



UNIVERSITY OF TRENTO - Italy
**Department of Information Engineering
and Computer Science**

Course - Web Architecture

Simple Java Enterprise Application - project report

Date: 05/11/2020

Done By:

Name: Ephrem Tiebeb Mekonnen
Matricola : 213006

Introduction

The idea of this project is to implement what we have learned in class by developing simple java enterprise applications. In this assignment, I require to do two separate tasks and I have done and named them mekonnen4a and mekonnen4b, the first and second tasks respectively.

The first project is to develop and deploy an EJB application that can be invoked from a standalone remote java client by following a tutorial from the following link.

(<http://www.mastertheboss.com/jboss-server/jboss-as-7/jboss-as-7-remote-ejb-client-tutorial>).

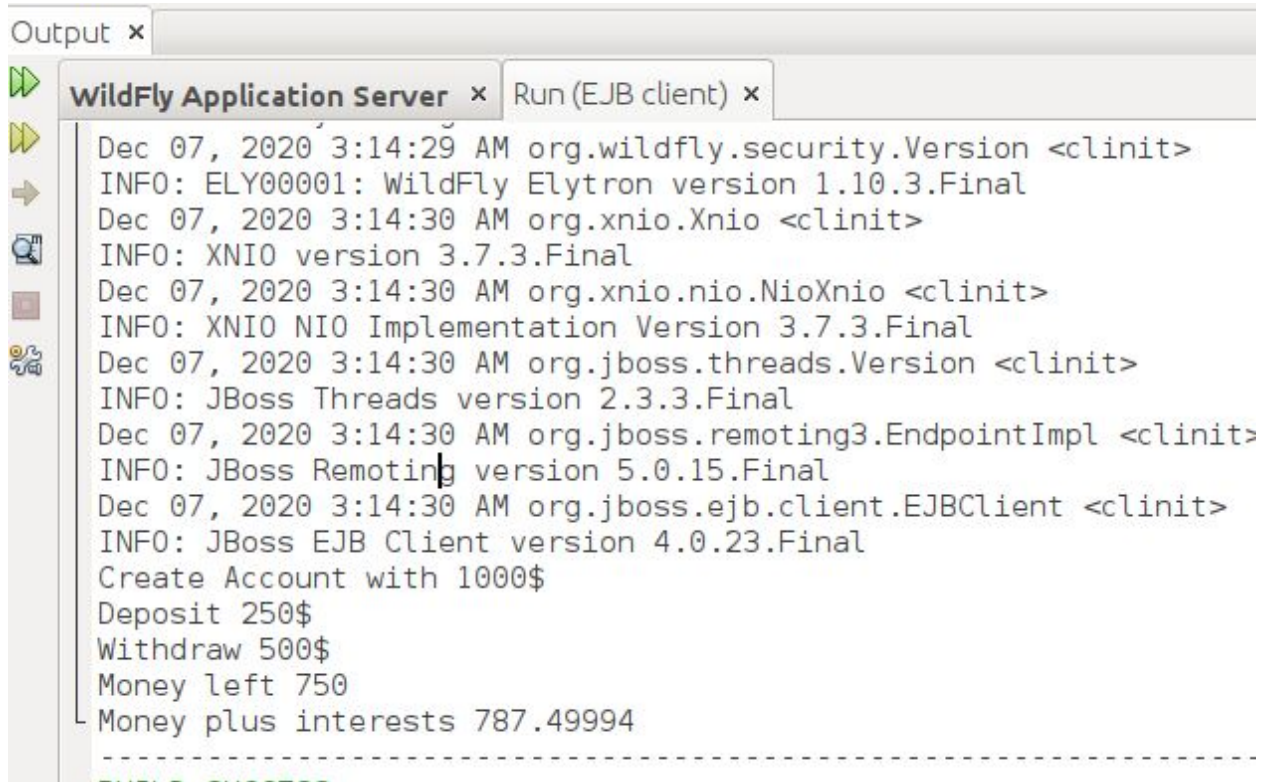
In the second application, it is similar to the first application except for the application accesses data from the derby database and the client application is a web application(it was a desktop application in the first application(Mekonnen 4a).

