**Automation with Java:**

Faster:

· Manual = 100 test cases ⇒ 100 minutes

· Automation = 100 test cases ⇒ 5 minutes (Bad)

· Facebook login with 1 user ⇒ Test data

o Manual = faster

o Automation =slow

·

· Facebook login with 100 users ⇒ Test data

o Manual = slow

o Automation =faster (data driven testing)

SQL = programing language (where to use = backend or database)

Java = programing language (where to use = Application)

**Types application:**

1. Web based application = need browser (FB, Amazon) ⇒ Selenium

2. Client based application = no need browser (Word) XXX NO

3. Mobile application ⇒ Selenium Appium

a. App = no need browser

i. Native = single technology >> download

ii. Hybrid = multiple tech >> may or may not be download

b. Web = need browser

Selenium = API (Application Programming Interface) ⇒ can interact with browser (Chrome, MS Edge, Firefox, Safari)

· middleman between code and browser

· Selenium drives the browser (open browser, close browser, click button, select anything)

Maven = build tool (can check our build ⇒ Maven acts as supervisor = monitor our automation /code, if all good = build success, if not good = build failed)

Eclipse = tool where to write coding, known as IDE (Integrated Development Environment), IntelliJ

Project: File >>new >> project >> maven>> maven project>>next>>select create simple project>>next>>

Group id: com.companyname

Artifact id: NameOfProject

finish

Project explorer:

1. 4 src (source code folder)

a. 2x test

i. Java = code

ii. Resources = supporting file (test data, test cases)

b. 2x main

i. Java = code

ii. resources= supporting file (test data, test cases)

2. JRE -1.5 ⇒ Java 5 >>> Java 8 or 11 by Maven compiler plugin

3. Target ⇒ report

4. Pom.xml ⇒ a file written in xml language

a. Why we need pom file

i. Pom known as heart of the maven

ii. Add/inject any library file/JAR/dependency file inside the pom file

iii. Add/inject any plugin inside the pom file

5. JAR/library file = supporting file

6. Plugin = modify any function

7. XML =

a. language

b. where to use = middle end (web server)

c. why = transfer data from one place to another place

d. similar like HTML (front end language)

e. <write>

**IT languages:**

1. Front end language/ browser end/ GUI (Graphical User Interface)

a. HTML

b. CSS ⇒ display graphic or data

c. JS

2. Middle end language/Web Server

a. XML

b. JSON ⇒ transfer data

3. Back-end language/Database

a. SQL ⇒ retrieve the data

4. Application language /coding

a. Java

b. Python

c. Ruby ⇒ doing coding and building application

d. C# (C sharp)

e. C++

f. .Net

Project Setup ⇒ in the pom.xml:

1. Add Maven Compiler Plugin
   1. Update to Java version 11
2. Add Selenium library file

Selenium latest version =

Project setup Ready ⇒ start coding

src/test/java ⇒ rt click>> new >> package>> small letter>> com.practice

src/main/java ⇒ rt click>> new >> package>> small letter>> com.practice

com.practice >>> rt click >>> new >>> class

project>>> src/test or main java>>> package>>> class >>> method>> write code inside the method

method ⇒ main

Project

· src/main/java

o package ⇒ com.practice

* class ⇒ method
* method:
  + Return method (no void word)
  + Non-return (void) method
    - * public void getText() {

}

Public ⇒ Access modifier

Void ⇒ Method type = returns nothing (non-return method)

getText = Method name

() = Method sign

{} = Method body/method description ⇒ Write actual CODE

* Static method

· Src/test/java

o Package ⇒ com.testrun

* Class ⇒ method
* Method:
* Main method ⇒ Runs the code

Class ⇒ First letter always upper case

Method ⇒ Everything lower case (at least the first letter is in lower case)

* main/java ⇒ all the coding (framework)
* test/java ⇒ where you run the test/code

**Package:**

* What is a package? ⇒ Package is a collection of classes.
* Naming convention ⇒ All letters (at least the beginning letter) should be lowercase

**Class:**

* What is a class? ⇒ Class is a collection of objects.
* Naming convention ⇒ The first letter should be uppercase
* Class is not real, it’s logical.
* Real life example ⇒ Animal, Human

**Object:**

* What is an object? ⇒ Object is a part (instance) of Java. (Object is a part of class)
* Object is inside the class
* Naming convention ⇒ First letter must be lowercase
* Real life example ⇒ cat, man, dog, myDog
* The creation of an object (instance) is called “Instantiation”
* When an object needs to be created?
  + If the method is **NOT** **static**

Formula for creating object:

ClassName objectName = new ClassName();

**Class classification/types:**

Name of a class/interface = 1st letter must uppercase

1. Outer class = 1st class
2. Inner class = inside the 1st class
3. Abstract class = Abstract keyword
4. Non abstract class/Normal class = abstract keyword **not** there
5. Interface = Class like structure/ similar like class

| **Outer class** | **Inner class** |
| --- | --- |
| 1st class created | Class inside class |
| Cannot use static | Can use static |

| **Abstract class** | **Non abstract class/Normal class** |
| --- | --- |
| Abstract keyword | No abstract keyword |
| Can handle both abstract & non-abstract methods | Can handle only non-abstract method |

| **Class** | **Interface** |
| --- | --- |
| Collection of objects | Similar structure like class |
| Class keyword | Interface keyword |
| Class can create object (instantiation) | Interface cannot create object (instantiation not possible) |

**Static meaning in java:**

* Static means **class** level
* Static method **no** need to create object
* Static method always needs static variable
* How to call static method:
  + ClassName.methodName();
  + methodName(); [when calling method from the same class]

**Method (Function):**

Method= is a piece of code

\*\*\* method always go inside the class

Parts:

* Method name/signature
* Parenthesis after the method name = ()
* Body or description ={//code}
* Is method return anything or not (return type)
  + Method returns nothing = void word before method name
  + Method returns something =
    - no void keyword
    - return statement

**Method Types:**

* **Method structure:**
  + Non-static void method (non-return method):
    - public void getVoidMethod() {

//write the code

}

public ⇒ Access modifier

void ⇒ keyword for void (non-return) method

getVoidMethod ⇒ method name

() ⇒ method signature

{} ⇒ Inside we write the code

* Static void method:
  + public static void getStaticVoid() {

//write the code

}

public ⇒ Access modifier

static ⇒ keyword for static method (no need to create an object)

void ⇒ keyword for void (non-return) method

getStaticVoid ⇒ method name

() ⇒ method signature

{} ⇒ Inside we write the code

* Abstract method:
  + public abstract void getAbstract();
    - abstract keyword should be present
    - There should be no {} (no method body)
    - Abstract method can be handled by abstract class and interface
* Non-void (return) & non-static method :
  + public getReturnMethod() {

//write the code

return something;

}

* Static return method:
  + public static getStaticReturnMethod() {

//write the code

return something;

}

* Main method:
  + Runs the java methods
  + public static void main(String[] args) {

//code (method body)

}

**Method Types:**

If static keyword ⇒ static method (**no** need to create an object)

If **no** static keyword ⇒ non-static method (**need** to create an object)

If void keyword ⇒ void method (non-return)

If no void keyword ⇒ return method (return statement)

If abstract keyword ⇒ abstract method

**Data types:**

See slide #43

Data are two types:

1. Primitive
   1. int ⇒ whole number ⇒ 1, 5, 34, 125 ⇒ Default value is 0
   2. double ⇒ fractional value ⇒ 10.25, 2.56 ⇒ Default value is 0.0
   3. boolean ⇒ true or false condition ⇒ Default value is false
2. Non-primitive
   1. String ⇒ any value with “double quote” ⇒ Default value is null

**Variable:**

Variable ⇒ Container where the data is stored

Variable>> int





How to write variable name ⇒ dataType variableName = value;

E.g. ⇒ int a = 6;

**Variable type:**

1. **Instance Variable** (Inside the class but outside the method)
   1. **Static variable** (Class level variable)
      1. Only handled by static method
   2. **Non-static variable**
      1. Can be handled by non-static method
2. **Local Variable** (Inside the method)

Interface ⇒ final Variable: public static final a=45;

Class A implements Interface ⇒ Variable a=45;

Class B implements Interface ⇒ Variable a=45;

**Access Modifier:**

See slide # 58 - 64

|  | **Another class** | **subclass** | **Same package – all class** | **Other package – all class** |
| --- | --- | --- | --- | --- |
| **Public** | Y | Y | Y | Y |
| **protected** | Y | Y | Y | Only subclass |
| **private** | No (only one class) | No | No | No |
| **default** | Y | No | Y | No |

| Public | Any package & any class |
| --- | --- |
| private | Only same class |
| protected | In sub class/child class & same package |
| Default | Same package |

**Parameter:**

Anything inside the method signature is a parameter.

Example:

public void getMoney(int a, int b){ //method body }

**Constructor:**

See Java Q&A pg 21 - 22

**What is a Constructor? ⇒**

1. It is a special type of method.
2. Constructor name must be the same as its class name.
3. Constructor should not be return type or void type.

**Why? ⇒**

1. To eliminate default value
2. Construct user define value
3. When we create object, constructor invoke automatically

**Java OOPs (Object Oriented Programming) Concept**

**Inheritance, Abstraction, Encapsulation, Polymorphism**

**Inheritance:**

Java Q&A pg 11 - 14

**What ⇒** Inheritance is a part of Java OOPs which allow to get common properties (method & reference variables) from one class to another.

**Why ⇒** To transfer common methods & variables (properties) from one class to another class.

**How ⇒**

1. Child Class **extends** Parent Class
2. Interface **extends** Interface
3. Class **implements** Interface

Parent class ⇒ Super class ⇒ Who has the property

Child class ⇒ Sub-class ⇒ Who gets the property

Inherit between class to another class/abstract class ⇒ extends

1st way to create an object ⇒ParentClass obj = new ChildClass();

2nd way to create an object ⇒ChildClass obj = new ChildClass();

Inherit between class & interfaces ⇒ implements

**How many types of Inheritance in Java ⇒**

**Three types:**

1. **Single level** (means child class extends parent class)
   1. Possible in class level
   2. Possible in abstract class level
   3. Possible in interface level
2. **Multi-level** (means Parent class extends Grand-Parent >>> Then child class extends Parent class)
   1. Possible in class level
   2. Possible in abstract class level
   3. Possible in interface level
3. **Multiple level** (means one class gets properties from multiple classes)
   1. NOT possible in class level or abstract class level
   2. But possible at the interface level.

