bank - EJB sample application

Application Overview

This sample applications includes an ejb module with a stateless session bean, jpa entity beans, a web client, and a very simple command line non-javaee client. There are no security features.

There are three jpa entities Account, Customer and ExchangeRate. They are not set up to demonstrate any entity relationships.

todo: rewrite to use jpa entity relationships

The jpa entities rely on the datsources defined in the SamplesDatasource plugin.

Application contents

The Banking application consists of the following list of packages and classes.

- org.apache.geronimo.samples.bank.ejb
  - BankManagerFacadeBean - Stateless Session Bean, acting as a service class for different application clients.
  - Account - Entity Bean, represent account entity related data in the DB.
  - Customer - Entity Bean, represents customer entity related data.
  - ExchangeRate - Entity Bean, represents exchange rate relative to a USD.
- org.apache.geronimo.samples.bank.web
  - CustomerServiceServlet - Dispatches web requests of Customer Account Balance Viewer to the service layer.
  - CommonServiceServlet - Dispatches web requests of Exchange Rate viewing scenario.

Finally, the banking application will be deployed as an EAR to the application server. The overview of the structural content of the EAR file is given in the following example.

```
|-Bank.ear
   +-BankEJB.jar
        +-META-INF
            +- persistence.xml
   +-BankWeb.war
        +-jsp
            +- customer_info.jsp
            +- customer_main.jsp
            +- error.jsp
            +- exchange_rates.jsp
            +- index.jsp
        +-WEB-INF
            +- web.xml
            +- classes
   +-META-INF
        +- application.xml
   +- BankDB.sql
```

First, we will look at how the business service layer of the application has been implemented with the help of EJBs.

There is no need for an openejb plan.

persistence.xml defines a persistence unit which is used by an EntityManagerFactory in order to talk to **SampleDatabase** through the **SampleDatasource** configuration. The name that is given to this **<persistent-unit>** will be used when referencing it via an annotation in the EJB. The **<jta-data-source>** and **< non-jta-data-source>** MUST NOT point to the same thing.

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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="BankPU">
        <description>Entity Beans for Bank</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>org.apache.geronimo.samples.bank.ejb.Account</class>
        <class>org.apache.geronimo.samples.bank.ejb.Customer</class>
        <class>org.apache.geronimo.samples.bank.ejb.ExchangeRate</class>
    </persistence-unit>
</persistence>
```

web.xml lists the servlets and url-patterns.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_5.xsd"
    version="2.5">

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

    <servlet>
        <display-name>CustomerServiceServlet</display-name>
        <servlet-name>CustomerServiceServlet</servlet-name>
        <servlet-class>org.apache.geronimo.samples.bank.web.CustomerServiceServlet</servlet-class>
    </servlet>

    <servlet>
        <display-name>CommonServiceServlet</display-name>
        <servlet-name>CommonServiceServlet</servlet-name>
        <servlet-class>org.apache.geronimo.samples.bank.web.CommonServiceServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>CustomerServiceServlet</servlet-name>
        <url-pattern>/customer_info</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>CommonServiceServlet</servlet-name>
        <url-pattern>/exchange_rates</url-pattern>
    </servlet-mapping>
</web-app>
```

No web plan is needed.

The geronimo plan **plan.xml** contains the db initialization gbean that runs a script to create tables and populate them. If you leave this out openjpa will create or update the structure of the tables on first access. The unprocessed version of this plan is at bank-jetty/src/main/plan/plan.xml. The processed version shown here with plugin name and all dependencies filled in can be found at bank-jetty/target/resources/META-INF/plan.xml after building the project.