Section 1: Deploying the tutorial

1.2. Screenshots

```
147 INFO [stdout] (default task-2) Money deposit. total is 1250
155 INFO [stdout] (default task-2) Money withdrawal. total is 750
```

Fig 1. Output after 250\$ deposited on the account created with 1000\$ balance withdrew 750\$ and total balance. (Server-side)



```

Output x
WildFly Application Server x Run (EJB client) x
Dec 07, 2020 3:14:29 AM org.wildfly.security.Version <clinit>
INFO: ELY00001: WildFly Elytron version 1.10.3.Final
Dec 07, 2020 3:14:30 AM org.xnio.Xnio <clinit>
INFO: XNIO version 3.7.3.Final
Dec 07, 2020 3:14:30 AM org.xnio.nio.NioXnio <clinit>
INFO: XNIO NIO Implementation Version 3.7.3.Final
Dec 07, 2020 3:14:30 AM org.jboss.threads.Version <clinit>
INFO: JBoss Threads version 2.3.3.Final
Dec 07, 2020 3:14:30 AM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 5.0.15.Final
Dec 07, 2020 3:14:30 AM org.jboss.ejb.client.EJBClient <clinit>
INFO: JBoss EJB Client version 4.0.23.Final
Create Account with 1000$
Deposit 250$
Withdraw 500$
Money left 750
Money plus interests 787.49994
-----

```

Fig 2. The final output from the client-side.

Section 2: Modification of the tutorial

2.1. Implementation

In this project, the application needs to be connected with a database. So the first step is to download the derby database and configure it with the EJB application which is deployed on the WildFly server. The EJB server application has also contained interfaces and implementation of the EJBs. For the application to work with a database, we need to create a class where we define our model and so I created a class named “User.java”(POJO persistent model) which comprises attributes and methods which is going to be mapped with a relational database using JPA.

After the POJO persistent model and database are created, the database should be configured with the WildFly server. Thus, we need to follow some steps to make it work (Tutorial I followed <https://tomylab.wordpress.com/2016/07/24/how-to-add-a-datasource-to-wildfly/>).

Finally, we need to enable the connection to use JPA. Adding support for JPA is achieved by defining the persistence.xml file in the META-INF directory within the application package. The persistence.xml file is the standard configuration file in JPA. In this file, I defined a uniquely named persistence unit which will be used by the EntityManager defined in AccountEJB.java. The provider attribute specifies the

underlying implementation of the JPA EntityManager. In JBoss, the default and supported/recommended JPA provider is Hibernate. The jta-data-source points to the JNDI name of the database connection to which this persistence unit maps.

Note that dependencies related to hibernate should be included in the pom.xml of the EJB-server application package:

Dependencies should be included in pom.xml of in order for the application to support Hibernate.

```
...  
<dependency>  
  <groupId>org.hibernate</groupId>  
  <artifactId>hibernate-entitymanager</artifactId>  
  <version>4.2.8.Final</version>  
</dependency>  
<dependency>  
  <groupId>org.hibernate</groupId>  
  <artifactId>hibernate-core</artifactId>  
  <version>5.2.11.Final</version>  
</dependency>  
<dependency>  
  <groupId>org.hibernate.validator</groupId>  
  <artifactId>hibernate-validator</artifactId>  
  <version>6.1.6.Final</version>  
</dependency>  
<dependency>  
  <groupId>org.hibernate</groupId>  
  <artifactId>hibernate-search-orm</artifactId>  
  <version>5.11.5.Final</version>  
</dependency>  
....
```

The last but not the least thing that needs to be taken into consideration is the transaction-related things. The application EJB-server is a simple banking application so that the system should ensure that a unit of work either fully completes, or the work is fully rolled back. I chose a bean-managed transaction. I used TransactionManagement annotation to declare a transaction type. The value of the Transaction management annotation is either CONTAINER or BEAN.[1]

