# TestNG

TestNG is a unit testing framework

**What are advantages we get when we use it with selenium?**

- Pass or Fail result for Test
- Test Batch Execution
- Pre and Post conditions for the test(Specifying the dependencies)
- Parameterization using xml file
- Test parallel execution (We use this concept for GRID)
- Data Driving using DataProviders
- Assertions (We can verifications using this)
- Reports
- Listeners to extend the way TestNG generate reports

**Install TestNG plugin in Eclipse:**

- Help -> Eclipse Market Place -> Search for TestNG -> Click on Install at TestNG -> Next -> Accept Terms -> Confirm -> Restart eclipse after installation
- Configure TestNG For Project: BuildPath -> Libraries tab -> Click on AddLibrary -> TestNG -> OK

**Staring With TestNG**

- TestNG executes the methods based on annotations
- @Test : A method with @Test is An Actual Test Method in TestNG class.
    - We will write main code here.
    - We can have any number test methods in one TestNG class.
    - These methods will be executed in alphabetic order.
    - We can prioritize them by specifying priority
- Create TestNG class
    - Right click on project -> new-> TestNG class
    - This will create a class with @Test Method
    - You can rename it
- Syntax

```
import org.testng.annotations.Test;

public class Test1 {
  @Test
  public void t1() {
        System.out.println("this is t1 method");
  }
}
```

- Above class is having a method with no priority
- We will not be having main method in testng class
- We don't need to call @Test methods for execution
- By default the methods will be executed based on annotations

- After execution observe that the status is displayed for only test methods
- By default every test method will have zero priority
- The methods with no priority will be executed first
- We can set priority using @Test(priority=1)

**Specifying Pre and Post Conditions**

- We can execute pre and post conditions
    - before/after executing class
    - before/after executing each test method
- @BeforeClass : A method with this annotation will be executed before executing the class
- @AfterClass : A method with this annotation will be executed after executing the class
- @BeforeMethod : A method with this annotation will be executed before executing the each @test method
- @AfterMethod : A method with this annotation will be executed after executing the each @Test method
- We can also use dependsonmethods to execute a method before executing current method
- If depends on method is failed then it is not going to execute current method
- you can use alwaysrun=true to force a method to execute even if depends on method is failed
- @BeforeSuite : A method with this annotation will be executed before executing the xml suite
- @AfterSuite : A method with this annotation will be executed after executing the xml suite
- @BeforeTest : A method with this annotation will be executed before executing the xml test
- @AfterTest : A method with this annotation will be executed after executing the xml test
- These BeforeSuite/Test AfterSuite/Test methods can be there in any class of that suite/test in xml file

**Grouping Tests**

- We can group test methods by specifying a group name
- We can specify like @Test(groups="A")
- We can specify multiple group name @Test(groups={"A","B"})
- We can able to execute the methods based on groups by including these groups in xml file

```
<groups>
	<run>
		<include name="A" />
	</run>
</groups>
```

- We can specify pre and post condition for the groups using @BeforeGroups and @AfterGroups
- While developing these pre and post conditions you must specify for which group names you wanted to execute these pre and post conditions
    - @BeforeGroups(groups="A")
    - @BeforeGroups(groups={"A","B"})

**Parameterization**

- We can run TestNG methods with the data created in xml file
- We have to use Parameters concept to achieve this
- Create Parameters in TestNG Xml file
    - If we create parameters under suite those can be accessible to any test of that suite
    - If we create parameters under test those can be accessible to the same test
- To create parameters we have to use parameters tag

```
<parameters>

    <parameter name="browser" value="edge" />

</parameters>
```

- We can create this under suite/test in xml
- If we create under suite then these parameters will available to all tests
- If we create under test then these parameters will available only to that test
- To configure the method in class to use these parameters we have to use @Parameters annotation
    - @Parameters({"browser"})
- We need to write this annotation above method definition
- This means that we are instructing TestNg to execute that method by passing browser parameter value specified in xml file
- So TestNG calls that method with a parameter value. We have to create a parameter to that method and configure that parameter with a static value in that method.

**Final code in Class File**

```
public class OhrmAddEmployee_Parameters {
	WebDriver driver;
	@BeforeClass
	@Parameters({"browser"})
	public void OpenApplication(String brName) {
		driver = WebUtils.getDriver(brName);
		WebUtils.openApplication(driver,
"http://opensource.demo.orangehrmlive.com/");

	}

	@Test
	public void Login() {

	driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("admin");

	driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("admin");
		driver.findElement(By.xpath("//input[@value='LOGIN']")).click();
	}

	@Test(dependsOnMethods="Login")
	public void AddEmployee() {
		driver.findElement(By.xpath("//a[.='PIM']")).click();
		driver.findElement(By.xpath("//a[contains(text(),'Add Emp')]")).click();


	driver.findElement(By.xpath("//input[@id='firstName']")).sendKeys("selenium");
		driver.findElement(By.xpath("//input[@id='lastName']")).sendKeys("hq");
		driver.findElement(By.xpath("//input[@id='btnSave']")).click();

	}

	@AfterClass
	public void closeApplication() {
		driver.quit();
```

```
	}
}
```

