```php
<?php
session_start();

$CONFIG_FILE = "./config/config.json";

$config = get_config($CONFIG_FILE);
$alerts = array();

// Functions & Helpers
function get_config($f) {
  return json_decode(file_get_contents($f));
}

function do_auth($username, $password, $challenge_url, $challenge_file) {
  $ch = curl_init();
  curl_setopt($ch, CURLOPT_URL, $challenge_url);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  curl_setopt($ch, CURLOPT_VERBOSE, true);
  curl_setopt($ch, CURLOPT_TIMEOUT, 5);
  curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 10);
  curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
  curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
  curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");
  curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
  $output = trim(curl_exec($ch));
  $info = curl_getinfo($ch);
  curl_close($ch);
  error_log(print_r($info,true));
  if ($info["http_code"] == 403 || $info["http_code"] == 401) {
    return array("message" => "Invalid Username or Password.");
  } else if ($info["http_code"] != 200) {
    return array("message" => "Authentication service is experiencing problems at the moment. Please try again later.");
  }

  $challenge = trim(file_get_contents($challenge_file));

  if (empty($challenge) || empty($output)) {
    return array("message" => "Authentication service is not configured properly.");
  }

  if ($challenge != $output) {
    return array("message" => "Authentication service is not active at the moment.");
  }

  return true;
}

function validate_payload($raw_payload, $raw_signature, $shared_key) {
  $remote_payload = base64_decode($_REQUEST[’payload’]);
  $signature = hash_hmac(’sha256’, $_REQUEST[’payload’], $shared_key);
  if (strtolower($signature) == strtolower($raw_signature)) {
    parse_str($remote_payload, $payload_array);
    return $payload_array;
  } else {
    return false;
  }
}

function get_redirect_url($base_url, $nonce, $data, $shared_key ) {
  $data["nonce"] = $nonce;
  $raw_payload = http_build_query($data);
  $payload = base64_encode($raw_payload);
  $signature = strtolower(hash_hmac(’sha256’, $payload, $shared_key));
  $query_string = "payload=" . urlencode($payload) . "&signature=" . urlencode($signature);
  return $base_url . "?" . $query_string;
}
```

```php

function get_full_name($username) {
  $system_name = exec('getent passwd ' . escapeshellarg($username) . ' | cut -d: -f5 | cut -d, -f1');
  return empty($system_name) ? "Unknown Name" : $system_name;
}
// End functions & helpers

if  (!empty($_REQUEST['submit'])) {
  $payload = validate_payload($_REQUEST['payload'], $_REQUEST['signature'], $config->sharedKey);
  $username = $_REQUEST['username'];
  $password = $_REQUEST['password'];

  if (! $payload === false)
    $authenticate = do_auth($username,$password,$config->challengeURL, $config->challengeFile);
  else
    $authenticate = array("message" => "Login request is expired. Please go back to the site and click long again.");

  if ($authenticate === true) {
    http_response_code(200);
    $data = array();
    $data["username"] = $username;
    $data["fullname"] = get_full_name($username);

    //Testing
    $ALT_REDIRECT = empty($_SESSION["REDIRECT_TO"]) ? $config->redirectURL : $_SESSION["REDIRECT_TO"];
    //End testing
    error_log("Redirect To :   " . $ALT_REDIRECT);

    $url = get_redirect_url($ALT_REDIRECT, $payload["nonce"], $data, $config->sharedKey);
    header('Location: '. $url);
    die();
  } else {
    http_response_code(403);
    array_push($alerts, $authenticate["message"]);
  }
} else {
  if (empty($_REQUEST['payload']) || empty($_REQUEST['signature'])) {
    $error = "Please provide a valid payload and a signature to use the authentication service.";
  } else {
    $payload = validate_payload($_REQUEST['payload'], $_REQUEST['signature'], $config->sharedKey);
    if ($payload === false) {
      $error = "Login request is expired. Please go back to the site and click login again.";
    }
  }
  if (!empty($_SERVER["HTTP_REFERER"]) && !empty($_REQUEST["redirect"])) {
    $_SESSION["REDIRECT_TO"] = $_SERVER["HTTP_REFERER"] . $_REQUEST["redirect"];
  }
}
?>
<!doctype html>
<html class="no-js">
<head>
  <meta charset="utf-8">
  <title>Auth</title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width">
  <link rel="shortcut icon" href="/favicon.ico">
  <!-- Place favicon.ico and apple-touch-icon.png in the root directory -->
  <!-- build:css(.) styles/vendor.css -->
  <!-- bower:css -->
  <!-- endbower -->
  <!-- endbuild -->
  <!-- build:css(.tmp) styles/main.css -->
  <link rel="stylesheet" href="styles/main.css">
  <!-- endbuild -->
  <!-- build:js scripts/vendor/modernizr.js -->
  <!-- endbuild -->

</head>
<body>
    <!--[if lt IE 10]>
      <p class="browsehappy">You are using an <strong>outdated</strong> browser. Please <a href="http://browsehappy.com/">upgrade your browser</a> to improve your experience.</p>
      <![endif]-->


      <div class="container">
        <div class="header">
          <ul class="nav nav-pills pull-right">
            <li class="active"><a href="#">Authenticate</a></li>
            <li><a href="#">About</a></li>
          </ul>
          <h3 class="text-muted">Lassonde Auth</h3>
        </div>
        <div class="alert-container">
          <?php
           if (!empty($alerts)) {
            foreach($alerts as $alert) {
              ?>
              <div class="alert alert-danger" role="alert">
                <span class="glyphicon glyphicon-exclamation-sign" aria-hidden="true"></span>
                <span class="sr-only">Error:</span>
                  <?php echo $alert ?>
              </div>
              <?php
            }
           }
          ?>
        </div>
        <div class="jumbotron">
        <?php if (empty($error)) { ?>
        <form role="form" id="login-form" action="" method="POST">
        <div class="form-group">
            <div class="input-group input-group-lg">
              <span class="input-group-addon">@</span>
              <input required type="text" name="username" class="form-control" placeholder="Username">
            </div>
        </div>
        <div class="form-group">
            <div class="input-group input-group-lg">
              <span class="input-group-addon">@</span>
              <input required type="password" name="password" class="form-control" placeholder="Password">
            </div>
        </div>
        <input type="hidden" name="submit" value="true"/>
        <input type="hidden" name="payload" value="<?php echo $_REQUEST["payload"] ?>"/>
        <input type="hidden" name="signature" value="<?php echo $_REQUEST["signature"] ?>"/>
        <p class="lead">You will be authenticated against EECS LDAP.</p>
            <input type="submit" name="submit" value="Login" class="btn btn-lg btn-primary"/>
            <br/><br/>

          </form>
        <?php } else { ?>
          <p class="h2 text-center">Error</p>
          <p class="h4 text-center"><?php echo htmlspecialchars($error) ?></p>
        <?php } ?>
        </div>

        <div class="row marketing">
          <div class="col-lg-4">
            <h4>What is this?</h4>
            <p>This authenticates.</p>
          </div>
          <div class="col-lg-4">
            <h4>What is this?</h4>
            <p>This authenticates.</p>
          </div>
          <div class="col-lg-4">
            <h4>What is this?</h4>
            <p>This authenticates.</p>
          </div>
        </div>

        <div class="footer">
          <p><span class="glyphicon glyphicon-heart"></span> from the Yeoman team</p>
        </div>

      </div>


      <!-- build:js(.) scripts/vendor.js -->
      <!-- bower:js -->
      <!-- endbower -->
      <!-- endbuild -->

    </body>
    </html>
```

```java
package b2b;

import java.io.File;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;

import beans.AggItemBean;
import beans.AggOrderBean;
import beans.ItemBean;
import beans.POBean;

// class that reads and consolidates purchase orders
public class Aggregator
{
        public Aggregator()
    {
    }

        public AggOrderBean getAggregatedOrder(String[] fnames, int id) throws Exception
        {
                // get purchase orders
                List<POBean> orders = this.getOrders(fnames);

                // map that temporally stores item list
                Map<String, AggItemBean> mp = new TreeMap<String, AggItemBean>();

                // aggregate orders
                Iterator<POBean> itr = orders.iterator();
                while (itr.hasNext())
                {
                        List<ItemBean> items = itr.next().getItems().getItemsList();

                        // loop over items in a single order
                        Iterator<ItemBean> itr2 = items.iterator();
                        while (itr2.hasNext())
                        {
                                ItemBean item = itr2.next();
                                String key = item.getNumber();

                                if (mp.containsKey(key))
                                {
                                        mp.get(key).addQuantity(item.getQuantity());
                                }
                                else
                                {
                                        mp.put(key, new AggItemBean(item));
                                }
                        }
                } // end of aggregating orders into map

                // store aggregated order into AggOrderBean
                List<AggItemBean> aggItems = new ArrayList<AggItemBean>();
                Iterator<AggItemBean> itr3 = mp.values().iterator();
                while (itr3.hasNext())
                {
                        aggItems.add(itr3.next());
                }

                return new AggOrderBean(aggItems, id);
        }

        private List<POBean> getOrders(String[] fnames) throws Exception
        {
                if (fnames == null)
                        throw new NullPointerException("invalid file names.");

                // define unmarshaller
                JAXBContext jc;
                Unmarshaller um;
                try
                {
                        jc = JAXBContext.newInstance(POBean.class);
                        um = jc.createUnmarshaller();
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs when defining unmarshaller.");
                }

                // ummarshal order files
                List<POBean> orders = new ArrayList<POBean>();
                for (int i = 0; i < fnames.length; i++)
                {
                        try
                        {
                                orders.add((POBean) um.unmarshal(new File(fnames[i])));
                        }
                        catch (Exception e)
                        {
                                throw new Exception("Error occurs during unmarshalling.");
                        }
                }

                return orders;
        }
}
```

```java
package b2b;

import java.io.File;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import beans.AggItemBean;
import beans.AggOrderBean;

public class Purchaser
{
        private List<Wholesaler> wholesalers;
        private String key;

        public Purchaser()
        {
                this.wholesalers = new ArrayList<Wholesaler>();
        }

        public void purchase(AggOrderBean order, String outfile) throws Exception
        {
                Iterator<AggItemBean> itr = order.getItems().iterator();
                while (itr.hasNext())
                {
                        AggItemBean item = itr.next();

                        // find best offer
                        Wholesaler best = null;
                        double bestOffer = Double.POSITIVE_INFINITY;
                        Iterator<Wholesaler> itr2 = this.wholesalers.iterator();
                        while (itr2.hasNext())
                        {
                                Wholesaler cur = itr2.next();
                                double offer = cur.quote(item.getNumber());
                                if (offer >= 0 && offer < bestOffer)
                                {
                                        best = cur;
                                        bestOffer = offer;
                                }
                        } // end of iterating wholesalers

                        // item out of stock
                        if (best == null)
                        {
                                item.setPrice(-1.0);
                                item.setWholesaler("No wholesaler");
                                item.setCode("Out of stock");
                                item.setOrdered(false);
                        }
                        else // take the best offer
                        {
                                String code;
                                try
                                {
                                        code = best.order(item.getNumber(), item.g
etQuantity(), this.key);
                                        if (code.equals("Invalid key!"))
                                        {
                                                item.setOrdered(false);
                                        }
                                        else
                                        {
                                                item.setOrdered(true);
                                        }
                                        item.setPrice(bestOffer);
                                        item.setCode(code);
                                        item.setWholesaler(best.getName());
                                }
                                catch (Exception e)
                                {
                                        item.setOrdered(false);
                                }
                        }
                } // end of iterating items

                // generate report
                this.generateReport(order, outfile);
        }

        public boolean addWholesaler(Wholesaler ws)
        {
                return this.wholesalers.add(ws);
        }

        public void setKey(String key)
        {
                this.key = key;
        }

        private void generateReport(AggOrderBean order, String filename) throws Exception
        {
                // marshaling
                try
                {
                        JAXBContext jc = JAXBContext.newInstance(order.getClass());
                        Marshaller mar = jc.createMarshaller();
                        mar.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
                        mar.setProperty(Marshaller.JAXB_FRAGMENT, Boolean.TRUE);
                        mar.marshal(order, new File(filename));
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs during marshalling.");
                }
        }
}
```

```java
package b2b;

import java.net.URL;

import javax.xml.soap.*;

import org.w3c.dom.NodeList;

public class Wholesaler
{
        private String name;
        private URL url;

        public Wholesaler(String name, URL url)
    {
                this.name = name;
                this.url = url;
    }

        public String getName()
        {
                return name;
        }

        public void setName(String name)
        {
                this.name = name;
        }

        public URL getUrl()
        {
                return url;
        }

        public void setUrl(URL url)
        {
                this.url = url;
        }

        public double quote(String itemNum) throws Exception
        {
                String[] params = {"itemNumber"};
                String[] values = {itemNum};

                return Double.parseDouble(this.operation("quote", params, values).item(0).getTextContent());
        }

        public String order(String itemNum, int quantity, String key) throws Exception
        {
                String[] params = {"itemNumber", "quantity", "key"};
                String[] values = {itemNum, String.valueOf(quantity), key};

                return this.operation("order", params, values).item(0).getTextContent();
        }

        private NodeList operation(String operation, String[] reqParams, String[] values) throws Exception
        {
                // create SOAP message
                SOAPMessage msg;
                try
                {
                        msg = MessageFactory.newInstance().createMessage();

                        MimeHeaders header = msg.getMimeHeaders();
                        header.addHeader("SOAPAction", "");

                        SOAPBody body = msg.getSOAPPart().getEnvelope().getBody();
                        SOAPElement elm = body.addChildElement(operation);
                        for (int i=0; i < reqParams.length; i++)
                        {
                                elm.addChildElement(reqParams[i]).addTextNode(values[i]);
                        }
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs when creating SOAP message.");
                }

                // connect to the wholesaler web service
                SOAPMessage resp;
                try
                {
                        SOAPConnection sc = SOAPConnectionFactory.newInstance().createConnection();
                        resp = sc.call(msg, this.url);
                        sc.close();
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs when setting up SOAP connection.");
                }

                // parse the response
                NodeList result;
                try
                {
                        result = resp.getSOAPPart().getEnvelope().getBody().getElementsByTagName(operation + "Return");
                        // resp.writeTo(System.out);
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs when parsing SOAP return.");
                }

                return result;
        }
}
```

```java
package beans;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

// item bean for aggregated order
@XmlRootElement
public class AggItemBean
{
        // @XmlAttribute
        private String number;
        // @XmlElement
        private String name;
        // @XmlElement
        private int quantity;

        // the following attributes may not be defined at creation
        // @XmlElement
        private double price;
        // @XmlElement
        private double extended;
        // @XmlElement
        private String wholesaler;
        // @XmlElement
        private String code;
        // @XmlElement
        private boolean ordered;

        public AggItemBean()
   {
           super();
   }

        public AggItemBean(ItemBean item)
        {
                this.number = item.getNumber();
                this.name = item.getName();
                this.quantity = item.getQuantity();
                this.ordered = false;
        }

        public AggItemBean(String number, String name, double price, int quantity,
         String wholesaler, String code)
   {
           this.number = number;
           this.name = name;
           this.price = price;
           this.quantity = quantity;
           this.extended = 0;
           this.wholesaler = wholesaler;
           this.code = code;
           this.ordered = false;
   }

        public void addQuantity(int amount)
        {
                this.quantity = this.quantity + amount;
        }

        // ==============   getters and setters  ====================
        @XmlAttribute
        public String getNumber()
        {
                return number;
        }

        public void setNumber(String number)
        {
                this.number = number;
        }

        @XmlElement
        public String getName()
        {
                return name;
        }

        public void setName(String name)
        {
                this.name = name;
        }

        @XmlElement
        public double getPrice()
        {
                return price;
        }

        public void setPrice(double price)
        {
                this.price = price;
        }

        @XmlElement
        public int getQuantity()
        {
                return quantity;
        }

        public void setQuantity(int quantity)
        {
                this.quantity = quantity;
        }

        @XmlElement
        public double getExtended()
        {
                this.updateExtended();
                return extended;
        }

        @XmlElement
        public String getWholesaler()
        {
                return wholesaler;
        }

        public void setWholesaler(String wholesaler)
        {
                this.wholesaler = wholesaler;
        }

        @XmlElement
        public String getCode()
        {
                return code;
        }

        public void setCode(String code)
        {
                this.code = code;
        }

        @XmlElement
        public boolean isOrdered()
        {
                return ordered;
        }

        public void setOrdered(boolean ordered)
        {
                this.ordered = ordered;
        }

        private void updateExtended()
        {
                if (this.ordered)
                {
                        this.extended = this.price * this.quantity;
                }
                else
                {
                        this.extended = 0;
                }
        }
}
```

```java
package beans;

import java.util.Iterator;
import java.util.List;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

// aggregated order bean
@XmlRootElement(name="procurement")
public class AggOrderBean
{
        // @XmlElement(name="item")
        private List<AggItemBean> items;
        // @XmlElement
        private double total;
        // @XmlAttribute
        private int id;

        public AggOrderBean()
    {
           super();
    }

        public AggOrderBean(List<AggItemBean> items, int id)
    {
           this.items = items;
           this.updateTotal();
           this.id = id;
    }

        // ==============   getters and setters  ====================
        @XmlElement(name="item")
        public List<AggItemBean> getItems()
        {
                return items;
        }

        public void setItems(List<AggItemBean> items)
        {
                this.items = items;
        }

        @XmlElement
        public double getTotal()
        {
                this.updateTotal();
                return total;
        }

        @XmlAttribute
        public int getId()
        {
                return id;
        }

        public void setId(int id)
        {
                this.id = id;
        }

        private void updateTotal()
        {
                this.total = 0;
                Iterator<AggItemBean> itr = this.items.iterator();
                while (itr.hasNext())
                {
                        this.total = this.total + itr.next().getExtended();
                }
        }
}
```

```java
package beans;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class CustomerBean
{
        // @XmlElement
        private String name;
        // @XmlAttribute
        private String account;

        public CustomerBean()
   {
           super();
   }

        public CustomerBean(String name, String account)
   {
           this.name = name;
           this.account = account;
   }

        // ==============   getters and setters  ====================
        @XmlElement
        public String getName()
        {
                return name;
        }

        public void setName(String name)
        {
                this.name = name;
        }

        @XmlAttribute
        public String getAccount()
        {
                return account;
        }

        public void setAccount(String account)
        {
                this.account = account;
        }
}
```

```java
package beans;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class ItemBean
{
        // @XmlElement
        private String name;
        // @XmlElement
        private double price;
        // @XmlElement
        private int quantity;
        // @XmlElement
        private double extended;        // = price * quantity
        // @XmlAttribute
        private String number;  // item number

        public ItemBean()
   {
           super();
   }

        public ItemBean(String name, double price, int quantity, double extended,
         String number)
   {
           this.name = name;
           this.price = price;
           this.quantity = quantity;
           this.extended = extended;
           this.number = number;
   }

        // ==============   getters and setters  ====================
        @XmlElement
        public String getName()
        {
                return name;
        }

        public void setName(String name)
        {
                this.name = name;
        }

        @XmlElement
        public double getPrice()
        {
                return price;
        }

        public void setPrice(double price)
        {
                this.price = price;
        }

        @XmlElement
        public int getQuantity()
        {
                return quantity;
        }

        public void setQuantity(int quantity)
        {
                this.quantity = quantity;
        }

        @XmlElement
        public double getExtended()
        {
                return extended;
        }

        public void setExtended(double extended)
        {
                this.extended = extended;
        }

        @XmlAttribute
        public String getNumber()
        {
