customer - Simple ejb application with a JPA entity

Application Overview

The customer sample is a very simple demonstration of a container managed jpa persistence context accessed from a stateless session bean, in turn accessed from a jsp based web app or standalone java client. Since jsps do no support dependency injection through annotations the ejb is looked up in an auxiliary class using the "legacy" java:comp/env jndi context.

JPA details

CustomerInfo.java is the entity bean that represents the Customer table in the database. By using @Entity, @Table(name = "customer"), and @Id it tells OpenEJB that this is an entity bean, which is representative of the table "customer" and has "customerId" as the primary key. By using these annotations no other configuration is needed inside openejb-jar.xml **CustomerInfo.java** includes 2 named queries, one is called **AllCustomers**; the other is called **Find Custmer**.

CustomerInfo.java

```
package com.service.customer.ejb;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "CustomerInfo")
@NamedQueries({@NamedQuery(name = "AllCustomers", query = "SELECT c from CustomerInfo c"),
        @NamedQuery(name = "FindCustomer", query = "SELECT c from CustomerInfo c  WHERE c.customerId=:
customerId")})
public class CustomerInfo implements Serializable {
    private String customerId;
    private String fullName;
    private String emailAddress;
    private String interests;

    public CustomerInfo() {

    }

    public CustomerInfo(String customerId, String fullName, String emailAddress,
                        String interests) {
        this.customerId = customerId;
        this.fullName = fullName;
        this.emailAddress = emailAddress;
        this.interests = interests;
    }

    @Id
    public String getCustomerId() {
        return customerId;
    }

    public String getFullName() {
        return fullName;
    }

    public String getEmailAddress() {
        return emailAddress;
    }

    public String getInterests() {
        return interests;
    }

    public void setCustomerId(String customerId) {
        this.customerId = customerId;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }

    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public void setInterests(String interests) {
        this.interests = interests;
    }
} // end CustomerInfo
```

Query **AllCustmers** is used from the stateless session bean called **ProcessCustomerSessionBean**, which relies on injection of a **PersistenceContext**; thus we are using container managed persistence contexts, and work is executed in a jta CMT transaction. The **ProcessCustomerSessionBean** class implements all the methods defined in its local and remote interfaces.

ProcessCustomerSessionBean.java

```
package com.service.customer.ejb;

import java.util.List;

import javax.ejb.Local;
import javax.ejb.Remote;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

@Stateless
@Local(ProcessCustomerSessionLocal.class)
@Remote(ProcessCustomerSessionRemote.class)
public class ProcessCustomerSessionBean {
    @PersistenceContext
    protected EntityManager em;

    public ProcessCustomerSessionBean() {

    }

    public List<CustomerInfo> findAllCustomers() {
        Query q = em.createNamedQuery("AllCustomers");
        List<CustomerInfo> customerList = q.getResultList();
        return customerList;
    }

    public CustomerInfo findCustomer(String key) {
        Query q = em.createNamedQuery("FindCustomer");
        q.setParameter("customerId", key);
        CustomerInfo c = (CustomerInfo) q.getSingleResult();
        return c;
    }
}
```

persistence.xml references certain Entity Beans and maps them to use certain database pools. In this case, the entity bean Customer is using the **SampleTxDatasourc** and **SampleNoTxDatasource** database pools.

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
             xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence
/persistence_1_0.xsd">
    <persistence-unit name="CustomerPU">
        <description>Entity Beans for CustomerInfo</description>
        <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
        <jta-data-source>SampleTxDatasource</jta-data-source>
        <non-jta-data-source>SampleNoTxDatasource</non-jta-data-source>
        <class>com.service.customer.ejb.CustomerInfo</class>
        <properties>
            <property name="openjpa.jdbc.SynchronizeMappings" value="false"/>
        </properties>
    </persistence-unit>
</persistence>
```

web.xml references the EJB that was created in ProcessCustomerSessionBean.java. By doing this we are allowing the JSP contents inside the WAR to use this EJB via JNDI.

web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.
xsd"
         version="2.5">

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <ejb-local-ref>
        <ejb-ref-name>ejb/ProcessCustomerSessionBean</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <local>com.service.customer.ejb.ProcessCustomerSessionLocal</local>
    </ejb-local-ref>
</web-app>
```

Usage

The app is visible at http://localhost:8080/customer/