MyuserAppClient.java

```
package edjee;

import entity.MyuserDTO;
import java.util.ArrayList;
import java.util.Scanner;
import javax.ejb.EJB;
import session.MyuserFacadeRemote;


public class MyuserAppClient {

   @EJB
   private static MyuserFacadeRemote myuserFacade;

   public MyuserAppClient() {
   }

   public static void main(String[] args) {
      MyuserAppClient client = new MyuserAppClient();
      // assuming inputs from keyboard or any GUI
      System.out.println("Welcome to the database");
      while (true) {
         System.out.println("1. Check current record\n"
         + "2. Create a record\n"
         + "3. Update a user\n"
         + "4. Delete a record\n"
         + "5. Get record by address\n"
         + "6. Exit");
         Scanner myObj = new Scanner(System.in);
         int userOption = Integer.parseInt(myObj.nextLine());

         if (userOption == 1) {
            Scanner myObj2 = new Scanner(System.in);
            System.out.println("Enter User ID");
            String userID = myObj2.nextLine();

            MyuserDTO user = client.getRecord(userID);
            System.out.println("Name: " + user.getName() + "\n"
            + "Password: " + user.getPassword() + "\n"
            + "Email: " + user.getEmail() + "\n"
            + "Phone: " + user.getPhone() + "\n"
            + "Address: " + user.getAddress() + "\n"
            + "SecQn: " + user.getSecQn() + "\n"
            + "SecAns: " + user.getSecAns() + "\n"
```

```
            );

        } else if (userOption == 2) {
            Scanner s = new Scanner(System.in);

            System.out.println("Enter UserID");
            String userID = s.nextLine();

            System.out.println("Name: ");
            String name = s.nextLine();

            System.out.println("Password: ");
            String password = s.nextLine();

            System.out.println("Email: ");
            String email = s.nextLine();

            System.out.println("Phone: ");
            String phone = s.nextLine();

            System.out.println("Address: ");
            String address = s.nextLine();

            System.out.println("Secqn:");
            String secqn = s.nextLine();

            System.out.println("Secans:");
            String secans = s.nextLine();

            MyuserDTO newUser = new MyuserDTO(userID, name, password, email, phone,
address, secqn, secans);
            client.createRecord(newUser);
        } else if (userOption == 3) {
            Scanner s = new Scanner(System.in);

            System.out.println("Enter UserID to Update");
            String userID = s.nextLine();

            System.out.println("Name: ");
            String name = s.nextLine();

            System.out.println("Password: ");
            String password = s.nextLine();

            System.out.println("Email: ");
            String email = s.nextLine();
```

```
            System.out.println("Phone: ");
            String phone = s.nextLine();

            System.out.println("Address: ");
            String address = s.nextLine();

            System.out.println("Secqn:");
            String secqn = s.nextLine();

            System.out.println("Secans:");
            String secans = s.nextLine();

            MyuserDTO updatedUser = new MyuserDTO(userID, name, password, email,
phone, address, secqn, secans);
            client.updateRecord(updatedUser);
         } else if (userOption == 4) {

            Scanner s = new Scanner(System.in);
            System.out.println("Enter UserID to Delete");
            String userID = s.nextLine();
            client.deleteRecord(userID);

         } else if (userOption == 5) {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter address");
            String address = s.nextLine();
            ArrayList<MyuserDTO> myuserList = client.getRecordsByAddress(address);

            for (int i = 0; i < myuserList.size(); i++) {
               System.out.println(myuserList.get(i).getName());
            }
         } else if (userOption == 6) {
            System.exit(1);
         } else {
            System.out.println("Invalid Selection");
         }
      }
   }

   public void showCreateResult(boolean result, MyuserDTO myuserDTO) {
      if (result) {
         System.out.println("Record with primary key " + myuserDTO.getUserid() + " has been
created in the database table.");
      } else {
         System.out.println("Record with primary key " + myuserDTO.getUserid()
         + " could not be created in the database table!");
      }
```

```
    }

    public Boolean createRecord(MyuserDTO myuserDTO) {
        return myuserFacade.createRecord(myuserDTO);
    }

    public MyuserDTO getRecord(String userId) {
        return myuserFacade.getRecord(userId);
    }

    public boolean updateRecord(MyuserDTO myuserDTO) {
        System.out.println("Updating record");
        return myuserFacade.updateRecord(myuserDTO);
    }

    public boolean deleteRecord(String userId) {
        return myuserFacade.deleteRecord(userId);
    }

    public ArrayList<MyuserDTO> getRecordsByAddress(String address) {
        return myuserFacade.getRecordsByAddress(address);
    }
}
```

