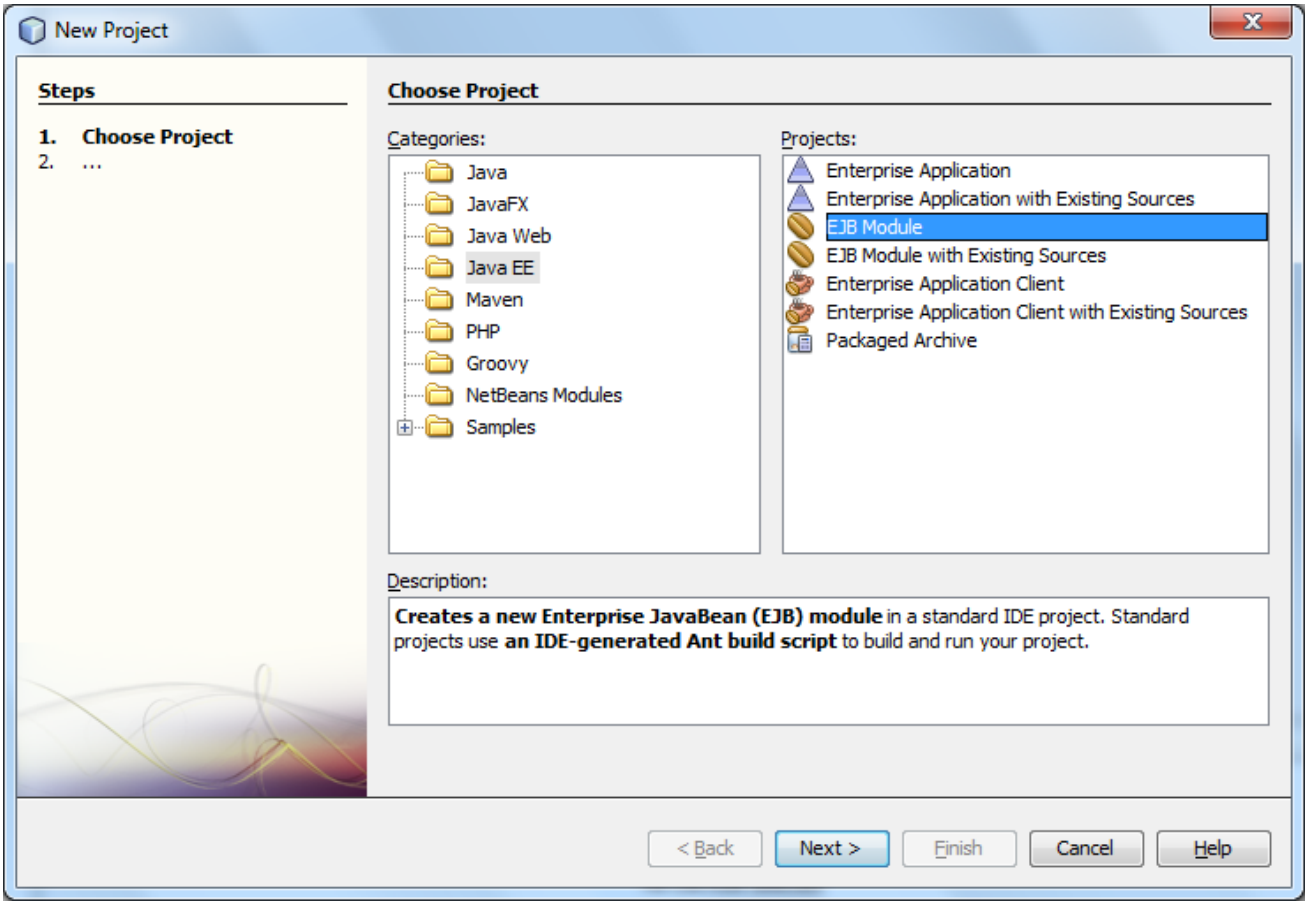
Container Managed Transaction

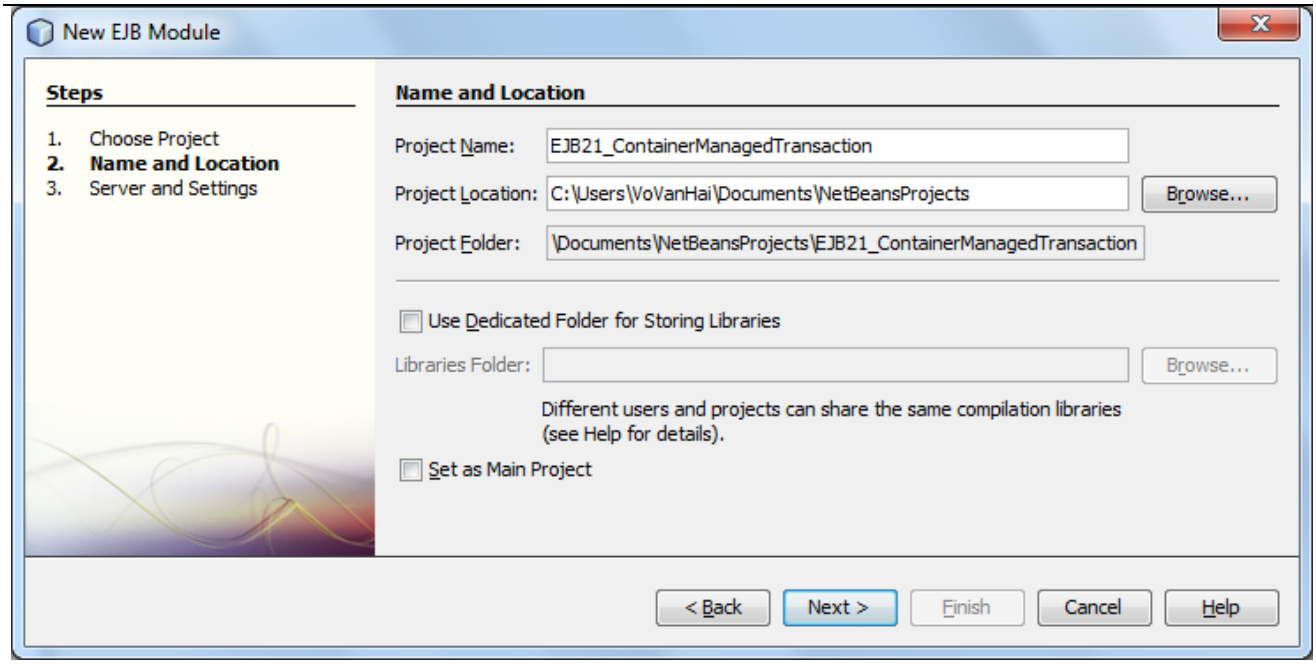
Database:

