

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

```
6 package model;
7
8 import java.sql.Date;
9
10 /** Account DTO class ...16 lines */
26 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 }
```

```

6 package model;
7
8 /** Category DTO class ...10 lines */
9
10 public class Category {
11     private int typeId;
12     private String categoryName;
13     private String memo;
14
15     /** Default constructor ...3 lines */
16     public Category() {...3 lines }
17
18     /** Constructor with attributes ...6 lines */
19     public Category(int typeId, String categoryName, String memo) {...5 lines }
20
21     public int getTypeId() {...3 lines }
22
23     public void setTypeId(int typeId) {...3 lines }
24
25     public String getCategoryName() {...3 lines }
26
27     public void setCategoryName(String categoryName) {...3 lines }
28
29     public String getMemo() {...3 lines }
30
31     public void setMemo(String memo) {...3 lines }
32 }

```

```

6 package model;
7
8 import java.sql.Date;
9
10 /** Product DTO class ...17 lines */
11
12 public class Product {
13     private String productId;
14     private String productName;
15     private String productImage;
16     private String brief;
17     private Date postedDate;
18     private Category type;
19     private Account account;
20     private String unit;
21     private int price;
22     private int discount;
23
24     /** Default constructor ...3 lines */
25     public Product() {...4 lines }
26
27     /** Constructor with attributes ...13 lines */
28     public Product(String productId, String productName, String productImage,
29         String brief, Date postedDate, Category type, Account account,
30         String unit, int price, int discount) {...12 lines }
31
32     public String getProductId() {...3 lines }
33     public void setProductId(String productId) {...3 lines }
34     public String getProductName() {...3 lines }
35     public void setProductName(String productName) {...3 lines }
36     public String getProductImage() {...3 lines }
37     public void setProductImage(String productImage) {...3 lines }
38     public String getBrief() {...3 lines }
39     public void setBrief(String brief) {...3 lines }
40     public Date getPostedDate() {...3 lines }
41     public void setPostedDate(Date postedDate) {...3 lines }
42     public Category getType() {...3 lines }
43     public void setType(Category type) {...3 lines }
44     public Account getAccount() {...3 lines }
45     public void setAccount(Account account) {...3 lines }
46     public String getUnit() {...3 lines }
47     public void setUnit(String unit) {...3 lines }
48     public int getPrice() {...3 lines }
49     public void setPrice(int price) {...3 lines }
50     public int getDiscount() {...3 lines }
51     public void setDiscount(int discount) {...3 lines }
52 }

```

2.2/- Interface

```
6 package model.dao;
7
8 import java.util.List;
9
10 /**
11  * Common methods used to perform business actions for Data Access Object
12  * {Using generic type for all of DAO classes}
13  * @author Huy Nguyễn Mai
14  */
15 public interface Accessible<T> {
16     int insertRec(T obj);
17     int updateRec(T obj);
18     int deleteRec(T obj);
19     T getObjectById(String id);
20     List<T> listAll();
21 }
```

2.3/- DAO classes

```
6 package model.dao;
7
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.util.ArrayList;
14 import java.util.List;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17 import javax.servlet.ServletContext;
18 import model.Account;
19 import utilities.ConnectDB;
20
21 /** accounts( ...14 lines */
22 public class AccountDAO implements Accessible<Account>{
23     private ServletContext sc;
24     private Connection con;
25     /** Default constructor ...4 lines */
26     public AccountDAO() throws ClassNotFoundException, SQLException{...4 lines }
27     /** Constructor to update ServletConfig object ...4 lines */
28     public AccountDAO(ServletContext sc)
29         throws ClassNotFoundException, SQLException{...4 lines }
30     /**...7 lines */
31     private Connection getConnect(ServletContext sc)
32         throws ClassNotFoundException, SQLException{...3 lines }
33     /** Add a new account to database ...5 lines */
34     @Override
35     public int insertRec(Account o) {...27 lines }
36     /** Update information of an account ...5 lines */
37     @Override
38     public int updateRec(Account o) {...29 lines }
39     /** Remove an account from database ...5 lines */
40     @Override
41     public int deleteRec(Account o) {...19 lines }
42     /** Get all account by Role in system ...5 lines */
43     public List<Account> listByRole(int role){...35 lines }
44     /** Get all accounts from database ...4 lines */
45     @Override
46     public List<Account> listAll() {...34 lines }
47     /** Get an account object by Account name ...5 lines */
48     @Override
49     public Account getObjectById(String id) {...33 lines }
50     /** Update the status of isUsed field of the account ...6 lines */
51     public int updateIsUsed(String acc, boolean isUsed){...21 lines }
52     /** Check the login information ...6 lines */
53     public Account loginSuccess(String acc, String pass){...4 lines }
54 }
```

```

6 package model.dao;
7
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.util.ArrayList;
14 import java.util.List;
15 import java.util.logging.Level;
16 import java.util.logging.Logger;
17 import javax.servlet.ServletContext;
18 import model.Category;
19 import utilities.ConnectDB;
20
21 /**...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 */
    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)
        throws ClassNotFoundException, SQLException {...3 lines }
55 /** Add a new category to database ...5 lines */
    @Override
    public int insertRec(Category o) {...23 lines }
84 /** Update information of a category ...5 lines */
    @Override
    public int updateRec(Category o) {...23 lines }
113 /** Remove a category from database ...5 lines */
    @Override
    public int deleteRec(Category o) {...19 lines }
138 /** Get all categories from database ...4 lines */
    @Override
    public List<Category> listAll() {...26 lines }
169 /** Get a category object by Id ...5 lines */
    @Override
    public Category getObjectById(String id) {...25 lines }
200 }

