Selenium – Day 3

November 2016

TEKSTROM
Nurturing quality. Cultivating talent.

Learning Objectives– Day 3

- ❖DAY 2 - Recap
- ❖Object Synchronization
- ❖Exception Handling
- ❖Introduction to Reporting Frameworks – TestNG, JUNIT
- ❖Introduction to Logging Framework - log4j
- ❖Test Data Management/Parameterization (using excel and xml)
- ❖Exercise : Enhance tests for diverse data and application behavior. Read data from excel and xml to input into application

TEKSTROM
Nurturing quality. Cultivating talent.

Object Synchronization

It is a mechanism which involves more than one components to work parallel with Each other.

Generally in Test Automation, we have two components

- Application Under Test
- Test Automation Tool.

Synchronization has 2 variants

- Unconditional
- Conditional Synchronization

Unconditional Synchronization

Specify timeout value. Selenium will wait until certain amount of time and then proceed further.

Examples: Wait() and Thread.Sleep();

Disadvantage:

Waiting time is fixed, though the application is ready script waits.

Advantage:

Where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition. In this situations, make the application to wait for certain amount of time by specifying the timeout value.

Conditional Synchronization

Specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.

In Selenium, implicit Wait, Explicit Wait and Fluent Wait are supported

Conditional Synchronization – Implicit Wait

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.

Implicit wait will not work all the commands/statements in the application. It will work only for "FindElement" and "FindElements" statements.

Syntax:

driver.manage.TimeOuts.implicitwait(6,Timeunit.SECONDS);

Example using implicit timeout

WebDriver driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://www.google.com");

Conditional Synchronization – Explicit Wait

Defines a wait statement for certain condition to be satisfied until the specified timeout period

If the Webdriver finds the element within the timeout period the code will get executed

Explicit wait is mostly used when we need to Wait for a specific content/attribute change after performing any action

Example:

When application gives AJAX call to system and get dynamic data and render on UI

- Drop-downs Country and State, based on the country value selected, the values in the state drop-down will change, which will take few seconds of time to get the data based on user selection.

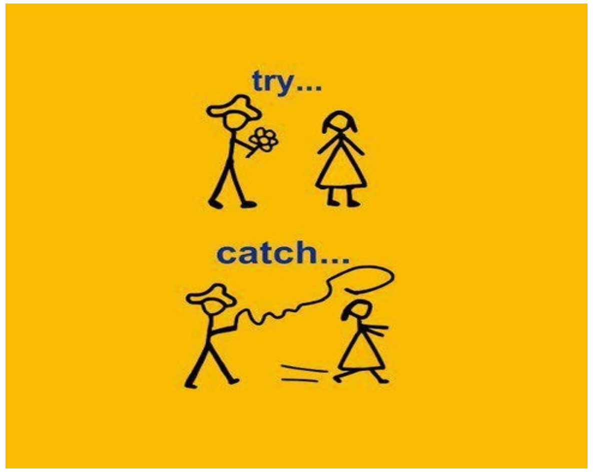
```
/*Explicit wait for state dropdown field*/
    WebDriverWait wait = new WebDriverWait(driver, 10);
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("statedropdown")));
```

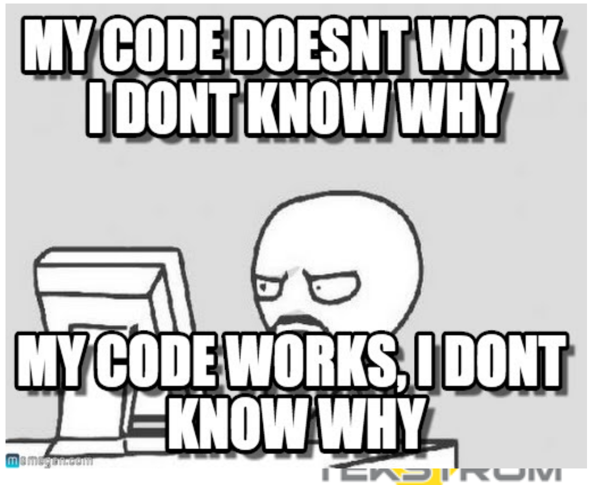
Exception Handling

What is Exception?

