**Seccion 4: Rest API Testing using RestAssured Automation**

**13. Overview on RestAssured API, Setting up Environment & Automating Rest API Tests:**

* Rest assured es más o menos una API que se usa para probar y automatizar la prueba de APIS
* Se pueden hacer diferentes tipos de request, y se pueden hacer validacviones en las respuestas.
* Es solo para servicios REST, no funciona para SOAP
* La pagina oficial es <https://rest-assured.io/> donde se puede encontrar mucha información
* Pre requisitos:

1. Java Instalation
2. Eclipse (IntelliJ idea)
3. TestNG: Framework quye es implementado encima de Java y es usado para organizar los casos de prueba, test suites.
4. Maven

* En Maven se van a especificar varias dependencias
* Cuando se crea un proyecto Maven, se crea un archivo llamado pom.xml el cual debemos actualizar
* Test NG se debe descargar de <https://mvnrepository.com/>
* En Inellij se va a file – Project structure – modules – dependencias se busca el lugar donde se descargó el testng y se instala
* En nave se va a code y añadir dependencias
* Vamos a necesitar varias dependencias:

1. REstAssured
2. TestNG
3. Json-simple
4. Apache poi

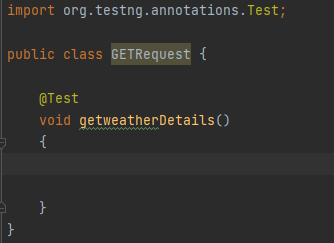
* Estas dependencias se pueden descargar de Mavenrepository, se buscan en el repositorio y se copia la dependencia
* Tener muy en cuenta la versión de la dependencia para poderla descargar sin problemas.
* Para la descarga del Apache POi se tiene que descargar 2. Esto para tener soporte de archivos de excel



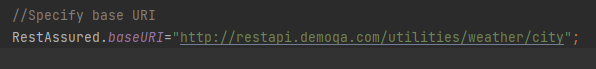
* ­En el archivo de Excel de la clase hay una archivo de Excel con varias request las cuales se van a hacer. También se capurará la respuesta y se mostrará
* Creamos una nueva clase

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



* Primero se dbe especificar la URL base



* Luegoi se crea el objeto del request, el objeto del responde.

Texto

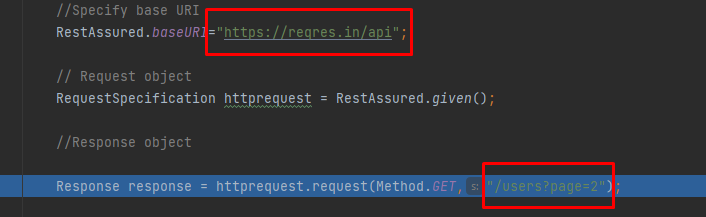
Descripción generada automáticamente

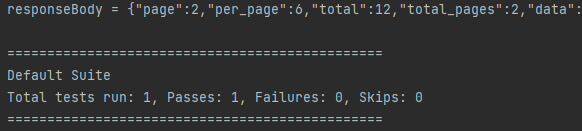
* Luego podemos sacar el body e imprimir . Lo pasamos a string porque generalmente llega como un json

Texto

Descripción generada automáticamente

* Es este caso el Api usado esta deprecado, se uso uno viejo cambiando los valores





* API usada: <https://reqres.in/api/users/2>

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* Ahora vamos a verificar algo en la respuesta

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Una captura de pantalla de una computadora

Descripción generada automáticamente

* Para el POST se reutiliza casi todo el código del anterior, pero debemos crear el POST

Texto

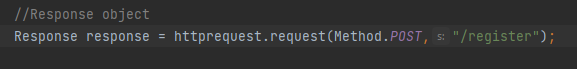
Descripción generada automáticamente

* En el header se debe especificar el formato y transofmra lo que ponemos en formato JSON

Texto

Descripción generada automáticamente

* Luego se crea el objeto para enviar



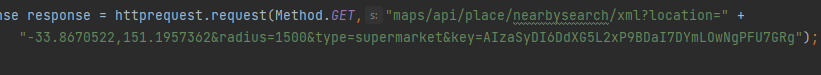
Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

* Para el tercer ejemplo se hará también un GET pero ahora validando los headers
* La key generada para mi es AIzaSyDI6DdXG5L2xP9BDaI7DYmLOwNgPFU7GRg



