SYSTEMES D'INFORMATION 2

TP 06 – JAVA SERVER PAGES

Jean-Christophe ROLDAN & Samuel MERTENAT

Télécommunications
T3a & T3f

Fribourg, le 11 janvier 2016

TABLE DES MATIERES

SYSTEMES D'INFORMATION 2

TP 06 – JAVA SERVER PAGES

1 INTRODUCTION

Ce sixième travail pratique est l'occasion de se familiariser avec un serveur d'application Apache Tomcat et de développer une petite application utilisant des *Java Server Pages*.

2 MISE EN PLACE

Pour la réalisation de ce travail pratique, il est nécessaire de mettre en place un serveur Tomcat. D'autre part, nous utiliserons le serveur de bases de données installé lors du TP précédent (PHP).

Le développement et les tests sont réalisés sur une machine Macintosh et la configuration est la suivante :

Ordinateur :

- Macbook Pro Retina 15.4, Mac OS X EL Capitan, version 10.11.2

Logiciels :

Apache Tomcat, version 8.0.30, port 8080

3 SPECIFICATION

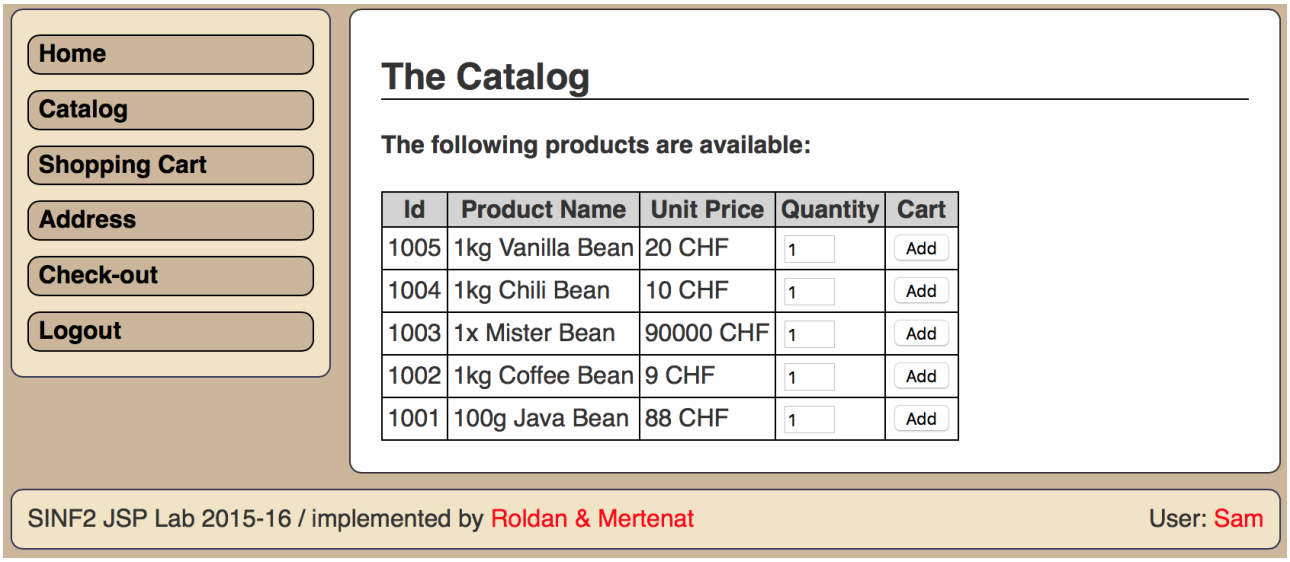
Ce chapitre a pour but de spécifier le plan de navigation du site web entre les différentes pages avec l'affichage des pages et les JavaBeans utilisés pour chacune d'entre-elles.