                return number;
        }

        public void setNumber(String number)
        {
                this.number = number;
        }
}
```

```java
package beans;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class ItemsWrapper
{
        @XmlElement(name="item")
        private List<ItemBean> list;

        public ItemsWrapper()
    {
           super();
    }

        public List<ItemBean> getItemsList()
        {
                return list;
        }

        public void setItemsList(List<ItemBean> list)
        {
                this.list = list;
        }
}
```

```
package beans;

import java.util.Calendar;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlSchemaType;

// Bean class for single purchase order
@XmlRootElement(name="order")
public class POBean
{
        // @XmlElement
        private CustomerBean customer;
        // @XmlElement(name="items")
        private ItemsWrapper items;
        // @XmlElement
        private double total;
        // @XmlElement
        private double shipping;
        // @XmlElement(name="HST")
        private double hst;
        // @XmlElement
        private double grandTotal;
        // @XmlAttribute
        private int id;
        // @XmlAttribute
        // @XmlSchemaType(name = "date")
        private Calendar submitted;

        public POBean()
    {
           super();
    }

        public POBean(CustomerBean customer, ItemsWrapper items, double total,
         double shipping, double hst, double grandTotal, int id,
         Calendar submitted)
    {
           this.customer = customer;
           this.items = items;
           this.total = total;
           this.shipping = shipping;
           this.hst = hst;
           this.grandTotal = grandTotal;
           this.id = id;
           this.submitted = submitted;
    }

        // ==============   getters and setters  ====================
        @XmlElement
        public CustomerBean getCustomer()
        {
                return customer;
        }

        public void setCustomer(CustomerBean customer)
        {
                this.customer = customer;
        }

        @XmlElement(name="items")
        public ItemsWrapper getItems()
        {
                return items;
        }

        public void setItems(ItemsWrapper items)
        {
```

```java
                this.items = items;
        }

        @XmlElement
        public double getTotal()
        {
                return total;
        }

        public void setTotal(double total)
        {
                this.total = total;
        }

        @XmlElement
        public double getShipping()
        {
                return shipping;
        }

        public void setShipping(double shipping)
        {
                this.shipping = shipping;
        }

        @XmlElement
        public double getHst()
        {
                return hst;
        }

        public void setHst(double hst)
        {
                this.hst = hst;
        }

        @XmlElement
        public double getGrandTotal()
        {
                return grandTotal;
        }

        public void setGrandTotal(double grandTotal)
        {
                this.grandTotal = grandTotal;
        }

        @XmlAttribute
        public int getId()
        {
                return id;
        }

        public void setId(int id)
        {
                this.id = id;
        }

        @XmlAttribute
        @XmlSchemaType(name = "date")
        public Calendar getSubmitted()
        {
                return submitted;
        }

        public void setSubmitted(Calendar submitted)
        {
                this.submitted = submitted;
        }
}
```

```java
package beans;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class POFileBean
{
        @XmlElement
        private String url;
        @XmlElement
        private int id;

        public POFileBean()
    {
           super();
    }

        public String getUrl()
        {
                return url;
        }

        public int getId()
        {
                return id;
        }
}
```

```java
package beans;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="PurchaseOrderFiles")
public class POFilesWrapper
{
        @XmlElement(name="PurchaseOrderFile")
        private List<POFileBean> lsPof;

        public POFilesWrapper()
   {
           super();
   }

        public List<POFileBean> getPOFileBeans()
        {
                return lsPof;
        }
}
```

```java
package ctrl;

import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;

import b2b.Aggregator;
import b2b.Purchaser;
import b2b.Wholesaler;
import beans.AggOrderBean;

public class Manager
{
        public static void main(String[] args)
        {
                // create aggregator and purchaser
                Aggregator agg = new Aggregator();
                Purchaser pch = new Purchaser();

                // add wholesalers
                try
                {
                        pch.addWholesaler(new Wholesaler("Toronto", new URL("http://roumani.eecs.yorku.ca:4413/axis/YYZ.jws")));
                        pch.addWholesaler(new Wholesaler("Vancouver", new URL("http://roumani.eecs.yorku.ca:4413/axis/YVR.jws")));
                        pch.addWholesaler(new Wholesaler("Halifax", new URL("http://roumani.eecs.yorku.ca:4413/axis/YHZ.jws")));
                }
                catch (MalformedURLException m)
                {
                        System.out.println("Invalid URL for wholesalers. Program aborted.");
                        System.exit(1);
                }

                // set purchase key
                pch.setKey("4413secret");

                // download PO files from eFood server
                PODownloader dl = new PODownloader("http://localhost:4413/eFoods/backend/orders", "http://localhost:4413");
                Date now = new Date();
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
                String dir = "./" + sdf.format(now) + "/";
                new File(dir).mkdir();

                try
                {
                        String[] fns = dl.downloadPOs(dir);

                        int id = Math.abs(now.hashCode());
                        dl.setStatus("pending");
                        AggOrderBean prcrm = agg.getAggregatedOrder(fns, id);
                        pch.purchase(prcrm, dir + "rpt" + id + ".xml");
                        dl.setStatus("purchased");
                        //dl.setStatus("new"); //restore for testing
                }
                catch (Exception e)
                {
                        System.out.println(e.getMessage() + " Program aborted.");
                        System.exit(1);
                }
        }

}
```

```java
package ctrl;

import java.io.File;
import java.io.FileOutputStream;
import java.net.ConnectException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.channels.Channels;
import java.nio.channels.ReadableByteChannel;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;

import beans.POFileBean;
import beans.POFilesWrapper;

public class PODownloader
{
        String pofwurl;
        String bseurl;
        List<POFileBean> pofs;

        public PODownloader(String poFilesWrapperUrl, String baseUrl)
    {
                this.pofwurl = poFilesWrapperUrl;
                this.bseurl = baseUrl;
    }

        public String[] downloadPOs(String dir) throws Exception
        {
                // download XML file that specifies urls of PO files
                String pofsname = dir + "POFiles.xml";
                this.download(pofwurl, pofsname);

                // get list of PO file urls
                List<String> urls = getPOUrls(pofsname);

                // download the PO files
                String[] fnames = new String[urls.size()];
                int i = 0;
                Iterator<String> itr = urls.iterator();
                while (itr.hasNext())
                {
                        String cur = itr.next();
                        fnames[i] = dir + this.getFileNameOnly(cur);
                        this.download(cur, fnames[i]);
                        i++;
                }

                return fnames;
        }

        private void download(String srcurl, String fname) throws Exception
        {
                try
                {
                        URL url = new URL(srcurl);
                        ReadableByteChannel src = Channels.newChannel(url.openStre
am());
                        FileOutputStream dst = new FileOutputStream(fname);
                        dst.getChannel().transferFrom(src, 0, Long.MAX_VALUE);
                        dst.close();
                }
                catch (MalformedURLException m)
                {
                        throw new Exception("Invalid URL!");
                }
                catch (ConnectException c)
                {
                        throw new Exception("Cannot connect to source URL.");
                }
                catch (Exception e)
                {
                        //e.printStackTrace();
                        throw new Exception("Error: fail to download file.");
                }
        }

        private List<String> getPOUrls(String fname) throws Exception
        {
                if (fname == null)
                        throw new NullPointerException("invalid file names.");

                // define unmarshaller
                JAXBContext jc;
                Unmarshaller um;
                try
                {
                        jc = JAXBContext.newInstance(POFilesWrapper.class);
                        um = jc.createUnmarshaller();
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs when defining unmarshall
er.");
                }

                // ummarshal order files
                POFilesWrapper pofw;
                try
                {
                        pofw = (POFilesWrapper) um.unmarshal(new File(fname));
                }
                catch (Exception e)
                {
                        throw new Exception("Error occurs during unmarshalling.");
                }

                if (pofw == null)
                        throw new Exception("POFilesWrapper does not generated!");


                List<String> urls = new ArrayList<String>();
                this.pofs = pofw.getPOFileBeans();
                if (pofs != null) {
                        Iterator<POFileBean> itr = pofs.iterator();
                        while (itr.hasNext())
                        {
                                urls.add(this.bseurl + itr.next().getUrl());
                        }
                }


                return urls;
        }

        private String getFileNameOnly(String url)
        {
                return url.substring(url.lastIndexOf('/') + 1);
        }

        public void setStatus(String status) throws Exception {
                if (pofs != null)
                {
                        String type = "application/x-www-form-urlencoded";
                        for(POFileBean pof: pofs) {
```

```
                    int id = pof.getId();
                    try {
                        URL u = new URL(this.bseurl + "/eFoods/backend/orders/" + id + "?ajax=1&status=" + status);
                        HttpURLConnection conn = (HttpURLConnection) u.openConnection();
                        conn.setDoOutput(true);
                        conn.setRequestMethod("POST");
                        conn.setRequestProperty( "Content-Type", type );
                        conn.setRequestProperty( "Content-Length", "0");
                        System.out.println("State changed\t:" + id + ":" +
                                status + ":" +
                                conn.getResponseMessage());
                        conn.disconnect();
                    } catch (Exception e) {
                        //e.printStackTrace();
                        //System.out.println(e.getMessage());
                        throw new Exception("set status failed! " + e.getMessage());
                    }
                } // end of traversal of POFileBean list
            }
        } // end of method setStatus

}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<faceted-project>
  <runtime name="Apache Tomcat v8.0"/>
  <fixed facet="jst.web"/>
  <fixed facet="java"/>
  <fixed facet="wst.jsdt.web"/>
  <installed facet="java" version="1.7"/>
  <installed facet="jst.web" version="3.1"/>
  <installed facet="wst.jsdt.web" version="1.0"/>
</faceted-project>
```

```java
package ctrl;

import java.io.IOException;

import javax.servlet.Servlet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import util.Route;

public class Analytics extends BaseCtrl implements Servlet
{
        private static final long serialVersionUID = 1L;
        /*
         * Route definitions for this controller
         */
        public static final int ANALYTICS_PAGE = 0x0a;

        @Override
        protected void processRequest(HttpServletRequest request,
              HttpServletResponse response) throws ServletException, IOException
        {

                Route route = getRoute(request);
                switch(route.getIdentifier()) {
                case ANALYTICS_PAGE:
                        // get analytic information from application scope
                        Long avgAddItemTime = (Long) this.getServletContext().getAttribute("avgAddItemTime");
                        Long avgCheckOutTime = (Long) this.getServletContext().getAttribute("avgCheckOutTime");
                        if (avgAddItemTime == null) avgAddItemTime = new Long(0);
                        if (avgCheckOutTime == null) avgCheckOutTime = new Long(0);
                        // poke information into request scope
                        request.setAttribute("avgAddItemTime", avgAddItemTime);
                        request.setAttribute("avgCheckOutTime", avgCheckOutTime);

                        // dispatch to JSPX
                        request.getRequestDispatcher("/partials/_analytics.jspx").forward(request, response);
                        break;
                default:
                        throw new ServletException("Route configuration error.");
                }
        }

}
```

```java
package ctrl;

import javax.servlet.Servlet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import model.UserBean;
import util.Route;
import util.SSOAuthenticator;

/**
 * Servlet implementation class Login
 */
public class Auth extends BaseCtrl implements Servlet {
        private static final long serialVersionUID = 1L;

        /*
         * Route definitions for this controller
         */
        public static final int ROUTE_INITAL = 0x0a;
        public static final int ROUTE_AUTHENTICATE = 0x0b;
        public static final int ROUTE_LOGOUT = 0x0c;
        public static final int ROUTE_USER_BADGE = 0x0d;


        protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws Exception {
                SSOAuthenticator auth = (SSOAuthenticator) getAuthenticator(request);
                Route route = getRoute(request);
                switch(route.getIdentifier()) {
                case ROUTE_INITAL:
                        request.getSession().setAttribute("LAST_STATE", request.getParameter("state"));
                        auth.SSORedirect(request, response);
                        break;
                case ROUTE_AUTHENTICATE:
                        if (auth.login(request, null, null)) {
                                UserBean usr = auth.getUser(request);
                                System.out.println("Login: Successfully authenticated as " + usr.getName());
                                String state = ((String) request.getSession().getAttribute("LAST_STATE"));
                                if (state == null || state.startsWith("backend/error")) { state = ""; }
                                String rdr = (request.getContextPath().length() == 0 ? "/" : request.getContextPath()) + "#!" + state;
                                System.out.println("Redirecting to: " + rdr);
                                response.sendRedirect(rdr);
                        } else {
                                throw new Exception("Sorry! We could not authenticate you at this moment.");
                        }
                        break;
                case ROUTE_LOGOUT:
                        auth.logout(request);
                        System.out.println("Login: Logged out");
                        break;
                case ROUTE_USER_BADGE:
                        request.setAttribute("currentUser", auth.getUser(request));
                        request.getRequestDispatcher("/partials/_userBadge.jspx").forward(
                                        request, response);
                        break;
                default:
                        throw new ServletException("Route configuration error.");
                }

        }

}
```

```java
package ctrl;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import util.Authenticator;
import util.Route;
import util.Router;
import model.*;


public abstract class BaseCtrl extends HttpServlet
{
        private static final long serialVersionUID = 1L;

        public class PagingHelper {
                int page;
                int limit;
                /**
                 * @return the page
                 */
                public int getPage() {
                        return page;
                }

                /**
                 * @return the limit
                 */
                public int getLimit() {
                        return limit;
                }

                public PagingHelper(int page, int limit) {
                        super();
                        this.page = page;
                        this.limit = limit;
                }
        }

    public BaseCtrl() {
        super();

        // TODO Auto-generated constructor stub
    }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                try {
                        processRequest(request, response);
                } catch (Exception e) {
                        e.printStackTrace();
                        throw new ServletException(e.getMessage());
                }

        }


        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                doGet(request, response);
        }

        protected abstract void processRequest(HttpServletRequest request, HttpServletResponse response) throws Exception;


        protected PagingHelper getPagination(HttpServletRequest request) {
                String pageStr, limitStr;
                int page, limit;
                if ((pageStr = request.getParameter("page")) != null )
                        page = Integer.parseInt(pageStr);
                else
                        page = BaseDAO.PAGE_ALL;

                if ((limitStr = request.getParameter("limit")) != null)
                        limit = Integer.parseInt(limitStr);
                else
                        limit = BaseDAO.LIMIT_ALL;
                return new PagingHelper(page, limit);
        }

        //Static helpers

        protected static EFoods getModel(HttpServletRequest request) {
                return (EFoods) request.getServletContext().getAttribute(ctrl.Front.MODEL_KEY);
        }

        protected static Route getRoute(HttpServletRequest request) {
                return (Route) request.getAttribute(Router.REQUEST_ROUTE_KEY);
        }

        protected static Authenticator getAuthenticator(HttpServletRequest request) {
                return (Authenticator) request.getServletContext().getAttribute(ctrl.Front.SSO_AUTHENTICATOR_KEY);
        }

}
```

```java
package ctrl;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.Servlet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import util.PurchaseOrderFiles;
import util.Route;
import util.SSOAuthenticator;
import util.PurchaseOrderFiles.PurchaseOrderFile;
import model.CartModel;
import model.EFoods;
import model.ItemBean;
import model.ItemDAO;

/**
 * Servlet implementation class Cart
 */

public class Cart extends BaseCtrl implements Servlet {
        private static final long serialVersionUID = 1L;
        /*
         * Route definitions for this controller
         */
        public static final int ROUTE_ALL = 0x0a;
        public static final int ROUTE_MANIPULATE = 0x0b;
        public static final int ROUTE_HISTORY = 0x0c;
        public static final int ROUTE_CHECKOUT = 0x0d;
        public static final int ROUTE_BADGE = 0x0e;
        public static final int ROUTE_SUCCESS = 0x0f;

    /**
     * @see BaseCtrl#BaseCtrl()
     */
    public Cart() {
        super();
        // TODO Auto-generated constructor stub
    }


        @Override
        protected void processRequest(HttpServletRequest request,
                        HttpServletResponse response) throws Exception {
                Route route = getRoute(request);
                SSOAuthenticator auth = (SSOAuthenticator) getAuthenticator(request);
                CartModel cart = CartModel.getInstance(request ,auth);
                EFoods model = getModel(request);
                PagingHelper paging = getPagination(request);
                String filePath = request.getServletContext().getRealPath(Orders.BASE_PATH);
                switch(route.getIdentifier()) {
                case ROUTE_ALL:
                        request.setAttribute("cartItems", cart.getCartItems());
                        request.setAttribute("total", cart.getTotal());
                        request.setAttribute("shipping", cart.getShipping());
                        request.setAttribute("tax", cart.getTax());
                        request.setAttribute("user", auth.getUser(request));
                        request.getRequestDispatcher("/partials/_cart.jspx").forward(
                                        request, response);
                        break;
                case ROUTE_MANIPULATE:
                        String itemNumber;
                        ItemBean item;
                        int newQty;
                        itemNumber = request.getParameter("item");
                        try {
                                newQty =  Integer.parseInt(request.getParameter("qty"));
                        } catch (NumberFormatException e) {
                                throw new Exception("Quantity must be a valid number.");
                        }

                        if (newQty < 0) throw new Exception("Quantity must be 0 or positive.");

                        item = model.items(itemNumber, ItemDAO.CAT_ALL, paging.getPage(), paging.getLimit(), ItemDAO.NO_FILTER).get(0);
                        cart.manipulateCart(item, newQty);

                        // poke number of cart items into session scope, for listener
                        request.getSession().setAttribute("cartItemsCount", cart.getCartItems().size());

                        break;
                case ROUTE_CHECKOUT:
                        System.out.println("Checking out");
                        if (cart.getCartItems().size() < 1) {
                                throw new Exception("Can't checkout empty cart!");
                        }
                        int orderId = model.createPurchaseOrder(cart, filePath);
                        // notify listener by poking attribute into session scope after client had checked out
                        cart.clear();
                        request.getSession().setAttribute("checkedOut", true);
                        String rdr = (request.getContextPath().length() == 0 ? "/" : request.getContextPath()) + "#!backend/cart/success?order=" + orderId;
                        response.sendRedirect(rdr);
                        break;
                case ROUTE_HISTORY:
                        PurchaseOrderFiles pofs = new PurchaseOrderFiles(filePath);
                        List<PurchaseOrderFile> pofList = pofs.getOrdersByUser(auth.getUser(request).getName());

                        List<PurchaseOrderFile> pendingOrders = new ArrayList<PurchaseOrderFile>();
                        List<PurchaseOrderFile> newOrders = new ArrayList<PurchaseOrderFile>();
                        List<PurchaseOrderFile> purchasedOrders = new ArrayList<PurchaseOrderFile>();

                        //Sorting the purchase orders by their type
                        for(PurchaseOrderFile pof : pofList)
                                if (pof.getStatus().equals(PurchaseOrderFile.STATUS_NEW))
                                        newOrders.add(pof);
                                else if (pof.getStatus().equals(PurchaseOrderFile.STATUS_PENDING))
                                        pendingOrders.add(pof);
                                else if (pof.getStatus().equals(PurchaseOrderFile.STATUS_PURCHASED))
                                        purchasedOrders.add(pof);

                        request.setAttribute("pendingOrders", pendingOrders);
                        request.setAttribute("newOrders", newOrders);
                        request.setAttribute("purchasedOrders", purchasedOrders);
                        request.getRequestDispatcher("/partials/_cartHistory.jspx").forward(
                                        request, response);
                        break;
                case ROUTE_SUCCESS:
                        request.getRequestDispatcher("/partials/_cartSuccess.jspx"
).forward(
                                request, response);
                        break;
                case ROUTE_BADGE:
                        request.setAttribute("cartItemCount", cart.getCartItems().
