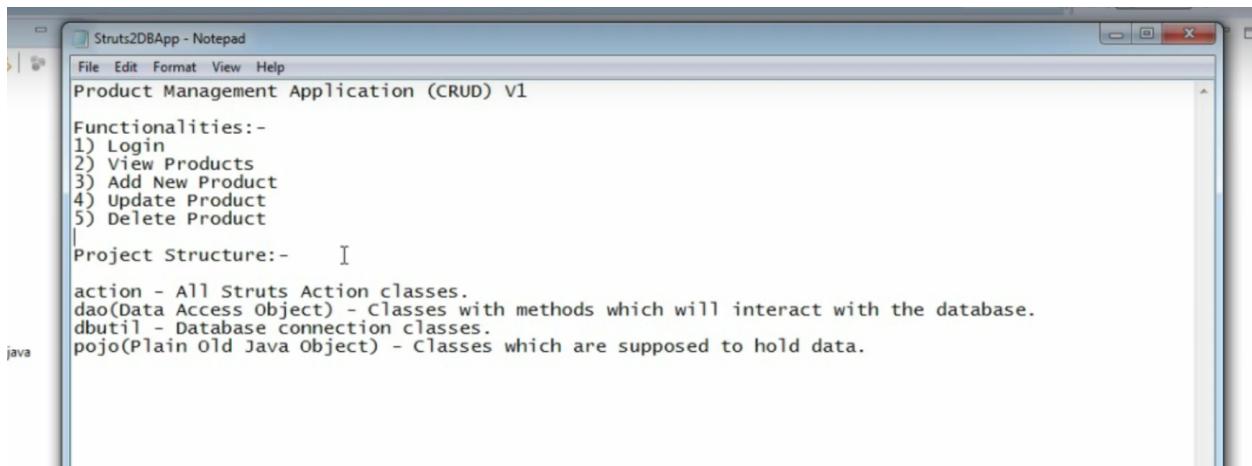
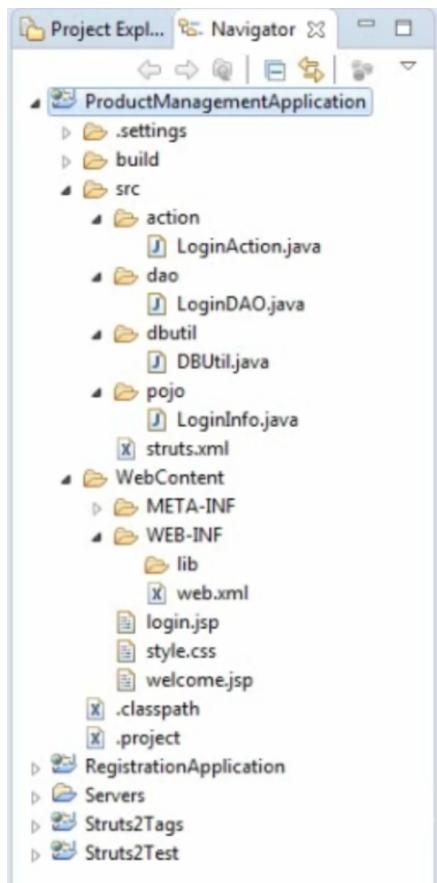


Struts 2 Database application V1 / Product Management Application



Struts2DBApp - Notepad
File Edit Format View Help
Product Management Application (CRUD) v1
Functionalities:-
1) Login
2) View Products
3) Add New Product
4) Update Product
5) Delete Product
Project Structure:-
action - All Struts Action classes.
dao(Data Access Object) - Classes with methods which will interact with the database.
dbutil - Database connection classes.
pojo(Plain Old Java Object) - Classes which are supposed to hold data.

We also have struts.xml web.xml and jsps and css as shown below.



Tables and data we require for this.