**Final code in XML File**

```
<suite name="demo">
	<parameters>
		<parameter name="browser" value="edge" />
	</parameters>
	<test name="smoke1">
		<classes>
			<class name="com.tng.ohrm.OhrmAddEmployee_Parameters" />
		</classes>
	</test>
</suite>
```

**Implementing Parallel Execution Concept Using GRID and TestNG**

- Selenium GRID is used for distributed execution
- we can distribute the execution to multiple remote machines
- It uses JSON Wire Protocol to give requests and to take response from/to hub and node

**Configure Selenium Grid**

- Download Selenium Server Jar file
  - Go to: http://www.seleniumhq.org/download/
  - Under Selenium Standalone Server
  - Click on version number
  - It downloads a jar file
    - To download specific version server jar file
    - Goto https://selenium-release.storage.googleapis.com/index.html
    - Click on version number
    - Download that specific version jar file
  - Keep it in safe folder
- Create HUB
  - Open cmd
  - Use cd and navigate to server jar location
    - Or you can goto server jar file folder manually
    - type cmd in folder path and press enter
    - you will get command window with the folder path
  - execute blow command to make the current machine as hub
  - java -jar selenium-server-standalone-3.5.3.jar -role hub
- This will give **Selenium Grid hub is up and running** message
  - It also gives an url
- Register nodes to hub
  - Nodes should have browser drivers and server jar file
  - Go to node machine -> Open cmd in node machine by going to server jar folder path like above process
- execute below command to register nodes with hub
- java -Dwebdriver.ie.driver="F:\SeleniumSoftware\BrowserDrivers\iedriverserver.exe" -Dwebdriver.chrome.driver="F:\SeleniumSoftware\BrowserDrivers\chromedriver.exe" -

Dwebdriver.gecko.driver="F:\SeleniumSoftware\BrowserDrivers\geckodriver.exe" -Dwebdriver.edge.driver="F:\SeleniumSoftware\BrowserDrivers\MicrosoftWebDriver.exe" -jar selenium-server-standalone-3.5.3.jar -role node -hub http://192.168.238.1:4444/grid/register/

- We should see a message in hub command prompt "Registered a node http://192.168.238.1:5555"
- Hub will use 4444 port number
- nodes will use 5555 port number
- Test Configuration:
    - Goto Node URL
    - click on console
    - click on create session
    - select the configured browser (for which we have given details when configuring node)
    - click on OK-> It should create that session in node machine
- Configure the code to access the node machine
- To execute programs in remote machine we have to use RemoteWebDriver class
- It takes two inputs
    - Node console URL
    - desired capabilities
- DesiredCapabilities is a class used to specify session details
- It creates a json object internally when program executed and creates teh same session in node machine
- https://github.com/SeleniumHQ/selenium/wiki/DesiredCapabilities

```
DesiredCapabilities cap = new DesiredCapabilities();
cap.setBrowserName("chrome");
// open chrome browser
WebDriver driver = new RemoteWebDriver(new URL(
"http://192.168.31.210:4444/wd/hub/"),cap);
```

**Note: See the video in drive on how to implement parallel execution on remote machines using TestNG and GRID**

**DataDriving In TestNG**

**Read and Write Data To Files**

- **Read Text Files**
- We can use FileReader and BufferedReader classes to read line by line data from text files
- FileReader cannot read line by line data. That's why we have to use buffered reader class

```
public static void main(String[] args) throws IOException {
	String flPath = "C:\\Users\\envy\\Desktop\\Aug29\\Data\\Demo1.txt";

	FileReader fr = new FileReader(flPath);
	BufferedReader br = new BufferedReader(fr);

	String ln = br.readLine();

	while(ln!=null) {
		System.out.println(ln);
```

```
            ln = br.readLine();
        }
        fr.close();
}
```

- We can use FileWriter and BufferedWriter classes to write line by line data to text files
- FileWriter cannot write line by line data. That's why we have to use buffered writer class

```
public static void main(String[] args) throws IOException {
        String flPath = "C:\\Users\\envy\\Desktop\\Aug29\\Data\\Demo2.txt";

        FileWriter fr = new FileWriter(flPath);
        BufferedWriter br = new BufferedWriter(fr);
        br.write("bcd");
        br.newLine();
        br.write("asdf");
        br.newLine();
        br.write("qwerty");
        br.newLine();

        br.close();
        fr.close();
}
```

**Create Folders**

- We can create folders using File Class
- This gives many methods to verify folder/file existence, Folder or File, to get absolute path, To get file name…etc
- We can create folder using MkDir and MKDirs methods
- MKDir can create a single directory
- MKDirs can create a complete folder path

```
public static void main(String[] args) {
        String flPath = "C:\\Users\\envy\\Desktop\\Aug29\\Data\\d1\\d2";

        File fl = new File(flPath);
        fl.mkdirs();
}
```

**Read and Write Data From Excel**

- To Read data from excel files there are two providers
    - JXL
        - Can read and write data only from xls files
    - Apche POI
        - Can read and write data from xls, xlsx, word, ppt, outlook….etc
- JXL is very old and the last update on JXL was in 2009
- POI is providing two sets of classes for XLS and XLSX
    - HSSF classes
        - Using this we can read data from xls files