Sample code of deposit method from AccountEJB.java to show how the annotation and other things are handled.

```
@TransactionAttribute(TransactionAttributeType.REQUIRED)
@Override
public boolean deposit(int account_number, float amount) {
    boolean is_deposited = false;
    try {
        userTransaction.begin();
        User account = em.find(User.class, account_number);

        if (account != null) {

            float money = account.getBalance();
            float newAmount = money + amount;
            account.setBalance(newAmount);
            em.persist(account);
            System.out.println("Money deposit. total is " + money);
            userTransaction.commit();
            is_deposited = true;
        } else {
            userTransaction.commit();
        }
    } catch (IllegalStateException ex) {
        try {
            userTransaction.rollback();
        } catch (IllegalStateException | SecurityException | SystemException ex1) {
            Logger.getLogger(AccountEJB.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } catch (NotSupportedException | SystemException | RollbackException |
HeuristicMixedException | HeuristicRollbackException | SecurityException ex) {
        Logger.getLogger(AccountEJB.class.getName()).log(Level.SEVERE, null, ex);
    }

    return is_deposited;
}
```

After creating the EJB-server application, I created a remote web application that invokes the EJB-server using JNDI-properties gained when you run the EJB-server.

2.2 Deployment

The application has a table named “users” in a database called BankDB. The table "users" has three columns (AccountNumber, OwnerName, Balance). Also, it has a record with 1, Ephrem, 5387.0 for AccountNumber, Owner name, Balance columns respectively.

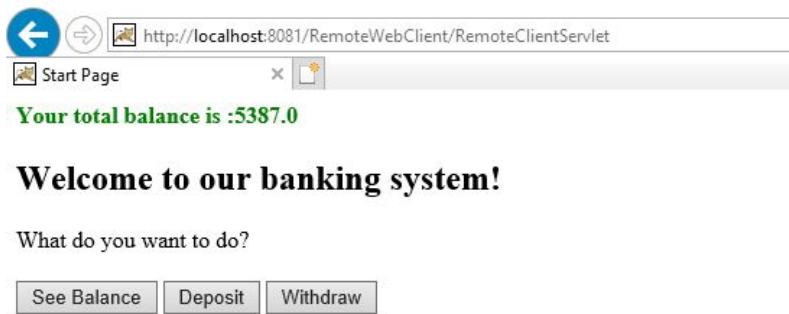


Fig 1. Page displayed when the button See Balance is clicked.

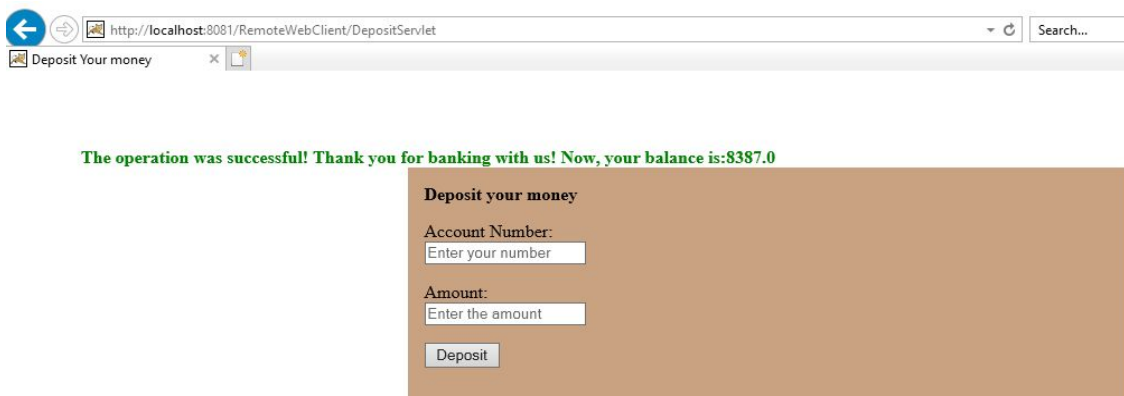


Fig 2. Page displayed after I deposited 2000\$ on account number 1.