The screenshot shows the Oracle SQL Command Line interface. It displays the following SQL queries and their results:

```

Connected.
SQL> desc login_info;
Name          Null?    Type
USER_NAME          VARCHAR2(30)
PASSWORD          VARCHAR2(30)

SQL> select * from login_info;
USER_NAME      PASSWORD
abhay          abhay123

SQL> desc product;
Name          Null?    Type
PROD_ID          VARCHAR2(30)
PROD_NAME        VARCHAR2(30)
PROD_CATEGORY    VARCHAR2(30)
PROD_PRICE       NUMBER(38)

SQL> set linesize 200;
SQL> select * from product;
PROD_ID      PROD_NAME           PROD_CATEGORY      PROD_PRICE
p001         iPhone              Mobile phones       10000
p002         Sony Bravia        Television          7000
p003         Nikes Shoes         Men's Fashion     4000
SQL>

```

Db util class contains get Connections and close connections methods

The screenshot shows the code editor with the file `DBUtil.java`. The code defines a static method `getConnection()` and a static method `closeConnection(Connection conn)`.

```

1 package dbutil;
2
3 import java.sql.*;
4
5 public class DBUtil {
6
7     public static Connection getConnection()  {
8         {
9             Connection conn = null;
10            try
11            {
12                Class.forName("oracle.jdbc.driver.OracleDriver");
13                conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","system");
14            }
15            catch(Exception e)
16            {
17                e.printStackTrace();
18            }
19            return conn;
20        }
21
22
23     public static void closeConnection(Connection conn)
24     {

```

Login Pojo class - setters and getters and `toString` override

The screenshot shows a Java code editor with three tabs at the top: DBUtil.java, LoginInfo.java (which is currently selected), and another unnamed tab. The code in LoginInfo.java defines a class with private fields for user name and password, a constructor that takes these parameters, and methods to get and set the user name.

```
1 package pojo;
2
3 public class LoginInfo {
4
5     String userName;
6     String password;
7
8     public LoginInfo() {
9         // TODO Auto-generated constructor stub
10    }
11
12     public LoginInfo(String userName, String password) {
13         super();
14         this.userName = userName;
15         this.password = password;
16     }
17
18     public String getUserName() {
19         return userName;
20     }
21
22     public void setUserName(String userName) {
23         this.userName = userName;
24     }
25 }
```

Login DAO class

The screenshot shows a Java code editor with three tabs at the top: DBUtil.java, LoginInfo.java, and LoginDAO.java (which is currently selected). The code in LoginDAO.java contains a static method isUserValid that takes a LoginInfo object and returns a boolean indicating if the user is valid. It connects to a database using DBUtil.getConnection(), creates a statement, and executes a query to check if the user exists. If the user is found, it sets the validStatus to true.

```
1 package dao;
2
3 import java.sql.Connection;
4
5 public class LoginDAO {
6
7     public static boolean isUserValid(LoginInfo userDetails)
8     {
9         boolean validStatus = false;
10        try
11        {
12            Connection conn = DBUtil.getConnection();
13            Statement st= conn.createStatement();
14            ResultSet rs= st.executeQuery("SELECT * FROM login_info WHERE user_name = '"+userDetails.getUserName());
15            while(rs.next())
16            {
17                validStatus = true;
18            }
19            DBUtil.closeConnection(conn);
20
21        }
22        catch(Exception e)
23        {
24        }
25    }
26 }
```

```

23
24         }
25         DBUtil.closeConnection(conn);
26     }
27     catch(Exception e)
28     {
29         e.printStackTrace();
30     }
31     return validStatus;
32 }
33 }
34 }
35

```

Servers Data Source Explorer Snippets Console

Tomcat v8.0 Server at localhost [Stopped, Synchronized]

