**Foods R US**

**An E-Commerce Website**

Designed and Developed By

**Wook Cho (cse: cho0708a)**

**Yang Bai (cse: baiyang1)**

**Jessica Sardinas Rodriguez (cse: invigil)**

*Submitted as the group project for the course*

*EECS4413 – Building E-Commerce Systems*

*Prof. H. Roumani – Fall 2018*

TABLE OF CONTENTS…

i. Design………………………………………………………………………………………1 - 2

Overview of data flow, architecture, key issues of design and phases of design progression. Describes any problems in design and any issues caused by previously flawed designs and their solutions.

ii. Implementation…………………………………………………………….……………….3 - 5

Overview of any implementation issues, testing methods and limitations with respect to the implementation and shortcomings due to implementation.

iii. The Team…………………………………………………………….……………………..6 - 7

Overview of scheduled meeting times, how work was kept up to date and consistent, lessons learned from the group project. Member names and signatures and member contribution is outlined in this section.

iv. Output Samples…………………………………………………………….…………………..8

Sample P/Os and their location on disk and sample completed P/O reports and their location on disk are shown here.

v. The Source Code…………………………………………………………….……………9-##

i. Servlets

i. Login pg. 10 -11

ii. Catalog pg. 14

iii. Cancel pg.15

iv. Category pg.15-17

v. Checkout pg.21-22

vi. Confirm pg.27-28

vii. Item pg.28-30

viii. MyAccount pg.34-35

ix. PODisplay pg.39-41

x. ShoppingCart pg.41-44

xi. Update pg. 50-52

ii. Analytics/ Special Features/ Adhoc

i. Analytics (jspx view for client) pg. 66

ii. Advertisement (Filter) pg. 70

iii. Main (Listener) pg.68-69

iv. Statistics (Calculation Class for the Listener) pg.69-70

v. Admin (Servlet for the Listener) pg. 66-67

iii. jspx

i.Home pg. 9

ii. Category pg. 16-17

iii. Checkout pg. 22-25

iv. Item pg. 30-32

v. MyAccount pg.36

vi. ShoppingCart pg. 44-47

v. Model

i. Engine pg.52-59

ii. CategoryBean pg.60

iii. ItemBean pg.61

iv. WritePO pg. 62-66

vi. CSS

i. Home pg. 10 - 13

ii. Category pg. 18-21

iii. Checkout pg. 25-27

iv. Item pg. 32-34

v. Account pg.37-38

vi. ShoppingCart pg.48-50

vii. Middleware

i. ReportGenerator pg.71

ii. Unionizer pg. 71-74

Design

System Architecture and Dataflow (highly simplified) see next page for details..



List<String> POs

List<String> POs

Renames processed POs by adding a \* prefix

Poreport\_#.xml

OAuth Server

Apache Tomcat Server

Serves

Main page

/efoods

Serves

Item

population

Provides a filter attribute

links page to

User identification

Add an item

Links back to



PO Directory

PO Report Directory

Order checkout by button press

Changes quantites

Generates new PO

List<String>POs



Viewable PO



Order confirmed by button press

Order cancelled by button press

Linked to by Catalog, Items, Category and Cart



Redirects to

Linked to by Catalog and Shopping Cart



Type class

Model Package: 

Type class

Displays PO in HTML format

The model package is responsible for all calculations in the B2C module, it returns all results by contacting the Engine class.

Dark squares are abstracted and not directly modified. Lightly colored ovals call the Model Package (the Engine class in particular)), while the darker ovals in the middleware rely on their own methods where Report Generator (main method) references the Unionizer (singleton).

The login servlet is linked to by various html links to allow the user to login anytime before checkout, it returns a verified user and name value from a OAuth link that is used as a session-wide account attribute.

Design Key Issues

1. How do we make sure the client is logged in when they checkout?
2. How do we keep a unique session login?
3. How do we maintain one shopping cart per client?
4. How do we make sure the client doesn’t add an invalid quantity to the shopping cart?
5. How can the client view P/Os without it effecting the middleware?
6. How can the middleware process a P/O only once without effecting the client’s ability to view a P/O?
7. How can we improve the search facility so it returns relevant searches?
8. What information should the P/O report actually have?
9. How do we allow the client to view items in a comfortable way?

Design Decisions

1. During checkout the user is immediately checked for an attribute called account, if that attribute is null then the user has never entered a login screen and authenticated so they are immediately asked to login.
2. We decided to add an attribute called “account” this account attribute is necessary to access any features that require being logged in such as viewing account info (displays a list of P/Os specific to the account) or checking out. The account attribute is given the value user which is obtained from an open authorization to a server that returns both “user” and “name”, this prevents the user from wrongly accessing things from other accounts for privacy reasons.
3. Each session has a specific attribute called order that is modified whenever the client changes their shopping cart in some way (updating it, adding an item, deleting an item, checking out), as such the shopping cart is maintained on a per session basis.
4. If the client tries to add an invalid entry or updates the cart with an invalid entry that is less than 1 or not numeric then an exception is thrown and an error below the shopping cart is displayed that states that the shopping cart could not be modified (either it could not be updated, or an item could not be added).
5. A P/O is displayed as long as it has the specified account name captured by a regular expression, as such it does not effect the middleware as long as the middleware does not remove the account name.
6. The middleware adds a prefix \* to the file name if the P/O has been processed, since the client can read P/Os like this still then it doesn’t effect B2C. The middleware does not process P/Os that have a prefix \* to the beginning, this cannot occur in unprocessed P/Os making this an effective marker.
7. The search facility was made more relevant by sorting the results in a lexicographically meaningful way, the entries that matched the search parameter closest (without assuming the search parameter is a prefix) were displayed first alongside any others that contained the parameter. This allows the user to quickly find all “sirloin” meats even if they do not know the brands.
8. The B2B requires knowing what to purchase and how much of something thus the Middleware’s P/O report only displays the item’s item code (since it is a unique label unlike the name of an item which could have subtyping) and the quantity of the item to be ordered. This was done to save space and traversal times by only keeping relevant information.
9. Clients can view their P/Os in their account info pages, this keeps a relevant page for clients to see all their past purchase orders that have not been removed yet (both unprocessed and processed P/Os), account info pages require authentication but offers a list of all P/Os the client has made. This was done to ensure that clients had privacy for their own accounts to a degree, the client should be able to keep secret how many P/Os they have and not see the P/Os of others without a direct link. This prevents accounts from being easily searchable and the invasion of privacy of other users. Of course, however direct URLs of P/Os are accessible without authentication and thus clients are able to access P/O orders even if they are not logged in with the account if they were given the URL by another client or bookmarked the URL, as per requirements.

Implementation

Implementation Key Issues

1. Should we use JSP or SPA?
2. How do we avoid unnecessary javascript?
3. How do we make sure the engine isn’t a superman class?
4. How can we manage servlets so they’re not doing any calculations and only handling requests and responses?
5. How do we handle errors?

Implementation Decisions

1. We decided to use JSP as most of us were more well trained in it and we valued the benefit of separation of concerns.
2. Often times it was hard to figure out how to implement something with only JSP, avoiding javascript was difficult at the beginning however code later completely migrated from javascript to coincide with the foundations of JSP. This was solved through more exposure and resolving problems amongst the group with communication (providing possible alternatives).
3. The engine by default plays a lot of roles, however adding too many methods makes it difficult for team members to read and understand and add their own methods if they had to introduce a new servlet, it also makes debugging code significantly harder. Many engine methods were migrated to different classes in the Model package in order to reduce this difficulty.
4. Due to the above reason migrating roles to the engine can be tricky and there was often, most notably the PODisplay servlet that made the blur between model, view and controller more apparent, due to the above problems it was easier in some cases to not fully migrate roles to other classes and blur the lines. This is thus both an issue with implementation and design (there was not a clear cut place for some servlets or features in our original plan and thus they had to be added haphazardly).
5. Errors on the server side were not shown to the user but quantity errors were given exceptions in their corresponding methods in the engine class and thus displayed to the user.

Testing

The project was tested without any testing units and was essentially done with a feature by feature testing practice. The team would continuously test when they added an additional feature the specific feature and then later would test the entire system again. Thus, the testing was only for functionality as apparent to the user, no special test cases were provided only manual testing was done.

Status

The project supports most requirements except for the analytics, it provides minimal support for an analytics system as it fails to accurately calculate any valuable statistics it does however redirect to a page called “Admin” that can supposedly be used to output such analytical statistics if they were to be numerically computable by computational methods. All other requirements are completed to standards. Furthermore, P/Os are not directly viewable by a onsite link without authentication, thus users are unable to view P/Os that are not part of an account that they are not authenticated for as a direct link on the page, however accessing a P/O that is stored by bookmarking the link or receiving a link from elsewhere does not require authentication, users are able to access past P/Os that they have knowledge of without authenticating but they are not able to be knowledgeable about P/Os that exist that are not relevant to their own account, this was done for privacy reasons. Thus, if Y authenticates they are shown only P/Os from Y and if X authenticates they cannot view Y’s P/Os they only see their own however if Y links X to the P/O then X is able to view the P/O despite not being authenticated as Y.