Class has method **declaration &** **implementation**

Interface has only method **declaration** (no implementation)

**Deadly diamond issue:**

See Java Q&A q#54 (pg 13)

* Multiple inheritance is possible at Interface level, because interface has only method declaration but no implementation.
* Multiple inheritance is NOT possible at class level, because class has both method declaration and implementation.
* So, if multiple inheritance happens at the class level, JVM will get confused and crash.

**Encapsulation:**

See Java Q&A pg 20 - 21

**What ⇒** Binding the variable & method in a single unit to protect the code from the outside (unauthorized access)

**How ⇒**

1. Variables should be private
2. Use getter & setter methods with public
   1. Right click ⇒ Source ⇒ Generate getter & setter
      1. Getter method is return type
      2. Setter method is void type

**Polymorphism:**

See Java Q&A pg 17 - 20

1. Poly ⇒ multiple
2. Morphism ⇒ structure

**What ⇒** same thing have different form or structure

Real life example: Water (liquid), solid (ice), gas (steam)

**Java ⇒** Using same method again and again

**Types of polymorphism:**

2 Types of polymorphism ⇒

**1. Method Overloading** **(Compile time polymorphism/ Static binding)**

**2.** **Method Overriding (Runtime polymorphism/ Dynamic binding)**

1. **Overloading (Compile time polymorphism/ Static binding):**
   1. Inside the same class
   2. Using the same method name again and again
   3. By changing the parameter
      1. Change the number of parameters
      2. Change the position of parameters
      3. Change the data type of parameters
2. **Overriding (Runtime polymorphism/ Dynamic binding):**
   1. It happens between two or more classes (Inheritance is involved)
   2. Method name should be the same
   3. Where/when to use
      1. Class extends abstract class (abstract method)
      2. Class implements interface (abstract method)

**Table: Difference between Overloading & Overriding**

| **Topic** | **Overloading** | **Overriding** |
| --- | --- | --- |
| Class | one | Two or more |
| Parameter | Must be changed | Must be the same |
| Inheritance | No | Yes |
| Return type | May or may not be the same | Must be the same |

**Which class can handle what type of methods:**

Class ⇒ Only non-abstract methods

Abstract class ⇒ Both abstract & non-abstract methods

Interface ⇒ Only abstract method

**Inheritance keywords:**

Between class to class or class to abstract class ⇒ extends

Between interface to interface ⇒ extends

Between class to interface ⇒ implements

**Abstraction:**

See Java Q&A pg 14 - 17

**What ⇒** Hiding the details and showing the outcome

**How ⇒**

1. Abstract class ⇒ abstraction 0 - 100%
   1. If 1 out of 2 method is abstract ⇒ abstraction is 50%
   2. If 3 out of 4 method is abstract ⇒ abstraction is 75%
   3. If 6 out of 10 method is abstract ⇒ abstraction is 60%
2. Interface ⇒ abstraction 100%

**Wrapper Class:**

Wrapper class is a class which contains primitive data.

| **Wrapper Class** | **Primitive Data Type** |
| --- | --- |
| Integer | int |
| Double | double |
| Boolean | boolean |

**String**

What ⇒

1. String is a class in Java.
2. Also a non-primitive data type.

**Difference among String, StringBuffer, & StringBuilder:**

Java q# 26 (pg 4).

**Java Condition:**

**If & else:**

See Java Q&A pg 8.

if (condition) {

//code

} else {

//code

}

**Java Loop:**

Java Q&A pg 8

**Basic For Loop:**

**Positive for loop ⇒**

//FORMULA ⇒ for(start;end;++) position = 0,1,2,3

CODE ⇒

for(int i=0;i<=100;i++) {////positive for loop/looping

System.out.println(i); }

**Negative for loop ⇒**

//FORMULA ⇒ for(end;start;--) position 3,2,1,0

CODE ⇒

for(int j=100;j>=0;j--) {//negative for loop/looping

System.out.println(j); }

**How to create an infinite loop ⇒**

Just remove the middle part (i.e. end point for positive loop & start point for negative loop)

**How to stop a loop ⇒**

CODE ⇒

for(int i=0;;i++) {

System.out.println(i);

if(i == 30) {//condition

break;

}

}

**How to reverse ⇒**

CODE ⇒ (Basic Java)

public static void getStringReverse(String name) {

for(int i=(name.length()-1);i>=0;i--) {

System.out.println(name.charAt(i));

}

}

CODE ⇒ Using StringBuilder or StringBuffer (Use reverse() method) [N.B. This will print in one line]

public static void getReverseWithStringBuilder(String name) {

StringBuilder obj = new StringBuilder(name);

System.out.println(obj.reverse());

}

**Java Exception:**

See Java Q&A pg 27 - 32

Dictionary meaning: Exception is an abnormal condition.

In Java, exception is an event that disrupts the normal flow of the program or methods.

**Exceptions are two types:** Checked (compile time) exception and Unchecked (runtime exception).

**Table: Difference between Checked Exception & Unchecked Exception**

| **Topic** | **Checked Exception** | **Unchecked Exception** |
| --- | --- | --- |
| When happened | Before running the code. | After running the code. |
| Compilation error (red color in code line) | Yes | No |
| Another name | Compile time exception | Runtime exception |
| How to handle | Try-catch block or throws | Only try-catch block |
| Example | Interrupted exception, IO Exception, SQL Exception, FileNotFound Exception, ClassNotFound Exception. | Null pointer exception, IndexOutOfBound Exception, Arithmetic exception |

**Try/catch block:**

try {//code

System.out.println(a/0);

} catch (Exception e) {//handling the exception

e.printStackTrace();

}

**Table: Difference between Exception & Error**

| **Exception** | **Error** |
| --- | --- |
| Exceptions can be handled or recovered. | Errors cannot be handled or recovered. |
| Exceptions are two types:   1. Checked Exception 2. Unchecked Exception | There is no such classification for errors. Errors are always unchecked. |
| In case of checked Exceptions, the compiler will have knowledge of checked exceptions and force to keep try-catch blocks. | In case of Errors, the compiler will not have knowledge of errors. |

**Table: Difference between throw & throws**

| **throw** | **throws** |
| --- | --- |
| “Throw” is used within the method. | “Throws” is used with the method signature. |
| “Throw” is followed by an instance. | “Throws” is followed by class. |
| You cannot throw multiple instances. | You can declare multiple exceptions; e.g. public void method() throws IOException, InterruptedException. |

**\*\*\*How do you handle exceptions in your project?**

* There are 2 types of exception: Checked Exception & Unchecked Exception.
* Checked Exceptions can be handled by try/catch block and throws.
* Unchecked Exceptions can be handled by try/catch block only.
* I use try/catch block as it’s a better option. It can be used for both types of exception.

**Java Collections:**

Java Q&A pg 22 - 27

What? ⇒ A group of objects in Java

Why do we need Collections? ⇒

1. Variable ⇒ can store only single value
2. Arrays ⇒ To handle multiple values, faster but the length is fixed
3. Collections ⇒ To handle multiple values, slower than array but dynamic

**Arrays:**

* Array is an object in Java.
* Data type is primitive (int, double, boolean)
* It is faster.
* The length is fixed.
* Arrays is two types ⇒
  + Single dimensional array
  + Multidimensional array
* More memory is used
* To create a **single** dimensional array, after the data type add “[]”.
  + To create a **multi** dimensional array, after the data type add “[][]”
* All the data will be inside {}

**Collections:**

* Data type is not primitive. Wrapper class object (Integer, Double, Boolean).
* The size is dynamic. (Can add/delete data)
* Performance is slower than Arrays.
* Less memory is used

**Table: Java collections names**

| **Interface** | **Why need to use** | **Classes inside interface** |
| --- | --- | --- |
| List (dynamic array) | * Search data faster * Allow as many as null/duplicate values | ArrayList = search (random access) faster (Index theory), but add/remove data is slower.  Linked List = Add/remove faster, but search is slow (Double linked list theory)  Vector = legacy class, synchronize (threadsafe)  Stack = LIFO (Last in fast out) |
| Set (dynamic array) | * Do not allow duplicates or when need to remove duplicate values | Hash Set = no order ⇒ faster  Linked Hash Set = insertion order  Tree Set = Ascending order |
| Map (internally set) | * Allow value pair (key & value). E.g. name = Michael   City= Bronx. | Hash Table = Synchronize (threadsafe)  Hashmap = no order ⇒ faster  Linked Hash Map = insertion order  Treemap = Ascending order |

Interface ⇒ Cannot create object (instantiation is not possible)

* Which collection do you use in your current project?
  + ⇒ Based on my requirements, I used List, Set, & Map.

Formula to write collection object:

Interface<WrapperClass> collectionObject = new ClassName<>();

Example:

List<Integer> myList = new ArrayList<>();

Map<String, String> myMap = new LinkedHashMap<>();

List & Set ⇒ To add values, we use add() method

Map ⇒ To add values, we use put() method

New York ⇒ Chicago (400 miles)

0 100 200 300 400

[2, 4, 4, 5, 8, 9]

Index theory:

0 ⇒ 2

1 ⇒ 4

2 ⇒ 4

3 ⇒ 5

4 ⇒ 8

5 ⇒ 9

Double linked list:

2 <-> 4 <-> 4 <-> 5 <-> 8 <-> 9

**Table: Difference between Array and ArrayList**

| **Topic** | **Array** | **ArrayList** |
| --- | --- | --- |
| Resizable | No (size is fixed) | Yes (size is dynamic) |
| Data type | Primitive (int, double, boolean) | Not primitive; wrapper class object (Integer, Double, Boolean). |
| How to find the Length | Length variable | size() method |
| Performance | Faster | Slower than array |
| Multidimensional | Yes | No (Collections is not multidimensional) |

Study all the differences in the Java Q&A.