* Verificamos en postman cuales son los headers que trae esta request

Texto

Descripción generada automáticamente

* Verificamos header

Texto

Descripción generada automáticamente

Texto, Escala de tiempo

Descripción generada automáticamente

**14. Validating API Responses, Status codes & Status lines:**

* Vamos a traer y revisar todos los headers. El archivo se llama GetAllHeaders
* Podemos usar validaciones de las usadas en ejercicios anteriores
* El método response.headers() retorna todos los headers.
* Los headers contienen 2 aprtes, hey y value
* Importamos la clase Headers de rest assured, esto capturará todos los headers

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Ahora debemos crear un for para sacalor todos, la key y el valor

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* De las request se pueden verificar: Status code, status line, response time, headers, response body,etc.
* Para esta pruesba usaremos el endpoint de los users

Texto

Descripción generada automáticamente

* Vamos a verificar que tiene el body con un assert



Texto

Descripción generada automáticamente

* Ahora miraremos como validar todos los valores
* Para obtener esto, en ves de tomar todo el body, se debe tomar cada uno de los valores, extraer cada valor.
* Se usa jsonpath para extraer los valores individuales



* De este path se extrae cada uno de los elementos. Cada uno de estos muestra el valor. En este caso se debe unsar un Json Finder para poder sber como buscar los valores, como estos están primero dentro de “data” hay que buscar la ruta completa.

Texto

Descripción generada automáticamente

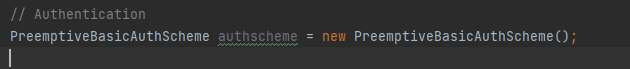
Texto

Descripción generada automáticamente con confianza baja

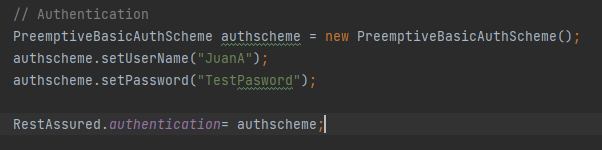
* Se puede hacer validaciones de la respuesta con un valor esperado:



* A veces las APIs necesitan una autenticación previa como un email, usuario y un password. En este momento no se tiene API para hacer la prueba pero se hará e código
* Para la autenticación se debe usar una clase especial



* Se debe crear un objeto de esa clase



* Los siguientes pasos son los que se usan automarizar

1. Definir la basURI
2. Crear el objeto request
3. Crear la respuesta
4. Crear el respondebody

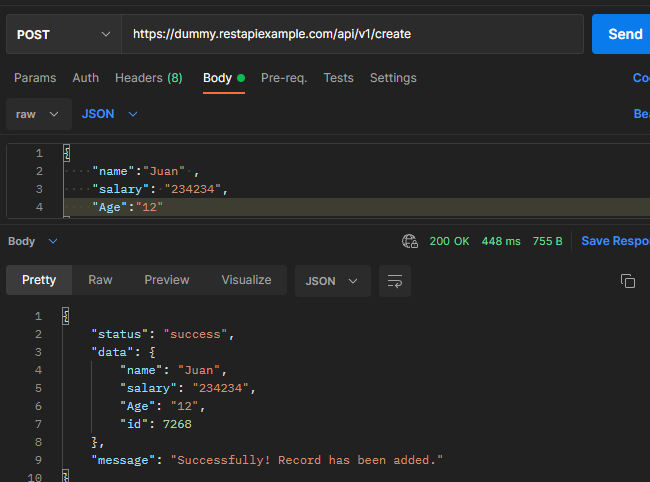
* Algunas de las validaciones que se pueden hacer son:

1. Status code
2. Status line
3. Headers (Content.type, content -length , content encoding, etc ….)

* Con Restassures Tambien se puede hacer DDT (Data driven test) para esto se debe tener:

1. Datos de prueba en Excel (Apache POI) – preparar los datos
2. Descargar apache POI y adjuntar jars al path del proyecto o Añadir Apache pio dependency en el pom.xml (ya las añadimos al inicio del modulo en el archivo pom)
3. Preparar un XL utility file (Java class file) el cual leerá los datos del archivo de Excel
4. Escribir un test den TestNG con método DataProvider

* El método DataProvider es el que se encarga de proveer los datos para el caso de prueba. Si tiene 5 datos, el TC se ejecutará 5 veces
* La api que se usará será: https://dummy.restapiexample.com/api/v1/create
* Se puede ver que la API esta funcionando