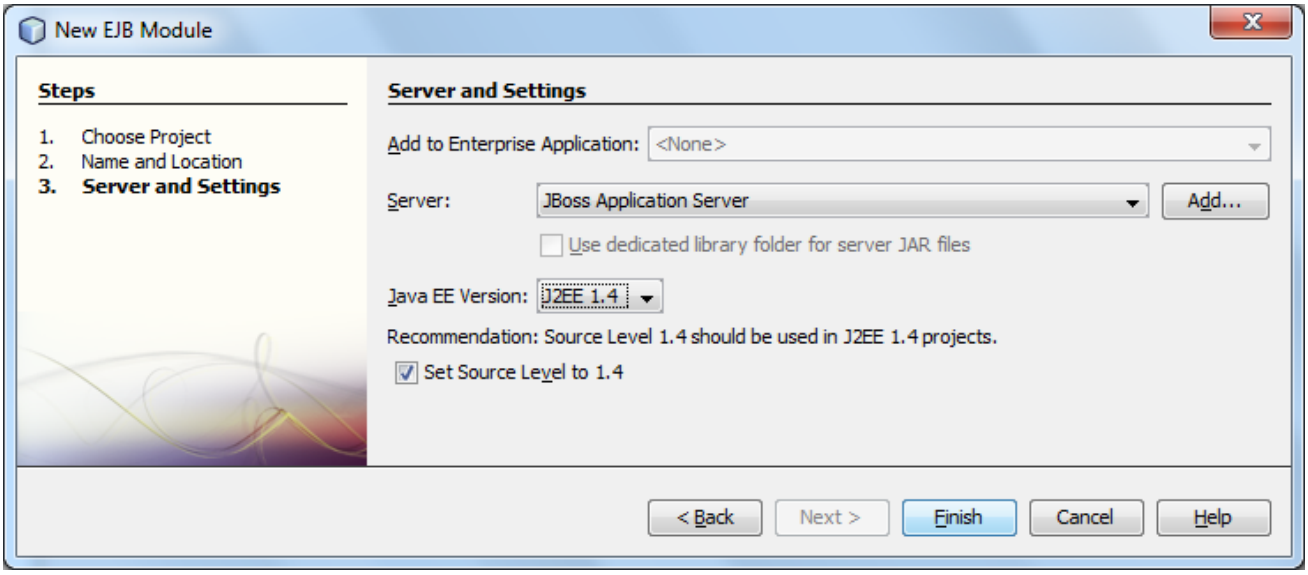
```
CREATE TABLE USERS
  (
    LOGIN_NAME VARCHAR(20) NOT NULL,
    PASSWORD VARCHAR(20)
  );

CREATE TABLE COURSES
  (
   COURSE_ID      VARCHAR(10) NOT NULL,
   COURSE_NAME   VARCHAR(64),
   FEE           NUMERIC(18),
   MAX_LIMIT     INT,
   CURR_ENROLL   INT
  );

CREATE TABLE STUDENTS
  (
    STUDENT_ID VARCHAR(12) NOT NULL,
    FIRST_NAME VARCHAR(15) ,
    LAST_NAME VARCHAR(15) ,
    ADDRESS VARCHAR(164),
    EMAIL_ADDRESS VARCHAR(64)
  );

CREATE TABLE ORDERS
  (
    ORDER_ID VARCHAR(64),
    STUDENT_ID VARCHAR(64),
    ORDER_DATE TIMESTAMP,
    AMOUNT NUMERIC(18),
    STATUS VARCHAR(64)
  );

CREATE TABLE ORDER_LINE_ITEMS
  (
    ID VARCHAR(64),
    COURSE_ID VARCHAR(64),
    ORDER_ID VARCHAR(64),
    FEE NUMERIC(18)
  );
CREATE TABLE ENROLLMENTS
  (
    ENROLLMENT_ID VARCHAR(64),
    STUDENT_ID VARCHAR(64),
    COURSE_ID VARCHAR(64)
  ) ;


INSERT INTO USERS VALUES ( 'ragae', 'ragae');
INSERT INTO USERS VALUES ( 'krishna', 'krishna');
INSERT INTO USERS VALUES ( 'raghava', 'raghava');
INSERT INTO USERS VALUES ( 'raymond', 'raymond');

INSERT INTO COURSES VALUES ('CS201', 'INTRODUCTION TO COMPUTERS
```

```
PROGRAMMING', 100, 25, 0);
INSERT INTO COURSES VALUES ('CS205', 'WORD PROCESSING',                200, 25, 0);
INSERT INTO COURSES VALUES ('CS231', 'DATABASE DESIGN',                300, 25, 0);
INSERT INTO COURSES VALUES ('CS105', 'XML PRIMER',                250, 20, 0);
INSERT INTO COURSES VALUES ('CS120', 'HTML AND WWW APPLICATION DESIGN',       200,
30, 0);
INSERT INTO COURSES VALUES ('CS305', 'ALGORITHMS', 100,                20, 0);
INSERT INTO COURSES VALUES ('CS405', 'AUTOMATA THEORY AND FORMAL
COMPUTATION',100, 15, 0);
