Une boutique en ligne
L'enregistrement de travail

Durée : De 1 mai à la fin d'août

Stage 2A

ZHANG Quan
q.zhang@net.estia.fr

Table des matières

目录

1 Introduction **3**

2 Analyse de base de données **4**
2.1 Diagramme de cas d'utilisation 4
2.2 Les relations parmi des entités 4
2.3 Les entités 4
2.4 La base de données 5

3 Les travaux de preparation **7**
3.1 Creation du projet 7
3.2 Introduction de l'environnement de développement 8
3.3 L'extraction de servlet 9
3.3.1 Analyse 9
3.3.2 Configuration de BaseServlet 12
3.4 Interface d'erreur unifiée 12
3.5 L'extraction de page front-end 14
3.6 Configuration de page defaut 15
3.7 Definition des valeurs constants 15

4 Découplage **16**
4.1 L'exemple: Hibernate de cadre SSH 16
4.2 Mise en œuvre du découplage 16

5 Développement du module utilisateur **18**
5.1 UserDaoImpl 18
5.2 Inscription 19
5.2.1 Captcha 19
5.2.2 JSP 21
5.2.3 La methode 'regist' 22
5.3 Authentification par courriel 24
5.4 Activiation 25
5.5 Login 26
5.5.1 JSP 26
5.5.2 La methode login 27
5.6 Exit 28

6 Affichage des produit et catégories **29**
6.1 CategoryDaoImpl 29
6.2 ProduitDaoImpl 29
6.3 Introduction de Redis 31
6.4 Afficher des catégories 32
6.4.1 Introduction de Ajax 32
6.4.2 Configuration des methodes JSON : JsonUtils 33

6.4.3 Réaliser un affichage catégories 35
6.5 Affichage les produit les plus vendus ou Nouveaux 35
6.6 Affichage des détails du produit 37
6.7 Affichage les produits par catégories 39

7 Mise en œuvre du panier d'achat 42
7.1 OrderDaoImpl 42
7.2 Réalisation d'entité Panier et Articles du panier 44
7.3 Ajouter les produits 45
7.4 Enlever les produits 47
7.5 Vider le panier 48
7.6 Générer des commandes 48

8 Payement 52
8.1 La liste de commandes 52
8.2 Les infors des commandes 53
8.3 Filtrage des autorisations 53

9 Système de gestion 55
9.1 Liste de category 55
9.2 Ajouter les categories 56
9.3 Afficher les produits 57
9.4 Ajouter les produits 57
9.4.1 Upload/Transfert des photos 57
9.5 Interroger la liste des commandes par statut 59
9.6 Interroger les détails de la commande 60
9.7 Modifier le statut de la commande 61

1 Introduction

L'objectif est de développer un site Internet et de le mettre en ligne. Le site Web mettra en œuvre les fonctions suivantes:

1. Gardez mon CV
2. Développer un album photo en ligne et partager les photos de mon voyage.
3. Partagez diverses vidéos.
4. Mettez en place une boutique en ligne.
5. Réalisez le système d'enregistrement et de connexion des utilisateurs. Selon le code d'enregistrement, différents types d'utilisateurs auront différentes opérations. (Les visiteurs ne sont autorisés à accéder qu'aux vidéos, albums, pages d'accueil personnelles; les enquêteurs sont autorisés à accéder aux CV, etc.)

La réalisation de ces fonctions est nécessaire pour l'avenir. Comme il ressort de ces exigences, la difficulté ne réside pas dans la construction de pages statiques, mais dans des pages dynamiques. Par conséquent, le thème de ce projet personnel choisit de réaliser **une boutique en ligne** pour maîtriser les technologies de développement front-end et back-end de la page web.

Afin de bien maîtriser cette technologie, une bonne capacité d'apprentissage est essentielle. J'espère qu'à la fin de ce projet, je pourrai développer toutes les fonctions que je souhaite. Le support technique de ce projet est le suivant:

1. Logiciel de programmation : Eclipse (Tomcat v9.0, jdk 1.8.0).
2. Langage de programmation front-end : HTML, JavaScript, css.
3. Langage de programmation back-end : Java.
4. Cadre frontal : jquery, bootstrap.
5. Environnement de développement intégré : phpstudy.
6. Outils de gestion graphique de la base de données MYSQL : SQLyog.
7. Serveur : Alibaba Cloud.
8. Technologie spéciale : Ajax, Redis, json, Layui