LoginAction class where each action class contain one execute method.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "ProductManagementApplication". It includes packages like "action", "dao", "dbutil", "pojo", and "WEB-INF". Under "WEB-INF", there are "lib", "login.jsp", "style.css", and "welcome.jsp".
- Navigator:** Shows the current file being edited is "LoginAction.java".
- Editor:** Displays the Java code for the "LoginAction" class. The code defines an "execute" method that checks if a user is valid and returns a status code ("success" or "input"). It also has a getter for "user_name".
- Bottom Status Bar:** Shows the message "Tomcat v8.0 Server at localhost [Stopped, Synchronized]".

```

package action;
import com.opensymphony.xwork2.ActionSupport;
public class LoginAction extends ActionSupport{
    String user_name;
    String password;
    public String execute() {
        String statusCode = "";
        boolean isUserValid = LoginDAO.isUserValid(new LoginInfo(user_name, password));
        if (isUserValid) {
            statusCode = "success";
        } else {
            statusCode = "input";
        }
        return statusCode;
    }
    public String getUserName() {
        return user_name;
    }
}

```

Screenshot of the Eclipse IDE showing the Struts configuration file (struts.xml) and the project structure.

Project Explorer:

- ProductManagementApplication
- .settings
- build
- src
 - action
 - LoginAction.java
 - dao
 - LoginDAO.java
 - dbutil
 - DBUtil.java
 - pojo
 - LoginInfo.java
 - struts.xml
- WebContent
 - META-INF
 - WEB-INF
 - lib
 - web.xml
 - login.jsp

struts.xml Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
"http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <package name="productManagementApp" extends="struts-default">
        <action name="LoginAction" class="action.LoginAction">
            <result name="success"/>/welcome.jsp</result>
            <result name="input"/>/login.jsp</result>
        </action>
    </package>
</struts>
```

Screenshot of the Eclipse IDE showing the JSP file (login.jsp) and the project structure.

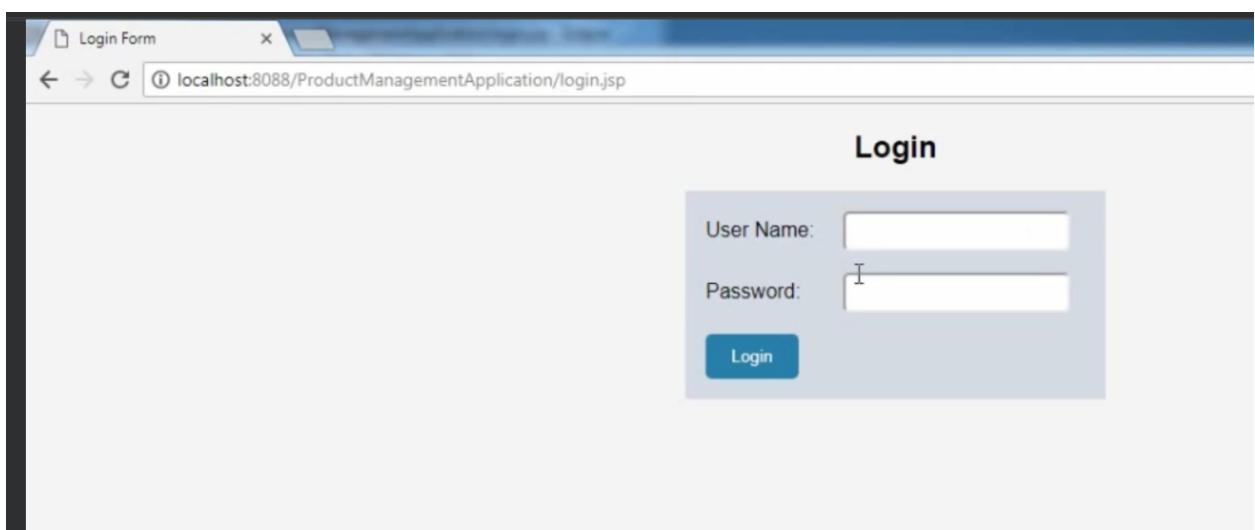
Project Explorer:

- DBUtil.java
- LoginInfo.java
- LoginDAO.java
- LoginAction.java
- struts.xml
- login.jsp

login.jsp Content:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Form</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div align="center">
    <h2>Login</h2>
    <s:form action="LoginAction" class="LoginForm">
        <s:textfield name="userName" label="User Name" class="formTextField" />
        <s:password name="password" label="Password" class="formTextField" />
        <s:submit value="Login" class="actionBtn" />
    </s:form>
</div>
</body>
</html>
```

