|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Application Interoperability with Web Services | |  | |  | | **Fabien Beaujean**  **Adrien Poupa** | | **Promotion 2018** | | **23/01/2017** |  |  | | --- | |  | |

## Choix techniques

Pour la réalisation de ce projet, nous avons choisi d’utiliser les composants suivants :

* Java 8
* Glassfish 4.1.1
* Jersey 2.2

## Implémentation

### Architecture du projet

Dans le cadre de ce projet, nous avons voulu garder une architecture aussi simple que possible. Pour cela, nous avons choisi de représenter nos entités (films, notes, acteurs etc.) grâce à de simple class Java (POJO) auxquelles nous avons ajouté l’annotation « xmlRootElement » pour pouvoir ensuite les utiliser dans les web services.

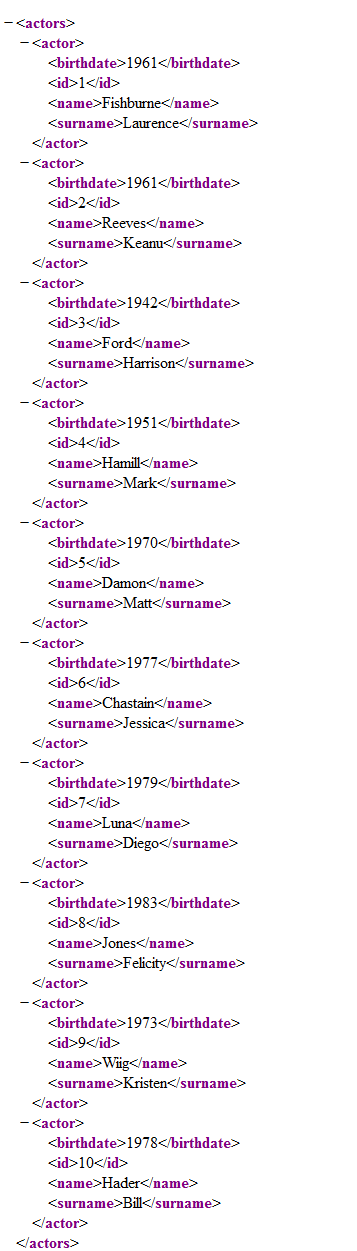
De plus, comme nous n’avons pas implémenté de persistance à l’aide d’une base de données dans ce projet, nous avons choisi de simuler cette persistance en utilisant pour chaque entité, une classe « container » singleton. Lors de la première instanciation de la classe après le démarrage du serveur, celle-ci est remplie avec les données écrites dans son constructeur. Ces données persistent jusqu’à l’arrêt du serveur. De cette façon, nous avons pu tester l’API Rest créée sans prendre du temps à gérer une base de données complète.

Enfin, pour chaque type de service différent (service pour les films, les acteurs etc.), nous avons créé une classe dédiée en y ajoutant les annotations nécessaires (« path », « produces », « consumes » etc.).

### Les services implémentés

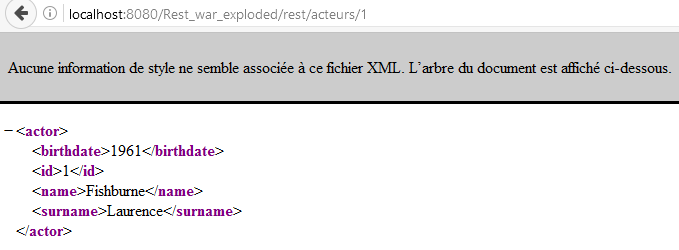
#### GET /acteurs

Ce service retourne la liste de tous les acteurs enregistrés dans l’application. Elle ne prend aucun paramètre.