3.1 CATALOG

CatalogBean : scope = application (le catalogue est le même pour tous les clients)

ShoppingCartBean : scope = session (pour pouvoir ajouter le produit au caddie)

UserBean : scope = session (pour vérifier si l'utilisateur est authentifié)



Actions / liens :

- Add → Recharge la page
  - Paramètres : pId + quantity

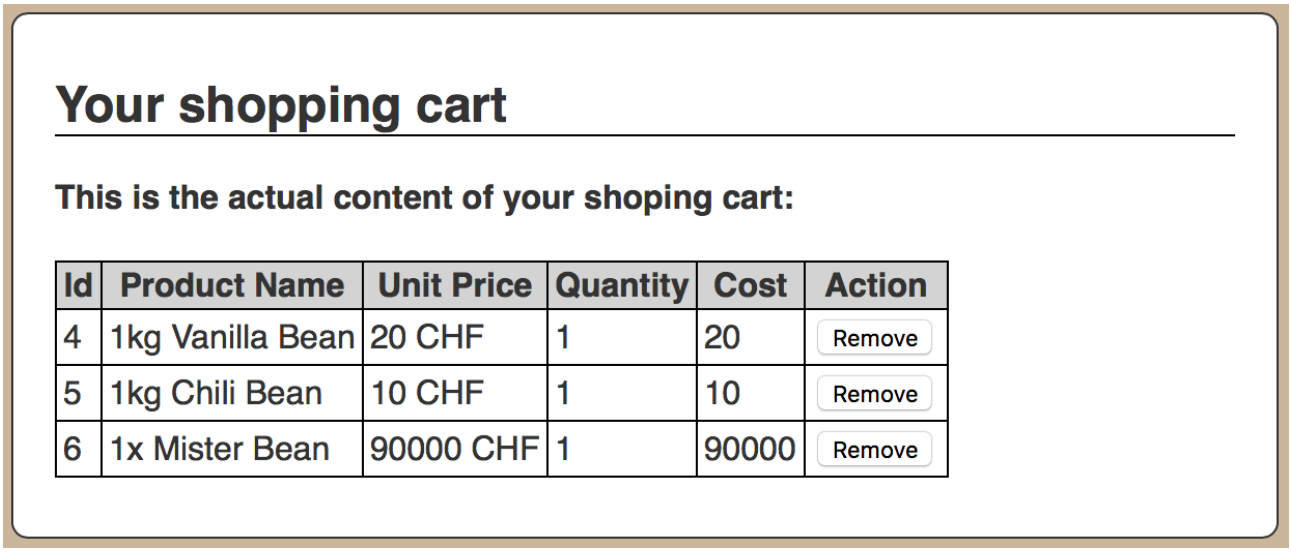
```
int id = Integer.parseInt(request.getParameter("pId"));
 int quantity = Integer.parseInt(request.getParameter("quantity"));
// Gets the product from the id and inserts it into the cart
CatalogItem itemToAdd = catalog.getCatalogItem(id);
cart.addToCart(itemToAdd, quantity);
```

3.2 SHOPPING CART

CatalogBean : scope = application (pour afficher les valeurs des différents champs qui correspondent au catalogue)

ShoppingCartBean : scope = session

UserBean : scope = session (pour vérifier si l'utilisateur est authentifié)



Your shopping cart

This is the actual content of your shoping cart:

Id	Product Name	Unit Price	Quantity	Cost	Action
4	1kg Vanilla Bean	20 CHF	1	20	Remove
5	1kg Chili Bean	10 CHF	1	10	Remove
6	1x Mister Bean	90000 CHF	1	90000	Remove

Actions / liens :

