Simple REST

Example simple-rest can be browsed at https://github.com/apache/tomee/tree/master/examples/simple-rest

Defining a REST service is pretty easy, simply ad @Path annotation to a class then define on methods the HTTP method to use (@GET, @POST, ...).

The Code

The REST service: @Path, @GET, @POST

Here we see a bean that uses the Bean-Managed Concurrency option as well as the @Startup annotation which causes the bean to be instantiated by the container when the application starts. Singleton beans with @ConcurrencyManagement(BEAN) are responsible for their own thread-safety. The bean shown is a simple properties "registry" and provides a place where options could be set and retrieved by all beans in the application.

```
package org.superbiz.rest;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;

@Path("/greeting")
public class GreetingService {
    @GET
    public String message() {
        return "Hi REST!";
    }

    @POST
    public String lowerCase(final String message) {
        return "Hi REST!".toLowerCase();
    }
}
```

Testing

Test for the JAXRS service

The test uses the OpenEJB ApplicationComposer to make it trivial.

The idea is first to activate the jaxrs services. This is done using @EnableServices annotation.

Then we create on the fly the application simply returning an object representing the web.xml. Here we simply use it to define the context root but you can use it to define your REST Application too. And to complete the application definition we add @Classes annotation to define the set of classes to use in this app.

Finally to test it we use cxf client API to call the REST service in get() and post() methods.

```
package org.superbiz.rest;

import org.apache.cxf.jaxrs.client.WebClient;
import org.apache.openejb.jee.SingletonBean;
import org.apache.openejb.junit.ApplicationComposer;
import org.apache.openejb.junit.EnableServices;
import org.apache.openejb.junit.Module;
import org.junit.Test;
import org.junit.runner.RunWith;

import java.io.IOException;

import static org.junit.Assert.assertEquals;

@EnableServices(value = "jaxrs")
@RunWith(ApplicationComposer.class)
public class GreetingServiceTest {
    @Module
    public SingletonBean app() {
        return (SingletonBean) new SingletonBean(GreetingService.class).localBean();
    }

    @Test
    public void get() throws IOException {
        final String message = WebClient.create("http://localhost:4204").path(
"/GreetingServiceTest/greeting/").get(String.class);
        assertEquals("Hi REST!", message);
    }

    @Test
    public void post() throws IOException {
        final String message = WebClient.create("http://localhost:4204").path(
"/GreetingServiceTest/greeting/").post("Hi REST!", String.class);
        assertEquals("hi rest!", message);
    }
}
```

Running

Running the example is fairly simple. In the "simple-rest" directory run:

```
$ mvn clean install
```

Which should create output like the following.

```
INFO - Cannot find the configuration file [conf/openejb.xml].  Will attempt to create
one for the beans deployed.
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
provider-id=Default Security Service)
INFO - Configuring Service(id=Default Transaction Manager, type=TransactionManager,
provider-id=Default Transaction Manager)
INFO - Creating TransactionManager(id=Default Transaction Manager)
INFO - Creating SecurityService(id=Default Security Service)
INFO - Initializing network services
INFO - Creating ServerService(id=httpejbd)
INFO - Creating ServerService(id=cxf-rs)
INFO - Initializing network services
INFO - Starting service httpejbd
INFO - Started service httpejbd
INFO - Starting service cxf-rs
INFO - Started service cxf-rs
INFO -   ** Bound Services **
INFO -   NAME                 IP              PORT
INFO -   httpejbd             127.0.0.1       4204
INFO - -------
INFO - Ready!
INFO - Configuring enterprise application: /opt/dev/openejb/openejb-
trunk/examples/GreetingServiceTest
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.rest.GreetingServiceTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Creating Container(id=Default Managed Container)
INFO - Using directory /tmp for stateful session passivation
INFO - Enterprise application "/opt/dev/openejb/openejb-
trunk/examples/GreetingServiceTest" loaded.
INFO - Assembling app: /opt/dev/openejb/openejb-trunk/examples/GreetingServiceTest
INFO - Existing thread singleton service in SystemInstance() null
INFO - Created new singletonService
org.apache.openejb.cdi.ThreadSingletonServiceImpl@12c9b196
INFO - Succeeded in installing singleton service
INFO - OpenWebBeans Container is starting...
INFO - Adding OpenWebBeansPlugin : [CdiPlugin]
INFO - All injection points are validated successfully.
INFO - OpenWebBeans Container has started, it took 11 ms.
INFO - Deployed Application(path=/opt/dev/openejb/openejb-
trunk/examples/GreetingServiceTest)
INFO - Setting the server's publish address to be http://127.0.0.1:4204/test
INFO - REST Service: http://127.0.0.1:4204/test/greeting/.*  -> Pojo
org.superbiz.rest.GreetingService
INFO - Undeploying app: /opt/dev/openejb/openejb-trunk/examples/GreetingServiceTest
INFO - Stopping network services
INFO - Stopping server services
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 sec
```

```
Results :
```

```
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```