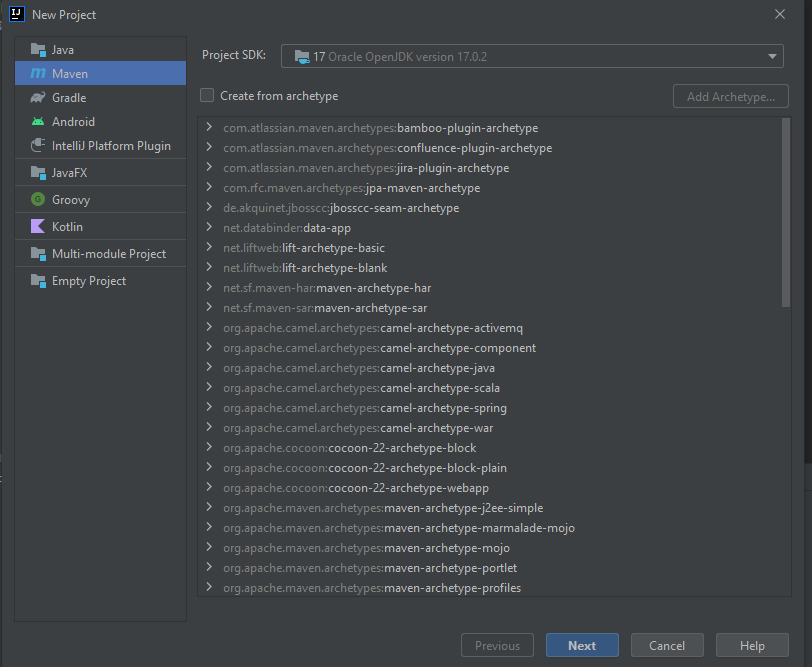
**Seccion 5 – Rest Assured Framework using BDD Approach**

**17. Environment setup:**

* Para esto se debe tener Java, Elicpse or intellij, Maven , TEstNG y las dependencias de Rest Assured
* Creamos un proyecto Maven



* Luego se deben instalar las dependencias:

1. RestAssured
2. Json Scheme validator
3. Json-pah
4. Xml path
5. Test NG
6. Java-hamcrest
7. Hamcrest-core

* En el archivo Pom.xml se deben poner las dependencias y las obtendremos del repositorio Maven
* Para BDD es importante tener las ultimas 3 dependencias

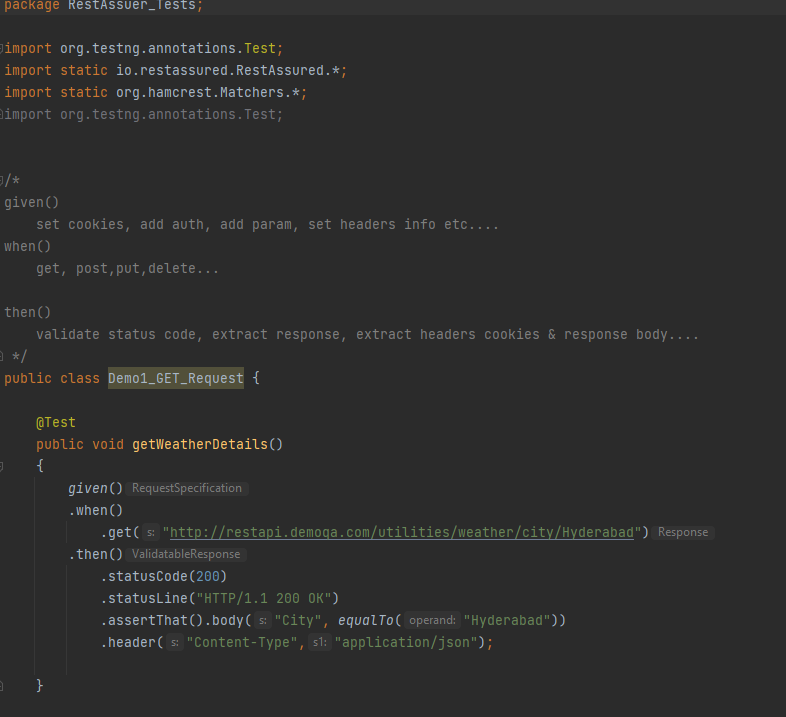
**18. Keywords used in BDD Approach & How to send GET Request:**

* En rest assured se divide en varias secciones

1. **Given():** set cookies, add auth, add params , set headers info.. etc
2. **When():** get, post, put, delete ….
3. **Then():** validate status code, extract response, extract headers cookies & response body