The Team

The work was not an official clear-cut division many parts of the project were done procedurally by everyone in the group. The group project was uploaded to a repository on GitHub and thus the group was allowed to update the project procedurally and fix past versions of the project that they might not have previously worked on. Whenever a teammate would upload a new version the teammate would inform the others by messaging them on the mobile messaging platform, What’s App, thus all group members were expected to update their version to the “latest” version as needed. As such, all group members have seen all parts of the implementation and are thus familiar enough with it to edit the code as all at one point have edited or fixed errors in previous versions of the project. However, it is the case that members did focus on certain areas even if other members would later come in to check and make necessary changes, this will be discussed in detail in after the next part.

Group projects can be difficult, to manage, as students we all have a lot of responsibilities outside the scope of one course and thus we do not all have equal time to put into the project and we work on the project at different times. Some members have jobs and a full course load and thus may not have a consistent schedule to work on the project. Thus, one lesson is that one needs to divide tasks as best they can and provide reasonable time frames for completion depending on the teammate, this is however difficult as finding the right balance for each person is nontrivial. Another lesson is that every groupmate has a different set of skills and style of coding, coding throughout the project should optimally follow a specific format so it is easy to read and understand, since we all come from different backgrounds it is a powerful lesson to have an arranged format, so all code can be in an easy to understand style. Our different set of skills should also be looked at and appropriately accounted for, for instance if one person has a lot of experience in CSS and user-friendly design it is a reasonable conclusion that they should have a major role in the design. If we were to start over, we might want to try and more closely define our roles and look into this in a deeper way, if everyone has the right balance then surely the project produced would be more impressive.

Contributions (are written by each person separately):

Wook Cho (Write Down Overview here)

Blahblah I’m a paragraph. Add Definitive detail.

Yang Bai

* Authentication
* All the CSS and layout
* Bug fixes in B2C
* Storage for the user’s selected items from shopping cart

Jessica Sardinas Rodriguez (B2C-B2B Middleware, Search Functionality, Adhoc Features, Code Clean-Up and Bug Fixes in B2C, Majority of the Report)

The middleware was fully developed by this individual and independently as a separate module, the individual minimally edited the B2C component by changing a regular expression for the WritePO component class to add a degree of connectivity in a meaningfully non-interfering way. This individual was also responsible for creating two special features, the search functionality and implementing the ad hoc advertisement feature, the features were independently designed and thus are easily removed and transferrable. Miscellaneous code cleanup was done by all individuals, this individual focused on providing type checking, null exception handling and client input related exception handling (the individual added error messages for invalid quantities and added if conditions to the HTML to prevent users from doing invalid actions such as checking out with an empty cart) , small scale transferring of methods (concatenating methods into one), sanitizing sql to prevent sql injection and servlet redirection changes for efficiency reasons. The majority of the report was also written by this individual however everyone contributed in this report more specifically the individual added diagrams, TOC, cover page, implementation section, majority of the design section and the output sample section.

Members learned about the elements of the project by communication via a messaging app as mentioned each member would alert other members when they had uploaded a new version of the project. Members would also describe the scope of what was changed in this version and would alert members before making any changes.

I hereby attest to the accuracy of the information contained in "The Team" section of this Report.

Wook Cho \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Yang Bai \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jessica Sardinas Rodriguez \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output Samples

Sample P/O (saved in the Home directory in a folder called PO)

﻿<order id="1" submitted="Sun Nov 25 14:27:07 EST 2018">

<customer account="invigil">

<name>Jessica Sardinas Rodriguez</name>

</customer>

<items>

<item number="0905A044 ">

<name>Minced Rib Meat by VX </name>

<price>6.49</price>

<quantity>1</quantity>

<extended>6.49</extended>

</item>

</items>

<total>6.49</total>

<shipping>0.0</shipping>

<HST>0.8437</HST>

<grandTotal>7.3337</grandTotal>

</order>

Sample Report P/O (saved in the Home directory in a folder called POReport)

﻿<orders generated="Sat Nov 24 17:21:12 EST 2018" total="3">

<items>

<item number="0905A306 ">

<quantity>2</quantity>

</item>

<item number="0905A044 ">

<quantity>11</quantity>

</item>

<item number="0905A365 ">

<quantity>2</quantity>

</item>

</items>

</orders>

The Source Code

Home page (jspx):

<?xml version="1.0" encoding="ISO-8859-1" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt" version="2.0">

<jsp:directive.page contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" session="false" />

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:c="http://java.sun.com/jsp/jstl/core">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Food R Us</title>

<link href="res/login.css" rel="stylesheet" type="text/css" />

</head>

<BODY>

<div id="title">

<c:if test="${username == null}">

<a href="Login.do" class="sign">Sign In</a>

</c:if>

<c:if test="${username != null}">

<a class="sign">${username}</a>

</c:if>

<a href="MyAccount.do" class="title">P/O Page</a>

<a href="ShoppingCart.do" class="title">My Cart</a>

</div>

<div id="caption">Foods R US</div>

<div id="notice">

<p>Foods R Us provides a wide variety of food

items at great prices. Our products are all of highest quality. A

void wasting time at your local grocery store and simply order food

items online and have then delivered right to your door.

</p>

</div>

<div id="start">

<a href="Category.do" class="shop">>>SHOP NOW</a>

</div>

</BODY>

</html>

</jsp:root>

Login (Servlet)

package ctrl;

import java.io.IOException;

import java.io.Writer;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

\* Servlet implementation class Login

\*/

@WebServlet("/Login.do")

public class Login extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public Login() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String result = request.getQueryString();

HttpSession session;

session = request.getSession();

if(session.getAttribute("account")==null) {

if(result != null)

{

String the\_username = request.getParameter("user");

String name = request.getParameter("name");

request.setAttribute("username", the\_username);

session.setAttribute("account", the\_username);

session.setAttribute("name", name);

response.sendRedirect(request.getRequestURL().toString());

}

else

{

response.sendRedirect("https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back=" + request.getRequestURL().toString());

}

} else

{

String the\_old\_username = (String) session.getAttribute("account");

request.setAttribute("username", the\_old\_username);

this.getServletContext().getRequestDispatcher("/eFoods").forward(request, response);

}

//}

//}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

Home page(css)

@charset "UTF-8";

BODY {

background-image: url("https://cmkt-image-prd.global.ssl.fastly.net/0.1.0/ps/3667914/1820/1213/m1/fpnw/wm1/mm0k1z4eopjimitemdhs6fmiqbvykaf8eubhbyldigxeuwfu2irobuuqz5pjkc1q-.jpg?1512111122&s=54f66daefac23151f5aa1b08d3c76cca");

background-size: cover;

/\* display: flex; \*/

background-repeat: no-repeat;

}

#title {

width: 100%;

}

#caption {

margin-top: 260px;

text-align: center;

font-family: sans-serif;

color: white;

width: 100%;

font-size: 96px;

font-weight:800;

float: right;

text-shadow: 3px 3px 3px #ff0000;

}

#notice {

margin-top: 60px;

margin-left: 100px;

color: white;

width: 20%;

color: white;

float: left;

}

#start {

margin-top: 120px;

margin-left: 20%;

float: left;

}

.shop {

text-decoration-line: none;

color: #f75000;

font-family: sans-serif;

font-size: 28px;

}

.sign, .title {

color: white;

float: right;

text-decoration-line: none;

}

.title:after {

content: " | ";

margin-left: 10px;

margin-right: 10px;

}

Catalog (Servlet)

package ctrl;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

import model.ItemBean;

@WebServlet("/eFoods")

public class Catalog extends HttpServlet {

private static final long serialVersionUID = 1L;

public Catalog() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

HttpSession session;

session = request.getSession();

String user\_name = (String) session.getAttribute("account");

request.setAttribute("username", user\_name);

this.getServletContext().getRequestDispatcher("/Home.jspx").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}}

Cancel (Servlet)

package ctrl;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

@WebServlet("/Cancel.do")

public class Cancel extends HttpServlet {

private static final long serialVersionUID = 1L;

public Cancel() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

HttpSession session;

session = request.getSession();

Engine engine = new Engine();

engine.doCancel(session);

this.getServletContext().getRequestDispatcher("/Category.do").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

Category (Servlet)

package ctrl;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

@WebServlet("/Category.do")

public class Category extends HttpServlet {

private static final long serialVersionUID = 1L;

public Category() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Engine engine = new Engine();

List categories = new ArrayList();

categories = Engine.doCategory();

HttpSession session;

session = request.getSession();

String result = request.getQueryString();

if(result != null) {

if(request.getParameter("auth") == null)

{

String the\_username = request.getParameter("user");

String name = request.getParameter("name");

session.setAttribute("account", the\_username);

session.setAttribute("name", name);

String user\_name = (String) session.getAttribute("account");

request.setAttribute("username", user\_name);

request.setAttribute("result", categories);

response.sendRedirect(request.getRequestURL().toString());

}

else

{

response.sendRedirect("https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back="+request.getRequestURL().toString());

}

} else

{

String user\_name = (String) session.getAttribute("account");

request.setAttribute("username", user\_name);

request.setAttribute("result", categories);

this.getServletContext().getRequestDispatcher("/Category.jspx").forward(request, response);

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

Category (jspx)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt" version="2.0">

<jsp:directive.page contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" session="false" />

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link href="res/category.css" rel="stylesheet" type="text/css" />

