Internet Technology project documentation

NOZHAN SOLTANPOUR, HESAM RIAZI, MAHYAR SHAVIRIAN, VALLA ZEHTAB

Dec 2008

Introduction

This system is ordered by Sunglass Persia, a sunglass shop which sells original brands of sunglass. The shop acts as a branch of those brands. The basic reason of ordering this system is to promote the shop, products and providing an online shopping as there is not a professional online shopping system in Iran which can offer standard e-services. There is not a sunglass shop in Iran which offer original brands, there for this shop is trying to succeed in the market share by offering the original brands of sunglasses. All brands could be found in the official web sites of the sunglasses such as RayBan, Prada, & Dior.
Making this system web-based is presumed to be successful in the sunglass market as the online shopping system is not practical in Iran yet. Omitting the physical transportation factor is the main reason for implementing this project online.
In order to place these products online they require a shopping basket system to be implemented which will allow their customers to select from a range of products, add them to a virtual shopping basket and check them out finally to complete order procedure. This system should be able to display product's name, model, picture and price. It also should be easy to use (simple interface) as sunglasses customers will include almost all ranges of people in terms of age and occupation. The other important feature that mentioned by the customer is to make the system scalable in order to expand some sections sequentially. Customer needs will fully specified in the Need Assessment stage in terms of priorities.

Scenario

1-Ali is a 19 years old student who is looking for a specific original model of RayBan sunglass which is not available in Iran. He will be redirected to Sunglass Persia after searching local online shops by Google. He will find his desired model by means of the search section of the site.

2- Sarah is a fashion designer who is looking for the specifications of the new models of sunglass and she doesn't have enough time to travel to city center. Therefore she will find her desired information via the search engine designed in the website without wasting her time in traffic.

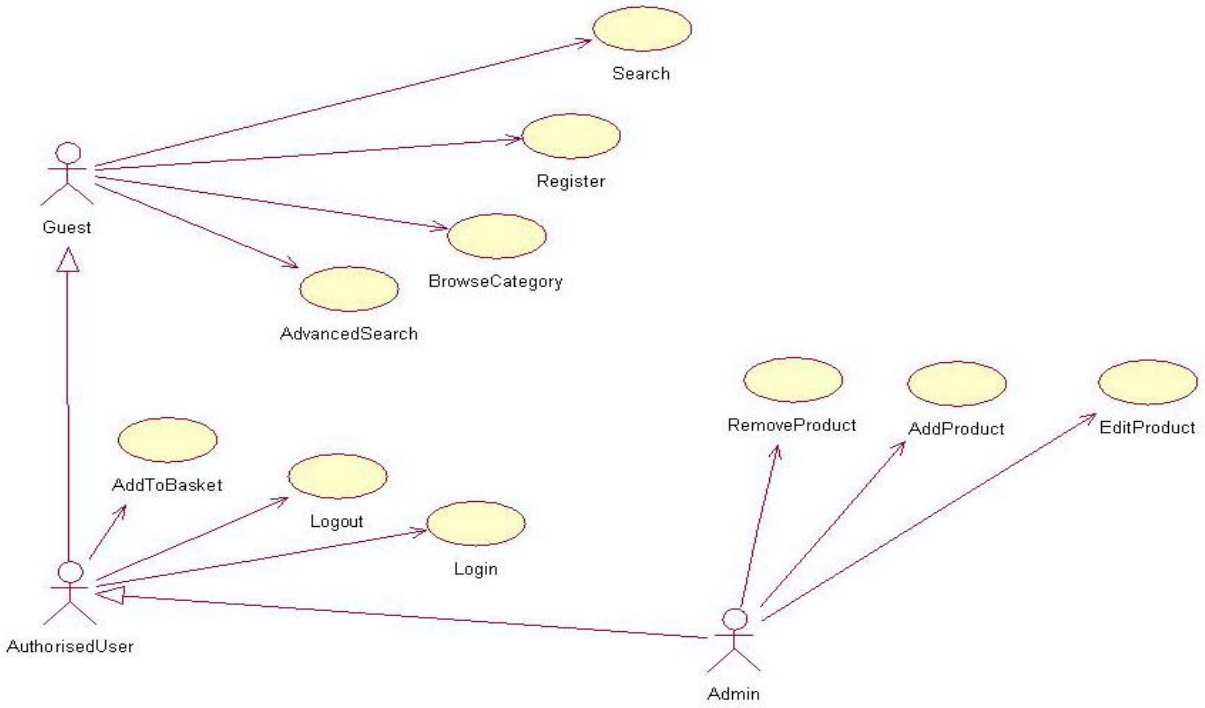
3- David wants to purchase a pair of sunglasses but is confused on his choices. There are different brands available that David can choose from but needs to compare them first. He needs to find a website on which different sunglasses from different brands are available along with their features. David could easily search along the sunglasses in order to find the desired one.