Para cada sección se usan diferentes comandos o palabras clave

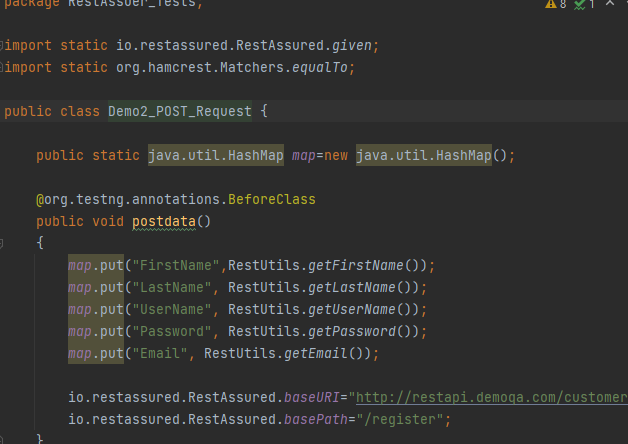
* En la carpeta de la clase se puede ver el archivo a usar



* Para que los given, then y then funcionen se deben tener las dependencias adecuadas como la hamcrest matchers
* En este caso en al seccion given no tenemos nada que poner
* Dentro del When poindremos el get y el url
* Dentro del when pondremos lo que queremos verificar

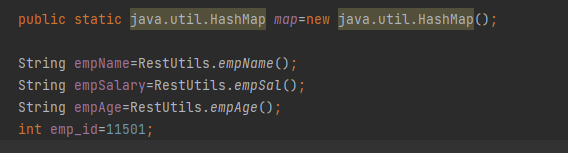
**19. How to send POST Request**

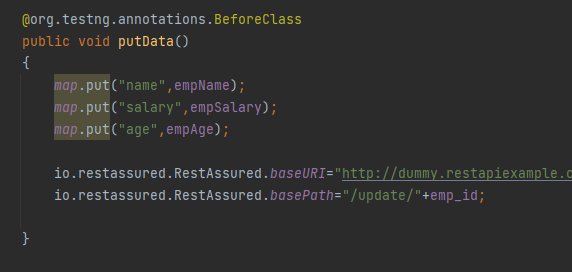
* Para este se debe tener lo que se va a enviar
* En este caso tendremos un archivo Restutils para generar los valores
* Lo que se hace con el map es almaxenar varios valores en esta clase
* Se llaman los datos desde el Restutils



**20. How to send PUT Reques:**

* Para el Put Tambien se debe tener algo que enviar.
* EL ejemplo se puede ver en la carpeta de la clase
* Al inicio se crea un hash map para poder halar de allí los datos que se necesitan





Texto

Descripción generada automáticamente

**21. How to send DELETE Request:**

* En la carpeta de la clase se puede ver el archivo

**22. Basic validations on JSON Response**

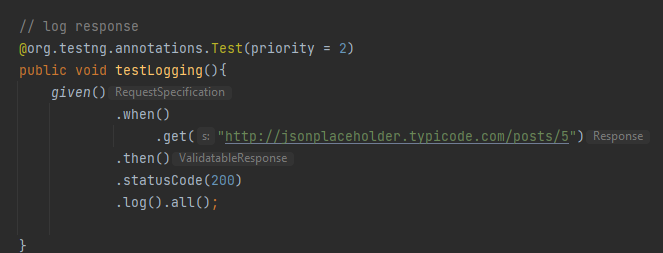
* Hay muchas cosas que se pueden validar en un JSON, por ejemplo:

1. Status code
2. Log response
3. Verifyings single content in response body
4. Verifying multiple content

Texto

Descripción generada automáticamente

* Si se quiere mostrar el LOG se pone .log().all
* Algo importante es que a los casos de prueba se les puede poner una prioridad y los casos se ejecutarán en ese orden
* Para el log se usa así:



* Para verificar algo especifico en el responde body, se debe tener en cuenta como es la respuesta, si tieen cosas dentro de la otra o no. Si es así se debe ingresar hasta este valor. A veces es necesario un buscador de valores de JSON o Json path Finder.
* Para estos pasos es importante tener 2 dependencias



* Queda entonces

Texto

Descripción generada automáticamente

* Para validar múltiple contenido usremos la Api <https://reqres.in/api/users?page=2> si se hace la request se ve que todos los usuarios tienen los mismos datos adentro. Entonces verificaremos en este caso que los Id que queramos estén dentro de la respuesta.