**Example of Array:**

public class PracticeArray {

static int a =2; //variable => cannot handle more than one value

static int[] b = {10, 20, 30, 50}; //array (single dimensional)

static int[] c = {2,3,5,8}; //array (single dimensional)

static int[][] d = {{10, 20, 30, 50},{2,3,5,8}}; //array (multidimensional) d = {b,c}

public static void main(String[] args) {

System.out.println(a);

System.out.println(Arrays.toString(b));

System.out.println(Arrays.toString(c));

System.out.println("Length of b="+b.length);

System.out.println(Arrays.deepToString(d));

}

}

**Example of List (ArrayList):**

public class PracticeArrayList {

public List<Integer> getArrayValue() {

List<Integer> myList = new ArrayList<>(); //upcasting >> add 2, 4, 4, 5, 8, 9

//To insert value in List, we use add() method

myList.add(2);

myList.add(4);

myList.add(4);

myList.add(5);

myList.add(8);

myList.add(9);

System.out.println(myList);

return myList;

}

public static void main(String[] args) {

PracticeArrayList obj = new PracticeArrayList();

obj.getArrayValue();

}

}

**Example of Set (HashSet):**

public class PracticeHashSet {

Set<Integer> mySet = new HashSet<>(); //remove duplicate 2, 4, 4, 5, 8, 9, 9

public Set<Integer> getArrayValue() {

//To add value in Set, we use add() method

mySet.add(2);

mySet.add(4);

mySet.add(4);

mySet.add(5);

mySet.add(8);

mySet.add(9);

mySet.add(9);

System.out.println(mySet); //{2, 4, 5, 8, 9}

return mySet;

}

public static void main(String[] args) {

PracticeHashSet obj = new PracticeHashSet();

obj.getArrayValue();

}

}

**Example of Map (Linked Hash Map):**

public class PracticeLinkedHashmap {

public static void main(String[] args) {

Map <String, String> myMap = new LinkedHashMap<>(); //add Name & City

myMap.put("Name", "Michael");

myMap.put("City", "New York");

System.out.println(myMap);

}

}

**Excel with Apache POI:**

* What is data?
  + Any information
* What is test data?
  + Any data related to testing
* How to get test data/which format?
  + Excel ⇒ Apache POI (JAR/lib/dependency)
  + Database ⇒ SQL language & connected with automation by JDBC connection
* Excel has old & new versions
  + Before 2007 ⇒ old ⇒ .xls ⇒ POI understands it as HSSF
  + After 2007 ⇒ new ⇒ .xlsx ⇒ POI understands it as XSSF
* Add the JAR for Apache POI

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>5.2.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.2.3</version>

</dependency>

Where do you get the JAR? ⇒

* School/practice ⇒ Maven repo
* Office ⇒ Maven repo/Office repo or artifactory

**Table:** **Difference between Read Excel & Write Excel**

| **Read Excel** | **Write Excel** |
| --- | --- |
| Data from Excel to automation | From automation to excel |
| FileInputStream class | FileOutputStream class |

**FileInputStream ⇒ serialization process** (Data ⇒ stream/byte code/machine readable code)

**FileOutputStream ⇒ deserialization process** (stream/byte code/machine readable code ⇒ Data)

* **How to read the excel in automation?**
  1. Pom file >> add Apache POI dependency file
  2. Create & save the Excel
  3. 1st need to find the Excel file path
  4. Convert data to stream ⇒ FileInputStream class
  5. Call my excel as workbook ⇒ create a workbook object
  6. Call my Sheet as worksheet & get my sheet by index or sheet name ⇒ workBookObject.getSheetAt(i) OR workBookObject.getSheet(“Sheet Name”)
  7. Find all row ⇒ sheetObject.getLastRowNum() or sheetObject.PhysicalRowNum()
  8. Loop all rows
     + To get row⇒ rowObject.getRow(i);
  9. Find all columns ⇒ rowObject.getLastCellNum()
  10. Loop all columns
  11. Get Cell value ⇒ rowObject.getCell(column index)
  12. Format data with Java DataFormatter

**The code:**

public class PracticeExcel {

FileInputStream fis; // FileInputStream object

XSSFWorkbook wb; // Workbook object

public void getExcelData(int rowIndex, int columnIndex) {

// Excel location or path

File file = new File("./src/main/resources/LogIn\_Test\_Cases.xlsx");

// Convert data to stream

try {

fis = new FileInputStream(file);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

// Apache POI methods to read the excel

// 1st step = read whole excel as a book = workbook

try {

wb = new XSSFWorkbook(fis);

} catch (IOException e) {

e.printStackTrace();

}

// Call my Sheet as worksheet & get my sheet by index or sheet name

XSSFSheet sht = wb.getSheetAt(0);

XSSFSheet sht1 = wb.getSheet("Practice1");

// Find all rows

int totalRow = sht.getPhysicalNumberOfRows();

// Loop all the rows

for (int i = 0; i < totalRow; i++) {

if (i == rowIndex) {

Row row = sht.getRow(i);

// Total number of columns

row.getLastCellNum();

// start another loop for column

for (int j = 0; j < totalRow; j++) {

if (j == columnIndex) {

Cell cell = row.getCell(j);

System.out.println(cell.getStringCellValue());

}

}

}

}

}

public static void main(String[] args) {

PracticeExcel obj = new PracticeExcel();

obj.getExcelData(5, 1);

}

}

**Database with JDBC (DB Automation):**

How ⇒

1. Lib/JAR/Connection JDBC.
2. I have to create a Connection string and pass
   1. DB URL
   2. Username
   3. Password
3. Create a Statement object.
4. Create a ResultSet object and pass my query.
5. Now all data will be stored inside the ResultSet object.
6. Now loop the ResultSet object and get data.

CODE ⇒ (No need to memorize)

public class Database {

public static void main(String[] args) {

Connection connection = null;

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

connection = DriverManager.getConnection("Data base URL", "user", "pass");

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_READ\_ONLY);

ResultSet rs = stmt.executeQuery(

"Select \* From employees");

ResultSetMetaData rsmd = rs.getMetaData();

int columnsNumber = rsmd.getColumnCount();

// Iterate through the data in the result set and display it.

ArrayList<String> columnValue = new ArrayList<String>();

while (rs.next()) {

//Print one row

for(int i = 1 ; i <= columnsNumber; i++){

System.out.print(rs.getString(i) + " "); //Print one element of a row

columnValue.add(rs.getString(i) );

}

System.out.println();//Move to the next line to print the next row.

}

} catch (Exception e) {

e.printStackTrace();

} finally {

// step5 close the connection object

try {

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

**Q. What do you know about Java?**

**Ans.** Discuss the following:

1. What is Java?
2. Basic Java
   1. Class
   2. Method
   3. Variable
   4. Data
   5. String
   6. Access Modifier
   7. Wrapper Class
   8. Java Condition
   9. Java Loop
3. Core Java
   1. OOPs Concept
      1. Inheritance
      2. Encapsulation
      3. Polymorphism
         1. Method Overloading
         2. Method Overriding
      4. Abstraction
   2. Exceptions
      1. Checked Exception
      2. Unchecked Exception
   3. Java Collections
      1. List
      2. Set
      3. Map

**Selenium:**

**How to setup the driver:**

**1st way:**

1. Create a folder named “Driver” in the project.
2. Download the specific driver for the browser.
3. Extract the driver.
4. Save the driver file in the Driver folder.
5. //System.setProperty("package name", "driver path");

System.setProperty("webdriver.chrome.driver", "./Driver/chromedriver\_v107.exe");

To visit any URL, you can use get() or navigate().

**Table: Difference between get() and navigate()**

| **get()** | **navigate()** |
| --- | --- |
| URL | URL |
| No | Go back |
| No | Go forward |
| No | Refresh the page |

**To inspect a webelement:**

1. Right click on the web element.
2. Click on “Inspect”.
3. Press Ctrl + F
4. Type the Xpath inside the textbox.
   1. //tagname[@locator=’value’]
      1. //\*[@locator=’value’]
      2. //\*[@name=’userName’]
   2. //tagname[text()=’value’]
      1. //\*[text()='SIGN-OFF']
   3. //tagname[contains(text(),’value’)]
      1. //\*[contains(text(),'Sign in')]
   4. To use multiple locators at the same type using “and”:
      1. //tagname[@locator=’value’ and @locator=’value’]
   5. To use multiple locators using “or”
      1. //tagname[@locator=’value’ or @locator=’value’]
   6. Copy the xpath
5. Formula to use CSS Selector:
   1. If using ID ⇒ tagname#value
   2. If using class ⇒ tagname.value
6. Go back to IDE
7. driver.findElement(By.xpath("//\*[@locator=’value’]")); [yellow is xpath]

**Grouping:**

When there are more than one webelement with the same locator value, then we use Grouping.

Example:

(//\*[@class='oxd-userdropdown-link'])[4]

**Difference between quit() & close()**

See Q#92 in the Selenium Q&A.

**Code for Highlighter class:**

public static void addColor(WebDriver driver, WebElement element){

JavascriptExecutor js= (JavascriptExecutor)driver; //type casting

js.executeScript("arguments[0].setAttribute('style', arguments[1]);",element, "color: red; border: 2px solid red;");

}

**Page Factory (PF):**

* What is a Page Factory?
  + A repository (storage) for all the xpath.
* Encapsulation ⇒
  + Binding the method & variables together to protect it from unwanted access
* Formula in Page Factory:

@FindBy(xpath="//\*[@class='login']")

private WebElement signinbtn;

@FindBy ⇒ FindBy annotation

//\*[@class='login'] ⇒ Xpath formula

Private ⇒ Access Modifier

WebElement ⇒ Data type

Signinbtn ⇒ Variable name

* Right click on PF >> Source >> Generate Getters & Setters >> Check “Select Getters” > Click “Generate”.
* Add constructor with the following code:

public MasterPageFactory(WebDriver driver) {

PageFactory.initElements(driver, this);

} //This constructor will handle the null pointer exception

**Important questions regarding Page Factory (PF) & Page Object Model (POM):**

* Tell me about the Selenium framework?
  + Selenium framework using Page Factory (aka Page Object Model)
* Differences between PF & POM

**Table: Differences between PF & POM**

| **Page Factory (PF)** | **Page Object Model (POM)** |
| --- | --- |
| New concept | Old concept |
| @FindBy annotation is used to search | By() method is used to search |
| Faster | Slower |
| Handle bigger application (more than 500 web elements) | Handle small application (less than 500 web elements) |

Nowadays, PF & POM both are the same. Both of them are known as “Page Object Model Framework”.