Internet Technology project documentation

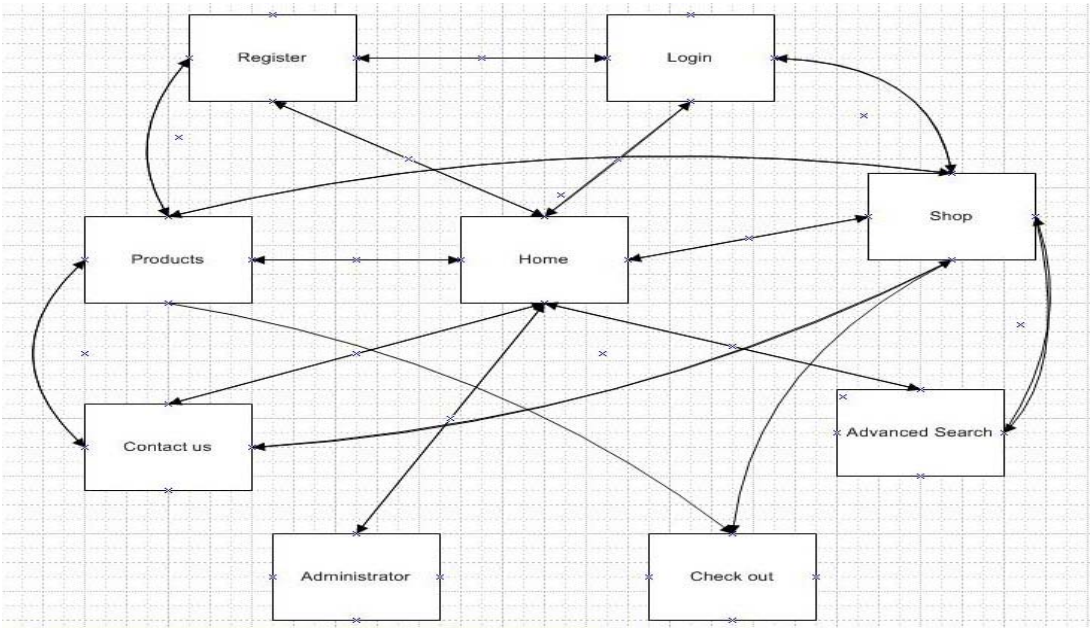
NOZHAN SOLTANPOUR, HESAM RIAZI, MAHYAR SHAVIRIAN, VALLA ZEHTAB

Dec 2008

System UseCase:



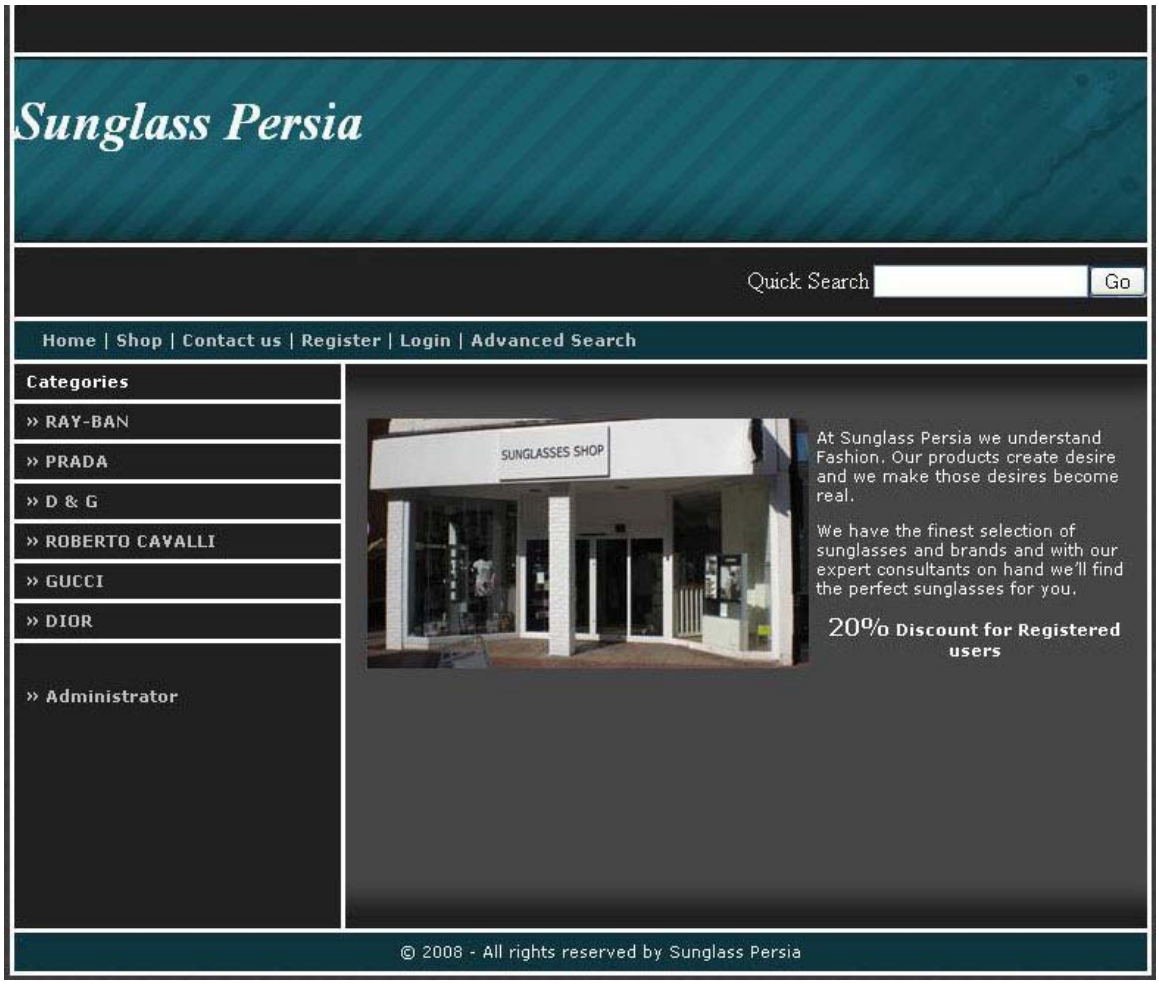
Website's Navigation



Home

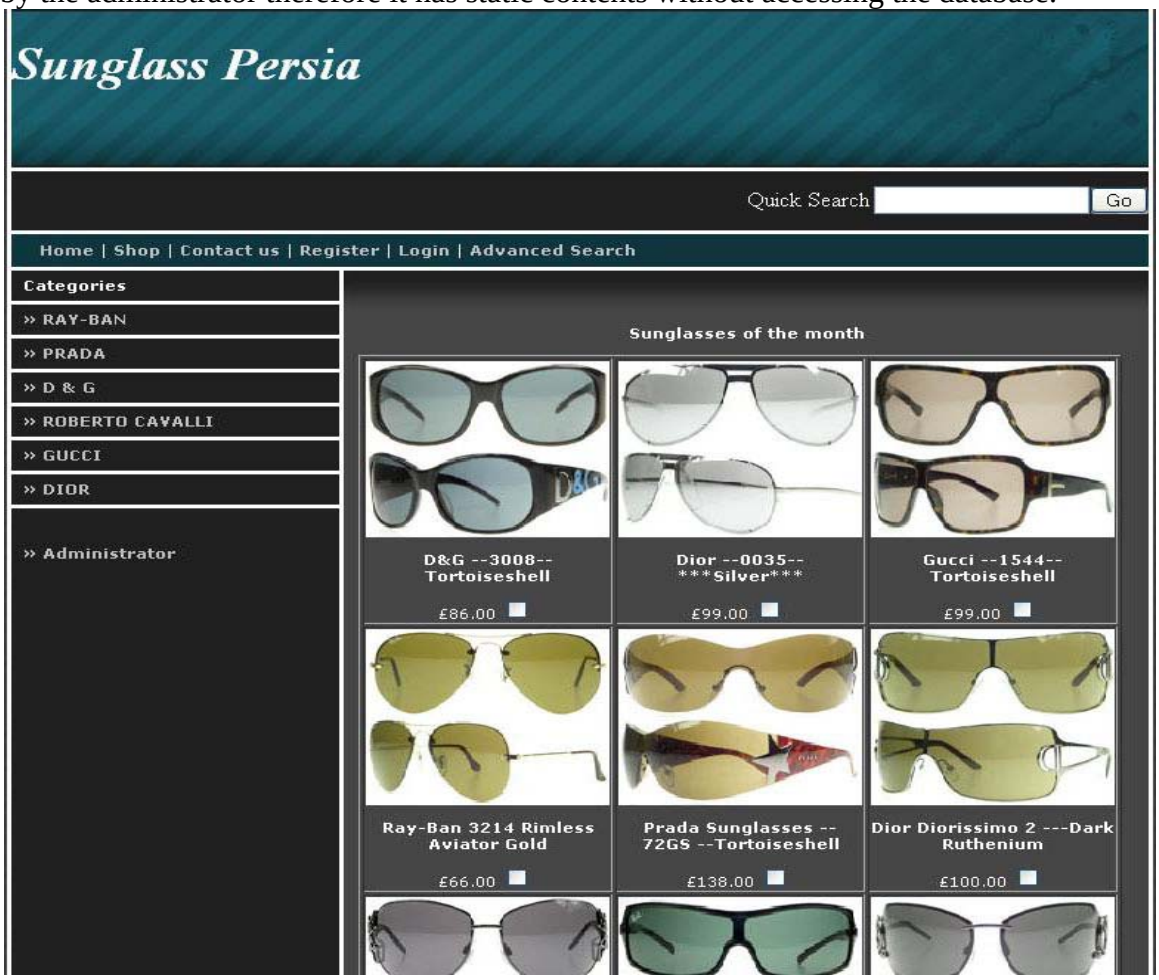
The Home page is the first page that user access. It contains general information about the Sunglass Persia and a picture of the shop. There are a number of hyperlinks and features which are renewed in all pages of the web site. These are listed below:

- Website banner with the shop's name
- Quick Search button
- Horizontal menu on the top of the site contains: Home | Shop | Contact us | Register | Login | Advanced Search
- Sunglasses categories/brands listed vertically on the left side
- Administrator link placed at the bottom of the vertical menu



Shop

The shop page contains the most popular sunglasses of the month. This page is updated monthly by the administrator therefore it has static contents without accessing the database.



Contact us

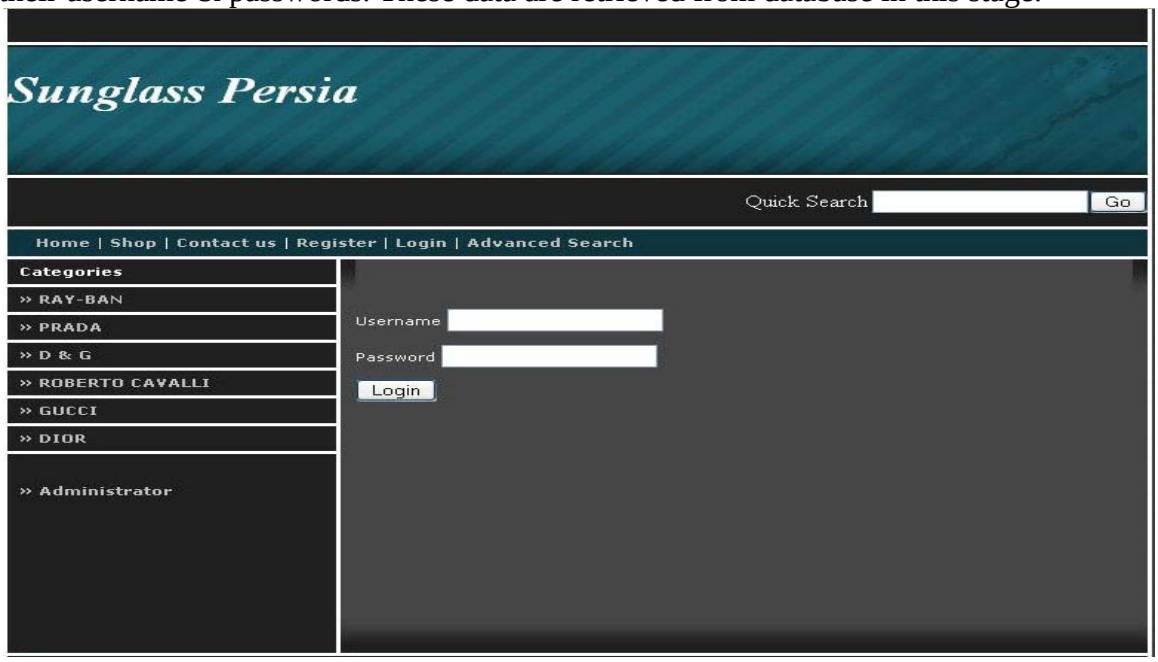
This page contains simple contacting information such as administrator's email and phone number.