Myuser.java

```
package enity;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@Table(name = "MYUSER")
@XmlRootElement
@NamedQueries({
```

```
    @NamedQuery(name = "Myuser.findAll", query = "SELECT m FROM Myuser m")
    , @NamedQuery(name = "Myuser.findByUserid", query = "SELECT m FROM Myuser m
WHERE m.userid = :userid")
    , @NamedQuery(name = "Myuser.findByName", query = "SELECT m FROM Myuser m
WHERE m.name = :name")
    , @NamedQuery(name = "Myuser.findByPassword", query = "SELECT m FROM Myuser m
WHERE m.password = :password")
    , @NamedQuery(name = "Myuser.findByEmail", query = "SELECT m FROM Myuser m
WHERE m.email = :email")
    , @NamedQuery(name = "Myuser.findByPhone", query = "SELECT m FROM Myuser m
WHERE m.phone = :phone")
    , @NamedQuery(name = "Myuser.findByAddress", query = "SELECT m FROM Myuser m
WHERE m.address = :address")
    , @NamedQuery(name = "Myuser.findBySecqn", query = "SELECT m FROM Myuser m
WHERE m.secqn = :secqn")
    , @NamedQuery(name = "Myuser.findBySecans", query = "SELECT m FROM Myuser m
WHERE m.secans = :secans")})
public class Myuser implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 6)
    @Column(name = "USERID")
    private String userid;
    @Size(max = 30)
    @Column(name = "NAME")
    private String name;
    @Size(max = 6)
    @Column(name = "PASSWORD")
    private String password;
    @Size(max = 30)
    @Column(name = "EMAIL")
    private String email;
    @Size(max = 10)
    @Column(name = "PHONE")
    private String phone;
    @Size(max = 60)
    @Column(name = "ADDRESS")
    private String address;
    @Size(max = 60)
    @Column(name = "SECQN")
    private String secqn;
    @Size(max = 60)
    @Column(name = "SECANS")
    private String secans;
```

```
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (userid != null ? userid.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Myuser)) {
            return false;
        }
        Myuser other = (Myuser) object;
        if ((this.userid == null && other.userid != null) || (this.userid != null &&
!this.userid.equals(other.userid))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "entiy.Myuser[ userid=" + userid + " ]";
    }

}
```

