esprit Se former autrement	Année Universitaire : 2021-2022 **Ateliers Services web** **UP_WEB**

Atelier REST – SWAGGER
JAX_RS Swagger

Objectifs

Après avoir créé un service web RESTFul qui fournira un exemple d'opérations CRUD. Notre objectif est de documenter ainsi que tester l'ensemble de ces services.

Pour ce faire on va utiliser l'outil SWAGGER , une implémentation qui nous permet de documenter et tester les API REST.

Nous utilisons des annotations sur les endpoints et les services. Ces informations se retrouveront dans un fichier JSON généré qui décrit notre API.

Un autre composant : Swagger-UI, qui donne une interface interactive à l'utilisateur final.

Les étapes

- Configurez Swagger pour qu'il reconnaisse les informations sur l'API RESTFull.
- Annotez les ressources afin que Swagger puisse les servir via sa liste de ressources.
- Annotez les modèles afin que Swagger puisse inclure les objets dans sa liste de ressources.

Rappel : Développement du service web RESTFul

Après avoir créer un service web de gestion des employés dans une entreprise.

Employe
-cin:String -nom:String -prenom:String
ajouterEmploye(Employe employe) modifierEmploye(Employe employe) chercherEmploye(String cin) afficherListeEmployes() supprimerEmploye(String cin)

Basé sur l'entité "**Employe**":

```java
@XmlRootElement
public class Employe {

	private String cin;
	private String nom;
	private String prenom;

	public String getCin() {
		return cin;
	}
	@XmlAttribute(name="id",required=true)
	public void setCin(String cin) {
		this.cin = cin;
	}
	public String getNom() {
		return nom;
	}
	@XmlElement(name="LastName")
	public void setNom(String nom) {
		this.nom = nom;
	}
	public String getPrenom() {
		return prenom;
	}
	@XmlElement(name="FirstName")
	public void setPrenom(String prenom) {
		this.prenom = prenom;
	}
	public Employe() {
		super();
		// TODO Auto-generated constructor stub
	}
}
```

1. Nous avons créer les ressources REST qui constituent le service web selon les descriptions suivantes :

   a) **addEmployee**

Requête `POST /employes` **Content-Type:** application/xml <employe id="123"> <FirstName>foulen</FirstName> <LastName>ben foulen</LastName> </employe>
Réponse **add successful** **Status 200OK**

   b) **updateEmployee**

Requête `PUT /employes` **Content-Type:** application/xml <employe id="123"> <FirstName>ahmed</FirstName> <LastName>ben salah</LastName> </employe>
Réponse **update successful** **Status 200OK**

   c) **searchEmployee**

Requête `GET /employes/123`
Réponse <employe id="123">

```
 <FirstName>foulen</FirstName>
 <LastName>ben foulen</LastName>
</employe>
```

Status 200OK

d) displayEmployeesList

Requête `GET /employes`

Réponse

```
<collection>
<employe id="123">
 <LastName>ben foulen</LastName>
 <FirstName>foulen</FirstName>
</employe>
<employe id="456">
 <LastName>ben foulen2</LastName>
 <FirstName>foulen2</FirstName>
</employe>
</collection>
```

Status 200OK

e) deleteEmployee

Requête `DELETE /employes/123`

Réponse

```
true
```

Status 200OK

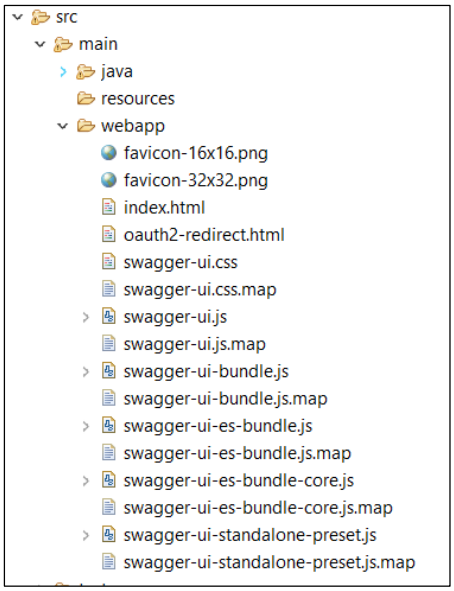
Configuration SWAGGER

Dans le fichier pom.xml, rajoutez la dependance swagger:

```
  <dependency>
    <groupId>io.swagger</groupId>
    <artifactId>swagger-jaxrs</artifactId>
    <version>1.5.22</version>
</dependency>
```

1. Télécharger Swagguer-ui qui represente une interface graphique d'interraction avec les resources d'un api : (https://github.com/swagger-api/swagger-ui)

Puis copier le contenu de swagger-ui\dist vers le projet, sous : src → main→ webapp



2. Modifier l'url dans le fichier « intex.html » du dossier « webapp » afin de se pointer vers swagger.json:

```
const ui = SwaggerUIBundle({
  url: "http://localhost:8080/GestionEmploye/api/swagger.json",
```

3. Dans la classe resources contenant les services web ajoutez les annotations swagger suivantes :

```
@Path("employes")
@Api
public class GestionEmploye {

    public static  List<Employe> employes=new ArrayList<Employe>();
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    @ApiOperation(value = "GET all Employes")
      @ApiResponses({
        @ApiResponse(code=200, message="Success")
      })
    public  Response  displayEmployeesList() {

        if(employes.size()!=0)
            return Response.status(Status.ACCEPTED).entity(employes).build();
        else
            return Response.status(Status.NOT_FOUND).entity("").build();

    }
```

L'annotation @Api permet d'active Swagger pour cette classe.

Les API RESTfull acceptent certaines données d'entrée et répondent avec des données de sortie. Les données d'entrée sont implicitement exprimées en tant que paramètres des méthodes de classe de ressources.

Dans jaxrs Swagger, les données de sortie sont exprimées à l'aide de l'élément de réponse de @ApiOperation. Si l'API contient plusieurs types de sortie, vous pouvez l'exprimer à l'aide de l'élément de réponse de @ApiResponse.

⇨ Faites les modifications necaissaires pour chaque méthode.

4. Dans la classe d'activation «RestActivator» sous le package «rest.utilities» activer la configuration swagger dans le constructeur de la classe « RestActivator »

```
@ApplicationPath("/api")
public class RestActivator extends Application {

    public RestActivator() {
          super();

          BeanConfig beanConfig = new BeanConfig();

          beanConfig.setVersion("1.0.0");
          beanConfig.setSchemes(new String[]{"http"});
          beanConfig.setHost("localhost:8080");
          beanConfig.setBasePath("GestionEmploye/api");
          beanConfig.setResourcePackage("rest.ressources");
          beanConfig.setScan(true);
       }
```

5. Pour permettre à swagger d'annoter les modèles, vous pouvez également annoter la classe de modèle comme suit :

```
@ApiModel(description = "entité enregistré dans swagger")
public class Employe {

    private int cin;
    private String nom;
    private String prenom;

    public Employe() {

    public Employe(int cin, String nom, String prenom) {

    @ApiModelProperty(value = "unique cin")
    public int getCin() {
        return cin;
    }
    public void setCin(int cin) {
        this.cin = cin;
    }
    @XmlElement(name="LastName")
    @ApiModelProperty(value = "get nom")
    public String getNom() {
        return nom;
    }
```

Déploiement et test du service web RESTFUL

1- Déployez le projet.

2. Tester le service web RESTful avec SWAGGER En tapant l'uri ***http://localhost:port-server/nom-projet/index.html***