# **Design and plans**

#### 1/- Database

- Create database and tables (accounts, category, products)
- Insert into data for testing

#### 2/- Model and ADO

- 2.1/- DTO => Entity class (account, category, product)
- 2.2/- Interface: Accessible with common methods needed
- 2.3/- Data Access Object: => define DAO for all entities class implements Accessible (AccountDAO, CategoryDAO, ProductDAO)

### Example

#### **Entity class**

```
package model;
   □ import java.sql.Date;
 8
 10 <sup>±</sup> /** Account DTO class ...16 lines */
     public class Account {
 27
         private String account;
 28
         private String pass;
 29
         private String lastName;
 30
         private String firstName;
 31
         private Date birthday;
 32
         private boolean gender;
 33
         private String phone;
 34
         private boolean isUse;
 35
         private int roleInSystem;
36
         /** Default constructor ...3 lines */
39 ⊞
         public Account() {...4 lines }
43
         /** Constructor with attributes ...12 lines */
 55
         public Account (String account, String pass, String lastName, String firstName,
 56
                        Date birthday, boolean gender, String phone, boolean isUse,
57 ±
                        int roleInSystem) {...11 lines }
68
         public String getAccount() {...3 lines }
71 +
         public void setAccount(String account) {...3 lines }
74
         public String getPass() {...3 lines }
77 ±
         public void setPass(String pass) {...3 lines }
80 +
         public String getLastName() {...3 lines }
83 🖽
         public void setLastName(String lastName) {...3 lines }
86 +
         public String getFirstName() {...3 lines }
89 🖽
         public void setFirstName(String firstName) {...3 lines }
92 +
         public Date getBirthday() {...3 lines }
95 ±
         public void setBirthday(Date birthday) {...3 lines }
98 +
         /** True: male | False: female ...4 lines */
102
         public boolean isGender() {...3 lines }
105
         public void setGender(boolean gender) {...3 lines }
108
         public String getPhone() {...3 lines }
111
         /** Only digits, begin with 03|05|07|08|09 ...4 lines */
115 🖽
         public void setPhone(String phone) [{...3 lines }
118
         /** True: being used | False: is prevented ...4 lines */
122 +
         public boolean isIsUse() {...3 lines }
125
         public void setIsUse(boolean isUse) |{...3 lines }
128
         /** Role in system {1: admin - others: staff} ...4 lines */
132 🖽
         public int getRoleInSystem() {...3 lines }
135 ±
         /** Role in system {1: admin - others: staff} ...4 lines */
139
         public void setRoleInSystem(int roleInSystem) {...3 lines }
142
```

```
package model;
 8 <sup>⊕</sup> /** Category DTO class ...10 lines */
18
    public class Category {
        private int typeId;
19
20
        private String categoryName;
21
        private String memo;
22
        /** Default constructor ...3 lines */
25
        public Category() {...3 lines }
28
         /** Constructor with attributes ...6 lines */
34
        public Category(int typeId, String categoryName, String memo) {...5 lines }
39
40
        public int getTypeId() {...3 lines }
43
44
        public void setTypeId(int typeId) {...3 lines }
47
48
        public String getCategoryName() {...3 lines }
51
52
        public void setCategoryName(String categoryName) {...3 lines }
55
56 <sup>±</sup>
        public String getMemo() {...3 lines }
59
60 <sup>±</sup>
        public void setMemo(String memo) {...3 lines }
63 }
  6
      package model;
  7
  8 import java.sql.Date;
 10 + /** Product DTO class ...17 lines */
 27
      public class Product {
          private String productId;
 28
 29
          private String productName;
 30
          private String productImage;
 31
          private String brief;
 32
          private Date postedDate;
 33
          private Category type;
 34
          private Account account;
          private String unit;
 35
 36
          private int price;
 37
          private int discount;
 38 🖽
          /** Default constructor ...3 lines */
 41
          public Product() {...4 lines }
 45 <sup>±</sup>
          /** Constructor with attributes ...13 lines */
 58
          public Product(String productId, String productName, String productImage,
 59
                          String brief, Date postedDate, Category type, Account account,
 60 <sup>±</sup>
                          String unit, int price, int discount) {...12 lines }
 72 +
          public String getProductId() {...3 lines }
 75 ±
          public void setProductId(String productId) {...3 lines }
 78 <sup>±</sup>
          public String getProductName() {...3 lines }
 81 ±
          public void setProductName(String productName) {...3 lines }
 84 +
          public String getProductImage() {...3 lines }
 87 \pm
          public void setProductImage(String productImage) {...3 lines }
 90 #
          public String getBrief() {...3 lines }
 93 🛨
          public void setBrief(String brief) {...3 lines }
 96 <sup>±</sup>
          public Date getPostedDate() [{...3 lines }
 99 🖽
          public void setPostedDate(Date postedDate) [{...3 lines }]
 102
          public Category getType() {...3 lines }
105 🖽
          public void setType(Category type) {...3 lines }
108
          public Account getAccount() {...3 lines }
111
          public void setAccount(Account account) {...3 lines }
114
          public String getUnit() {...3 lines }
117
          public void setUnit(String unit) {...3 lines }
120 🖽
          public int getPrice() {...3 lines }
123
          public void setPrice(int price) {...3 lines }
126
          public int getDiscount() [{...3 lines }]
129
          public void setDiscount(int discount) [{...3 lines }]
132
```

