(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 [Oracle Docs](http://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html),  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[difference between iterator and enumeration](http://javahungry.blogspot.com/2013/06/difference-between-iterator-and-enumeration-collections-java-interview-question-with-example.html).

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.

### [Jenkins](https://jenkins.io/)

|  |
| --- |
| //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)

<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”, [test@test.com](mailto:test@test.com)).

when().

post(<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([http://....).path(“thumbnailurl”)](http://....).path(\“thumbnailurl\”));

Or

String href = get([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(<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([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(<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(“<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(<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”, <https://spi.fonts.com/rest/json/Accounts/>).

then().

statusCode(400);

}