INSERT INTO COURSES VALUES ('CS400', 'COMPUTER SYSTEMS: ARCHITECTURE,
DESIGN',100, 15, 0);

INSERT INTO STUDENTS VALUES ('ragae', 'RAGAE',   'GHALY',     '15 DIAMOND ST, BOSTON,
MA', 'GHALY@XYZ.COM' );
INSERT INTO STUDENTS VALUES ('krishna', 'KRISHNA', 'KOTHAPALLI','11 ORANGE AVE,
SACRAMENTO, CA', 'KRISHNA@XYZ.COM' );
INSERT INTO STUDENTS VALUES ('sam', 'SAM',     'LE',        '12 APPLEBEE RD, LOS ANGELES,
CA', 'SAM@XYZ.COM' );
INSERT INTO STUDENTS VALUES ('paul', 'PAUL',    'ODONALD',   '99 GOLDGATE BL, MIAMI,
FL', 'PAUL@XYZ.COM' );
INSERT INTO STUDENTS VALUES ('laura', 'LAURA',   'GHALY',     '606 CLINTON CT, CHICAGO,
IL', 'LAURA@XYZ.COM' );
```

Tạo Java EE Project có tên EJB21_ContainerManagedTransaction.







Code cho các lớp:

Lớp EnrollmentCart.java

```
package vovanhai.wordpress.com;
import java.util.*;
import javax.ejb.*;
import java.rmi.RemoteException;
/**
  EnrollmentCart is the remote interface for enrollment cart
  stateful session bean.
 */
public interface EnrollmentCart extends EJBObject
{
   public void addEnrollment(EnrollmentInfo enroll) throws RemoteException;
   public void deleteEnrollment(int itemId) throws RemoteException;
   public Collection getEnrollments() throws RemoteException;
   public void emptyCart() throws RemoteException;
   public void register() throws InsufficientRoomException, RemoteException;
}
```

Lớp EnrollmentCartHome.java

```
package vovanhai.wordpress.com;
import java.rmi.RemoteException;
import javax.ejb.*;
/**
  EnrollmentCartHome is the remote home interface for
  the enrollment cart stateful session bean.
 */
public interface EnrollmentCartHome extends EJBHome
{
  EnrollmentCart create() throws CreateException, RemoteException;
}
```

