(1) Hashtable vs Hashmap

1. Synchronization or Thread Safe :  This is the most important difference between two . HashMap is non synchronized and not thread safe.On the other hand, HashTable is thread safe and synchronized.

When to use HashMap ?  answer is if your application do not require any multi-threading task, in other words hashmap is better for non-threading applications. HashTable should be used in multithreading applications.

2.Null keys and null values :  Hashmap allows one null key and any number of null values, while Hashtable do not allow null keys and null values in the HashTable object.

3.Iterating the values:  Hashmap object values are iterated by using iterator .HashTable is the only class other than vector which uses enumerator to iterate the values of HashTable object.

4. Fail-fast iterator  : The iterator in Hashmap is fail-fast iterator while the enumerator for Hashtable is not.

According to

HYPERLINK "http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html" \t "\_blank"

Oracle Docs

,  if the Hashtable is structurally modified at any time after the iterator is created in any way except the iterator's own remove method , then the iterator will throw ConcurrentModification Exception.

Structural modification means adding or removing elements from the Collection object (here hashmap or hashtable) . Thus the enumerations returned by the Hashtable keys and elements methods are not fail fast.We have already explained the

HYPERLINK "http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html" \t "\_blank"

difference between iterator and enumeration

.

5. Performance :  Hashmap is much faster and uses less memory than Hashtable as former is unsynchronized . Unsynchronized objects are often much better in performance in compare to synchronized  object like Hashtable in single threaded environment

6. Superclass and Legacy :  Hashtable is a subclass of Dictionary class which is now obsolete in Jdk 1.7 ,so ,it is not used anymore. It is better off externally synchronizing a HashMap or using a ConcurrentMap implementation (e.g ConcurrentHashMap).HashMap is the subclass of the AbstractMap class. Although Hashtable and HashMap has different superclasses but they both are implementations of the "Map"  abstract data type.

(2) Iterator vs Enumeration

The most common interview question in Collections is What is the difference between iterator and enumeration.

Iterator

Iterator is the interface and found in the java.util package.

It has three methods

\*hasNext()

\*next()

\*remove()

Enumeration

Enumeration is also an interface and found in the java.util package .

An enumeration is an object that generates elements one at a time. It is used for passing through a collection, usually of unknown size.

The traversing of elements can only be done once per creation.

It has following methods

\*hasMoreElements()

\*nextElement()

An iterator over a collection. Iterator takes the place of Enumeration in the Java collections framework.

Iterators differ from enumerations in two ways:

Iterators allow the caller to remove elements from the underlying collection during the iteration with well-defined semantics.

Method names have been improved.

(3) Access level modifiers determine whether other classes can use a particular field or invoke a particular method. There are two levels of access control:

At the top level—public, or package-private (no explicit modifier).

At the member level—public, private, protected, or package-private (no explicit modifier).

A class may be declared with the modifier public, in which case that class is visible to all classes everywhere. If a class has no modifier (the default, also known as package-private), it is visible only within its own package (packages are named groups of related classes — you will learn about them in a later lesson.)

At the member level, you can also use the public modifier or no modifier (package-private) just as with top-level classes, and with the same meaning. For members, there are two additional access modifiers: private and protected. The private modifier specifies that the member can only be accessed in its own class. The protected modifier specifies that the member can only be accessed within its own package (as with package-private) and, in addition, by a subclass of its class in another package.