* **How to build PF?**
  + Use Java OOPs concept (Encapsulation)
  + All web elements will be private access modifier
  + Generate getter method
  + Call the getter method in the code
* **Do you face any challenges in PF?**
  + Yes, I face null pointer exception
* **How to overcome null pointer exceptions?**
  + Create Java constructor
  + Use PageFactory.initElements inside the constructor

**To import all the files**: shortcut ⇒ Ctrl + Shift + O

**Code for Screenshot class:**

public static void getScreenShot(WebDriver driver, String name){

File scrFile1 = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

try {

FileUtils.copyFile(scrFile1, new File("./ScreenShots/"+name+".png")); } catch (IOException e) {

e.printStackTrace();

}

**Code for the BaseConfig class (to read Config.properties file):**

public static String getConfigValue(String value) {//return method

//file location

File file = new File("./Config.properties");

//file data ==> stream

FileInputStream fis = null;

try {

fis = new FileInputStream(file);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

//to read each value

Properties pro = new Properties();

try {

pro.load(fis);

} catch (IOException e) {

e.printStackTrace();

}

System.out.println(pro.getProperty(value));

return pro.getProperty(value);

}

public static void main(String[] args) {

getConfigValue("URL");

getConfigValue("Username");

getConfigValue("Password");

}

}

**Difference between Selenium & QTP/UFT:**

See Q# 10 from Selenium Q&A.

**To open browser code in Selenium version 4.6.0 or later:**

Instead of

//System.setProperty("package name", "driver path");

We can do the following:

1. Add WebDriverManager by Boni Garcia to the pom.xml file from the Maven Repo.
2. Write the following coding: WebDriverManager.chromedriver().setup();

**Relative xpath & Absolute xpath:**

See Selenium Q&A q# 52 (pg 9)

USA/New York/Brooklyn/Ozone Park/Euclid St. ⇒ Absolute path

//Euclid St. ⇒ Relative path

OrangeHRM/Credential123/Login/Username ⇒ Absolute path

//Username ⇒ Relative path

**Selenium Exceptions:**

* Element Not Found Exception or NoSuchElementException:
  + This exception will be thrown when:
    - Xpath wrong
    - Application down
    - Server down
    - Slow net
  + How to solve it?
    - Slow net ⇒ Selenium Wait (Explicit Wait)
    - Xpath wrong ⇒ correct the Xpath
    - Application or server down ⇒ inform your dev or the Help Desk
* StaleElementException:
  + This exception is thrown:
    - Xpath is right
    - But the xpath is hidden inside the HTML DOM
  + How to handle it?
    - Try any Java loop & check the locator for more than 4 times OR
    - Use Fluent Wait
* WebDriverException (driver issue)
* IllegalStateException (version issue)
* TimeoutException (explicit wait = 10 secs)
* NoAlertPresentException (there is no alert pop up)
* NoSuchWindowException

**Selenium Wait:**  
See Selenium Q&A pg 13.

Updated code for Explicit Wait:

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));

**How to open other browsers:**

See Selenium Q&A q# 24 (pg 5).

**Headless/Ghost browsers:**

See Selenium Q&A q# 29 (pg 6).

**Challenges using Internet Explorer & Firefox:**

See Selenium Q&A q# 30 (pg 6).

**Differences between Xpath & CSS Selector:**

See Selenium Q&A pg 9 & q# 58 (pg 11).

**How to find an Xpath or locator from a web page table:**

See Selenium Q&A q# 57 & q# 58 (pg 10).

**Xpath methods:**

See Selenium Q&A pg q# 63 (pg 11).

**Code for clicking using Actions class:**

Actions ac = new Actions(driver);

ac.click().perform();

ac.clickAndHold().perform();

ac.contextClick().perform(); //to right click

ac.doubleClick().perform();

ac.moveToElement().perform(); //for mouse-over or mouse hover

ac.dragAndDrop(drag, drop).perform();//to drag and drop

**Code for clicking using JavascriptExecutor (to click on hidden object):**

JavascriptExecutor js = (JavascriptExecutor)driver; //type casting

js.executeScript(“java script”,mpf.getSigninbtn());

**How to handle the drop down menu:**

Select sc = new Select(driver.findElement(By.name(“country”)));

sc.selectByVisibleText(“BANGLADESH”);

sc.selectByIndex(i);

sc.selectByValue(“CANADA”);

**How to handle alert (small window that’s not a part of the browser)**

⇒ CODE (By using Alert interface)

Alert alert = driver.switchTo().alert();

alert.accept(); alert.dismiss(); alert().getText(); alert().sendKeys();

**Situations that Selenium cannot handle by itself:**

See Selenium Q&A q# 88 (pg 18) and q# 92 (pg 19).

**How to handle multiple windows in Selenium:**

See Selenium Q&A q# 97 (pg 19).

**Difference between Assert & Verify:**

See Selenium Q&A q# 103 (pg 21).

**How to handle cookies in Selenium:**

See Selenium Q&A q# 114 (pg 26 - 27).

⇒ First, in the MasterPageFactory

@FindBy (xpath="xpath formula")

private List<WebElement> acceptCookiesBtn;

CODE ⇒

if(mpf..getAcceptCookiesBtn().size()>0) {//list total no = size()

Highlighter.addColor(driver, mpf.getAcceptCookiesBtn().get(0));

wait = new WebDriverWait(driver, 30);

// wait.until(ExpectedConditions.elementToBeClickable(mpf.getAcceptCookiesBtn()));

//

//

// Screenshot.getScreenShot(driver, "Accept all cookies option");

mpf.getAcceptCookiesBtn().get(0).click();

}

**Questions related Automation Framework:**

See Selenium Q&A (pg 22 - 27).

**Login credentials for Zoopla:**

URL= <https://www.zoopla.co.uk/house-prices/>

username= sarowerny@gmail.com

password= student2020

**TestNG:**

STCM(Easy to remember)

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

**How to run group testing in TestNG?**

CODE ⇒

⇒ @Test(groups = {“Smoke”})

⇒ Create testng.xml then after <test> write:

<groups>

<run>

<include name="Smoke"></include>

</run>

</groups>

Then run as TestNG suite

**How to set priority in TestNG?**

1. ⇒ @Test(priority = 0),@Test(priority = 1)
2. With both grouping and prioritizing

⇒ @Test(groups = {“Smoke”},priority = 0)

**How to run only failed tests in TestNG?**

⇒ Go to test-output folder > testng-failed.xml > right click > “run as TestNG suite”

**How to not run any test cases in TestNG?**

⇒ @Test(enabled = false)

**How to run test #5 if test #4 passes in TestNG?**

⇒ dependsOnMethods

@Test(dependsOnMethods = {“checkEnvironment”})

**How to pass parameters in TestNG?**

TestNG generally does not allow parameters. To allow parameter, follow the steps below:

⇒ @Test

@Parameters(“myname”) //name from xml

⇒ create testng.xml

⇒ after <test> write:

<parameter name = “myname” value = “Rezwan”></parameter>

<classes>

<class name = “packageName.className”></class>

<classes> (Then run as TestNG suite)

\*\*\*To run multiple parameters, copy from <test> to </test>

**Differences between Cross Browser Testing & Parallel Browser Testing:**

See TestNG Q&A q# 20 (pg 5).

**Coding to run cross browser testing:**

public void crossBrowserLogin(String browser) {

if (browser.equalsIgnoreCase("chrome")) {

WebDriverManager.chromedriver().setup();

driver = new ChromeDriver();

System.out.println("Browser opened=" + browser);

} else if (browser.equalsIgnoreCase("firefox")) {

WebDriverManager.firefoxdriver().setup();

driver = new FirefoxDriver();

System.out.println("Browser opened=" + browser);

} else {

System.out.println("Not found any browser");

}

**Differences between Assert & Verify:**

See Selenium Q&A q# 103 (pg 21)

Some of the Assert method I use from TestNG to do assertion:

1. Assert.assertEquals(actual value, expected value)
2. Assert.assertTrue(true condition, string message)
3. Assert.assertFalse(false condition, string message)

**How to create report in TestNG:**

1. Import ExtentReports to the pom.xml

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>5.0.9</version>

</dependency>

1. Add the following coding as variables:

ExtentReports extent;

ExtentTest test;

1. Add the following coding inside @BeforeTest

ExtentSparkReporter reporter = new ExtentSparkReporter("target/Batch30.html");

reporter.config().setDocumentTitle("Testng Automation");

reporter.config().setReportName("Smoke test");

reporter.config().setTheme(Theme.DARK);

extent = new ExtentReports();

extent.attachReporter(reporter);

extent.setSystemInfo("Company ", "SMART TECH");

extent.setSystemInfo("Testing environment ", "QA");

extent.setSystemInfo("Team ", "QA");

extent.setSystemInfo("Tester Name: ", System.getProperty("user.name"));

1. Add the following coding inside @Test

test = extent.createTest("Login test");

1. Add the following coding inside @AfterTest

extent.flush();

**GitHub:**

**GitHub (Similar tools: Bit Bucket, TFS):**

* What is GitHub?
  + It’s a version control tool.
  + It stores all old versions of the code.
  + Created 1st time = Login
    - QA > UAT > Prod ⇒ Release ⇒ Version 1.0 (bug)
    - Fix the bug ⇒ QA ⇒ Released in the next version ⇒ 1.1 (bug fixed)
    - If any major issue happens with the next release, then roll back to the previous version of the coding.
* Why do we use Github?
  + To store all old versions of coding (If anything bad happens in the future, we can roll back to the previous version).
  + To share the coding between team members.













3

**Conflict:**

A conflict arises when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other.