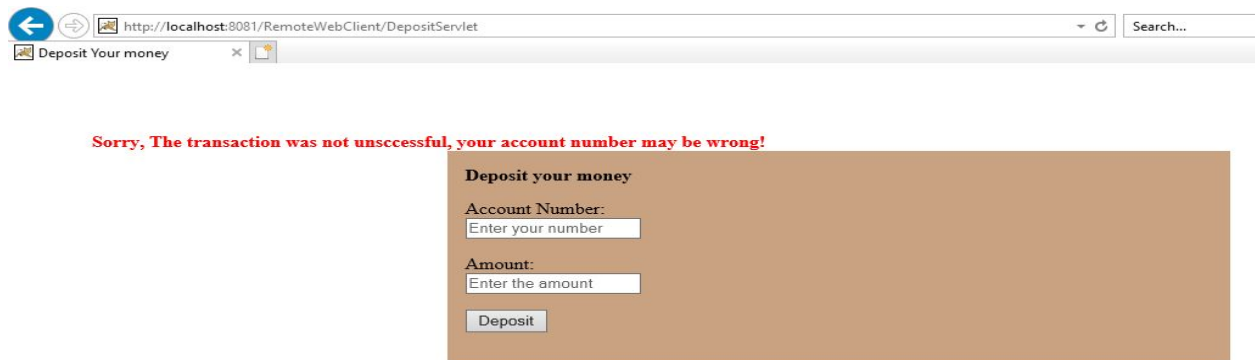


Fig 3. Page displayed when I tried to deposit money with a wrong account number

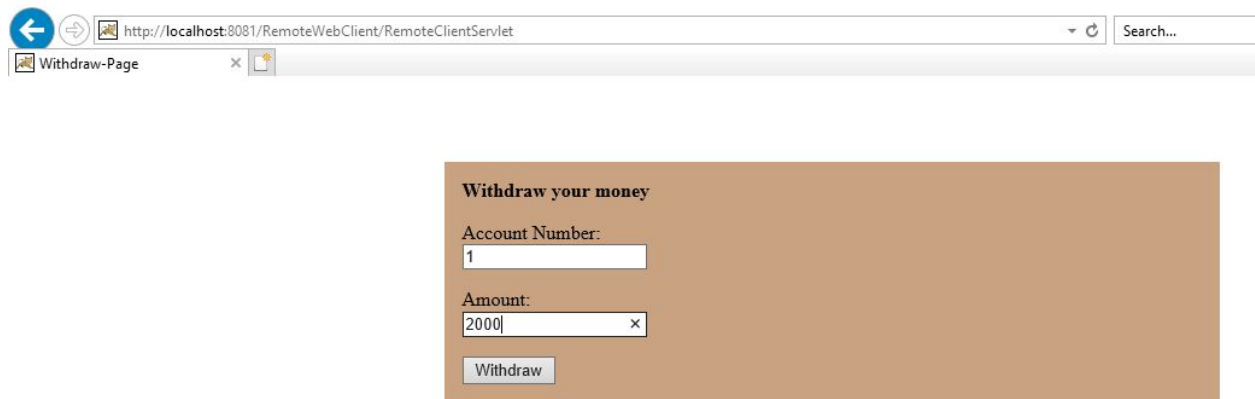


Fig 4. Trying to withdraw 2000\$ from account number 1

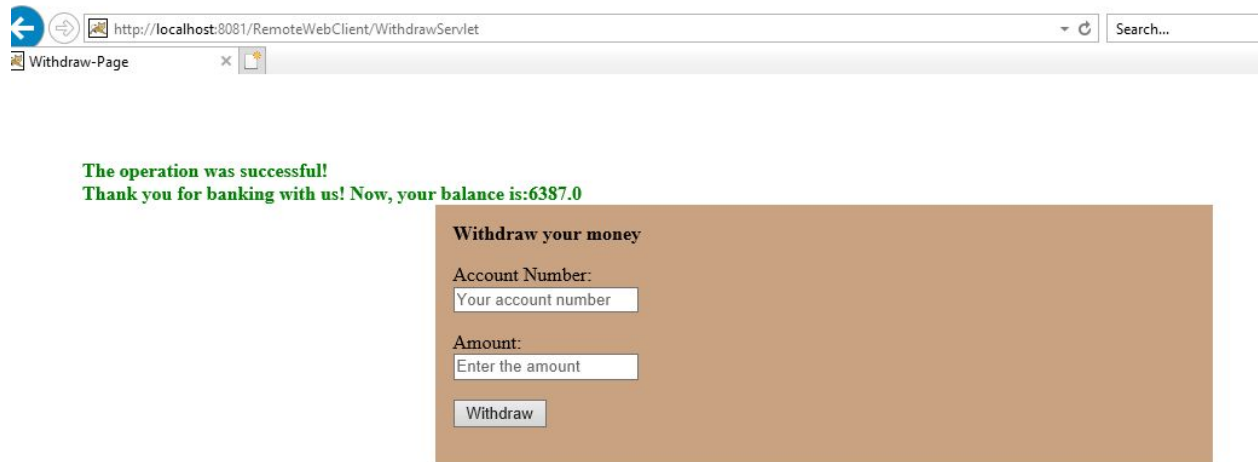


Fig 5. Successfully deducted 2000\$ from account number 1.

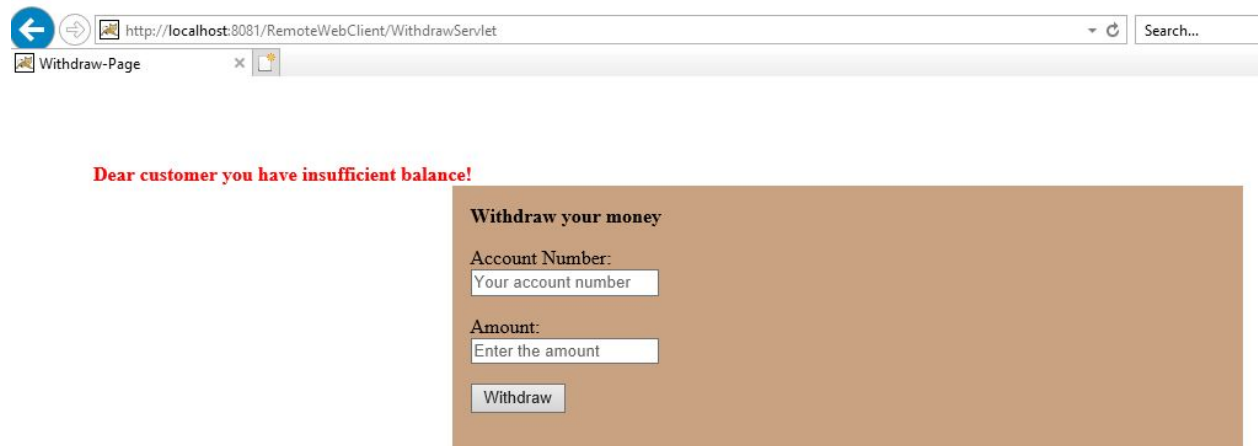


Fig 6. Page displayed when customer try to withdraw more than his/her available balance

2.3. Comments and Notes

During the process to do this project(task) I encountered a problem with Netbeans IDE. After I followed complete steps to configure the data source in the standalone.xml file found in the WildFly installation directory and when I tried to deploy the application, it did not work. The problem was that the WildFly server was reading standalone-full.xml instead of standalone.xml file where I defined the database data source. The problem was solved by providing a correct standalone.xml file to the WildFly server in Netbeans IDE(Go to service tab -> Right-click on the WildFly server-> and then properties-> browse and provide the standalone.xml for the configuration file).

Reference

- (1) "EJB Transaction Management Example | Examples Java Code Geeks - 2020." *Examples Java Code Geeks*, <https://www.facebook.com/javacodegeeks>, 22 Apr. 2015, <https://examples.javacodegeeks.com/enterprise-java/ejb3/transactions/ejb-transaction-management-example/>.