REST on EJB

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

Help us document this example! Click the blue pencil icon in the upper right to edit this page.

User

```
package org.superbiz.rest;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@NamedQueries({
        @NamedQuery(name = "user.list", query = "select u from User u")
}
```

UserService

```
package org.superbiz.rest;

import javax.ejb.Lock;
import javax.ejb.LockType;
import javax.ejb.Singleton;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.ws.rs.DELETE;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import java.util.ArrayList;
import java.util.List;

/**
```

```
 * Outputs are copied because of the enhancement of OpenJPA.
 *
 */
@Singleton
@Lock(LockType.WRITE)
@Path("/user")
@Produces(MediaType.APPLICATION_XML)
public class UserService {
    @PersistenceContext
    private EntityManager em;

    @Path("/create")
    @PUT
    public User create(@QueryParam("name") String name,
                       @QueryParam("pwd") String pwd,
                       @QueryParam("mail") String mail) {
        User user = new User();
        user.setFullname(name);
        user.setPassword(pwd);
        user.setEmail(mail);
        em.persist(user);
        return user;
    }

    @Path("/list")
    @GET
    public List<User> list(@QueryParam("first") @DefaultValue("0") int first,
                           @QueryParam("max") @DefaultValue("20") int max) {
        List<User> users = new ArrayList<User>();
        List<User> found = em.createNamedQuery("user.list", User.class).
setFirstResult(first).setMaxResults(max).getResultList();
        for (User u : found) {
            users.add(u.copy());
        }
        return users;
    }

    @Path("/show/{id}")
    @GET
    public User find(@PathParam("id") long id) {
        User user = em.find(User.class, id);
        if (user == null) {
            return null;
        }
        return user.copy();
    }

    @Path("/delete/{id}")
    @DELETE
    public void delete(@PathParam("id") long id) {
        User user = em.find(User.class, id);
```

```
        if (user != null) {
            em.remove(user);
        }
    }

    @Path("/update/{id}")
    @POST
    public Response update(@PathParam("id") long id,
                           @QueryParam("name") String name,
                           @QueryParam("pwd") String pwd,
                           @QueryParam("mail") String mail) {
        User user = em.find(User.class, id);
        if (user == null) {
            throw new IllegalArgumentException("user id " + id + " not found");
        }

        user.setFullname(name);
        user.setPassword(pwd);
        user.setEmail(mail);
        em.merge(user);

        return Response.ok(user.copy()).build();
    }
}
```

persistence.xml

```
<persistence version="2.0"
             xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
                      http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
  <persistence-unit name="user">
    <jta-data-source>My DataSource</jta-data-source>
    <non-jta-data-source>My Unmanaged DataSource</non-jta-data-source>
    <class>org.superbiz.rest.User</class>
    <properties>
      <property name="openjpa.jdbc.SynchronizeMappings" value=
"buildSchema(ForeignKeys=true)"/>
    </properties>
  </persistence-unit>
</persistence>
```