### 2.2/- Interface

```
package model.dao:
8 = import java.util.List;
10 🖣 /**
     * Common methods used to perform business actions for Data Access Object
11
12
     * (Using generic type for all of DAO classes)
13
   * @author Huy Nguyễn Mai
*/
14
    public interface Accessible<T> {
       int insertRec(T obj);
        int updateRec(T obj);
        int deleteRec(T obj);
        T getObjectById(String id);
        List<T> listAll();
21
```

#### 2.3/- DAO classes

```
package model.dao;
  8 7 import java.sql.Connection;
     import java.sql.PreparedStatement;
import java.sql.ResultSet;
  9
 10
     import java.sql.SQLException;
 11
     import java.sql.Statement;
 12
 13
     import java.util.ArrayList;
     import java.util.List;
 14
 15
      import java.util.logging.Level;
     import java.util.logging.Logger;
 16
     import javax.servlet.ServletContext;
 17
 18
     import model.Account;
     import utilities.ConnectDB;
 19
 20
 21 ± /** accounts( ...14 lines */
 35
     public class AccountDAO implements Accessible<Account>{
         private ServletContext sc;
         private Connection con;
 38 +
          /** Default constructor ...4 lines */
  <u>Q</u> +
          public AccountDAO() throws ClassNotFoundException, SQLException {...4 lines }
          /** Constructor to update ServletConfig object ...4 lines */
 46 ±
 50
          public AccountDAO(ServletContext sc)
  <u>Q.</u> +
                            throws ClassNotFoundException, SQLException {...4 lines }
 55 ±
          /**...7 lines */
 62
          private Connection getConnect(ServletContext sc)
 63 ±
                                  throws ClassNotFoundException, SQLException {...3 lines }
 66 ±
          /** Add a new account to database ...5 lines */
 71
          @Override
 a +
          public int insertRec(Account o) {...27 lines }
 99 🖽
          /** Update information of an account ...5 lines */
104
          @Override
 (E)
          public int updateRec(Account o) {...29 lines }
134 ±
          /** Remove an account from database ...5 lines */
139
          @Override
 a +
          public int deleteRec(Account o) {...19 lines }
159 \pm
          /** Get all account by Role in system ...5 lines */
164 🖽
          public List<Account> listByRole(int role) {...35 lines }
199 ⊞
          /** Get all accounts from database ...4 lines */
203
          @Override
 (1)
          public List<Account> listAll() {...34 lines }
238 ±
          /** Get an account object by Account name ...5 lines */
243
          @Override
 a +
          public Account getObjectById(String id) {...33 lines }
277 ±
          /** Update the status of isUsed field of the account ...6 lines */
283 ⊞
          public int updateIsUsed(String acc, boolean isUsed) {...21 lines }
304 ⊞
          /** Check the login information ...6 lines */
310 =
          public Account loginSuccess(String acc, String pass) {...4 lines }
314
```

```
package model.dao;
   pimport java.sql.Connection;
     import java.sql.PreparedStatement;
     import java.sql.ResultSet;
10
    import java.sql.SQLException;
11
    import java.sql.Statement;
 12
    import java.util.ArrayList;
13
     import java.util.List;
 14
     import java.util.logging.Level;
 15
 16
     import java.util.logging.Logger;
 17
     import javax.servlet.ServletContext;
 18
     import model.Category;
 19
     import utilities.ConnectDB;
 20
 21 E /**...4 lines */
 25
     public class CategoryDAO implements Accessible<Category>{
         private ServletContext sc;
         private Connection con;
28 ⊞
         /** Default constructor ...3 lines */