Any event during script execution that causes changes to application/script work flow **OR** Abruptly stops script execution

Exceptions are those which can be handled at the run time




MY CODE DOESNT WORK I DONT KNOW WHY
MY CODE WORKS, I DONT KNOW WHY

Exception Handling Triggered

Exceptions triggered by the script:

NoSuchElement exceptions – Whenever the element is not found in DOM, you will get this exception

Timeout exceptions – Object/Page not loaded in given time.

```
/*Explicit wait for state dropdown field*/
WebDriverWait wait = new WebDriverWait(driver, 10);
 wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("elementid")));
```

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds

File exceptions – File not present or no permissions

Null Pointer exceptions – Object reference is not available

Exception Handling – try example

Example:

```
try{
        int x=0;
        int y=10;
        System.out.println(y/x);
}
```

catch: catch block contains handling code if any exception occurs in try block. try must follow catch block. try after catch or finally is mandatory.

Syntax:

```
      catch(Exception e){
            statements....//contains handling code
      }
```

Exception Handling – try catch example

Example:

```
try{
        int x=0;
        int y=10;
        System.out.println(y/x);
}

catch(Exception e){
        System.out.println("Exception has been handled" + e);
}

System.out.println("code after try catch block");
```

When an exception raises it creates the exception object e.
This object contains name of the exception class, cause and location of exception

Exception Handling – Display

Exception Information displaying methods are:

printStackTrace(): prints the stack trace , exception name and description

toString(): returns a text message describing the exception name and description

getMessage(): displays the description of exception

These three methods are present in Throwable class.

Exception Handling – Display example

Example:

```
package com.seleniumeasy.ExceptionHandling;
public class ExceptionMethods {
        public static void main(String[] args) {
        try{
                int x=0;
                int y=10;
                System.out.println(y/x);
        }
        catch(ArithmeticException ae){
                ae.printStackTrace();
                System.out.println(ae.toString());
                System.out.println(ae.getMessage());
        }
        }
}
```

Output:

```
java.lang.ArithmeticException: / by zero
java.lang.ArithmeticException: / by zero
/ by zero at com.seleniumeasy.ExceptionHandling.ExceptionMethods.main(ExceptionMethods.java:9)
```

Reporting

What is the purpose of reporting?

- Serves as health card
- Actual state of the product
- Red/Green status

What are the essential qualities of a good test report?

Brevity –

- A report should be short and concise.
- It should clearly state the proportion of total no. of success or failure.

Traceability –

- Captures all the footprints that could lead to the root cause of a failure.

Selenium Webdriver doesn’t have a built-in reporting feature

Reporting Frameworks

Traceability –

- It must provide the ability to review the following.
    - Historical data for test cases and failures.
    - Age of a particular failure.

Sharable –

- It should support a format which you can share through email or integrate with CI tools like Jenkins/Bamboo.

Test coverage –

- It should highlight the test coverage for the following.
    - Test coverage of the module under test.
- Test coverage of the application under test.

TEKSTROM
Nurturing quality. Cultivating talent.

Reporting - TestNG

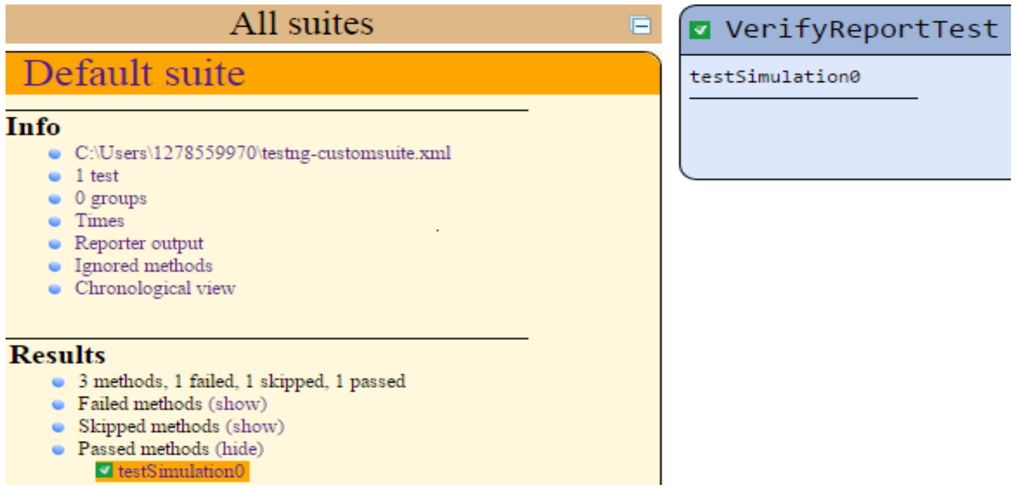
- TestNG (NG - Next Generation) is a open source automation testing framework inspired from JUnit and Nunit.
- However it introduces new/more functionalities that make it more powerful and easier to use.
- TestNG is designed to cover all categories of tests – unit, functional, end-to-end, integration, etc.