MyuserFacade.java

```
package session;

import entity.MyuserDTO;
import enity.Myuser;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
```

```
@Stateless
public class MyuserFacade implements MyuserFacadeRemote {


    @PersistenceContext(unitName = "ED-JEE-DTO-ejbPU")
    private EntityManager em;

    protected EntityManager getEntityManager() {
        return em;
    }

    private void create(Myuser myuser) {
        em.persist(myuser);
    }

    private void edit(Myuser myuser) {
        em.merge(myuser);
    }

    private void remove(Myuser myuser) {
        em.remove(em.merge(myuser));
    }

    private Myuser find(Object id) {
        return em.find(Myuser.class, id);
    }

    public boolean createRecord(MyuserDTO myuserDTO) {

        if (find(myuserDTO.getUserid()) != null) {
            System.out.println("User with ID already exist");
            return false;
        }

        try {
            Myuser myuser = this.myDTO2DAO(myuserDTO);
            this.create(myuser);
            return true;
        } catch (Exception e) {
            System.out.println("Create user failed");
            return false;
        }
    }
```

```
private Myuser myDTO2DAO(MyuserDTO myuserDTO) {
    Myuser myuser = new Myuser();
    myuser.setUserid(myuserDTO.getUserid());
    myuser.setName(myuserDTO.getName());
    myuser.setPassword(myuserDTO.getPassword());
    myuser.setEmail(myuserDTO.getEmail());
    myuser.setPhone(myuserDTO.getPhone());
    myuser.setAddress(myuserDTO.getAddress());
    myuser.setSecqn(myuserDTO.getSecQn());
    myuser.setSecans(myuserDTO.getSecAns());
    return myuser;
}

 private MyuserDTO myDAO2DTO(Myuser myuser) {
    MyuserDTO myuserDTO = new MyuserDTO(myuser.getUserid(), myuser.getName(),
    myuser.getPassword(), myuser.getEmail(), myuser.getPhone(),
    myuser.getAddress(), myuser.getSecqn(), myuser.getSecans());
    return myuserDTO;
}

 public MyuserDTO getRecord(String userId) {
    System.out.println("Getting record");
    Myuser myuser = this.find(userId);
    if (myuser == null) {
       System.out.println("User is null");
       return null;
    }
    MyuserDTO myuserDTO = myDAO2DTO(myuser);

    return myuserDTO;
}

 public boolean updateRecord(MyuserDTO myuserDTO) {
    if (find(myuserDTO.getUserid()) == null) {
       System.out.println("Invalid user ID - non-existent");
       return false;
    }
    try {
       Myuser myuser = this.myDTO2DAO(myuserDTO);
       System.out.println("Update user in process");
       this.edit(myuser);
       return true;
    } catch (Exception e) {
       System.out.println("Error, user could not be updated!" + e);
       return false;
    }
}
```

```
    public boolean deleteRecord(String userId) {
        MyuserDTO myuserDTO = this.getRecord(userId);
        if (myuserDTO == null) {
            return false;
        }
        Myuser myuser = myDTO2DAO(myuserDTO);

        this.remove(myuser);
        return true;
    }

    public ArrayList<MyuserDTO> getRecordsByAddress(String address) {
        Query myQuery =
em.createNamedQuery("Myuser.findByAddress").setParameter("address", address);
        List<Myuser> daoList = myQuery.getResultList();
        ArrayList<MyuserDTO> dtolist = new ArrayList<>();

for (int i = 0; i < daoList.size(); i++) {
            System.out.println(daoList.get(i));
            MyuserDTO myuserdto = myDAO2DTO(daoList.get(i));
            dtolist.add(myuserdto);
        }
        return dtolist;
    }
}
```

MyuserFacadeRemote.java

```
package session;

import entity.MyuserDTO;
import java.util.ArrayList;
import javax.ejb.Remote;

@Remote
public interface MyuserFacadeRemote {
    boolean createRecord(MyuserDTO myuserDTO);

    MyuserDTO getRecord(String userid);

    boolean updateRecord(MyuserDTO myuserDTO);

    boolean deleteRecord(String userId);

    ArrayList<MyuserDTO> getRecordsByAddress(String address);
}
```

OUTPUT

```
Welcome to the database
1. Check current record
2. Create a record
3. Update a user
4. Delete a record
5. Get record by address
6. Exit
1
Enter User ID
000001
Name: PeterSmith
Password: 123456
Email: psmith@swin.edu.au
Phone: 9876543210
Address: Swinburne EN510f
SecQn: What is my name?
SecAns: Peter

1. Check current record
2. Create a record
3. Update a user
4. Delete a record
5. Get record by address
6. Exit
5
Enter address
Swinburne EN510f
PeterSmith
1. Check current record
2. Create a record
3. Update a user
4. Delete a record
5. Get record by address
6. Exit
6
Java Result: 1
run:
BUILD SUCCESSFUL (total time: 50 seconds)
```

4.1. MyUser.java is responsible for doing ORM, because it hides SQL queries from OO logic, together with design implementation (i.e. public constructors).

4.2.

Using MyUserFacade have several advantages

. By having one bean per instance, you're guaranteed to be threads safe.

. Reduce any potential startup time that a bean might have. While Session Beans are "stateless", they only need to be stateless with regards to the client. By pooling beans you reduce the lookups to only happening when the bean is created.

. Use bean pool as a means to throttle traffic.