

Concepts avancés de Jakarta RESTful Web Services

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1 Les liens sémantiques	1
1.1 Etendre JAX-RS avec les Providers ..	1
1.2 Les filtres : application au statut et à la pagination	2
1.3 Schéma d'une API REST	2

Les exemples suivants sont accessibles dans le dépôt :

☒ : [ebpro/sample-jakarta-rest-full](https://github.com/ebpro/sample-jakarta-rest-full)

Détails

- **URL:** <https://github.com/ebpro/sample-jakarta-rest-full>
- **Branch:** develop
- **Local Path:** /home/jovyan/work/github/ebpro/sample-jakarta-rest-full
- **Latest Commit:** 7331765 - chore: add _quarto-utils submodule + .devcontainer (SOTA pattern)
- **Timestamp:** 2026-10-14T17:53:58Z

Clone Command:

```
git clone -b develop https://github.com/ebpro/sample-jakarta-rest-full
```

1 Les liens sémantiques

Les liens sémantiques entre les ressources peuvent être représentés à l'aide d'URLs de façon standard ((RFC8288)[<https://tools.ietf.org/html/rfc8288>]). Une liste complète est donnée sur le site de l'IANA : <https://www.iana.org/assignments/link-relations/link-relations.xhtml>. Les plus classiquement utilisés avec REST sont self, first, previous, next et last. Pour simplifier l'usage de l'API ces liens peuvent être fournis dans l'entête d'une réponse http et/ou dans le corps de la réponse (par exemple lors de la pagination).

```
Link: <http://MyServer:8080/MyApp/Persons/1>; rel="first-person",  
<http://MyServer:8080/MyApp/Persons/10>; rel="next-person", <http://MyServer:8080/MyApp/Persons/8>; rel="previous-person", <http://MyServer:8080/MyApp/Persons/90>; rel="last-person"
```

1.1 Etendre JAX-RS avec les Providers

D'une façon générale une classe annotée par `@Provider` ajoute des fonctions à JAXRS (traitement des exceptions, conversion des données, ...).

1.1.0.a Les mappers : application aux exceptions

Le traitement des exceptions peut être simplifié en utilisant des mappers (cf. package exceptions et mapper) qui s'appliquent automatiquement lors qu'une exception est émise.

Dans ce cas un objet Response est construit manuellement pour contrôler le détail du corps et de l'entête.

```
Error: failed to read file /home/
jovyan/work/materials/github/ebpro/
sample-jakarta-rest-full/src/main/
java/fr/univtl/bruno/samples/jaxrs/
exceptions/NotFoundException.java: /
home/jovyan/work/materials/github/eb-
pro/sample-jakarta-rest-full/src/main/java/
fr/univtl/bruno/samples/jaxrs/exceptions/
NotFoundException.java
```

```
Error: failed to read file /home/
jovyan/work/materials/github/ebpro/
sample-jakarta-rest-full/src/main/
java/fr/univtl/bruno/samples/jaxrs/
mappers/BusinessExceptionMapper.java: /
home/jovyan/work/materials/github/eb-
pro/sample-jakarta-rest-full/src/main/java/
fr/univtl/bruno/samples/jaxrs/mappers/
BusinessExceptionMapper.java
```

If a resource doesn't exist an exception is raised, and the 404 http status code is returned

1.2 Les filtres : application au statut et à la pagination

```
Error: failed to read file /home/
jovyan/work/materials/github/ebpro/
sample-jakarta-rest-full/src/main/java/
fr/univtl/bruno/samples/jaxrs/status/
StatusFilter.java: /home/jovyan/work/ma-
terials/github/ebpro/sample-jakarta-rest-full/
src/main/java/fr/univtl/bruno/samples/jaxrs/
status/StatusFilter.java
```

Un exemple de filtre appliqué à la requête sera donné dans la partie sur la sécurité.

1.3 Schéma d'une API REST

- WSDL un autre langage de service Web permet de décrire un tel schéma.
- Le langage WADL est spécifique à REST.

- Par défaut Jersey génère une description WADL `/application.wadl`
- Cependant, pour cela l'application nécessite une implantation du standard pour traiter les documents XML appelé JAXB qui n'est plus fournie en standard dans le JDK depuis la version 8 (nous utilisons ici la version 15).

Si JAXB est dans le classpath (cf. pom.xml) la génération d'une description WADL est automatique.

<https://jakarta.ee/specifications/restful-ws/3.0/jakarta-restful-ws-spec-3.0.html#context>