* Se debe crear el uri, el objeto y un json object para almacenar los datos



* Para enviar el json se debe especificar el content type



* Se añade luego los datos del JSON

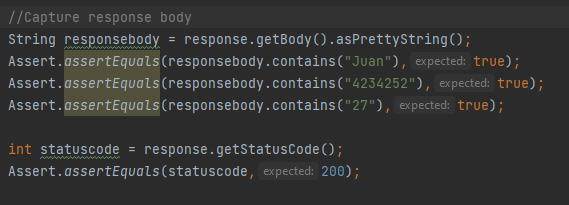


* En este momento ya se tiene toda la información necesaria para enviar el post, ahora se debe crear el request

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

* Caprturamos la respuesta y ponemos algunios asserts

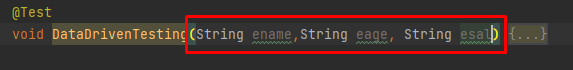


* Con el DDT lo que se quiere hacer es repetir el mismo test pero con diferentes datos
* Se creara un nuevo método dentro del mismo test case
* En este nuevo método se pueden mter los datos para pasar al método principal. Cada una de las llave sería una fila y los valores se ubicarían en una columna diferente

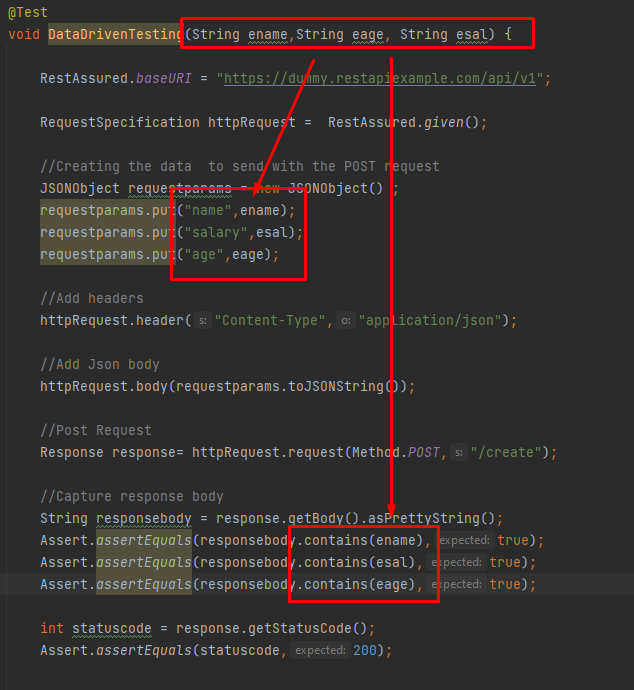
Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente

* Ahora, los valores se que pusieron en esta nueva clase, se deben ingresar a la clase de arriba



* Estos valores se deben poner en la primera clase, donde van los Json, en vez de harcodear se debe usar esta variable



* Además hay que invitarle al TEST de donde va a tomar los datos





* Ahora.. en vez de tener los datos harcodeados, los podemos traer desde el Excel
* En la documentación de la clase se encuentra el archivo XL utilities que ayuda cuando se tiene algún Excel. Ayuda a contar las columnas, filas, etc
* El archivo de Excel se puede encontrar en los documentos de la clase
* Se reutilizará parte del código usado con los datos harcodeados
* El Excel se debe poner en la misma carpeta del proyecto

Texto

Descripción generada automáticamente con confianza media

* Se debe crear una variable con el path del archivo

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Para leer el numero de filas y de columnas se utiliza el XL utils

Texto

Descripción generada automáticamente

* Debemos crear un for anidado para saber el numero de filas, columnas y recorrerlas

Texto

Descripción generada automáticamente

**15. Rest Assured Framework using TestNG, Maven,Jenkins, Git & GitHub - Part 1**

* Los frameworks tiene unos objetivos

1. Re usabildiad
2. Mantenimiento

* Esto para mantener la constancia y que otros lo puedan usar
* En el mercado hay muchos frameworks diferentes, TEstNG, cucumber, JUnit etc.
* Nosotros mismos podemos crear un framework o podemos usar los que queramos
* Encontramos entonces varios pasos para poder elegir un framework antes de empezar a automarizar:

**Analyse**

* Pre-definidos: TestNG, Junit, Cucumber, Robot, etc… (Herrameintas para probar diferentes cosas, APIS, test unitarios, etc).
* Customized/user defined framewok: Data driven testing, Hybrid framework, etc..
* Hybrid Approach: RestAssuerd, TestNH, Maven, Jenkings (CI), git-Github.
* API Test cases

**Implementation of automation Framework**

1. Implementation/development
2. Execution
3. Maintainance
4. Implementation/development
5. Maven Project: Para usar las dependencias, también sirve para ejecutarlo en diferentes partes (Eclipse, IntelliJ)
6. Update pomn.xml
7. Crear file structure

* Una estructura de archivos debería quedar como

Project\_name

|

Base (package )(donde se tiene todo lo común)

|

Test cases (package )

|

Utilities(package )

|

Logs (Folder)

|

Reports (Folder)

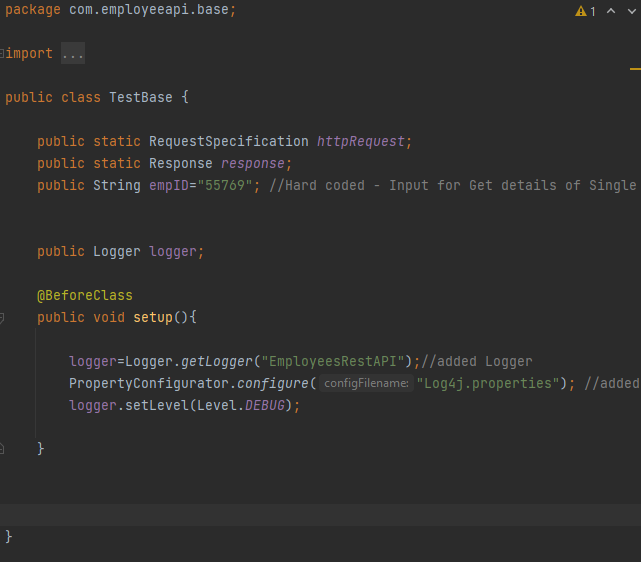
|

Pom.xml

|

TestNG.XML

* En kla carpeta de la case se encuentra un archivo con los test cases que vamos a automatizar
* En la misma carpeta esta el proyecto que vamos a usar para esta clase
* En el archivo TestBase se encuentra todo lo que se va a usar en todos los tes cases, para no tener que repertirlo, se llaman desde allí.



* Para llamarlo desde un test case se usa:

Texto

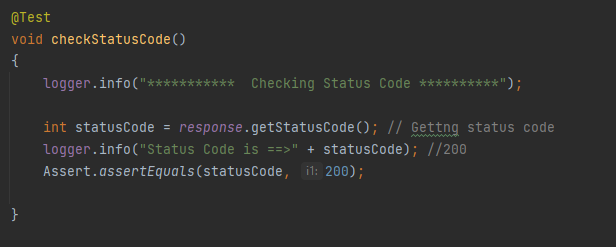
Descripción generada automáticamenteTexto

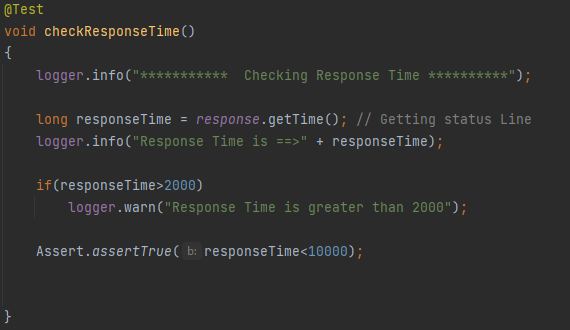
Descripción generada automáticamente

* Se envia la request



* Luego se hacen las validaciones:





Texto

Descripción generada automáticamente

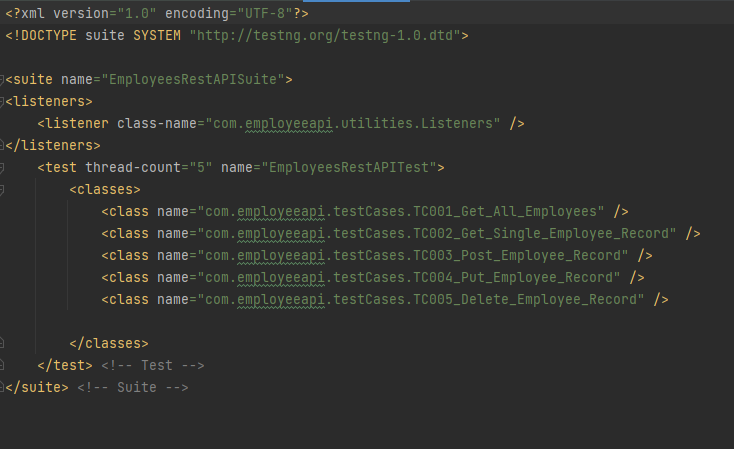
Texto

Descripción generada automáticamente

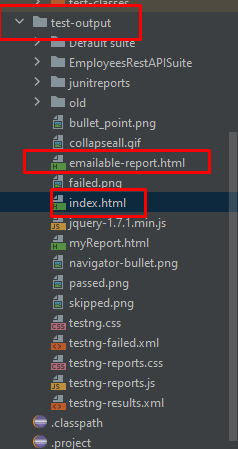
Texto

Descripción generada automáticamente

* En el proyecto y archivo se pueden ver el resto de casos de prueba
* En el la carpeta Logs, se pueden ver los logs de cada una de las request hechas
* Se puede crear un archivo XML donde se ponen los casos que se quieren para una Suite



* En la carpera Test-output se pueden ver reportes como el index.html



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Para generar los reportes se debe:

1. Agregar la dependencia extend report en pom.xml
2. Tener un utility file ne el cual se puede generar reportes (Listeners ) que dice que pasa cuando falla, que dice cuando pasa, etc…

**16. Rest Assured Framework using TestNG, Maven,Jenkins, Git & Github-Part 2**

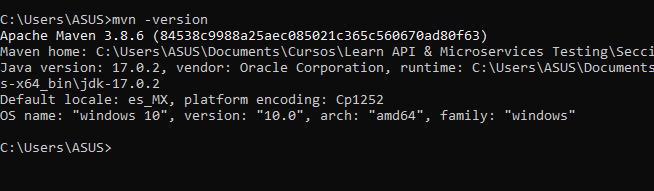
* En esta clase se verá la parte de la ejecución y mantenimiento
* De cuantas formas se pude ejecutar el framework/test cases

1. TestNG.xml
2. Pom.xml
3. Command promt
4. run.bat
5. Jenkins

* En el pom se agregan unos plugins para poder ejecutar el proyecto a través de pom.xml- Los plugins son Maven-compiler-plugin y Maven-surefire-plugin



* Para ejecutar desde el command promt se debe primero ubicar la dirección exacta del proyecto
* Antes de correrlo debemos instalar Maven software en nuestro sistema operativo
* Para descargarlo poner Apache Maven en Google. Luego se extrae
* Luego se debe configurar la dirección de la carpeta bin , y se debe agregar a las variables de entorno y se pone en la variable path
* Luego se verifica si sí se instaló bien poniendo ene l command promt mvn -version



* Para correrlo se pone en la carpera del proyecto, en el cmd mvn clean install . Con esto se ejecuta el proyecto.
* Otra cos que se puede hacer es crear un bar file , en el cual se pone la dirección y el comando Maven del punto anterior y al correr el run.bat se corre todo el suit

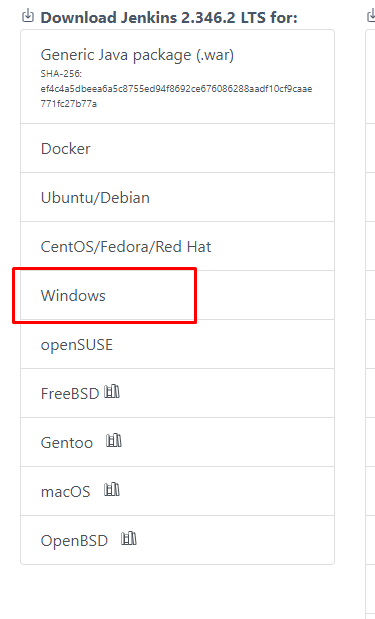
Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente con confianza media

* Por ultimo se tiene Jenkins, la cual es una herramienta de integración continua que sirve para realizar pruebas en un pipeline
* Para poderlo correr en Jenkins primero se debe instalar Jenkins. Iremos a la pagina de Jenkins para poder descargar. descargaremos el paquete Java de Jenkins



* En mi caso Jenkins no está funcionando
* Para hacer control de versiones se usa Git y Github.