<title>Food R Us</title>

</head>

<BODY>

<div id="title">

<a href="eFoods" class="home">HOME</a>

<c:if test="${username == null}">

<a href="Category.do?auth=y" class="sign">Sign In</a>

</c:if>

<c:if test="${username != null}">

<a class="sign">${username}</a>

</c:if>

<a href="MyAccount.do" class="title">P/O Page</a>

<a href="ShoppingCart.do" class="title">My Cart</a>

</div>

<div id = "pic">

<a class="H3">Category</a>

<ul id="tabs">

<li><a href="Checkout.do" class="all-tabs">Check out</a></li>

</ul>

</div>

<div id="cata">

<div class="choice">

<a href="Item.do?value=All" class="fill-div">All</a><br />

</div>

<c:if test="${!empty result}">

<c:forEach items="${result}" var="st">

<div class="choice">

<a href="Item.do?value=${st.name }" class="fill-div">${st.name }

<p>

${st.description }

</p>

</a>

</div>

</c:forEach>

</c:if>

</div>

</BODY>

</html>

</jsp:root>

Category (css)

@charset "UTF-8";

#tabs li {

float: right;

margin-right: 2%

}

#tabs {

position: absolute;

top: 15px;

right:20px;

font-family: sans-serif;

text-decoration-line: none;

list-style-type: none;

margin-right:10px;

margin-top:40px;

}

#title {

height: 25px;

width: 100%;

}

#pic {

white-space: nowrap;

background-image:

url("https://cmkt-image-prd.global.ssl.fastly.net/0.1.0/ps/3667914/1820/1213/m1/fpnw/wm1/mm0k1z4eopjimitemdhs6fmiqbvykaf8eubhbyldigxeuwfu2irobuuqz5pjkc1q-.jpg?1512111122&s=54f66daefac23151f5aa1b08d3c76cca");

background-size: cover;

/\* display: flex; \*/

background-repeat: no-repeat;

width:100%;

height:200px;

}

H3 {

position: absolute;

left: 45%;

}

#cata {

position: absolute;

width: 100%;

float:left;

top: 40%;

}

.home {

color: #f75000;

text-decoration-line: none;

}

.sign, .title {

color: #f75000;;

float: right;

text-decoration-line: none;

}

.H3 {

width: 200px;

height:30x;

border-radius: 20px;

margin-left: 30px;

margin-top:30px;

color:white;

font-size:48px;

font-family: sans-serif;

font-weight: 800;

text-shadow: 1px 1px 1px #ff0000;

float:left;

}

.choice {

/\* Internet Explorer 10 \*/

display:-ms-flexbox;

-ms-flex-pack:center;

-ms-flex-align:center;

/\* Firefox \*/

display:-moz-box;

-moz-box-pack:center;

-moz-box-align:center;

/\* Safari, Chrome, and Opera \*/

display:-webkit-box;

-webkit-box-pack:center;

-webkit-box-align:center;

/\* W3C \*/

display:box;

box-pack:center;

box-align:center;

width: 200px;

height: 200px;

margin-top: 15px;

border: 2px solid;

border-color: #f75000;

padding: 10px;

box-shadow: 3px 3px #FFA54F;

float: left;

border-radius: 20px;

margin-left: 5%;

text-align: center;

}

.fill-div {

/\* display: block; \*/

/\* height: 100%; \*/

/\* width: 100%; \*/

font-size:36px;

font-family: sans-serif;

font-weight: 800;

/\* margin-top: 100px; \*/

text-decoration-line: none;

}

p {

font-family: sans-serif;

font-weight: 500;

font-size:16px;

}

.title:after {

content: " | ";

margin-left: 10px;

margin-right: 10px;

}

.all-tabs {

color: white;

float: right;

font-weight: 550;

font-size:20px;

/\* text-decoration-line: none; \*/

line-height:1.3;

}

Checkout (Servlet)

package ctrl;

import java.io.IOException;

import java.io.Writer;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

import model.ItemBean;

@WebServlet("/Checkout.do")

public class Checkout extends HttpServlet {

private static final long serialVersionUID = 1L;

public Checkout() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Engine engine = new Engine();

String oAuth = "https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back="+request.getRequestURL().toString();

String user ="";

String name = "";

double the\_shipping = 5.0;

HttpSession session;

session = request.getSession();

if(session.getAttribute("account")==null) {

if((request.getQueryString() == null)) {

response.sendRedirect(oAuth);

}else {

user = request.getParameter("user");

name = request.getParameter("name");

session.setAttribute("account", user);

session.setAttribute("name", name);

String user\_name = (String)session.getAttribute("account");

request.setAttribute("username", user\_name);

List<ItemBean> orders = engine.result(session);

request.setAttribute("result", orders);

double subtotal = engine.calSubTotal(orders);

double total= engine.calTotal(subtotal);

request.setAttribute("sub\_total", subtotal);

if(subtotal >= 100)

the\_shipping = 0.00;

request.setAttribute("shipping", the\_shipping);

request.setAttribute("hst", Engine.HST);

request.setAttribute("total", total);

response.sendRedirect(request.getRequestURL().toString());

}

}else {

String user\_name = (String)session.getAttribute("account");

request.setAttribute("username", user\_name);

List<ItemBean> orders = engine.result(session);

request.setAttribute("result", orders);

Double subtotal = engine.calSubTotal(orders);

double total = engine.calTotal(subtotal);

double hst = subtotal \* Engine.HST;

request.setAttribute("sub\_total", subtotal);

if(subtotal >= 100)

the\_shipping = 0.00;

request.setAttribute("shipping", the\_shipping);

request.setAttribute("hst", hst);

request.setAttribute("total", total);

this.getServletContext().getRequestDispatcher("/Checkout.jspx").forward(request, response);

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

Checkout (jspx)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt" version="2.0">

<jsp:directive.page contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" session="false" />

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link href="res/Checkout.css" rel="stylesheet" type="text/css" />

<title>Food R Us</title>

</head>

<BODY>

<div id="title">

<a href="eFoods" class="home">HOME</a> <a class="sign">${username}</a>

<a href="MyAccount.do" class="title">P/O Page</a> <a

href="ShoppingCart.do" class="title">My Cart</a>

</div>

<div id="the-head">

<a class="caption">Check Out</a>

<ul id="tabs">

<li><a href="Category.do" class="all-tabs">Category</a></li>

<li><a href="ShoppingCart.do" class="all-tabs">Shopping cart</a></li>

</ul>

</div>

<c:if test="${!empty result}">

<table class="list">

<colgroup align="left">

<col width="30%" />

<col width="30%" />

<col width="38%" />

</colgroup>

<tr align="left">

<th>Name</th>

<th>Qty</th>

<th>Total for product</th>

</tr>

<tr align="left">

<td colspan="3"><hr size="1px" /></td>

</tr>

<c:forEach var="st" items="${result}">

<tr class="list-line">

<td>${st.name }</td>

<td>${st.qty}</td>

<td>$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${st.qty \* st.price }" /></td>

</tr>

</c:forEach>

<c:if test="${result != null}">

<tr align="left">

<td colspan="3"><hr size="1px" /></td></tr>

</c:if> </table></c:if>

<table cellpadding="0" cellspacing="2" class="table-total">

<tr>

<td><u>Price calculation:</u><br /></td></tr>

<tr>

<td class="col1"><strong>Sub Total:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${sub\_total }" /></td>

</tr>

<tr>

<td class="col1"><strong>HST:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${hst }" /></td>

</tr>

<tr>

<td class="col1"><strong>Shipping Cost:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${shipping }" /></td>

</tr>

<tr>

<td colspan="2"><hr size="1px" width="100%" /></td>

</tr>

<tr>

<td class="col1"><strong>Total:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${total }" /></td>

</tr>

</table>

<c:if test="${!empty result}">

<form method="POST" action="Confirm.do">

<input class="button" value="Confirm Order" name="confirm" type="submit" />

</form>

</c:if>

<form method="POST" action="Category.do">

<input class="button" value="Continue Shopping" type="submit" />

</form>

<c:if test="${!empty result}">

<form method="POST" action="Cancel.do">

<input class="button" value="Cancel Order" name="cancel" type="submit" />

</form>

</c:if>

</BODY>

</html>

</jsp:root>

Checkout (css)

@charset "UTF-8";

#tabs li {

float: right;

margin-right: 2%

}

#tabs {

position: absolute;

top: 15px;

right:20px;

font-family: sans-serif;

text-decoration-line: none;

list-style-type: none;

margin-right:10px;

margin-top:40px;

}

.caption {

width: 200px;

height:30x;

border-radius: 20px;

margin-left: 30px;

margin-top:30px;

color:white;

font-size:48px;

font-family: sans-serif;

font-weight: 800;

text-shadow: 1px 1px 1px #ff0000;

float:left;

}

#title {

height: 25px;

width: 100%;

}

.home {

color: #f75000;

text-decoration-line: none;

}

.sign, .title {

color: #f75000;

float: right;

text-decoration-line: none;

}

.title:after {

content: " | ";

margin-left: 10px;

margin-right: 10px;

}

.all-tabs {

color: white;

float: right;

font-weight: 550;

font-size:20px;

/\* text-decoration-line: none; \*/

line-height:1.3;