size());
                        request.getRequestDispatcher("/partials/_cartBadge.jspx").
forward(
                                request, response);
                        break;
                default:
                        throw new ServletException("Route configuration error.");
                }

        }







}
```

```java
package ctrl;

import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import util.Route;
import model.*;

@WebServlet(name = "Category", displayName = "Category")
public class Category extends BaseCtrl {
        private static final long serialVersionUID = 1L;

        /*
         * Route definitions for this controller
         */
        public static final int ROUTE_ALL = 0xff;
        public static final int ROUTE_BY_ID = 0x0a;

        protected void processRequest(HttpServletRequest request,
                        HttpServletResponse response) throws Exception {
                EFoods model = getModel(request);
                Route route = getRoute(request);
                List<CategoryBean> results = null;
                int routeType;
                if (route == null)
                        routeType = -1;
                else
                        routeType = route.getIdentifier();

                switch (routeType) {
                case ROUTE_BY_ID:
                        int id = Integer.parseInt(route.getMatcher().group("catId"));
                        results = model.categories(id);
                        break;
                case ROUTE_ALL:
                        results = model.categories(CategoryDAO.ID_ALL);
                        break;
                default:
                        throw new ServletException("Route configuration error.");
                }

                request.setAttribute("results", results);
                request.getRequestDispatcher("/partials/_category.jspx").forward(
                                request, response);

        }

}
```

```java
package ctrl;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletConfig;
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 org.apache.tomcat.util.codec.binary.Base64;

import model.*;
import util.*;

/**
 * Servlet implementation class Front
 */
@WebServlet("/backend/*")
@SuppressWarnings("serial")
public class Front extends HttpServlet {
        private static final long serialVersionUID = 1L;

        //Context objects
        public static final String MODEL_KEY = "com.eFoods.ctrl.Front.EFOODS";
        private static final String ROUTER_KEY = "com.eFoods.ctrl.Front.ROUTER";
        public static final String SSO_AUTHENTICATOR_KEY = "com.eFoods.ctrl.Front.HTTP_AUTHENTICATOR";


        //SSO Configuration
        //private static final String SSO_AUTHENTICATOR_URL = "http://localhost:9000/";
        private static final String SSO_AUTHENTICATOR_URL = "https://www.cse.yorku.ca/~cse11011/4413/auth/";
        private static final String SSO_RECIEVER_URL = "backend/login/authenticate";
        private static final String SSO_SHARED_KEY = "7fcbc0e27e7bb31e4ad7a39677eebdaf6b94c9db7dcdb2ebf25791298609a4a4";
        private static final List<String> SSO_ADMINS = new ArrayList<String>() {{
                add("cse11011"); //List of admin users
                add("cse13195");
                add("cse13122");
        }};


    /**
     * @see HttpServlet#HttpServlet()
     */
    public Front() {
        super();
        // TODO Auto-generated constructor stub
    }

        /**
         * @see Servlet#init(ServletConfig)
         */
        public void init(ServletConfig config) throws ServletException {
                super.init();

                //Initializing Model
                EFoods model = new EFoods();

                //Initializing authentication module
                SSOAuthenticator auth;
                try {
                        auth = new SSOAuthenticator(SSO_AUTHENTICATOR_URL, SSO_REC
IEVER_URL, SSO_SHARED_KEY, SSO_ADMINS);
                } catch (Exception e) {
                        throw new ServletException(e.getMessage());
                }


                Router appRouter = new Router();
                /*
                 * Start Route Definitions
                 */
                //List All categories
                appRouter.addRoute(new Route("^/category(/)?$","Category", Route.METHOD_GET, Category.ROUTE_ALL, false));
                //List All items from a category
                appRouter.addRoute(new Route("^/category/(?<catId>[0-9]+)(/)?$","Item", Route.METHOD_GET, Item.ROUTE_BY_CATEGORY, false));
                //List all items
                appRouter.addRoute(new Route("^/item(/)?$","Item", Route.METHOD_GET, Item.ROUTE_ALL,false));
                //Render a partial of a single item
                appRouter.addRoute(new Route("^/item/(?<itemNumber>[0-9a-zA-Z]+)/partial(/)?$","Item", Route.METHOD_GET, Item.ROUTE_BY_NUMBER,false));
                //Render an item as a result listing
                appRouter.addRoute(new Route("^/item/(?<itemNumber>[0-9a-zA-Z]+)(/)?$","Item", Route.METHOD_GET, Item.ROUTE_BY_NUMBER_FULL,false));
                //Free text search on all items
                appRouter.addRoute(new Route("^/search(/)?$","Item", Route.METHOD_GET, Item.ROUTE_BY_SEARCH,false));
                //Authentication initiation endpoint
                appRouter.addRoute(new Route("^/login(/)?$","Auth", Route.METHOD_GET, Auth.ROUTE_INITAL, false));
                //Authentication redirect reciever endpoint
                appRouter.addRoute(new Route("^/login/authenticate(/)?$","Auth", Route.METHOD_GET, Auth.ROUTE_AUTHENTICATE, false));
                //Log out current user
                appRouter.addRoute(new Route("^/logout(/)?$","Auth", Route.METHOD_GET, Auth.ROUTE_LOGOUT, true));
                //Render the badge that shows user information or logged in link if not auth
                appRouter.addRoute(new Route("^/user(/)?$","Auth", Route.METHOD_GET, Auth.ROUTE_USER_BADGE, false));
                //Render current cart
                appRouter.addRoute(new Route("^/cart(/)?$","Cart", Route.METHOD_GET, Cart.ROUTE_ALL, false));
                //Change elements in the cart. Accepted GET "item=?&qty=?
                appRouter.addRoute(new Route("^/cart(/)?$","Cart", Route.METHOD_POST, Cart.ROUTE_MANIPULATE, false));
                //Initiate checkout of the current cart
                appRouter.addRoute(new Route("^/cart/checkout(/)?$","Cart", Route.METHOD_POST, Cart.ROUTE_CHECKOUT, true));
                //Render purchase history page
                appRouter.addRoute(new Route("^/cart/history(/)?$","Cart", Route.METHOD_GET, Cart.ROUTE_HISTORY, true));
                //Render checkout success page
                appRouter.addRoute(new Route("^/cart/success(/)?$","Cart", Route.METHOD_GET, Cart.ROUTE_SUCCESS, true));
                //Render cart item count badge
                appRouter.addRoute(new Route("^/cart/badge(/)?$","Cart", Route.METHOD_GET, Cart.ROUTE_BADGE, false));
                //Render generic error page (accepts b64 string as an error message to show)
                appRouter.addRoute(new Route("^/error/(?<Base64EncodedMessage>.*)?","Misc", Route.METHOD_GET, Misc.ERROR_PAGE, false));
                //Render analytics page
                appRouter.addRoute(new Route("^/analytics(/)?$","Analytics", Route.METHOD_ANY, Analytics.ANALYTICS_PAGE, true, true));
                //Render XML listing of available orders
                appRouter.addRoute(new Route("^/orders(/)?$","Orders", Route.METHOD_GET, Orders.ROUTE_ALL_NEW, false));
                //Change status of an order (Accepts GET parameter "state" being o
```

```java
ne of "new", "pending" or "purchased")
		appRouter.addRoute(new Route("^/orders/(?<orderId>[0-9]+)(/)?$","Orders", Route.METHOD_POST, Orders.ROUTE_UPDATE_STATUS, false));
		//Permenantly remove all orders from the system. Destructive action!
		appRouter.addRoute(new Route("^/orders/nuke(/)?$","Orders", Route.METHOD_GET, Orders.ROUTE_NUKE, false));
		/*
		 * End Route definitions
		 */

		//Poking the context
		config.getServletContext().setAttribute(MODEL_KEY, model);
		config.getServletContext().setAttribute(ROUTER_KEY, appRouter);
		config.getServletContext().setAttribute(SSO_AUTHENTICATOR_KEY, auth);

	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

		// ensure to create a session whenever a fresh request comes
		request.getSession();

		Router router =  (Router) request.getServletContext().getAttribute(ROUTER_KEY);
		Authenticator httpAuth = (SSOAuthenticator) request.getServletContext().getAttribute(SSO_AUTHENTICATOR_KEY);

		Route route;
		try {
			if ((route = router.getRoute(request)) != null) {
				if (!route.isRequireAuthentication() || httpAuth.isAuthenticated(request)) {
					System.out.println("Serving Route: " + route.getMethod() + " " + request.getPathInfo() + " (" + route.getUrlPattern() + ") " +
							(route.isRequireAuthentication() ? " (Authenticated as " + ((SSOAuthenticator) httpAuth).getUser(request).getName() + ")": ""));
					request.getServletContext().getNamedDispatcher(route.getDestination()).forward(request, response);
				} else {
					throw new Exception("You are not authorized to access this page. Please login.");
				}
			} else {
				System.out.println("Route not found!: " + request.getPathInfo());
				throw new Exception("Requested path is not found.");
			}
		} catch (Exception e) {
			String exceptionMsg = e.getMessage();
			if (exceptionMsg == null) {
				exceptionMsg = "Unknown error.";
			}
			System.out.println("Front controller exception! : " + exceptionMsg);
			String encodedError = Base64.encodeBase64String(exceptionMsg.getBytes());
			if(request.getParameter("main") != null) {
				System.out.println("Routing via Misc.");
				request.setAttribute("encodedError", encodedError);
				request.getServletContext().getNamedDispatcher("Mi
sc").forward(request, response);
                        } else if (request.getParameter("ajax") != null) {
                                 response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
                                 response.setContentType("application/json");
                                 response.getOutputStream().print("{\"error\": \"" + exceptionMsg.replaceAll("\"", "\\\"") + "\"}");
                        } else {
                                String rdr = (request.getContextPath().length() == 0 ? "/" : request.getContextPath()) + "#!backend/error/" + encodedError;;
                                System.out.println("Redirecting to: " + rdr);
                                response.sendRedirect(rdr);
                        }
                }
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
                doGet(request, response);
        }

}
```

```java
package ctrl;

import java.util.List;



import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import util.Route;
import util.SSOAuthenticator;
import model.*;


@WebServlet(name = "Item", displayName = "Item")
public class Item extends BaseCtrl {
        private static final long serialVersionUID = 1L;

        /*
         * Route definitions for this controller
         */
        public static final int ROUTE_ALL = 0xff;
        public static final int ROUTE_BY_NUMBER = 0x0b;
        public static final int ROUTE_BY_CATEGORY = 0x0c;
        public static final int ROUTE_BY_SEARCH = 0x0d;
        public static final int ROUTE_BY_NUMBER_FULL = 0x0e;

        /**
         * Process the request and response. Uses the model, authenticator, route
         * to manage the request appropriately.
         */
        protected void processRequest(HttpServletRequest request,
                        HttpServletResponse response) throws Exception {
                EFoods model = getModel(request);
                Route route = getRoute(request);
                PagingHelper paging = getPagination(request);
                SSOAuthenticator auth = (SSOAuthenticator) getAuthenticator(request);
                CartModel cart = CartModel.getInstance(request ,auth);
                List<ItemBean> results = null;
                int routeType;
                if (route == null)
                        routeType = -1;
                else
                        routeType = route.getIdentifier();


                switch (routeType) {
                case ROUTE_BY_CATEGORY:
                        int catId = Integer.parseInt(route.getMatcher().group("catId"));
                        CategoryBean cat = model.categories(catId).get(0);

                        results = model.items(ItemDAO.NUMBER_ALL, catId, paging.getPage(), paging.getLimit(), ItemDAO.NO_FILTER);
                        request.setAttribute("cartItems", cart.mappedCartItems());
                        request.setAttribute("category", cat);
                        request.setAttribute("results", results);
                        request.getRequestDispatcher("/partials/_items.jspx").forward(request,
                                        response);
                        break;
                case ROUTE_BY_NUMBER_FULL:
                        String number = route.getMatcher().group("itemNumber");
                        results = model.items(number, ItemDAO.CAT_ALL, ItemDAO.PAGE_ALL, ItemDAO.LIMIT_ALL, ItemDAO.NO_FILTER);
                        request.setAttribute("cartItems", cart.mappedCartItems());
                        request.setAttribute("category", results.get(0).getCategory());
                        request.setAttribute("results", results);
                        request.getRequestDispatcher("/partials/_items.jspx").forward(request,
                                        response);
                        break;
                case ROUTE_BY_NUMBER:
                        String number2 = route.getMatcher().group("itemNumber");
                        ItemBean item = model.items(number2, ItemDAO.CAT_ALL, ItemDAO.PAGE_ALL, ItemDAO.LIMIT_ALL, ItemDAO.NO_FILTER).get(0);
                        request.setAttribute("cartItem", cart.getCartItemFor(item));
                        request.setAttribute("item", item);
                        request.getRequestDispatcher("/partials/_item.jspx").forward(request,
                                        response);
                        break;
                case ROUTE_ALL:
                        results = model.items(ItemDAO.NUMBER_ALL, ItemDAO.CAT_ALL, paging.getPage(), paging.getLimit(), ItemDAO.NO_FILTER);
                        request.setAttribute("results", results);
                        request.getRequestDispatcher("/partials/_items.jspx").forward(request,
                                        response);
                        break;
                case ROUTE_BY_SEARCH:
                        String filter = request.getParameter("filter");
                        System.out.println(filter);
                        results = model.items(ItemDAO.NUMBER_ALL, ItemDAO.CAT_ALL, paging.getPage(), paging.getLimit(), filter);
                        request.setAttribute("results", results);
                        request.getRequestDispatcher("/partials/_items.jspx").forward(request,
                                        response);
                        break;
                default:
                        throw new Exception("routing error! not suppose to be here");
                }
        }



}
```

```java
package ctrl;

import java.io.IOException;

import javax.servlet.Servlet;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.tomcat.util.codec.binary.Base64;

import util.Route;

/**
 * Servlet implementation class Cart
 */

public class Misc extends BaseCtrl implements Servlet {
        private static final long serialVersionUID = 1L;
        public static final int ERROR_PAGE = 0x0a;
    /**
     * @see BaseCtrl#BaseCtrl()
     */
    public Misc() {
        super();
        // TODO Auto-generated constructor stub
    }


        @Override
        protected void processRequest(HttpServletRequest request,
                        HttpServletResponse response) throws ServletException, IOException {
                Route route = getRoute(request);

                int identifier = ERROR_PAGE;

                switch(identifier) {
                case ERROR_PAGE:
                        String errorMessage;
                        String encodedError = (String) request.getAttribute("encodedError");
                        if (encodedError == null) {
                                try {
                                        encodedError = route.getMatcher().group("Base64EncodedMessage");
                                        errorMessage= new String(Base64.decodeBase64(encodedError));
                                } catch (Exception e) {
                                        errorMessage = "Something went wrong somewhere! :(";
                                }
                        } else {
                                errorMessage= new String(Base64.decodeBase64(encodedError));
                        }

                        request.setAttribute("errorMessage", errorMessage);
                        request.getRequestDispatcher("/partials/_serverError.jspx").forward(
                                        request, response);
                        break;
                default:
                        throw new ServletException("Routing configuration error.");
                }

        }







}
```

```java
package ctrl;

import java.io.StringWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.transform.stream.StreamResult;

import model.POFileWrapper;
import model.POFilesWrapper;
import util.PurchaseOrderFiles;
import util.PurchaseOrderFiles.PurchaseOrderFile;
import util.Route;

public class Orders extends BaseCtrl {

        /**
         * 
         */
        private static final long serialVersionUID = 1L;

        public static final int ROUTE_ALL_NEW = 0x0a;
        public static final int ROUTE_UPDATE_STATUS = 0x0b;
        public static final int ROUTE_NUKE = 0x0c;
        public static final String BASE_PATH = "/PurchaseOrders/";



        @Override
        protected synchronized void processRequest(HttpServletRequest request,
                        HttpServletResponse response) throws Exception {
                Route route = getRoute(request);
                String basePath = request.getServletContext().getRealPath(BASE_PATH);
                String baseUrl = request.getContextPath() + BASE_PATH;
                PurchaseOrderFiles pofs;

                switch(route.getIdentifier()) {
                case ROUTE_ALL_NEW:
                        pofs = new PurchaseOrderFiles(basePath);
                        List<PurchaseOrderFile> pofList = pofs.getOrdersByStatus(PurchaseOrderFile.STATUS_NEW);
                        List<POFileWrapper> pofWrapperList = new ArrayList<POFileWrapper>();
                        for(PurchaseOrderFile pof: pofList) {
                                pofWrapperList.add(new POFileWrapper(baseUrl + pof.getFileNameOnly(),
                                                pof.getOrderId()));
                        }
                        POFilesWrapper pofsWrapper = new POFilesWrapper(pofWrapperList);
                        String xmlOut = getXML(pofsWrapper);
                        response.getWriter().write(xmlOut);

                        break;
                case ROUTE_UPDATE_STATUS:
                        int orderId = Integer.parseInt(route.getMatcher().group("orderId"));
                        String status = request.getParameter("status");
                        pofs = new PurchaseOrderFiles(basePath);
                        pofs.updateStatus(orderId, status);
                        break;
                case ROUTE_NUKE:
                                pofs = new PurchaseOrderFiles(basePath);
                                pofs.nuke();
                        break;
                default:
                        throw new ServletException("Routing configuration error.")