Lớp EnrollmentCartEJB.java

```
package vovanhai.wordpress.com;

import java.util.*;
import javax.ejb.*;
import javax.naming.*;
import java.sql.*;
import java.rmi.RemoteException;

/**
   EnrollmentCartEJB is stateful session bean, representing  the
   private conversation of a specific client. It keeps track of the
   user's current selection of courses.
 */

public class EnrollmentCartEJB implements SessionBean, SessionSynchronization
{
   private SessionContext ejbCtx;
   private HashSet cart;
   private boolean isFailed = false;

   public EnrollmentCartEJB(){}
   // SessionSynchronization methods

    public void afterBegin() {
    }

    public void beforeCompletion() {
        if (isFailed) {
            ejbCtx.setRollbackOnly();
        }
    }

    public void afterCompletion(boolean committed) {
        if (committed == false) {
            throw new EJBException("Transaction afterCompletion failed");
        }
        System.out.println("afterCompletion: Transaction succeeds...");
    }
   // Business methods
   public void addEnrollment(EnrollmentInfo enroll) throws RemoteException {
```

```
    cart.add(enroll);
  }
  public void register() throws InsufficientRoomException, RemoteException {
    java.sql.Statement stmt = null;
    java.sql.Statement stmt2 = null;
    java.sql.Connection conn = null;
    System.out.println("register: started..");
    try{
       InitialContext initCtx  = new InitialContext();
      javax.sql.DataSource ds =
                          (javax.sql.DataSource)initCtx.lookup ("java:comp/env/jdbc/studentsDB");
      conn = ds.getConnection();
      stmt = conn.createStatement();
      stmt2 = conn.createStatement();
    } catch (Exception e){
      e.printStackTrace();
                  try{conn.close();}catch(Exception xx){}
    }
    try{
     Iterator it = cart.iterator();
     while (it.hasNext()){
      EnrollmentInfo enroll=(EnrollmentInfo)it.next();
      ResultSet rs = stmt.executeQuery
        ("SELECT * FROM COURSES WHERE COURSE_ID='" +
             enroll.getCourseId()+"'");
      while (rs.next()){
          int mlimit = rs.getInt("MAX_LIMIT");
          int curr   = rs.getInt("CURR_ENROLL");
          if (mlimit < curr + 1){
            isFailed = true;
            ejbCtx.setRollbackOnly();
            throw new InsufficientRoomException();
          }else{
            curr++;
            stmt2.executeUpdate("UPDATE COURSES SET CURR_ENROLL =" + curr
              + " WHERE COURSE_ID='" + enroll.getCourseId()+"'");
            System.out.println("register: Database updated for: "+
                enroll.getCourseId());
          }
       }
     }
     System.out.println("register: success..");
                  conn.close();
    }catch (SQLException ex){
                  try{conn.close();}catch(Exception xx){}
      // This is a system level exception
      throw new EJBException
      ("Transaction rollback due to SQLException: "+ ex.getMessage());
    }
  }
  public Collection getEnrollments() throws RemoteException {
    return cart;
  }

  public void emptyCart() throws RemoteException {
    cart.clear();
  }
  public void deleteEnrollment(int itemId) throws RemoteException {
```

```
        for(Iterator i = getEnrollments().iterator(); i.hasNext(); ) {
          EnrollmentInfo tmpEnrol = (EnrollmentInfo) i.next();
          if (tmpEnrol.getItemId() == itemId) {
            getEnrollments().remove(tmpEnrol);
            break;
          }
        }
      }
   // EJB methods
   public void setSessionContext(SessionContext ctx){
     this.ejbCtx = ctx;
   }
   public void ejbCreate() throws CreateException{
     cart = new HashSet();
   }
   public void ejbRemove() {}
   public void ejbActivate(){}
   public void ejbPassivate(){}
}
```

Lớp EnrollmentInfo.java

```
package vovanhai.wordpress.com;

public class EnrollmentInfo implements java.io.Serializable {

    public int itemId = 0;
    public int studentId = 0;
    public String courseId = null;

    public EnrollmentInfo(int itemId, int studentId, String courseId) {
        this.itemId = itemId;
        this.studentId = studentId;
        this.courseId = courseId;
    }

    public int getItemId() {
        return itemId;
    }

    public int getStudentId() {
        return studentId;
    }

    public String getCourseId() {
        return courseId;
    }
}
```

Lớp InsufficientRoomException.java

```
package vovanhai.wordpress.com;

public class InsufficientRoomException extends Exception {
```

```
    public InsufficientRoomException() {
        super();
    }

    public InsufficientRoomException(Exception e) {
        super(e.toString());
    }

    public InsufficientRoomException(String s) {
        super(s);
    }
}
```

Các file cấu hình

File ejb-jar.xml : ở đây ta dùng CMT nên ta qui định <transaction-type> là Container. <method-name> có giá trị * có nghĩa là tất cả các phương thức trong bean đều có giao dịch. Bạn nên chỉ định chỉ 1 số phương thức nào cần giao dịch mà thôi.