Register

This page includes registration form which customers should fill it to sign up. All the data gathered in this page is recorded in the **Users** table of the data base.

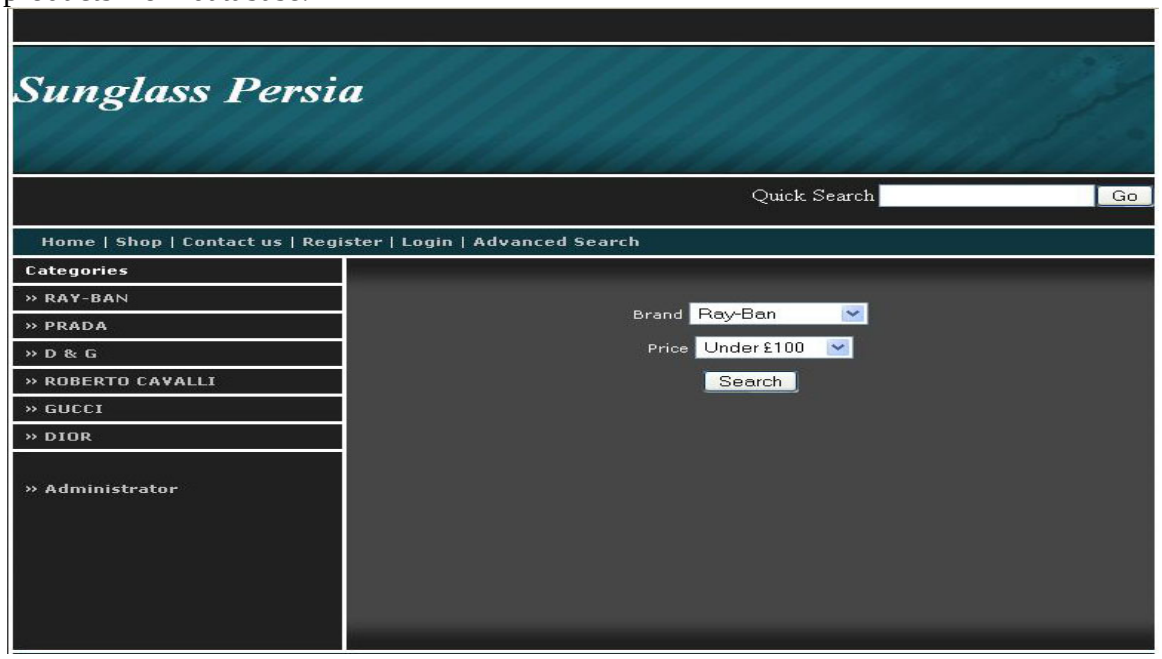
Login

Users submitted through registration form are able to sign in via login page where they enter their username & passwords. These data are retrieved from database in this stage.



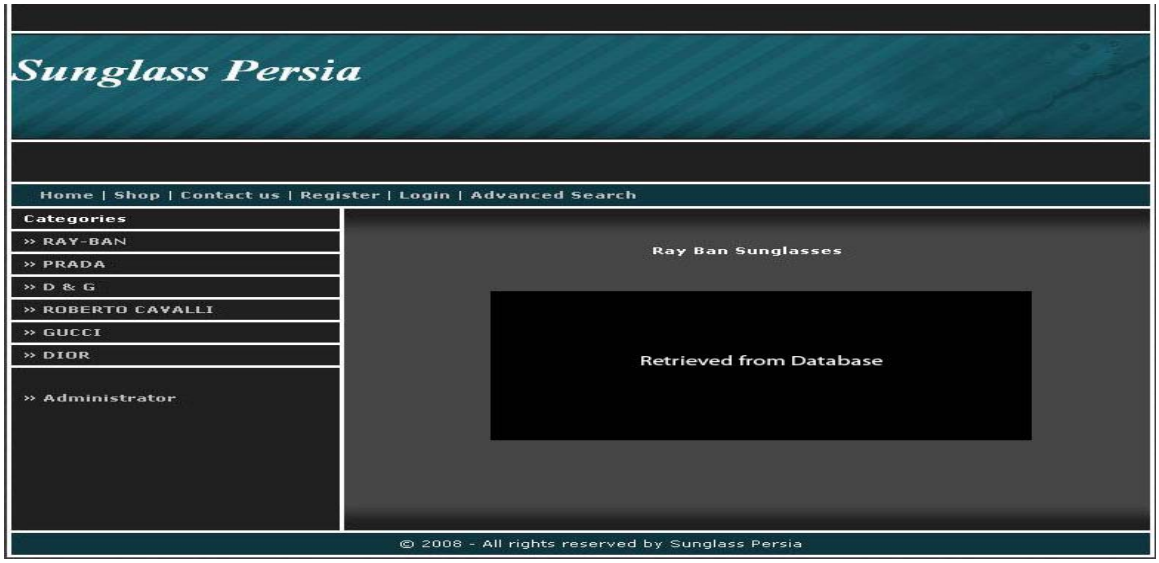
Advanced Search

Customers are able to specify their desired products by means of advanced search where they could specify their ideal brands and price through a search engine which search and find products from database.