- GENERATE REPORTS USING TESTNG

- TestNG library brings a very convenient reporting feature.

- Html report combines the detailed information like the errors, test groups, execution time, step-by-step logs and TestNG XML file.

- Summary report is the trimmed version and informs about the test pass/fail/skip count.



All suites

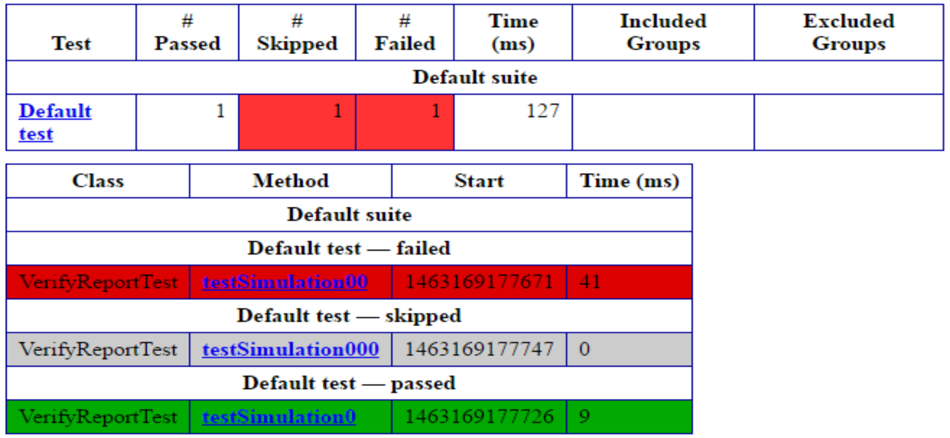
Default suite

Info
- C:\Users\1278559970\testng-customsuite.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

Results
- 3 methods, 1 failed, 1 skipped, 1 passed
- Failed methods (show)
- Skipped methods (show)
- Passed methods (hide)
  - ✓ testSimulation0

✓ VerifyReportTest

testSimulation0



Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Default suite						
Default test	1	1	1	127		

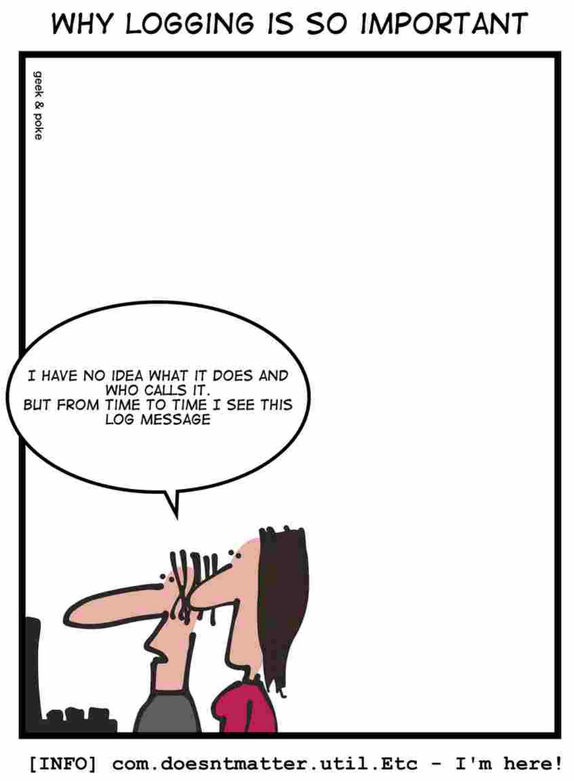
Class	Method	Start	Time (ms)
Default suite			
Default test — failed			
VerifyReportTest	testSimulation00	1463169177671	41
Default test — skipped			
VerifyReportTest	testSimulation000	1463169177747	0
Default test — passed			
VerifyReportTest	testSimulation0	1463169177726	9