```

```

6   package model.dao;
7
8   import java.sql.Connection;
9   import java.sql.PreparedStatement;
10  import java.sql.ResultSet;
11  import java.sql.SQLException;
12  import java.sql.Statement;
13  import java.util.ArrayList;
14  import java.util.List;
15  import java.util.logging.Level;
16  import java.util.logging.Logger;
17  import javax.servlet.ServletContext;
18  import model.Account;
19  import model.Category;
20  import model.Product;
21  import utilities.ConnectDB;
22
23  /** products( ...15 lines */
24
25  public class ProductDAO implements Accessible<Product>{
26      private ServletContext sc;
27      private Connection con;
28
29      /** Default constructor ...3 lines */
30      public ProductDAO() throws ClassNotFoundException, SQLException{...4 lines }
31
32      /** Constructor to update ServletConfig object ...4 lines */
33      public ProductDAO(ServletContext sc)
34          throws ClassNotFoundException, SQLException{...4 lines }
35
36      /** Return connection with default parameter or from web descriptor ...7 lines */
37
38      private Connection getConnect(ServletContext sc)
39          throws ClassNotFoundException, SQLException{...3 lines }
40
41      /** Add a new product to database ...5 lines */
42      @Override
43      public int insertRec(Product o) {...30 lines }
44
45      /** Update information of a product ...5 lines */
46      @Override
47      public int updateRec(Product o) {...30 lines }
48
49      /** Remove product from database ...5 lines */
50      @Override
51      public int deleteRec(Product o) {...18 lines }
52
53      /** Get all products by categoryId ...5 lines */
54      public List<Product> listByCategory(int categoryId) {...43 lines }
55
56      /** Get all products from database ...4 lines */
57      @Override
58      public List<Product> listAll() {...42 lines }
59
60      /** Get a product object by Id ...5 lines */
61      @Override
62      public Product getObjectById(String id) {...41 lines }
63  }

```

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

Web descriptor (web.xml)

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://
3  <servlet>
4  </servlet>
5  <servlet-mapping>
6  </servlet-mapping>
7  <context-param>
8  <param-name>hostAddress</param-name>
9  <param-value>localhost</param-value>
10 </context-param>
11 <context-param>
12 <param-name>instance</param-name>
13 <param-value>BODUA_GROUP\\BODUAGROUP</param-value>
14 </context-param>
15 <context-param>
16 <param-name>dbName</param-name>
17 <param-value>ProductIntro</param-value>
18 </context-param>
19 <context-param>
20 <param-name>userName</param-name>
21 <param-value>product.Demo</param-value>
22 </context-param>
23 <context-param>
24 <param-name>userPass</param-name>
25 <param-value>123456</param-value>
26 </context-param>
27 <context-param>
28 <param-name>dbPort</param-name>
29 <param-value>1433</param-value>
30 </context-param>
31 </context-param>
32 </context-param>
33 </context-param>
34 </context-param>
35 </context-param>
36 </context-param>
37 </context-param>
38 </context-param>
39 </context-param>
40 </web-app>

```

ConnectDB with 2 constructor