#### GET /acteurs/{id}

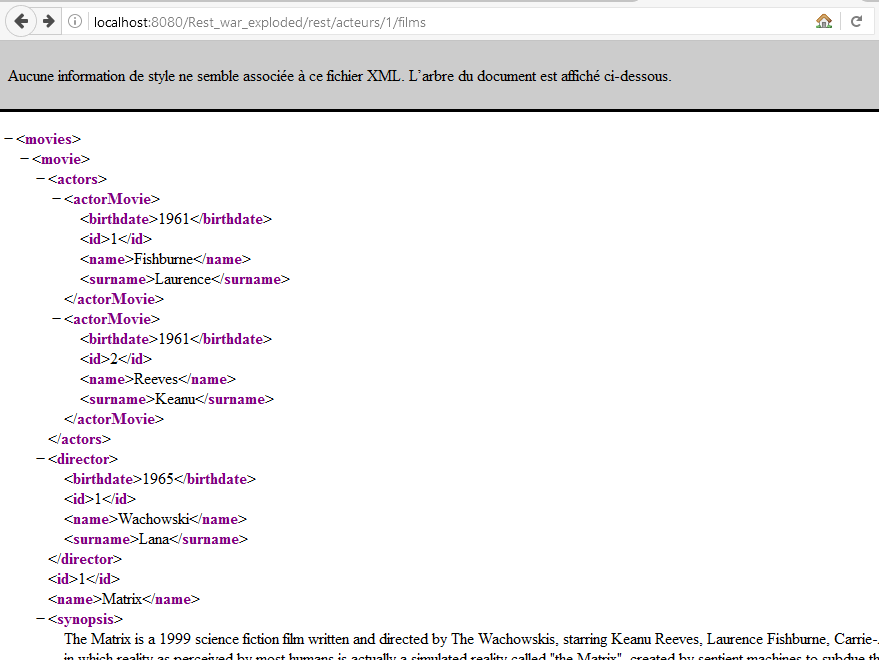
Ce service renvoie dans un document xml les informations concernant un acteurs dont le numéro est passé en paramètre



#### GET /acteurs/{id}/films

Ce service renvoie tous les films dans lesquels a tourné un acteur dont le numéro a été passé en paramètre. L’algorithme utilisé pour récupérer les films à partir d’un acteur est assez simple. On parcourt la liste de tous les films. Pour chaque film, on regarde si la liste d’acteurs contient l’id de l’acteur passé en paramètre. Si c’est le cas, on ajoute le film a une liste.

for (Movie m : MovieContainer.*getInstance*()) {  
 for (Actor a : m.getActors().getActorMovie()) {  
 if (a.getId() == actorId) {  
 moviesOutput.add(m);  
 }  
 }  
}



#### GET /commentaires/{id}

Ce service renvoie la liste des commentaires sur un film dont le numéro est passé en paramètre.