TEKSTROM
Nurturing quality. Cultivating talent.

Logging

Sometimes logging is considered to be an overhead upon the existing script creation mechanism but it is one of the best practices if used in the accurate proportion because of the following advantages

- Provides a greater understanding of test execution
- Can be stored externally for post execution analysis
- Helpful in debugging test failures


WHY LOGGING IS SO IMPORTANT
geek & poke
I HAVE NO IDEA WHAT IT DOES AND WHO CALLS IT. BUT FROM TIME TO TIME I SEE THIS LOG MESSAGE

```
[INFO] com.doesntmatter.util.Etc - I'm here!
```

TEKSTROM Nurturing quality. Cultivating talent.

Logging – log4j

Popular logging framework and widely used - log4j

- Log4j is a fast, flexible and reliable logging framework (APIS)
- It is distributed under the Apache Software License.
- Log4J has been ported to the C, C++, C#, Perl, Python, Ruby and Eiffel Languages.
- This is popularly used in small to large scale Selenium Automation projects.

Log4j has mainly 3 components. These components represent

- details about the log level
- formats of the log message in which they would be rendered
- saving mechanisms.

Components of Log4j:

- Loggers - capturing logging information
- Appenders - publishing logging information to various preferred destinations.
- Layouts - formatting logging information in different styles

TEKSTROM Nurturing quality. Cultivating talent.

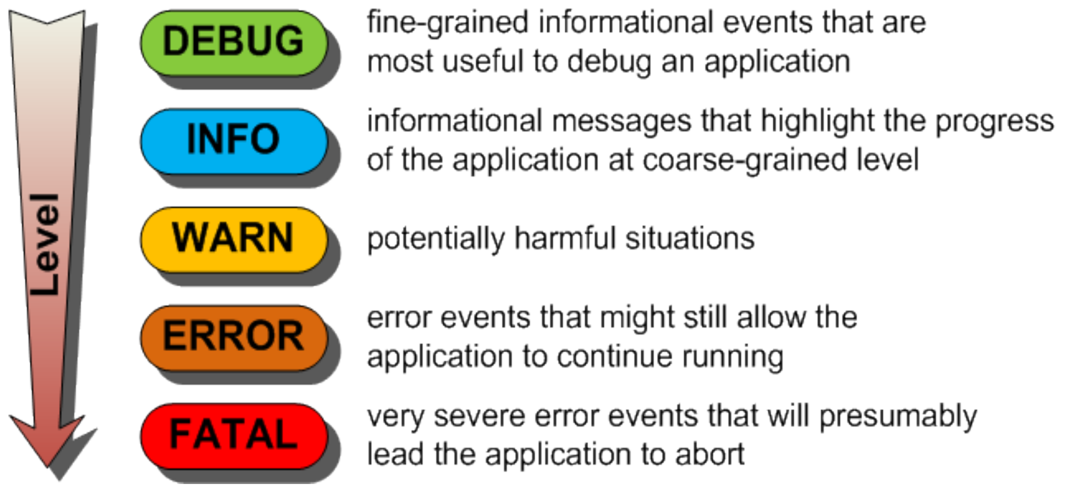
Log4j - Loggers

Loggers: The following steps need to done in order to implement loggers in the project.

1. Creating an instance of Logger class
2. Defining the log level

- Logger Class – It is a java based utility that has got all the generic methods already implemented so that we are enabled to use log4j.

- Log levels –These are used for printing the log messages. There are primarily five kinds of log levels.