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC
'-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN'
'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>
<ejb-jar>
    <enterprise-beans>
        <session>
            <ejb-name>EnrollmentCart</ejb-name>
            <home>vovanhai.wordpress.com.EnrollmentCartHome</home>
            <remote>vovanhai.wordpress.com.EnrollmentCart</remote>
            <ejb-class>vovanhai.wordpress.com.EnrollmentCartEJB</ejb-class>
            <session-type>Stateful</session-type>
            <transaction-type>Container</transaction-type>
            <resource-env-ref>
                <resource-env-ref-name>jdbc/studentsDB</resource-env-ref-name>
                <resource-env-ref-type>javax.sql.DataSource</resource-env-ref-type>
            </resource-env-ref>
        </session>
    </enterprise-beans>
    <assembly-descriptor>
        <container-transaction>
            <description>description of transaction</description>
            <method>
                <ejb-name>EnrollmentCart</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
        </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

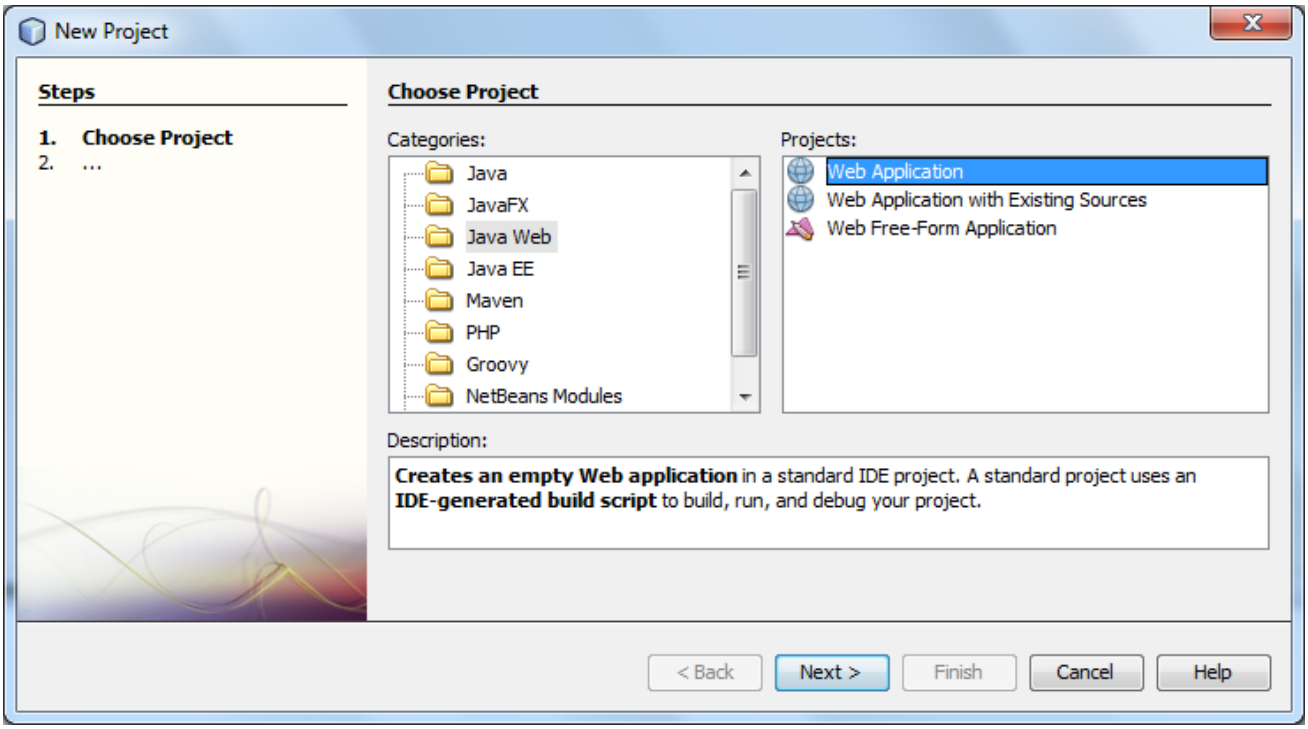
File jboss.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