- Remove → Recharge la page
  - Paramètre : pId

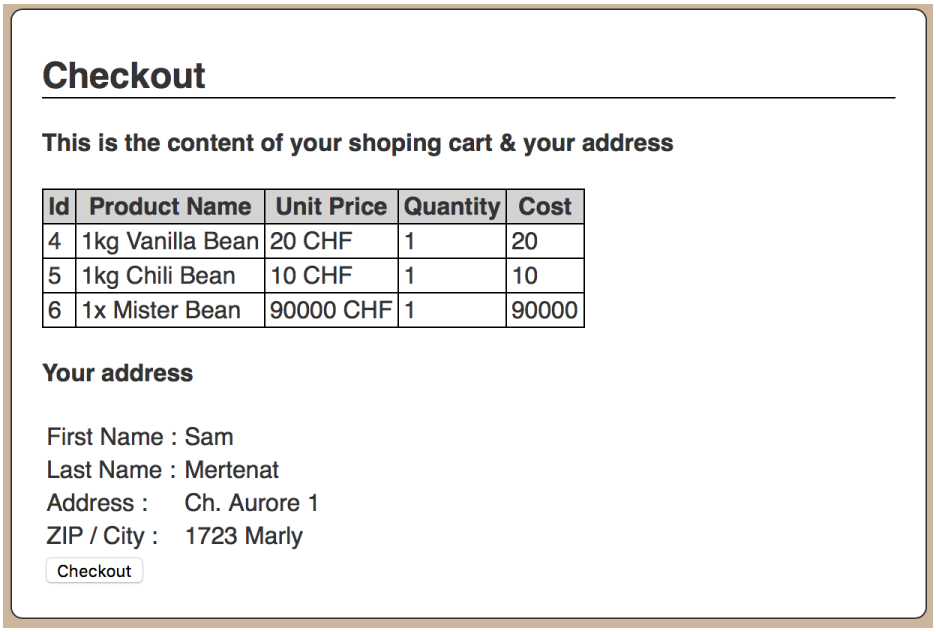
```
int id = Integer.parseInt(request.getParameter("pId"));
CatalogItem itemToRemove = catalog.getCatalogItem(id);
cart.removeFromCart(id); // removes the item from the cart
```

3.3 CHECKOUT

CatalogBean : scope = application

ShoppingCartBean : scope = session

UserBean : scope = session (pour vérifier si l'utilisateur est authentifié)



Checkout

This is the content of your shoping cart & your address

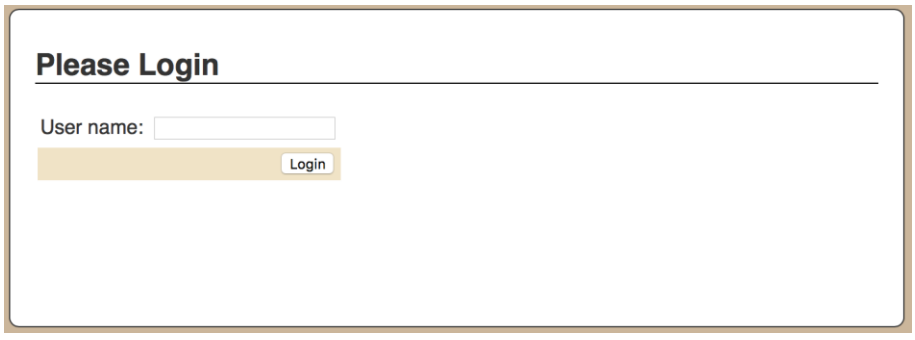
Id	Product Name	Unit Price	Quantity	Cost
4	1kg Vanilla Bean	20 CHF	1	20
5	1kg Chili Bean	10 CHF	1	10
6	1x Mister Bean	90000 CHF	1	90000

Your address

First Name : Sam
Last Name : Mertenat
Address : Ch. Aurore 1
ZIP / City : 1723 Marly
Checkout

3.4 LOGIN / LOGOUT

UserBean : scope = session (pour supprimer le Bean)