Level
DEBUG
fine-grained informational events that are most useful to debug an application
INFO
informational messages that highlight the progress of the application at coarse-grained level
WARN
potentially harmful situations
ERROR
error events that might still allow the application to continue running
FATAL
very severe error events that will presumably lead the application to abort

TEKSTROM Nurturing quality. Cultivating talent.

Log4j - Architecture

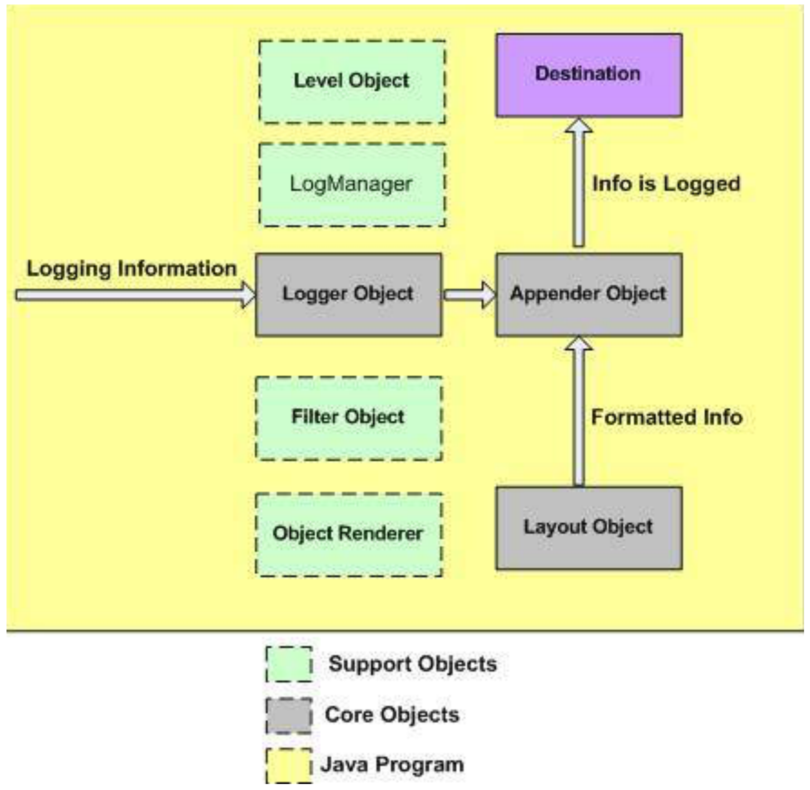
log4j API follows a layered architecture

Each layer provides different objects to perform different tasks

Layered architecture makes the design flexible and easy to extend in future

There are two types of objects available with log4j framework

- Core Objects: These are mandatory objects of the framework. They are required to use the framework
- Support Objects: These are optional objects of the framework. They support core objects to perform additional but important tasks



Log4j – Log Manager

Log manager reads the configuration and sets up the logging appropriately.

log4j.properties is the configuration file – works on key-pair method

<u>log4j.properties syntax</u>

Following is the syntax of log4j.properties file for an appender X:

```
# Define the root logger with appender X
log4j.rootLogger = DEBUG, X

# Set the appender named X to be a File appender
log4j.appender.X=org.apache.log4j.FileAppender

# Define the layout for X appender
log4j.appender.X.layout=org.apache.log4j.PatternLayout
log4j.appender.X.layout.conversionPattern=%m%n
```

Log4j – Appenders

log4j provides Appender objects which are primarily responsible for printing logging messages to different destinations such as consoles, files, sockets, NT event logs, etc.

Each Appender object has different properties associated with it, and these properties indicate the behavior of that object

Other Appenders are AsyncAppender, JMSAppender, JDBCAppender etc.

Property	Description
layout	Appender uses the Layout objects and the conversion pattern associated with them to format the logging information.
target	The target may be a console, a file, or another item depending on the appender.
level	The level is required to control the filtration of the log messages.
threshold	Appender can have a threshold level associated with it independent of the logger level. The Appender ignores any logging messages that have a level lower than the threshold level.
filter	The Filter objects can analyze logging information beyond level matching and decide whether logging requests should be handled by a particular Appender or ignored.