**How to prevent conflict (to not to make it happen):**

Before any push, always try to get the new code like below then push or merge

* Pull ⇒ Rebase ⇒ push/merge

**How to handle conflict (if a conflict happens, how to handle it):**

* Team>> synchronize workspace >> select conflict class

1. Override
2. Mark as merge

* If my code is the good one ⇒ then mark as merge
* If Git repo code is good ⇒ then override

Solve:

1. Rename the project or delete this project
2. File >> import>> git >> project from git>> clone URL>>
   1. Repo ULR
   2. User
   3. Pass
   4. Next

**SDLC:** Requirement ⇒ Design ⇒ Development ⇒ QA ⇒ UAT ⇒ Prod

Branches:

* Main branch = **Master branch** ⇒ Release (PO)
* Sub branch ⇒ create
  + Dev branch ⇒ Developers
  + QA branch ⇒ QA
  + Stage branch

**How to push the code to Github?:**

1. Open the project in Git bash
   1. Right click on the project >> Show in Local Terminal >> Git Bash
2. Write the following Git commands for the **initial push**:
   1. git init ⇒ To initialize the project in the Github
   2. git add . ⇒ To save/stash a copy of my code
   3. git commit -m “Initial push” ⇒ To write a commit message
   4. git branch -M master ⇒ The name of the branch that I want to push the code
   5. git remote add origin repoURL ⇒ The repo URL created in Github
   6. git push -u origin master ⇒ To push the code in the branch/sub-branch
3. If there is an issue with authentication (invalid username/password):

For username/email ⇒

git config --global user.name “github\_user”

git config --global user.email “github\_email”

git config --global credential.username “github user”

git config --global credential.email “github\_email”

(If still login issue, Control Panel >> User Account >> Credential Manager >> Manage WIndow Credential >> remove or add user

For password ⇒ Does not need password; in case it’s needed

git config --global user.password “github\_password”

1. Write the following git commands for pushing after the initial push:
   1. git add . ⇒ To save/stash a copy of my code
   2. git commit -m “Any logical message” ⇒ To write a commit message
   3. git push -u origin master ⇒ To push the code in the branch/sub-branch
2. How to pull code and rebase?

GIT BASH CODE ⇒

git pull --rebase

**Cucumber:**

* What? = tool/~~framework/API~~ ⇒ tool for Automation
* Why need? = BDD approach or BDD framework
* BDD = behavior driven development
* Other development
  + TDD = test driven development
  + ATDD = acceptance test driven development

**Table: Differences between BDD, TDD, & ATDD**

| **BDD** | **TDD** | **ATDD** |
| --- | --- | --- |
| Behavior Driven Development | Test Driven Development | Acceptance Test Driven Development |
| Focus = Business logic/need/requirement | Focus = testing | Focus = acceptance testing |
| Development and testing focus business need | QA discusses with dev how to test, based on the testing process>> dev will develop application.  Testing will easily, less time and less bug | Development and testing focus end user need or acceptance |

Advantage:

* Easy to understand for non-technical people (like BA, PM)
* Components
  + Feature file = manual steps ⇒ run test
  + Step definition = actual code
  + Runner file
    - To run test
    - Report
    - Integrate unit testing framework like JUnit or TestNG

**CUCUMBER setup:**

· Lib/JAR/dependency in pom file

o Vimal cucumber = old

o IO cucumber = new = school

· Cucumber plugin like TestNG plugin

· Src/test/resource >> file >> function.feature>> Login.feature

**Feature file:**

Manual test cases

* Feature ⇒ what feature/function I’m testing (what to test).
* Description ⇒ write some important requirement/need (optional)
* Background ⇒ Common step (optional)
* Scenario / Scenario Outline ⇒ detail steps (how to test)
  + Steps: Given + When + Then ⇒ Gherkin language
    - Given ⇒ Setup/Pre-action
    - When ⇒ Action (actual testing)
    - Then ⇒ validation (hard assert/assert or soft assert)
    - And / Or ⇒ same steps multiple
  + Example:
    - Given open the browser
    - And pass the URL
    - When enter the username
    - And enter the password
    - And click on Login
    - Then validate login is successful

**Step definition:**

* All the coding goes here
* Cucumber annotation
  + @Given
  + @When
  + @Then

**Runner file:**

* Why need it
  + Run the test
  + To generate report
  + Integrate TestNG/JUnit
  + To use Cucumber hook (TestNG annotation like @BeforeTest, @Test, @AfterTest)
  + Cucumber options
    - Plugin = report (json)
    - Feature = feature file path
    - Glue = step def package name
    - Cucumber hooks = TestNG annotation inside cucumber runner file
    - Monochrome = true (to remove all? mark and easy to read console)
    - dry run = To check feature file
    - strict = To check step def >> all manual steps has corresponding code
    - tag = specific scenario

**Dependency files:**

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/net.masterthought/maven-cucumber-reporting -->

<dependency>

<groupId>net.masterthought</groupId>

<artifactId>maven-cucumber-reporting</artifactId>

<version>5.7.0</version>

</dependency>

* **Scenario & Scenario Outline:**

**Scenario:** As Automation practice user, i can successfully login with valid credentials

*Given* open browser and send URL

*When* login with valid credentials

|sarowerny@gmail.com|student|<== Data table è one test data

Not allow multiple set of test data

Always run one time

*Then* Check login was success

**Scenario Outline:** As a Zoopla user, I can successfully login with valid credentials

*Given* Setup driver and send URL

*When* login with valid credential “<user>” and “<pass>”

*Then* validate login and close the browser

**Examples://example table**

|user |pass |

|sarowerny@gmail.com|student| 1st

|sarowerhome@gmail.com|123456| 2nd

Allow multi set data

1. **Runner file code:**

@CucumberOptions(publish = true,plugin = { "pretty", "html:target/site/cucumber-pretty", "rerun:target/rerun.txt",

"json:target/cucumber.json",

},

features = { "src/test/resources/AutomationExerciseLogin" },

glue = { "com.stepdef" }, // package name

monochrome = true, //remove all ?? & console will be easily readable

dryRun = true, //check feature file

strict = true //check stepdef

// ,tags= {"@Function"}

)

public class LoginRunner extends AbstractTestNGCucumberTests {

@BeforeTest

public void setup() {

// code

}

}

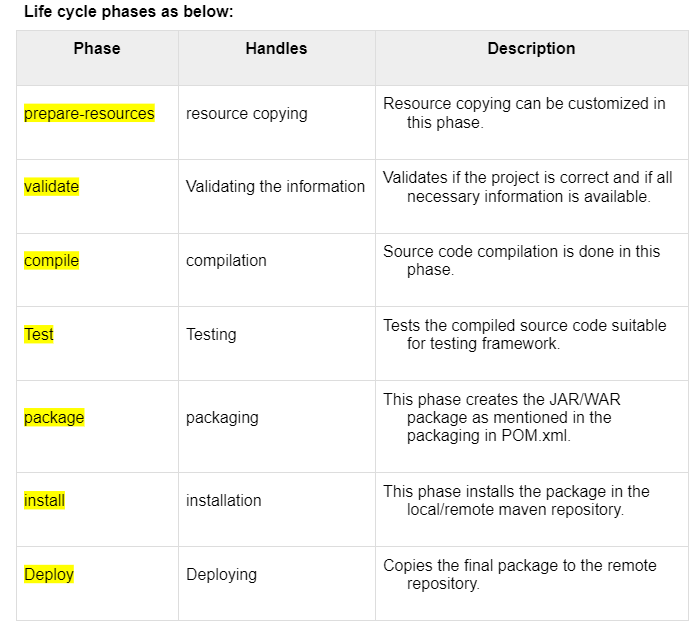
Link to practice automation: <https://www.automationexercise.com/>

Email: rezwanislam857@gmail.com

Password: Boston2023

**MAVEN:**

1. Add the Surefire plugin inside the pom file.
   * 1. To check maven build
     2. To help to generate report
2. Pass .java or testng.xml inside the Surefire plugin
3. Open terminal >> use maven goal
4. Maven goal (keyword maven) ⇒ To check the Maven build
   * 1. mvn clean =
        1. Remove all old JAR/lib/dependency files
        2. Empty target folder
     2. mvn install = install again all JAR/lib/dependency + run test cases
     3. mvn test = run test cases
5. Use combination:
   * 1. mvn clean install
     2. mvn clean test
6. **What’s the Maven lifecycle/ phases/ stages:**



**(Easy way to remember):**

**P V C T P I D**

**P** ⇒ Prepare resources

**V** ⇒ Validate

**C** ⇒ Compile

**T** ⇒ Test

**P** ⇒ Package

**I** ⇒ Install

**D** ⇒ Deploy

**Set-up Maven in Windows:**

1. Go to Apache Maven to download Maven:<https://maven.apache.org/download.cgi>

2. Download Binary zip archive for Windows.

3. Extract all.

4. Copy the Maven folder.

5. Paste it in C Drive > Program Files

6. Go inside the Maven folder > Copy the path

7. Go back to “Environment Variable”.

8. Under “User Variable”:

a. Click on “New”

b. Variable Name: MAVEN\_HOME

c. Variable value: copy & paste the Maven path

9. Go to the Bin folder in Maven > Copy the path.

10. Under the “System Variable”:

a. Select “Path”.

b. Click on “Edit”.

c. Click on “New”.

d. Paste the Maven/bin path

11. Open CMD and check if the Maven is downloaded successfully ⇒ mvn -version

**Jenkins:**

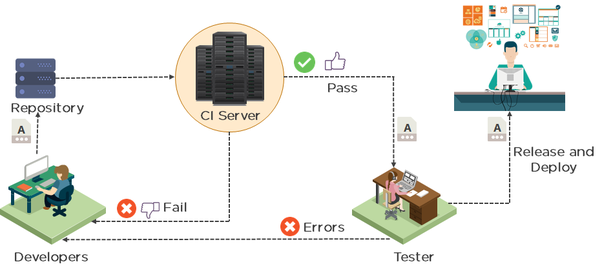
CI is a process that most developers follow to keep their code base intact. It's mostly a common practice when you work in a group environment. For example, Building a new house. There will be multiple contractors working on the site. So, if we have installed the window glasses and the paint person comes in and paints the house, there is a high chance that he will drop some paint on the glasses or break the glass. So, the inspector comes and checks it every day to see if something is broken or not. The same process is applied to constructing a new code(Software). CI system gathers all your code from different developers and makes sure it compiles and builds fine.

**Jenkins is the inspector,**

**Jenkins is nothing but a middleman between your code repo and your build server**. It checks for changes

On server every few minutes

If it finds them, it gathers them and sends them to your build server. That's what Jenkens



**Tell me about your CI/CD process:**

In my project, I have like 500 test cases in my regression suite

I need to set up Jenkins for continuous integration and schedule run

In the middle of the night after 1 am it will run and should send the email to the assigned person with build status

I checked the build status in the morning

If any test cases fail, I need to find out why it failed and Solve the problem and run the build.