}

#the-head {

white-space: nowrap;

background-image:

url("https://cmkt-image-prd.global.ssl.fastly.net/0.1.0/ps/3667914/1820/1213/m1/fpnw/wm1/mm0k1z4eopjimitemdhs6fmiqbvykaf8eubhbyldigxeuwfu2irobuuqz5pjkc1q-.jpg?1512111122&s=54f66daefac23151f5aa1b08d3c76cca");

background-size: cover;

/\* display: flex; \*/

background-repeat: no-repeat;

width:100%;

height:200px;

}

table {

width: 89%;

margin-top: 30px;

margin-left: 5%;

margin-right:5%;

}

.list-line {

height:30px;

}

.table-total {

width: 500px;

}

.col1 {

width: 300px;

text-align:left;

}

.col2 {

width:100px;

text-align:right;

}

.button {

margin-top: 10px;

margin-left: 5%;

}

Confirm (Servlet)

package ctrl;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

@WebServlet("/Confirm.do")

public class Confirm extends HttpServlet {

private static final long serialVersionUID = 1L;

public Confirm() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String account = (String) request.getSession().getAttribute("account");

String name = (String) request.getSession().getAttribute("name");

HttpSession session;

session = request.getSession();

Engine engine = new Engine();

try

{

engine.WritePO(account, name, session);

request.setAttribute("account", account);

} catch (Exception e)

{

// TODO Auto-generated catch block

e.printStackTrace();

}

this.getServletContext().getRequestDispatcher("/MyAccount.do").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

Item (Servlet)

package ctrl;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

import model.ItemBean;

@WebServlet("/Item.do")

public class Item extends HttpServlet {

private static final long serialVersionUID = 1L;

public Item() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Engine engine = new Engine();

String category = request.getParameter("value");

List<ItemBean> items = new ArrayList<ItemBean>();

HttpSession session;

session = request.getSession();

if(category != null) {

if(category.equals("All")) {

items = Engine.doItem("");

}

else {

items = Engine.doItem(category);

}

session.setAttribute("cat", category);

}

String result = request.getQueryString();

if(result != null && request.getParameter("value") == null) {

if(request.getParameter("auth") == null)

{

String the\_username = request.getParameter("user");

String name = request.getParameter("name");

session.setAttribute("account", the\_username);

session.setAttribute("name", name);

String user\_name = (String) session.getAttribute("account");

request.setAttribute("username", user\_name);

request.setAttribute("result", items );

request.setAttribute("category", session.getAttribute("cat"));

response.sendRedirect(request.getRequestURL().toString() + "?value=" + session.getAttribute("cat"));

}

else

{

response.sendRedirect("https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back="+request.getRequestURL().toString());

}

} else if(request.getParameter("value") != null)

{

if(request.getParameter("search")== null ) {

request.setAttribute("result", items );

}

else {

String item=request.getParameter("itemName");

request.setAttribute("result", engine.doSearchOnList(items,item));}

request.setAttribute("category", session.getAttribute("cat"));

String user\_name = (String)session.getAttribute("account");

request.setAttribute("username", user\_name); this.getServletContext().getRequestDispatcher("/Item.jspx").forward(request, response);}}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);}}

Item (jspx)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt" version="2.0">

<jsp:directive.page contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" session="false" />

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:c="http://java.sun.com/jsp/jstl/core">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Food R Us</title>

<link href="res/item.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div id="title">

<a href="eFoods" class="home">HOME</a>

<c:if test="${username == null}">

<a href="Item.do?auth=y" class="sign">Sign In</a>

</c:if>

<c:if test="${username != null}">

<a class="sign">${username}</a>

</c:if>

<a href="MyAccount.do" class="title">P/O Page</a>

<a href="ShoppingCart.do" class="title">My Cart</a>

</div>

<div id="the-head">

<a class="cata">${category}</a>

<ul id="tabs">

<li><a href="Category.do" class="all-tabs">Category</a></li>

<li><a href="Checkout.do" class="all-tabs">Check out</a></li>

</ul>

</div>

<div id="engine"><form method="POST" action="" role="form">

<input type="text" name="itemName" size="90%" placeholder="search for item..">

</input>

<span style="margin:5px;"><input type="submit" value="Search" name="search"></input></span>

</form> </div>

<div id="content">

<c:if test="${!empty result}">

<c:set var="count" value="0" scope="page" />

<table width="98%">

<colgroup align="left">

<col width="30%" />

<col width="10%" />

<col width="10%" />

<col align="center" width="10%" />

<col width="28%" />

</colgroup>

<tr align="left">

<th>Name</th>

<th>Number</th>

<th align="center">Price</th>

<th>Qty</th>

<th>Qty Purchase</th>

</tr>

<c:forEach var="st" items="${result}">

<tr>

<td>${st.name }</td>

<td>${st.number }</td>

<td align="center"><fmt:formatNumber type="number" minFractionDigits = "2" maxFractionDigits = "2"value="${st.price }" /></td>

<td>${st.qty }</td>

<td>

<form method="POST" action="ShoppingCart.do">

<label> <input size="3" name="qty"value="1" type="text" /> <input value="${st.name } "

name="name" type="hidden" /> <input value="${st.number } "

name="number" type="hidden" /> <input value="${cat }"

name="return" type="hidden" /> <input value="${st.price }"

name="price" type="hidden" /></label> <input value="Add To Cart" name="add"

type="submit" />

</form>

</td>

<c:set var="count" value="${count + 1}" scope="page" />

</tr>

</c:forEach>

</table>

<div style="width: 98%; float: left">

<hr />

</div>

</c:if>

</div>

</body>

</html>

</jsp:root>

Item (css)

@charset "UTF-8";

#tabs li {

float: right;

margin-right: 2%

}

#tabs {

position: absolute;

top: 15px;

right:20px;

font-family: sans-serif;

text-decoration-line: none;

list-style-type: none;

margin-right:10px;

margin-top:40px;

}

#title {

height: 25px;

width: 100%;

}

.home {

color: #f75000;

text-decoration-line: none;

}

.cata {

width: 200px;

height:30x;

border-radius: 20px;

margin-left: 30px;

margin-top:30px;

color:white;

font-size:48px;

font-family: sans-serif;

font-weight: 800;

text-shadow: 1px 1px 1px #ff0000;

float:left;

}

.sign, .title {

color: #f75000;

float: right;

text-decoration-line: none;

}

.title:after {

content: " | ";

margin-left: 10px;

margin-right: 10px;

}

#the-head {

white-space: nowrap;

background-image:

url("https://cmkt-image-prd.global.ssl.fastly.net/0.1.0/ps/3667914/1820/1213/m1/fpnw/wm1/mm0k1z4eopjimitemdhs6fmiqbvykaf8eubhbyldigxeuwfu2irobuuqz5pjkc1q-.jpg?1512111122&s=54f66daefac23151f5aa1b08d3c76cca");

background-size: cover;

/\* display: flex; \*/

background-repeat: no-repeat;

width:100%;

height:200px;

}

td .price {

text-align: right;

}

#content {

position: absolute;

top: 32%;

padding: 1%;

margin-top: 2%;

width: 50%;

left: 25%;

border: 1px solid;

border-radius: 20px;

}

#engine {

position: absolute;

top: 20%;

left: 25%;

}

.all-tabs {

color: white;

float: right;

font-weight: 550;

font-size:20px;

/\* text-decoration-line: none; \*/

line-height:1.3;

}

input[name="add"] {

float: right;

margin-left: 3%;

}

MyAccount (Servlet)

package ctrl;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

@WebServlet("/MyAccount.do")

public class MyAccount extends HttpServlet {

private static final long serialVersionUID = 1L;

public MyAccount() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String account = (String) request.getSession().getAttribute("account");

Engine engine = new Engine();

if(account!=null) {

request.setAttribute("username", account);

List<String> POlist = engine.resultList(account);

request.setAttribute("result", POlist);

this.getServletContext().getRequestDispatcher("/MyAccount.jspx").forward(request, response);

}

else {

if((request.getQueryString() == null)) {

String oAuth = "https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back="+request.getRequestURL().toString();

response.sendRedirect(oAuth);

}

else {

String user = request.getParameter("user");

String name = request.getParameter("name");

request.getSession().setAttribute("name", name);

request.getSession().setAttribute("account", user);

String result = request.getRequestURL().toString();

response.sendRedirect(result);

}

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

MyAccount (jspx)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt" version="2.0">

<jsp:directive.page contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" session="false" />

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:c="http://java.sun.com/jsp/jstl/core">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link href="res/account.css" rel="stylesheet" type="text/css" />

<title>Food R Us</title>

</head>

<body>

<div id="title">

<a href="eFoods" class="home">HOME</a>

<c:if test="${username == null}">

<a href="Item.do?auth=y" class="sign">Sign In</a>

</c:if>

<c:if test="${username != null}">

<a class="sign">${username}</a>

</c:if>

<a href="MyAccount.do" class="title">P/O Page</a>

<a href="ShoppingCart.do" class="title">My Cart</a>

</div>

<div id="the-head">

<a class="order\_his">Order History</a>

<ul id="tabs">

<li><a href="Category.do" class="all-tabs">Category</a></li>

</ul>

</div>

<div id="content">

<c:if test="${!empty result}">

<c:forEach var="st" items="${result}">

<a href="PODisplay.do?view=${st }" target='\_blank'>${st }</a><br/>

</c:forEach>

</c:if>

<c:if test="${empty result}">

<i> No POs found! Looks like you need to shop more! :)</i>

</c:if></div></body></html></jsp:root>

account (css)