Products

Products are categorized in terms of brand's name. Each brand displays its models in a separate page. All brand's names, models, pictures and prices are retrieved from **Sunglasses** table of the database.



Administrator

Admin page is designed for website administrator. He/She could login through a login page and add or delete products. Adding a product includes inserting a record into the database while Deleting involves two stages: 1st is recalling a database and 2nd is deleting a desired record.

Check Out

Check out page is the final stage of online shopping where the customer's order (item, price …) is displayed.

Project outline

The system will allow customers to purchase goods online. The server side section is intended to work with JSP by means of Servlet technology to access the database built by MySQL, since the number of the system's users are limited therefore it's much easier to manage. JSP codes are generated by NETBEANS application.

Keeping simple has been tried to be considered in every stage of project design. It's also tried to make project platform independent for launching on both Windows & Linux. System requirements are totally listed below:

Windows®	Linux®
Intel® Pentium® II 450MHz or faster processor (or equivalent)	Modern processor (800MHz or faster)
64MB of RAM	64MB of RAM, 16MB of graphics memory
Web browser (IE, Mozilla FireFox, Opera,…)	Same
Java Script enabled in web browser	Same
EasyPhp or Xamp for launching the system on local system	Same

Access Layers

There are three access layers in this system each have specific use cases:

1- Unauthorized user: Can simply browse brand pages, can search brands either by means of Quick-Search or Advanced-Search, & also has the ability to register and become an authorized user.
2- Authorized user: Implements all the use cases of the unauthorized user as well as the ability to login, logout & buy products.
3- Administrator: In fact this feature is designed for the website administrator(shop's owner) The administrator can login via admin username & password & is able to add, edit or remove products.

Technical outline

Using three-tier structure:

1- Interface (web pages) in JSP

Web pages involved in getting user's request through forms are listed below:

Register.jsp:

```
<form name="form1" id="form1" method="post" action="RegServlet">
    <input type="hidden" name="action" value="newMember">
<label>Name
      <input type="text" name="name" id="name" value="">
      </label>


      <label>Family Name
      <input type="text" name="family" id="family">
      </label>

        <label>Username
      <input type="text" name="username" id="username">
      </label>
      <label>Password
      <input type="password" name="password" id="password">
      </label>
      <label>Address
      <input type="text" name="address" id="address">
      </label>
```

```
        <label>Tel
        <input type="text" name="tel" id="tel">
</label>
                <label>Email
        <input type="text" name="email" id="email">
        </label>
          <input type="submit" name="submit" id="button" value="Submit">
</form>
```

Login.jsp:

```
<form name="form1" method="post" action="LoginServlet">
    <input type="hidden" name="action" value="doLogin">
    <label>Username
      <input type="text" name="Username" id="textfield">
    </label>

  <label>
  Password
    <input type="password" name="Password" id="textfield2">
    </label>
    <input type="submit" name="login" id="button" value="Login">

</form>
```

Advanced_search.jsp:

```
<form name="form2" action="ProdServlet?action=advancedSearch" method="post">
                        <label> Brand
                          <select name="brand" id="brand">
                            <option value="Ray-Ban" selected>Ray-Ban</option>
                            <option value="Prada">Prada</option>
                            <option value="D_G">D&amp;amp;G</option>
                            <option value="RobertoCavalli">RobertoCavalli</option>
                            <option value="Gucci">Gucci</option>
                            <option value="Dior">Dior</option>
                          </select>

                        </label>

                                        <label>Price
                          <select name="unitprice" id="unitprice">
                            <option value="99" selected>Under &pound;100</option>
```

```
                        <option value="101">Greater  &pound;100</option>
                    </select>
                </label>

                <label>
                    <input type="submit" name="button" id="button"
value="Search">
                </label>
                                </form>
```

Login Session (included in pages via jsp tags):

```
<%
                int loginid=0;
                if(session.getAttribute("loginid")!=null){
                    loginid=Integer.parseInt((String)session.getAttribute("loginid"));
                }else loginid=0;
                %>

                <%if(loginid>0){%>
            <a href="LoginServlet?action=logout">Logout</a>
            <%}else{%>
           <a href="login.jsp">Login</a>
```

Profile.jsp:

```
<form name="form1" id="form1" method="post"
action="RegServlet?action=updateProfile">
<%
      ResultSet result=(ResultSet)request.getAttribute("result");
      result.next();
      %>
<input type="hidden" name="id" value="<%=result.getString("id")%>">
<label>Name
      <input type="text" name="name" id="name"
value="<%=result.getString("name")%>">
      </label>

      <label>Family Name
      <input type="text" name="family" id="family"
value="<%=result.getString("fname")%>">
      </label>
      <label>Username
```

```
        <input readonly type="text" name="username" id="username"
value="<%=result.getString("username")%>">
        </label>

        <label>Password
        <input type="password" name="password" id="password"
value="<%=result.getString("password")%>">
        </label>
        <label>Address
        <input type="text" name="address" id="address"
value="<%=result.getString("address")%>">
        </label>
        <label>Tel
        <input type="text" name="tel" id="tel" value="<%=result.getString("tel")%>">
</label>
        <label>Email
        <input type="text" name="email" id="email"
value="<%=result.getString("email")%>">
        </label>
</form>
```