* What is Jenkins? ⇒ Jenkins is a tool.
* What kind of tool? ⇒ CI (Continuous Integration) & CD (Continuous Deployment) tool
* CI = Continuous Integration
  + Means ⇒ Continuous process of integration
  + Integration testing ⇒ Testing two or more units together
  + Types of testing
    - Code testing
      * White box testing (code testing/unit testing)
      * Testing each unit of code
      * Single unit testing
    - Smoke testing ⇒ Testing of most critical & important functionalities at the initial build/stage

Test suite ⇒ Collection of test cases

Regression suite ⇒ Collection of regression test cases (old important test cases)

Sprint 1: total test cases = 14

* Smoke = 2
* Functional =
  + Positive = 10
  + Negative = 2
* Important test cases: (based on the requirement/business need) = 8
  + Separate regression test suite = 8

Sprint 2: total test cases = 46

* Smoke = 1
* Functional =
  + Positive = 21
  + Negative = 4
* Sanity = 20
* Important test cases = 32
  + Add these 32 test cases in regression suite = 8+32 = 40

Sprint 3: total test cases = 20

Important test cases = 6

Add this 6 test cases in the regression suite = 40 + 6 = 46

Regression run:

* Daily *(99%)*
* In every sprint

**Jenkins setup:**

* I set up Jenkins with the help of the DevOPS team.
* DevOPS
  + Networking
  + Maven + GitHub + Jenkins = QA
  + Unix
  + Shell scripting
* Where do you set up the Jenkins? / What kind of server
  + Server
    - Windows server (physical server) ⇒ current project
    - Linux server (physical)
  + Cloud server
    - AWS (Amazon Web Server)
    - HP
    - Dell
    - Google
  + Container
    - Docker container ⇒ Linux server
    - S3 bucket ⇒ AWS ⇒ My company is trying to move to the AWS S3 Bucket
* Tell me about your current project CI & CD process?/Tell me about your CI & CD experience?
  + In my current project, I have 700+ test cases in my regression suite.
  + I set up Jenkins for CI and scheduled daily runs.
  + It started in the middle of the night like 3 am to run daily and send email with the build status.
  + It takes about 30 to 40 mins to run all the test cases.
  + I checked the build in the morning and
  + If failed >> then find out the reason and solve it
  + Then rerun the build.
* Tell me about your testing environment? And among those in which environment you run automation testing?
  + Most of the testing is done in a QA environment.
  + Also I do testing in a dev environment with the developer.
  + Prod ⇒ In prod, I only do manual testing, but I don’t do any automation testing in prod.
* How to connect Jenkins with your automation project?
  + 1st I build my automation project & check local build.
  + I push my project to GitHub when the build is success.
  + Then I connect my GitHub project inside the Jenkins.
* How to schedule/trigger Jenkins for CI pipeline or CI process or CI regression or CI automation?
  + I need to setup build trigger inside my Jenkins project.
  + I can trigger my build
    - Build Scheduler and here I need to set up the day and time to start Jenkins.
    - I can also do it by the GitHub hook trigger and Jenkins will kick off build with any one pushes new code in Github repo.
  + Did you setup the trigger options?
    - My DevOPS team helps me to setup daily.
  + How to setup the email inside the Jenkins?
    - I need to setup my office email server host & port.
    - Enable email notification.
    - Put my team members’ email addresses.

**How to set-up Jenkins:**

1. Download jenkins.exe from: <https://www.jenkins.io/download/> for Windows or Mac.
2. Install the Jenkins in the available port (default port is 8080).
3. Sign up from ⇒ <https://accounts.jenkins.io/signup>
4. Go to the “localhost:8080” to open Jenkins and login using credential.
   1. May need to enter admin password
5. To manage plugins
   1. Click on “Manage Jenkins” on the left side.
   2. Click on “Manage Plugins”.
   3. Click on “Available”.
   4. Add the following plugins:
      1. Maven integration plugin
      2. Git integration plugin
      3. Maven pipeline plugin (if pipeline is needed)
      4. Cucumber or Extent report plugin (if report is needed)
6. Go back to “Manage Jenkins” and click on “Global Tool Configuration”.
   1. Add JDK
      1. Name: JAVA\_HOME
      2. Unselect “Install Automatically”
      3. Add the Java path location inside JAVA\_HOME.
   2. Git
      1. Check “Install automatically”.
   3. Add Maven
      1. Name: MAVEN\_HOME
      2. Check “Install automatically”.
7. Click on “Dashboard” and click on “New item” to create a new project.
   1. Enter project name.
   2. Select “Maven Project” and click “OK”.
   3. Select “Github project” and paste the Git project URL.
   4. Under “Source code management”, select “Git” and paste the repo URL.
8. Build Triggers ⇒ to set-up build periodically
9. Build ⇒ add maven goal
   1. For maven clean install ⇒ clean install
10. Post Build ⇒ If you want the report, select it
    1. Cucumber report
11. Click on “Build now” to run the test.
    1. Click on “Check console”.
    2. After the test is completed, check the report.

**Table: Difference between Maven project & Pipeline project**

| **Usual project (maven)** | **Pipeline project** |
| --- | --- |
| Only show my build and test pass or failed | Stage 1: connect to GitHub |
|  | Stage2: compile code |
|  | Stage 3: Test run |
|  | Stage 4: report generation |
| Pom.xml | Groovy script |

**Code for MasterPageFactory for AutomationExercise.com:**

public class MasterPageFactory {

public MasterPageFactory(WebDriver driver) {

PageFactory.initElements(driver, this);

}

@FindBy(xpath="//\*[contains(text(), 'Login')]")

private WebElement loginbtn1;

@FindBy(xpath="(//\*[@name='email'])[1]")

private WebElement email1;

@FindBy(xpath="//\*[@name='password']")

private WebElement password;

@FindBy(xpath="(//\*[contains(text(), 'Login')])[4]")

private WebElement loginbtn2;

@FindBy(xpath="//\*[contains(text(), 'Logout')]")

private WebElement logoutbtn;

public WebElement getLoginbtn1() {

return loginbtn1;

}

public WebElement getEmail1() {

return email1;

}

public WebElement getPassword() {

return password;

}

public WebElement getLoginbtn2() {

return loginbtn2;

}

public WebElement getLogoutbtn() {

return logoutbtn;

}

**Pom.xml:**

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.7.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.3.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.7.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>5.0.9</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.10.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.11.0</version>

</dependency>

</dependencies>

<!-- Build plugin to force Maven JDK 1.8 compliance -->

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

**Framework:** Hybrid framework with BDD approach

* Can you tell me some framework name
  + Data driven
  + Keyword driven
  + Hybrid
  + Testng/Junit framework
  + BDD framework 🡺 cucumber tool
  + New
    - Robot framework
    - Reactor framework
    - Jasmine framework
* Difference

| Data driven | Keyword driven | Hybrid |
| --- | --- | --- |
| Focus is to handle multiple set of data | Focus is to run multiple combinations of function or methods based on Excel keyword | Combination of multiple framework |
| Automation method mostly same/fixed | Automation method different based on keyword | Can handle different set of data  Can handle to call different method based on keyword |
| Data will be different | Data mostly fixed here |  |
|  |  |  |
|  |  |  |

* Keyword driven
  + Excel 🡺 write keyword
    - Login🡺 run base login method
    - Select product🡺 run product method
    - Handle cart 🡺 run cart method
    - Payment🡺 run payment method
  + Test script
* What do you mean Hybrid?
  + Combination of multiple framework technology
  + Such as Data driven + keyword driven
* What are things need to know before starting any framework?
  + Business logic or need or requirement
  + Business flow
  + Application technology
    - Web based with HTML 🡺 Selenium
    - Client based 🡺 QTP/UFT , Test complete ,API
    - Non GUI Application 🡺 API
    - AngularJS 🡺 protector
  + Application functions
  + Manual testing process
  + Then I have discuss my team or manager

**API Testing:**

**API Manual Testing**

Swargger API Dcouments

Response Code

Response Time

Data Format

Type of response body

What do we need?

URL : Uniform Resource Locator

URI: Uniform Resource Identifier

API Methods

Requiresments/PBI

Set up Environment

Validation

Who creates or provides the API Doc?

Normally, my Dev create/provide the API doc but some time I can create with the help of dev.

Data Handling in API

SOAP: Simple Object Access Protocol

XML Data: <name>

JSON Data: {“name”}

**Example of put() & patch()**

Md Jaman ⇒ in the server

Name, DOB, Address, Phone, Email

Put() ⇒ Name, DOB, Address, Phone, Email

Patch() ⇒ Phone

URL & URI:

https://www.google.com/search?q=new+york&rlz=1C1CHBF\_enUS944US944&sxsrf=AJOqlzWDDba7Z5JuhcjJ4\_FObmvqUdAcmQ%3A1674002014782&ei=Xj7HY4GVL4CeptQPyOmFiAw&ved=0ahUKEwiBvMa778\_8AhUAj4kEHch0AcEQ4dUDCBA&uact=5&oq=new+york&gs\_lcp=Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMQgwEQQzIKCC4Q1AIQsQMQQzIECAAQQzIKCC4QsQMQgwEQQzIKCAAQsQMQgwEQQzIHCC4QsQMQQzIECAAQQzIICAAQsQMQgwEyCgguELEDEIMBEEMyCgguELEDEIMBEEM6BAgjECc6DgguEIAEELEDEIMBENQCOgsIABCABBCxAxCDAToRCC4QgAQQsQMQgwEQxwEQ0QM6CwguEIAEELEDEIMBOhAILhCxAxCDARDHARDRAxBDOg0ILhDHARDRAxDUAhBDOgQILhBDOgoILhDHARDRAxBDOggILhCABBCxAzoHCAAQsQMQQzoHCC4Q1AIQQzoKCAAQgAQQhwIQFDoRCC4QgwEQrwEQxwEQsQMQgAQ6BQgAEIAEOgUILhCABEoECEEYAEoECEYYAFAAWPIQYOkSaAFwAXgAgAHiAYgBzwiSAQUzLjUuMZgBAKABAcABAQ&sclient=gws-wiz-serp

URL ⇒ [https://www.google.com](https://www.google.com/) ⇒ URL

URI ⇒ /search

Parameter/Query string ⇒ ?q=new+york

Authentication code/token ⇒ &rlz=1C1CHBF\_enUS944US944&sxsrf=AJOqlzWDDba7Z5JuhcjJ4\_FObmvqUdAcmQ%3A1674002014782&ei=Xj7HY4GVL4CeptQPyOmFiAw&ved=0ahUKEwiBvMa778\_8AhUAj4kEHch0AcEQ4dUDCBA&uact=5&oq=new+york&gs\_lcp=Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMQgwEQQzIKCC4Q1AIQsQMQQzIECAAQQzIKCC4QsQMQgwEQQzIKCAAQsQMQgwEQQzIHCC4QsQMQQzIECAAQQzIICAAQsQMQgwEyCgguELEDEIMBEEMyCgguELEDEIMBEEM6BAgjECc6DgguEIAEELEDEIMBENQCOgsIABCABBCxAxCDAToRCC4QgAQQsQMQgwEQxwEQ0QM6CwguEIAEELEDEIMBOhAILhCxAxCDARDHARDRAxBDOg0ILhDHARDRAxDUAhBDOgQILhBDOgoILhDHARDRAxBDOggILhCABBCxAzoHCAAQsQMQQzoHCC4Q1AIQQzoKCAAQgAQQhwIQFDoRCC4QgwEQrwEQxwEQsQMQgAQ6BQgAEIAEOgUILhCABEoECEEYAEoECEYYAFAAWPIQYOkSaAFwAXgAgAHiAYgBzwiSAQUzLjUuMZgBAKABAcABAQ&sclient=gws-wiz-serp

Read all the Q&A from API Q&A.