Login.jsp



Product DAO

The screenshot shows the Eclipse IDE interface with the 'Project Explorer' and 'Navigator' toolbars at the top. The 'Project Explorer' on the left lists the 'ProductManagementApplication' project structure, including '.settings', 'build', 'src' (containing 'action', 'dao', 'dbutil', 'pojo', and 'struts.xml'), 'WebContent' (containing 'META-INF', 'WEB-INF' (with 'lib' and 'web.xml'), 'login.jsp', and 'style.css'). The 'Navigator' tab is selected. The main editor window displays the 'DBUtil.java' code:

```
10  public static List<Product> getAllProducts()
11  {
12      List<Product> productList = new ArrayList<Product>();
13      try
14      {
15          Connection conn = DBUtil.getConnection();
16          Statement st= conn.createStatement();
17          ResultSet rs= st.executeQuery("SELECT * FROM product");
18          while(rs.next())
19          {
20              Product product = new Product(rs.getString("prod_id"),rs.getString("prod_name"),r
21              productList.add(product);
22          }
23      }
24
25      DBUtil.closeConnection(conn);
26
27  }
28  catch(Exception e)
29  {
30      e.printStackTrace();
31  }
32 }
```

In Welcome action class we want to display all the products

The screenshot shows the Eclipse IDE interface with the 'Project Explorer' and 'Navigator' toolbars at the top. The 'Project Explorer' on the left lists the 'ProductManagementApplication' project structure, identical to the previous screenshot. The 'Navigator' tab is selected. The main editor window displays the 'WelcomeAction.java' code:

```
10  public class WelcomeAction extends ActionSupport{
11
12      List<Product> products;
13
14      public void initializeProducts() {
15          products = ProductManagementDAO.getAllProducts();
16      }
17
18      public String execute() {
19          initializeProducts();
20          return "success";
21      }
22
23      public List<Product> getProducts() {
24          return products;
25      }
26
27      public void setProducts(List<Product> products) {
28          this.products = products;
29      }
30 }
```

Include Welcome in the struts page.

```

4      "http://struts.apache.org/dtds/struts-2.5.dtd"
5
6<struts>
7  <package name="productManagementApp" extends="struts-default">
8    <action name="LoginAction" class="action.LoginAction">
9      <result name="success">/welcome.jsp</result>
10     <result name="input">/login.jsp</result>
11   </action>
12   <action name="welcomeAction" class="action.WelcomeAction">
13     <result name="success">/welcome.jsp</result>
14   </action>
15 </package>
16 </struts>

```

The screenshot shows an IDE interface with multiple tabs at the top: DBUtil.java, LoginInfo.java, struts.xml, login.jsp, Product.java, ProductMana..., WelcomeActi..., welcome.jsp, and a blank tab. The struts.xml file is open, displaying the configuration code above. The welcome.jsp file is also visible in the background.

```

13<div align="center">
14   <h2>Welcome</h2>
15 </div>
16
17<table width="750" class="productTable" align="center">
18   <thead>
19     <tr>
20       <th>Product ID</th>
21       <th>Product Name</th>
22       <th>Product Category</th>
23       <th>Product Price</th>
24       <th colspan="2">Actions</th>
25     </tr>
26   </thead>
27
28<s:iterator value="products" var="product">
29   <tr>
30     <td>
31       <s:property value="#product.productId"/>
32     </td>
33     <td>
34       <s:property value="#product.productName"/>
35     </td>
36     <s:action name="edit" href="editProduct.jsp?productId={#product.productId}" />
37   </tr>
38 </s:iterator>