@charset "UTF-8";

#tabs li {

float: right;

margin-right: 2%

}

#tabs {

position: absolute;

top: 15px;

right:20px;

font-family: sans-serif;

text-decoration-line: none;

list-style-type: none;

margin-right:10px;

margin-top:40px;

}

#title {

height: 25px;

width: 100%;

}

.home {

color: #f75000;

text-decoration-line: none;

}

.order\_his {

width: 200px;

height:30x;

border-radius: 20px;

margin-left: 30px;

margin-top:30px;

color:white;

font-size:48px;

font-family: sans-serif;

font-weight: 800;

text-shadow: 1px 1px 1px #ff0000;

float:left;

}

.sign, .title {

color: #f75000;

float: right;

text-decoration-line: none;

}

.title:after {

content: " | ";

margin-left: 10px;

margin-right: 10px;

}

#the-head {

white-space: nowrap;

background-image:

url("https://cmkt-image-prd.global.ssl.fastly.net/0.1.0/ps/3667914/1820/1213/m1/fpnw/wm1/mm0k1z4eopjimitemdhs6fmiqbvykaf8eubhbyldigxeuwfu2irobuuqz5pjkc1q-.jpg?1512111122&s=54f66daefac23151f5aa1b08d3c76cca");

background-size: cover;

/\* display: flex; \*/

background-repeat: no-repeat;

width:100%;

height:200px;

}

#content {

position: absolute;

top: 32%;

padding: 1%;

margin-top: 2%;

width: 50%;

left: 25%;

border: 1px solid;

border-radius: 20px;

line-height:1.5;

}

#engine {

position: absolute;

top: 20%;

left: 25%;

}

.all-tabs {

color: white;

float: right;

font-weight: 550;

font-size:20px;

/\* text-decoration-line: none; \*/

line-height:1.3;

}

PODisplay(Servlet)

package ctrl;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.io.StringReader;

import java.io.Writer;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.InputSource;

import org.xml.sax.SAXException;

@WebServlet("/PODisplay.do")

public class PODisplay extends HttpServlet {

private static final long serialVersionUID = 1L;

public PODisplay() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Writer out = response.getWriter();

String poName = request.getParameter("view");

String filename = "PO/" + poName + ".xml";

File fXmlFile = new File(filename);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder;

String html = "<html><body>";

try

{

dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(fXmlFile);

// ----------------- root element: order -------------------

Element order = doc.getDocumentElement();

String id = order.getAttribute("id");

String date = order.getAttribute("submitted");

// -----------------customer node -------------------

Element customer = (Element) order.getElementsByTagName("customer").item(0);

String account = customer.getAttribute("account");

html+= "<h3> Details of Order:</h3>";

html+= "account number: " + account +"<br/>";

html+= "order number: " + id + "<br/>";

html+= "order date: " + date + "<br/>";

html += "</body></html>";

html+= "<br/><h3> Your order items:</h3>";

html+="<table><tr><th>Product ID</th><th>Name</th><th>Price</th><th>Quantity</th><th>Extended</th></tr>";

// -----------------item node -------------------

// Element items = (Element) order.getElementsByTagName("items");

NodeList item = doc.getElementsByTagName("item");

for(int i=0;i<item.getLength();i++) {

Node nitem = item.item(i);

Element eitem = (Element) nitem;

String pid = eitem.getAttribute("number");

String name = eitem.getElementsByTagName("name").item(0).getTextContent();

String price = eitem.getElementsByTagName("price").item(0).getTextContent();

String quantity = eitem.getElementsByTagName("quantity").item(0).getTextContent();

String extended = eitem.getElementsByTagName("extended").item(0).getTextContent();

html += "<tr><td>"+ pid+ "</td><td>" + name+ "</td><td>" + price +

"</td><td>" + quantity + "</td><td>" + extended+ "</td></tr>";

}

html += "</table>";

String total = order.getElementsByTagName("total").item(0).getTextContent();

String shipping = order.getElementsByTagName("shipping").item(0).getTextContent();

String hst = order.getElementsByTagName("HST").item(0).getTextContent();

String grandtotal = order.getElementsByTagName("grandTotal").item(0).getTextContent();

int pos = hst.indexOf(".");

hst = hst.substring(0, pos + 3);

pos = grandtotal.indexOf(".");

grandtotal = grandtotal.substring(0, pos + 3);

html +="</br>Sub-Total: $" + total + "<br/>";

html +="HST: $" + hst + "<br/>";

html +="Shipping: $" + shipping + "<br/>";

html +="Grand Total:$" + grandtotal + "<br/>";

html += "</body></html>";

out.write(html);

} catch (ParserConfigurationException | SAXException e)

{

html += "<p><a href=' Login.do'>Back to Dashboard</a></p>";

html += "<p>Error " + e.getMessage() + "</p>";

out.write(html);

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

ShoppingCart(Servlet)

package ctrl;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

import model.ItemBean;

@WebServlet("/ShoppingCart.do")

public class ShoppingCart extends HttpServlet {

private static final long serialVersionUID = 1L;

public ShoppingCart() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Engine engine = new Engine();

HttpSession session;

session = request.getSession();

double the\_shipping = 5.0;

String result = request.getQueryString();

if(result != null) {

if(request.getParameter("auth") == null)

{

String the\_username = request.getParameter("user");

String name = request.getParameter("name");

session.setAttribute("account", the\_username);

session.setAttribute("name", name);

String user\_name = (String) session.getAttribute("account");

request.setAttribute("username", user\_name);

List<ItemBean> orders = engine.result(session);

request.setAttribute("result", orders);

double subtotal = engine.calSubTotal(orders);

double total= engine.calTotal(subtotal);

double hst = subtotal \* Engine.HST;

request.setAttribute("sub\_total", subtotal);

if(subtotal >= 100)

the\_shipping = 0.00;

request.setAttribute("shipping", the\_shipping);

request.setAttribute("hst", hst);

request.setAttribute("total", total);

response.sendRedirect(request.getRequestURL().toString());

}

else

{

response.sendRedirect("https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back="+request.getRequestURL().toString());

}

} else {

String user\_name = (String)session.getAttribute("account");

request.setAttribute("username", user\_name);

if(request.getParameter("add") == null) {

List<ItemBean> orders = engine.result(session);

request.setAttribute("result", orders);

double subtotal = engine.calSubTotal(orders);

double total= engine.calTotal(subtotal);

double hst = subtotal \* Engine.HST;

request.setAttribute("sub\_total", subtotal);

if(subtotal >= 100)

the\_shipping = 0.00;

request.setAttribute("shipping", the\_shipping);

request.setAttribute("hst", hst);

request.setAttribute("total", total);

this.getServletContext().getRequestDispatcher("/ShoppingCart.jspx").forward(request, response);

}else {

String qty = request.getParameter("qty");

String name = request.getParameter("name");

String number = request.getParameter("number");

String price = request.getParameter("price");

String category = request.getParameter("return");

request.setAttribute("qty",qty);

request.setAttribute("name", name);

request.setAttribute("number", number);

request.setAttribute("price", price);

request.setAttribute("category", category);

try {

List<ItemBean> orders = engine.doAdd(name,number,qty,price, session);

request.setAttribute("result", orders);

double subtotal = engine.calSubTotal(orders);

double total= engine.calTotal(subtotal);

double hst = subtotal \* Engine.HST;

request.setAttribute("sub\_total", subtotal);

if(subtotal >= 100)

the\_shipping = 0.00;

request.setAttribute("shipping", the\_shipping);

request.setAttribute("hst", hst);

request.setAttribute("total", total);

}

catch(Exception e){

request.setAttribute("result",engine.result(session));

request.setAttribute("error",e.getMessage());

request.setAttribute("add", null);

}

this.getServletContext().getRequestDispatcher("/ShoppingCart.jspx").forward(request, response);

}

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

ShoppingCart(jspx)

<?xml version="1.0" encoding="ISO-8859-1" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt" version="2.0">

<jsp:directive.page contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" session="false" />

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link href="res/shoppingcart.css" rel="stylesheet" type="text/css" />

<title>Food R Us</title>

</head>

<BODY>

<div id="title">

<a href="eFoods" class="home">HOME</a>

<c:if test="${username == null}">

<a href="ShoppingCart.do?auth=y" class="sign">Sign In</a>

</c:if>

<c:if test="${username != null}">

<a class="sign">${username}</a>

</c:if>

<a href="MyAccount.do" class="title">P/O Page</a> <a

href="ShoppingCart.do" class="title">My Cart</a>

</div>

<div id="the-head">

<a class="caption">Shopping cart</a>

<ul id="tabs">

<li><a href="Category.do" class="all-tabs">Category</a></li>

<li><a href="Checkout.do" class="all-tabs">Check out</a></li>

</ul>

</div>

<form method="POST" action="Update.do">

<c:set var="count" value="1" scope="page" />

<c:if test="${!empty result}">

<table>

<colgroup align="left">

<col width="30%" />

<col width="20%" />

<col width="10%" />

<col width="10%" />

<col width="18%" />

<col width="10%" />

</colgroup>

<tr align="left">

<th>Name</th>

<th>Number</th>

<th>Price</th>

<th>Qty</th>

<th>Total for product</th>

<th>Action</th>

</tr>

<tr align="left">

<td colspan="6"><hr size="1px" /></td>

</tr>

<c:forEach var="st" items="${result}">

<tr>

<td>${st.name }</td>

<td>${st.number }</td>

<td>$ <fmt:formatNumber type="number" minFractionDigits="2"maxFractionDigits="2" value="${st.price }" /></td>