        - Horrible spread sheet format
    - XSSF classes
        - Using this we can read data from xlsx files
        - XML spread sheet format
    - We have some interfaces which are implemented by hssf and xssf classes
- Configure Apache POI
    - Goto https://poi.apache.org/download.html
    - Under Binary distribution click on poi-bin-3.17-20170915.zip
    - Download and extract the zip file
    - Add all jars from all folders to project build path

**Read All Data From Excel**

```
public static void main(String[] args) throws IOException {

	String flPath ="C:\\Users\\envy\\Desktop\\Aug29\\Data\\Ohrm.xlsx";
	FileInputStream fls = new FileInputStream(flPath);

	//get work book
	Workbook wb;
	if(flPath.substring(flPath.length()-4).equalsIgnoreCase("xlsx")) {
		wb = new XSSFWorkbook(fls);
	}else {
		wb = new HSSFWorkbook(fls);
	}

	//get work sheet
	Sheet sht = wb.getSheet("AddEmp");

	//get row count
	int rc = sht.getLastRowNum();

	//get column count
	int cc = sht.getRow(0).getLastCellNum();

	//get data row by row
	for(int r=0;r<=rc;r++) {
		for(int c=0;c<cc;c++) {

	System.out.println(sht.getRow(r).getCell(c,MissingCellPolicy.CREATE_NUL
L_AS_BLANK).toString());
		}
	}

	//close workbook
	wb.close();
}
```

**Read Data From Testcase**

```
public static HashMap<String, String> getTcData(String tcid, String shtName,
String flPath) throws IOException {
	// String flPath ="C:\\Users\\envy\\Desktop\\Aug29\\Data\\Ohrm.xlsx";
	FileInputStream fls = new FileInputStream(flPath);

	// get work book
	Workbook wb;
	if (flPath.substring(flPath.length() - 4).equalsIgnoreCase("xlsx")) {
		wb = new XSSFWorkbook(fls);
	} else {
		wb = new HSSFWorkbook(fls);
	}

	// get work sheet
	Sheet sht = wb.getSheet(shtName);

	// get row count
	int rc = sht.getLastRowNum();

	// get column count
	int cc = sht.getRow(0).getLastCellNum();
	HashMap<String, String> tcMap = new HashMap<>();

	// get data row by row
	for (int r = 1; r <= rc; r++) {

		if (sht.getRow(r).getCell(0).toString().equalsIgnoreCase(tcid)) {

			for (int c = 0; c < cc; c++) {
				String kName = sht.getRow(0).getCell(c,
MissingCellPolicy.CREATE_NULL_AS_BLANK).toString();
				String kVal = sht.getRow(r).getCell(c,
MissingCellPolicy.CREATE_NULL_AS_BLANK).toString();

				tcMap.put(kName, kVal);
			}

			break;
		}
	}

	// close workbook
	wb.close();

	return tcMap;
}
```

**Read All Data From Excel and Link Data Provider**

```
@DataProvider
public Object[][] getTcData() throws IOException {

	String flPath = "C:\\Users\\envy\\Desktop\\Aug29\\Data\\Ohrm.xlsx";
	FileInputStream fls = new FileInputStream(flPath);

	// get work book
	Workbook wb;
	if (flPath.substring(flPath.length() - 4).equalsIgnoreCase("xlsx")) {
		wb = new XSSFWorkbook(fls);
	} else {
		wb = new HSSFWorkbook(fls);
	}

	// get work sheet
	Sheet sht = wb.getSheet("AddEmp");

	// get row count
	int rc = sht.getLastRowNum();

	// get column count
	int cc = sht.getRow(0).getLastCellNum();

	Object[][] tData = new Object[rc][1];

	// get data row by row
	for (int r = 1; r <= rc; r++) {
		HashMap<String, String> tcMap = new HashMap<>();
		for (int c = 0; c < cc; c++) {
			String kName = sht.getRow(0).getCell(c,
MissingCellPolicy.CREATE_NULL_AS_BLANK).toString();
			String kVal = sht.getRow(r).getCell(c,
MissingCellPolicy.CREATE_NULL_AS_BLANK).toString();

			tcMap.put(kName, kVal);
		}
		tData[r - 1][0] = tcMap;
	}

	// close workbook
	wb.close();

	return tData;
}
```

**Using Above Data Provider in TestNG**

```
public class OhrmAddEmployee_DDT {
	WebDriver driver;
```

```
	@BeforeClass
	public void OpenApplication() {
		// open new chrome window
		driver = DriverFactory.getDriver("chrome");

		// specify page load
		driver.manage().timeouts().pageLoadTimeout(10, TimeUnit.SECONDS);

		// specify element wait
		driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

		// maximize window
		driver.manage().window().maximize();

		// goto orange hrm login page
		driver.get("http://opensource.demo.orangehrmlive.com");
	}

	@Test
	public void Login() {
		// enter text on username

	driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys("ad
min");

		// enter text on password

	driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys("ad
min");

		// click on login button
		driver.findElement(By.xpath("//input[@id='btnLogin']")).click();

	}

	@Test(dependsOnMethods = "Login", dataProvider = "getTcData",
dataProviderClass = OhrmDP.class)
	public void AddEmployee(HashMap<String, String> empData) {
		// click on PIM link
		driver.findElement(By.xpath("//a[.='PIM']")).click();

		// click on add employee link
		driver.findElement(By.xpath("//a[contains(text(),'Add
Emp')]")).click();

		// enter text on first name

	driver.findElement(By.xpath("//input[@id='firstName']")).sendKeys(empDa
ta.get("fname"));
```

```
		// enter text on last name

	driver.findElement(By.xpath("//input[@id='lastName']")).sendKeys(empDat
a.get("lname"));

		// click on save button
		driver.findElement(By.xpath("//input[@id='btnSave']")).click();

	}

	@AfterClass
	public void closeApplication() {
		driver.quit();
	}
}
```