```

On successful login let it redirect to welcomeAction

```

4      "http://struts.apache.org/dtds/struts-2.5.dtd"
5
6<struts>
7  <package name="productManagementApp" extends="struts-default">
8    <action name="LoginAction" class="action.LoginAction">
9      <result name="success" type="redirect">welcomeAction</result>
10     <result name="input">/login.jsp</result>
11   </action>
12   <action name="welcomeAction" class="action.WelcomeAction">
13     <result name="success">/welcome.jsp</result>
14   </action>
15 </package>
16 </struts>

```

The screenshot shows a web browser window with the URL udemy.com/course/struts-2-framework-for-beginners/learn/lecture/10541754#overview. The browser's address bar and tab bar are visible at the top. Below the browser window, the Udemy course navigation bar is shown, featuring links like 'Kafka Portal - Login', 'Folios', 'gpo', 'spring', 'GAP Utilities', 'stockoverflow', 'OMS', 'zoom', 'GLV', 'sessions', 'driving', and 'I'. The main content area displays the title 'Struts 2 Framework for Beginners' and a 'Welcome' message. Below the message is a table with the following data:

Product ID	Product Name	Product Category	Product Price	Actions	
p001	iPhone	Mobile phones	10000	Update	Delete
p002	Sony Bravia	Television	7000	Update	Delete
p003	Nikes Shoes	Men's Fashion	4000	Update	Delete

The screenshot shows an IDE interface with the project structure on the left and the code editor on the right. The project structure for 'ProductManagementApplication' is visible, including packages for action, dao, dbutil, pojo, and struts.xml. The code editor displays the `DBUtil.java` file, focusing on the `addProduct` and `updateProduct` methods. The `addProduct` method is shown below:

```
public static int addProduct(Product product)
{
    int status = 0;
    try
    {
        Connection conn = DBUtil.getConnection();
        PreparedStatement ps = conn.prepareStatement("INSERT INTO product VALUES(?, ?, ?, ?)");
        ps.setString(1, product.getProductId());
        ps.setString(2, product.getProductName());
        ps.setString(3, product.getProductCategory());
        ps.setInt(4, product.getProductPrice());
        status = ps.executeUpdate();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return status;
}
```

Java EE - ProductManagementApplication/src/action/AddAction.java - Eclipse

```

10 public class AddAction extends ActionSupport{
11     String productId;
12     String productName;
13     String productCategory;
14     Integer productPrice;
15
16     public String execute() {
17         String statusCode = "";
18         Product product = new Product(productId, productName, productCategory, productPrice);
19         int recordAdded = ProductManagementDAO.addProduct(product);
20         if (recordAdded == 1) {
21             statusCode = "success";
22         } else {
23             statusCode = "error";
24         }
25         return statusCode;
26     }
27
28     public String getProductId() {
29         return productId;
30     }
31 }

```

Project Explorer Navigator Servers Data Source Explorer Snippets Console

Tomcat v8.0 Server at localhost [Stopped, Synchronized]

struts.xml login.jsp WelcomeAction... welcome.jsp ProductManagementApplication AddAction.java addProduct.jsp error.jsp

```

1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1"%>
3 <%@ taglib prefix="s" uri="/struts-tags" %>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
5 <html>
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8 <title>Add New Product</title>
9 <link rel="stylesheet" href="style.css">
10 </head>
11 <body>
12
13 <div align="center">
14     <h2>Add New Product</h2>
15     <s:form action="addAction" class="formTable">
16         <s:textfield name="productId" label="Product Id" class="formTextField"/>
17         <s:textfield name="productName" label="Product Name" class="formTextField"/>
18         <s:textfield name="productCategory" label="Product Category" class="formTextField"/>
19         <s:textfield name="productPrice" label="Product Price" class="formTextField"/>
20         <s:submit value="Add Product" class="actionBtn"/>
21     </s:form>
22 </div>
23

```

Welcome

Add New Product

Product ID	Product Name	Product Category	Product Price	Actions	
p001	iPhone	Mobile phones	10000	Update	Delete