HYPERLINK "https://jenkins.io/"

Jenkins

//EXAMPLE 1 - This is the worst kind of wait and should almost never be used

Thread.Sleep(10000)

//EXAMPLE 2 - This is a better wait that dynamically checks for the presence of an element for a maximum amount of time, a bit burdensome

IWebDriver driver = new FirefoxDriver();

driver.Url = "http://somedomain/url\_that\_delays\_loading";

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>

{

return d.FindElement(By.Id("someDynamicElement"));

});

//EXAMPLE 3 - This is the most convenient method provided to us by the Webdriver API

var wait = new WebDriverWait(\_driver,TimeSpan.FromSeconds(10));

wait.Until(ExpectedConditions.ElementIsVisible(By.Id("elementId")));

JUnit Cucumber Tutorial 11 - Drop Downs, Checkboxes and Radios Buttons

Esignlive

API Test:

URL vs URI

URL: https://min-api.cryptocompare.com/data/price/

http:// :80

https:// :443

Domain: min-apin.cryptocompare.com

Path : /data/price

Parameter : fsym=BTC

JSON (learn)

HYPERLINK "http://www.rest-assured.io"

http://www.rest-assured.io

hamcrest

homework# google map API test

REST Assured

SOAP: Simple Object Access Protocol, it relies heavily on XML and schema

REST: representational State Transfer. The result can be XML, JSON, YAML

REST Examples

(1) given().

get(“http://....).

then().

body(“RestResponse.result.name”, is(“Italy”));

or

then().

Root(“RestResponse.result).

Body(“name”, is(“Italy”));

(2) detachroot:

(3) org.hamcrest.Matcher library

XML

body(hasXPath(“/customer/firstname”), containsString(“Sue”));

(4) testPostReqire(){

given().

header(“AppKey”, “key-value”).

param(“wfsfirst\_name”, “first”).

param(“wfslast\_name”, “last”).

param(“wfsemail”,

HYPERLINK "mailto:test@test.com"

test@test.com

).

when().

post(

HYPERLINK "http://Api.font.com/rest/json/Accounts/"

http://Api.font.com/rest/json/Accounts/

.

then().

statusCode(401).log().all();

}

(5) String responseAsString = get(“http:....).asString();

(6) InputStream stream = get(“http://....).asInputStream();

(7) byte[] byteArray = get(“http://.....).asByteArray();

(8) Extract details using path

String href =

when().

get(“http://...).

then().

extract().

path(“url”);

when().get(href).then().statusCode(200);

(9) Extract details using path in one line

String href = get(

HYPERLINK "http://....).path(\“thumbnailurl\”)"

http://....).path(“thumbnailurl”)

;

Or

String href = get(

HYPERLINK "http://...).andReturn().jsonPath().getString(\“thumbnailurl\”)"

http://...).andReturn().jsonPath().getString(“thumbnailurl”)

;

(10) Extract details as response for further use

Resonse response = when().

get(“http://....”).

then().

extract().

response();

System.out.println(“ContentType “ + response(“ContentType”);

System.out.pintln(“href “ = response.path(“url”));

System.out.println(“status code:”+response.statusCode());

(11) Groovy feature advantages:

a. /\* verify response type\*/

public void testContentType(){

given().

get(

HYPERLINK "http://services.groupkt.com/country/get/isocode/cn"

http://services.groupkt.com/country/get/isocode/cn

).

then().

statusCode().

contentType(ContentType.JSON);

or contentType(ContentType.XTML);

or contentType(ContentType.XML);

}

b. /\* this test will verify the response schema with predefined existing schema path src/test/resources/geo-schema.json

@Test

Public void testSchema(){

given().

get(“http://geo.groupkt.com/ip/172.217.4.14/json).

then().

assertThat().body(matches.JsonSchemaInClasspath(“test3.geo.schema123.json”));

}

c. /\* verify is some expected name present in response or not

@Test

Public void testPresentOfElements(){

given().

get(“http://services.groupkt.com/country/search?text=lands”).

then().

.body(“RestResponse.result.name”, hasItems(“Cayman Islands”, “Cook Islands”)).log().all();

}

d. /\* RestAssured implemented in Groovy and hence Groovy advantages can be taken

here we are adding length of all “alpha3\_code” code coming in response

@Test

Public vlid testLengthOfResponse(){

when().

get(“http://services.groupkt.com/country/search?text=islands”).

then().