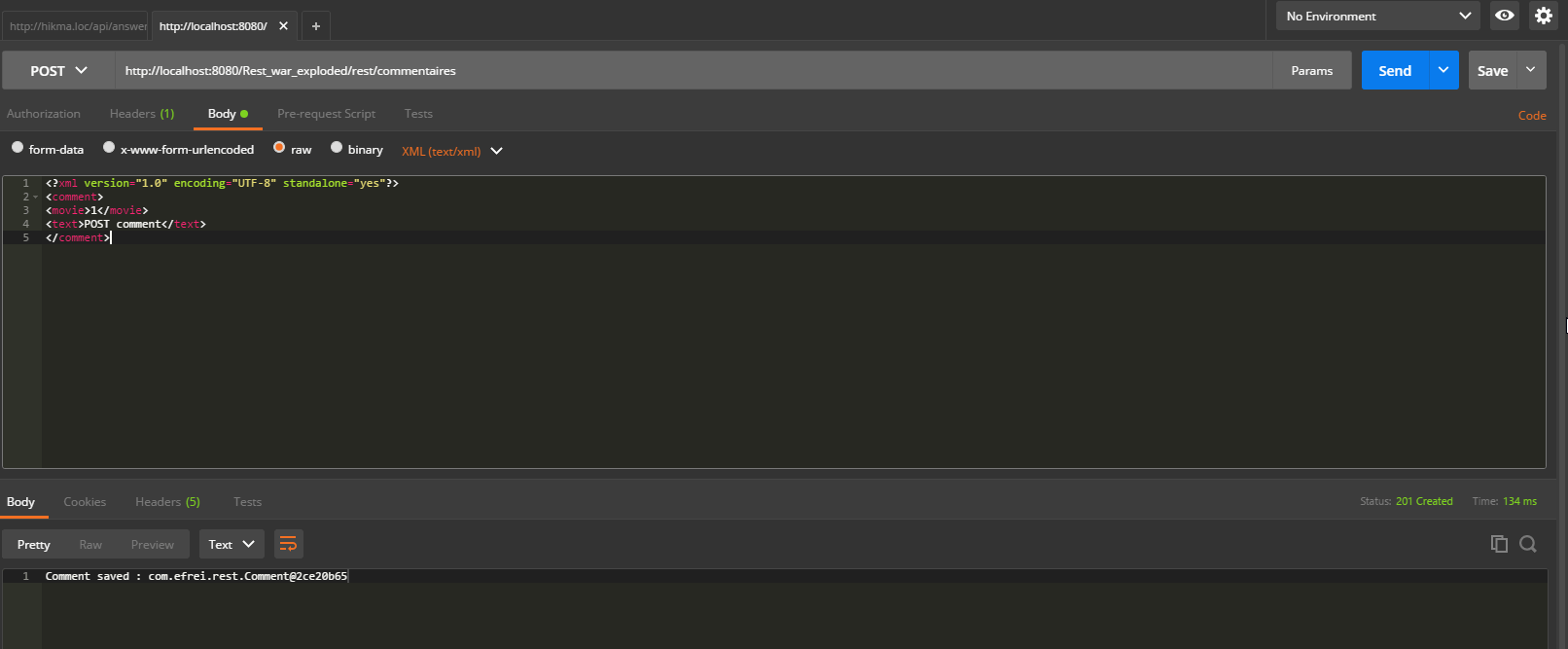
#### POST /commentaires

Ce service permet d’enregistrer un nouveau commentaire sur un film. Il est décliné en 2 versions. La première version peut être appelée en passant en paramètre un document xml contenant les informations sur le commentaire

@POST  
@Consumes(MediaType.*TEXT\_XML*)  
public Response addComment(Comment comment);

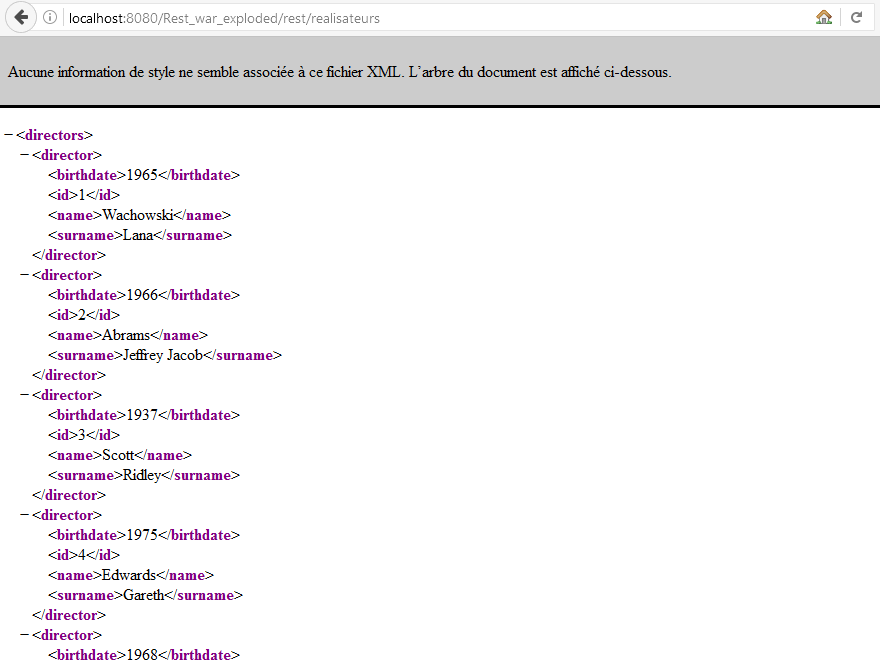
La seconde version peut être appelée via un formulaire HTML classique, en passant en paramètre, l’id du film et le contenu du commentaire :

@POST  
@Produces(MediaType.*TEXT\_HTML*)  
@Consumes(MediaType.*APPLICATION\_FORM\_URLENCODED*)  
public void addCommentForm(@FormParam("movieId") String movieId,  
 @FormParam("commentText") String comentText,  
 @Context HttpServletResponse servletResponse);