<td><input size="3" value="${st.qty }" name="${st.number }"type="text" /></td>

<td>$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${st.qty \* st.price }" /></td>

<td><input value="${st.number }" name="delete"type="checkbox" />Delete</td>

<input value="${st.number }" name="item${count }" type="hidden" />

<c:set var="count" value="${count + 1}" scope="page" />

</tr>

</c:forEach>

<c:if test="${result != null}">

<tr align="left">

<td colspan="6"><hr size="1px" /></td>

</tr>

</c:if>

</table>

<input value="${count }" name="itemNo" type="hidden" />

<input value="${category }" name="category" type="hidden" />

<input class="button" value="Update" type="submit" />

</c:if>

</form>

<p></p>

<c:if test="${!empty advert }">

<div class="advert">

You might also be interested in.. <br/>

<strong> ${advert.name } </strong> | number: ${advert.number} | price:

<fmt:formatNumber type="number" maxFractionDigits="2"value="${advert.price }" /> |

quantity: ${advert.qty }

<form method="POST" action="ShoppingCart.do">

<label> Quantity to Purchase: <input size="3" name="qty"value="1" type="text" /> <input value="${advert.name } "

name="name" type="hidden" /> <input value="${advert.number } "

name="number" type="hidden" /> <input value="${cat}"name="return" type="hidden" /> <input value="${advert.price }"name="price" type="hidden" /></label> <input value="Add To Cart" name="add"type="submit" /></form></div>

</c:if>

<c:if test="${!empty error}">

<div style="margin: 10px; text-align: center; color: red">

${error} <br />

</div>

</c:if>

<p></p>

<c:if test="${empty error}">

<table cellpadding="0" cellspacing="2" class="table-total">

<tr>

<td><u>Price calculation:</u><br /></td>

</tr>

<tr>

<td class="col1"><strong>Sub Total:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number"

minFractionDigits="2" maxFractionDigits="2" value="${sub\_total }" />

</td>

</tr>

<tr>

<td class="col1"><strong>HST:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number"

minFractionDigits="2" maxFractionDigits="2" value="${hst }" /></td>

</tr>

<tr>

<td class="col1"><strong>Shipping Cost:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number" minFractionDigits="2"

maxFractionDigits="2" value="${shipping }" /></td>

</tr>

<tr>

<td colspan="2"><hr size="1px" /></td>

</tr>

<tr>

<td class="col1"><strong>Total:</strong></td>

<td class="col2">$ <fmt:formatNumber type="number"minFractionDigits="2" maxFractionDigits="2" value="${total }" /></td>

</tr>

</table>

</c:if>

<form method="POST" action="Item.do?value=${category }">

<input class="button" value="Continue Shopping" type="submit" />

</form>

<c:if test="${!empty result}">

<form method="POST" action="Checkout.do">

<input class="button" value="Check out" type="submit" />

</form>

</c:if>

</BODY>

</html>

</jsp:root>

ShoppingCart(css)

@charset "UTF-8";

#tabs li {

float: right;

margin-right: 2%

}

#tabs {

position: absolute;

top: 15px;

right:20px;

font-family: sans-serif;

text-decoration-line: none;

list-style-type: none;

margin-right:10px;

margin-top:40px;

}

#tabs {

list-style-type: none;

}

#title {

height: 25px;

width: 100%;

}

.home {

color: #f75000;

text-decoration-line: none;

}

.caption {

width: 200px;

height:30x;

border-radius: 20px;

margin-left: 30px;

margin-top:30px;

color:white;

font-size:48px;

font-family: sans-serif;

font-weight: 800;

text-shadow: 1px 1px 1px #ff0000;

float:left;

}

.sign, .title {

color: #f75000;

float: right;

text-decoration-line: none;

}

.title:after {

content: " | ";

margin-left: 10px;

margin-right: 10px;

}

.all-tabs {

color: white;

float: right;

font-weight: 550;

font-size:20px;

/\* text-decoration-line: none; \*/

line-height:1.3;

}

#the-head {

white-space: nowrap;

background-image:

url("https://cmkt-image-prd.global.ssl.fastly.net/0.1.0/ps/3667914/1820/1213/m1/fpnw/wm1/mm0k1z4eopjimitemdhs6fmiqbvykaf8eubhbyldigxeuwfu2irobuuqz5pjkc1q-.jpg?1512111122&s=54f66daefac23151f5aa1b08d3c76cca");

background-size: cover;

/\* display: flex; \*/

background-repeat: no-repeat;

width:100%;

height:200px;

}

table {

width: 89%;

margin-top: 30px;

margin-left: 5%;

margin-right:5%;

}

hr {

width:100%;

size: 2px;

}

.button {

margin-top: 10px;

margin-left: 5%;

}

.advert{

margin-top: 10px;

margin-bottom: 10px;

margin-left: 5%;

border-left: 8px solid red;

background-color: #ECECEC;

padding: 10px;

width: 50%;

}

.table-total {

width: 500px;

}

.col1 {

width: 300px;

text-align:left;

}

.col2 {

width:100px;

text-align:right;

}

Update (Servlet)

package ctrl;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import model.Engine;

import model.ItemBean;

@WebServlet("/Update.do")

public class Update extends HttpServlet

{

private static final long serialVersionUID = 1L;

public Update()

{

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

List<ItemBean> orders = new ArrayList<ItemBean>();

Engine engine = new Engine();

HttpSession session;

session = request.getSession();

double the\_shipping = 5.0;

String numOfItem = request.getParameter("itemNo");

if (request.getParameter("delete") != null)

{

String checkbox[] = request.getParameterValues("delete");

engine.doDelete(checkbox, session);

}

try

{

for (int i = 1; i < Integer.parseInt(numOfItem); i++)

{

String name = request.getParameter("item" + i);

String qty = request.getParameter(name);

orders = engine.doUpdate(name, qty, session);

}

request.setAttribute("result", orders);

Double subtotal = engine.calSubTotal(orders);

Double total = engine.calTotal(subtotal);

if(subtotal >= 100)

the\_shipping = 0.00;

request.setAttribute("category", request.getParameter("category"));

request.setAttribute("sub\_total", subtotal);

request.setAttribute("hst", Engine.HST);

request.setAttribute("shipping", the\_shipping);

request.setAttribute("total", total);

} catch (Exception e)

{

request.setAttribute("result", engine.result(session));

request.setAttribute("error", e.getMessage());

request.setAttribute("add", null);

}

String user\_name = (String)session.getAttribute("account");

request.setAttribute("username", user\_name);

this.getServletContext().getRequestDispatcher("/ShoppingCart.jspx").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

doGet(request, response);

}

}

Engine (Model)

package model;

import java.io.File;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.util.TreeMap;

import javax.servlet.http.HttpSession;

public class Engine

{

public static final String LOCAL\_DB = "jdbc:derby://localhost:64413/EECS;user=student;password=secret";

public static final double HST= 0.13;

public Engine() //when using singleton, always make constructor private

{

}

public static void createOrder(HttpSession session) {

Map<String, ItemBean> emptyorder = new HashMap<String, ItemBean>();

session.setAttribute("order", emptyorder);

}

public static List<ItemBean> doItem(String category) {

try {

Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

Connection con = DriverManager.getConnection(LOCAL\_DB);

Statement s = con.createStatement();

s.executeUpdate("set schema roumani");

String query = "Select \* FROM Item WHERE NAME LIKE ?";

PreparedStatement p = con.prepareStatement(query);

p.setString(1, "%"+category+"%");

ResultSet r = p.executeQuery();

List<ItemBean> items = new ArrayList<ItemBean>();

while (r.next())

{ ItemBean item = new ItemBean();

item.setNumber(r.getString("NUMBER"));

item.setName(r.getString("NAME"));

item.setQty(r.getInt("QTY"));

item.setPrice(r.getDouble("PRICE"));

items.add(item);

}

r.close(); s.close(); con.close();

return items;

}

catch (Exception e) {

return new ArrayList<ItemBean>();

}

}

//This assumes the number is UNIQUE!

public static ItemBean getItem(String number) {

try {

Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

Connection con = DriverManager.getConnection(LOCAL\_DB);

Statement s = con.createStatement();

s.executeUpdate("set schema roumani");

String query = "Select \* FROM Item WHERE NUMBER=?";

PreparedStatement p = con.prepareStatement(query);

p.setString(1, number);

ResultSet r = p.executeQuery();

r.next();

ItemBean item = new ItemBean();

item.setNumber(r.getString("NUMBER"));

item.setName(r.getString("NAME"));

item.setQty(r.getInt("QTY"));

item.setPrice(r.getDouble("PRICE"));

r.close(); s.close(); con.close();

return item;

}

catch (Exception e) {

return new ItemBean();

}

}

public static List<CategoryBean> doCategory() {

try {

Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();

Connection con = DriverManager.getConnection(LOCAL\_DB);

Statement s = con.createStatement();

s.executeUpdate("set schema roumani");

String query = "Select \* FROM Category";

ResultSet r = s.executeQuery(query);

