

**IFC – Laboratoire n°4**

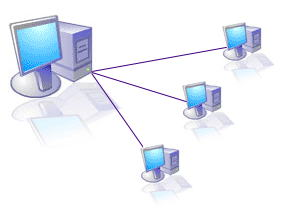
**Servlets et API Session : Mise en place d’un commerce electronique**

**Professeur**

**M. Markus JATON**

**Assistant**

M.Grégory RUCH

****

**Auteurs**

Steve LIENHARD

Arnaud BURKALTHER

**Dates :** 17 Avril 2010

hesso**Temps à disposition :** 4 semaines

**Date de début:** 25.03.2010

**Date de fin :** 29.04.2010

**Table des matières :**

[Introduction 4](#_Toc260235447)

[Serveur Web 4](#_Toc260235448)

[Installation et utilisation de tomcat 4](#_Toc260235449)

[Installation et utilisation d’orion 5](#_Toc260235450)

[Servlets 6](#_Toc260235451)

[Hello World 6](#_Toc260235452)

[doGet et doPost 7](#_Toc260235453)

[PrintWriter 7](#_Toc260235454)

[Affichage de la date et de l’heure 7](#_Toc260235455)

[Panier 7](#_Toc260235456)

[Modèle MVC 7](#_Toc260235457)

[Modèle 8](#_Toc260235458)

[Vue 8](#_Toc260235459)

[Contrôleur 8](#_Toc260235460)

[Les classes 9](#_Toc260235461)

[Traitement 9](#_Toc260235462)

[Articles 9](#_Toc260235463)

[Stock 9](#_Toc260235464)

[Formulaire 9](#_Toc260235465)

[API Session 9](#_Toc260235466)

[Questions supplémentaires 10](#_Toc260235467)

[Vol de session 10](#_Toc260235468)

[Durée de vie d’une session 10](#_Toc260235469)

[Interdiction des cookies 11](#_Toc260235470)

[Enumération 11](#_Toc260235471)

[Fermeture du navigateur 11](#_Toc260235472)

[Multi-client 11](#_Toc260235473)

[Modèle MVC 11](#_Toc260235474)

[Avantages/Inconvénients 11](#_Toc260235475)

[Conclusion 12](#_Toc260235476)

# Introduction

De nos jours, de multiples technologies Web existent, certaines meilleures que d’autres. Il est de ce fait primordial de se familiariser avec ces différentes techniques afin de rester à la page et de ne pas être dépassé. Parmi les langages Web qui existent, java a implémenter sa propre façon de faire, tant au niveau client que serveur. Les applications J2EE au niveau client se prénomment les « applets » et celle au niveau serveur les « servlets ». Ce sont les servlets qui vont nous intéressé dans le cadre de ce laboratoire. Comme nous le verrons, ces servlets offrent un certain nombre d’avantages que n’ont pas certain langages plus connus.

# Serveur Web

Afin de pouvoir développer notre application, il nous est nécessaire d’installer et de configurer un serveur web local supportant les servlets Java. Plusieurs choix se sont offerts à nous :

* **Orion :** pour sa simplicité d’installation et d’utilisation
* **Tomcat :** le plus célèbre et utilisé à l’heure actuelle
* **Jetty :** Léger, très populaire et simple à installer

Notre choix s’est finalement porté sur tomcat et orion.

Nous avons volontairement choisi deux serveurs web pour pouvoir d’une part nous familiariser avec les deux serveurs, ce qui peut être utile dans le future, et également pour pouvoir comparer les avantages et inconveignants de ceux-ci. Voici une brève marche à suivre pour l’installation des deux serveurs.

## Installation et utilisation de tomcat

Nous allons expliquer brièvement comment installer ce serveur sur une plateforme Linux Ubuntu (L’installation sur Windows est quelque peu différente, mais toute la doc nécessaire est disponible sur le Web).

La dernière version de tomcat disponible est la version 6. Nous avons donc téléchargé et installer les paquets utiles à son installation : tomcat6, tomcat6-admin, tomcat6-examples.

Une fois les paquets installés, le serveur peut-être lancé avec la commande :

* *sudo /etc/init.d/tomcat6 start*

et stoppé avec la commande :

* *sudo /etc/init.d/tomcat6 stop*

***Remarque :*** *Les commandes principales de tomcat se trouvent dans /etc/init.d/tomcat6*

A l’aide de notre navigateur préféré, nous pouvons accéder à l’accueil de tomcat à l’adresse :

* [*http://localhost:8080*](http://localhost:8080)

Afin d’accéder à l’interface « manager » de tomcat, il est nécessaire de modifié le fichier xml suivant :

* /usr/local/tomcat/conf/tomcat-users.xml

Et d’y ajouter un utilisateur dont le rôle est « manager ».