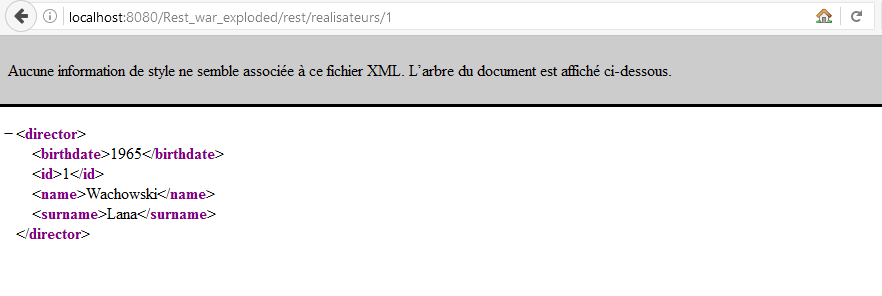
#### GET /realisateurs

Ce service renvoie la liste des réalisateurs qui sont contenus dans la base de données



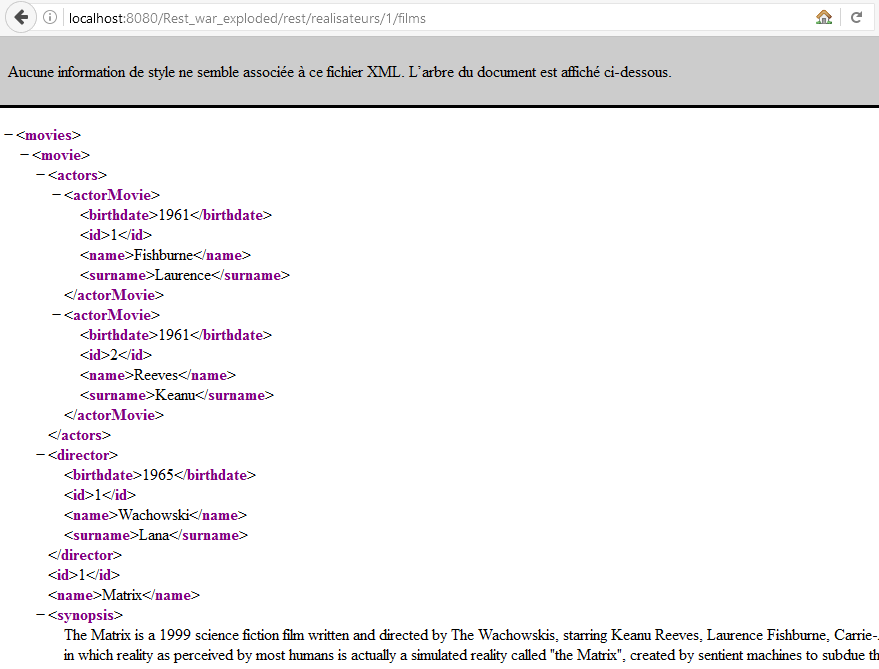
#### GET /realisateurs/{id}

Ce service renvoie les informations d’un réalisateur dont l’id a été passé en paramètre.



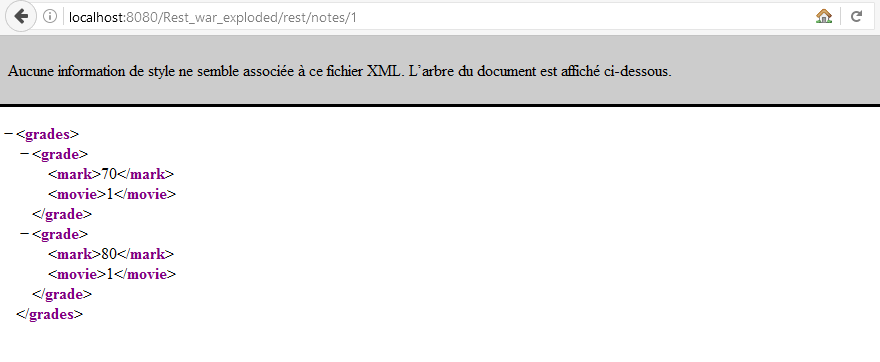
#### GET /realisateurs/{id}/films

Ce service renvoie tous les films qui ont été réalisés par le réalisateur dont l’id a été passé en paramètre



#### /GET /notes/{id}

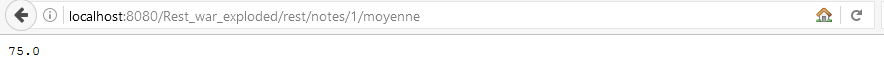
Ce service affiche toutes les notes du film dont l’id a été passé en paramètre :