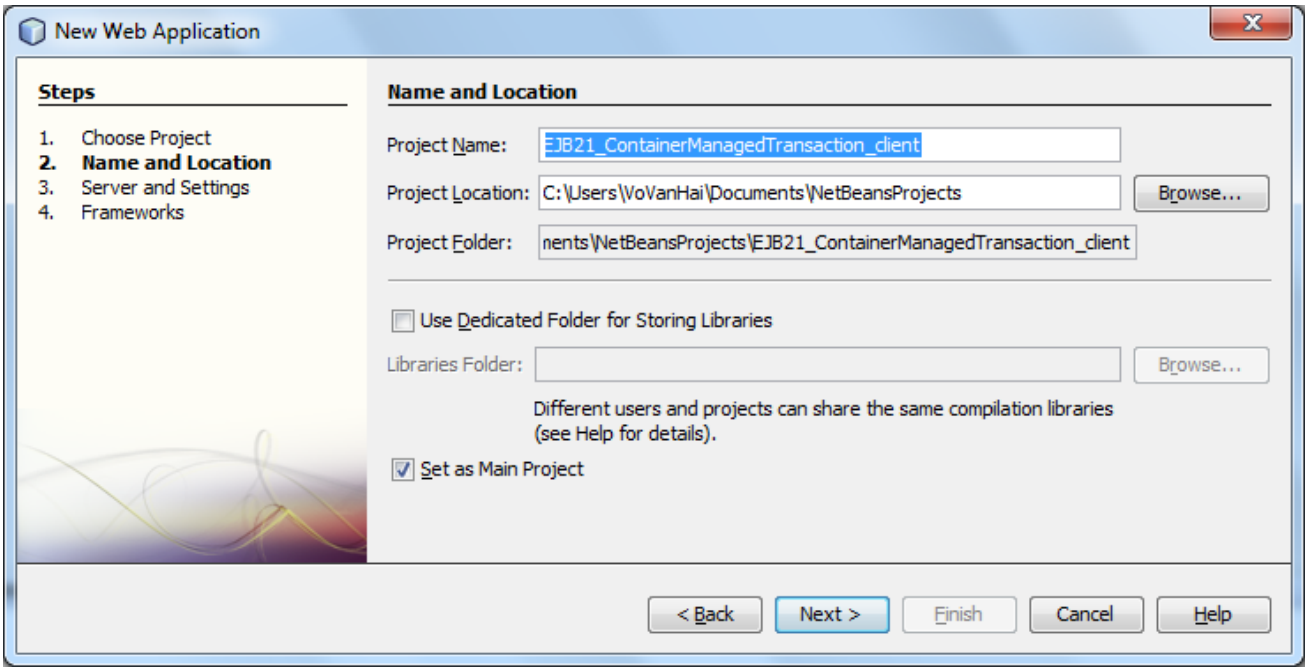
```
<jboss>
   <enterprise-beans>
      <session>
         <ejb-name>EnrollmentCart</ejb-name>
         <jndi-name>cmt/EnrollmentCartCMT</jndi-name>
         <resource-env-ref>
            <resource-env-ref-name>jdbc/studentsDB</resource-env-ref-name>
            <jndi-name>java:/Students_DSN</jndi-name>
         </resource-env-ref>
      </session>
   </enterprise-beans>

   <container-configurations>
      <container-configuration>
         <container-name>Standard Stateful SessionBean</container-name>
         <container-cache-conf>
            <cache-policy>org.jboss.ejb.plugins.LRUStatefulContextCachePolicy</cache-policy>
            <cache-policy-conf>
               <max-capacity>10</max-capacity>
            </cache-policy-conf>
         </container-cache-conf>
      </container-configuration>
   </container-configurations>
</jboss>
```