.body(“RestResponse.result.apha3\_code\*.length().sum()”, greaterThan(10));

}

e. /\* to get all attribute as List\*/

@Test

public void restGetResponseAsList(){

String response = get(

HYPERLINK "http://services.groupkt.com/country/search?test=lands\”).asString()"

http://services.groupkt.com/country/search?test=lands”).asString()

;

List<String> ls=from(response).getList(“RestResponse.result.name);

System.out.println(“ListSize: “ +ls.size());

for(String country: ls){

if(country.equals(“Solomon Islands”))

System.out.println(“found my place);

}

f. /\* to get response as list and apply some conditions on it

the groovy has an implicit variable called ‘it’ which represents the current item in the list

@Test

public void testConditionOnList(){

String response = get(

HYPERLINK "http://services.groupkt.com/country/search?text=lands).asString()"

http://services.groupkt.com/country/search?text=lands).asString()

;

List<String> ls=from(response).getList(“RestResponse.result.findAll(it.name.length() > 40).name”);

System.out.println(ls);

}

(12) Headers and Cookies

a. /\* extract details as String and fetching further details w/o using json path

@Test

public void testJsonPath1(){

String repsonseStr =

when().

get(“

HYPERLINK "http://jsonplaceholder.typicode.com/photos"

http://jsonplaceholder.typicode.com/photos

).

then().

extract().asString();

List<Integer> albumIds = from(responseStr).get(“id”);

System.out.println(albumIds.size());

}

b. /\* extract details as String and fetching further details using JSONPath

@Test

public void testJsonPath2(){

String json=

when().

get(“http:/ services.groupkt.com/country/get/all”).

then().

extract().asString();

JsonPath jsonPath = new JsonPath(json).setRoot(“RestResponse.result”);

List<String> list=jsonPath.get(“name”);

System.out.println(list.size());

}

c. /\*get response headers\*/

@Test

public testResponseHeaders(){

Response response = get(“http://jsonplaceholder.typicode.com/photos);

String headerCFRAY = response.getHeader(“CF-RAY”);

// get all headers

Headers headers = response.getHeaders();

for(Header h: headers){

System.out.println(h.getName()+”:”+h.getValue());

}

}

d. /\* to get cookies \*/

public testResponseHeaders(){}

Response response = get(“http://jsonplaceholder.typicode.com/photos);

Map<String, String> cookies = response.getCookies();

for(Map.entry<String, String>entry: cookies.entrySet()){

System.out.println(entry.getKey()+”:”+entry.getValue());

}

}

e. /\* to get detailed cookies\*/

public void testDetailedCookies(){

Response response = get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos"

http://jsonplaceholder.typicode.com/photos

);

Cookie a = response.getDetailedCookie(”\_\_cfduid “);

System.out.println(“Detailed:” +a.hasExpiryDate());

System.out.println(“Detailed:” +a.getExpiryDate());

System.out.println(“Detailed:” +a.hasValue());

}

(13) Setting Request Data

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*Generally CONNECT used with HTTPS request

// http methods (GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE)

// URL encoding please refer to https://www.tutorialspoint.com/http/http\_url\_encoding.htm

@Test

Public void testConnectRequest(){

when().

request(“CONNECT”, “

HYPERLINK "https://spi.fonts.com/rest/json/Accounts/"

https://spi.fonts.com/rest/json/Accounts/

).

statusCode(400)

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* in GET request we can set the query parameter

@Test

Public void TestQueryParameters(){

given().

queryParam(“A”, “A val”).

queryParm(“B”, “B Val”).

when().