Add New Product

localhost:8088/ProductManagementApplication/addProduct.jsp

Add New Product

Product Id:	<input type="text" value="p004"/>
Product Name:	<input type="text" value="test"/>
Product Category:	<input type="text" value="test"/>
Product Price:	<input type="text" value="3444"/>
<input type="button" value="Add Product"/>	

New product added

Welcome

Add New Product

Product ID	Product Name	Product Category	Product Price	Actions	
p001	iPhone	Mobile phones	10000	Update	Delete
p002	Sony Bravia	Television	7000	Update	Delete
p003	Nikes Shoes	Men's Fashion	4000	Update	Delete
p004	test	test	3444	Update	Delete

Update product

```

42<td>
43    <s:property value="#product.productPrice"/>
44</td>
45<td>
46    <a
47        href="updateDataAction?productId=<s:property value="#product.productId"/>">
48            <button class="actionBtn">Update</button>
49        </a>
50    </td>
51<td>
52    Delete
53    </td>

```

The screenshot shows an IDE interface with the Project Explorer on the left and the code editor on the right. The code editor displays the `DBUtil.java` file, specifically the `getProductById` method. The code uses JDBC to query a database for a product by its ID.

```
    }
    return productList;
}
public static Product getProductById(String productId)
{
    Product product = null;
    try
    {
        Connection conn = DBUtil.getConnection();
        PreparedStatement ps = conn.prepareStatement("SELECT * FROM product WHERE prod_id = ?");
        ps.setString(1, productId);
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            product = new Product(rs.getString("prod_id"),rs.getString("prod_name"),rs.getString("prod_category"),
            rs.getInt("prod_price"));
        }
    } catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

Return product

The screenshot shows the `updateProduct` method from the `DBUtil.java` file. It takes a `Product` object as input and updates it in the database using JDBC.

```
    }
    public static int updateProduct(Product product)
    {
        int status = 0;
        try
        {
            Connection conn = DBUtil.getConnection();
            PreparedStatement ps= conn.prepareStatement("UPDATE product SET prod_name=?, prod_category=?, prod_p
            ps.setString(1, product.getProductName());
            ps.setString(2, product.getProductCategory());
            ps.setInt(3, product.getProductPrice());
            ps.setString(4, product.getProductId());
            status = ps.executeUpdate();
        } catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Return status either 1 or 0;

The screenshot shows the `UpdateDataAction` class, which extends `ActionSupport`. It has fields for `productId`, `productName`, `productCategory`, and `productPrice`. The `execute` method retrieves a `Product` object by its ID and returns "success". The `getProductId` method returns the `productId`.

```
9
10 public class UpdateDataAction extends ActionSupport{
11
12     String productId;
13     String productName;
14     String productCategory;
15     Integer productPrice; I
16
17     public String execute() {
18         Product product = ProductManagementDAO.getProductById(productId);
19         productId = product.getProductId();
20         productName = product.getProductName();
21         productCategory = product.getProductCategory();
22         productPrice = product.getProductPrice();
23         return "success";
24     }
25
26     public String getProductId() {
27         return productId;
28     }
29 }
```

```
<package name="productManagementApp" extends="struts-default">
    <action name="LoginAction" class="action.LoginAction">
        <result name="success" type="redirect">welcomeAction</result>
        <result name="input"/>/login.jsp</result>
    </action>
    <action name="welcomeAction" class="action.WelcomeAction">
        <result name="success"/>/welcome.jsp</result>
    </action>
    <action name="addAction" class="action.AddAction">
        <result name="success" type="redirect">welcomeAction</result>
        <result name="error"/>/error.jsp</result>
    </action>
    <action name="updateDataAction" class="action.UpdateDataAction">
        <result name="success"/>/updateProduct.jsp</result>
        <result name="error"/>/error.jsp</result>
    </action>
```

```
<%@ page contentType="text/html; charset=ISO-8859-1" %>
<%@ taglib prefix="s" uri="struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Update Product</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div align="center">
    <h2>Update Product</h2>
    [<s:form action="updateAction" class="formTable">
        <s:textfield name="productId" label="Product Id" class="formTextField" readonly="true"/>
        <s:textfield name="productName" label="Product Name" class="formTextField"/>
        <s:textfield name="productCategory" label="Product Category" class="formTextField"/>
        <s:textfield name="productPrice" label="Product Price" class="formTextField"/>
        <s:submit value="Update Product" class="actionBtn"/>
    </s:form>
</div>
</body>
</html>
```