List<CategoryBean> categories = new ArrayList<CategoryBean>();

while (r.next())

{ CategoryBean category = new CategoryBean();

category.setId(r.getInt("ID"));

category.setName(r.getString("NAME"));

category.setDescription(r.getString("DESCRIPTION"));

category.setPicture(r.getString("PICTURE"));

categories.add(category);

}

r.close(); s.close(); con.close();

return categories;

}

catch (Exception e) {

return new ArrayList<CategoryBean>();

}

}

//Searches over a list for a char set search parameter

public List<ItemBean> doSearchOnList(List<ItemBean> items, String item){

List<ItemBean> relevantItems = new ArrayList<ItemBean>();

if(!items.isEmpty()){

for(ItemBean o : items){

if(o.getName().contains(item)){

relevantItems.add(o);

}

}

HashMap<ItemBean,Integer> value= new HashMap<ItemBean,Integer>();

TreeMap<ItemBean,Integer> sorted\_map = new TreeMap<ItemBean,Integer>(new ValueComparator(value));

// ValueComparater is below this

Iterator<ItemBean> a = relevantItems.iterator();

while(a.hasNext()){

ItemBean itemA= a.next();

int valueA= Math.abs(itemA.getName().compareToIgnoreCase(item));

value.put(itemA, valueA);

}

sorted\_map.putAll(value);

List<ItemBean> listItems = new ArrayList<ItemBean>();

Iterator<Entry<ItemBean, Integer>> b = sorted\_map.entrySet().iterator();

while (b.hasNext()) {

listItems.add((ItemBean) b.next().getKey());

}

return listItems;

}

else {

return items;

}

}

//sort items by proximity

class ValueComparator implements Comparator<ItemBean> {

Map<ItemBean,Integer> map;

public ValueComparator(Map<ItemBean,Integer> map) {

this.map = map;

}

public int compare(ItemBean a, ItemBean b) {

if (map.get(a) >= map.get(b)) {

return -1;

} else {

return 1;

}

}

}

public void doCancel(HttpSession session) {

if(session.getAttribute("order") != null)

{

Map<String, ItemBean> order = (Map<String, ItemBean>) session.getAttribute("order");

order.clear();

session.setAttribute("order", order);

}

}

public Double calSubTotal(List<ItemBean> orders) {

double total=0.0;

for(int i=0;i<orders.size();i++) {

ItemBean order = orders.get(i);

total = total + order.getQty() \* order.getPrice();

}

total = (double)Math.round(total\*100)/100;

return total;

}

public Double calTotal(double subtotal){

double total = 0.0;

total = subtotal + subtotal\*HST;

if(subtotal < 100)

total = total + 5;

total = (double)Math.round(total\*100)/100;

return total;

}

public List<ItemBean> doAdd(String name, String number, String qtyStr, String priceStr, HttpSession session) throws Exception

{

if(session.getAttribute("order") == null)

{

Map<String, ItemBean> order = new HashMap<String, ItemBean>();

session.setAttribute("order", order);

}

Map<String, ItemBean> order = (Map<String, ItemBean>) session.getAttribute("order");

if(qtyStr.matches("[1-9][0-9]\*")) {

int qty= Integer.parseInt(qtyStr);

double price= Double.parseDouble(priceStr);

if(!order.containsKey(number)) {

ItemBean newItem = new ItemBean();

newItem.setName(name);

newItem.setNumber(number);

newItem.setPrice(price);

newItem.setQty(qty);

order.put(number, newItem);

}else {

ItemBean inOrder = order.get(number);

inOrder.setQty(inOrder.getQty() + qty);

order.put(number, inOrder);

}

session.setAttribute("order", order);

return result(session);

}

else{

throw new Exception("Item not added: Error: Quantity must be a number greater than 1");

}

}

public void doDelete(String[] name, HttpSession session){

if(session.getAttribute("order") != null)

{

Map<String, ItemBean> order = (Map<String, ItemBean>) session.getAttribute("order");

for(int i=0;i<name.length;i++) {

order.remove(name[i]);

}

session.setAttribute("order", order);

}

}

public List<ItemBean> doUpdate(String number, String qty, HttpSession session) throws Exception{

if(session.getAttribute("order") == null)

{

Map<String, ItemBean> order = new HashMap<String, ItemBean>();

session.setAttribute("order", order);

}

Map<String, ItemBean> order = (Map<String, ItemBean>) session.getAttribute("order");

if(qty.matches("[0-9]\*")) {

int quantity = Integer.parseInt(qty);

if(order.containsKey(number)) {

if(quantity <= 0) {

order.remove(number);

}

else {

ItemBean inOrder = order.get(number);

inOrder.setQty(quantity);

order.put(number, inOrder);

}

}

session.setAttribute("order", order);

return result(session);

}

else {

throw new Exception("Cart couldn't update: Error: Quantity must be a number greater than 0");

}

}

public List<ItemBean> result(HttpSession session){

Map<String, ItemBean> order = (Map<String, ItemBean>) session.getAttribute("order");

List<ItemBean> orders = new ArrayList<ItemBean>();

if(order != null) {

for (ItemBean o : order.values()) {

orders.add(o);

}

}

return orders;

}

public List<String> resultList(String account){

List<String> POlist = new ArrayList<String>();

String directory = "PO/";

File file = new File(directory);

if(file.list() != null) {

String [] accountPos = file.list();

Pattern p = Pattern.compile("([\*]?po"+account+"\_[0-9]+).xml");

for(int i=0; i< accountPos.length; i++) {

Matcher m = p.matcher(accountPos[i]);

if(m.matches()) {

POlist.add(m.group(1));

}

}

}

return POlist;

}

public void WritePO(String account, String name, HttpSession session) throws Exception {

List<ItemBean> order = this.result(session);

WritePO PO = new WritePO(order, account, name);

PO.WritetoXML();

createOrder(session);

}

}

CategoryBean (Bean)

package model;

public class CategoryBean

{

private int id;

private String name;

private String description;

private String picture;

public int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public String getDescription()

{

return description;

}

public void setDescription(String description)

{

this.description = description;

}

public String getPicture()

{

return picture;

}

public void setPicture(String picture)

{

this.picture = picture;

}

}

ItemBean(Bean)

package model;

public class ItemBean

{

private String number;

private String name;

private double price;

private int qty;

public String getNumber()

{

return number;

}

public void setNumber(String number)

{

this.number = number;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public double getPrice()

{

return price;

}

public void setPrice(double price)

{

this.price = price;

}

public int getQty()

{

return qty;

}

public void setQty(int qty)

{

this.qty = qty;

}

}

WritePO (Model)

package model;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Attr;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

public class WritePO

{

List<ItemBean> orders = new ArrayList<ItemBean>();

String account;

String name;

public WritePO(List<ItemBean> orders, String account, String name) {

this.orders = orders;

this.account = account;

this.name = name;

}

public void WritetoXML() throws TransformerException{

double total = 0.0;

try {

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

// root elements

Document doc = docBuilder.newDocument();

Element rootElement = doc.createElement("order");

doc.appendChild(rootElement);

// set attribute to order element

Attr id = doc.createAttribute("id");

// Id is from total unprocessed in the PO directory..

File file = new File("PO");

if(file.exists()) {

id.setValue(file.list().length+1+ "");

}

else {

id.setValue("1");

}

// was:

//id.setValue("1");

rootElement.setAttributeNode(id);

Attr date = doc.createAttribute("submitted");

date.setValue(new Date().toString());

rootElement.setAttributeNode(date);

// customer elements

Element customer = doc.createElement("customer");

rootElement.appendChild(customer);

// set attribute to customer element

Attr acc = doc.createAttribute("account");

acc.setValue(account);

customer.setAttributeNode(acc);

// customer name elements

Element customer\_name = doc.createElement("name");

customer\_name.appendChild(doc.createTextNode(name));

customer.appendChild(customer\_name);

//items element

Element items = doc.createElement("items");

rootElement.appendChild(items);

for(int i=0;i<orders.size();i++) {

ItemBean order = (ItemBean) orders.get(i);

String itemName = order.getName();

String itemNo = order.getNumber();

double price = (double)Math.round(order.getPrice()\* 100)/100;

int qty = order.getQty();

double extended = price \* qty;

total = total + extended;

//item element

Element item = doc.createElement("item");

items.appendChild(item);

//set attribute to item element

Attr number = doc.createAttribute("number");

number.setValue(itemNo);

item.setAttributeNode(number);

//name element

Element item\_Name = doc.createElement("name");

item\_Name.appendChild(doc.createTextNode(itemName));

item.appendChild(item\_Name);

Element item\_price = doc.createElement("price");

item\_price.appendChild(doc.createTextNode(Double.toString(price)));

item.appendChild(item\_price);

Element item\_qty = doc.createElement("quantity");

item\_qty.appendChild(doc.createTextNode(Integer.toString(qty)));

item.appendChild(item\_qty);

Element item\_extended = doc.createElement("extended");

item\_extended.appendChild(doc.createTextNode(Double.toString(extended)));

item.appendChild(item\_extended);

}

total = (double)Math.round(total\*100)/100;

Element total\_price = doc.createElement("total");

total\_price.appendChild(doc.createTextNode(Double.toString(total)));

rootElement.appendChild(total\_price);

double shipping\_cost = 5.0;

if(total >= 100)

shipping\_cost = 0.0;

