Chapter 1. Use JAX-RS 1.1 in JOnAS

Table of Contents

1.1. Introduction

JAX-RS stands for *J*ava *A*PI for *X*ML *R*ESTful *S*ervices (JAX-RS). It's a Java EE 6 specification that introduces the REST [http://en.wikipedia.org/wiki/Representational_State_Transfer] support in the Java EE platform.

Its goal is to provide an easy to use REST development model based on annotations for the Java platform. It's a *server-side only* specification (no client API is described).

The developer is responsible to declare REST *resources* that will act as entry point to its application:

```
// The Java class will be hosted at the URI path "/helloworld"
@Path("/helloworld")
public class HelloWorldResource {

    // The Java method will process HTTP GET requests
    @GET
    // The Java method will produce content identified by the MIME Media type "text/plain"
    @Produces("text/plain")
    public String helloWorld() {
        // Return some cliched textual content
        return "Hello World";
    }
}
```

1.2. JAX-RS Applications

A JAX-RS application is an application model abstraction that allows the developer to specify the list of resources and singleton classes that will be exposed by the REST container.

```
// Identifies the application path that serves as the base URI for all resource URIs
 provided by Path
@ApplicationPath("/application")
public class MyApplicaton extends Application {

    // Returns the list of resources that will be managed by the container
    public Set<Class<?>> getClasses() {
        Set<Class<?>> s = new HashSet<Class<?>>();
        s.add(HelloWorldResource.class);
        return s;
    }
}
```

1.3. JAX-RS Resources

Resources are the main concept manipulated all along JAX-RS.

1.3.1. Using Resources

Root resource classes are POJOs (Plain Old Java Objects) that are annotated with `@Path`, have at least one method annotated with `@Path` or a resource method designator annotation such as `@GET`, `@PUT`, `@POST`, or `@DELETE`. Resource methods are methods of a resource class annotated with a resource method designator.

`HelloWorldResource` is a root resource:

```
@Path("/helloworld")
public class HelloWorldResource {

    // Resource method
    @GET
    public String getClichedMessage() {
        return "Hello World";
    }
}
```

1.3.1.1. Sub Resources

Root resource methods can return types that are also resources (annotated with `@Path`), that allows application decomposition.

`HelloResource`:

```
@Path("/hello")
public class HelloResource {

    // Resource method
    @GET
    public WorldResource getClichedMessage() {
        return new WorldResource();
    }
}
```

And it's sub-resource `WorldResource`:

```
@Path("/world")
public class WorldResource {

    @GET
    public String getClichedMessage() {
        return "Hello World";
    }
}
```

1.3.2. Using Parameters

1.3.2.1. @PathParam

`@Path` declares an *URI path template*: it can contains variable definition. This variable can then be accessed as a resource method parameter:

```
@GET
@Path("/hello/{username}")
public String hello(@PathParam("username") String username) {
    return "Hello " + username;
}
```

A callable URI:

```
/hello/Guillaume
```

1.3.2.2. @QueryParam

The HTTP query parameters (ex: `?param=value`) are also accessible as mapped method parameter:

```
@GET
public String hello(@QueryParam("username") String username) {
    return "Hello " + username;
}
```

A callable URI:

```
/hello?username=Guillaume
```

1.3.3. Using Mime-Types

Resources can be configured to accept and produces specific input (respectively output) types.

1.3.4. @Consumes

The `@Consumes` annotation declares the expected MIME type that will be accepted by a resource.

Only HTTP requests with the right `Accept` header will be dispatched to the resource method:

```
@POST
@Consumes("text/plain")
public void postClichedMessage(String message) {
    // ...
}
```

1.3.5. @Produces

The `@Produces` annotation declares the MIME type that will be returned by this resource method:

```
@GET
@Produces({"application/xml", "application/json"})
public String doGetAsXmlOrJson() {
    ...
}
```

1.4. Activation in JOnAS

In order to activate JAX-RS on JOnAS, the 'jaxrs' service must be declared in the `$JONAS_BASE/conf/jonas.properties` in the `jonas.services` property.

```
jonas.services    registry,jmx,... ,jaxrs
```

JOnAS is using Jersey (the reference implementation) as default JAX-RS provider.

Warning

The JAX-RS support in JOnAS is currently limited to Web Applications

Note

JAX-RS Supports needs a Servlet 3.0 container: JOnAS '`web`' service in use must be Tomcat7 (at time of writing):

```
#
```

```
###################### JOnAS Web container service configuration
#
#  Set the name of the implementation class of the web container service.
jonas.service.web.class    org.ow2.jonas.web.tomcat7.Tomcat7Service
```

1.5. Resources

- JAX-RS Specification [http://jcp.org/en/jsr/detail?id=311]
- Representational State Transfer (REST [http://en.wikipedia.org/wiki/Representational_State_Transfer])
- Oracle Jersey [http://jersey.java.net] (Reference Implementation) + Documentation [http://jersey.java.net/nonav/documentation/latest/index.html]
- Apache CXF [http://cxf.apache.org] (ASL2 Licensed Implementation) + Documentation [http://cxf.apache.org/docs/jax-rs.html]