Instituto de Ciencias Tecnológicas

Alumno: Mauricio Troncoso
Profesor: Abel Saavedra
http://localhost:8090/index.html

Login en Api Rest

Hay distintas maneras de hacer un login en una aplicación API Rest-SpringBoot. Puede ser a través de AWS Token, A través de la dependencia *Security* o como en este caso a través de una validación en una base de datos.

Esta es una continuación de documentación del proyecto API-REST presentado en la ACT3.

Al igual como explique en la documentación anterior, nuestro **Login** que en la API se llamara **Usuario,** también posee la estructura **CERS ->** CONTROLLER, ENTITY, REPOSITORY y SERVICE

Entity:

Primero creamos nuestra entidad, en este caso será “usuario”

```
package com.example.demo.entity;
import javax.persistence.Column;

@Entity
@Table(name="usuario")
public class usuario {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="id", unique =true ,nullable=false)
    private long id;
    @Column(name="user",length=20,nullable=false)
    private String user;
    @Column(name="pass",length=20,nullable=false)
    private String pass;
    @Column(name="role",length=20,nullable=false)
    private String role;
```

@Entity -> Registra nuestra clase como una entidad
@Table-> señala la creación de la tabla
@Column -> Señala la creación de una columna
@ID -> Nos genera la llave primaria como ID y junto a **@GeneratedValue** la hacemos Auto Incrementable.

Luego se generan los constructores a utilizar y los **Getters and Setters.**

Repository:

En este caso, el Repository va a ser muy diferente de los vistos en la documentación anterior con las entidades anteriores. Ya que la función de mi **Usuario** es buscar en la base de datos usuarios con parámetros específicos (User y Password), se debe crear una Query modificada para realizar lo necesario en la Base de Datos.

```
@Repository
public interface usuariorepository extends JpaRepository<usuario,Long>{

 public  usuario findByUser(String user);

@Query(value="Select role FROM usuario WHERE user = :user and pass = :pass")
    String findbyuserandpass(@Param("user") String user,@Param("pass") String pass);
}
```

Con los parámetros **@Repository** indicamos que esta clase será un Repository y con el parámetr**o** **@Query** Genero una query personalizada como se ve en la imagen. En este caso, la Query busca un **role** en la tabla **usuario** cuando el **user** y **pass** sean igual a lo recibe en los parámetros de abajo, que están separados por **@Param** y son enviados a través de la invocación del **Service** en el **Controller.**

Service:

La capa **Service** del Login solamente tiene un método que es **getrol** este método recibe dos parámetros y los envia a **findbyuserandpass** que se encuentra en el **Repository** y con esos parámetros se realiza la **Query.**

```
7  @Service
8  public class UsuarioService {
9
10 @Autowired
11 private usuariorepository userRepo;
12
13 public String getrol(String user, String pass) {
14     return userRepo.findbyuserandpass(user, pass);
15 }
16
17 }
18
```

Controller:

Para finalizar tenemos nuestro **Controller** en donde señalamos la ruta a través de **@RequestMapping,** ruta que usara nuestra **VISTA** para conectarse al **Login.**

Lo importante acá es la forma en que recibe parámetros nuestro Controller, ya que como se puede apreciar, recibe dos parámetros en la **URI->**`get/usuario/{user}/{pass}.`

```
@RestController
@RequestMapping("api/usuario")

public class UsuarioController {
    @Autowired
    private UsuarioService userService;
    @PostMapping
    public ResponseEntity<usuario> create (@RequestBody usuario newuser){
        return null;
    }
    @GetMapping("get/usuario/{user}/{pass}")
    public String getrole(@PathVariable String user,@PathVariable String pass) {
     return userService.getrol(user, pass);

    }
}
```

Para poder explicar mejor esta idea, agregare una imagen de la VISTA y como se conecta:

```
URL url = new URL("http://localhost:8090/api/usuario/get/usuario/"+request.getParameter("user")+"/"+request.getParameter("pass"));
```

Se realiza una modificación de la URL agregándole los datos que llegan desde el formulario Login de la vista y divididos por un "/" genera la URI completa de nuestro **Controller Login** en la api, esta recibe los datos, realiza todo el proceso esperado y genera una respuesta que es recibida por la **VISTA** y permite el realizar el **LOGIN** del usuario.