Une fois la configuration effectuée, nous pouvons ensuite nous connecter à l’interface d’administration du serveur à l’adresse :

* <http://localhost:8080/manager/html>

***Remarque :*** *Les informations relatives à l’installation de tomcat sur Ubuntu sont disponibles sur : http://doc.ubuntu-fr.org/tomcat.*

## Installation et utilisation d’orion

L’installation d’orion est nettement plus aisée que celle de tomcat, ce qui est comme mentionné précédemment son principal avantage. Il suffit de télécharger le serveur et de placer son répertoire à l’endroit souhaiter sur le disque. La commande pour démarrer le serveur est :

* *sudo java –jar orion.jar (dans le repertoire d’orion)*

Le serveur orion est ensuite accéssible directement à l’adresse :

* <http://localhost>

Pour pouvoir executer les servlets crées, il suffit de placer les fichier « .class » dans le répertoire :

* *sudo orion/default-web-apps/WEB-INF/classes*

Les servlets sont ensuite accessibles à l’adresse :

* <http://localhost>/servlet/nom\_servlets

# Servlets

Les servlets Java offrent la possibilité de générer dynamiquement des pages HTML sur un serveur Web. Par le terme « dynamique », nous entendons le fait que la page est générée sur la base de données entrées par l’utilisateur, par l’intermédiaire d’un formulaire par exemple, mais également de données extraites d’une BDD.

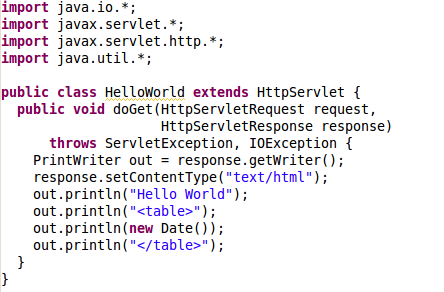
Le commerce électronique est un exemple typique d’utilisation de servlets, et c’est justement ce que nous allons implémenter lors de ce laboratoire.

Contrairement aux plugs-ins habituels, le servlet n’a pas besoin d’être installé, puisque son chargement est dynamique et peut également être téléchargé d’un serveur externes, tout comme peut l’être une applet.

Les servlets peuvent être utilisé par un serveur http **compatible java**, il ne convient de ce fait pas à n’importe quel serveur http. Les avantages et inconvénients de ces servlets sont indiqués plus loin dans le rapport.

# Hello World

La première application demandée dans le cadre de ce laboratoire est l’implémentation du fameux « Hello World » en utilisant des servlets. Etant donné que l’implémentation ce cette servlet est donné dans le cours, nous allons expliquer les différentes parties qui la composent :



Nous remarquons premièrement qu’il est nécessaire d’importer les paquetages javax.servlet.\* et javax.servlet.http qui servent à importer le code nécessaire à l’implémentation des servlets.

Notre classe principale noté « HelloWorld » doit forcément hériter de HttpServlet pour pouvoir fonctionner.

## doGet et doPost

Vient ensuite la méthode doGet qui prend en paramètre un requête de type HttpServletRequest, permettant de récupérer des informations des données entrantes (en-tête de requête http), ainsi qu’une reponse de type HttpServletResponse, permettant dans ce cas d’obtenir un PrintWriter afin de pouvoir écrire sur le flux de sortir (la page html).

Cette méthode est similaire à l’instruction $\_GET[] de PHP, qui permet de récupérer les valeurs des variables d’un formulaire, en passant par l’url de la page.

Pour les servlets, il existe également la méthode doPost(…), qui tout comme la méthode $\_POST[] de php, implique que les données ne seront pas contenues dans l’URL, ce qui malgré un temps de traitement légèrement accru, propose une meilleure sécurité. Ces deux méthodes renvoient deux types d’exception à traiter.

## PrintWriter

Comme expliqué dans le paragraphe précédent, le printWriter permet dans notre cas d’écrire sur le flux de sortie, à savoir la page HTML. Nous déclarons donc un PrintWriter out par le biais du paramètre response sur lequel nous pouvons appliquer la méthode getWriter() qui nous renvoie le flux sur la page html. Il nous suffit ensuite d’afficher « HelloWorld » sur ce flux.