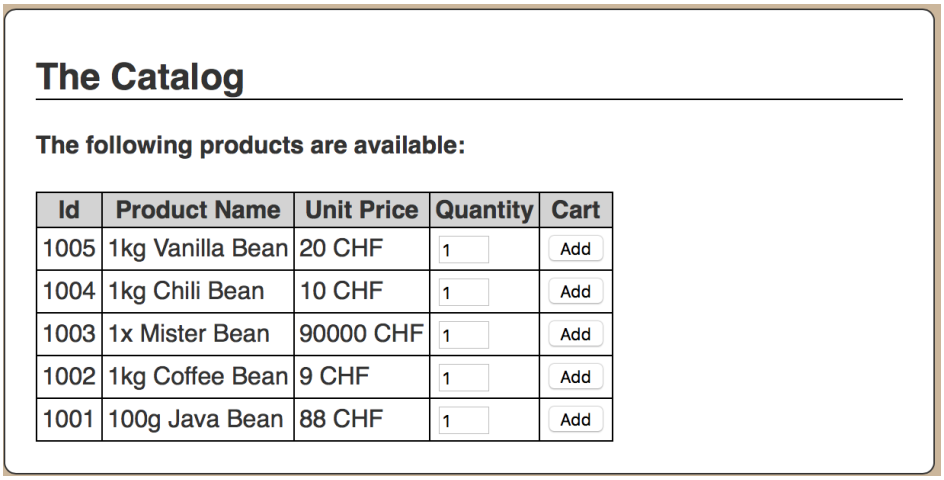
Please Login

User name:
Login

4 REALISATION

4.1 CATALOG

Une fois authentifié, il est possible depuis cette page, d'ajouter des éléments du catalogue au caddie en spécifiant la quantité souhaitée puis en appuyant sur le bouton « Add ».



The Catalog

The following products are available:

Id	Product Name	Unit Price	Quantity	Cart
1005	1kg Vanilla Bean	20 CHF	1	Add
1004	1kg Chili Bean	10 CHF	1	Add
1003	1x Mister Bean	90000 CHF	1	Add
1002	1kg Coffee Bean	9 CHF	1	Add
1001	100g Java Bean	88 CHF	1	Add

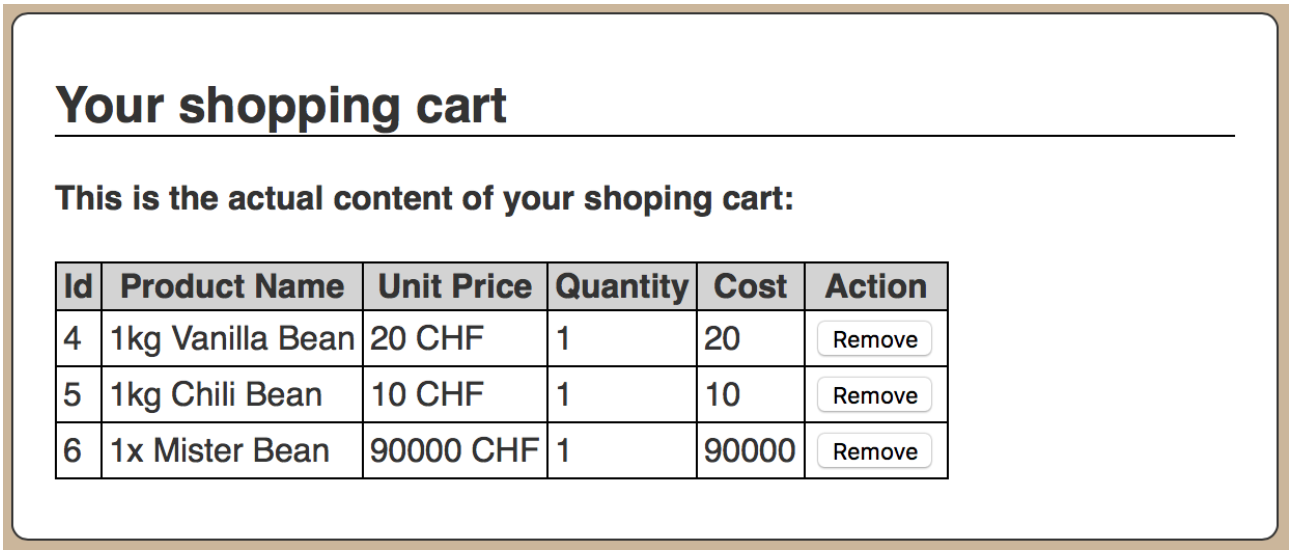
Le code ci-dessous nous permet d'afficher le contenu du catalogue grâce à une énumération de tous les produits spécifiés dans la classe « CatalogBean ».

```
Enumeration<CatalogItem> cat = catalog.getCatalogItems();
CatalogItem item = null;
while (cat.hasMoreElements()) {
  item = cat.nextElement();
  // loop repeating the following form (one form per line/product)
  %>
  <!-- we use a form to send the product ID to add to the cart -->
  <form action="catalog.jsp" method="post">
    <tr>
    <td><%= item.getId() %>
    <input name="pId" type="hidden" id="pId" value="<%= item.getId() %>" /></td>
    <td><%= item.getName() %></td>
    <td><%= item.getPrice() %> CHF</td>
    <td><input name="quantity" type="text" id="quantity" size="4" value="1"/></td>
    <td><input type="submit" name="submit" value="Add" /></td>
    </tr>
  </form>
<%
} // (end of loop)
%>
```