Get(“

HYPERLINK "https://spi.fonts.com/rest/json/Accounts/"

https://spi.fonts.com/rest/json/Accounts/

).

then().

statusCode(400);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* In POST request we can set form parameter

@Test

Public void TestFormParameters(){

given().

formParam(“A”, “A val”).

formParm(“B”, “B Val”).

when().

post(“http://spi.fonts.com/rest/json/Domains/“).

then().

statusCode(400);

}

/\* to set parameters recommended way

If request is GET then param will be treated as queryParam

If request is POST the param will be treated as formParam

Param(“A”, “A value”)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* te set multiple value parameters we can pass list, multiple values or no values in param

@Test

Public void testSetValueParameters(){

List<String> list = new ArrayList<String>();

list.add(“one”);

list.add(“two”);

given().

param(“A”, “Val1”, “Val2”,”val3”).

param(“B”).

param(“C”, list).

when().

get(“https://api.fonts.com/rest/json/Accounts/“).

then().

statusCode(400);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* to path parameters

Public void testSetPathParameters(){

given().

pathParam(“type”, “json”).

pathParam(“section”, “Domains”).

when().

post(“https://api.fonts.com/rest/{type}/{section}/“).

then().

statusCode(400);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* cookies can be set in request param

@Test\

Public void testSetCookiesInRequest(){

given().

cookie (“\_\_utmt”, “1”).

when().

get(

HYPERLINK "http://www.webservicex.com/globalweather.asmx?op=GeocitiesByCountry"

http://www.webservicex.com/globalweather.asmx?op=GeocitiesByCountry

).

then().

statusCode(200);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* multiple cookies can be set in request param

Todo: test example not runable code

@Test

public void testSetMultiCookiesInRequest(){

//to set multiple value

given()cookie(“key”, “val1”, “val2”);

//to set detailed cookies

Cookie cookie = new Cookie.Builder(“some\_cookie”, “some\_value”).setSecured(true).setComment(“some comment”).build();

given().cookie(cookie).when().get(“/cookie”).then().assertThat().body(equalTo(“x”));

//set multiple detailed cookies\

Cookie cookie1 = new Cookie.Builder(“some\_cookie”, “some\_value”).setSecured(true).setComment(“some comment”).build();

Cookie cookie2 = new Cookie.Builder(“some\_cookie”, “some\_value”).setSecured(true).setComment(“some comment”).build();

Cookies cookies = new Cookies(Cookie1, cookies2);

given().cookies(cookies).when().get(“/cookie”).then().assertThat().body(equalTo(“x”));

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* we can pass single header, headers with multiple values and multiple headers

Public void testSetHeaders(){

given().

header(“k”, “v”).

header(“k10”, “val1”, “val2”, “val3”).

headers(“k1”, “v1”, “v2”, “v3”).

when().

get(

HYPERLINK "https://api.fonts.com/rest/json/Accounts/"

https://api.fonts.com/rest/json/Accounts/

).

then().

statusCode(400);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* Content type can also be set

public void testSetContentType (){

given().

contentType(“ContentType.JSON).

cententType(“applicaton/json; charset=utf-8”).

when().

get(

HYPERLINK "https://api.fonts.com/rest/json/Accounts/"

https://api.fonts.com/rest/json/Accounts/

).

then().

statusCode(400);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* Response Verification

/\* status code verification

@Test

public void testStatusInResponse (){

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/).then().assertThat().statusCode(200).log().all()"

http://jsonplaceholder.typicode.com/photos/).then().assertThat().statusCode(200).log().all()

;

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/).then().assertThat().statusLine(\“HTTP/1.1"

http://jsonplaceholder.typicode.com/photos/).then().assertThat().statusLine(“HTTP/1.1

200 OK”).log().all();

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/).then().assertThat().statusLine("

http://jsonplaceholder.typicode.com/photos/).then().assertThat().statusLine(

containsString(“OK”));;

}

/\* headers verification

@Test

public void testHeaderInResponse (){

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/).then().assertThat().header(\“X-Powered-By"

http://jsonplaceholder.typicode.com/photos/).then().assertThat().header(“X-Powered-By

, “Express”);

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/).then().assertThat().headers(\“Vary"

http://jsonplaceholder.typicode.com/photos/).then().assertThat().headers(“Vary

, “Accept-Encoding”, “Content-Type”, containsString(“json”));

}

/\* content type verification

@Test

public void testContentTypeInResponse (){

given().get(http://jsonplaceholder.typicode.com/photos/).then().assertThat().contentType(ContentType.JSON);