Read only true because for that product id we are doing an update.

```
<div align="center">
    <h2>Update Product</h2>
    [<s:form action="updateAction" class="formTable">
        <s:textfield name="productId" label="Product Id" class="formTextField" readonly="true"/>
        <s:textfield name="productName" label="Product Name" class="formTextField"/>
```

Update action class

```
struts.xml login.jsp welcome.jsp ProductM... UpdateDataAc... updateProduc... Login Form UpdateAction... »3
```

```
1 package action;
2
3@import com.opensymphony.xwork2.ActionSupport;
4
5 public class UpdateAction extends ActionSupport{
6
7     String productId;
8     String productName;
9     String productCategory;
10    Integer productPrice;
11
12    public String execute() {
13        String statusCode = "";
14        Product product = new Product(productId, productName, productCategory, productPrice);
15        int recordUpdated = ProductManagementDAO.updateProduct(product);
16        if (recordUpdated == 1) {
17            statusCode = "success";
18        } else {
19            statusCode = "error";
20        }
21        return statusCode;
22    }
23
24    <!-- Struts 2 annotations -->
25    <!-- Struts 2 annotations -->
26    <!-- Struts 2 annotations -->
27    <!-- Struts 2 annotations -->
```

```
</action>
<action name="updateAction" class="action.UpdateAction">
    <result name="success" type="redirect">welcomeAction</result>
    <result name="error">/error.jsp</result>
</action>
</package>
```

updateDataAction?productId=p004

Update Product

Product Id:	p004
Product Name:	test updated
Product Category:	test updated
Product Price:	50

```

Enter user name: system
Connected.
SQL> set linesize 200
SQL> select * from product;
PROD_ID          PROD_NAME        PROD_CATEGORY      PROD_PRICE
p801             iPhone           Mobile phones       10000
p802             Sony Bravia     Television          7000
p803             Nikes Shoes     Men's Fashion      4000
p804             test             test                 3444
SQL> select * from product;
PROD_ID          PROD_NAME        PROD_CATEGORY      PROD_PRICE
p801             iPhone           Mobile phones       10000
p802             Sony Bravia     Television          7000
p803             Nikes Shoes     Men's Fashion      4000
p804             test updated    test updated         5000
SQL>

```

Delete product

Screenshot of a web browser showing a JSP page for managing products. The page displays a table of products with columns: PROD_ID, PROD_NAME, PROD_CATEGORY, and PROD_PRICE. The last row shows a product with ID p804, name 'test', category 'test', and price 3444. Below the table is a table of contents.

PROD_ID	PROD_NAME	PROD_CATEGORY	PROD_PRICE
p801	iPhone	Mobile phones	10000
p802	Sony Bravia	Television	7000
p803	Nikes Shoes	Men's Fashion	4000
p804	test	test	3444

Table of contents:

- struts.xml
- login.jsp
- welcome.jsp
- ProductMana...
- UpdateDataAc...
- updateProduc...
- DeleteAction...

```

<s:property value="#product.productName"/>
</td>
<td>
    <s:property value="#product.productCategory"/>
</td>
<td>
    <s:property value="#product.productPrice"/>
</td>
<td>
    <a href="updateDataAction?productId=<s:property value="#product.productId"/>">
        <button class="actionBtn">Update</button>
    </a>
</td>
<td>
    <a href="deleteAction?productId=<s:property value="#product.productId"/>">
        <button class="actionBtn">Delete</button>
    </a>
</td>