* También se pueden setear parámetros y headeres para enviar las request
* Esto se debe poner en el given
* Se ua el .param y dentro se pone el key y el valor

Texto

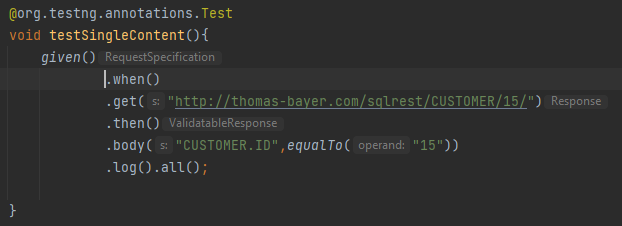
Descripción generada automáticamente

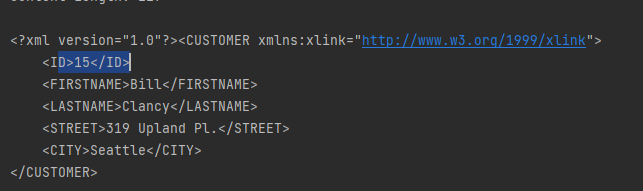
**23. Basic validations on XML Response:**

* En las respiestas XML Tambien podemos revisar ciertas cosas como por ejemplo:

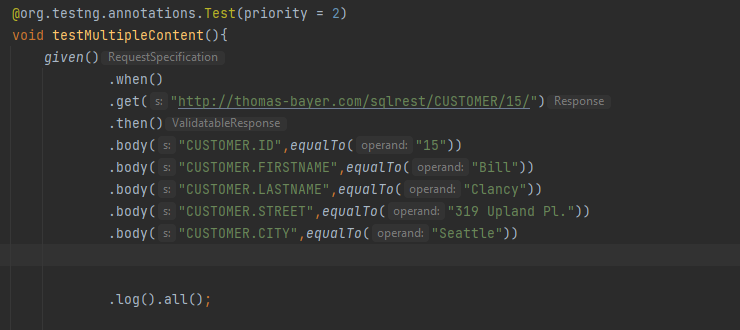
1. Single content
2. Multiple content
3. Verifying all the content in one go
4. Find value using XML path
5. Xpath with text() fuction

* Crearemos entonces una nueva clase de java para poner los ejemplos
* Se usará la API : <http://thomas-bayer.com/sqlrest/CUSTOMER/15/>
* En la seccion then es donde verificamos el body
* Para el single content,d ebemos buscar el XML path





* Para veririfcar contenido multiple se hace de la siguiente manera: se crean diferentes valdiaciones en el body. Se debe poner el XMl file



* Otra manera de hacer es ponerlo todo en la misma línea. Se debe poner todo junto y se debe poner el .text()

Texto

Descripción generada automáticamente

* Ahora, buscaremos con el xml path. Se usa hasXpath y al final lo que contiene con contant.String



* Haremos un nuevo ejemplo con otra Api ncon mas datos: <http://thomas-bayer.com/sqlrest/INVOICE/>

Texto

Descripción generada automáticamente

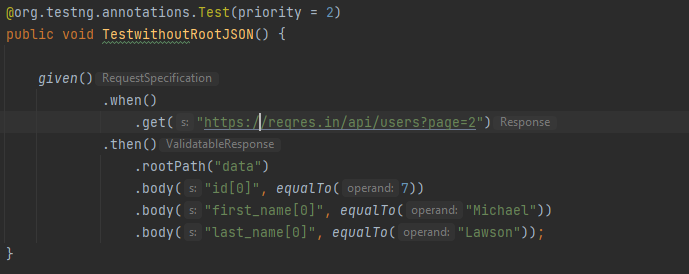
**24. Root Settings, rootPath & detachRootPath**

* Se pude hacer unr equest y bsucar cosas en el body sin usar el root, es decir, hay que repetir muchas veces un mismo texto. Este se hace en JSON

Texto

Descripción generada automáticamente

* También se pude añadir un root path
* En esta caso también se puede usar el equaltoo se puede usar uno nuevo que se llama “is”. Ambos son equivalentes



Pantalla de un video juego

Descripción generada automáticamente con confianza media

* Hay otra manera que es usando detachRootpaht
* Se usa cuando en el request se tiene varias partes, en este momento no tenemos una APï así pero escribiremos el ejemplo