;
                }

        }

        private String getXML(Object wrapper) throws Exception {
                StringWriter sWriter = null;
                try {
                        JAXBContext jaxbCtx = JAXBContext.newInstance(wrapper
                                        .getClass());
                        Marshaller marshaller = jaxbCtx.createMarshaller();

                        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                                        Boolean.TRUE);
                        sWriter = new StringWriter();

                        marshaller.marshal(wrapper, new StreamResult(sWriter));

                        // add the stylesheet
                        StringBuffer sb = new StringBuffer(sWriter.toString());


                        return sb.toString();
                } catch (Exception e) {
                        throw e;
                } finally {
                        if (sWriter != null)
                                sWriter.close();
                }


        }

}
```

```java
package filter;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.DispatcherType;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;

import model.CartModel.CartItem;
import model.EFoods;
import model.ItemBean;
import model.ItemDAO;

/**
 * Servlet Filter implementation class CrossSell
 */
@WebFilter(dispatcherTypes = { DispatcherType.REQUEST, DispatcherType.FORWARD, DispatcherType.INCLUDE }, urlPatterns = { "/backend/cart", "/partials/_item.jspx" })
@SuppressWarnings("unchecked")
public class CrossSell implements Filter {

        @SuppressWarnings("serial")
        private static Map<String, String> TO_CROSS_SELL = new HashMap<String, String>() {
                {
                        put("0905A044", "0905A123"); //Add cross sell items <ItemAddedToCart,ItemToCrossSell>
                        put("0905A112", "0905A127");
                        put("0905A123", "0905A112");
                        put("0905A127", "0905A044");
                        put("1409S413", "2002H712");
                }
        };

        public static String ACTIVE_CROSS_SELLS_MAP_KEY = "com.eFoods.filter.ACSMap";

        /**
         * Default constructor.
         */
        public CrossSell() {
                // TODO Auto-generated constructor stub
        }

        /**
         * @see Filter#destroy()
         */
        public void destroy() {
                // TODO Auto-generated method stub
        }

        /**
         * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
         */
        public void doFilter(ServletRequest request, ServletResponse response,
                        FilterChain chain) throws IOException, ServletException {
                // TODO Auto-generated method stub
                // place your code here
                try {
                        HttpServletRequest httpRequest = (HttpServletRequest) requ
```

```java
est;
            HttpServletResponse httpResponse = (HttpServletResponse) response;
            if (!(httpRequest == null || httpResponse == null)) {
                if (httpRequest.getPathInfo() != null && httpRequest.getPathInfo().matches("^/cart(/)?"))
                    filterCart(httpRequest, httpResponse, chain);
                else
                    filterItem(httpRequest, httpResponse, chain);
            } else {
                chain.doFilter(request, response);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
            chain.doFilter(request, response);
        }
    }

    /**
     * @see Filter#init(FilterConfig)
     */
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }

    private void filterCart(HttpServletRequest request, HttpServletResponse response, FilterChain chain) throws IOException, ServletException {
        if (request.getMethod().equals("POST")) {
            try {
                int quantity = Integer.parseInt(request.getParameter("qty"));
                String number = (String) request.getParameter("item");
                if (!number.equals("")) {
                    Map<String, String> activeCrossSells = getCrossSellMap(request);
                    if (quantity > 0) { // Added to cart
                        System.out.println("Added to cart moment");
                        if (TO_CROSS_SELL.get(number) != null)
                            activeCrossSells.put(number, TO_CROSS_SELL.get(number));
                    } else {
                        System.out.println("Removed from cart moment");
                        activeCrossSells.remove(number);
                    }
                    setCrossSellMap(request, activeCrossSells);
                } else {
                    throw new Exception("Empty item number sent from client");
                }
            } catch (Exception e) {
                System.out.println(e.getMessage());
                e.printStackTrace();
            }
        }
        chain.doFilter(request, response);
    }

    private void filterItem(HttpServletRequest request, HttpServletResponse response, FilterChain chain) throws IOException, ServletException {
        if (request.getMethod().equals("GET")) {
            try {
                CartItem ci = (CartItem) request.getAttribute("car
tItem");
				EFoods model = (EFoods) request.getServletContext().getAttribute(ctrl.Front.MODEL_KEY);

				Map<String, String> activeCrossSells = getCrossSellMap(request);

				if (ci != null && activeCrossSells != null) {
					String crossSellThis = activeCrossSells.get(ci.getItem().getNumber());
					if (crossSellThis != null) {
						ItemBean crossSellItem = model.items(crossSellThis, ItemDAO.CAT_ALL, ItemDAO.PAGE_ALL, ItemDAO.LIMIT_ALL, ItemDAO.NO_FILTER).get(0);
						request.setAttribute("crossSellItem", crossSellItem);
						RequestDispatcher rd = request.getRequestDispatcher("/partials/_crossSellItem.jspx");
						if (request.getDispatcherType().equals(DispatcherType.FORWARD))
							rd.forward(request, response);
						else if (request.getDispatcherType().equals(DispatcherType.INCLUDE))
							rd.include(request, response);
						return;
					}
				}
			} catch (Exception e) {
				System.out.println(e.getMessage());
				e.printStackTrace();
			}
		}
		chain.doFilter(request, response);
	}

	private Map<String,String> getCrossSellMap(HttpServletRequest request) {
		Map<String, String> csm = (HashMap<String,String>) request
				.getSession().getAttribute(ACTIVE_CROSS_SELLS_MAP_KEY);
		return csm == null ? new HashMap<String, String>() : csm;
	}

	private void setCrossSellMap(HttpServletRequest request, Map<String,String> csm) {
		request.getSession().setAttribute(ACTIVE_CROSS_SELLS_MAP_KEY, csm);
	}

}
```

```java
package listener;

import javax.servlet.annotation.WebListener;
import javax.servlet.http.HttpSessionAttributeListener;
import javax.servlet.http.HttpSessionBindingEvent;

/**
 * Application Lifecycle Listener implementation class TimeStatistic
 *
 */
@WebListener
public class TimeStatistician implements HttpSessionAttributeListener {

    /**
     * Default constructor.
     */
    public TimeStatistician() {
        // TODO Auto-generated constructor stub
    }

        /**
     * @see HttpSessionAttributeListener#attributeRemoved(HttpSessionBindingEvent)
     */
    public void attributeRemoved(HttpSessionBindingEvent se)  {
         // DO NOTHING!
    }

        /**
     * @see HttpSessionAttributeListener#attributeAdded(HttpSessionBindingEvent)
     */
    public void attributeAdded(HttpSessionBindingEvent se)  {
         if (se.getName().equals("cartItemsCount"))
         {
                 long frsVstTime = se.getSession().getCreationTime();           // fresh visit time
                 long curTime = System.currentTimeMillis();     // current time (when first item added)

                 updateAvgAddItemTime(se, curTime - frsVstTime);

                 se.getSession().setAttribute("cartLastManipTime", curTime);
         }
         else if (se.getName().equals("checkedOut"))    // when client has checked out
         {
                 long frsVstTime = se.getSession().getCreationTime();           // fresh visit time
                 long curTime = System.currentTimeMillis();     // current time (when client checked out)

                 updateAvgCheckOutTime(se, curTime - frsVstTime);
         }
    }

        /**
     * @see HttpSessionAttributeListener#attributeReplaced(HttpSessionBindingEvent)
     */
    public void attributeReplaced(HttpSessionBindingEvent se)  {
         if (se.getName().equals("cartItemsCount"))
         {
                 long lstMnpTime = (long) se.getSession().getAttribute("cartLastManipTime");      // item last added time
                 long curTime = System.currentTimeMillis();     // current time
                 int oldCount = (int) se.getValue();
                 int newCount = (int) se.getSession().getAttribute("cartItemsCount");

                 // when there is item added to the cart
                 if (newCount > oldCount)
```

```java
                        {
                                updateAvgAddItemTime(se, curTime - lstMnpTime);
                                se.getSession().setAttribute("cartLastManipTime", curTime);
                        }
                }
        }

        // Do not remove static, since we need class-wide synchronization
        private static synchronized void updateAvgAddItemTime(HttpSessionBindingEvent evn, long duration)
        {
                if (evn.getSession().getServletContext().getAttribute("avgAddItemTime") == null)
                {
                        // when the very first item added
                        evn.getSession().getServletContext().setAttribute("avgAddItemTime", duration);
                        evn.getSession().getServletContext().setAttribute("NItemsAdded", (long) 1);
                }
                else
                {
                        long avgTime = (long) evn.getSession().getServletContext().getAttribute("avgAddItemTime");
                        long N = (long) evn.getSession().getServletContext().getAttribute("NItemsAdded");

                        // cast to double for computation
                        double durAcc = (new Long(duration)).doubleValue();
                        double avgTimeAcc = (new Long(avgTime)).doubleValue();
                        double NAcc = (new Long(N)).doubleValue();
                        // update statistic
                        double newAvgTime = (NAcc / (NAcc + 1.0)) * avgTimeAcc + (1.0 / (NAcc + 1.0)) * durAcc;

                        evn.getSession().getServletContext().setAttribute("avgAddItemTime", (new Double(newAvgTime)).longValue());
                        evn.getSession().getServletContext().setAttribute("NItemsAdded", N + 1);
                }
        }

        // Do not remove static, since we need class-wide synchronization
        private static synchronized void updateAvgCheckOutTime(HttpSessionBindingEvent evn, long duration)
        {
                if (evn.getSession().getServletContext().getAttribute("avgCheckOutTime") == null)
                {
                        // when the very order placed
                        evn.getSession().getServletContext().setAttribute("avgCheckOutTime", duration);
                        evn.getSession().getServletContext().setAttribute("NCheckedOut", (long) 1);
                }
                else
                {
                        long avgTime = (long) evn.getSession().getServletContext().getAttribute("avgCheckOutTime");
                        long N = (long) evn.getSession().getServletContext().getAttribute("NCheckedOut");

                       // cast to double for computation
                        double durAcc = (new Long(duration)).doubleValue();
                        double avgTimeAcc = (new Long(avgTime)).doubleValue();
                        double NAcc = (new Long(N)).doubleValue();
                        // update statistic
                        double newAvgTime = (NAcc / (NAcc + 1.0)) * avgTimeAcc + (1.0 / (NAcc + 1.0)) * durAcc;

                        evn.getSession().getServletContext().setAttribute("avgCheckOutTime", (new Double(newAvgTime)).longValue());
                        evn.getSession().getServletContext().setAttribute("NCheckedOut", N + 1);
                }
        }
}
```

```java
package model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.Iterator;
import java.util.List;
import java.util.Queue;

public class BaseDAO {


        public final static int ID_ALL = -1;
        public final static int PAGE_ALL = 1;
        public final static int LIMIT_ALL = -1;
        public BaseDAO() {
                // TODO Auto-generated constructor stub
        }

        protected class PrepareInstruction {
                static final int TYPE_STRING = 0x0a;
                static final int TYPE_INT = 0x0b;
                int type;
                Object value;
                public PrepareInstruction(int type, Object value) {
                        super();
                        this.type = type;
                        this.value = value;
                }
        }

        /**
         * Creates a new string based a list of where clauses, this list
         * is joined by the operator.
         *
         * @param wheres
         * @param operator
         * @return A string joined by the passed operator
         */
        protected String createWhereString(List<String> wheres, String operator) {
                String whereString = "";
                Iterator<String> iter = wheres.iterator();
        whereString += iter.hasNext() ? iter.next() : "";
        while (iter.hasNext()) whereString += " "+ operator + " " + iter.next();
        return "(" + whereString + ")";
        }

        /**
         *
         * @param query
         * @param piQ
         * @param connection
         * @return The prepared statement.
         * @throws Exception
         */
        protected PreparedStatement instaPrepareStatement(String query, Queue<Prep
areInstruction> piQ, Connection connection) throws Exception {
                PreparedStatement ps = connection.prepareStatement(query);
                int idx=0;
                while (piQ.size() > 0) {
                        PrepareInstruction pi = piQ.poll();
                        idx++; //Start with 1
                        switch (pi.type) {
                        case PrepareInstruction.TYPE_STRING:
                                ps.setString(idx, (String) pi.value);
                                break;
                        case PrepareInstruction.TYPE_INT:
                                ps.setInt(idx, (int) pi.value);
                                break;
                        default:
                                throw new Exception("Instructions are invalid");
                        }
                }
                return ps;
        }

        /**
         * Creates a string for paging. This is used to fetch
         * a specific amount of rows from the database.
         *
         * @param page
         * @param limit
         * @return
         */
        protected String pagingString(int page, int limit) {
                String ret = "";

                if (page > 0) {
                        ret += " OFFSET " + (page - 1) * limit + " ROWS ";
                        if (limit != LIMIT_ALL) {
                                ret += " FETCH NEXT " + limit + " ROWS ONLY";
                        }
                }

                return ret;
        }
}
```

```
package model;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;

import model.CartModel.CartItem;

public class CartItemBean {

        @XmlAttribute
        String number;

        @XmlElement
        String name;

        @XmlElement
        float price;

        @XmlElement
        int quantity;

        @XmlElement
        String extended;

        public CartItemBean() {
                // TODO Auto-generated constructor stub
        }
        public CartItemBean(CartItem cartItem) {
                this.number = cartItem.getItem().getNumber();
                this.name = cartItem.getItem().getName();
                this.price = cartItem.getItem().getPrice();
                this.quantity = cartItem.getQuantity();
                this.extended = cartItem.getItemTotal();
        }

}
```

```
package model;

import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

import model.CartModel.CartItem;

@XmlRootElement
public class CartItemsWrapper
{
        @XmlElement(name="item")
        private List<CartItemBean> itemsList;

        public CartItemsWrapper()
   {
           super();
           // TODO Auto-generated constructor stub
   }

        public CartItemsWrapper(List<CartItem> cartItems)
        {
                List<CartItemBean> cartItemsList = new ArrayList<CartItemBean>();
                for(CartItem ci: cartItems)
                        cartItemsList.add(new CartItemBean(ci));

                this.itemsList = cartItemsList;
        }
}
```

```java
package model;

import java.math.BigDecimal;
import java.math.MathContext;
import java.math.RoundingMode;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import util.SSOAuthenticator;

public class CartModel {

        public class CartItem {
                ItemBean item;
                int quantity;

                /**
                 * @return the item
                 */
                public ItemBean getItem() {
                        return item;
                }

                /**
                 * @return the quantity
                 */
                public int getQuantity() {
                        return quantity;
                }

                public CartItem(ItemBean item, int quantity) {
                        super();
                        this.item = item;
                        this.quantity = quantity;
                }

                public String getItemTotal() {
                        return (new BigDecimal(Float.toString(item.price))).multip
ly(new BigDecimal(quantity)).toPlainString();
                }

        }

        private static final BigDecimal TAX_MULTIPLIER = new BigDecimal("0.13");
        private static final BigDecimal FREE_SHIPPING_LIMIT = new BigDecimal("100.