Log4j – Layout

PatternLayout is used with Appender. Possible options are:

- DateLayout
- HTMLLayout
- PatternLayout
- SimpleLayout
- XMLLayout

Using HTMLLayout and XMLLayout, you can generate log in HTML and in XML format as well

Test Data Management - Parameterization

Parameterization makes testing the application more easy

Multiple data to the application at runtime can be passed through parameterization

Concept of parameterization can be achieved by Data driven testing

Selenium Webdriver doesn't have built in structure or method to parameterize your test

We shall discuss 2 variants of parameterization
- Excel, TestNG/XML

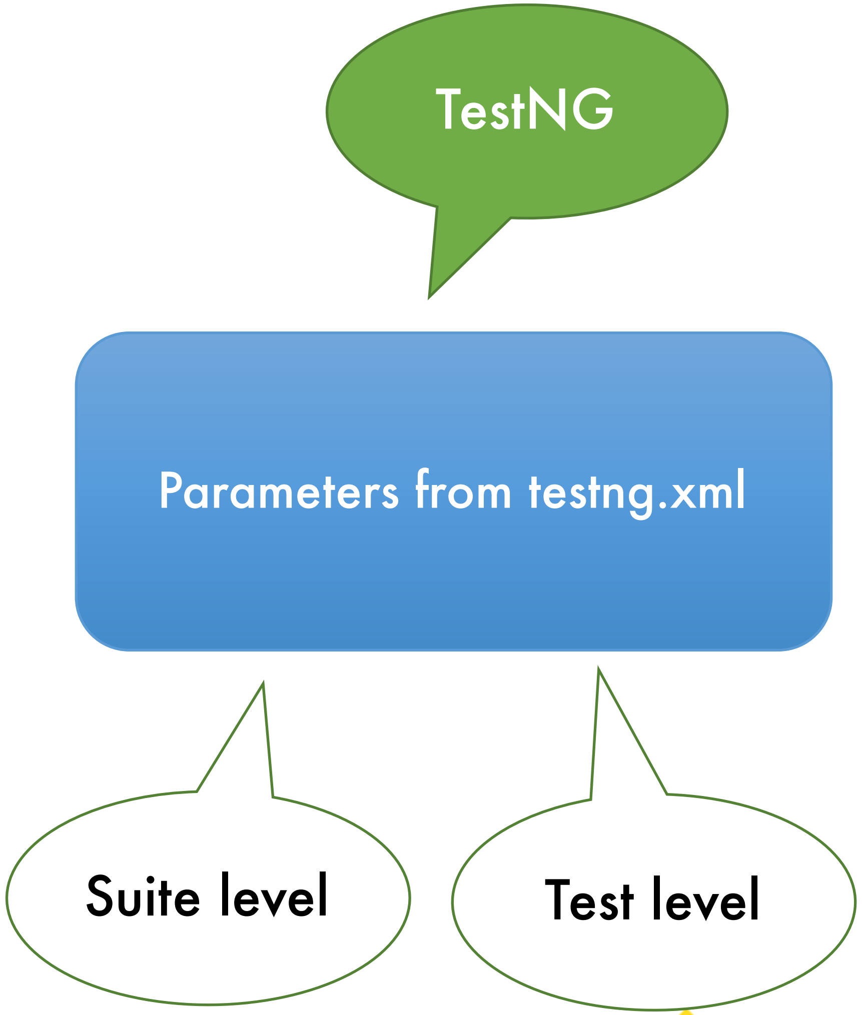
Parameterization using xml

Parameterization makes testing the application more easy

Multiple data to the application at runtime can be passed through parameterization

Concept of parameterization can be achieved by Data driven testing

TestNG can be used for parameterization
Using Parameters annotation in TestNG XML file
@Parameters({"name", "search key"})



Exercise

- Enhance tests for diverse data and application behavior

Day 3 - Summary

Today we learnt :

- ❖Object Synchronization - Explicit & Implicit Waits
- ❖Exception Handling
- ❖Reporting Frameworks likeTestNG, JUNIT, log4j
- ❖Test Data Management and Parameterization using excel and xml