plan.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-1.2">
  <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2">
    <dep:moduleId>
      <dep:groupId>org.apache.geronimo.samples</dep:groupId>
      <dep:artifactId>bank-jetty</dep:artifactId>
      <dep:version>2.1.2</dep:version>
      <dep:type>car</dep:type>
    </dep:moduleId>
    <dep:dependencies>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.samples</dep:groupId>
        <dep:artifactId>sample-datasource</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>jetty6</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>jasper</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>openejb</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
      <dep:dependency>
        <dep:groupId>org.apache.geronimo.configs</dep:groupId>
        <dep:artifactId>openjpa</dep:artifactId>
        <dep:version>2.1.2</dep:version>
        <dep:type>car</dep:type>
      </dep:dependency>
    </dep:dependencies>
    <dep:hidden-classes/>
    <dep:non-overridable-classes/>
  </dep:environment>
  <gbean name="DBInitialization" class="org.apache.geronimo.connector.DatabaseInitializationGBean">
    <!--<attribute name="testSQL">select * from customer</attribute>-->
    <attribute name="path">BankDB.sql</attribute>
    <reference name="DataSource">
      <name>SampleTxDatasource</name>
    </reference>
  </gbean>
</application>
```

Sample Database

The sample database that is being used to demonstrate this application is inbuilt Derby database. The name of the sample database is **BankDB** and it consists of three tables, CUSTOMER ,ACCOUNT and EXCHANGE_RATE. The fields for each of these tables are described below.

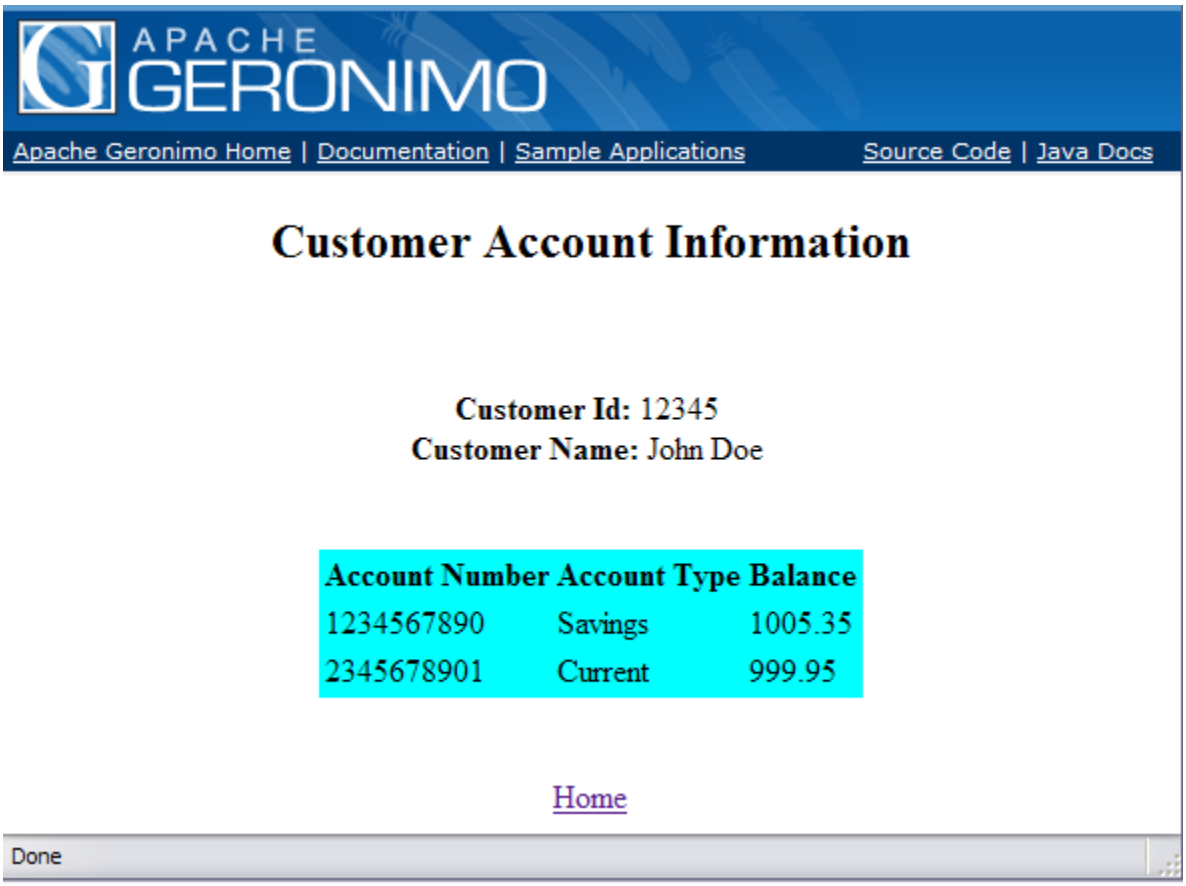
Table Name	Fields
CUSTOMER	customerId (PRIMARY KEY) name

ACCOUNT	accountNumber (PRIMARY KEY) accountType balance customerId
EXCHANGERATE	rateId (PRIMARY KEY) currency rate

The **CUSTOMER** table stores the data related to the customers. It stores only the identification number and and the name. **ACCOUNT** table has a unique account number for identification. Account type and balance are the other information stored. Account.customerId is a foriegn key to the Customer table which is the owner of the Account. **EXCHANGERATE** table has a primary key of rateId for an identification. Each record of EXCHANGERATE has currency name and rate paid by the bank.

Banking Web Application

To test the sample web application open a browser and type http://localhost:8080/bank. It will forward you to the index page of banking application which has direct links to the view customer and exchange rate information. To view the list of account information of each customer, provide a relavant customer id in the DB. Exchange rate page will display list of all currencies in the exchange rate table. (Note that 'bank' is case-sensitive)



Banking command line client

The bank command line client does a JNDI lookup to the remote interface. Later, the list of ExchangeRate is obtained and printed out on the screen.

BankClient.java