Chaque bouton « Add » appelle la même page en requête « POST » avec les paramètres « pId » et « quantity ». Si l'opération se déroule sans erreur, un message de succès s'affiche.

```
<%
try {
  // Retrieves the product ID (method POST)
  if (request.getParameter("pId") != null) {
    int id = Integer.parseInt(request.getParameter("pId"));
    int quantity = Integer.parseInt(request.getParameter("quantity"));
    // Gets the product from the id and inserts it into the cart
    CatalogItem itemToAdd = catalog.getCatalogItem(id);
    cart.addToCart(itemToAdd, quantity);
%>
<p><span style="color:red">Successfully added <%= quantity %>x <%= itemToAdd.getName()
%></span></p>
<%
    }
} catch (Exception e) {…}
%>
```

4.2 SHOPPING CART



Une fois ajoutés depuis le catalogue, les produits sont listés dans le caddie avec le même principe qu'avant : l'énumération des produits sauvegardés dans le « ShoppingCartBean ». Il est ici possible de supprimer l'un des produits en appuyant sur « Remove », bouton qui va appeler la même page pour effectuer l'opération. Grâce à un petit script JavaScript, nous rechargeons ensuite la page, afin d'avoir le tableau à jour.

```
// Javascript code to reload the page
<script>
  location.reload(true);
</script>
```

4.3 LOGIN / LOGOUT

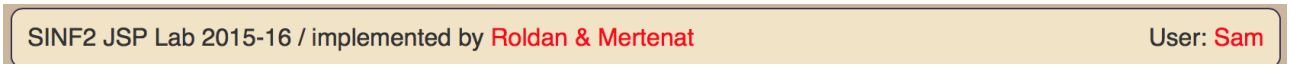
Pour le login, le formulaire appelle le script JSP « connection.jsp » qui s'occupe de créer un nouveau JavaBean pour la session et de définir le nom d'utilisateur grâce au « setter » implémenté dans la classe « UserBean ».

Comme il n'y a aucune vérification de mot de passe ou autre à faire, une simple redirection sur la page « catalog.jsp » est finalement faite.

```
<jsp:useBean id="user" class="shop.UserBean" scope="session" />
<jsp:setProperty name="user" property="*" />

<% user.setUserName(request.getParameter("username")); %>
<jsp:forward page="catalog.jsp" />
```

Une fois authentifié, le nom d'utilisateur change dans le « footer » et le menu « Login » se transforme en « Logout ».



Par ailleurs, afin de vérifier sur chaque page accessible après login que l'utilisateur s'est bien authentifié, les lignes suivantes ont été placées dans les fichiers « catalog.jsp », « cart.jsp », « address-form.jsp » et « checkout.jsp ».