2 Analyse de base de données

2.1 Diagramme de cas d'utilisation

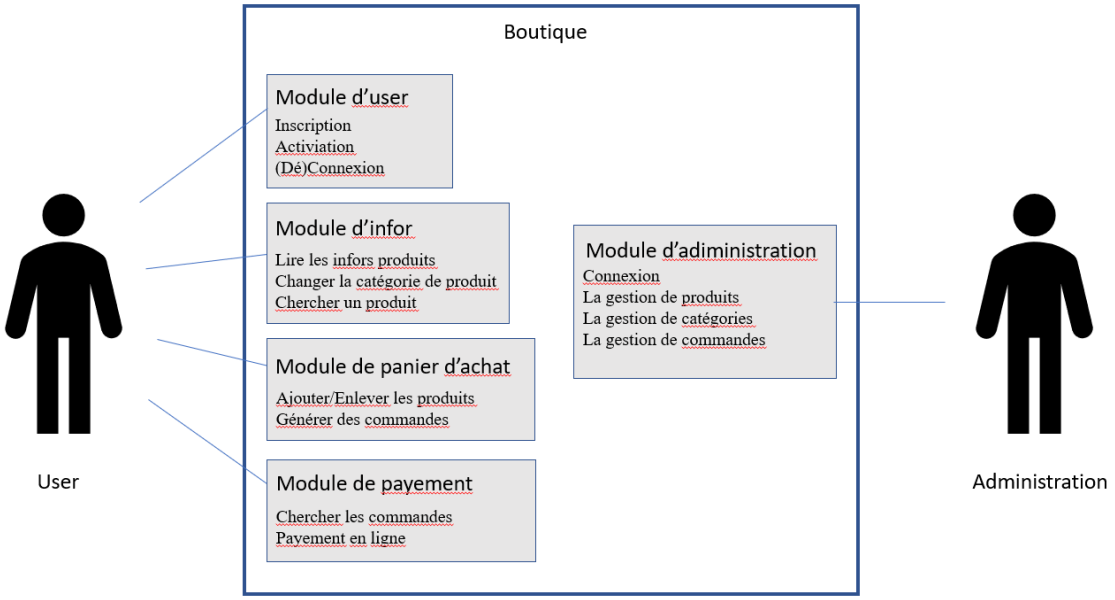


Fig. 1: Diagramme de cas d'utilisation

2.2 Les relations parmi des entités

Users et orders: 1,n
produits et orders: n,n
Catégories et produits: 1,n
Parce que la relation entre les commandes et les produits est n,n. On a besoin d'une nouveau tableau comme 'orderitem' pour traiter les relations entre les deux.