UserServiceTest

```
package org.superbiz.rest;
```

```
import org.apache.cxf.jaxrs.client.WebClient;
import org.apache.openejb.OpenEjbContainer;
import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;
import javax.naming.NamingException;
import javax.ws.rs.core.Response;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertNull;
import static junit.framework.Assert.fail;

public class UserServiceTest {
    private static Context context;
    private static UserService service;
    private static List<User> users = new ArrayList<User>();

    @BeforeClass
    public static void start() throws NamingException {
        Properties properties = new Properties();
        properties.setProperty(OpenEjbContainer.OPENEJB_EMBEDDED_REMOTABLE, "true");
        context = EJBContainer.createEJBContainer(properties).getContext();

        // create some records
        service = (UserService) context.lookup("java:global/rest-on-ejb/UserService");
        users.add(service.create("foo", "foopwd", "foo@foo.com"));
        users.add(service.create("bar", "barpwd", "bar@bar.com"));
    }

    @AfterClass
    public static void close() throws NamingException {
        if (context != null) {
            context.close();
        }
    }

    @Test
    public void create() {
        int expected = service.list(0, 100).size() + 1;
        Response response = WebClient.create("http://localhost:4204")
                .path("/user/create")
```

```
                .query("name", "dummy")
                .query("pwd", "unbreakable")
                .query("mail", "foo@bar.fr")
                .put(null);
        List<User> list = service.list(0, 100);
        for (User u : list) {
            if (!users.contains(u)) {
                service.delete(u.getId());
                return;
            }
        }
        fail("user was not added");
    }

    @Test
    public void delete() throws Exception {
        User user = service.create("todelete", "dontforget", "delete@me.com");

        WebClient.create("http://localhost:4204").path("/user/delete/" + user.getId()
).delete();

        user = service.find(user.getId());
        assertNull(user);
    }

    @Test
    public void show() {
        User user = WebClient.create("http://localhost:4204")
                .path("/user/show/" + users.iterator().next().getId())
                .get(User.class);
        assertEquals("foo", user.getFullname());
        assertEquals("foopwd", user.getPassword());
        assertEquals("foo@foo.com", user.getEmail());
    }

    @Test
    public void list() throws Exception {
        String users = WebClient.create("http://localhost:4204")
                .path("/user/list")
                .get(String.class);
        assertEquals(
                "<users>" +
                        "<user>" +
                        "<email>foo@foo.com</email>" +
                        "<fullname>foo</fullname>" +
                        "<id>1</id>" +
                        "<password>foopwd</password>" +
                        "</user>" +
                        "<user>" +
                        "<email>bar@bar.com</email>" +
                        "<fullname>bar</fullname>" +
```

```
                        "<id>2</id>" +
                        "<password>barpwd</password>" +
                        "</user>" +
                        "</users>", users);
    }

    @Test
    public void update() throws Exception {
        User created = service.create("name", "pwd", "mail");
        Response response = WebClient.create("http://localhost:4204")
                .path("/user/update/" + created.getId())
                .query("name", "corrected")
                .query("pwd", "userpwd")
                .query("mail", "it@is.ok")
                .post(null);

        JAXBContext ctx = JAXBContext.newInstance(User.class);
        Unmarshaller unmarshaller = ctx.createUnmarshaller();
        User modified = (User) unmarshaller.unmarshal(InputStream.class.cast(response
.getEntity()));

        assertEquals("corrected", modified.getFullname());
        assertEquals("userpwd", modified.getPassword());
        assertEquals("it@is.ok", modified.getEmail());
    }
}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.rest.UserServiceTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
INFO - openejb.home = /Users/dblevins/examples/rest-on-ejb
INFO - openejb.base = /Users/dblevins/examples/rest-on-ejb
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
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 - Found EjbModule in classpath: /Users/dblevins/examples/rest-on-
ejb/target/classes
INFO - Beginning load: /Users/dblevins/examples/rest-on-ejb/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/rest-on-ejb
INFO - Configuring Service(id=Default Singleton Container, type=Container, provider-
id=Default Singleton Container)
```

```
INFO - Auto-creating a container for bean UserService: Container(type=SINGLETON,
id=Default Singleton Container)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean org.superbiz.rest.UserServiceTest:
Container(type=MANAGED, id=Default Managed Container)
INFO - Configuring PersistenceUnit(name=user)
INFO - Configuring Service(id=Default JDBC Database, type=Resource, provider-
id=Default JDBC Database)
INFO - Auto-creating a Resource with id 'Default JDBC Database' of type 'DataSource
for 'user'.
INFO - Configuring Service(id=Default Unmanaged JDBC Database, type=Resource,
provider-id=Default Unmanaged JDBC Database)
INFO - Auto-creating a Resource with id 'Default Unmanaged JDBC Database' of type
'DataSource for 'user'.
INFO - Adjusting PersistenceUnit user <jta-data-source> to Resource ID 'Default JDBC
Database' from 'My DataSource'
INFO - Adjusting PersistenceUnit user <non-jta-data-source> to Resource ID 'Default
Unmanaged JDBC Database' from 'My Unmanaged DataSource'
INFO - Enterprise application "/Users/dblevins/examples/rest-on-ejb" loaded.
INFO - Assembling app: /Users/dblevins/examples/rest-on-ejb
INFO - PersistenceUnit(name=user,
provider=org.apache.openjpa.persistence.PersistenceProviderImpl) - provider time 407ms
INFO - Jndi(name="java:global/rest-on-ejb/UserService!org.superbiz.rest.UserService")
INFO - Jndi(name="java:global/rest-on-ejb/UserService")
INFO -
Jndi(name="java:global/EjbModule1789767313/org.superbiz.rest.UserServiceTest!org.super
biz.rest.UserServiceTest")
INFO - Jndi(name="java:global/EjbModule1789767313/org.superbiz.rest.UserServiceTest")
INFO - Created Ejb(deployment-id=org.superbiz.rest.UserServiceTest, ejb-
name=org.superbiz.rest.UserServiceTest, container=Default Managed Container)
INFO - Created Ejb(deployment-id=UserService, ejb-name=UserService, container=Default
Singleton Container)
INFO - Started Ejb(deployment-id=org.superbiz.rest.UserServiceTest, ejb-
name=org.superbiz.rest.UserServiceTest, container=Default Managed Container)
INFO - Started Ejb(deployment-id=UserService, ejb-name=UserService, container=Default
Singleton Container)
INFO - Deployed Application(path=/Users/dblevins/examples/rest-on-ejb)
INFO - Initializing network services
INFO - Creating ServerService(id=httpejbd)
INFO - Creating ServerService(id=admin)
INFO - Creating ServerService(id=ejbd)
INFO - Creating ServerService(id=ejbds)
INFO - Creating ServerService(id=cxf-rs)
INFO - Initializing network services
  ** Starting Services **
  NAME                 IP              PORT
  httpejbd             127.0.0.1       4204
  admin thread         127.0.0.1       4200
  ejbd                 127.0.0.1       4201
  ejbd                 127.0.0.1       4203
```

```
-------
Ready!
WARN - Query "select u from User u" is removed from cache  excluded permanently. Query
"select u from User u" is not cached because it uses pagination..
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.102 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
```