Si le visiteur ne s'est pas authentifié, il est redirigé sur la page login.

```
<%
if (user.getUserName() == null) {
  response.sendRedirect("http://localhost:8080/jsp-lab/login-form.jsp");
}
%>
```

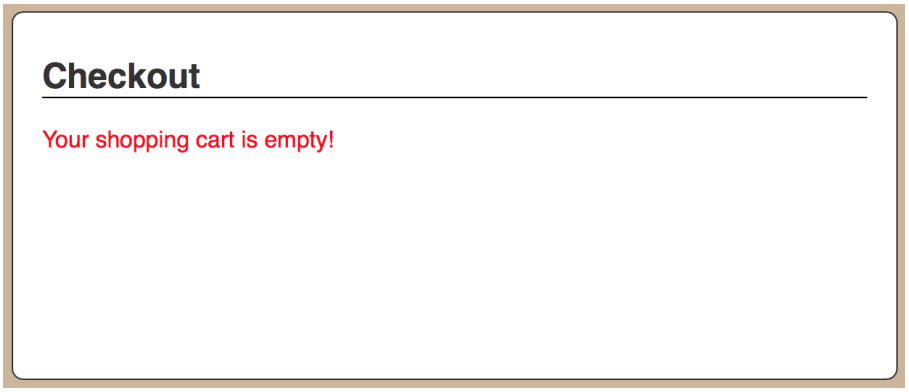
Quant à l'opération de logout, elle consiste à supprimer le JavaBean créé précédemment pour la session de l'utilisateur et à rediriger le visiteur vers la page d'accueil.

```
<%<% session.invalidate(); %>
<jsp:forward page="index.jsp" />
```

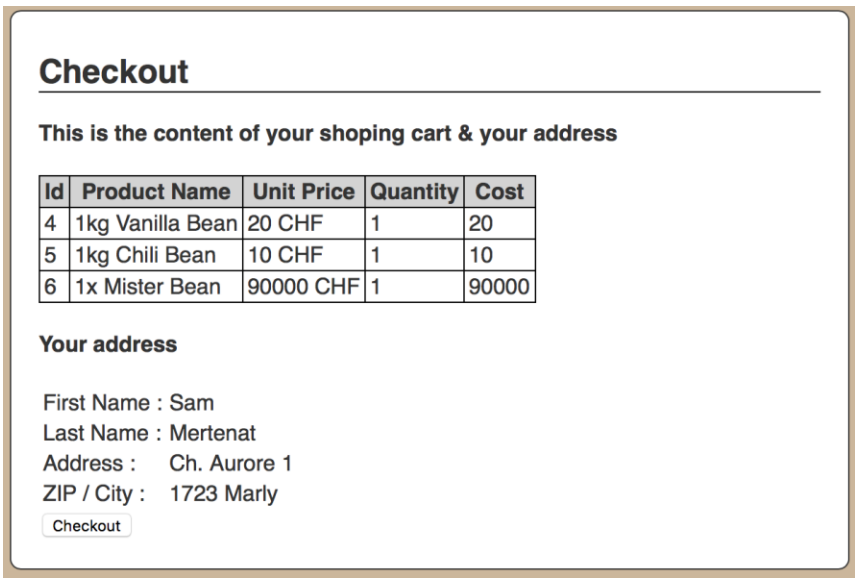
4.4 CHECKOUT

Cette page permet de confirmer son achat et l'envoyer. Cependant, si l'adresse postale n'est pas encore définie, un message s'affiche, demandant d'utiliser le formulaire sur la page « address ».

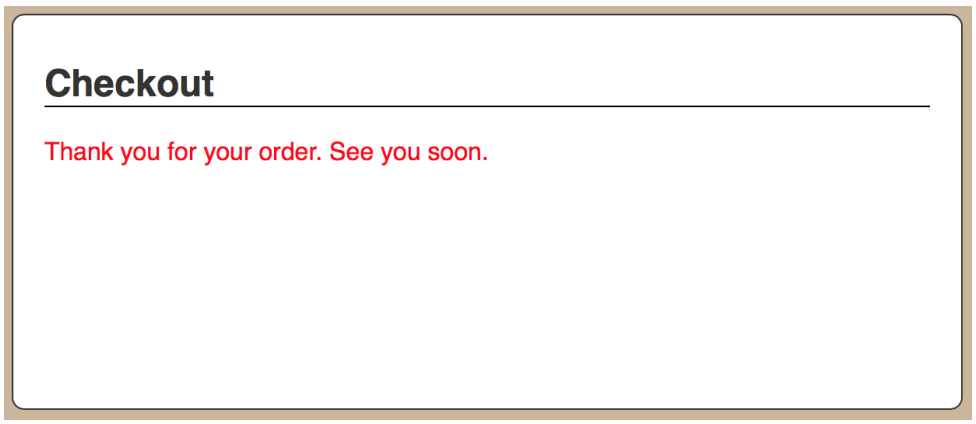
D'autre part, si le caddie est vide, un autre message s'affiche.