2.3 Les entités

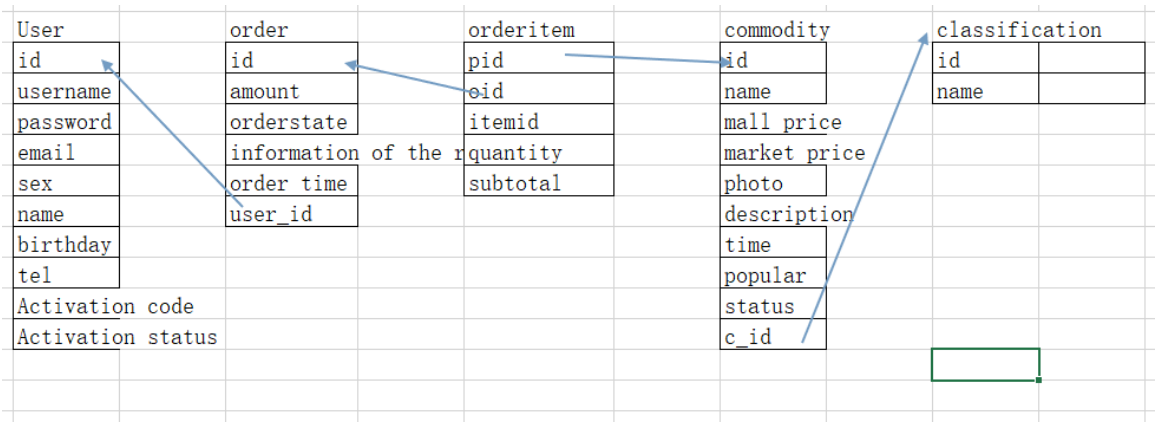


Fig. 2: Les relations parmi des parties

2.4 La base de données

Tableau de 'user':

```
create database store38;
use store38;
CREATE TABLE `user` (
`uid` varchar(32) NOT NULL,
`username` varchar(20) DEFAULT NULL,
`password` varchar(20) DEFAULT NULL,
`name` varchar(20) DEFAULT NULL,
`email` varchar(30) DEFAULT NULL,
`telephone` varchar(20) DEFAULT NULL,
`birthday` date DEFAULT NULL,
`sex` varchar(10) DEFAULT NULL,
`state` int(11) DEFAULT NULL,
`code` varchar(64) DEFAULT NULL,
PRIMARY KEY (`uid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Tableau de 'order':

```
CREATE TABLE `orders` (
`oid` VARCHAR(32) NOT NULL,
`ordertime` DATETIME DEFAULT NULL,
`total` DOUBLE DEFAULT NULL,
`state` INT(11) DEFAULT NULL,
`address` VARCHAR(30) DEFAULT NULL,
`name` VARCHAR(20) DEFAULT NULL,
`telephone` VARCHAR(20) DEFAULT NULL,
`uid` VARCHAR(32) DEFAULT NULL,
PRIMARY KEY (`oid`)
) ENGINE=INNODB DEFAULT CHARSET=utf8;
```

Tableau de 'orderitem':

```
CREATE TABLE `orderitem` (
`itemid` VARCHAR(32) NOT NULL,
`count` INT(11) DEFAULT NULL,
`subtotal` DOUBLE DEFAULT NULL,
`pid` VARCHAR(32) DEFAULT NULL,
`oid` VARCHAR(32) DEFAULT NULL,
PRIMARY KEY (`itemid`),
KEY `fk_0001` (`pid`),
KEY `fk_0002` (`oid`),
CONSTRAINT `fk_0001` FOREIGN KEY (`pid`) REFERENCES `product` (`pid`),
CONSTRAINT `fk_0002` FOREIGN KEY (`oid`) REFERENCES `orders` (`oid`)
) ENGINE=INNODB DEFAULT CHARSET=utf8;
```

Tableau de 'commodity' ou bien 'produit':