## Affichage de la date et de l’heure

Afin d’afficher l’heure et la date dans le navigateur, il nous suffit simplement d’afficher dans le flux de sortie l’objet date.

# Panier

L’idée de cette application est d’implémenter un caddie virtuel permettant à un utilisateur de le remplir avec différents articles pour finalement passer à la caisse. Bien que ce caddie puisse être implémenté en PHP, il est évident que dans le cadre de ce laboratoire, il faille utiliser les servlets.

Pour ce faire nous avons implémenté un petit magasin virtuel qui propose diverses sortes de légumes à choix. Ces légumes sont stockés et sont épuisables s’il n’y en a plus dans le stock. Le client va donc se connecter et choisir les différents légumes qu’il veut acheter. Il va ensuite valider sa demande puis, si sa requête est correcte, va être redirigé vers la page affichant le contenu de son panier. Il peut ensuite choisir de recommander un article comme bon lui semble, ou sinon, valider sa commande et passer à la caisse. Afin de découvrir l’aspect des sessions dans les servlets, il est demandé que si le client se déconnecte soudainement du site pour y revient, son panier doit rester dans son état actuel (il ne doit donc pas être réinitialisé.)

Ci-dessous est présenté le schéma de notre application sous forme de « modèle MVC ».

# Modèle MVC

Le modèle présenté ci-dessous présente brièvement la place de chaque classe que nous avons implémentée, dans le modèle MVC.

**Modèle**

Stock

Article

**Vue**

Formulaire

**Controleur**

Traitement

Panier

Nous avons choisis de modéliser notre application de la manière suivante :

## Modèle

Nous avons créé une classe article qui contiendra les informations propres à chaque article, une classe Stock qui contiendra le nombre d’article de chaque sorte ainsi qu’une classe panier qui permet de stocker ce que le client a inséré dans son panier. Ces 3 classes font partie du modèle puisqu’elle modélise les données que l’on manipulera.

## Vue

La vue de notre application est tout simplement la page html qui affichera les informations comme le stock restant ou encore ce que le client a mis dans son panier.

## Contrôleur

Le contrôleur est composé de la classe « traitement » qui va permettre de créer les différentes instances des objets utilisés dans notre application. Le détail de cette classe est présenté dans la partie suivante.

# Les classes

## Traitement

Cette classe est le contrôleur de l’application. C’est ici que sera crée le panier ainsi que le stock (comme attribut statique) et les différents articles qui seront en ventes. Les articles sont ajoutés au stock grâce à la méthode « initialisation ». Suit ensuite la méthode principale « doPost ». C’est dans cette méthode qu’est initialisé la session et que les éléments envoyés par le formulaire sont ajoutés dans le panier. Cette classe permet aussi d’afficher l’état actuel du panier sous forme d’un tableau html.

## Articles

Cette classe permet simplement de stocker les diverses informations que comporte chaque article, à savoir : le nom de l’article, le prix, et la quantité.

## Stock

La classe stock comprend une liste des articles qui feront partie d’un stock. Un stock est créé et qui symbolise le stock du magasin et un autre stock qui symbolise le panier.

Cette classe comprend la méthode « void ajouterStock(Article article) » qui permet d’ajouter un article au stock, la méthode « int nombreArticle() » qui retourne le nombre d’articles compris dans le stock ainsi que la méthode « boolean retirerArticle(Article article) ». Une dernière méthode a été implémentée afin de pouvoir vider complètement le stock.

## Formulaire

Cette classe permet d’afficher la partie formulaire du code. Elle sera essentiellement composée de code html sa méthode principale « doPost ».

# API Session

Le fait que http soit un protocole sans état, donc qui coupe les connexions après chaque requête, implique qu’il faut utiliser un des mécanismes externes au protocole et propre au langage utilisé. Dans notre cas, Java propose la notion puissante de session et réservé à l’usage des servlets.

L’API Session de Java et très complet et comporte un certain nombre de méthodes. Nous allons expliquer le fonctionnement de celles que nous avons utilisé dans le cadre de ce laboratoire, ainsi que quelques unes qui nous semblent importantes

La création d’un objet session se fait de la manière suivante :

* HttpSession session = request.getSession(**true**);

La session est un objet de type « HttpSession » que nous initialisons grâce au paramètre request qui nous vient de la méthode doPost.