00");
        private static final BigDecimal SHIPPING_CHARGE = new BigDecimal("5.00");
        private static final String CART_SESSION_KEY = "com.eFoods.model.Cart";

        private List<CartItem> cartItems;
        private UserBean account;
        private BigDecimal total;
        private BigDecimal shipping;
        private BigDecimal tax;

        public CartModel(UserBean account) {
                super();
                this.cartItems = new ArrayList<CartItem>();
                this.account = account;
                this.total = BigDecimal.ZERO;
                this.shipping = SHIPPING_CHARGE;
                this.tax = BigDecimal.ZERO;
        }

        public synchronized void manipulateCart(ItemBean item, int qty)
```

```java
            throws Exception {
        try {
            for (Iterator<CartItem> i = cartItems.listIterator(); i.hasNext();) {
                CartItem ci = i.next();
                if (ci.item.equals(item)) {
                    if (qty < 1) {
                        i.remove();
                        System.out.println("Cart\t: Item removed. "
                                + item.number);
                    } else {
                        ci.quantity = qty;
                        System.out.println("Cart\t: Item qty updated to " + qty
                                + " for " + item.number);
                    }
                    return;
                }
            }
            if (qty > 0) {
                cartItems.add(new CartItem(item, qty));

                System.out.println("Cart\t: Item added. " + qty + " of "
                        + item.number);
            } else {
                throw new Exception(
                        "Invalid operation. Cannot set qty to 0 when it doesn’t exist, NOOB");
            }
        } finally {
            computeCart();
        }

    }

    private void computeCart() {
        this.total = calculateTotal();
        this.shipping = calculateShipping();
        this.tax = calculateTax();

        System.out.println("Cart : Refreshed. Total:" + total + " Shipping:"
                + shipping + " Tax:" + tax);
    }

    private BigDecimal calculateShipping() {
        if (isFreeShipping())
            return round(BigDecimal.ZERO, 4);
        return round(SHIPPING_CHARGE, 4);
    }

    private BigDecimal calculateTotal() {
        BigDecimal t = BigDecimal.ZERO;
        for (CartItem ci : cartItems) {
            System.out.println(new BigDecimal(ci.item.getPrice()));
            System.out.println(new BigDecimal(ci.quantity));
            t = t.add(new BigDecimal(ci.item.getPrice()).multiply(
                    new BigDecimal(ci.quantity)));
            System.out.println(t);
        }
        System.out.println(round(t,4));
        return round(t, 4);
    }

    private BigDecimal calculateTax() {
        return round((isFreeShipping() ? total.multiply(TAX_MULTIPLIER) : total.add(shipping).multiply(TAX_MULTIPLIER)), 3);
        }

        /**
         *
         * @param number
         * @param places
         * @return rounded BigDecimal to passed number of places.
         */
        private BigDecimal round(BigDecimal number, int places) {
                return number.round(new MathContext(places, RoundingMode.HALF_UP))
;
        }

        public boolean isFreeShipping() {
                return (total.compareTo(FREE_SHIPPING_LIMIT) >= 0);
        }

        /**
         * @param cartItems
         *            the cartItems to set
         */
        public List<CartItem> getCartItems() {
                return cartItems;
        }

        /**
         * Gets the CartItem object for the given item
         *
         * @param item
         * @return corresponding CartItem if found, null if doesnt exist in cart.
         */
        public CartItem getCartItemFor(ItemBean item) {
                for (CartItem ci : cartItems)
                        if (ci.item.equals(item))
                                return ci;
                return null;
        }

        /**
         * Creates a map of cart itesm so that it could be accessed directly from
item number
         * @return map containing item number as key and cartitem as value
         */
        public Map<String, CartItem> mappedCartItems() {
                Map<String, CartItem> map = new LinkedHashMap<String, CartItem>();
                for (CartItem i : this.getCartItems())
                        map.put(i.getItem().getName(), i);
                return map;
        }

        /**
         * @param wot
         */
        public void setCartItems(List<CartItem> cartItems) {
                this.cartItems = cartItems;
        }

        /**
         * @return the account
         */
        public UserBean getAccount() {
                return account;
        }

        /**
         * @param account
         *            the account to set
         */
        public void setAccount(UserBean account) {
                if ((this.account == null || !this.account.equals(account)) && acc
```

```java
ount != null)
            System.out.println("CartModel: Ownership changed to "
                    + account.name);
        this.account = account;
    }

    /**
     * @return the total
     */
    public BigDecimal getTotal() {
        return total;
    }

    /**
     * @param total
     *            the total to set
     */
    public void setTotal(BigDecimal total) {
        this.total = total;
    }

    /**
     * @return the shipping
     */
    public BigDecimal getShipping() {
        return shipping;
    }

    /**
     * @param shipping
     *            the shipping to set
     */
    public void setShipping(BigDecimal shipping) {
        this.shipping = shipping;
    }

    /**
     * @return the tax
     */
    public BigDecimal getTax() {
        return tax;
    }

    /**
     * @param tax
     *            the tax to set
     */
    public void setTax(BigDecimal tax) {
        this.tax = tax;
    }

    public static CartModel getInstance(HttpServletRequest request,
            SSOAuthenticator auth) {
        CartModel dis;
        if ((dis = (CartModel) request.getSession().getAttribute(
                CART_SESSION_KEY)) == null) {
            synchronized (CartModel.class) {
                if ((dis = (CartModel) request.getSession().getAttribute(
                        CART_SESSION_KEY)) == null) {
                    dis = new CartModel(null);
                    request.getSession().setAttribute(CART_SESSION_KEY, dis);
                    System.out.println("Cart: Initialized");
                }
            }
        }

        dis.setAccount(auth.getUser(request));
        return dis;
```

```
	}

	public void clear() {
		this.cartItems = new ArrayList<CartItem>();
		computeCart();
	}

	/**
	 *
	 * @return The grandTotal is cumulative amount of tax + shipping + total
	 */
	public BigDecimal getGrandTotal() {
		return tax.add(total).add(shipping);
	}

}
```

```java
package model;

public class CategoryBean {
	private int id;
	private String name;
	private String description;
	private byte[] picture;


	public CategoryBean(int id, String name, String description, byte[] picture) {
		super();
		this.id = id;
		this.name = name;
		this.description = description;
		this.picture = picture;
	}


	/**
	 * @return the id
	 */
	public int getId() {
		return id;
	}

	/**
	 * @param id the id to set
	 */
	public void setId(int id) {
		this.id = id;
	}

	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * @return the description
	 */
	public String getDescription() {
		return description;
	}

	/**
	 * @param description the description to set
	 */
	public void setDescription(String description) {
		this.description = description;
	}

	/**
	 * @return the picture
	 */
	public byte[] getPicture() {
		return picture;
	}

	/**
	 * @param picture the picture to set
	 */
	public void setPicture(byte[] picture) {
		this.picture = picture;
	}


}
```

```java
package model;

import java.math.BigInteger;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;


import util.eFoodsDataSource;
public class CategoryDAO extends BaseDAO {



        public CategoryDAO() {}

        public List<CategoryBean> retrieve() throws Exception {
                return retrieve(ID_ALL);
        }




        public List<CategoryBean> retrieve(int id) throws Exception {
                Connection connection = null;
                PreparedStatement preparedStatement;
                ResultSet resultSet;
                List<CategoryBean> retval = new ArrayList<CategoryBean>();



                try {
                        connection = eFoodsDataSource.getConnection();

                        if(id==ID_ALL) {
                                String query = "SELECT id, name, description, picture FROM category";
                                preparedStatement = connection.prepareStatement(query);
                        } else {
                                String query = "SELECT id, name, description, picture FROM category WHERE id = ?";
                                preparedStatement = connection.prepareStatement(query);
                                preparedStatement.setInt(1,  id);
                        }

                        resultSet = preparedStatement.executeQuery();

                        while (resultSet.next()) {
                                retval.add(new CategoryBean(resultSet.getInt("id"),
                                                resultSet.getString("name"),
                                                resultSet.getString("description"),
                                                new BigInteger(resultSet.getString("picture"), 16).toByteArray()));
                        }
                } catch (Exception e) {
                        throw e;
                } finally {
                        if (connection != null)
                                connection.close();
                }

                return retval;
                        }

}
```

```
package model;

import java.io.FileWriter;
import java.io.StringWriter;
import java.util.List;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.transform.stream.StreamResult;

import util.PurchaseOrderFiles;

public class EFoods {
        private CategoryDAO categoryDAO;
        private ItemDAO itemDAO;
        public EFoods() {
                        categoryDAO = new CategoryDAO();
                        itemDAO = new ItemDAO();
        }

        public List<CategoryBean> categories(int id) throws Exception {
                return categoryDAO.retrieve(id);
        }

        public List<ItemBean> items(String number, int catId, int page, int limit, String filter) throws Exception {
                return itemDAO.retrieve(number, catId, page, limit, filter);
        }

        /**
         * Creates a purchase order based on the current items in the cart.
         *
         * @param cart
         * @param filepath
         * @return the purchase order id
         * @throws Exception
         */
        public synchronized int createPurchaseOrder(CartModel cart, String filepath) throws Exception {
                PurchaseOrderFiles pofs = new PurchaseOrderFiles(filepath);
                int orderId = pofs.getNextOrderId();
                PurchaseOrderWrapper poWrapper = new PurchaseOrderWrapper(cart, orderId);
                StringWriter sWriter = null;
                FileWriter fWriter = null;
                try {
                        JAXBContext jaxbCtx = JAXBContext.newInstance(poWrapper
                                        .getClass());
                        Marshaller marshaller = jaxbCtx.createMarshaller();

                        //marshaller.setProperty(Marshaller.JAXB_SCHEMA_LOCATION,
                        //              "SIS.xsd");
                        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                                        Boolean.TRUE);
                        marshaller.setProperty(Marshaller.JAXB_FRAGMENT, Boolean.TRUE);

                        sWriter = new StringWriter();
                        sWriter.write("<?xml version=’1.0’?>");
                        sWriter.write("\n");
                        sWriter.write("<?xml-stylesheet type=’text/xsl’ href=’PurchaseOrders.xsl’ ?>");
                        sWriter.write("\n");
                        marshaller.marshal(poWrapper, new StreamResult(sWriter));

                        pofs.storeNewOrder(orderId, cart.getAccount().getName(), sWriter.toString());
                        return orderId;
                } catch (Exception e) {
                        throw e;
                } finally {
                        if (fWriter != null)
                                fWriter.close();
                        if (sWriter != null)
                                sWriter.close();
                }

        }





}
```

```java
package model;

public class ItemBean {

	String number;
	String name;
	float price;
	int qty;
	int onOrder;
	int reOrder;
	CategoryBean category;
	int supplierId;
	int costPrice;
	String unit;

	public ItemBean(String number, String name, float price, int qty, int onOr
der,
				int reOrder, CategoryBean category, int supplierId, int co
stPrice,
				String unit) {
		super();
		this.number = number;
		this.name = name;
		this.price = price;
		this.qty = qty;
		this.onOrder = onOrder;
		this.reOrder = reOrder;
		this.category = category;
		this.supplierId = supplierId;
		this.costPrice = costPrice;
		this.unit = unit;
	}

	/**
	 * @return the number
	 */
	public String getNumber() {
		return number;
	}

	/**
	 * @param number the number to set
	 */
	public void setNumber(String number) {
		this.number = number;
	}

	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * @return the price
	 */
	public float getPrice() {
		return price;
	}

	/**
	 * @param price the price to set
	 */
	public void setPrice(float price) {
		this.price = price;
	}

	/**
	 * @return the qty
	 */
	public int getQty() {
		return qty;
	}

	/**
	 * @param qty the qty to set
	 */
	public void setQty(int qty) {
		this.qty = qty;
	}

	/**
	 * @return the onOrder
	 */
	public int getOnOrder() {
		return onOrder;
	}

	/**
	 * @param onOrder the onOrder to set
	 */
	public void setOnOrder(int onOrder) {
		this.onOrder = onOrder;
	}

	/**
	 * @return the reOrder
	 */
	public int getReOrder() {
		return reOrder;
	}

	/**
	 * @param reOrder the reOrder to set
	 */
	public void setReOrder(int reOrder) {
		this.reOrder = reOrder;
	}

	/**
	 * @return the category
	 */
	public CategoryBean getCategory() {
		return category;
	}

	/**
	 * @param category the category to set
	 */
	public void setCategory(CategoryBean category) {
		this.category = category;
	}

	/**
	 * @return the supplierId
	 */
	public int getSupplierId() {
		return supplierId;
	}

	/**
	 * @param supplierId the supplierId to set
	 */
	public void setSupplierId(int supplierId) {
		this.supplierId = supplierId;
	}

	/**
	 * @return the costPrice
	 */
	public int getCostPrice() {
		return costPrice;
	}

	/**
	 * @param costPrice the costPrice to set
	 */
	public void setCostPrice(int costPrice) {
		this.costPrice = costPrice;
	}

	/**
	 * @return the unit
	 */
	public String getUnit() {
		return unit;
	}

	/**
	 * @param unit the unit to set
	 */
	public void setUnit(String unit) {
		this.unit = unit;
	}

	/* (non-Javadoc)
	 * @see java.lang.Object#hashCode()
	 */
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((number == null) ? 0 : number.hashCode(
));
		return result;
	}

	/* (non-Javadoc)
	 * @see java.lang.Object#equals(java.lang.Object)
	 */
	@Override
	public boolean equals(Object obj) {
		if (this == obj) {
			return true;
		}
		if (obj == null) {
			return false;
		}
		if (!(obj instanceof ItemBean)) {
			return false;
		}
		ItemBean other = (ItemBean) obj;
		if (number == null) {
			if (other.number != null) {
				return false;
			}
		} else if (!number.equals(other.number)) {
			return false;
		}
		return true;
	}

```

```java
}
```

```
package model;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

import util.eFoodsDataSource;

public class ItemDAO extends BaseDAO {

        public final static String NUMBER_ALL = "-1";
        public final static int CAT_ALL = -1;
        public final static String NO_FILTER = "__!!__";

        public static final String BASE_QUERY = "SELECT  I.number, "
                        + "I.name,  I.price,  I.qty,  I.onorder, "
                        + "I.reorder,  I.catid,  I.supid,  I.costprice, "
                        + "I.unit,  C.id as \"CAT_ID\",  C.name as \"CAT_NAME\", "
                        + "C.description as \"CAT_DESCRIPTION\" FROM ITEM I "
                        + "INNER JOIN CATEGORY C  ON I.catId = C.id";

        public ItemDAO() {}

        /**
         * Retrieves all the items from the database.
         * @return A List containing all the items.
         * @throws Exception
         */
        public List<ItemBean> retrieve() throws Exception {
                return retrieve(NUMBER_ALL);
        }


        /**
         * Retrieves a single item from the database.
         * @param number
         * @return A List containing the single item.
         * @throws Exception
         */
        public List<ItemBean> retrieve(String number) throws Exception {
                return retrieve(number, CAT_ALL, PAGE_ALL, LIMIT_ALL, NO_FILTER);
        }

        /**
         * Creates a query for the DB based on the passed parameters.
         *
         * @param number
         * @param catId
         * @param page
         * @param limit
         * @param filter
         * @return The list of the items passed on the generated query.