* Tell me about the **HTTP status code**:
  + 1xx ⇒ informational
  + 2xx ⇒ Success/OK (Test passed)
  + 3xx ⇒ Redirection
  + 4xx ⇒ Client error/Application error (Application down) ⇒ Test failed
  + 5xx ⇒ Server error (server down) ⇒ Test failed

**How to set up Postman on your computer?**

1. Go to your browser and search for “Download Postman” or go the download link: <https://www.postman.com/downloads/>
   1. If you’re using a Windows OS, download 64-bit.
2. After installing is done, open the Postman.
3. Create an account.
4. After you get to the Postman welcome page, Click on Workspaces dropdown menu and then click on My Workspace.
5. Then click on: Collections >> Create Collection.
6. Right click on the Collection name >> Add Request.
7. Select request type (i.e. GET, POST, PUT, DELETE).

**Manual testing with Postman ⇒**

1. Application
   1. Non-GUI application ⇒ Link/URL
   2. Server
      1. Link/URL
      2. Host & Port
2. Protocol ⇒ HTTP
3. HTTP Protocol methods:
   1. GET
   2. POST
   3. PUT
   4. PATCH
   5. DELETE

Dummy URL for API testing: <https://dummy.restapiexample.com/>

Dummy URL for authentication testing: <https://restful-booker.herokuapp.com/auth>

Then in the message body:

{

"username" : "admin",

"password" : "password123"

}

**GET request testing:**

1. Select the request type as GET.
2. Paste the URL in the Postman.
3. Click on the “Send” button.
4. Click on “Tests”.
5. Write test cases using the snippets.
   1. Status code
      1. pm.**test**("Status code is 200", **function** () {

pm.response.to.have.status(200);

});

* 1. Response time
     1. pm.**test**("Response time is less than 2000ms", **function** () {

pm.expect(pm.response.responseTime).to.be.below(2000);

});

* 1. Response body: Contains string
     1. pm.**test**("Body contains success", **function** () {

pm.expect(pm.response.text()).to.include("success");

});

* 1. Data format ⇒ Content-Type header check
     1. pm.**test**("Content-Type is present", **function** () {

pm.response.to.have.header("Content-Type");

});

* 1. Response data check
     1. Null or not ⇒ Contains string then change it to .not.null
        1. pm.**test**("Response body is not null", **function** () {

pm.expect(pm.response.text()).not.null;

});

* + 1. Check attribute value or string value ⇒ JSON value check
       1. pm.expect(jsonData.data[0].employee\_name).to.eql("Tiger Nixon");

**POST request testing:**

1. Select the request type as POST.
2. Paste the URL in the Postman.
3. Select Body.
   1. Select raw data.
   2. Select JSON.
4. Pass the date in JSON format.
5. Click on the “Send” button.
6. Click on “Tests”.
7. Write test cases using the snippets.
   1. Status code
   2. Response time
   3. Response body: Contains string
   4. Data format ⇒ Content-Type header check
   5. Response data check
      1. Null or not ⇒ Contains string then change it to .not.null
      2. Check attribute value or string value ⇒ JSON value check

**PUT request testing:**

1. Select the request type as PUT.
2. Paste the URL in the Postman.
3. Select Body.
   1. Select raw data.
   2. Select JSON from the dropdown.
4. Pass the date in JSON format.
5. Click on the “Send” button.
6. Click on “Tests”.
7. Write test cases using the snippets.
   1. Status code
   2. Response time
   3. Response body: Contains string
   4. Data format ⇒ Content-Type header check
   5. Response data check
      1. Null or not ⇒ Contains string then change it to .not.null
      2. Check attribute value or string value ⇒ JSON value check

**DELETE request testing:**

1. Select the request type as DELETE.
2. Paste the URL in the Postman.
3. Click on the “Send” button.
4. Click on “Tests”.
5. Write test cases using the snippets.
   1. Status code
   2. Response time
   3. Response body: Contains string
   4. Data format ⇒ Content-Type header check
   5. Response data check

**API Testing summary:**

1. Manual ⇒ Postman
2. Automation ⇒ Java + RestAssured
3. Before manual or automation API Testing
   1. API Document ⇒
      1. Jira Confluence
      2. Swagger Document
   2. Who created this document? ⇒ I created with the help of my dev OR my dev created
   3. API document Contains ⇒
      1. Environment setup
      2. Requirement/ acceptance criteria
      3. API methods to call/test
      4. API Testing flow or process
      5. API rule or proto (written in xml file)
4. Manual testing (using Postman) ⇒
   1. 1st setup as per the API document.
   2. Create a project (Collection).
   3. Call/Add API requests (GET/POST/PUT/PATCH/DELETE)
   4. Write test cases to validate
      1. Response status code 200
      2. Response time
      3. Status code has string ⇒ contains “ok” or “success”
      4. Data format ⇒ application/json
      5. Response body contains string
         1. Not null
         2. Contains any specific value
      6. JSON value check
         1. 1st employee name = Tiger Nixon
   5. How to pass parameters or query strings?
      1. Select Params >> pass the key & the value
   6. How to do authentication of API testing?
      1. 1st can send my testing user & pass to the non-GUI application or server by using POST method.
      2. Then the server will generate a key/token.
      3. Then my URL + q= & API key
      4. Another way ⇒
         1. Kerberos authentication/dll ⇒ runtime creation of authenticator key based on user
         2. CODE ⇒ Given().auth.preemptive.basic(“user”, “password);

**API Testing with server:**

* Check if you’ve node.js installed in your computer
  + Open the CMD/Terminal
  + Run: node -v
* Download: node js from a browser
* Open: CMD/terminal
* Install server **Windows**: npm install -g json-server
* For **Mac**: sudo npm install -g
* **Windows** start server: json-server --watch db.json
* **Mac** start server: npx json-server --watch db.json OR json-server db.json

**API Testing Automation - Java with Rest-Assured:**

**Setting up API Testing automation framework:**

1. Add the following dependency files in the pom.xml
   1. Java version 8
   2. Rest-assured
   3. Apache groovy
   4. TestNG (for assertion)
   5. Cucumber core
   6. Cucumber TestNG
   7. Maven Surefire Plugin

**Testing GET/DELETE method:**

1. Create a Response object
   1. Pass the URL
2. Write validation test cases

CODE ⇒

1. **How to do the GET method testing in automation?**

CODE⇒

public class TestingGetWithTestNG {

Response response;

@BeforeTest

public void getSetup() {

response = RestAssured.get("http://dummy.restapiexample.com/api/v1/employees");

}

@Test

public void getStatusCode() {

System.out.println("Status code ="+response.getStatusCode());

Assert.assertEquals(response.getStatusCode(), 200);

}

@Test

public void getContainType() {

System.out.println("Data format ="+response.getContentType());//json

Assert.assertEquals(response.getContentType(), "application/json");

Assert.assertTrue(response.getContentType().contains("json"));

}

@Test

public void getResponseTime() {

System.out.println("Response time="+response.getTime());//time

Assert.assertTrue(response.getTime()<2000);

}

@Test

public void getResponseNullOrNot() {

System.out.println("Is response null or not="+response.toString()!=null);

}

@AfterTest

public void tearDown() {

}

}

**Testing POST method:**

1. To add test data, create an object of JsonObject class and addProperty()
2. RequestSpecification and add data inside the body in Json format.
3. Pass the URL.
4. Validate the test cases.

CODE ⇒

//1st ==> add test data = JsonObject class and addProperty()

JsonObject json = new JsonObject();

//json.addProperty(“Key”, “Value”);

json.addProperty("Name", "Rezwan");

json.addProperty("Profession", "QA Automation Engineer");

//2nd ==> RequestSpecification and add data inside the body

RequestSpecification rsf = RestAssured.given();

rsf.header("Content-Type", "application/json");

rsf.body(json.toString()); //passing test data inside the message body in Json format

//3rd ==> Pass the URL

Response response = rsf.post("https://dummy.restapiexample.com/api/v1/create"); //pass the link here

//4th ==> validate

System.out.println("Status code ="+response.getStatusCode()); //status code should be 200

Assert.assertEquals(response.getStatusCode(), 200);

System.out.println("response time in ms ="+response.getTime()); //should display response time in ms

Assert.assertTrue(response.getTime()<2000);

System.out.println("Data format ="+response.getContentType());

//Assert.assertEquals(response.getContentType(), "application/json; charset=utf-8");

Assert.assertTrue(response.getContentType().contains("json"));

System.out.println("Response body should not be null ="+response.toString()!=null);

//specific JSON attribute value or how to parse Json data

JsonPath jp = response.jsonPath();

Assert.assertEquals(jp.get("Name"), "Rezwan");

Assert.assertEquals(jp.get("Profession"), "QA Automation Engineer");

**Some important interview questions in API:**

1. **List a few authentication techniques used in API testing:**
2. Session / Cookies based Authentication
3. Basic Authentication (I do this authentication)
4. Digest Authentication
5. OAuth
6. Above all are the few most important authentication techniques.

**2. What are the main challenges you face in the API testing in your project?**

1. API documentation
2. Access to the database
3. Call sequencing

**3. What is the difference between authorization and authentication?**

⇒ Authentication (e.g. Login) is a process of presenting your credentials to the system and then the system validates your credentials. These credentials tell the system about who you are.