```sql
CREATE TABLE `product` (
`pid` varchar(32) NOT NULL,
`pname` varchar(50) DEFAULT NULL,
`market_price` double DEFAULT NULL,
`shop_price` double DEFAULT NULL,
`pimage` varchar(200) DEFAULT NULL,
`pdate` date DEFAULT NULL,
`is_hot` int(11) DEFAULT NULL,
`pdesc` varchar(255) DEFAULT NULL,
`pflag` int(11) DEFAULT NULL,
`cid` varchar(32) DEFAULT NULL,
PRIMARY KEY (`pid`),
KEY `sfk_0001` (`cid`),
CONSTRAINT `sfk_0001` FOREIGN KEY (`cid`) REFERENCES `category` (`cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Tableau de 'classification' ou bien 'catagory':

```sql
CREATE TABLE `category` (
`cid` varchar(32) NOT NULL,
`cname` varchar(20) DEFAULT NULL,
PRIMARY KEY (`cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

3 Les travaux de preparation

3.1 Creation du projet

1. Créer Les tableaus comme 'user' (Comme dans 2.4).

Field	Type
uid	varchar(32) NOT NULL
username	varchar(20) NULL
password	varchar(20) NULL
name	varchar(20) NULL
email	varchar(30) NULL
telephone	varchar(20) NULL
birthday	date NULL
sex	varchar(10) NULL
state	int(11) NULL
code	varchar(64) NULL

Fig. 3: La configuration de données(tableau de 'User')

2. Créer un nouveau projet:
   - **Importer le package jar**: 'Device Driver', c3p0(Pour la connexion de bases de données), dbutils(Pour trouver ou bien stoker les données), beanutils(Pour l'inscription d'utilisateur), jstl(Pour les pages web), mail(Envoyer les mails pour activiation).....
   - **Créer les fichiers Back-end**: Les packages, les interfaces, services, servlets, et les utils..... ce qu'on va travailler dedans.
   - **Créer les fiches de pages web** Des pages fondamentales, les pakages css, les fichiers d'image....

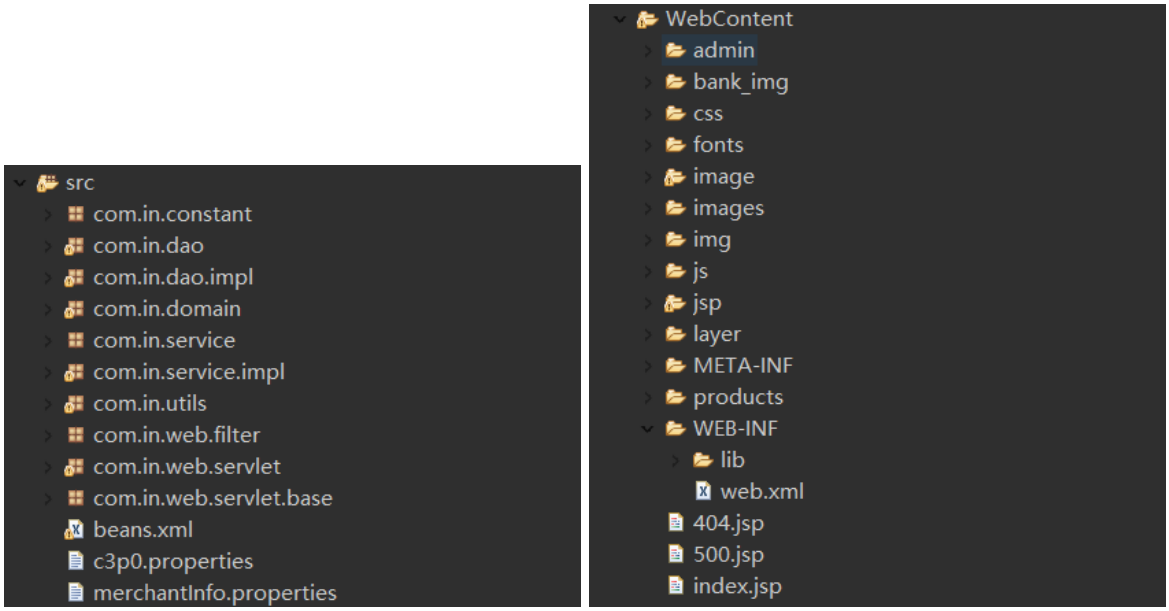


Fig. 4: La structure de code

3. Configurer les fiches comme c3p0.properties, index.jsp, web.xml... afin de connecter la base de données.

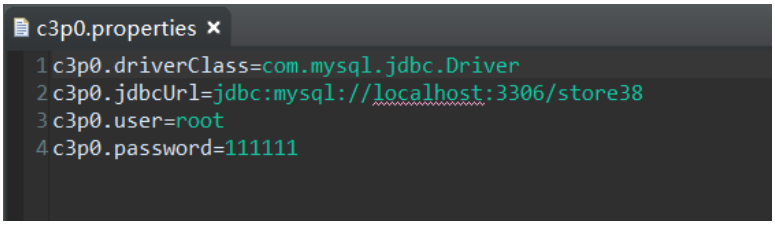


Fig. 5: La configuration de c3p0

3.2 Introduction de l'environnement de développement

phpstudy est un environnement de développement connu en Chine comme Wampserver. Dans ce projet, je vais l'utiliser avec **SQLyog** pour la gestion de données. La configuration :

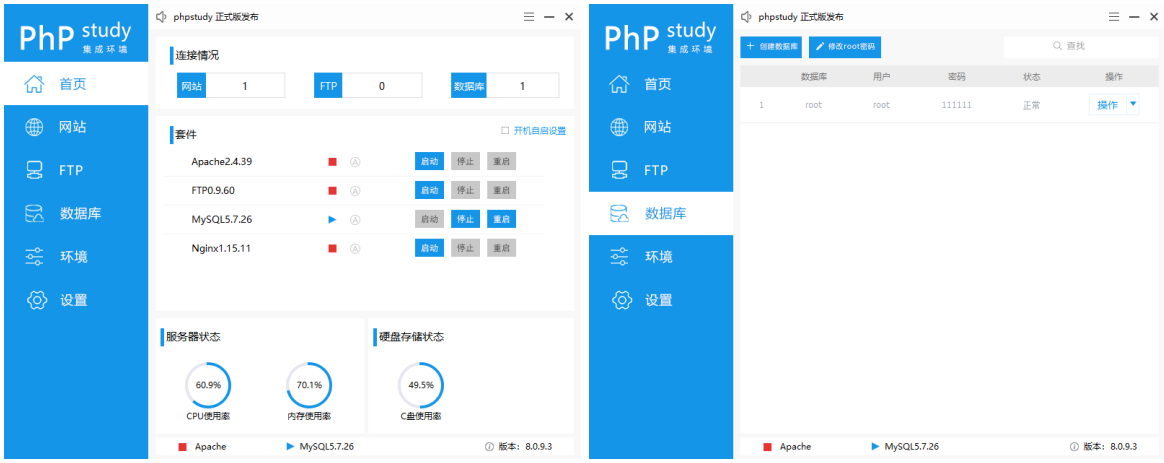
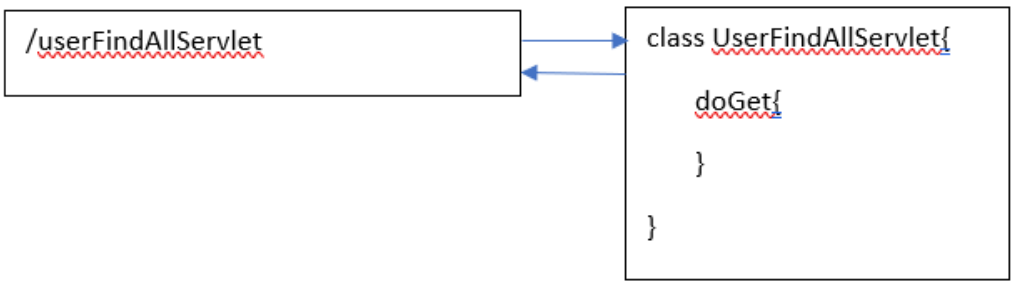


Fig. 6: La configuration de phpstudy

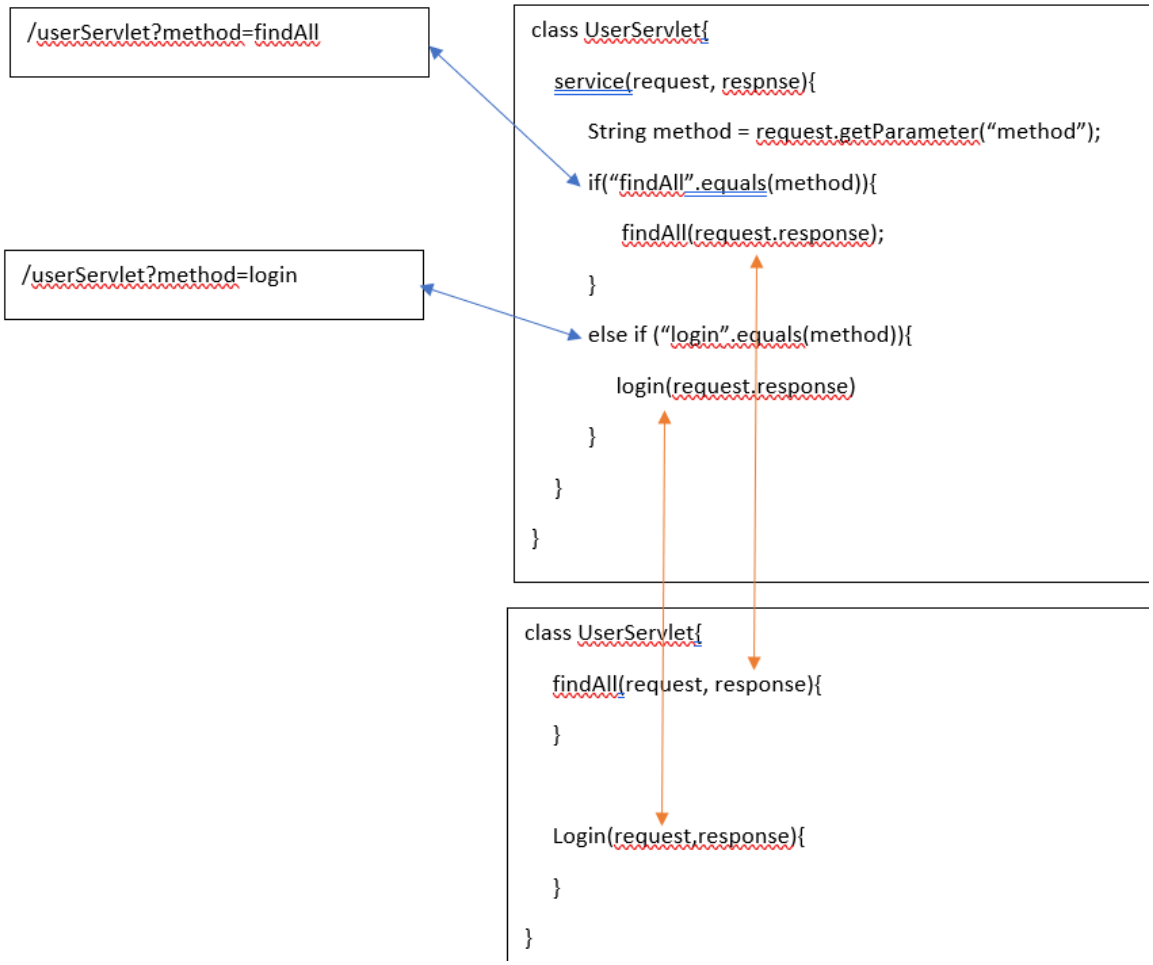
3.3 L'extraction de servlet

3.3.1 Analyse

- Methode 1: Une path correspendent un servlet



- Methode 2: Des paths correspendent un servlet



- Methode 3: Des paths correspendent un servlet et on va creer BaseServlet et utiliser la methode "invoke"