         * @throws Exception
         */
        public List<ItemBean> retrieve(String number, int catId, int page, int lim
it, String filter) throws Exception {
                Connection connection = null;
                PreparedStatement preparedStatement;
                ResultSet rs;
                List<ItemBean> retval = new ArrayList<ItemBean>();

                try {
                        connection = eFoodsDataSource.getConnection();


                                        Queue<PrepareInstruction> instructions=new LinkedL
ist<PrepareInstruction>();
                                List<String> wheres = new ArrayList<String>();

                                if (!(number.equals(NUMBER_ALL))) {
                                        wheres.add("I.number = ?");
                                        instructions.add(new PrepareInstruction(PrepareInstruction.TYPE_STRING, number));
                                }
                                if (catId != ID_ALL) {
                                        wheres.add("C.id = ?");
                                        instructions.add(new PrepareInstruction(PrepareInstruction.TYPE_INT, catId));
                                }

                                if (!filter.equals(NO_FILTER)) {
                                        wheres.add("LOWER(I.name) LIKE LOWER( ? )");
                                        instructions.add(new PrepareInstruction(PrepareInstruction.TYPE_STRING, "%" + filter + "%"));
                                }
                                String query = BASE_QUERY;
                                if (wheres.size() > 0) { query += " WHERE " + createWhereString(wheres, "AND"); }

                        query += pagingString(page, limit);
                        System.out.println("DB Query: " + query.replaceFirst(BASE_QUERY, ""));
                                preparedStatement = instaPrepareStatement(query, instructions, connection);



                        rs = preparedStatement.executeQuery();

                        while (rs.next()) {
                                CategoryBean cat = new CategoryBean(rs.getInt("cat_id"),
                                                rs.getString("cat_name"),
                                                rs.getString("cat_description"), null);
                                ItemBean item = new ItemBean(rs.getString("number"),
                                                rs.getString("name"), rs.getFloat("price"),
                                                rs.getInt("qty"), rs.getInt("onorder"),
                                                rs.getInt("reorder"), cat, rs.getInt("supid"),
                                                rs.getInt("costprice"), rs.getString("unit"));
                                retval.add(item);
                        }
                } catch (Exception e) {
                        throw e;
                } finally {
                        if (connection != null)
                                connection.close();
                }

                return retval;
        }

}
```

```java
package model;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="PurchaseOrderFiles")
public class POFilesWrapper {

        @XmlElement(name="PurchaseOrderFile")
        List<POFileWrapper> pofList;
        public POFilesWrapper() {
                // TODO Auto-generated constructor stub
        }

        public POFilesWrapper(List<POFileWrapper> pofList) {
                this.pofList = pofList;
        }

}
```

```java
package model;

import javax.xml.bind.annotation.XmlElement;

public class POFileWrapper {
        @XmlElement
        String url;

        @XmlElement
        int id;

        /**
         * @return the url
         */
        public String getUrl() {
                return url;
        }
        /**
         * @return the id
         */
        public int getId() {
                return id;
        }

        public POFileWrapper() {
                // TODO Auto-generated constructor stub
        }

        public POFileWrapper(String url, int i) {
                super();
                this.url = url;
                this.id = i;

        }
}
```

```java
package model;
import java.util.Date;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlSchemaType;

import model.CartModel;

@XmlRootElement(name="order")
public class PurchaseOrderWrapper {

        @XmlAttribute
        private int id;

        @XmlAttribute
        @XmlSchemaType(name="date")
        Date submitted;

        @XmlElement
        UserBean customer;

        @XmlElement(name="items")
        CartItemsWrapper items;

        @XmlElement
        String total;

        @XmlElement
        String shipping;

        @XmlElement
        String HST;

        @XmlElement
        String grandTotal;

        public PurchaseOrderWrapper() {};

        public PurchaseOrderWrapper(CartModel cart, int orderId) {
                this.id = orderId;
                this.submitted = new Date();
                this.customer = cart.getAccount();
                this.total = cart.getTotal().toPlainString();
                this.shipping = cart.getShipping().toPlainString();
                this.HST = cart.getTax().toPlainString();
                this.grandTotal = cart.getGrandTotal().toPlainString();
                this.items = new CartItemsWrapper(cart.getCartItems());
        }
        /**
         * @return the grandTotal
         */
        public String getGrandTotal() {
                return grandTotal;
        }

        /**
         * @return the total
         */
        public String getTotal() {
                return total;
        }

        /**
         * @return the shipping
         */
        public String getShipping() {
                return shipping;
        }

        /**
         * @return the hST
         */
        public String getHST() {
                return HST;
        }




}

```

```java
package model;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;

public class UserBean {

        @XmlAttribute(name="account")
        String name;


        String fullName;


        boolean admin;

        public UserBean() {

        }

        public UserBean(String name, String fullName, boolean admin) {
                super();
                this.name = name;
                this.fullName = fullName;
                this.admin = admin;
        }



        /**
         * @return the name
         */
        public String getName() {
                return name;
        }

        /**
         * @return the admin
         */
        public boolean isAdmin() {
                return admin;
        }


        /**
         * @return the fullName
         */
        @XmlElement(name="name")
        public String getFullName() {
                return fullName;
        }


}
```

```
package tags;

import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.SimpleTagSupport;

public class MinSec extends SimpleTagSupport
{
        private static int THOUSAND = 1000;
        private static int BASE = 60;
        private long milliseconds;

        public MinSec()
   {
           super();
   }

        @Override
   public void doTag() throws JspException, IOException
   {
           super.doTag();

                long seconds = milliseconds / THOUSAND;
                long m = seconds / BASE;
                long s = seconds % BASE;

                JspWriter out = this.getJspContext().getOut();
                if (m != 0)
                        out.write(m + "m");
                out.write(s + "s");
   }

        public void setMilliseconds(long milliseconds)
        {
                this.milliseconds = milliseconds;
        }
}
```

```java
package util;

import javax.servlet.http.HttpServletRequest;


public abstract class Authenticator {

        protected String sessionKey;

        public abstract boolean isAuthenticated(HttpServletRequest request);

        public abstract boolean login(HttpServletRequest request, String username,
                        String password) throws Exception;

        public abstract void logout(HttpServletRequest request) throws Exception;


}
```

```
package util;

import java.sql.Connection;
import java.sql.SQLException;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class eFoodsDataSource {

        private static final String DATASOURCE_CONTEXT = "java:/comp/env/jdbc/EECS";
        private static final String SCHEMA = "roumani";
        private static DataSource dataSource;


        /**
         * @return the dataSource
         * @throws NamingException
         */
        public static DataSource getDataSource() throws NamingException {
                if (dataSource == null) {
                        synchronized(eFoodsDataSource.class) {
                                if (dataSource == null) {
                                        Context initialContext = new InitialContext();
                                        dataSource = (DataSource) initialContext.lookup(DATASOURCE_CONTEXT);
                                }
                        }
                }
                return dataSource;
        }

        public static Connection getConnection() throws SQLException, NamingException {
                Connection c = getDataSource().getConnection();
                c.createStatement().executeUpdate("set schema " + SCHEMA);
                return c;
        }
        protected eFoodsDataSource() {}


}
```

```java
package util;

import java.io.File;
import java.io.FileWriter;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PurchaseOrderFiles {
        public class PurchaseOrderFile {
                public static final String STATUS_NEW = "new";
                public static final String STATUS_PENDING = "pending";
                public static final String STATUS_PURCHASED = "purchased";

                String fileName;
                int orderId;
                String userName;
                String status;
                String modifiedDate;
                /**
                 * @return the modifiedDate
                 */
                public String getModifiedDate() {
                        return modifiedDate;
                }

                /**
                 * @return the fileName
                 */
                public String getFileName() {
                        return fileName;
                }

                /**
                 * @return the oderId
                 */
                public int getOrderId() {
                        return orderId;
                }

                /**
                 * @return the userName
                 */
                public String getUserName() {
                        return userName;
                }

                /**
                 * @return the status
                 */
                public String getStatus() {
                        return status;
                }

                public String getFileNameOnly() {
                        return new File(fileName).getName();
                }

                /*
                 * This constructor will automatically determine the file name given the required information.
                 * Ensures that file names with illiegal schemes will not be saved.
                 */
                public PurchaseOrderFile(File basePath, int orderId, String userName,
                                String status) throws Exception {
                        String filename = orderId + "_" + userName + "_" + status + ".xml";
                        if (!filename.matches(MATCHER_REGEX))
                                throw new Exception("Order filename invalid. Check parameters. " + filename);
                        File leFile = new File(basePath, filename);
                        this.fileName = leFile.getAbsolutePath();
                        this.orderId = orderId;
                        this.userName = userName;
                        this.status = status;
                        this.modifiedDate = new Date(leFile.lastModified()).toString();
                }
        }
        protected static final String MATCHER_REGEX = "^(?<poId>[1-9][0-9]*)_(?<accountId>[A-Za-z0-9]+)_(?<statusId>new|pending|purchased)\\.xml$";



        private File basePath;
        private List<PurchaseOrderFile> purchaseOrderFiles;

        public PurchaseOrderFiles(String basePath) throws Exception {
                this.basePath = new File(basePath);
                this.purchaseOrderFiles = allOrderFiles();
        }

        /*
         * Traverses all files in the purchase order folder and creates a List of
         *  PurchaseOrderFile classes ready to be used by other methods.
         */
        public List<PurchaseOrderFile> allOrderFiles() throws Exception {
                File[] files = basePath.listFiles();
                List<PurchaseOrderFile> results = new ArrayList<PurchaseOrderFile>();
                List<Integer> idArray = new ArrayList<Integer>();
                for (File file : files) {
                        if (file.isFile()) {
                                if (!file.getName().endsWith(".xml")) continue; // Avoid stupid mac osx .DS_store files
                                Matcher matcher = Pattern.compile(MATCHER_REGEX).matcher(file.getName());
                                if (matcher.matches()) {
                                        int currId = Integer.parseInt(matcher.group("poId"));
                                        String account = matcher.group("accountId");
                                        String status = matcher.group("statusId");
                                        if (idArray.contains(currId))
                                                throw new Exception("Non unique id " + currId  );
                                        else
                                                idArray.add(currId);
                                        results.add(new PurchaseOrderFile(basePath,currId,account,status));
                                } else {
                                        throw new Exception("Malformed xml name " + file.getName());
                                }
                        }
                }
                return results;
        }

        /*
         * Finds the highest order id and return int added 1
         */
        public int getNextOrderId() throws Exception {
                int[] ids = new int[purchaseOrderFiles.size()];
```

```java
                int i = 0;
                for(PurchaseOrderFile pof: purchaseOrderFiles)
                        ids[i] = pof.getOrderId();
                        i++;
                Arrays.sort(ids);
                if (ids.length > 0) {
                        return ids[ids.length - 1] + 1;
                } else {
                        return 1;
                }
        }

        /*
         * Creates a new purchase order file
         */
        public void storeNewOrder(int orderId, String accountName, String order) throws Exception {
                int nextId = getNextOrderId();
                if (nextId != orderId)
                        throw new Exception("Order ID mismatch. ");
                PurchaseOrderFile pof = new PurchaseOrderFile(basePath, orderId, accountName, PurchaseOrderFile.STATUS_NEW);
                FileWriter fWriter = null;
                try {
                        fWriter = new FileWriter(pof.getFileName());
                        fWriter.write(order);
                        purchaseOrderFiles.add(pof);
                        System.out.println("PO: Stored purchase order " + pof.getFileName());
                } catch (Exception e) {
                        throw e;
                } finally {
                        if (fWriter != null)
                                fWriter.close();
                }

        }

        public List<PurchaseOrderFile> getOrdersByUser(String username) {
                List<PurchaseOrderFile> result = new ArrayList<PurchaseOrderFile>();
                for(PurchaseOrderFile pof : purchaseOrderFiles) {
                        if (pof.getUserName().equals(username))
                                result.add(pof);
                }
                return result;
        }

        public List<PurchaseOrderFile> getOrdersByStatus(String status) {
                List<PurchaseOrderFile> result = new ArrayList<PurchaseOrderFile>();
                for(PurchaseOrderFile pof : purchaseOrderFiles) {
                        if (pof.getStatus().equals(status))
                                result.add(pof);
                }
                return result;
        }

        public PurchaseOrderFile getOrderById(int orderId) {
                for(PurchaseOrderFile pof : purchaseOrderFiles) {
                        if (pof.getOrderId() == orderId)
                                return pof;
                }
                return null;
        }

        /*
         * Renames an order file in order to change it’s status
         */
        public void updateStatus(int orderId, String status) throws Exception {
                PurchaseOrderFile pof = getOrderById(orderId);
                if (pof != null) {
                        String fname = pof.getFileName();
                        PurchaseOrderFile newPof = new PurchaseOrderFile(basePath, orderId, pof.getUserName(), status);
                        String newFname = newPof.getFileName();
                        File oldFile = new File(fname);
                        File newFile = new File(newFname);
                        if (oldFile.renameTo(newFile)) {
                                this.purchaseOrderFiles = allOrderFiles();
                        } else {
                                throw new Exception("Could not rename file");
                        }
                } else {
                        throw new Exception("Order ID does not exist");
                }

        }

        /*
         * Deletes all order files in the system. Destructive action.
         */
        public void nuke() throws Exception {
                for(PurchaseOrderFile pof : purchaseOrderFiles) {
                        new File(pof.getFileName()).delete();
                }
                this.purchaseOrderFiles = allOrderFiles();
        }


}
```

```java
package util;

import java.util.regex.*;

import javax.servlet.http.HttpServletRequest;

public class Route {

        public static final String METHOD_GET = "GET";
        public static final String METHOD_POST = "POST";
        public static final String METHOD_ANY = "ANYTHING";

        private String urlPattern;
        private String destination;
        private String method;
        private int destinationType;
        private boolean requireAuthentication;
        private Matcher matcher;
        private int identifier;
        private boolean requireAdmin;

        /**
         * @return the matcher
         */
        public Matcher getMatcher() {
                return matcher;
        }

        public Route(String urlPattern, String destination, String method, int ide
ntifier,
                                boolean requireAuthentication) {
                this(urlPattern, destination, method, identifier, requireAuthentic
ation, false);
        }

        public Route(String urlPattern, String destination, String method, int ide
ntifier,
                                boolean requireAuthentication, boolean requireAdmin) {
                super();
                this.urlPattern = urlPattern;
                this.destination = destination;
                this.method = method;
                this.identifier = identifier;
                this.requireAuthentication = requireAuthentication;
                this.requireAdmin = requireAdmin;
        }

        /**
         * @return the identifier
         */
        public int getIdentifier() {
                return identifier;
        }

        /**
         * @param identifier the identifier to set
         */
        public void setIdentifier(int identifier) {
                this.identifier = identifier;
        }

        /**
         * @return the urlPattern
         */
        public String getUrlPattern() {
                return urlPattern;
        }

        /**
         * @return the destination
```

```java
	 */
	public String getDestination() {
		return destination;
	}

	/**
	 * @return the destinationType
	 */
	public int getDestinationType() {
		return destinationType;
	}

	/**
	 * @return the method
	 */
	public String getMethod() {
		return method;
	}

	/**
	 * @return the requireAuthentication
	 */
	public boolean isRequireAuthentication() {
		return requireAuthentication;
	}

	/**
	 * @return the requireAdmin
	 */
	public boolean isRequireAdmin() {
		return requireAdmin;
	}

	/**
	 * @param requireAdmin the requireAdmin to set
	 */
	public void setRequireAdmin(boolean requireAdmin) {
		this.requireAdmin = requireAdmin;
	}

	/*
	 * Tests if the given http request matches to this route
	 */
	public boolean match(HttpServletRequest request) {
		Matcher m = Pattern.compile(this.urlPattern).matcher(request.getPa
thInfo());
		if (m.matches() && verifyMethod(request.getMethod())) {
			this.matcher = m;
			return true;
		} else {
			return false;
		}
	}

	private boolean verifyMethod(String m) {
		if (m == null || this.method.equals(METHOD_ANY)
				|| this.method.equals(m))
			return true;
		else
			return false;

	}

}
```

```java
package util;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

public class Router {

        public static final String REQUEST_ROUTE_KEY = "request-route";

        List<Route> routes;

        public Router() {
                this.routes = new ArrayList<Route>();
        }

        public void addRoute(Route route) {
                routes.add(route);
        }

        /**
         * Loop through all configured routes and find one that matches.
         * @param request
         * @return the route with respect to the request
         */
        public Route getRoute(HttpServletRequest request) {
                for (Route r : routes)
                        if (r.match(request)){
                                request.setAttribute(REQUEST_ROUTE_KEY, r);
                                return r;
                        }

                return null;
        }

        /**
         * @return the routes
         */
        public List<Route> getRoutes() {
                return routes;
        }

}
```

```java
package util;

import java.net.URLEncoder;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.xml.bind.DatatypeConverter;

import org.apache.catalina.util.RequestUtil;
import org.apache.tomcat.util.codec.binary.Base64;

import model.UserBean;

public class SSOAuthenticator extends Authenticator {

        private static final String SESSION_KEY_PREFIX = "com.eFoods.util.Authentication.SSO";
        private static final String NONCE_VALUE_KEY = "com.eFoods.util.Authentication.SSO.nonce.value";
        private static final String NONCE_TIME_KEY = "com.eFoods.util.Authentication.SSO.nonce.time";
        private static final int NONCE_EXPIRY = 10; //minutes
        private static final int NONCE_LENGTH = 16;

        private String sharedKey;
        private String sessionKey;
        private String ssoEndpoint;
        private String ssoReciever;
        private List<String> adminUsers;

        public SSOAuthenticator(String ssoEndpoint, String ssoReciever,  String sharedKey, List<String> adminUsers) throws Exception {
                if (ssoEndpoint == null || ssoEndpoint.length() < 1)
                        throw new Exception("Endpoint must not be blank!");
                if (sharedKey == null || sharedKey.length() < 1)
                        throw new Exception("Shared key must not be blank!");
                if (adminUsers.size() < 1)
                        throw new Exception("No admin users in the system!");
                if (ssoReciever == null || ssoReciever.length() < 1)
                        throw new Exception("SSO Recieve URL must not be empty!");
                this.ssoEndpoint = ssoEndpoint;
                this.ssoReciever = ssoReciever;
                this.sharedKey = sharedKey;

                //Creates a unique sesion key for this SSO endpoint
                this.sessionKey = SESSION_KEY_PREFIX + "." + ssoEndpoint.hashCode();
                this.adminUsers = adminUsers;

        }

        @Override
        public boolean isAuthenticated(HttpServletRequest request) {
                UserBean user = getUser(request);
                Route currentRoute = (Route) request.getAttribute(Router.REQUEST_ROUTE_KEY);
                if (user != null && currentRoute != null) {
                        boolean needAdmin = currentRoute.isRequireAdmin();
                        boolean hasAdmin = user.isAdmin();
                        return needAdmin ? needAdmin == hasAdmin : true;
                }
                return false;
        }

        public UserBean getUser(HttpServletRequest request) {
                return ((UserBean) request.getSession().getAttribute(this.sessionK
ey));
        }

        private void setUser(HttpServletRequest request, UserBean user) {
                request.getSession().setAttribute(this.sessionKey, user);
        }

        /**
         *Generates a nonce, constructs a payload and redirects to the auth endpoi
nt
         */
        public void SSORedirect(HttpServletRequest request, HttpServletResponse re
sponse) throws Exception {
                String nonce = generateNonce(NONCE_LENGTH);
                String rawPayload = "nonce=" + nonce;
                String payload = new String(Base64.encodeBase64(rawPayload.getByte
s()));
                String signature = encode(sharedKey,payload).toLowerCase();

                String encodedPayload = URLEncoder.encode(payload, "UTF-8");
                String encodedSignature = URLEncoder.encode(signature, "UTF-8");
                String encodedReciever = URLEncoder.encode(ssoReciever, "UTF-8");
                String queryString = "payload=" + encodedPayload + "&signature=" +
 encodedSignature + "&redirect=" + encodedReciever;
                storeNonce(request, nonce);
                System.out.println("Auth: Nonce created.");
                response.sendRedirect(ssoEndpoint + "?" + queryString);
        }

        /**
         * Accepts a payload and a signature and verifies the authenticity of the
request by comparing the nonce.
         * Set’s the user in to the current session.
