


Jakarta EE Web Total


WebServices REST em Jakarta EE usando JAX-RS





Tópicos Abordados




- RESTful WebServices
- Operações do HTTP
- Status do HTTP
- MIME Types
- WebServices com a API JAX-RS
- @Produces e @Consumes
- Path Parameters
- Query Parameters
- Cliente para RESTful WebServices



RESTful WebServices



- REST é um estilo arquitetural
 - *Representational State Transfer*
- Um WebService é visto como um recurso (*resource*)
- É identificado através de uma URI
 - *Uniform Resource Identifier*
- WebServices que usam o estilo REST são conhecidos como *RESTful WebServices*



Exemplo de RESTful WebService

URI

<http://www.softblue.com.br/services/users>

JSON

```
[{
  id: 1,
  name: "José",
  email: "jose@email.com"
},
{
  id: 2,
  name: "Paulo",
  email: "paulo@email.com"
}]
```

JSON e XML são muito usados

Operações


- O protocolo HTTP suporta várias operações – GET, POST, PUT, DELETE, etc.
- RESTful WebServices tiram vantagens dessas operações

Operação	Significado
GET	Ler um resource
POST	Criar um resource
PUT	Atualizar um resource
DELETE	Excluir um resource

Exemplos de Operações


/services/user/17	GET	Retorna os dados do usuário com ID 17
/services/user/17	DELETE	Exclui o usuário com ID 17
/services/user/17	PUT	Atualiza os dados do usuário com ID 17 (os dados são enviados no corpo do HTTP)
/services/users	GET	Retorna os dados de todos os usuários
/services/users	POST	Cadastra um novo usuário (os dados são enviados no corpo do HTTP)

Status do HTTP




- Toda requisição HTTP a um servidor resulta em uma resposta (*status*)


Status	Descrição
200	OK
201	Created
400	Bad Request
403	Forbidden
404	Not Found
500	Internal Server Error




MIME Types



- O MIME Type é o tipo de dado que está sendo transportado pelo protocolo HTTP
- É a forma do cliente e do servidor se “entenderem”
 - O cliente envia no header da requisição HTTP qual o MIME Type dos dados que ele está enviando ao servidor
 - O servidor envia no header da resposta do HTTP o MIME Type referente aos dados que ele está enviando ao cliente




MIME Types




- Os MIME Types são padronizados
- Alguns dos MIME Types mais comuns usados na troca de dados em WebServices:


MIME TYPE	Descrição
application/json	JSON
application/xml	XML
text/plain	Texto puro
text/html	HTML




A API JAX-RS



- O JAX-RS é uma API que pertence ao Java EE
- Permite criar RESTful WebServices de forma fácil, rápida e intuitiva




Criando Web Services com JAX-RS



- É preciso definir o caminho raiz onde os WebServices da aplicação vão responder

```
@ApplicationPath("/services")
public class JAXRSConfig extends Application {
}
```

Define a URI base */services/**



Criando Web Services com JAX-RS



- Uma classe Java é usada para definir o Webservice

```
@Path("/users")
public class UserService {
    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<User> list() {
        //...
    }
}
```

Define a URI do serviço como */users*

Método chamado na operação GET

Retorna os dados no formato JSON

A URI é: */services/users*



4

@Produces & @Consumes



- Ao criar um Webservice, é preciso informar que tipo de informação ele consome e que tipo de informação ele produz

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public List<User> list() {
    //...
}
```

Envia JSON
para o cliente

```
@POST
@Consumes(MediaType.TEXT_PLAIN)
public void create(String txt) {
    //...
}
```

Recebe TEXTO
do cliente



Path Parameters



- É possível fornecer parâmetros para o Webservice diretamente na URI

```
@Path("/users")
public class UserService {

    @GET
    @Path("/{userId}")
    @Produces(MediaType.APPLICATION_JSON)
    public User find(@PathParam("userId") int id) {
        //...
    }
}
```

A URI é: `/services/users/20`
(o 20 é passado como parâmetro para o método `find()`)



Query Parameters



- É possível fornecer parâmetros para o Webservice via *query string*

```
@Path("/users")
public class UserService {

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public User find(@QueryParam("userId") int id) {
        //...
    }
}
```

A URI é: `/services/users?userId=20`
(o 20 é passado como parâmetro para o método `find()`)



Cliente para RESTFul Web Services




- O JAX-RS permite fazer requisições a WebServices existentes a partir da interface Client

```
User user = ClientBuilder.newClient()
    .target(URI)
    .path("15")
    .request(MediaType.APPLICATION_JSON)
    .get(User.class);
```



Cliente para RESTFul Web Services



- Para as operações de POST, PUT e DELETE é comum o retorno do tipo Response

```
Reponse response = ClientBuilder.newClient()
    .target(URI)
    .path("15")
    .request()
    .delete();
```

O objeto *Response* permite recuperar o status HTTP da resposta e outras informações relevantes