Authorization is a process of allowing or denying someone from accessing something once authentication is completed.

**POM File for API Testing:**

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

<!-- https://mvnrepository.com/artifact/commons-lang/commons-lang -->

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>4.1.0</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.1.0</version>

</dependency>

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20210307</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.10.4</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-core -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-core</artifactId>

<version>6.10.4</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>6.10.4</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-picocontainer -->

<!-- <dependency> <groupId>io.cucumber</groupId> <artifactId>cucumber-picocontainer</artifactId>

<version>6.10.4</version> <scope>test</scope> </dependency> -->

<!-- https://mvnrepository.com/artifact/net.masterthought/maven-cucumber-reporting -->

<dependency>

<groupId>net.masterthought</groupId>

<artifactId>maven-cucumber-reporting</artifactId>

<version>5.3.0</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>5.0.8</version>

</dependency>

<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>rest-assured</artifactId>

<version>4.2.0</version>

</dependency>

<dependency>

<groupId>org.codehaus.groovy</groupId>

<artifactId>groovy</artifactId>

<version>3.0.0-alpha-4</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.5</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M4</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>TestSuite.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

<plugin>

<groupId>net.masterthought</groupId>

<artifactId>maven-cucumber-reporting</artifactId>

<version>5.3.0</version>

<executions>

<execution>

<id>execution</id>

<phase>verify</phase>

<goals>

<goal>generate</goal>

</goals>

<configuration>

<projectName>ExecutionResult</projectName>

<skip>false</skip>

<outputDirectory>${project.build.directory}/cucumber-report-html</outputDirectory>

<inputDirectory>target</inputDirectory>

<jsonFiles>

<param>\*\*/\*.json</param>

</jsonFiles>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

**Interview Preparations Guidelines:**

**Coding to practice for the interview:**

Practice the following coding from Java Q&A ⇒

18, 32, 33, 34, 35, 37, 39/43, 42, 142, 143, 148, 151

**Job Portals you need to apply:**

1. LinkedIn
2. Indeed
3. Monster
4. GlassDoor
5. Dice
6. CareerBuilder
7. ZipRecruiter
8. Any other job portals

**Things you need to do ASAP:**

1. Set up a professional voicemail on your phone.
   1. Take the call when you can talk freely.
   2. Set up the msg like: “Hi! This is First Name. Thanks for calling. Sorry, I’m unable to take the call this time. Please leave a brief message with your name & number. I’ll get back to you as soon as I can.”
2. Set up a professional signature on your Gmail:

**Thanks!**

**Sincerely yours,**

**First Name**

**Cell: (888)-888-8888**

[**youremail@gmail.com**](mailto:youremail@gmail.com)

1. Open your account in the job portals.

**Notes from Sobhan Bhai:**

**Easy Way Learn Java**

**Java**: It is an object and function oriented programming language. We can use Java in any Operating System and any Browser.

**Core Java**: OOPs concept is core Java. As a Tester we should know core java.

**What is class?**

Class is the blueprint of an object, a collection of objects. It is imaginary but logical.

Like: Animal(Class) Under Animal you can imagine a lot of objects such a dog, cat, tiger etc.

Human: (class) Under human you can see Man: Sobhan, Karim

**What is Object:**

An object is a real world substance, it could be anything that has a physical and logical identity.

Like: Table, Book, Sobhan, coin etc.

How do you identify an object?

By things we identify an object

1. State/attribute
2. Behavior

State/attribute is represent - Data type which we use as a variable

Like: Sobhan, Age-45, SSN:345787888

Any character and number together is called String

Sobhan= data type is String

Sobhan123 = **String** type of Data

Age is 45 = **int**

Behavior represents the functionality or activities of the object. ( Sobhan can run, can work, can sleep, Method, inside the method you write code and run)

IDE: Integrated Development Environment

**Method in Java**

In Java method means, where we write the code and variables

**Methods has three parts**

1. Method name/Signature
2. Method Parenthesis ()
3. Method Body { }

Method name always start with small letter

Example:

Public static void myDog() {

}

Here **myDog** is a Method

Before () whatever name that is known as method name or method Signature

Main method we use to run to code

**How to call the methods in the Main methods to run?**

Ans: We call the methods in the main method, depending on the Static word in the methods, means: if there is any Static word in the method, we need to call ClassName.Methodname.

Public static void myDog(){

}

Classname(Animal).myDog

**Public void myDog(){**

**}**

If there is not static word in the method then we need to create an object.

**How to create an object?**

**Classname obj = new classname();**

**Static Meaning in Java**

1. Static means class level
2. Static method no need create an object
3. Static method can handle only static variable
4. Static method can not be inheritance
5. Static method can not be overriding

**Variable**: It is a container who hold the data

How many types of Variables

Three Types

**Local Variable**

It is declared inside the Method: (int a-10;)

**Class Level or Static Variable**

It has be declare outside of the method, inside the class but static word must be apply

Instance Variable

**Class in Java**

Concrete Class/Regular/Non Abstract

Abstract Class

Abstract class must have Abstract Keyword before the class Keyword

Abstract class can handle regular and abstract method.

**Constructor in Java**

Constructor is a special type of method in Java. Constructor name should be the same as Class name.

Should not be a return type or void method. It does not return any value.

Constructor created automatically when we create an object in that class.

Type of Constructor

Default constructor (Non - parameterized)

Parameterized Constructor

**Exception in Java**

Exception means something is wrong, not regular, error, is not working properly.

**Two type of exception in Java**

**Checked Exceptions:**

Before running the code(compile time) you need to handle the exception.

ClassNotFound Exception

FileNotFound Exception

IOException

**UnChecked Exception**

After running the code you need to handle it.

Runtime Exception

NullPointer Exception

IllegalStateException

ArrayIndexOutofBound exception

NumberFormat Exception

Arithmetic Exception

**Java Looping**

For Loop

for(initialization, Condition, increment ){

It keeps on looping unit the condition is satisfied

}

Array: Arrays are objects which store multiple variable of the similar data types

Each variable is know as Array Elements

Array length is fixed

Array length is 10

Array follow index theory

First index is 0

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Array length \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**JAVA COLLECTIONS**

Collection: Framework/API (Interface and classes)

The Collection framework is a collection of **Interfaces and classes** which help us storing and processing data efficiently.

**The important Interfaces in this API are:**

List

Set

Map

**List:** A list is an order collection. List can be small or big, It can be duplicate and Null

Below classes are in the List Interface

ArrayList

LinkedList

Vector

**Set:** A Set is a collection that can not contain any duplicate

Below classes are in the Set Interface

HashSet

LinkedHashSet

TreeSet

**SELENIUM**

**What is Selenium?**

Selenium is an Automation suite or API (Application Programming Interface) We used for automated web-based application only. It is support all the programming language except C++. It support all the operating system and all the browsers.

**Its an API,**

**Only Automated web-based Appliction**

**Support all Languages except C++**

**Support all OS**

**Support all Browsers**

What are the different Components of Selenium?

Selenium IDE: It follow the record and play back principle and it is work with FireFox

Selenium RC: Remote Control

**Selenium Web Driver**: Used for automated web based applications through all the browser and methods

**Selenium Grid**: Selenium Grid is a concept of distribution of test cases in different browsers different OS, and different languages.

How does WebDriver (Selenium) know your username, PW, and any other web elements?

Ans: By the Selenium Locator

How to find Sign in Button in HTML Code?

Looking for Sign In

Before = sign whatever is there that is known as a Locator/Attribute

And After = sign what ever is there that is known as a Value of the locator

ID = sign in

Id is locator/attribute and Sign in is a value

What is Xpath regular Formula

//\*[@attribute =’value’]

//\*[@id=’value’]

**PageFactory**

What is Pagefactory/Page Object Model?

It is a repository(storage) where we store all the xpath. It is easy to handle if any xpath is changed in the future.

Formula: @FindBy(xpath=”//\*[@name=’passwd']”)

Code of PageFactory

We need to use Jave Encapsulation

@FindBy(xpath=”//\*[@name=’passwd']”)

private WebElement Sign In

private WebElement User Name

private WebElement PW

We use here only getter methods

Once all done we need to use constructor

PageFactory.initElements(driver, this)

**TestNG**

It is a java unit testing framework

JUnit: it is also a unit testing framework

**How to download TestNG?**

It’s Official website is : Testng.org

In Eclipse go to help>>>>.Select Eclipse MarketPlace>>>>> In the Scrch

Type TestNG and Hit Enter >>>> you can see TestNG for Eclipse

Then click Install Then Next and couple of Next and Accept the License

And ReStart your Eclispe and done

Basic Annotation

This is the order TestNG follows:

@Before Suite

@Before Test

@Before Class

@Before Method

@Test

@After Method

@After Class

@After Test

@After Suite

@Before Test ---- This is like before Setup

@Test ----- This is called main, like Java main method - This is called aTest Case

@After Test --- Close or quit

**Why we need TestNg**

We need TestNg to generate report

Easy to understand TestNG Annotation

**What are the benefits of using TestNG?**

We can generate reports through TestNg.

In TestNG annotations are easy to understand

We can generate three type of report

We can control the test execution

We can do parallel and cross browser testing

Fail test cases can be executed from Testngfail.xml

We can run tests from Testng.xml, We can run test Suite.

We can do grouping, priorities test cases, enabled, desabled, depending on Methods, Include and exclude the test cases.

**Cucumber**

What is Cucumber?

Cucumber is Software Testing tools. It supports Behavior Driven Development(BDD). It was originally written by Ruby Programming Languare.

**Cucumber Annotation:**

@Given - Pre-Action (Setup or opening info)

@When - Start all action (Type UID, PW, Click Login Button)

@Then - Validation: After action, means what is the result or outcome

Note: If you need any additional step we can use **And**

How feature file look like?

**Feature:** What to Test

User able to login with the valid username and Password (Login Functilnality)

**Scenario**: How to test or what is result ( User able to login with valid username, Password and verify the Homepage.

Given : User able to open any browser

And : User able to enter the URL

When: User able to enter the username

And: User able to enter the Password

And: User Click to Submit Button

Then: User able to login Succefully

Then: User able to verify is name on the homepage

**In Cucumber Automation you need at least Three file**

1. Feature file : Test Step (Like manual test case)
2. Step Definition File: (Java) we write the code here
3. Test Runner File: (Java) for running the code with different options.