         */
        @Override
        public boolean login(HttpServletRequest request, String uname,
                        String pwd) throws Exception {
                Map<String,String[]> payload = new HashMap<String,String[]>();
                String localSignature, localNonce, remoteNonce, remotePayload, rem
oteSignature, rawPayload, userName, userFullName;

                remotePayload = request.getParameter("payload");
                remoteSignature = request.getParameter("signature");
                localSignature = encode(sharedKey, remotePayload);
                localNonce = retrieveNonce(request);

                //Only use the nonce once!
                expireNonce(request);

                rawPayload = new String(Base64.decodeBase64(remotePayload));
                RequestUtil.parseParameters(payload, rawPayload, "UTF-8");


                System.out.println("Auth : remote Payload: " + remotePayload);
                System.out.println("Auth : raw Payload: " + rawPayload);


                try {
                        remoteNonce = payload.get("nonce")[0];
                        userName = payload.get("username")[0];
                        userFullName = payload.get("fullname")[0];
                } catch (Exception e) {
                        throw new Exception("Invalid payload.");
                }
```

```java


                //Initial validation
                if (!localSignature.toLowerCase().equals(remoteSignature.toLowerCase()))
                        throw new Exception("Payload verification failed.");
                if (localNonce == null)
                        throw new Exception("Request expired. Please re-initiate login.");
                if  (!localNonce.equals(remoteNonce))
                        throw new Exception("Invalid nonce. Please re-initiaite login.");
                //End initial validation

                //Data validation
                if (userName == null || userName.length() < 1)
                        throw new Exception("Invalid username data in payload");
                if (userFullName == null || userFullName.length() < 1)
                        throw new Exception("Invalid user full name data in payload");

                /*
                 * FOLLOWING CODE IS COMMENTED OUT FOR THE SAKE OF MARKER. AS THERE IS NO WAY
                 * TO ADD HIM/HER AS AN ADMIN WITHOUT PRIOR KNOWLEDGE OF THEIR USERNAMES
                 */
                //boolean isAdmin = false;
                //if (adminUsers.contains(userName))
                //      isAdmin = true;
                boolean isAdmin = true;
                /*
                 * End testing code
                 */

                setUser(request, new UserBean(userName, userFullName, isAdmin));
                return isAuthenticated(request);
        }

        @Override
        public void logout(HttpServletRequest request) throws Exception {
                HttpSession session = request.getSession(false);
                //Clears the session before invalidating
                if (session != null) {
                        Enumeration<String> e = session.getAttributeNames();
                        while (e.hasMoreElements()) {
                                String key = (String) e.nextElement();
                                session.setAttribute(key, null);
                        }
                        session.invalidate();
                }
                if (isAuthenticated(request))
                        throw new Exception("Could not log out successfully!");
        }

        protected static void storeNonce(HttpServletRequest request, String nonce) {
                request.getSession().setAttribute(NONCE_VALUE_KEY, nonce);
                request.getSession().setAttribute(NONCE_TIME_KEY, System.currentTimeMillis());
        }

        protected static String retrieveNonce(HttpServletRequest request) {
                Long timeAtStore;
                try {
                        timeAtStore = (Long) request.getSession().getAttribute(NONCE_TIME_KEY);
                } catch (NumberFormatException e) {
                        timeAtStore = null;
                }
        if (timeAtStore != null && timeAtStore > (System.currentTimeMillis() - (NONCE_EXPIRY * 1000 * 60 * 10))) {
            return (String) request.getSession().getAttribute(NONCE_VALUE_KEY);
        } else {
            //Expire the nonce if the maximum time has been passed.
            expireNonce(request);
            return null;
        }


    }

    protected static void expireNonce(HttpServletRequest request) {
        request.getSession().setAttribute(NONCE_VALUE_KEY, null);
        request.getSession().setAttribute(NONCE_TIME_KEY, null);
        System.out.println("Auth: Nonce destroyed");
    }
    protected static String encode(String secret, String data) throws Exception {
        String hashFunction = "HmacSHA256";
        Mac instance = Mac.getInstance(hashFunction);
        SecretKeySpec keySpec = new SecretKeySpec(secret.getBytes(),
                hashFunction);

        instance.init(keySpec);
        return DatatypeConverter.printHexBinary((instance.doFinal(data.getBytes()))).toLowerCase();
    }

    protected static String generateNonce(int length) throws NoSuchAlgorithmException {
        byte[] nonce = new byte[NONCE_LENGTH];
        Random rand = SecureRandom.getInstance ("SHA1PRNG");
        rand.nextBytes(nonce);
        return DatatypeConverter.printHexBinary(nonce);
    }



}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false"/>
        <jsp:output omit-xml-declaration="true" />


        <div class="col-md-6">
                <div class="contact">
                        <p> Contact Information </p>
                        <p> 4700 Keele St, Toronto, ON M3J 1P3 </p>
                        <p> (416) 736-2100 </p>
                </div>

        </div>

        <div class="col-md-6">
                <iframe width="600" height="450" frameborder="0" style="border:0"
                src="https://www.google.com/maps/embed/v1/place?q=York%20University%2C%20Keele%20Street%2C%20Toronto%2C%20ON%2C%20Canada&amp;key=AIzaSyDqdeEV19s2GM_hCoj1eu-pKy37wB5_U9I">;</iframe>
        </div>

</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

        <!-- Cart / Log In -->
        <div class="col-md-12 header">
                <!--  Title -->
                <div class="pull-left">
                        <div class="logo">
                                <h1><a href="/eFoods/">eFoods</a></h1>
                        </div>
                </div>
                <div class="pull-right">
                        <div class="cart">
                                <div class="btn-group">
                                        <button type="button" class="btn btn-default btn-lg" data-ajaxify="backend/user" id="user-info">

                                        </button>

                                        <button type="button" class="btn btn-default btn-lg" data-ajaxify="backend/cart/badge" id="cart-info">

                                        </button>
                                </div>
                        </div>
                </div>

        </div>
</jsp:root>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" 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>
<meta name="viewport" content="width=device-width, initial-scale=1"></meta>
<meta charset="utf-8"></meta>
<title>Foods R Us</title>

<!-- Start CSS -->
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css"
        type="text/css" />
        <link href="//maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css" rel="stylesheet"
        type="text/css"/>
        <link href="//cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.1.6/css/material-wfont.css" rel="stylesheet"
        type="text/css"/>
        <link href="//cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.1.6/css/ripples.min.css" rel="stylesheet"
        type="text/css"/>
        <link href="//cdnjs.cloudflare.com/ajax/libs/bootstrap-material-design/0.1.6/fonts/Material-Design-Icons.woff" rel="stylesheet"
        type="text/css"/>
        <link href='http://fonts.googleapis.com/css?family=Lato:300,400' rel='stylesheet'
        type='text/css'/>
        <link rel="stylesheet"
        href="${pageContext.request.contextPath}/res/css/spinkit.css"
        type="text/css"/>
        <link rel="stylesheet"
        href="${pageContext.request.contextPath}/res/css/main.css"
        type="text/css" title="cse4413" media="screen, print" />

<!-- Start JS -->
</head>
<body>
        <!-- alert container -->
        <div id="alert-container"><jsp:text /></div>
        <!-- wrapper over content -->
        <div class="container">

                <!-- header -->
                <div class="row">
                        <jsp:include page="Head.jspx" />
                </div>

                <div class="jumbotron">
                        <jsp:include page="/partials/_search-bar.jspx">
                        </jsp:include>
                </div>


                <div class="row" data-ajaxify="backend/category/" id="main-content">
                        <p class="lead text-center"> Please wait loading... </p>
                </div>


                <!-- footer -->
                <div class="row footer">
```

```
                        <jsp:include page="Footer.jspx" />
                </div>
        </div>

        <script type="text/javascript"
                src="${pageContext.request.contextPath}/res/js/main.js"><jsp:text /></script>
        <script type="html/partial" id="alert-partial">
<div class="efoods-alert alert-hide" onclick="eFoods.util.hideAlert(this)">
        <div class="panel panel-danger">
        <div class="panel-heading">
                <h3 class="panel-title">Warning</h3>
        </div>
        <div class="panel-body">
                ###MSG###
            </div>
        </div>
</div>
                </script>
</body>
        </html>
</jsp:root>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Context privileged="true" reloadable="true">
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
  <Manager pathname=""/>
  <Resource name="jdbc/EECS" factory="org.apache.tomcat.jdbc.pool.DataSourceFactor
y"
            type="javax.sql.DataSource"
            driverClassName="org.apache.derby.jdbc.ClientDriver"
            url="jdbc:derby://roumani.eecs.yorku.ca:64413/CSE"
            username="student" password="secret"/>
  <ResourceLink global="jdbc/EECS" name="jdbc/EECS" type="javax.sql.DataSource"/>
</Context>
```

```
<?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:my="urn:jsptld:/WEB-INF/taglib.tld"
        version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />


        <h1>Foods R Us! Data Analytics</h1>
        <br/>
        <table id="tb" border="0">
                <tr>
                        <td><strong>Average time for customers to add an item to cart:</strong></td>
                        <td><my:MinSec milliseconds="${avgAddItemTime}"/></td>
                </tr>
                <tr>
                        <td><strong>Average time between a fresh visit to check out:</strong></td>
                        <td><my:MinSec milliseconds="${avgCheckOutTime}"/></td>
                </tr>
        </table>

</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />
<a href="backend/category/" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">Back to all categories</a>
        <table class="table table-striped table-hover">
                <thead>
                        <tr>
                                <th>Item</th>
                                <th>Price</th>
                                <th>Quantity</th>
                                <th>Total</th>
                        </tr>
                </thead>
                <tbody>
                        <c:forEach items="${cartItems }" var="r">
                        <tr>
                                <td>
                                        <a href="backend/item/${r.item.number }" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">${r.item.name }</a>
                                </td>
                                <td>${r.item.price }</td>
                                <td>
                                <form method="POST" action="backend/cart"
                                        data-refresh-ids="cart-info, main-content"
                                        onsubmit="return eFoods.app.handleForm(this);">
                                        <input type="number" name="qty" value="${r.quantity }" /> <input type="hidden"
                                                name="item" value="${r.item.number }" /> <input type="submit"
                                                class="" name="itemAdd" value="Change" />
                                </form>
                                </td>
                                <td>${r.getItemTotal() }</td>
                        </tr>
                        </c:forEach>
                </tbody>
        </table>

        <div class="checkout-pricing">
                <p>Total: ${total.toPlainString() }</p>
                <p>Shipping: ${shipping.toPlainString() }</p>
                <p>Tax: ${tax.toPlainString() }</p>
                <p>SubTotal: ${total.add(shipping).add(tax).toPlainString() }</p>
        </div>


        <c:choose>
    <c:when test="${not empty user}">
        <form method="POST" action="backend/cart/checkout" >
                        <input type="submit" class="btn btn-primary btn-lg" name="cartSubmit" value="Checkout" />
                </form>
    </c:when>
    <c:otherwise>
        <a class="btn btn-primary btn-lg" href="backend/login" onClick="return eFoods.util.statefulHref(this);">Checkout</a>
    </c:otherwise>
        </c:choose>
</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

        <!-- TODO: ajax item counter -->
        <a href="backend/cart/" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">
                <span class="badge"> ${cartItemCount } </span>
                <span class="glyphicon glyphicon-shopping-cart"></span>
        </a>
</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

<a href="backend/category/" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">Back to all categories</a>

<c:if test="${not empty alertInfo }">
        <div class="alert alert-dismissable alert-info">
            <strong>Heads up! </strong> ${alertInfo }
        </div>
</c:if>

<c:if test="${not empty alertWarn }">
        <div class="alert alert-dismissable alert-warn">
            <strong>Warning! </strong> ${alertWarn }
        </div>
</c:if>



<h1> Your purchase history. </h1>
<h3> New Orders</h3>
        <table class="table table-striped table-hover">
                <thead>
                        <tr>
                                <th>Order Id</th>
                                <th>Date Placed</th>
                                <th>View</th>
                        </tr>
                </thead>
                <tbody>
                        <c:forEach items="${newOrders }" var="r">
                        <tr>
                                <td>
                                        ${r.orderId }
                                </td>
                                <td>${r.modifiedDate }</td>
                                <td>
                                <a href="PurchaseOrders/${r.getFileNameOnly() }">Open.</a>
                                </td>
                        </tr>
                        </c:forEach>
                </tbody>
        </table>

<h3> Pending </h3>
        <table class="table table-striped table-hover">
                <thead>
                        <tr>
                                <th>Order Id</th>
                                <th>Date Placed</th>
                                <th>View</th>
                        </tr>
                </thead>
                <tbody>
                        <c:forEach items="${pendingOrders }" var="r">
                        <tr>
                                <td>
                                        ${r.orderId }
                                </td>
                                <td>${r.modifiedDate }</td>
                                <td>
                                <a href="PurchaseOrders/${r.getFileNameOnly() }">Open.</a>
                                </td>
                        </tr>
                        </c:forEach>
                </tbody>
        </table>


<h3> Purchased </h3>
        <table class="table table-striped table-hover">
                <thead>
                        <tr>
                                <th>Order Id</th>
                                <th>Date Placed</th>
                                <th>View</th>
                        </tr>
                </thead>
                <tbody>
                        <c:forEach items="${purchasedOrders }" var="r">
                        <tr>
                                <td>
                                        ${r.orderId }
                                </td>
                                <td>${r.modifiedDate }</td>
                                <td>
                                <a href="PurchaseOrders/${r.getFileNameOnly() }">Open.</a>
                                </td>
                        </tr>
                        </c:forEach>
                </tbody>
        </table>

</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

<h1>Congratulations!</h1>
<p>Your order was successfully placed. </p>
<p>You can check the status of all your orders by visiting your <a href="backend/cart/history" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">purchase history</a> page</p>
</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

        <!--  TODO: add jspx looping over contents -->
        <!-- main content -->
        <!-- need to add a loop here to go through the content -->
        <!-- fields are
        picture
        name
        price
        item id
        -->
    <div class="col-md-12">
        <h1> All Categories </h1>
    </div>
        <c:forEach items="${results }" var="r">
                <div class="col-md-4 col-lg-3 col-sm-6">
                        <div class="thumbnail well lg-well">
                                <a href="backend/category/${r.id }" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);"> <img
                                        src="${pageContext.request.contextPath}/pics/ids/${r.id }-300x200.jpg"
                                        class="img-responsive"></img>
                                </a>
                                <div class="caption">
                                        <p>${r.name }</p>
                                        <p>${r.description }</p>
                                        <p class="hidden">${r.id }</p>
                                </div>
                        </div>
                </div>
        </c:forEach>

</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />


        <div class="thumbnail item  well well-lg">
                <div class="caption">
                        <p class="item-name">${item.name }</p>
                        <p>\$${item.price } CDN</p>

                                                                <form method="POST" action="backend/cart"
                                data-refresh-ids="cart-info, item-${item.number }"
                                onsubmit="return eFoods.app.handleForm(this);" class="clearfix form-control-wrapper">
                                <c:choose>
                                        <c:when test="${not empty cartItem }">
                                <span class="pull-left">
                                        <label for="number">Qty:</label>
                                        <input class="item-quantity form-control" type="number" name="qty" value="${cartItem.quantity }" /> <input type="hidden"
                                        name="item" value="${item.number }" />
                                </span>
                                <span class="pull-right">
                                        <input type="submit"
                                        class="btn btn-primary" name="itemAdd" value="Change" />
                                </span>
                                <span class="pull-left"><small><b>Note:</b> This item is in your cart.</small></span>
                                        </c:when>
                                        <c:otherwise>
                                <span class="pull-left">
                                                <label for="number">Qty:</label>
                                                <input class="item-quantity form-control"  type="number" name="qty" value="1" />
                                                <input type="hidden" name="item" value="${item.number }" />
                                </span>
                                <span class="pull-right">
                                                <input type="submit" class="btn btn-primary" name="itemAdd" value="Add" />
                                </span>
                                        </c:otherwise>
                                </c:choose>
                        </form>
                </div>
                <hr/>
                        <div class="caption clearfix">
                                <p class="h5" style="padding-bottom: 10px;"><b>Suggested Purchase</b></p>
                                <a href="backend/item/${crossSellItem.number }" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">
                                        <img src="http://placehold.it/64x64" class="pull-left img-responsive"></img>
                                        <span class="pull-right" style="max-width: 130px;">
                                        <p>${crossSellItem.name }</p>
                                        <p>\$${crossSellItem.price } CDN</p>
                                        </span>
                                </a>
                        </div>
        </div>


</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

        <h1>${category.name }</h1>
        <a href="backend/item/filter/" data-refresh-ids="main-content"
                onClick="return eFoods.app.handleHref(this, true);">Back to all
                categories</a>
        <c:forEach items="${results }" var="r">
                <div class="col-md-4" data-ajaxify="backend/item/${r.number }" id="item-${r.number }">
                        <c:set var="item" scope="request" value="${r }" />
                        <c:set var="cartItem" scope="request" value="${cartItems[r.name] }" />
                        <jsp:include page="_item.jspx" />
                </div>
        </c:forEach>

</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />


        <div class="thumbnail item well well-lg">
                <img src="http://placehold.it/195x195" class="img-responsive"></img>
                <div class="caption">
                        <p class="item-name">${item.name }</p>
                        <p>\$${item.price } CDN</p>

                                                <form method="POST" action="backend/cart"
                                data-refresh-ids="cart-info, item-${item.number }"
                                onsubmit="return eFoods.app.handleForm(this);" class="clearfix form-control-wrapper">
                                <c:choose>
                                        <c:when test="${not empty cartItem }">
                                <span class="pull-left">
                                        <label for="number">Qty:</label>
                                        <input class="item-quantity form-control" type="number" name="qty" value="${cartItem.quantity }" /> <input type="hidden"
                                        name="item" value="${item.number }" />
                                </span>
                                <span class="pull-right">
                                        <input type="submit"