         public CategoryDAO() throws SQLException, ClassNotFoundException { ... 4 lines }
35 ⊞
         /** Constructor to update ServletConfig object ...4 lines */
 39
         public CategoryDAO(ServletContext sc)
                            throws ClassNotFoundException, SQLException {...4 lines }
44
         /** Return connection with default parameter or from web descriptor ...7 lines */
         private Connection getConnect(ServletContext sc)
 51
52
                                  throws ClassNotFoundException, SQLException {...3 lines }
55 <sup>±</sup>
         /** Add a new category to database ...5 lines */
 60
         @Override
 (E)
         public int insertRec(Category o) {...23 lines }
84
         /** Update information of a category ...5 lines */
89
         @Override
         public int updateRec(Category o) {...23 lines }
113
         /** Remove a category from database ...5 lines */
118
         @Override
         public int deleteRec(Category o) {...19 lines }
138
         /** Get all categories from database ...4 lines */
         public List<Category> listAll() {...26 lines }
169
         /** Get a category object by Id ...5 lines */
         @Override
174
a +
         public Category getObjectById(String id) {...25 lines }
200 }
```

```
6
     package model.dao;
  8 Fimport java.sql.Connection;
     import java.sql.PreparedStatement;
 10
     import java.sql.ResultSet;
 11
     import java.sql.SQLException;
 12
     import java.sql.Statement;
     import java.util.ArrayList;
 13
 14
     import java.util.List;
     import java.util.logging.Level;
 15
 16
     import java.util.logging.Logger;
 17
     import javax.servlet.ServletContext;
     import model.Account;
 18
     import model.Category;
 19
 20
      import model.Product;
     import utilities.ConnectDB;
 21
 23 (** products( ...15 lines */
 38
     public class ProductDAO implements Accessible<Product>{
  <u>@</u>
          private ServletContext sc;
          private Connection con;
 41
         /** Default constructor ...3 lines */
  <u>Q.</u> ⊕
          public ProductDAO() throws ClassNotFoundException, SQLException {...4 lines }
 48
          /** Constructor to update ServletConfig object ...4 lines */
          public ProductDAO(ServletContext sc)
 52
  <u>Q</u> +
                            throws ClassNotFoundException, SQLException {...4 lines }
 57 <sup>±</sup>
          /** Return connection with default parameter or from web descriptor ...7 lines */
 64
          private Connection getConnect(ServletContext sc)
 65 ⊞
                               throws ClassNotFoundException, SQLException {...3 lines }
 68 <sup>±</sup>
          /** Add a new product to database ...5 lines */
 73
          Moverride
  (
          public int insertRec(Product o) {...30 lines }
104
          /** Update information of a product ...5 lines */
 109
          @Override
  a #
          public int updateRec(Product o) {...30 lines }
140
          /** Remove product from database ...5 lines */
 145
          @Override
  a +
          public int deleteRec(Product o) {...18 lines }
164
          /** Get all products by categoryId ...5 lines */
169
          public List<Product> listByCategory(int categoryId) [{...43 lines }]
212
          /** Get all products from database ...4 lines */
216
          @Override
  3 +
          public List<Product> listAll() {...42 lines }
 259 ⊞
          /** Get a product object by Id ...5 lines */
 264
          @Override
 (
          public Product getObjectById(String id) {...41 lines }
306
```