**Data Driving a Scenario**

```
public class OhrmAddEmployee_DDT_Scenario {
	WebDriver driver;
	HashMap<String, String> tcData;

	@BeforeClass
	public void OpenApplication() throws IOException {

		tcData = DataFactory.getTcData("TC_OHRM_AddEmployee_001", DataConstants.globalDataSheet,
				DataConstants.dataFilePath);
		// open new chrome window
		driver = DriverFactory.getDriver("chrome");

		// specify page load
		driver.manage().timeouts().pageLoadTimeout(10, TimeUnit.SECONDS);

		// specify element wait
		driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

		// maximize window
		driver.manage().window().maximize();

		// goto orange hrm login page
		driver.get("http://opensource.demo.orangehrmlive.com");
	}

	@Test
	public void Login() {
		// enter text on username

	driver.findElement(By.xpath("//input[@id='txtUsername']")).sendKeys(tcData.get("username"));

		// enter text on password

	driver.findElement(By.xpath("//input[@id='txtPassword']")).sendKeys(tcData.get("password"));

		// click on login button
		driver.findElement(By.xpath("//input[@id='btnLogin']")).click();

	}

	@Test(dependsOnMethods = "Login")
	public void AddEmployee() {
		// click on PIM link
		driver.findElement(By.xpath("//a[.='PIM']")).click();
```

```
		// click on add employee link
		driver.findElement(By.xpath("//a[contains(text(),'Add Emp')]")).click();

		// enter text on first name

	driver.findElement(By.xpath("//input[@id='firstName']")).sendKeys(tcData.get("firstname"));

		// enter text on last name
		driver.findElement(By.xpath("//input[@id='lastName']")).sendKeys(tcData.get("lastname"));

		// click on save button
		driver.findElement(By.xpath("//input[@id='btnSave']")).click();

	}

	@AfterClass
	public void closeApplication() {
		driver.quit();
	}
}
```

**Write Data to Excel**

```
public static void main(String[] args) throws IOException {

	Workbook wb = new XSSFWorkbook();
	Sheet sht = wb.createSheet("Sheet1");
	Row row1 = sht.createRow(0);
	row1.createCell(0).setCellValue("asdf");

	String flPath = "C:\\Users\\envy\\Desktop\\Aug29\\Data\\Demo.xlsx";

	FileOutputStream fs = new FileOutputStream(flPath);
	wb.write(fs);
	// close workbook
	wb.close();
}
```

**Write Data to Existing Excel**

```
public static void main(String[] args) throws IOException {

	String flPath = "C:\\Users\\envy\\Desktop\\Aug29\\Data\\Demo.xlsx";

	FileInputStream fis = new FileInputStream(flPath);
	Workbook wb = new XSSFWorkbook(fis);
	Sheet sht = wb.getSheet("Sheet1");

	int rc = sht.getLastRowNum();

	Row row1 = sht.createRow(rc + 1);
	row1.createCell(0).setCellValue("updated value");

	FileOutputStream fs = new FileOutputStream(flPath);
	wb.write(fs);
	// close workbook
```

```
	wb.close();
}
```

**Read Data From Database row by row**

```
public static void main(String[] args) throws SQLException {
	// Database URL.
	// "company" Is database name. You can change It as per your database
name.
	String sqldb_url = "jdbc:mysql://localhost:3307/company";
	// Use your database username here. It Is "root" for root account.
	String sqldb_uname = "root";
	String sqldb_pass = "admin";
	// To Create database connection.
	Connection connect = DriverManager.getConnection(sqldb_url,
sqldb_uname, sqldb_pass);

	Statement q = connect.createStatement();
	ResultSet rs = q.executeQuery("select * from emp");
	while (rs.next()) {

		System.out.println(rs.getString(1));
		System.out.println(rs.getString(2));
		System.out.println(rs.getString(3));
		System.out.println(rs.getString(4));
	}

	connect.close();
}
```

**Read Data From Database col by col**

```
public static void main(String[] args) throws SQLException {
	// Database URL.
	// "company" Is database name. You can change It as per your database
name.
	String sqldb_url = "jdbc:mysql://localhost:3307/company";
	// Use your database username here. It Is "root" for root account.
	String sqldb_uname = "root";
	String sqldb_pass = "admin";
	// To Create database connection.
	Connection connect = DriverManager.getConnection(sqldb_url,
sqldb_uname, sqldb_pass);

	Statement q = connect.createStatement();
	ResultSet rs = q.executeQuery("select * from emp");

	ResultSetMetaData rsm = rs.getMetaData();
	int cc = rsm.getColumnCount();
```

```
	for (int c = 1; c <= cc; c++) {
		System.out.println("************");
		System.out.println(rsm.getColumnName(c));
		System.out.println("************");

		while (rs.next()) {
			System.out.println(rs.getString(c));
		}
		rs.beforeFirst();
	}
	connect.close();
}
```