}

/\* body text verification

@Test

public void testBodyInResponse (){

String responseString = get(

HYPERLINK "http://www.thomos-bayer.com/sqlrest/CUSTOMERS/02/).asString()"

http://www.thomos-bayer.com/sqlrest/CUSTOMERS/02/).asString()

;

given().get().then().assertThat().body(equalTo(responseString));

}

/\* body attribute verifications using java & lambda expression

@Test

public void testBodyParametersInResponse (){

//Java 7

given().

get(“http://jsonplaceholder.typicode.com/photos/1”).

then().

body(“thumbnailUrl”, new ResponseAwareMatcher<Response>(){

public Matcher<?> matcher(Response response){

return equalTo(

HYPERLINK "http://placehold.it/150/92c952"

http://placehold.it/150/92c952

);

}

});

// with java 8 lambda expression

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/phots/1\”).then().body(\“thumbnailUrl"

http://jsonplaceholder.typicode.com/phots/1”).then().body(“thumbnailUrl

, response-> equalTo(“http://placehold.it/150/92c952”));

given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/phots/1\”).then().body(\“thumbnailUrl"

http://jsonplaceholder.typicode.com/phots/1”).then().body(“thumbnailUrl

, endsWith(“/92c952”));

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* Performance Testing using Rest Assured

@Test

public void testResponseTime(){

long t = given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/).time()"

http://jsonplaceholder.typicode.com/photos/).time()

;

System.out.println(“Time(ms) : “ +t);

}

@Test

public void testResponseTimeInUnit(){

long t = given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/1/).timeIn(TimeUnit.MILLISECONDS)"

http://jsonplaceholder.typicode.com/photos/1/).timeIn(TimeUnit.MILLISECONDS)

;

System.out.println(“Time(ms) : “ +t);

}

@Test

public void testResponseTimeAssertion (){

long t = given().get(

HYPERLINK "http://jsonplaceholder.typicode.com/photos/1/).timeIn(TimeUnit.MILLISECONDS)"

http://jsonplaceholder.typicode.com/photos/1/).time(lessThan(500L))

;

}

}

Wordpress REST API

HYPERLINK "http://demo.wp-api.org/wp-json/wp/v2/posts"

http://demo.wp-api.org/wp-json/wp/v2/posts

HYPERLINK "http://www.jsoneditoronline.org"

http://www.jsoneditoronline.org

pairwise testing

disable test

Junit @Ignore(“not implemented”)

TestNg :@Test(enabled = false)

TestNg

@Test(timeout=1000)

Difference between Assertion and verification

Verification and assertion is little different in software testing so don't be confused. Verification will just verify but assertion will first verify and if result is not as per expectation then it will stop execution of that specific test method.

Pairwise Testing also known as All-pairs testing is a testing approach taken for testing the software using combinatorial method. It's a method to test all the possible discrete combinations of the parameters involved.

What is TestNG:

TestNG is a testing framework inspired from Junit

TestNG has multiple classes interface and methods which will make testers task easy.

TestNG provides so many additional functionality using them you can create very robust framework itself.

Advantages of TestNG

.Default reporting

.Annotations

.run your tests in arbitrarily big thread pools

.Easy and flexible test configuration

Support for data-driven testing (with @DataProvider)

Support for parameters

Easy way to execute TestSuite

.Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven etc)

.grouping features and many more

The priority can set the order for the execution of the tests

@Test(priority=, description=””)

Skipped the test

@Test(enabled = false)

How to create dependency between test cases in selenium webdriver

@Test(dependsOnMethods=” “)

HYPERLINK "https://www.youtube.com/watch?v=sPhYn0K8d5k"

https://www.youtube.com/watch?v=sPhYn0K8d5k

One examples:

Import org.testng.annotations.Test;

public class MyApplication2{

Webdriver driver;

@BeforeClass

Public void startBrowser(){

driver = new ChromeDriver();