                                        class="btn btn-primary" name="itemAdd" value="Change" />
                                </span>
                                <span class="pull-left"><small><b>Note:</b> This item is in your cart.</small></span>
                                        </c:when>
                                        <c:otherwise>
                                <span class="pull-left">
                                                <label for="number">Qty:</label>
                                                <input class="item-quantity form-control"  type="number" name="qty" value="1" />
                                                <input type="hidden" name="item" value="${item.number }" />
                                </span>
                                <span class="pull-right">
                                                <input type="submit" class="btn btn-primary" name="itemAdd" value="Add" />
                                </span>
                                        </c:otherwise>
                                </c:choose>
                        </form>
                </div>
        </div>


</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

        <div class="col-md-12">
                <p class="h4"><a href="backend/category" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">All Categories</a></p>
                <h1><a href="backend/category/${category.id }" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">${category.name }</a></h1>
                <div class="icon-previous">
                        <a href="javascript:window.history.back()"><span class="mdi-navigation-arrow-back"/></a>
                </div>
        </div>

        <c:forEach items="${results }" var="r">
                <div class="col-md-4 col-sm-6 col-lg-3" data-ajaxify="backend/item/${r.number }/partial" id="item-${r.number }">
                        <c:set var="item" scope="request" value="${r }" />
                        <c:set var="cartItem" scope="request" value="${cartItems[r.name] }" />
                        <jsp:include page="_item.jspx" />
                </div>
        </c:forEach>

</jsp:root>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" 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>
</head>
<body>

        <div class="text-center row">
                <h2 class="home-heading"> The best way to shop and order Food.</h2>
                <p> Choose foods ranging from meats to ice cream.</p>
                <div class="home-search-bar col-md-12">
                        <form method="get" action="backend/search" id="search-form"
                        data-refresh-ids="main-content" onSubmit="return false;">
                                <input name="filter" class="search-input"  id="item-search" type="text"
                                value="" placeholder="Search from a wide assortment of foods."/>
                                <input type="hidden" name="page" value="1"/>
                                <input type="hidden" name="limit" value="25"/>
                        </form>
                </div>
        </div>

</body>
</html>
</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />
<a href="javascript:window.history.back()">Go back.</a>

<h1>Looks like we got a problem...</h1>
<br/>
<c:if test="${not empty errorMessage }">
        <h2>${errorMessage}</h2>
</c:if>
<c:if test="${empty errorMessage }">
        <h2>We screwed up. Our bad. Please try again!</h2>
</c:if>

<br/>
<h3>Here’s a picture of a kitten to brighten your day.</h3>
<br/>
<img src="http://placekitten.com/g/200/300"></img>
</jsp:root>
```

```
<?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" version="2.0">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output omit-xml-declaration="true" />

        <c:choose>
                <c:when test="${not empty currentUser }">
                        <p class="login-text">Hello, ${currentUser.fullName }!  |</p>
                        <c:if test="${currentUser.admin }">
                                <p class="login-text">(<a href="backend/analytics" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">Analytics</a>)  |</p>
                        </c:if>
                        <p class="login-text"><a href="backend/cart/history" data-refresh-ids="main-content" onClick="return eFoods.app.handleHref(this, true);">Purchase History</a>  |</p>
                        <p class="login-text"><a href="backend/logout" data-refresh-ids="user-info, cart-info, main-content" onClick="return eFoods.app.handleHref(this);">Log Out</a></p>
                </c:when>
                <c:otherwise>
                        <p class="login-text"><a href="backend/login" onClick="return eFoods.util.statefulHref(this);">Log In</a></p>
                </c:otherwise>
        </c:choose>

</jsp:root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
                <html>
                        <head>
                                <title>Purchase Order Report</title>
                        </head>
                        <body>
                                <h1> Order Information </h1>
                                <xsl:apply-templates/>
                </body>
        </html>
    </xsl:template>

    <xsl:template match="order">
        <div>
                <p>ID: <xsl:value-of select="@id"/></p>
                <p>Time Submitted: <xsl:value-of select="@submitted"/></p>
        </div>

        <xsl:apply-templates select="customer"/>
        <table border="1" cellpadding="5">
                <tr>
                        <th>Item Number</th>
                        <th>Item Name</th>
                        <th>Price</th>
                        <th>Quantity</th>
                        <th>Extended</th>
                </tr>
                <xsl:for-each select="./items/item">
                        <tr>
                                <td><xsl:value-of select="@number"/></td>
                                        <td><xsl:value-of select="./name"/></td>
                                        <td><xsl:value-of select="./price"/></td>
                                        <td><xsl:value-of select="./quantity"/></td>
                                        <td><xsl:value-of select="./extended"/></td>
                                </tr>
                        </xsl:for-each>
                </table>

        <h3> Costs </h3>
        <p>Total: <xsl:value-of select="./total"/></p>
        <p>Shipping: <xsl:value-of select="./shipping"/></p>
        <p>HST: <xsl:value-of select="./HST"/></p>
        <p>Grand Total: <xsl:value-of select="./grandTotal"/></p>

    </xsl:template>

    <xsl:template match="customer">
        <p>Customer Name: <xsl:value-of select="./name"/></p>
    </xsl:template>

</xsl:stylesheet>
```

```
/*
 * Main CSS rules
 */

html, body {
    margin-top: 20px;
    font-family: 'Lato';
    font-size: 14px;
}

.header {
    margin-bottom: 100px;
}

.login-text {
    display: inline;
    font-size: 1.0em;
}

.footer {
        margin-top: 100px;
}

.logo {

}

a {
        text-decoration: none;
        color: inherit;
}

a:hover {
        text-decoration: none;
}

/* Level this with the logo */
.cart {
        padding-top: 20px;
}

.home-landing {
}

.home-heading {
        fomt-family: Lato;
        font-weight: 300;
}

.home-search-bar {
        padding-left: 20px;
        margin-top: 50px;
        margin-bottom: 50px;
}

.search-input {
        height: 2em;
        font-size: 2em;
        width: 100%;
        padding-left: 60px;
}

.contact > p {
        font-size: 1.45em;
}

.item-quantity {
        width: 30px;
        margin: 15px 5px;
        display: inline-block;
        text-align: center;
}

.contact {
        background: white;
        padding: 10px;
        height: 450px /* Make sure this matches the map */
}

.item {
        border-radius: 5px;
        margin: 10px;
        padding: 10px !important;
        height: 335px;
}

.icon-previous {
        font-size: 60px;
        display: inline-block;
        background: white;
        border-radius: 20px;
}

.checkout-pricing {
        font-size: 1.5em;
}

.item p {
        margin:0;
        padding:0;
}

.item .item-name {
        text-overflow: ellipsis;
        white-space: nowrap;
        overflow: hidden;
}

.efoods-alert
{
    position: fixed;
        margin-right: auto;
        margin-left: auto;
        margin-top: 20px;
        width: 500px;
        left: 0;
        right: 0;
        z-index: 9999;
}
.efoods-alert:hover {
        cursor: pointer;
}
.efoods-alert.alert-show {
         animation: fadein 0.3s;
    -moz-animation: fadein 0.3s; /* Firefox */
    -webkit-animation: fadein 0.3s; /* Safari and Chrome */
    -o-animation: fadein 0.3s; /* Opera */
}

.efoods-alert.alert-hide {
         animation: fadeout 0.3s;
    -moz-animation: fadeout 0.3s; /* Firefox */
    -webkit-animation: fadeout 0.3s; /* Safari and Chrome */
    -o-animation: fadeout 0.3s; /* Opera */
}

@keyframes fadein {
    from {
        opacity:0;
    }
    to {
        opacity:1;
    }
}
@-moz-keyframes fadein { /* Firefox */
    from {
        opacity:0;
    }
    to {
        opacity:1;
    }
}
@-webkit-keyframes fadein { /* Safari and Chrome */
    from {
        opacity:0;
    }
    to {
        opacity:1;
    }
}
@-o-keyframes fadein { /* Opera */
    from {
        opacity:0;
    }
    to {
        opacity: 1;
    }
}

@keyframes fadeout {
    from {
        opacity:1;
    }
    to {
         opacity:0;
    }
}
@-moz-keyframes fadeout { /* Firefox */
    from {
        opacity:1;
    }
    to {
         opacity:0;
    }
}
@-webkit-keyframes fadeout { /* Safari and Chrome */
    from {
        opacity:1;
    }
    to {
        opacity:0;
    }
}
@-o-keyframes fadeout { /* Opera */
    from {
        opacity:1;
    }
    to {
                opacity:0;
    }
}
```

```
/**
 *
 */

//For archaic browsers that doesn’t support console logging (*cough* IE.. *cough)
if (typeof console == "undefined") {
    this.console = {log: function() {}};
}


//Define the app scope
var eFoods = {};
//Utility functions
eFoods.util = {};
//Standard errors -- todo:
eFoods.util.errors = {};
//Core app code
eFoods.app = {};
//App vars
eFoods.vars = {
        MAIN_URL : "backend/category",
        CONTENT_DIV : "main-content"
};

eFoods.util.removeClass = function(element, className) {
   element.className = element.className.replace(new RegExp(className, ’g’), ’’ );
}
eFoods.util.addClass = function(element, className) {
        element.className += " " + className;
}
eFoods.util.showAlert = function(msg) {
        var partial = document.getElementById("alert-partial");
        var partialHTML = partial.innerHTML.replace(’###MSG###’, msg);
        var alertContainer = document.getElementById("alert-container");
        alertContainer.innerHTML = partialHTML;
        var alerts = alertContainer.querySelectorAll(".efoods-alert");
        for(var i=0;i < alerts.length; i++) {
                eFoods.util.removeClass(alerts[i],"alert-hide")
                eFoods.util.addClass(alerts[i],"alert-show");
                setTimeout((function(element) {
                        return function() {
                                eFoods.util.hideAlert(element);
                        }
                })(alerts[i]), 5000);
        }
}

eFoods.util.hideAlert = function(element) {
        console.log(element);
        var alertContainer = document.getElementById("alert-container");
        eFoods.util.removeClass(element,"alert-show")
        eFoods.util.addClass(element,"alert-hide");
        setTimeout((function(el) {
                return function() {
                        alertContainer.removeChild(el);
                }
        })(element), 300);
}

// Returns the baseURL, ex. http://localhost:4413
eFoods.util.baseURL = function() {
        path = window.location.href.split(’/’);
        protocol = path[0];
        host = path[2];
        return protocol + "//" + host;
}

//Ajax helper. Takes url and 2 callbacks.
eFoods.util.doAjax = function(url, data, method, onSuccess, onFailure) {

        //Default callbacks when not supplied
        if (!onSuccess) {
                onSuccess = function(request) {
                        console.log("Request successful");
                        console.log(request);
                }
        }
        if (!onFailure) {
                onFailure = function(request) {
                        console.log("Request failed");
                        console.log(request);
                        try {
                                var msg = JSON.parse(request.responseText);
                                eFoods.util.showAlert(msg.error);
                        }  catch (e) {
                                console.log(request.responseText)
                        }
                }
        }
        //End default callbacks

        //Handler for xmlHttpRequest
        var handler = function(request) {
                if ((request.readyState == 4)) {
                        if (request.status == 200) {
                                onSuccess(request);
                        } else {
                                onFailure(request)
                        }

                }
        }

        var request = new XMLHttpRequest();
        var postData = null;
        var hasQS = url.split("?").length > 1;
        url = url + (hasQS ? "&" : "?") + "ajax=true";

        if (method.trim().toUpperCase() == "GET") { //If GET we append data with a '?' mark
                url += data.length > 0 ? "&" + data : ""
                request.open(method, url, true);
        } else if (method.trim().toUpperCase() == "POST") { //If POST we set the content type and assign data as PostData
                postData = data;
                request.open(method, url, true);
                request.setRequestHeader('Content-type',
                                'application/x-www-form-urlencoded');
        } else {
                console.log(this, method);
                throw "dont know about that http method";
        }

        request.onreadystatechange = function() {
                handler(request);
        };
        request.send(postData);
};

//Convert values in a form in to a query string for ajax submission
eFoods.util.formToQueryString = function(form) {
        var elem = form.elements;
        var params = "";
        for (var i = 0; i < elem.length; i++) {
                if (elem[i].tagName == "SELECT") {
                        value = elem[i].options[elem[i].selectedIndex].value;
                } else {
                        value = elem[i].value;
                }
                params += elem[i].name + "=" + encodeURIComponent(value) + "&";
        }
        return params;
}

//Get current app state from hashbang
eFoods.util.getState = function() {
        var state = "";
        if (window.location.hash && window.location.hash.substring(0, 2) == "#!") {
                state = window.location.hash.substring(2);
        }
        return state.length < 1 ? null : state
}

//Set hashbang state (or default state when empty) to app state
eFoods.util.setState = function() {
        var state = this.getState();
        if (state) {
                url = state
        } else {
                url = eFoods.vars.MAIN_URL;
        }
        var ele = document.getElementById(eFoods.vars.CONTENT_DIV);
        ele.setAttribute(’data-ajaxify’, url)
}

//Ajaxify, looks for  the tag data-ajaxify and populates the inside via ajax usisng "GET"
eFoods.app.ajaxify = function(element) {
        var url = element.getAttribute("data-ajaxify");
        var isMain = false;
        // Check if we’re changing app state
        if (element.getAttribute("id") == eFoods.vars.CONTENT_DIV) {
                window.location.hash = "#!" + url;
                isMain = true;
        }

        eFoods.util.doAjax(url, (isMain ? "main=true" : ""), "GET", function(request) {
                console.log("Ajaxified.", url, element.id);
                element.innerHTML = request.responseText;
        });
};

//Ajaxify everything with the tag!
eFoods.app.ajaxifyAll = function() {
        var elements = document.querySelectorAll(’[data-ajaxify]’);
        for (i = 0; i < elements.length; i++) {
                (function(target) {
                        eFoods.app.ajaxify(target)
                })(elements[i]); //This a little bit of js sorcery called self-invoking closure scoping.
        }
}

//Invoke this to call a hyperlink. See the dom element for the required tags. This will also refresh given dom elements on success (used to refresh cart badge and user badge)
//ajaxifyTarget = on true, instead of refreshing the target, perform the ajax call and put the response in to the target dom element (used to change pages - category, item, car)
eFoods.app.handleHref = function(href, ajaxifyTarget) {
        try {
                var url = href.getAttribute("href");
                var refreshDoms = href.getAttribute("data-refresh-ids").split(",");
                if (ajaxifyTarget) {
                        for (i = 0; i < refreshDoms.length; i++) {
                                (function(target) {
                                        target.setAttribute("data-ajaxify", url);
                                        eFoods.app.ajaxify(target);
                                }(document.getElementById(refreshDoms[i].trim())));
                        }
                } else {
                        eFoods.util.doAjax(url, "", "GET", function(request) {
                                console.log("Request success!" + url);
                                for (i = 0; i < refreshDoms.length; i++) {
                                        (function(target) {
                                                eFoods.app.ajaxify(target);
                                        })(document.getElementById(refreshDoms[i].trim()))
                                }
                        });
                }

        } catch (e) {
                //throw e
                console.log(e);
        }
        return false;
};

//Ajax form submission
eFoods.app.handleForm = function(form, ajaxifyTarget) {
        try {
                var method = form.method;
                var url = form.getAttribute('action');
                var refreshDoms = form.getAttribute("data-refresh-ids").split(",");
                var formData = eFoods.util.formToQueryString(form);
                console.log("Submitting form", method, url, refreshDoms, formData, form);

                if (ajaxifyTarget) {
                        for (i = 0; i < refreshDoms.length; i++) {
                                (function(target) {
                                        console.log("Target url for form" , url + "?" + formData);
                                        target.setAttribute("data-ajaxify", url + "?" + formData);
                                        eFoods.app.ajaxify(target);
                                }(document.getElementById(refreshDoms[i].trim())));
                        }
                } else {
                        eFoods.util.doAjax(url, formData, method, function(request) {
                                console.log("Request success!" + url);
                                for (i = 0; i < refreshDoms.length; i++) {
                                        (function(target) {
                                                eFoods.app.ajaxify(target);
                                        })(document.getElementById(refreshDoms[i].trim()))
                                }
                        });
                }
        } catch (e) {
                console.log(e, e.stack);
        }
        return false;
};

//Watch for any changes in the hash bang, update app state if needed
eFoods.app.watchState = function() {
        var ele = document.getElementById(eFoods.vars.CONTENT_DIV);
        var interval = setInterval(function() {
        var appState = ele.getAttribute('data-ajaxify');
                if (appState != eFoods.util.getState()) {
                        eFoods.util.setState();
                        eFoods.app.ajaxify(ele);
                }
        }, 250)
}

eFoods.util.statefulHref = function(ele) {
        var state = eFoods.util.getState();
        var hasQS = ele.href.split("?").length > 1;
        var toAppend = (hasQS ? "&" : "?") +  (state ? "state=" + encodeURIComponent(state) : "");
        window.location.href = ele.href + toAppend;
        return false;
}

eFoods.util.addEvents = function(handlers, ev, fn) {
        console.log('Adding event for', ev, handlers);
        for (var i = 0; i < handlers.length; i++) {
                var h = handlers[i];
                h.addEventListener(ev, fn);
        }
}

//Initialization code. We'll call this at the bottom of the file.
eFoods.app.init = function() {
        console.log(eFoods.util.baseURL());

        eFoods.util.setState();
        eFoods.app.ajaxifyAll();
        eFoods.app.watchState();


};

document.onreadystatechange = function () {
    if (document.readyState == "interactive") {
        eFoods.app.init();
    }
}

/* The idea for this module is as the user is typing
 * the main page is updated with foods matching the search.
 *
 * We use a timeout to avoid sending requests for every keystroke.
 */
var searchBar = document.getElementById('item-search');
var searchForm = document.getElementById('search-form');

var searchTimeout;
searchBar.addEventListener('keyup', function() {
        // an ajax request will be only sent only after 400ms
        if (searchTimeout) {
                clearTimeout(searchTimeout);
        }
        searchTimeout = setTimeout(function() {
                // do ajax here
                eFoods.app.handleForm(searchForm, true);
        }, 400)
})
```

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/configure.dtd">
<Configure class="org.eclipse.jetty.webapp.WebAppContext">
        <New id="DS" class="org.eclipse.jetty.plus.jndi.Resource">
                <Arg></Arg>
                <Arg>java:comp/env/jdbc/EECS</Arg>
                <Arg>
                        <New class="org.apache.derby.jdbc.ClientConnectionPoolDataSource">
                                <Set name="DatabaseName">CSE</Set>
                                <Set name="ServerName">roumani.eecs.yorku.ca</Set>
                                <Set name="PortNumber">64413</Set>
                                <Set name="User">student</Set>
                                <Set name="Password">secret</Set>
                        </New>
                </Arg>
        </New>
</Configure>
```

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

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
        version="2.0">

    <description>custom tags for eFood</description>
    <tlib-version>1.0</tlib-version>
    <short-name>CustomTagLibraryEfood</short-name>

    <tag>
        <description>converts milliseconds to #m#s format</description>
        <name>MinSec</name>
        <tag-class>tags.MinSec</tag-class>
        <body-content>empty</body-content>
        <attribute>
                <name>milliseconds</name>
                <required>true</required>
                <rtexprvalue>true</rtexprvalue>
        </attribute>
    </tag>

</taglib>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <display-name>eFoods</display-name>
  <welcome-file-list>
    <welcome-file>Home.jspx</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>Category</servlet-name>
    <servlet-class>ctrl.Category</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Item</servlet-name>
    <servlet-class>ctrl.Item</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Auth</servlet-name>
    <servlet-class>ctrl.Auth</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Cart</servlet-name>
    <servlet-class>ctrl.Cart</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Misc</servlet-name>
    <servlet-class>ctrl.Misc</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Orders</servlet-name>
    <servlet-class>ctrl.Orders</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Analytics</servlet-name>
    <servlet-class>ctrl.Analytics</servlet-class>
  </servlet>
</web-app>
```