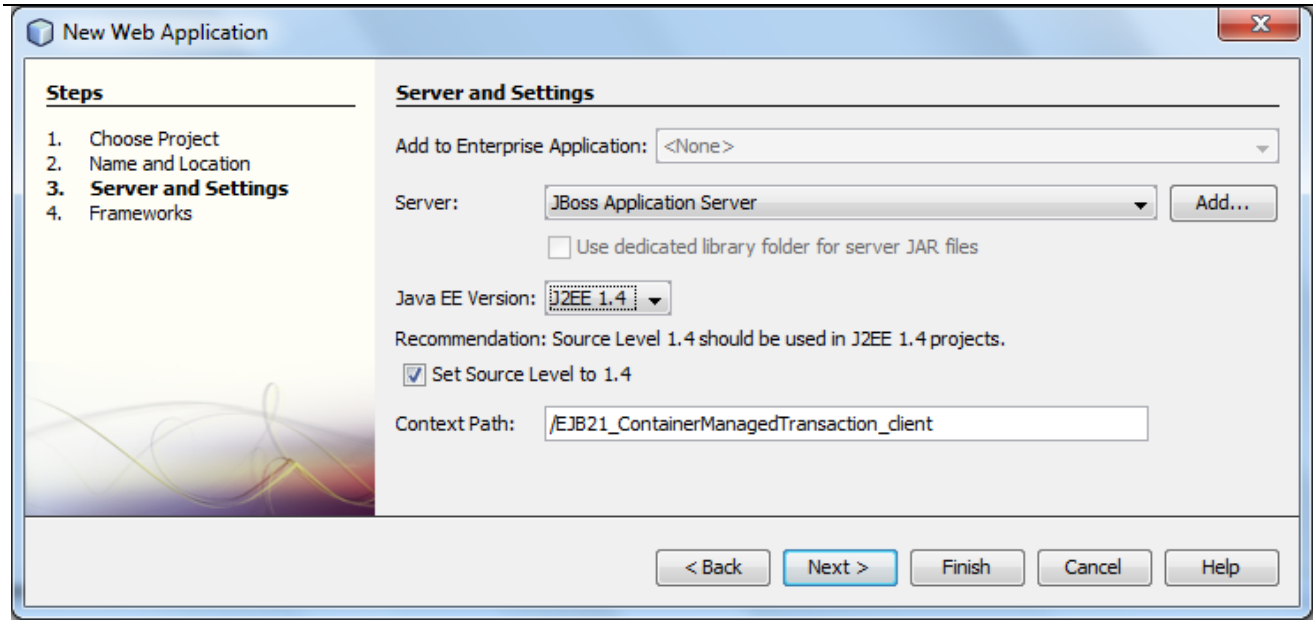
Tiến hành triển khai

Client

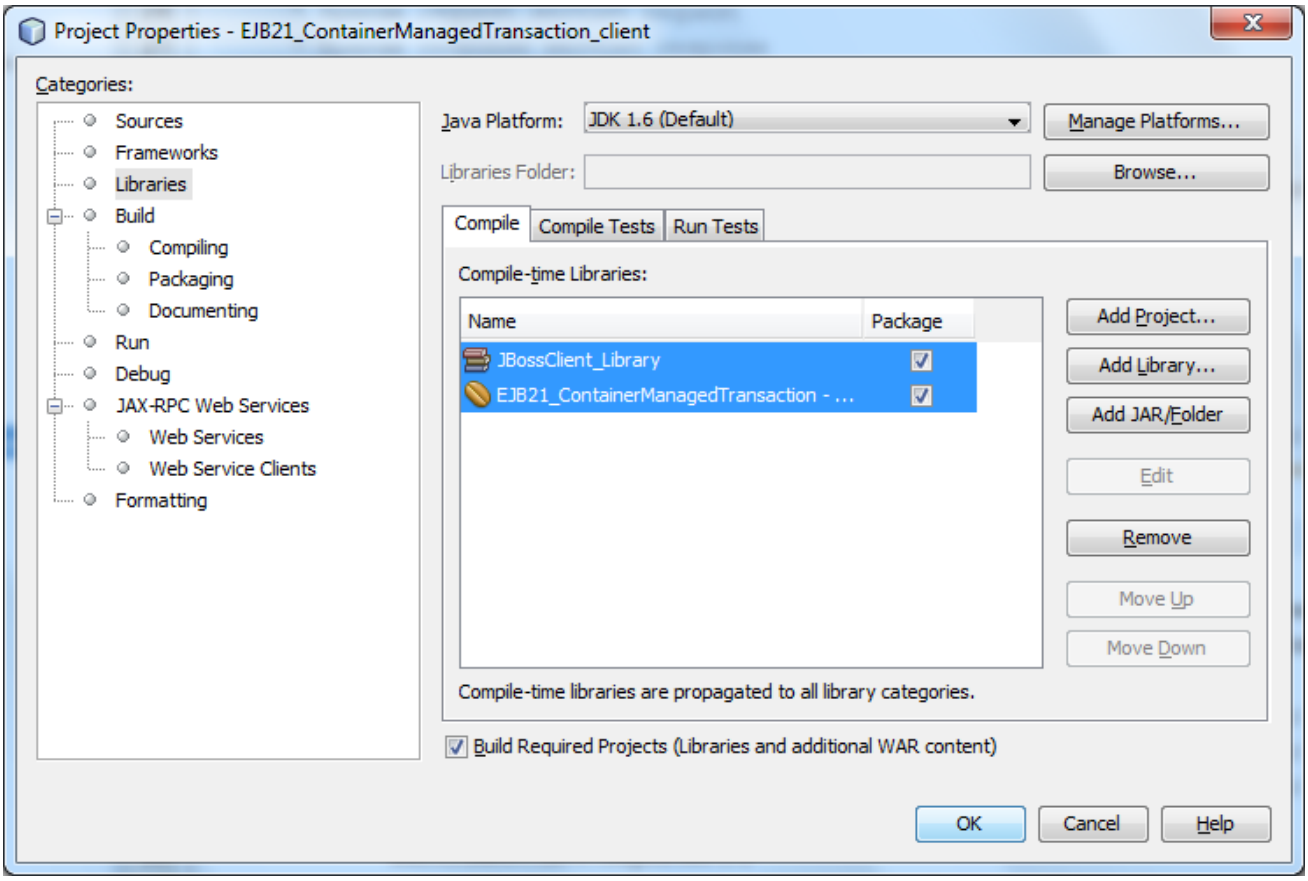


Đặt tên: EJB21_ContainerManagedTransaction_client





Tham chiếu đến ejb project và thư viện jboss client

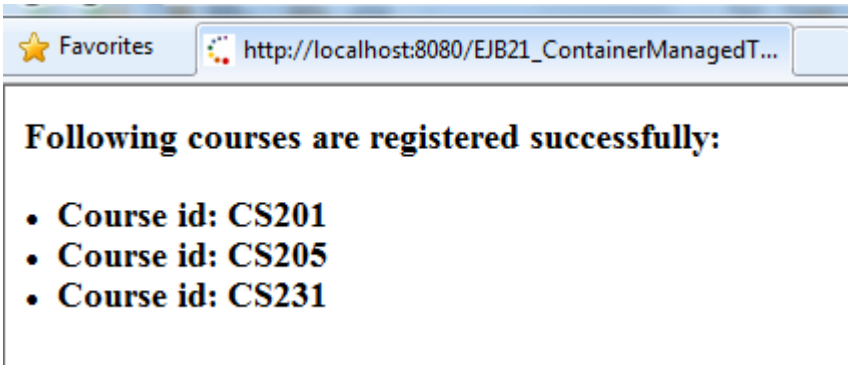


Thêm vào 1 servlet có tên CMTServlet. Code cho phần xử lý như sau:

```
…
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        System.setProperty("java.naming.factory.initial",
                "org.jnp.interfaces.NamingContextFactory");
```

```
            System.setProperty("java.naming.provider.url", "localhost:1099");
            System.setProperty("java.naming.factory.url.pkgs", "org.jboss.naming");

            Context initialContext = new InitialContext();
            Object object = initialContext.lookup("cmt/EnrollmentCartCMT");
            EnrollmentCartHome enrollmentCartHome = (EnrollmentCartHome)
javax.rmi.PortableRemoteObject.narrow(object, EnrollmentCartHome.class);
            EnrollmentCart enrollmentCart = (EnrollmentCart) enrollmentCartHome.create();

            // Add enrollment courses
            enrollmentCart.addEnrollment(new EnrollmentInfo(1, 15, "CS201"));
            enrollmentCart.addEnrollment(new EnrollmentInfo(2, 15, "CS205"));
            enrollmentCart.addEnrollment(new EnrollmentInfo(3, 15, "CS231"));
            // register the courses
            enrollmentCart.register();
            out.print("<h3>Following courses are registered successfully:<h3>");
            Collection coll = enrollmentCart.getEnrollments();
            for (Iterator i = coll.iterator(); i.hasNext();) {
                EnrollmentInfo enroll = (EnrollmentInfo) i.next();