Quick-Search (included in pages via jsp tags):

```
<form action="ProdServlet?action=quickSearch" method="post"><span
class="style1"><span class="style2">Quick Search</span>
     <input name="q" type="text"><input name="Go" type="submit"
value="Go"></span>
   </form>
```

2- Data Access Layer (include JDBCManager) : JDBCManager is a java class where the connection method to database is defined :

```
public class JDBCManager {
Connection a;
   public void disconnectFromDbase() {
      if(a==null){
         return;
      }
      try{
```

```
            a.close();
            a=null;
            return;

        }
        catch(SQLException e){
              System.out.println(e);
        }
    }

        public void setConnection() {

        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(JDBCManager.class.getName()).log(Level.SEVERE, null, ex);
            ex.printStackTrace();
        }

        try {
            a = (Connection)
DriverManager.getConnection("jdbc:mysql://localhost:3306/sunglasses", "root", "");

        } catch (SQLException sqlexception) {
            System.out.println("Error :SQL Connection");
        }

    }
public ResultSet submitExecuteQuery(String s) throws SQLException {
        if (a==null){
            setConnection();
        }
        Statement statement;

        try {
            statement = a.createStatement();
        }
        catch (SQLException e) {
            setConnection();
            try {
                statement = a.createStatement();
            }
            catch(SQLException e1){
```

```
            throw e1;
          }

        }
        ResultSet rs;

        try{
          rs= statement.executeQuery(s);

        }
        catch(SQLException e){
          throw e;
        }

          return rs;
        }

      public void submitUpdateQuery(String s)
        throws SQLException
      {
        if(a == null)
          setConnection();
        Statement statement;
        try
        {
          statement = a.createStatement();
        }
        catch(SQLException _ex)
        {
          setConnection();
          try
          {
            statement = a.createStatement();
          }
          catch(SQLException sqlexception1)
          {
            System.err.println("Error: dbase(submitQuery): createStatement failed - servlet
exiting");
            throw sqlexception1;
          }
        }
        try
        {
```

```
            statement.executeUpdate(s);
            return;
        }
        catch(SQLException sqlexception)
        {
            System.err.println((new StringBuilder()).append("Error: dbase(submitQuery):
execute query failed: ").append(sqlexception).append("\nQuery:").append(s).toString());
            throw sqlexception;
        }
    }

    }
```

3- Common Layer: All entities needed to be modified dynamically are initiated in this layer. For this application two classes are defined in order to handle the information of Products(Products.java) & Users(Users.java)

User.java

All the users information are initiated here as well as the methods needed for getting queries from database :

```
public class Users {
    public Users (){
      id=0;
      name="";
      fname="";
      username="";
      password="";
      address="";
      tel="";
      email="";
    }

  public Users(int i){
      id=i;
      name="";
      fname="";
      username="";
      password="";
      address="";
      tel="";
      email="";
```

```
try{
    JDBCManager jn = new JDBCManager();
    ResultSet rs = null;
    jn.setConnection();
   String s = "SELECT * from user WHERE id="+i;
   if ((rs = jn.submitExecuteQuery(s)).next())
   {
      name = rs.getString("name");
      fname = rs.getString("fname");
      username = rs.getString("username");
      password = rs.getString("password");
      address = rs.getString("address");
      tel = rs.getString("tel");
      email = rs.getString("email");


   }
   jn.disconnectFromDbase();
   return;


  }
  catch (SQLException _ex)
  {
     return;
  }
  public int id;
public String name;
public String fname;
public String username;
public String password;
public String address;
public String tel;
public String email;
 }
```

Methods defined in Users.java

1- addUser(): When an unauthorized user register, his/her information inserted to data base via this method:

```
public void addUsers()
   {
      try
      {
         JDBCManager jdbcManager = new JDBCManager();
         jdbcManager.setConnection();
         String s = null;
         s = "Insert into user (name,fname,username,password,address,tel,email ) values (";
         s+="'"+name+"',";
         s+="'"+fname+"',";
         s+="'"+username+"',";
         s+="'"+password+"',";
         s+="'"+address+"',";
         s+="'"+tel+"',";
         s+="'"+email+"')";
         jdbcManager.submitUpdateQuery(s);

         jdbcManager.disconnectFromDbase();
         return;
      }
      catch(SQLException sqlexception)
      {
         System.out.println(sqlexception.toString());
      }
   }
```

2- giveUseres(): Passing the user information through RegServlet when the user wants to see his/her profile and update it:

```
public ResultSet giveUser() throws SQLException
   {
      ResultSet resultset;
      JDBCManager jdbcManager;
      (jdbcManager = new JDBCManager()).setConnection();
      String s2 = null;
      s2 ="SELECT  *   from user WHERE   id="+id;
      resultset = jdbcManager.submitExecuteQuery(s2);
      return resultset;
```

```
    }
```