- TestNG is providing DataProvider annotation to create data driven tests
- A method with @DataProvider returns two dimensional array
  - First Dimension specifies how many times the test method should execute
  - second dimension specifies how many parameters the test method should contain
- We have to specify dataprovider name in @test annotation definition
- We can use the data provider name in @Test(DataProvider=”methodName”)
- We can also store data provider methods in a different class and use them
- In that case we have to specify the name of the class
  - @Test(dataProvider="getTcData",dataProviderClass=OhrmDP.class)
- We can write excel data reading code in side of data provider methods and use them in test methods
- The below data provider method will read data from excel sheet and return a two dimensional array which contains hash map for each index

```
@DataProvider
public Object[][] getTcData() throws FileNotFoundException, IOException{
	String flPath = "C:\\Users\\envy\\Desktop\\Aug16Sel\\Data\\TestData_old.xls";

	// depends on file extension create hssf or xssf instance
	Workbook wb;
	if (flPath.substring(flPath.length() - 3).equalsIgnoreCase("xls")) {
		wb = new HSSFWorkbook(new FileInputStream(flPath));
	} else {
		wb = new XSSFWorkbook(new FileInputStream(flPath));
	}

	// get work sheet
	Sheet sht = wb.getSheet("Sheet1");

	// get row count
	int rc = sht.getLastRowNum();

	// get columns count
	int cc = sht.getRow(0).getLastCellNum();

	Object[][] tData = new Object[rc][1];

	for (int r = 1; r <= rc; r++) {
		HashMap<String, String> tcMap=new HashMap<>();
```

```
            for (int c = 0; c < cc; c++) {
                  String cName = sht.getRow(0).getCell(c).toString();
                  String cVal = sht.getRow(r).getCell(c,
   MissingCellPolicy.CREATE_NULL_AS_BLANK).toString();
                  tcMap.put(cName, cVal);
            }
            tData[r-1][0]=tcMap;
      }

      wb.close();
      return tData;

   }
```

**Using Above Data Provider in Class**

```
@Test(dependsOnMethods="Login",dataProvider="getTcData",dataProviderClass=OhrmDP.clas
s)
public void AddEmployee(HashMap<String, String> tcData) {
      driver.findElement(By.xpath("//a[.='PIM']")).click();
      driver.findElement(By.xpath("//a[contains(text(),'Add Emp')]")).click();

      driver.findElement(By.xpath("//input[@id='firstName']")).sendKeys(tcData.get("
fname"));
      driver.findElement(By.xpath("//input[@id='lastName']")).sendKeys(tcData.get("l
name"));
      driver.findElement(By.xpath("//input[@id='btnSave']")).click();

}
```

**Exception Handling**

- Java throws an exception when a statement is failed
- To know why the program is failed we can observe the exception
- Java will throw an exception depends on what is failed
- Exception will have the details of failure
- A timedoutexception will tell us that an element is not displayed/attribute is not same with in specified time out.
- A NosuchElementEception will tell us the element is not found with the locator what we have given
- A nullpointerexception will tell us that the driver variable is declared with type but not assigned with any value.
- Like these different exceptions will have different meanings.
- Scenario1:
    - For example to verify an element is exist or not, selenium is not providing any statement which returns elements existence is true or not. If we use driver.findElemet and if element is not exist it throws an exception that is NoSuchElementException. But we want to have a method which returns true or false based on element existence.
- Scenario2:
    - When we are writing program to enter text on text box, If the textbox is not visible Then we get an exception elementnotvisible. The program will get stopped immediately. We develop 100's of programs and execute overnight. If any program failed then we need minimum information

about why that is failed before we debug that program. If we know the issue before we debug then we can save lot of time.

- We can handle exceptions using try catch blocks
- **What is try catch block?**
  - try catch block is used to create exception handler.
  - We can execute some code using catch blocks when error thrown by java
  - In try block we specify all the main code.
  - In catch block we specify handler code
  - We can implement multiple catch blocks depends on exception and with handler code.
  - If we use try catch block java thinks that we are handling the exception and program status will be shown as passed.
  - **syntax:**

```
try{

main code;

}

catch(exceptionname variable){

        handler code;

}

catch(exceptionname variable){

        handler code;

}

catch(exceptionname variable){

        handler code;

}
```

- We can write multiple catch blocks based exception name
- There is one more block called "finally" which will be executed as post condition to try block
- The finally block will be executed irrespective of exception occur or not
- The final syntax will be like Try{ some code } catch(Exception e){some code} finally{ some code }
- throws keyword:
  - This is used to specify a method throws some exceptions which are not handling by try catch block
  - In this case either you should develop try catch block for those exceptions or you need to use throws keyword after method definition
- throw keyword
  - This is used to throw some error by the statement. We can intentionally throw some exception
  - In catch block after we log the result we make the test fail by using throw keyword
- Types of Exceptions:
  - Checked Exceptions
  - Unchecked Exceptions/Runtime Exceptions
- checked exceptions are known to the compiler

