

JAX-RS et Servlets

Que connaissez-vous comme type de services web ?

SERVICE CALLING MADE EASY



RULE 1: SOAP IS MUCH MORE
POLITE THAN REST

REST

Style d'architecture web reposant sur HTTP

Utilise les méthodes HTTP

Pas de format, ni de pré-requis

Servlets

Composant Java générant du contenu dynamiquement

Spécification javax.servlet

Actuellement version 3.1

JAX-RS

La spécification JAX-RS décrit REST en Java

Il existe de multiples implémentations :

1. JBoss RESTEasy
2. Jersey

Servlets

Servlets

S'exécute côté serveur dans un conteneur de servlets (Tomcat ...)

Possède un cycle de vie : init, alive, destroy

Reçoit des requêtes HTTP et produit des réponses HTTP

Génération de pages HTML / JSP

Cycle de vie

- Création et initialisation de la servlet
- Gestion de requêtes
- Destruction et libération des ressources

Servlets

Le conteneur de servlets (tomcat, ...) gère le cycle de vie des servlets

Mécanisme de cache, asynchrone ...

Modification direct de la réponse HTTP

JAX-RS est une surcouche à la spécification servlet

Fonctionnement

- Un client fait une requête HTTP au serveur
- La requête est transmise au conteneur par le serveur
- Le conteneur décide à quelle Servlet transférer la requête (comment ?)

Application Web

Une application web est un espace virtuel

Le fichier web.xml décrit toute la configuration de l'espace virtuel

Doit être placé dans le répertoire WEB-INF

web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">

  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>examples.Hello</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

Conclusion

Si vous voulez en savoir plus :

http://jlguillaume.free.fr/www/documents/teaching/ntw1314/LI385_C1_servlets.pdf

Très verbeux

JBoss RESTEasy

Introduction

Implémentation complète de JAX-RS

Embarque un client

Gère l'asynchrone, compression, cache ...

JAX-RS vs Servlet

Pourquoi JAX-RS plutôt que Servlet ?

JAX-RS vs Servlet

Pourquoi JAX-RS plutôt que Servlet ?

- Tout est annotation
- Fait pour répondre à ce type de besoin
- Très simple d'utilisation

Hello World

```
@Path("/api")
@Produces("text/plain")
public class Endpoint {

    @GET
    public Response hello() {
        return Response.ok("Hello World").build();
    }
}
```

@Path

Peut se mettre sur une classe / méthode

Définit le chemin d'accès à la ressource

Exemple : `@Path("/users/fetch")`

Paramètres

Plusieurs moyens de passer des paramètres :

- @PathParam
- @QueryParam
- @FormParam
- @HeaderParam
- @CookieParam

@PathParam

Permet de récupérer un paramètre défini dans l'URL

```
@Path("/mobile/play/{modelId}")  
public Response play(@PathParam("modelId") String modelId) {  
    return Response.ok(modelId).build();  
}
```

La requête sera /mobile/play/test

@QueryParam

Permet de récupérer un paramètre de l'URL

```
@Path("/mobile/play/{modelId}")  
public Response play(@QueryParam("modelId") String modelId) {  
    return Response.ok(modelId).build();  
}
```

La requête sera /mobile/play?modelId=test

@DefaultValue

Permet de donner une valeur par défaut si le paramètre n'est pas spécifié.

```
@Path("/mobile/play")
public Response play(@QueryParam("modelId") @DefaultValue("test")
String modelId) {
    return Response.ok(modelId).build();
}
```

Que renvoie la requête /mobile/play ?

@Produces / @Consumes

Spécifie les types mimes acceptés et renvoyés par le service web

```
@Path("/api/model")  
@Produces(MediaType.APPLICATION_JSON)  
@Consumes(MediaType.APPLICATION_JSON)  
public class ModelController {  
}
```

Propre à la classe ou à une méthode

@GZIP

Permet de GZIP le body et/ou la réponse d'une requête

```
@Path("/api")
@Consumes("application/json")
public class MyProxy {

    @PUT
    @GZIP
    public Response put(@GZIP String content) {

    }
}
```

@Cache

Uniquement sur les requêtes GET

Équivalent du Cache-Control HTTP

Ne fonctionne que sur un code 200

Déploiement

Avoir un web.xml bien configuré

Préparer une classe Application

Générer un fichier war

Application

Classe Java décrivant les ressources utilisées

Singletons : Ne sont instanciés qu'une fois

Classes : Différents scopes

Application

```
@ApplicationPath("")
public class App extends Application {

    @Override
    public Set<Object> getSingletons() {
        Set<Object> sets = new HashSet<>(1);
        sets.add(new TestEndpoint()); // Add an endpoint
        return sets;
    }

    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> sets = new HashSet<>(1);
        sets.add(GsonProvider.class); // add a filter
        return sets;
    }
}
```

JSON

Gestion automatique si les bonnes bibliothèques sont spécifiées / configurées

Les bibliothèques principales : jackson, gson

Ne faites pas cela vous même !

Regardez l'exemple GsonProvider dans project/dant/

JSON

@POST

@Path("/account")

```
public Account updateAccount(Account account) {  
    System.out.println("Received account " + account);  
    account.setUpdated(System.currentTimeMillis());  
    return account;  
}
```


Gestion des erreurs : ExceptionMapper

Dès lors qu'une exception est levée, celle-ci sera catchée par le serveur

```
public class RuntimeExceptionMapper implements ExceptionMapper<RuntimeException> {  
  
    public Response toResponse(RuntimeException e) {  
        return Response.status(400).entity(e.getMessage()).build();  
    }  
}
```

Définit la façon dont on gère les exceptions. A spécifier dans Application.

Conclusion

JAX-RS est très simple d'utilisation

Très utilisé en entreprise

Vous devrez l'utiliser pour vos projets