3- updateUser(): When the user enters new information, the database is being updated by means of this method:

```
public void updateUser()
   {
      try
      {
         JDBCManager jdbcManager = new JDBCManager();
         jdbcManager.setConnection();
         String s = null;
         s = "update user set name='"+name+"' , fname='"+fname+"' ,
password='"+password+"' , address='"+address+"' , tel='"+tel+"' , email='"+email+"' where
id="+id;
         jdbcManager.submitUpdateQuery(s);

         jdbcManager.disconnectFromDbase();
         return;
      }
      catch(SQLException sqlexception)
      {
         System.out.println(sqlexception.toString());
      }
   }
```

Products.java

All the products information are initiated here as well as needed methods for getting queries from database:

```
public class Products {

   public int id;
   public String model;
   public String color;
   public String brand;
   public int unitPrice;
   public String img1;
   public String img2;

   public Products() {
   id = 0;
```

```
 model = "";
 brand = "";
 color = "";
 unitPrice = 0;
 img1 = "";
 img2 = "";
 }

 public Products(int i) {
 id = 0;
 model = "";
 brand = "";
 color = "";
 unitPrice = 0;
 img1 = "";
 img2 = "";
  try{
 JDBCManager jm = new JDBCManager();
ResultSet rs=null;
     jm.setConnection();
     String s="select * from product where id=i";
     if( (rs = jm.submitExecuteQuery(s)).next()){
          id=rs.getInt("id");
          model=rs.getString("model");
          brand=rs.getString("brand");
          color=rs.getString("color");
          unitPrice=rs.getInt("unitPrice");
          img1=rs.getString("img1");
          img2=rs.getString("img2");

     }
     jm.disconnectFromDbase();
     return;
 }
  catch(SQLException e){
      return;
  }
 }
```

Methods defined in Products.java

1- quickSearch(): This method is defined in order to make a Quick-Search for user. When a user enters a model name (even similar to models existed in database), models in database are retrieved and presented.

```
public ResultSet quickSearch(String q) throws SQLException
   {
      ResultSet resultset;
      JDBCManager jdbcManager;
      (jdbcManager = new JDBCManager()).setConnection();
      String s2 = null; s2 ="SELECT  *   from product WHERE   model like'%"+q+"%' ";
      resultset = jdbcManager.submitExecuteQuery(s2);
      return resultset;
   }
```

2- advancedSearch(): Search a product in terms of its brand & price:

```
public ResultSet advancedSearch(String enq) throws SQLException
   {
      ResultSet resultset;
      JDBCManager jdbcManager;
      (jdbcManager = new JDBCManager()).setConnection();
      String s2 = null;
      s2 ="SELECT  *   from product WHERE   brand='"+brand+"' and unitprice "+enq+"
'"+unitPrice+"'";
      resultset = jdbcManager.submitExecuteQuery(s2);
      return resultset;
   }
```

Servlets

RegServlet.java: Handel the user's requests & calls methods defined in Users.java in accordance with the user's request:

```
public class RegServlet extends HttpServlet {

   /**
   * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
   * @param request servlet request
   * @param response servlet response
   */
```

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException, SQLException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String s = request.getParameter("action") != null ? request.getParameter("action") :
"";
        if(s.equals("newMember")){

        try {
            Users users=new Users();
            users.name = request.getParameter("name");
            users.fname = request.getParameter("family");
            users.username = request.getParameter("username");
            users.password = request.getParameter("password");
            users.address = request.getParameter("address");
            users.tel = request.getParameter("tel");
            users.email = request.getParameter("email");

            users.addUsers();


getServletConfig().getServletContext().getRequestDispatcher("/login.jsp").forward(reque
st, response);

        } finally {
            out.close();
        }
        }


        if(s.equals("profile")){

        try {
            Users users=new Users();
            users.id = Integer.parseInt(request.getParameter("id"));

            request.setAttribute("result", users.giveUser());


getServletConfig().getServletContext().getRequestDispatcher("/profile.jsp").forward(req
uest, response);
```

```
        } finally {
            out.close();
        }
        }


        if(s.equals("updateProfile")){

        try {
            Users users=new Users();
            users.id = Integer.parseInt(request.getParameter("id"));
            users.name = request.getParameter("name");
            users.fname = request.getParameter("family");
            users.address = request.getParameter("address");
            users.password = request.getParameter("password");
            users.tel = request.getParameter("tel");
            users.email = request.getParameter("email");

            users.updateUser();


getServletConfig().getServletContext().getRequestDispatcher("/home.jsp").forward(requ
est, response);

        } finally {
            out.close();
        }
        }


    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
    * Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
```

```
            processRequest(request, response);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    /**
    * Handles the HTTP <code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    /**
    * Returns a short description of the servlet.
    */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