                out.print("<li>Course id: " + enroll.getCourseId());
            }
            enrollmentCart.remove();
        } catch (Exception ex) {
            out.println(ex.getMessage());
        } finally {
            out.close();
        }
    } …
```

Thực thi servlet, kết quả như sau :



Kết quả trong Jboss console

Database trước khi chạy

Table - dbo.COURSES

COURSE_ID	COURSE_NAME	FEE	MAX_LIMIT	CURR_ENROLL
CS201	INTRODUCTION TO COMPUTERS PROGRAMMING	100	25	11
CS205	WORD PROCESSING	200	25	7
CS231	DATABASE DESIGN	300	25	7
CS105	XML PRIMER	250	20	0
CS120	HTML AND WWW APPLICATION DESIGN	200	30	0
CS305	ALGORITHMS	100	20	0
CS405	AUTOMATA THEORY AND FORMAL COMPUTATION	100	15	0
CS400	COMPUTER SYSTEMS: ARCHITECTURE, DESIGN	100	15	0

Sau khi chạy :

Table - dbo.COURSES

COURSE_ID	COURSE_NAME	FEE	MAX_LIMIT	CURR_ENROLL
CS201	INTRODUCTION TO COMPUTERS PROGRAMMING	100	25	12
CS205	WORD PROCESSING	200	25	8
CS231	DATABASE DESIGN	300	25	8
CS105	XML PRIMER	250	20	0
CS120	HTML AND WWW APPLICATION DESIGN	200	30	0
CS305	ALGORITHMS	100	20	0
CS405	AUTOMATA THEORY AND FORMAL COMPUTATION	100	15	0
CS400	COMPUTER SYSTEMS: ARCHITECTURE, DESIGN	100	15	0

Thay đổi thông tin về CURR_ENROLL như hình sau đó thực thi client

COURSE_ID	COURSE_NAME	FEE	MAX_LIMIT	CURR_ENROLL
CS201	INTRODUCTION...	100	25	25
CS205	WORD PROCES...	200	25	9
CS231	DATABASE DESIGN	300	25	9
CS305	ALGORITHMS	100	20	0
CS400	COMPUTER SYS...	100	15	0
CS405	AUTOMATA THE...	100	15	0
CS105	XML PRIMER	250	20	0
CS120	HTML AND WW...	200	30	0

Số lượng CURR_ENROLL bằng với MAX_LIMIT nên không thể cập nhật được. Điều đó làm cho việc cập nhật 2 mẫu tin kia không xảy ra (transaction).