Afin de pouvoir récupérer le nombre d’accès au panier du client, nous avons créé une variable nbAcces par le biais de la méthode getValue appliquée sur la session. Cette méthode renvoie le nombre de connexion effectuée sur la session.

* Integer nbAcces = (Integer) session.getValue("sessiontest.counter");

A noter que nous ne proposons pas à l’utilisateur de clore la session. Celle-ci sera donc fermée après l’écoulement d’un timeout par défaut, qui est de 30 minutes. Le timeout est logiquement configurable, grâce à la méthode :

* public void **setMaxInactiveInterval**(int interval)

La valeur du timeout peut être récupérée par :

* public int **getMaxInactiveInterval**()

Nous pouvons également récupérer les attributs de n’importe quel objet appartenant à la session grâce à la méthode :

* public java.lang.Object **getAttribute**(java.lang.String name)

# Questions supplémentaires

## Vol de session

Voler une session sur un site bancaire, bien qu’étant passible d’emprisonnement, est tout à fait possible. Une des techniques possible pour se faire et de faire une attaque que l’on appelle  « Man in the middle ». Sans entrer dans les détails, cette attaque consiste à se faire passer pour le serveur par rapport au client. Ainsi, toutes les informations qui seront destinés au serveur transiteront par la machine du pirate qui pourra ensuite les exploiter à des fins néfastes. De ce fait, il pourra compromettre les informations à destinations d’un site bancaire et voler la session. Les conséquences peuvent être très graves puisque le pirate peut faire ce qu’il veut. La session initiée par la servlet n’a aucun moyen de vérifier si la source est bel et bien le bon client.

## Durée de vie d’une session

La durée de vie d’une session par défaut est fixée à 30 minutes, mais cette valeur peut être facilement modifiée, comme expliqué dans la partie « API Session ».

## Interdiction des cookies

Le fait d’interdire les cookies implique la fermeture de la session en cours, et donc que le panier est vidé. Ces cookies sont crées dans l’implémentation des sessions faite par Java.

## Enumération

Le risque de cette manière de faire est que si le client remplit son formulaire, l’envoie au serveur, puis modifier les données, puis renvoie au serveur etc…le serveur va à chaque fois stocké les données envoyées par l’utilisateur précédemment plus les nouvelles. Il en résulte donc l’inconvénient que si le client modifie 20 fois le formulaire, la page des informations stockée par le serveur va devenir très grande.

## Fermeture du navigateur

Il n’est pas possible de détecter la fermeture du navigateur depuis la servlet. Si on veut gérer ce genre de cas, il nous est possible par contre de modifier le timeout de la session pour le mettre plus bas et ainsi, lorsque le client se reconnectera, la session aura été interrompue par ce moyen.

## Multi-client

L’application implémentée gère très bien le fait que plusieurs utilisateurs se connectent en même temps et fassent leurs achats. Ce cas est géré par les sessions, qui vont attribuer un Id session à chaque utilisateur, qui vont ensuite pouvoir interagir avec la servlet comme bon leur semble, et ceci, sans perturber les interactions d’un autre utilisateur.

## Modèle MVC

Le modèle MVC ne peut pas être parfaitement respecté du fait que la vue fait également partie du contrôleur, dans le sens que les méthode doGet() et doPost() se trouvent dans le contrôleur alors que ceux-ci devraient faire partie de la vue.

# Avantages/Inconvénients

**Avantage :**

* Performance
* Langage de programmation très utilisé
* Bibliothèque très complète pour les requêtes http
* Flexibilité : connexion à une BDD ou à un serveur possible
* Portabilité
* Faible coût

**Désavantage :**

* **Nécessite la machine virtuelle java sur le client**
* **Prend une place non négligeable sur le serveur**

# Conclusion

Ce laboratoire nous a permis de nous familiariser avec la technique des servlets. Nous avons pu découvrir cette technologie Web moderne offerte par Java, et nous familiariser avec cette manière de faire. Malgré les autres technologies web permettant d’implémenter les mêmes choses qu’une servlets, nous sommes maintenant en mesure de nous rendre compte des principaux avantages et inconvénients de cette méthode. D’autre part, nous avons également découvert l’utilité de concevoir des sessions au travers de l’API Session, ainsi que quelques aspects de sécurité à ne surtout pas négliger dans le cadre d’une application Web.