}

@Test

Public void startApp(){

driver.get(

HYPERLINK "https://enterprise-demo.orangehrmlive.com/auth/login"

https://enterprise-demo.orangehrmlive.com/auth/login

);

driver.mange().timeouts().implicitlyWait(30, TimeUnit.SECOND);

String currentURL = driver.getCurrentUrl();

Assert.assertTrue(currentURL.contains(“auth/login”));

}

@Test(dependsOnMethods=”startApp”)

public void loginApp(){

driver.findElement(By.xpath(“.//\*[@id=’txtUsername’]”)).sendKeys(“admin”);

driver.findElement(By.xpath(“.//\*[@id=’txtPassword’]”)).sendKeys(“admin”);

driver.findElement(By.xpath(“.//\*[@id=’btnLogin’]”)).click();

boolean status = driver.findElement(By.xpath(“.//\*[@id=’menu\_admin\_viewAdminModule’]/b”)).isDisplayed();

Assert.assertTrue(status);

}

@Test(dependsOnMethods=”loginApp”)

public void logoutApp(){

driver.findElement(By.xpath(“.//\*[@id=’welcome’]”)).click();

driver.findElement(By.xpath(“//a[test()=’Logout’]”)).click();

Assert.assertTrue(driver.findElement(By.xpath(“.//\*[@id=’txtUsername’]”)).isDisplayed());

}

@AfterClass

public void closeApp(){

driver.quit();

}

}

HYPERLINK "https://www.youtube.com/channel/UCcTII5pbZYkU4fgFtb4uesg"

Mukesh otwani

(a great instructor for explaining web testing in Selenium)

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on.

How Jenkins works:

Developers check their source code -> Jenkins will pick up the changed source code and trigger a build and run any tests if required -> the build output will be available in the Jenkins dashboards. Automatic notifications can also be sent back to the developer.

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals

 Automated software testing is important due to following reasons:

Manual Testing of all work flows, all fields , all negative scenarios is time and cost consuming

It is difficult to test for multi lingual sites manually

Automation does not require Human intervention. You can run automated test unattended (overnight)

Automation increases  speed of test execution

Automation helps increase  Test Coverage

Manual Testing can become boring and hence error prone.

ROI: Return on Investment

The following category of test cases are not suitable for automation:

Test Cases that are newly designed and not executed manually  atleast once

Test Cases for which the requirements are changing frequently

Test cases which are executed on ad-hoc basis.

When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing. Such kind of tests are executed only once unless we uncover the defects

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application

While drafting a test case do include the following information

The description of what requirement is being tested

The explanation of how the system will be tested

The test setup like: version of application under test, software, data files, operating system, hardware, security access, physical or logical date, time of day, prerequisites such as other tests and any other setup information pertinent to the requirements being tested

Inputs and outputs or actions and expected results

Any proofs or attachments

Use active case language

Test Case should not be more than 15 steps

Automated test script is commented with inputs, purpose and expected results

Setup offers alternative to pre-requisite tests

With other tests, it should be incorrect business scenario order

Test Basis is defined as the source for creation of test Cases. It can be the Application itself or the requirement documents like SRS (Software Requirement Specification), BRS (Business Requirement Specification), etc.

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

In other words, it is a document that maps and traces user requirement wi

ith test cases. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should miss while doing Software testing.

 We need an easy way or special techniques that can select test cases intelligently from the pool of test-case, such that all test scenarios are covered.

 We use two techniques - Equivalence Partitioning & Boundary Value Analysis testing techniques to achieve this.

The basic idea in boundary value testing is to select input varia

at their:

Minimum

Just above the minimum

A nominal value

Just below the maximum

Maximum

Decision Table Testing is a good way to deal with a combination of inputs, which produce different results. It helps reduce test effort in verifying each and every combinations of test data, at the same time ensuring complete coverage

State Transition testing is defined as the testing technique in which changes in input conditions cause's state changes in the Application under Test (AUT).

Normal (Web)