### 2.4/- ConnectDB class and Context parameters to configure the connection string

Web descriptor (web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
       web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="htt

servlet>
          <context-param>
          <param-name>hostAddress</param-na</pre>
             <param-value>localhost</param-value>
14
15
16
         </context-param>
         17
18
19
         <context-param>
20
21
22
         <param-name>dbName</param-name>
<param-value>ProductIntro</param-value>
         </context-param>
         <context-param>
  <param-name>userName</param-name</pre>
             <param-value>productDemo</param-value>
         </context-param>
          <param-name>userPass</param-name>
<param-value>123456</param-value>
31
         <context-param>
             <param-name>dbPort</param-name</pre>
              <param-value>1433</param-value>
         </context-param>
```

### ConnectDB with 2 constructor

```
package utilities;
 7
 8 | import java.sql.Connection;
     import java.sql.DriverManager;
 9
    import java.sql.SQLException;
10
11
     import javax.servlet.ServletContext;
12
13 * /**...4 lines */
17
    public class ConnectDB {
         private String hostName;
         private String instance;
        private String port;
        private String dbName;
        private String user;
 <u>Q.</u>
         private String pass;
24
         /** Default constructor ...3 lines */
27 🖣
         public ConnectDB() {
            this.hostName="127.0.0.1";
28
                                                                  //--- localhost
            this.instance="BODUA GROUP\\BODUAGROUP";
29
            this.port="1433";
30
31
            this.dbName="Human";
             this.user="qlHuman";
32
             this.pass="1111111";
33
34
35 ±
         /** Constructor to read some parameters from web ...4 lines */
39 ₽
         public ConnectDB(ServletContext sc) {
40
            this.hostName=sc.getInitParameter("hostAddress");
41
             this.instance=sc.getInitParameter("instance");
42
             this.dbName=sc.getInitParameter("dbName");
43
             this.port=sc.getInitParameter("dbPort");
44
             this.user=sc.getInitParameter("userName");
45
             this.pass=sc.getInitParameter("userPass");
46
47
         /** Format string using parameters to connection to SQL Server database ...4 lines */
51
         public String getURLString() {
52
             String fm = "jdbc:sqlserver://%s\\%s:%s;databaseName=%s;user=%s;password=%s;";
53
             return String.format(fm, this.hostName, this.instance.trim(),
54
                                      this.port, this.dbName, this.user, this.pass);
55
56 <sup>±</sup>
         /** Get a connection to database ...6 lines */
62 🖣
         public Connection getConnection() throws ClassNotFoundException, SQLException{
63
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
             return DriverManager.getConnection(getURLString());
64
65
66
```

### 3/- Pages structure

### Public pages

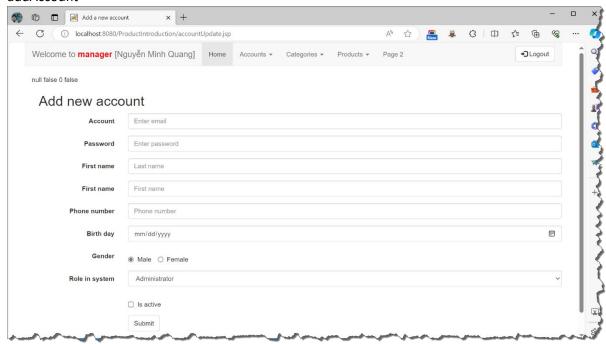
- Index.html
- product portfolio
- product details
- login

### Private pages

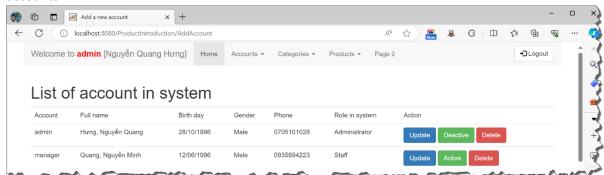
- Dashboard

\_

- addAccount



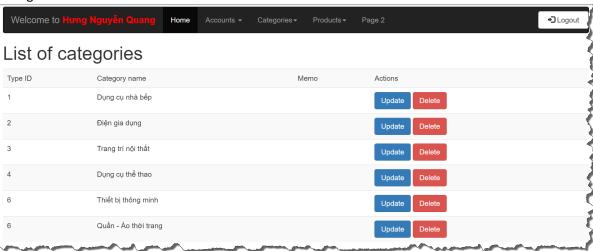
## accounts



addCategory



- Categories



- Some featurer for the Product are the same