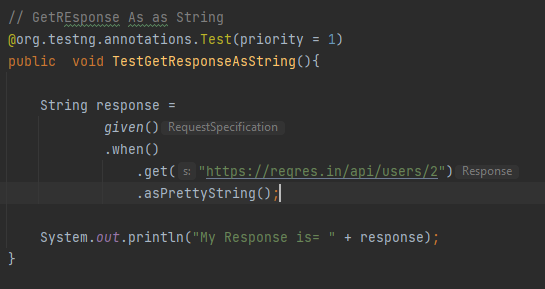


* Esto mismo se puede usar en XML



**25. Different ways to read Response:**

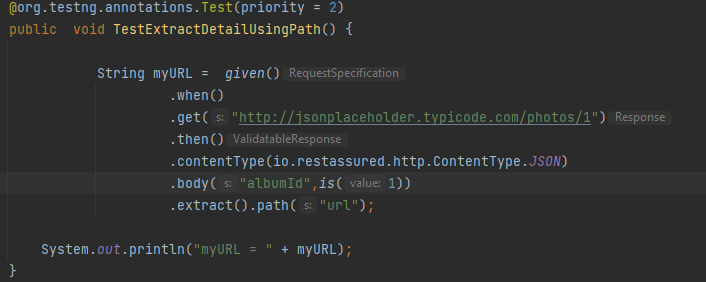
* Hay diferentes formas de tomar la respuesta en esta clase se verá cuales son
* Recordemos que si nos e tiene nada importante en el given este se puede quitar
* AsString



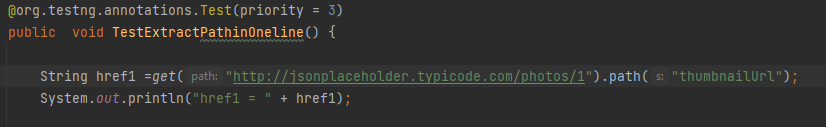
* Todo lo anterior se puede poner en una sola línea.

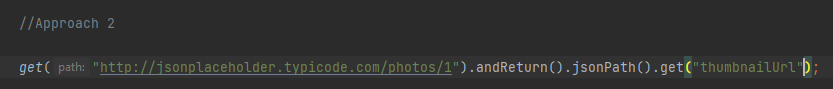


* Otra forma es extraer detalles con el path . Se pone el pagh y nos retorna el valor exacto que queremos



* Todo lo anterior se puede poner en una sola línea . En este caso lo haremos de 2 formas





Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Estas respuestas se puede reutilizar

Texto

Descripción generada automáticamente

* También podemos extraer toda la respuesta y almacenarla para usarla luego
* Se debe importar el Response de rest asssured

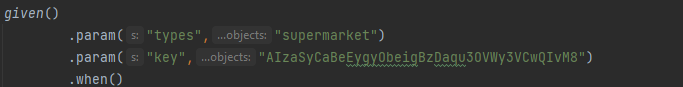


Texto

Descripción generada automáticamente

**26. Working with Response Headers:**

* Usaremos la API de Google: <https://maps.googleapis.com/maps/api/place/nearbysearch/xml?location=-33.8670522,151.1957362&radius=500&types=supermarket&key=AIzaSyCaBeEygyObeigBzDaqu3OVWy3VCwQIvM8>
* Con esto vamos a capturar diferentes valores de los headers que se reciben
* Vamos a pasar algunos de los valores de los parámetros a parte



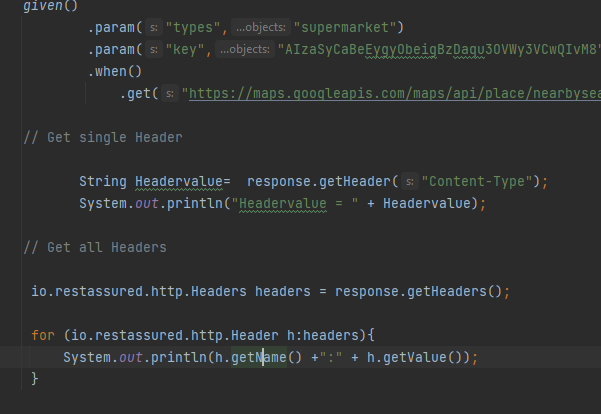
* Para obtener los headers se una tdentro del then el .header y se especifica el valor



* Ahora vamos a obtener todos los headers ya tomar de allí solo 1



* Para obtener todos los headers



Texto

Descripción generada automáticamente

**27. Working with Response Cookies:**

* En esta clase vermeos lo siguiente:

1. Como validar cookies desde la respuesta
2. Obtener (GET) cookies especificas
3. Obtener todas las cookies
4. Obtener información detallada de las cookies

* Usatemos la API <http://jsonplaceholder.typicode.com/photos>
* La APi cambio y ya no devuelve cookies, se hará el desarrollo pero no se puede validar
* Primero validaremos una cookies y su valor

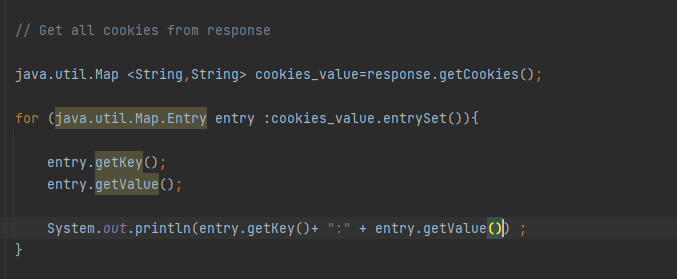
Texto

Descripción generada automáticamente

* Para extraer una cookie en especifico



* Para obtener todas las cookies se llama la funcion .getCookies().
* Se usa la funcion Map para extraer cada nombre con su valor
* Usamos un for para acceder al map y sacar juntos el nombre y valor de la cookie



* Para obtener información detallada del cookie
* Se crea un objeto del tipo Cookie, y lueg con esto objeto almacenado se puede extraer información con diferentes métodos como .hasExpireDate()



**28. Working with Authentication API's:**

* Muchas APIs tiene alguna autentuicacion, en este caso se tienen :

1. Basic /digest
2. Bearer toque

* La autenticación quiere decir que se necesita un usuario y un password para poder ingresar
* En este momento no se tiene API con autenticación pero igual se hará el desarrollo necesario
* Para la autenticación básica se tiene esto:

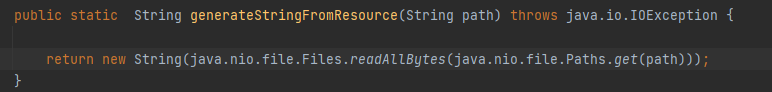


* Para el Digest se hace de una manera similar

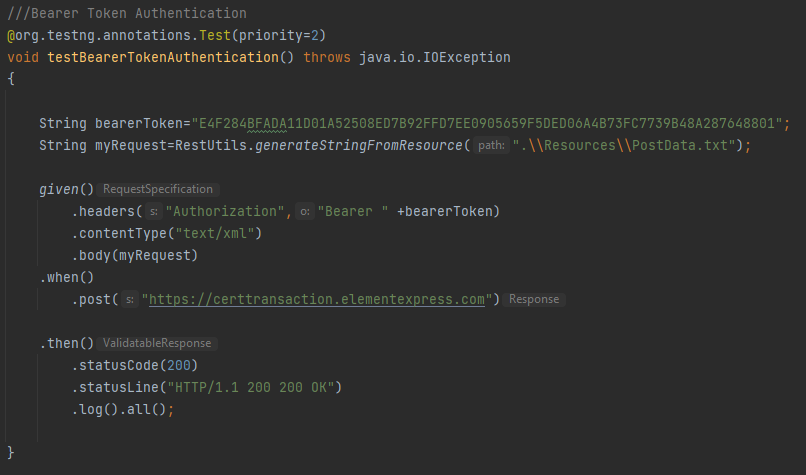


* Para el Bearer token se debe hacer con un POST ya que se debe enviar cierta información. Para este se usara el API <http://certtransaction.elementexpress.com>

Como se debe enviar unos datos, se crea un xml que los contenga y se crea en el REeutils una nueva funcion para obtener los datos de este archivo



* Luego se llama desde la prueba
* En el given se ponen los datos de autorización, tipo de contenido y cuerpo a enviar
* En el when se hace un post



**29. Types of Parameters & Working with API parameters:**

* Se veran varios tipos de parámetros que se pueden usar

1. Query parameters
2. Path parameters

* En esa clase se verá como pasar los patametros de forma dinámica
* Se usará el endpoint <http://jsonplaceholder.typicode.com/comments?postId=1>
* En este caso se pasará el postId = 1 como un query parameter



* Esta es la forma de hacer un query parameter, ahora veremos como hacer un path parameter