```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The Project Explorer on the left lists several Java packages and files, including build, src (containing action, dao, dbutil, pojo, struts.xml), WebContent (META-INF, WEB-INF, lib), and various JSP and XML files like addProduct.jsp, error.jsp, login.jsp, welcome.jsp, struts.xml, and web.xml. The Navigator view shows the current file open: LoginInfo.java. The code editor displays the following Java code:

```
    e.printStackTrace();
}
return status;
}

public static int deleteProduct(String productId)
{
    int status = 0;
    try
    {
        Connection conn = DBUtil.getConnection();
        PreparedStatement ps= conn.prepareStatement("DELETE FROM product where prod_id = ?");
        ps.setString(1, productId);
        status = ps.executeUpdate();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return status;
}
}
```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The Project Explorer on the left lists the same project structure as the previous screenshot. The Navigator view shows the current file open: DeleteAction.java. The code editor displays the following Java code:

```
1 package action;
2
3 import com.opensymphony.xwork2.ActionSupport;
4
5 public class DeleteAction extends ActionSupport{
6     String productId;
7
8     public String execute() {
9         String statusCode = "";
10        int recordDeleted = ProductManagementDAO.deleteProduct(productId);
11        if (recordDeleted == 1) {
12            statusCode = "success";
13        } else {
14            statusCode = "error";
15        }
16        return statusCode;
17    }
18
19    public String getProductId() {
20        return productId;
21    }
22
23 }
```

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The Project Explorer on the left lists the same project structure. The Navigator view shows the current file open: struts.xml. The code editor displays the following Struts configuration:

```
</action>
<action name="deleteAction" class="action.DeleteAction">
    <result name="success" type="redirect">welcomeAction</result>
    <result name="error">/error.jsp</result>
</action>
</package>
```

Welcome				
Add New Product				
Product ID	Product Name	Product Category	Product Price	Actions
p001	iPhone	Mobile phones	10000	Update Delete
p002	Sony Bravia	Television	7000	Update Delete
p003	Nikes Shoes	Men's Fashion	4000	Update Delete
p004	test updated	test updated	5000	Update Delete

SQL> select * from product;	PROD_ID	PROD_NAME	PROD_CATEGORY	PROD_PRICE
	p001	iPhone	Mobile phones	10000
	p002	Sony Bravia	Television	7000
	p003	Nikes Shoes	Men's Fashion	4000
	p004	test updated	test updated	5000
SQL> select * from product;	PROD_ID	PROD_NAME	PROD_CATEGORY	PROD_PRICE
	p001	iPhone	Mobile phones	10000
	p002	Sony Bravia	Television	7000
	p003	Nikes Shoes	Men's Fashion	4000
SQL> _				

Running the application - validation

Update Product

Product Id:	<input type="text" value="p003"/>
Product Name:	<input type="text" value="Nikes Shoes"/>
Product Category:	<input type="text" value="Men's Fashion"/>
Invalid field value for field "productPrice".	
Product Price:	<input type="text" value="sdfsf"/> I <input type="text" value="sdfsf"/>
Update Product	

Notice validation. That is because of below tag in add action and update action.

```
14      </action>
15      <action name="addAction" class="action.AddAction">
16          <result name="success" type="redirect">welcomeAction</result>
17          <result name="error">/error.jsp</result>
18          <result name="input">/addProduct.jsp</result>
19      </action>
20      <action name="updateDataAction" class="action.UpdateDataAction">
21          <result name="success">/updateProduct.jsp</result>
22          <result name="error">/error.jsp</result>
23      </action>
24      <action name="updateAction" class="action.UpdateAction">
25          <result name="success" type="redirect">welcomeAction</result>
26          <result name="error">/error.jsp</result>
27          <result name="input">/updateProduct.jsp</result>
28      </action>
```