```
package org.apache.geronimo.samples.bank.client;

import java.util.List;
import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;

import org.apache.geronimo.samples.bank.ejb.BankManagerFacadeRemote;
import org.apache.geronimo.samples.bank.ejb.ExchangeRate;

public class BankClient {
    public static void main(String[] args) {


        Properties p = new Properties();
        p.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.apache.openejb.client.RemoteInitialContextFactory");
        p.setProperty(Context.PROVIDER_URL, "ejbd://localhost:4201");
        try {
            Context ic = new InitialContext(p);
            BankManagerFacadeRemote bmr = (BankManagerFacadeRemote) ic.lookup("BankManagerFacadeBeanRemote");
            List<ExchangeRate> rates = bmr.getExchangeRates();
            System.out.println("Exchange Rates");
            for (int i = 0; i < (int) rates.size(); i++) {
                ExchangeRate rate = rates.get(i);
                System.out.println("Currency: " + rate.getCurrency());
                System.out.println("Rate: " + rate.getRate());
                System.out.println();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Please note that "BankManagerFacadeBeanRemote" is used here as the look up name instead of "BankManagerFacadeRemote". Otherwise, you'll get an error like this -

javax.naming.NameNotFoundException: /BankManagerFacadeRemote does not exist in the system. Check that the app was successfully deployed.

To test the command line client first determine a useable (although excessive classpath) by running

```
mvn dependency:build-classpath
```

If you are using windows platform, you need to replace the ":" separator that separates the classpath jars with the ";" separator. Also, if you use the default maven 2 home, you need to replace "c:\documents and settings" with "c:\docume~1".

then in bank-client run

```
java -cp <output from above>:target/bank-client-<version>.jar org.apache.geronimo.samples.bank.client.BankClient
```

```
C:\gsamples\branches\2.1\samples\bank\bank-client>java -cp C:\docume~1\Administr
ator\.m2\repository\org\apache\geronimo\samples\bank-ejb\2.1.2-SNAPSHOT\bank-ejb
-2.1.2-SNAPSHOT.jar;C:\docume~1\Administrator\.m2\repository\org\apache\geronimo
\specs\geronimo-annotation_1.0_spec\1.1.1\geronimo-annotation_1.0_spec-1.1.1.jar
;C:\docume~1\Administrator\.m2\repository\org\apache\geronimo\specs\geronimo-ejb
_3.0_spec\1.0.1\geronimo-ejb_3.0_spec-1.0.1.jar;C:\docume~1\Administrator\.m2\re
pository\org\apache\geronimo\specs\geronimo-jpa_3.0_spec\1.1.1\geronimo-jpa_3.0_
spec-1.1.1.jar;C:\Docume~1\Administrator\.m2\repository\org\apache\geronimo\spec
s\geronimo\jsp_2.1_spec\1.0.1\geronimo-jsp_2.1_spec-1.0.1.jar;C:\Docume~1\Admini
strator\.m2\repository\org\apache\geronimo\specs\geronimo-servlet_2.5_spec\1.1.2
\geronimo-servlet_2.5_spec-1.1.2.jar;C:\docume~1\Administrator\.m2\repository\or
g\apache\openejb\javaee-api\5.0-1\javaee-api-5.0-1.jar;C:\docume~1\Administrator
\.m2\repository\org\apache\openejb\openejb-client\3.0\openejb-client-3.0.jar;tar
get\bank-client-2.1.2-SNAPSHOT.jar org.apache.geronimo.samples.bank.client.BankC
lient
Exchange Rates
Currency: EURO
Rate: 0.812

Currency: YEN
Rate: 111.15

Currency: SLR
Rate: 99.18
```