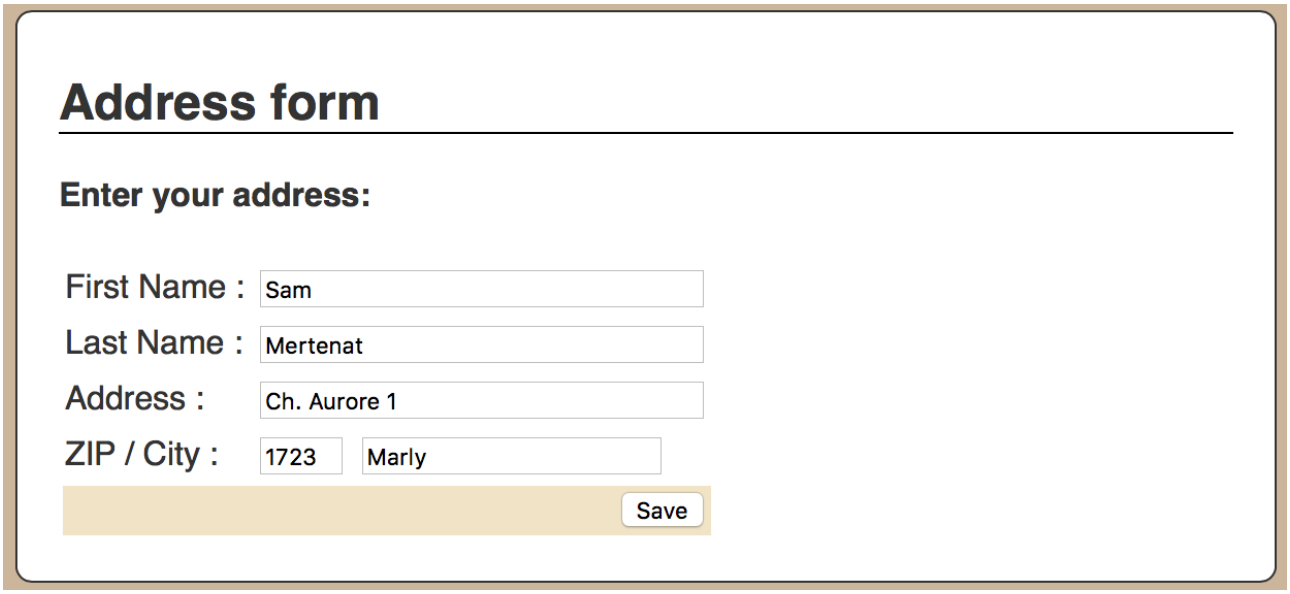
Dans le cas où toutes les conditions sont remplies pour afficher la page, nous obtenons un résumé de la commande ainsi que l'adresse de livraison.



Une fois la commande passée avec le bouton « Checkout », un message de remerciement s'affiche.



4.5 ADDRESS



Avant de pouvoir utiliser la page « Checkout », il est nécessaire de passer par ici pour définir son adresse postale grâce au formulaire ci-dessus.

5 CONCLUSION

Ce travail pratique a été l'occasion d'appréhender le fonctionnement des Java Script Pages en réalisant de petits scripts afin de gérer un shop au moyen de « JavaBean » au lieu d'utiliser une base de données,