- While developing the code if any statement throws an exception then compiler will find that and ask you to use throws or implement try catch block.
    - ex: thread.sleep will throw InterruptedException
    - File class will throw filenotfoundexception
    - URL class will throw MalformedURLException
- You must use try catch block or use throws keyword if any statement is generating checked exception.
- Unchecked exceptions occur in runtime.
- Compiler will not find these in developing stage.
    - Ex: ArrayOutOfBoundException, NoSuchElementException, TimedOutException
- Compiler doesn't know about unchecked exceptions. That's why it is your choice to create try catch block.
- But either it is checked/unchecked we handle them using try catch block.
- If it is checked the compiler will ask you to implement try catch while developing code and for unchecked you do the same after executing the code.
- In real time we create try catch blocks sometimes
    - To skip the exception and continue the execution
    - To customize the result messages
    - To give alternate execution code
- write code to find element existence

```
public boolean iselementExist(WebDriver driver, By locator) {
	try {
		driver.findElement(locator);
		return true;
	} catch (NoSuchElementException nse) {
		return false;
	}
}
```

- write code to customize result

```
@Test
public void login() {
	try {
		// enter text on username textbox
		driver.findElement(By.id("txtUsername1")).sendKeys("admin");

		// enter text on password
		driver.findElement(By.cssSelector("#txtPassword")).sendKeys("admin");

		// click on login button
		driver.findElement(By.cssSelector("input[value='LOGIN']")).click();
	} catch (NoSuchElementException nse) {
		System.out.println("test is failed because element is not found");
		// write logic to create custom results
		// make the test fail as usual
		throw nse;
	}

}
```

- write code to enter text using

```
public void enterText(WebDriver driver,By locator, String txtToEnter){

	try{
		WebElement elm = getElement(driver,locator);
		elm.clear();
		elm.sendKeys(txtToEnter);
	}
	catch(exception e){
		reporter.log("failed to enter text")
		throw e;
	}
}
```

- Terms:
  - try : block
  - catch: block
  - finally: block
  - throws: used when any statement is throwing checked exception and if we dont implement try catch
  - throw : used to throw the exception intentionally based on situation
- throw new NoSuchElementException();
- throw new NoSuchElementException("element not found");

- NoSuchElement
  - An element could not be located on the page using the given search parameters.
- NoSuchFrame
  - A request to switch to a frame could not be satisfied because the frame could not be found.
- NoSuchWindow
  - A request to switch to a different window could not be satisfied because the window could not be found.
- StaleElementReference
  - An element command failed because the referenced element is no longer attached to the DOM.
- ElementIsNotSelectable
  - An attempt was made to select an element that cannot be selected.
- ElementNotVisible
  - An element command could not be completed because the element is not visible on the page.
- InvalidElementState
  - An element command could not be completed because the element is in an invalid state (e.g. attempting to click a disabled element).

```
//**********************
//Sample method to handle exception
//**********************
  public void Login() throws IOException {
     try{
          orangeHRM.loginPage.enterUserName("admin");
          orangeHRM.loginPage.enterPassword("admin1");
```

```
            orangeHRM.loginPage.clickOnLoginButton();
            Assert.assertTrue(orangeHRM.welcomePage().isWelcomeDisplayed());
    }
    catch(NoSuchElementException ne){
             File scrFile =
            ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
             FileUtils.copyFile(scrFile, new File("C: \\ErrImages\\Test1.png"));
            Assert.fail("Failed to login because element is not found");
    }

    catch(AssertionError ae){

    }

    finally{
       Reporter.log("Login completed");
       }
    }
```

**Reporting Results**

- TestNG is providing Reporter class to create reports
- We can use Reporter.log("message"); to send reports to html page
- TestNG create reports under test output folder in project
- If this folder is not displayed refresh the project
- To see the reports go to index.html or emailable report.html under test-output folder
- TestNG doesn't show reports for before and after related methods
- In real time we customize reporting mechanism for projects
    - We can customize the existing TestNG reporting mechanism
    - We can create new reporting mechanism on your own
    - We can use third party reporting mechanism
- We can customize TestNG reports using listeners concept
- TestNG providing some classes and interfaces to extend the TestNG works. These are called listener classes or interfaces.
- If we create new reporting mechanism we must make sure that supports TestNG
- To use thridparty reports You must know about how to integrate third party reporting mechanism
- We can integrate third party reporting mechanism with TestNG reporting mechanism using listeners concept
- We are going to use extentReports to generate html reports for the tests

**Extent Reports**

- ExtentReports are 3rd party reporting mechanism
- Until 2.41.2 version these are open source
- After that they made a community edition
- Configure Extent Reports
    - goto http://extentreports.com/community/
    - click on java 2.41.2
    - click on download icon in opened google drive
    - it will download extentreports-java-2.41.2.zip
    - extract to safe folder location

- Add all jars from all folders to build path
- //specify where to create html file
- ExtentReports report = new ExtentReports("src\\..\\Reports\\"+this.getClass().getSimpleName()+".html");
- 
- //create test
- ExtentTest test = report.startTest(this.getClass().getSimpleName());
- 
- test.log(StepStatuc,"StepName","StepDescription");
- 
- test.log(LogStatus.PASS,"AddEmployee","employee added");

