execution of the test script, it is necessary to get the results of the execution. The reports are customized in the framework such that the summary report and the detailed report are stored in html format.
- Test-driven development does not perform sufficient testing in situations where full functional tests are required to determine success or failure, due to extensive use of unit tests.



Copyright © Capgemini 2015. All Rights Reserved 28

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.