University of Innsbruck
Bernhard Fritz 1316136
January 3rd, 2016

Web Services PS 703128

REST Web Services

Required software:

- ✓ Java JDK: http://www.oracle.com/technetwork/java/javase/downloads/
- ✓ Eclipse: http://www.eclipse.org/downloads/
- ✓ Eclipse WTP: http://www.eclipse.org/webtools/
- ✓ CXF Framework: http://cxf.apache.org/

Exercise 1 – REST Web Services design

Consider the Remote Printer interface shown below. Redesign the interface applying REST principles. Define the URI hierarchy that you consider appropriate to cover the operations of the interface.

```
public interface RemotePrinter {
        /**
         * Submit a new print job.
         *
        * @param text the text to be printed
        * @return the id of the print job.
         */
        public int submitJob(String text);

        /**
         * Check the job's completion status.
        *
        * @param id the job id
        * @return <code>true</code> if the job is complete and
        * <code>false</code> otherwise.
        *
        * @throws RemoteException if a remote exception occurs
        */
        public boolean isComplete(int id);

        /**
        * Queries the printer status.
        * @return printer status in a human readable form.
        * @throws RemoteException if a remote exception occurs
        */
        public String getPrinterStatus();
}
```

XML Requests/Responses:

POST /printer/jobs HTTP/1.1

Host: http://localhost:9000

Content-Type: application/xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobTemplate>
  <text>foobar</text>
</jobTemplate>
```

Response:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<job>
  <completed>false</completed>
  <id>0</id>
  <text>foobar</text>
</job>
```

GET /printer/jobs/0 HTTP/1.1

Host: http://localhost:9000

Response:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<job>
  <completed>false</completed>
  <id>0</id>
  <text>foobar</text>
</job>
```

Send another request after 10 seconds have passed and you will recieve following response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<job>
  <completed>true</completed>
  <id>0</id>
  <text>foobar</text>
</job>
```

GET /printer HTTP/1.1

Host: http://localhost:9000

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobs>
  <status>There are 1 jobs in the queue, 0 of which are completed!</status>
</jobs>
```

Send another request after 10 seconds have passed and you will recieve following response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobs>
  <status>There are 1 jobs in the queue, 1 of which are completed!</status>
</jobs>
```

JSON Requests/Responses:

POST /printer/jobs HTTP/1.1

Host: http://localhost:9000

Content-Type: application/json

Accept: application/json

```
{
  "text": "foobar"
}
```

Response:

```
{
 "id": 0,
 "text": "foobar",
 "completed": false
}
```

GET /printer/jobs/0 HTTP/1.1

Host: http://localhost:9000

Accept: application/json

Response:

```
{
 "id": 0,
 "text": "foobar",
 "completed": false
}
```

Send another request after 10 seconds have passed and you will recieve following response:

```
{
 "id": 0,
 "text": "foobar",
 "completed": true
}
```

GET /printer HTTP/1.1

Host: http://localhost:9000

Accept: application/json

Response:

```
{
 "status": "There are 1 jobs in the queue, 0 of which are completed!"
}
```

Send another request after 10 seconds have passed and you will recieve following response:

```
{
 "status": "There are 1 jobs in the queue, 1 of which are completed!"
}
```

Exercise 2 – REST Web Services implementation CXF

Implement the Printer REST Web Service based on JAX-RS and the CXF framework.

extract RESTfulWebService.zip
open terminal in directory RESTfulWebService and type: ./gradlew run

Exercise 3 – Common Mistakes in REST

Justify in your design/implementation of the RemotePrinter interface how you avoided the following common REST mistakes:

- Tunneling everything through GET
  I used GET as well as POST requests as seen in PrinterService.java.

- Ignoring MIME types
  My server supports XML as well as JSON requests/responses as seen in PrinterService.java.
  The client has to specify a “Content-Type” header to specify the request body format.
  If the client specifies an “Accept” header the server will respond accordingly.

How would you add support for JSON response types? Modify your implementation to support both JSON and XML as data encoding.

To support JSON response types the JAX-RS annotations have to be modified from
@Produces({“application/xml”})
to
@Produces({“application/xml”, “application/json”})
as seen in PrinterService.java.
In addition to that a JSON provider has to be added to JAXRSServerFactoryBean. I decided to use the JacksonJsonProvider as seen in Server.java.