**Assertions**

- What kind of verifications will you be doing in application as part of manual testing?
  - Is element exist in application?
  - Is element displayed in application?
  - Is element attribute having expected value?
- We can use if condition to create the verifications
- Is element displayed in application?

```
if(element.isDisplayed()){
send pass report
}else{
send fail report
}
```

- Is element attribue having expected value?

```
If(element.getAttribute(attributename).equals("expectedAttributeValue")){
send pass report
}else{
send fail report
}
```

- Is element exist in application?
- To verify element existence we have find element. If element is exist it will find the element. If element is not exist it will throw nosuchelement exception.
- We have to use try catch block and skip the exception to return result.

```
public boolean iselementExist(WebDriver driver, By locator) {
try {
   driver.findElement(locator);
   return true;
} catch (NoSuchElementException nse) {
   return false;
   }
}
```

- To make a program fail on a verification failure using if condition

```
if(element.isDisplayed()){
   send pass report
```

```
}else{
    send fail report
    //throw new nosuchelementexception("element is not displayed");
    throw new assertionerror("verification failed");
}
```

- TestNG is providing Assertions where it simplifies creating verifications
- We have asserttrue, assertfalse, assertequals, assertnotequals, assertnull, assertnotnull kind of statement which will verify and throw the assertionerror exception automatically.
- To verify element displayed assertTrue(element.isDisplayed());
- This statement will expect a true value to be returned. If element is not displayed then it will throw assertionerror exception, it is displayed it will continue execution.
- These are called hard asserts
- On failure these will throw assertionerror exception
- When exception occurs the test will stop the execution and goto next test for execution
- TestNg also have softasserts
- Softasserts doesnt throws assertion exception until we use a statment assertall
- we can create multiple verifications and at the end we assert all
- the verifications will be executed at that point and give us result

```
SoftAssert sa = new SoftAssert();
sa.asserttrue();
sa.asserttrue();
sa.asserttrue();
sa.assertAll();
```

# Maven

- Maven is a build tool for Java Projects
- It gives below features
    - Folder structure
    - Facility to download jar files automatically
    - More portability of projects
    - TestNG Integration
    - Running multiple TestNg xml files
    - Integration with Jenkins
- We have to create maven projects to create dependencies in POM.xml
- Configure maven in eclipse:
    - Help -> Eclipse Market Place -> Maven integration for eclipse
    - Click on install
- Create Maven Project:
    - File > New -> Project -> Type maven -> Select Maven Project
    - Click on next -> next -> next
    - Give any name for group id and artifact id
        - These are useful if we want to update our software into maven repository
        - We don't need to worry about these names
        - Maven creates a project with artifact id name
        - Group id is like url in maven repository and should be unique if we wanted to update in maven repository
    - Click on Finish
    - It will create a maven project
- It give two src folder paths
- main: all reusable code we keep here
- test: all tests we develop here
- it's not compulsory and it's just a standard
- It will also create package with group and artifact ids. We can delete it if we don't want.

**Note:** Sometimes maven by default creates the project with java 1.5 environment. But we should use java 1.8 as the execution environment from selenium 3 onwards. For that In Project Build Path Remove Java 1.5 library and add java 1.8 library. Also change the compiler from 1.5 to 1.8

**POM.XML**

- It gives a POM.xml file
- This file is going have all the dependencies that are needed to develop and execute the project
- We have to specify the maven dependency information of any software in this file
- Lot of software maintain their build in maven repository
- We need to get the dependency xml code for those software and keep it POM.xml file under dependencies tag

**Prepare pom.xml file**

- Copy the dependencies from respective sites and paste it under dependencies tag in pom.xml file
- Get selenium dependency code from http://www.seleniumhq.org/download/maven.jsp

```
<dependency>
	<groupId>org.seleniumhq.selenium</groupId>
	<artifactId>selenium-java</artifactId>
	<version>3.5.3</version>
</dependency>
```

- Get testng dependency code from http://testng.org/doc/maven.html

```
<dependency>
	<groupId>org.testng</groupId>
	<artifactId>testng</artifactId>
	<version>6.11</version>
	<scope>test</scope>
</dependency>
```

- Like this you can also search for dependency of any software in maven repository https://mvnrepository.com/
- Add local repository variable:
  - Right click on Project -> build path -> configure build path -> click on add variable -> select M2_REPO -> OK
- Start writing programs in regular way

**Executing TestNG XML files from POM.xml**

- To execute TestNG xml files from pom.xml we have to add some plug in code in to pom.xml file
- Maven uses surefire plugin for executing TestNG xml files
- You can more information on surefire plugin from below url
  - http://maven.apache.org/surefire/maven-surefire-plugin/examples/testng.html
- Configure surefire plugin code in POM.xml file
  - specify below code above </project> tag

```
<build>
	<plugins>
		<plugin>
			<groupId>org.apache.maven.plugins</groupId>
			<artifactId>maven-surefire-plugin</artifactId>
			<version>2.20</version>
			<configuration>
				<suiteXmlFiles>
					<suiteXmlFile>testng xml file1</suiteXmlFile>
					<suiteXmlFile>testng xml file2</suiteXmlFile>
				</suiteXmlFiles>
			</configuration>
		</plugin>
	</plugins>
</build>
```

## Jenkins

- Jenkins is a continues integration tool
- We can schedule the execution using Jenkins

Prerequisites for Jenkins schedule:

- Configure maven in windows
    - Goto https://maven.apache.org/download.cgi
    - download apache-maven-3.5.0-bin.zip file and extract to a safe folder
    - **create environment path variables**
    - Run Dialog -> Type sysdm.cpl -> Enter
    - It will open system properties window
    - Goto advanced tab
    - Click on Environment variables
    - Click on new under user variables
    - Variable name as MAVEN_HOME
    - Variable value as maven folder path (C:\Program Files\Apache Software Foundation\apache-maven-3.3.9)
    - Goto Path Variable or create a new variable if Path variable not exist
    - Variable name as Path
        - append value as C:\Program Files\Apache Software Foundation\apache-maven-3.3.9\bin
        - Click on OK
    - Goto command prompt -> Use mvn –v command to see the configured maven version
- **Download and configure Jenkins**
    - goto https://jenkins.io/ -> click on download
    - Click on Generic Java Package (.war)
    - This will download war file. (web archive file)
    - open cmd -> navigate to the path where we have war file -> type java -jar jenins.war -> enter
    - Jenkins configuration will start
    - It will by default configure in 8080 port number
    - We can access Jenkins using http://localhost:8080
    - wait for Jenkins is fully up and running message and access the url
    - If we access for the first time then it asks for password
        - Find the password in cmd window
        - Copy and give the password -> click on continue
    - It will ask to install plugins -> click on install suggested plugins
    - Wait for Jenkins to install all plugins
    - Create an admin user for login -> save -> finish
    - Click on Start Using Jenkins
    - create a new job (New Item)
    - Observe that maven Job is not displayed
    - **Install maven plugin**

        - Click on manage Jenkins
        - Click on manage plugins
        - Click on Available tab
        - Search for Maven Integration plugin and select the checkbox
        - click on install without restart
    - **Configure java to Jenkins**
        - Manage jenkins -> click on global tool configuration
        - Click on add JDK
        - Uncheck install automatically
        - Give any name
        - Give Java JDK path as value (JAVA_HOME)
        - Click on Apply
    - **Configure maven to Jenkins**
        - Manage jenkins -> click on global tool configuration
        - Under Maven
        - Uncheck install automatically
        - Give Anyname
        - Give extracted maven folder path as value (MAVEN_HOME)
        - Click on Apply
    - **Configure mail (I am using Gmail)**
        - Jenkins -> Manage Jenkins -> Configure System
        - Under E-mail Notification
        - Specify smtp server as  smtp.gmail.com
        - Click advanced
        - Select SMTP Authentication
        - give mail and password
        - Click on advanced
        - Select user ssl
        - Smtp port number: 465
        - Reply to address: same email id
        - Test it by sending the mail
        - Give a mail id and click on test configuration
        - It should display email successfully sent message
        - Click on save

**Create Jobs**

- Jenkins -> Click on New Item
- Give Job Name
- Select Maven Project
- Click on OK
- This will create maven job
- Specify Build Information
    - Click on Build Tab
    - Specify the path of POM.xml file

    - Click on Apply
- Specify Build Triggers
    - Deselect all check boxes and select build periodically
    - Specify the schedule execution time
        - MINUTE (0-59), HOUR (0-23), DAY (1-31), MONTH (1-12), DAY OF THE WEEK (0-7)
        - */5 * * * * means every 5 minutes
        - 5 * * * * means the 5th minute of every hour
        - H 7 * * * daily 7 am slot( any time between 7-7:59)
        - 30 08 * * 1-6 start build daily at 08:30 in the morning, Monday – Fridays:
        - 00 0,12 * * 0-5 weekday daily build twice a day, at lunchtime 12:00 and midnight 00:00, Sunday to Thursday:
        - H 16 * * 1-5 start build daily in the late afternoon between 4:00 p.m. – 4:59 p.m. or 16:00 -16:59 depending on the projects hash:
        - @midnight or 59 23 * * 6 start build at midnight, every Saturday
        - H(0,30) 02 01 * * every first of every month between 2:00 a.m. – 02:30 a.m. :
- Click on Save
- This will execute the JOB as per scheduled time
- If displays blue color means the build/execution is passed
- If you want to execute right away then click on build now
- Specify Email configuration to the job
    - In jenkins job under build settings
    - Select email notification
    - Give the recipient email ids
    - Select options of when you wanted get mail
    - Click on save
    - From now onwards you the recipients get mail about the build status when ever the job is executed
- **Creating Freestyle Project Job (Without maven)**
    - We can use this freestyle jobs for executing TestNG xml files directly
    - For this we must specify where all jar files are located
    - Goto eclipse
    - Create a new Java project
    - Create a folder Lib under project
    - Copy all jar files under that folder
    - Create a TestNG test class and xml file
    - Create a file with .bat extension with below code and name it as test
        - set myProjectpath=C:\Users\envy\Desktop\SeleniumOct3rdSpace\Project2\
          cd %myProjectpath%
          set classpath=%myProjectpath%\bin;%myProjectpath%\Lib*
          java org.testng.TestNG %myProjectpath%\tngClassOhrm.xml
    - Goto Jenkins
    - Click on new item
    - Give name and click on free style project

    - Specify Build trigger
    - Under build Click on Add Build setup
    - Select Execute windows batch command
    - Specify the path of test.bat file
- Save