#### GET /notes/{id}/moyenne

Ce service calcule et renvoie la moyenne de l’ensemble des notes du film dont l’id a été passé en paramètre. L’algorithme de calcul est très simple ; il récupère la liste des notes, sélectionne parmi elles celles du film demandé puis ensuite calcule la moyenne des valeurs de la liste récupérée avant de l’afficher en plain text :

@Path("{id}/moyenne")  
@GET  
@Produces(MediaType.*TEXT\_PLAIN*)  
public Double getGradeAverageMovie(@PathParam("id") Long movieId) {  
  
 return GradeContainer.*getInstance*()  
 .stream()  
 .filter(p-> p.getMovie() == *toIntExact*(movieId))  
 .mapToDouble(Grade::getMark)  
 .average()  
 .orElse(0);  
}



#### POST /notes

Ce service permet d’enregistrer une nouvelle note pour un film. Il est décliné en 2 versions. La première version peut être appelée en passant en paramètre un document xml contenant les informations sur la note :

@POST  
@Consumes(MediaType.*TEXT\_XML*)  
public Response addGrade(Grade grade) {  
 GradeContainer.*getInstance*();  
 GradeContainer.*addGrade*(grade);  
  
 String result = "Grade saved : " + grade;  
 return Response.*status*(201).entity(result).build();  
}

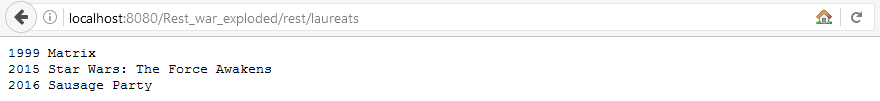
La seconde version peut être appelée via un formulaire HTML classique, en passant en paramètre, l’id du film et la note désirée :

@POST  
@Produces(MediaType.*TEXT\_HTML*)  
@Consumes(MediaType.*APPLICATION\_FORM\_URLENCODED*)  
public void addGradeForm(@FormParam("movieId") String movieId,  
 @FormParam("mark") String mark,  
 @Context HttpServletResponse servletResponse) throws IOException {  
  
 Grade grade = new Grade(Integer.*parseInt*(movieId), Integer.*parseInt*(mark));  
  
 GradeContainer.*getInstance*();  
 GradeContainer.*addGrade*(grade);  
  
 servletResponse.sendRedirect("../note.html");  
}

#### GET /laureats

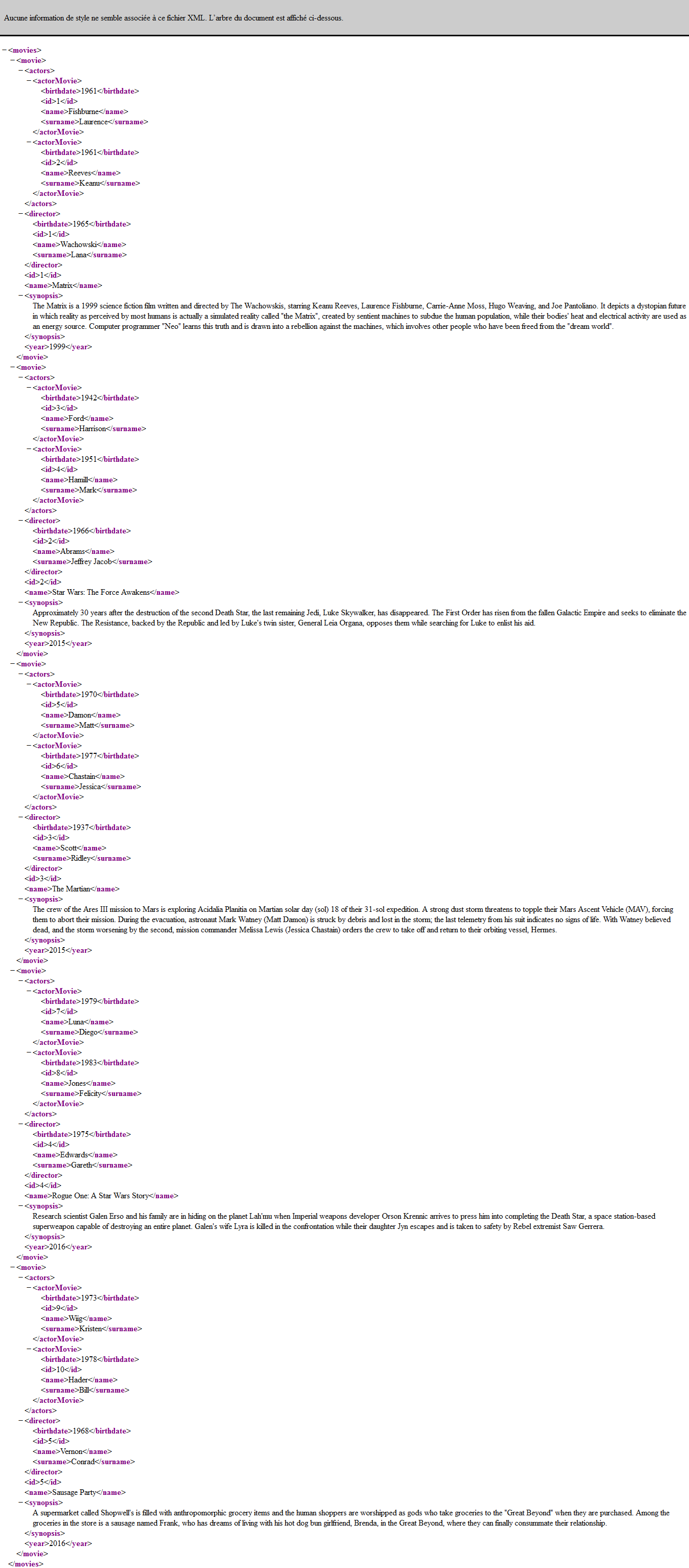
Ce service renvoie au format texte la liste de tous les films qui ont remporté un oscar. L’algorithme utilisé est assez simple. Il parcourt la liste de tous les films enregistrés et pour chaque film, il vérifié s’il a gagné. Si c’est le cas, il ajoute à un StringBuilder les informations sur le film à afficher (année et nom) :