ProdServlet.java: Handel the user's requests of products (include search, advanced search &…) & calls the desired methods from the class Products.java:

```
public class ProdServlet extends HttpServlet {

    /**
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
    * @param request servlet request
    * @param response servlet response
    */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
```

```
    throws ServletException, IOException, SQLException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(true);

        String s = request.getParameter("action") != null ? request.getParameter("action") :
"";
        if(s.equals("searchBrand")){
        try {
            Products products=new Products();
            products.brand = request.getParameter("brand");

            request.setAttribute("result", products.searchOnBrand());
            request.setAttribute("brand", products.brand);

getServletConfig().getServletContext().getRequestDispatcher("/resultBrand.jsp").forward
(request, response);

        } finally {
            out.close();
        }
        }
        if(s.equals("advancedSearch")){
        try {
            Products products=new Products();
            products.brand = request.getParameter("brand");
            products.unitPrice = Integer.parseInt(request.getParameter("unitprice"));
            String enq=(products.unitPrice==99) ? "<=" : ">=";

            request.setAttribute("result", products.advancedSearch(enq));
            request.setAttribute("brand", products.brand);

getServletConfig().getServletContext().getRequestDispatcher("/resultBrand.jsp").forward
(request, response);

        } finally {
            out.close();
        }
        }
        if(s.equals("quickSearch")){
        try {
            Products products=new Products();
            String q = request.getParameter("q");
```

```
            request.setAttribute("result", products.quickSearch(q));

getServletConfig().getServletContext().getRequestDispatcher("/resultBrand.jsp").forward
(request, response);
        } finally {
            out.close();
        }
        }
        if(s.equals("addToCart")){
            String
mysession=String.valueOf(Math.random());session.setAttribute("mysession",
mysession);
            String loginid=(String)session.getAttribute("loginid");
            Products products=new Products();
            String count=request.getParameter("count");
            String prodId[]=new String[Integer.parseInt(count)];
            String qty[]=new String[Integer.parseInt(count)];
            String choosed[]=new String[Integer.parseInt(count)];
            int countG=0;
            for(int i=1;i<=Integer.parseInt(count);i++){
              choosed[i-1]=request.getParameter("choosed"+i);
              if(choosed[i-1]!=null){
                 prodId[countG]=request.getParameter("id"+i);
                 qty[countG]=request.getParameter("qty"+i);
                 countG++;
              }
            }
            for(int j=0;j<countG;j++){
               products.insertInvoice(loginid,qty[j],prodId[j],mysession);
            }
            request.setAttribute("result", products.viewCart(mysession));

getServletConfig().getServletContext().getRequestDispatcher("/viewCart.jsp").forward(r
equest, response);
        }

        if(s.equals("addProduct")){

        try {
            Products products=new Products();
            products.brand = request.getParameter("brand");
            products.model = request.getParameter("model");
```

```
            products.unitPrice = Integer.parseInt(request.getParameter("price"));
            products.img1 = request.getParameter("imagepath1");
            products.img2 = request.getParameter("imagepath2");


            products.addProduct();


getServletConfig().getServletContext().getRequestDispatcher("/addProduct.jsp").forward
(request, response);

        } finally {
            out.close();
        }
        }
        if(s.equals("updateProducts")){

        try {
            Products products=new Products();
            products.id = Integer.parseInt(request.getParameter("id"));
            products.brand = request.getParameter("brand");
            products.model = request.getParameter("model");
            products.unitPrice=Integer.parseInt(request.getParameter("price")) ;
            products.img1 = request.getParameter("imagepath1");
            products.img2 = request.getParameter("imagepath2");

            products.updateProducts();


getServletConfig().getServletContext().getRequestDispatcher("/home.jsp").forward(requ
est, response);

        } finally {
            out.close();
        }
        }
         if(s.equals("deleteProduct")){

        try {
            Products products=new Products();
            products.id = Integer.parseInt(request.getParameter("id"));
            products.brand = request.getParameter("brand");
            products.model = request.getParameter("model");
```

```
            products.unitPrice=Integer.parseInt(request.getParameter("price")) ;
            products.img1 = request.getParameter("imagepath1");
            products.img2 = request.getParameter("imagepath2");

            products.deleteProduct();


getServletConfig().getServletContext().getRequestDispatcher("/home.jsp").forward(requ
est, response);

        } finally {
            out.close();
        }
        }




    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
    * Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    /**
    * Handles the HTTP <code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

```
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

    /**
    * Returns a short description of the servlet.
    */
    public String getServletInfo() {
        return "Short description";
    }
    // </editor-fold>
}
```

LoginServlet.java: Handle the login information by getting username & password as parameters and check them via checkUser() method defined in Users.java, if correct the user is permitted to login and the session is created, if not the invalid data message is displayed:

```
public class LoginServlet extends HttpServlet {

    /**
    * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
methods.
    * @param request servlet request
    * @param response servlet response
    */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException, SQLException {
        try {
      //JOptionPane.showMessageDialog(null, "we are in LoginServlet!");
      HttpSession session = request.getSession(true);

      String user = request.getParameter("Username");
      String pass = request.getParameter("Password");
      System.out.println(user);
      System.out.println(pass);
      String s = request.getParameter("action") != null ? request.getParameter("action") :
"";

      if(s.equals("doLogin")){
```

```
            int i=Users.checkUser(user, pass);
            new Users(i);
             if(i != 0){
                session.setAttribute("loginid",String.valueOf(i));
                session.setAttribute("user",user);
                System.out.println("Salaam! No. "+session.getAttribute("user"));
                response.sendRedirect("home.jsp");
            }else{
               request.setAttribute("message","Error: Username or Password incorrect");
               System.out.println(request.getAttribute("message"));

getServletConfig().getServletContext().getRequestDispatcher("/login.jsp").forward(reque
st, response);
                //response.sendRedirect("login.jsp");
            }
        }

        if(s.equals("logout"))
        {
            request.getSession().invalidate();
            request.setAttribute("message", "<center>Logout successful!</center>");

getServletConfig().getServletContext().getRequestDispatcher("/login.jsp").forward(reque
st, response);
        }




        } finally {

        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
    * Handles the HTTP <code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

```
            try {
                processRequest(request, response);
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        /**
        * Handles the HTTP <code>POST</code> method.
        * @param request servlet request
        * @param response servlet response
        */
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
            try {
                processRequest(request, response);
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }

        /**
        * Returns a short description of the servlet.
        */
        public String getServletInfo() {
            return "Short description";
        }

        private boolean checkUser(String user, String pass) {
            return true;
            //throw new UnsupportedOperationException("Not yet implemented");
        }

}
```