**30. Request Specification Builder & Response Specification Builder:**

* Algunas veces necesitamos validar las mismas cosas o pasar los mismos parametros para diferentes APIS
* Para esto en el Caso de TestNG se va a crear un setuop para pasar los parámetros y la validación de las respuestas

Setup()

{

Request params

Respoonse validation

}

* Se deben crear 2 variables

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Se deben crear unos objetos para poder usarlos y llamar lo que se necesite

Texto

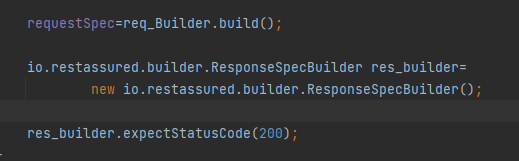
Descripción generada automáticamente

* Luego dentro de la clase setup se crea el objeto y se añaden los parámetros que se quieren usar

Texto

Descripción generada automáticamente

* Se pueden agregar cookies, body y muchas otras cosas
* Luego se debe asignar los parámetros a las request especification



* Ahora, para todos los TEST que se creen de acá en adelante en esta clase, todos estos datos serán comunes, tanto los parámetros como las validaciones de las respuestas

Texto

Descripción generada automáticamente

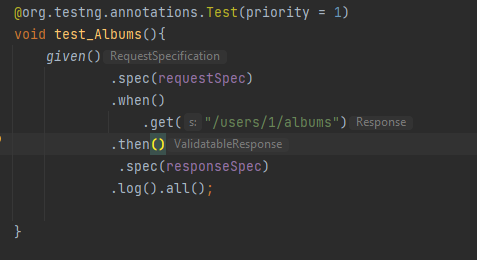
* Usaremos entonces 3 API

<http://jsonplaceholder.typicode.com/users/1/albums>

<http://jsonplaceholder.typicode.com/users/1/todos>

<http://jsonplaceholder.typicode.com/users/1/post>

* Para poder usar lo anteriormente mencionado se usa el .spec() y se llama lo que se necesita



* ­Para las otras request se usa lo mismo, solo se cambian los parametros del path



**31. Logging:**

* Esto se usa para capturar los logs
* Parta este ejercicio se usará la API <https://reqres.in/api/users?page=2>
* Con .log().body() se obtiene los datos del body del request
* Igualmente se puede a ver con los headers

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* Igual con las cookies

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

* Para capturar todo de una vez, se puede usar el .log().all()



* Para la segunda parte corrermoe sn esta API <https://reqres.in/api/users/23> . Esta no nos da respuesta pero si trae cookies y headers
* Con el log también se puede mostrar solo si tiene errores

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Un error es diferente de una falla, en este caso error es cuando no se esta recibiendo nada
* También se puede mostrar solo si se tiene cierto status code, en este caso 404



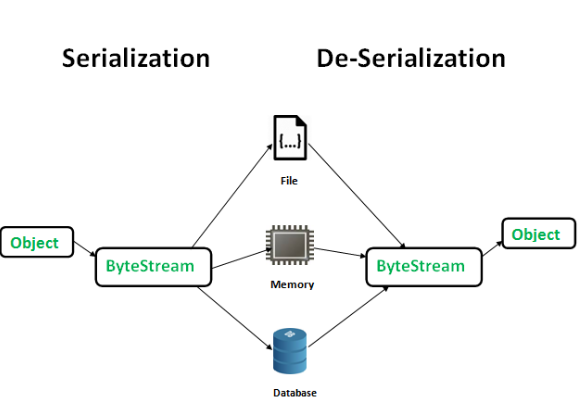
* Se puede mostrar el log en caso que una validación falle, en este caso también se debe agregar la validación

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**32. Serialization & De-serialization:**

* Este es un concepto relacionado con Java
* Serialización es un proceso de pasar un objeto (o poner el estado del objeto) que se tiene en java a otro tipo de archivo
* De-serializacion es convertir (leer el estado del objeto) ese archivo en particular en una objeto de java
* Como se crea un txt y no se la da una path especifico este se crea en la misma carpeta



Texto

Descripción generada automáticamente

* Este archivo contiene lo que se puso en la clase test
* Ahora el movimiento contrario, del archivo queremos extraer la clase

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* Esto se usa generalmente cuando se quiere hacer un POST, en este caso se puede mapear como hemos hecho anteriormente o también se puede simplemente crear un objeto como el de esta clase, hacer una serialización o deserialización y enviar el formato Json

**33. Rest API Testing using using Serialization & De-serialization JSON:**