StringBuilder list = new StringBuilder();  
for (Movie m : MovieContainer.*getInstance*()) {  
 if (m.hasWon()) {  
 list.append(m.getYear()).append(" ").append(m.getName()).append(System.*getProperty*("line.separator"));  
 }  
}  
  
return list.toString();



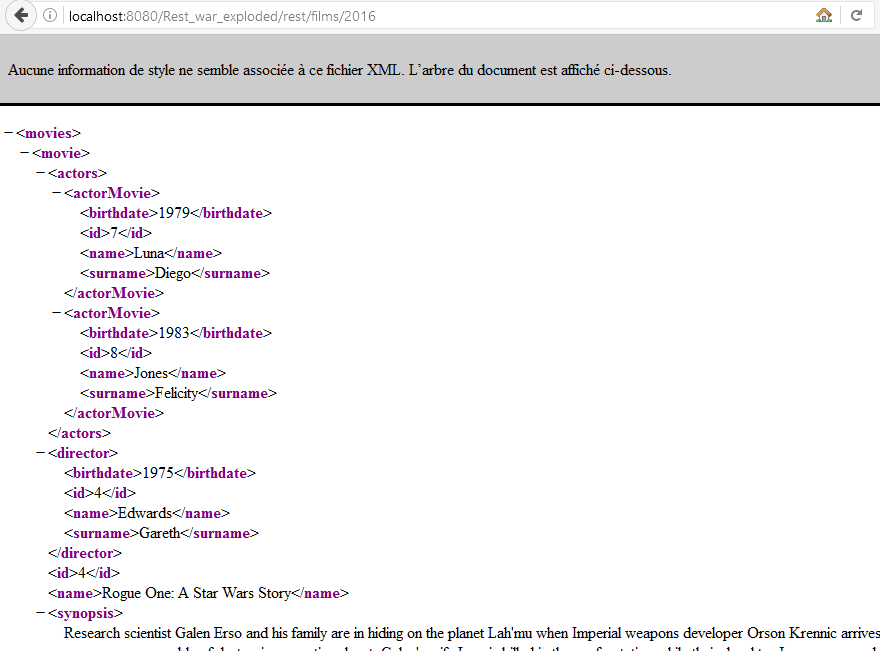
#### GET /films

Ce service retourne la liste de tous les films qui sont enregistrés sur le serveur :



#### GET /films/{year}

Ce service renvoie la liste de tous les films qui sont sortis pendant l’année passée en paramètre :



#### GET /films/{year}/laureat

Ce service renvoie la liste des films qui ont gagné un oscar pendant l’année donnée en paramètre :