Element shipping = doc.createElement("shipping");

shipping.appendChild(doc.createTextNode(Double.toString(shipping\_cost)));

rootElement.appendChild(shipping);

double hst = total \* 0.13;

hst = (double)Math.round(hst\*100)/100;

Element total\_hst = doc.createElement("HST");

total\_hst.appendChild(doc.createTextNode(Double.toString(hst)));

rootElement.appendChild(total\_hst);

Element grandtotal = doc.createElement("grandTotal");

grandtotal.appendChild(doc.createTextNode(Double.toString(hst+total+shipping\_cost)));

rootElement.appendChild(grandtotal);

// write the content into xml file

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

String directory = "PO/";

new File(directory).mkdirs();

int poNumber=1;

String [] accountPos = new File(directory).list();

Pattern p = Pattern.compile("[\*]?po"+account+"\_[0-9]+.xml");

for(int i=0; i< accountPos.length; i++) {

Matcher m = p.matcher(accountPos[i]);

if(m.find()) {

poNumber= poNumber +1;

}

}

String poNumberString= poNumber < 10 ? "0"+poNumber : poNumber+"";

String PO = directory+ "/po"+ account +"\_"+poNumberString+".xml";

StreamResult result = new StreamResult(new File(PO));

transformer.transform(source, result);

}catch (ParserConfigurationException pce) {

pce.printStackTrace();

}

}

}

Analytics (jspx)

<?xml version="1.0" encoding="UTF-8" ?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"

xmlns:c="http://java.sun.com/jsp/jstl/core"

xmlns:fmt="http://java.sun.com/jsp/jstl/fmt"

version="2.0">

<jsp:directive.page contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8" session="false"/>

<jsp:output doctype-root-element="html"

doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"

doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

omit-xml-declaration="true" />

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Analytics</title>

</head>

<body>

Average time to add an item: <fmt:formatNumber

value="${addItem}" maxFractionDigits="2"/> seconds<br/>

Average time to checkout: <fmt:formatNumber

value="${checkout}" maxFractionDigits="2"/> seconds

</body>

</html>

</jsp:root>

Admin(Servlet)

package analytics;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/Admin.do")

public class Admin extends HttpServlet {

private static final long serialVersionUID = 1L;

public Admin() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

Main m= new Main();

request.setAttribute("addItem", m.getAvgAddItem());

request.setAttribute("checkout", m.getAvgCheckOut());

this.getServletContext().getRequestDispatcher("/Analytics.jspx").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

Main(Listener)

package analytics;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

import javax.servlet.ServletRequestAttributeEvent;

import javax.servlet.ServletRequestAttributeListener;

import javax.servlet.ServletRequestEvent;

import javax.servlet.annotation.WebListener;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import javax.servlet.http.HttpSessionActivationListener;

import javax.servlet.http.HttpSessionEvent;

/\*\*

\* Application Lifecycle Listener implementation class Main

\*

\*/

@WebListener

public class Main implements HttpSessionActivationListener, ServletRequestAttributeListener {

Statistics s;

public Main() {

// TODO Auto-generated constructor stub

s = Statistics.getInstance();

}

public void requestInitialized(ServletRequestEvent sre) {

String servletPath= ((HttpServletRequest)sre.getServletRequest()).getServletPath().toString();

if(servletPath.equals("/Checkout.do")){

HttpSession session =((HttpServletRequest)sre.getServletRequest()).getSession();

s.doAvgCheckout(session);

}

}

public double getAvgCheckOut() {

return s.getAvgCheckOut();

}

public double getAvgAddItem() {

return s.getAvgAddItem();

}

public void sessionCreated(HttpSessionEvent se) {

s.addSession(se);

}

}

Statistics( Listener Calculations)

package analytics;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

import javax.servlet.http.HttpSession;

import javax.servlet.http.HttpSessionEvent;

import java.util.HashMap;

public class Statistics

{

private static Statistics instance = null;

private Map<HttpSession, Date> sessionCreation = new HashMap<HttpSession, Date>();

private double avgCheckoutTime =0;

private int totalCheckouts=0;

private double avgAddTime =0;

private int totalUniqueUserItemAdds=0; // condition: an attempt to add an item was made

private Statistics() {

}

public synchronized static Statistics getInstance()

{

if (instance == null) instance = new Statistics();

return instance;

}

public void doAvgCheckout(HttpSession session) {

if(this.sessionCreation.containsKey(session)) {

Date created = this.sessionCreation.get(session);

Date current=new Date();

long time= created.getTime() - current.getTime();

double sum = (this.totalCheckouts \* this.avgCheckoutTime) + time;

this.totalCheckouts++;

this.avgCheckoutTime= sum/this.totalCheckouts;

}

}

public void addSession(HttpSessionEvent se) {

this.sessionCreation.put(se.getSession(), new Date());

}

public double getAvgCheckOut() {

return this.avgCheckoutTime;

}

public double getAvgAddItem() {

return this.avgAddTime;

}

}

Advertisement (Filter)

package adhoc;

import java.io.IOException;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.annotation.WebFilter;

import javax.servlet.http.HttpServletRequest;

import model.Engine;

@WebFilter("/ShoppingCart.do")

public class Advertisement implements Filter {

public Advertisement() {

}

public void destroy() {

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

if(((HttpServletRequest)request).getServletPath().toString().equals("/ShoppingCart.do")){

if(request.getParameter("add") != null) {

String number = request.getParameter("number");

if(number.equals("1409S413 ")) {

request.setAttribute("advert", Engine.getItem("2002H712"));

}}

chain.doFilter(request, response);

}

}

public void init(FilterConfig fConfig) throws ServletException {

}}

ReportGenerator (Main Class)

import java.io.File;

import java.util.ArrayList;

import java.util.List;

public class ReportGenerator

{

public static void main(String[] args)

{

String dir1 = "/home/user/PO";

File file = new File(dir1);

if(file.exists()) {

String [] allPos = new File(dir1).list();

List<String> unprocessedPos= new ArrayList<String>();

for(int i=0; i< allPos.length;i++) {

if(allPos[i].charAt(0)!='\*') {

unprocessedPos.add(allPos[i]);

}

}

Unionizer union = new Unionizer();

union.UnionizeXML(unprocessedPos);

}

}

}

Unionizer (Helper Class)

import java.io.File;

import java.io.IOException;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Attr;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

public class Unionizer

{

/\* Unionizies all POs to obtain the total quantity of each item to

be supplied, it generates a report that lists each item by number

and quantity ONLY, everything else is assumed to be consumer fluff\*/

public void UnionizeXML (List<String> POs) {

try {

DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

// root elements

Document doc = docBuilder.newDocument();

Element rootElement = doc.createElement("orders");

doc.appendChild(rootElement);

Attr total = doc.createAttribute("total");

total.setValue(""+POs.size());

rootElement.setAttributeNode(total);

Attr generated= doc.createAttribute("generated");

generated.setValue(new Date().toString());

rootElement.setAttributeNode(generated);

//unionize items

Element items = doc.createElement("items");

rootElement.appendChild(items);

//create a hashmap of items and traverse through them

HashMap<String,Integer> itemQty = poMarkers(POs);

Iterator it = itemQty.entrySet().iterator();

while (it.hasNext()) {

Map.Entry pair = (Map.Entry)it.next();

Element item = doc.createElement("item");

items.appendChild(item);

Attr number = doc.createAttribute("number");

number.setValue(pair.getKey().toString());

item.setAttributeNode(number);

Element quantity = doc.createElement("quantity");

item.appendChild(quantity);

quantity.appendChild(doc.createTextNode(pair.getValue().toString()));

it.remove();

}

// write the content into xml file

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

String dir1 = "/home/user/POReport/";

new File(dir1).mkdirs();

int rNumber = new File(dir1).list().length;

String rTot= rNumber < 10 ? "0"+rNumber : rNumber+"";

String r = dir1+ "/report\_"+rTot +".xml";

StreamResult result = new StreamResult(new File(r));

transformer.transform(source, result);

}catch (Exception pce) {

pce.printStackTrace();

}

}

private HashMap<String,Integer> poMarkers(List<String> POs) {

String dir2 = "/home/user/PO/";

HashMap<String,Integer> itemQty = new HashMap<String,Integer>();

Iterator<String> it= POs.iterator();

while(it.hasNext()) {

try

{

String item = it.next();

File oldfile= new File(dir2+item);

File newfile = new File(dir2+"\*"+item);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(oldfile);

NodeList iList= doc.getElementsByTagName("item");

for(int i=0;i< iList.getLength();i++) {

Node iNode= iList.item(i);

Element iElement= (Element) iNode;

if(itemQty.containsKey(iElement.getAttribute("number"))) {

int qty = itemQty.get(iElement.getAttribute("number"));

String value= iElement.getElementsByTagName("quantity").item(0).getTextContent();

qty= qty+ Integer.parseInt(value);

itemQty.replace(iElement.getAttribute("number"), qty);

}

else {

String key = iElement.getAttribute("number");

String value= iElement.getElementsByTagName("quantity").item(0).getTextContent();

itemQty.put(key, Integer.parseInt(value));

}

}

oldfile.renameTo(newfile);

} catch (Exception e)

{

e.printStackTrace();

}

}

return itemQty;

}

}