```
6 package utilities;
7
8 import java.sql.Connection;
9 import java.sql.DriverManager;
10 import java.sql.SQLException;
11 import javax.servlet.ServletContext;
12
13 /**...4 lines */
14
17 public class ConnectDB {
18     private String hostName;
19     private String instance;
20     private String port;
21     private String dbName;
22     private String user;
23     private String pass;
24     /** Default constructor ...3 lines */
25     public ConnectDB() {
26         this.hostName="127.0.0.1"; //--- localhost
27         this.instance="BODUA_GROUP\\BODUAGROUP";
28         this.port="1433";
29         this.dbName="Human";
30         this.user="qlHuman";
31         this.pass="111111";
32     }
33
34     /** Constructor to read some parameters from web ...4 lines */
35     public ConnectDB(ServletContext sc) {
36         this.hostName=sc.getInitParameter("hostAddress");
37         this.instance=sc.getInitParameter("instance");
38         this.dbName=sc.getInitParameter("dbName");
39         this.port=sc.getInitParameter("dbPort");
40         this.user=sc.getInitParameter("userName");
41         this.pass=sc.getInitParameter("userPass");
42     }
43
44     /** Format string using parameters to connection to SQL Server database ...4 lines */
45     public String getURLString() {
46         String fm = "jdbc:sqlserver://%s\\%s;databaseName=%s;user=%s;password=%s;";
47         return String.format(fm, this.hostName, this.instance.trim(),
48                               this.port, this.dbName, this.user, this.pass );
49     }
50
51     /** Get a connection to database ...6 lines */
52     public Connection getConnection() throws ClassNotFoundException, SQLException{
53         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
54         return DriverManager.getConnection(getURLString());
55     }
56 }
57 }
```

3/- Pages structure

Public pages

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

Private pages

- Dashboard
-

- addAccount

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/ProductIntroduction/accountUpdate.jsp'. The page has a header with a welcome message 'Welcome to **manager** [Nguyễn Minh Quang]' and navigation links: Home, Accounts, Categories, Products, and Page 2. A 'Logout' button is in the top right. Below the header, the text 'null false 0 false' is visible. The main heading is 'Add new account'. The form contains the following fields: 'Account' (text input with placeholder 'Enter email'), 'Password' (text input with placeholder 'Enter password'), 'First name' (text input with placeholder 'Last name'), 'First name' (text input with placeholder 'First name'), 'Phone number' (text input with placeholder 'Phone number'), 'Birth day' (text input with placeholder 'mm/dd/yyyy' and a calendar icon), 'Gender' (radio buttons for 'Male' and 'Female', with 'Male' selected), and 'Role in system' (a dropdown menu currently showing 'Administrator'). At the bottom of the form, there is a checkbox labeled 'Is active' and a 'Submit' button.

accounts

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/ProductIntroduction/AddAccount'. The page has a header with a welcome message 'Welcome to **admin** [Nguyễn Quang Hưng]' and navigation links: Home, Accounts, Categories, Products, and Page 2. A 'Logout' button is in the top right. The main heading is 'List of account in system'. Below the heading is a table with the following data:

Account	Full name	Birth day	Gender	Phone	Role in system	Action
admin	Hung, Nguyễn Quang	28/10/1996	Male	0705101028	Administrator	<button>Update</button> <button>Deactive</button> <button>Delete</button>
manager	Quang, Nguyễn Minh	12/06/1996	Male	0935694223	Staff	<button>Update</button> <button>Active</button> <button>Delete</button>

- addCategory

The screenshot shows a web form titled 'New category'. It contains two main sections: 'Category name:' with a text input field containing 'Nhóm hàng mới', and 'Memo :' with a text area containing 'Những sản phẩm dùng cho du lịch, thám hiểm'. At the bottom of the form is a 'Save' button.

- Categories

Welcome to Hưng Nguyễn Quang				Home	Accounts ▾	Categories ▾	Products ▾	Page 2	Logout
List of categories									
Type ID	Category name		Memo		Actions				
1	Dụng cụ nhà bếp				Update	Delete			
2	Điện gia dụng				Update	Delete			
3	Trang trí nội thất				Update	Delete			
4	Dụng cụ thể thao				Update	Delete			
6	Thiết bị thông minh				Update	Delete			
6	Quần - Áo thời trang				Update	Delete			

- Some featurer for the Product are the same