```
class BaseServlet{
    service(request,response){
    String methodName=request.getParameter("method");
    Class clazz = this.getClass();
    Method method = clazz.getMethod(methodName, HttpServletRequest.class,HttpServletResponse.class);
    method.invoke(this, request, response);
    }
}
```

```
/userServlet?method=findAll
```

```
class UserServlet extends BaseServlet{
    findAll(request, response){

    }
}
```

- Methode 4: Completer BaseServlet et retourner dans la page Jsp

```
class BaseServlet{
    service(request,response){
        String methodName=request.getParameter("method");
        Class clazz = this.getClass();
        Method method = clazz.getMethod(methodName, HttpServletRequest.class,HttpServletResponse.class);
        String jspPath=method.invoke(this, request, response);
        If(jspPath != null){request.getRequestDispatcher(jspPath).forward(request, response);
        }
    }
}
```

```
/userServlet?method=findAll
```

```
class UserServlet extends BaseServlet{
    findAll(request, response){
    return "/jsp/list.jsp";
    }
}
```

Après sélection, nous choisissons la dernière méthode. Il peut non seulement résoudre le problème de

redondance des fichiers, mais également éviter certains problèmes inutiles.

3.3.2 Configuration de BaseServlet

On va configurer BaseServlet, ca va realiser les fonctionnalités suivantes:

- Recevoir la paramètre "methode"
- Comparer et déterminer la méthode à exécuter
- Exécuter la methode
- Demande de transfert à la page jsp par la valeur retour de methode

```java
public class BaseServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        try {
            // 1. Obtenir le nom de la methode
            String mName = request.getParameter("method");

            //1.1 Determiner si les parametres de methode sont transmis
            if(mName == null || mName.trim().length()==0) {
                mName = "index";
            }

            //2. Obtenir un objet de methode
            Method method = this.getClass().getMethod(mName, HttpServletRequest.class, HttpServletResponse.class);

            //3. Laisser la methode s'executer et recever la valeur de retour.
            String path = (String) method.invoke(this, request, response);

            //4. Determiner si la valeur de retour est vide, sinon transfert de demande
            if(path != null) {
                request.getRequestDispatcher(path).forward(request, response);
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException();
        }
    }
        public String index(HttpServletRequest request, HttpServletResponse response)throws  ServletException,IOException{
            response.setContentType("text/html;charset=utf-8");
            response.getWriter().println("~~Sans blague svp~~~");
            return null;
        }
}
```

3.4 Interface d'erreur unifiée

On cree pages d'erreur 500 et 404:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    亲，不好意思，内部错误。Désolé, erreur interne.
</body>
</html>
```

Fig. 7: page 500

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
  亲，请检查你的访问路径，功能暂时还没有开发出来。La fonction n'a pas encore été développée, merci de vérifier votre chemin d'accès.
</body>
</html>
```

Fig. 8: page 404

Configuration de web.xml:

```
<servlet>
    <description></description>
    <display-name>UserServlet</display-name>
    <servlet-name>UserServlet</servlet-name>
    <servlet-class>com.in.web.servlet.UserServlet</servlet-class>
</servlet>
<servlet>
    <description></description>
    <display-name>BaseServlet</display-name>
    <servlet-name>BaseServlet</servlet-name>
    <servlet-class>com.in.web.servlet.base.BaseServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>UserServlet</servlet-name>
    <url-pattern>/user</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>BaseServlet</servlet-name>
    <url-pattern>/base</url-pattern>
</servlet-mapping>
<error-page>
<error-code>500</error-code>
<location>/500.jsp</location>
</error-page>
<error-page>
<error-code>404</error-code>
<location>/404.jsp</location>
</error-page>
```

Attention! Chaque foi on cree un servlet, on doit changer url et ajouter dans la fiche web.xml.

3.5 L'extraction de page front-end

On cree un jsp s'appelle 'head.jsp' pour controler tous les fiches front-end et pour déterminer l'état de connexion malgré le saut de page. Pour le faire, on met Barre de menu et la Barre de navigation dedans.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
    <!--
                Menu
            -->
            <div class="container-fluid">
            <div class="col-md-4">
                    <img src="${pageContext.request.contextPath}/img/logo3.png" />
                </div>
                <div class="col-md-3">
                    <img src="${pageContext.request.contextPath}/img/header.png" />
                </div>
                <div align="right" class="col-md-5" style="padding-top:20px">
                    <ol class="list-inline">
                        <c:if test="${empty user}">
                            <li><a href="${pageContext.request.contextPath }/user?method=loginUI">Je me connecte</a></li>
                            <li><a href="${pageContext.request.contextPath }/user?method=registUI">Je m'inscris</a></li>
                        </c:if>
                        <c:if test="${not empty user}">
                           Bonjour! ${user.name }
                           <li><a href="${pageContext.request.contextPath }/order?method=findMyOrdersByPage&pageNumber=1">Mes commandes</a></li>
                            <li><a href="${pageContext.request.contextPath }/user?method=logout">Deconnection</a></li>
                        </c:if>
                        <li><a href="${pageContext.request.contextPath }/jsp/cart.jsp">Panier</a></li>
                    </ol>

                </div>
            </div>
```

```
<!--
    Navigation
-->
<div class="container-fluid">
    <nav class="navbar navbar-inverse">
        <div class="container-fluid">
            <!-- Brand and toggle get grouped for better mobile display -->
            <div class="navbar-header">
                <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expan
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="${pageContext.request.contextPath }">Accueil</a>
            </div>

            <!-- Collect the nav links, forms, and other content for toggling -->
            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav" id="c_ul">
                </ul>
                <form class="navbar-form navbar-right" role="search">
                    <div class="form-group">
                        <input type="text" class="form-control" placeholder="Search">
                    </div>
                    <button type="submit" class="btn btn-default">Submit</button>
                </form>

            </div>
            <!-- /.navbar-collapse -->
        </div>
        <!-- /.container-fluid -->
    </nav>
</div>
```

Au-dessus des fichiers, on doit ajouter un 'taglib' pour déterminer l'état de connexion et importer jsp 'head.jsp' dedans.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>


<%@include file="/jsp/head.jsp" %>
```

3.6 Configuration de page defaut

On defini la page defaut (index.jsp) pour la Saut automatique. La configuration dans index.jsp(/WebContent/index.jsp):

```
<%
response.sendRedirect(request.getContextPath()+"/jsp/index.jsp");
//saut automatique a la page d'accueil
%>
```

3.7 Definition des valeurs constants

```
public interface Constant {
  /*
   * Activiation
   */
  int USER_IS_NOT_ACTIVE = 0;
  int USER_IS_ACTIVE = 1;

  /*
   * Mémoriser le nom du compte
   */
  String SAVE_NAME = "ok";
  /**
     *La clé de la liste de catagory stockée dans redis
     */
    String STORE_CATEGORY_LIST="STORE_CATEGORY_LIST";

    /**
     * adresse du serveur de redis
     */
    String REDIS_HOST = "192.168.17.136";

    /**
     * Numéro de port du serveur de redis
     */
    int  REDIS_PORT =  6379;
    /**
     * Les produits plus vendus
     */
    int PRODUCT_IS_HOT = 1;

    /**
     * Statut de produit
     */
    int PRODUCT_IS_UP = 0;
    int PRODUCT_IS_DOWN = 1;
    /**
```

```
/**
 * Statut de Order (non payé)
 *
 */
int ORDER_WEIFUKUAN=0;

/**
 * Payé
 */
int ORDER_YIFUKUAN=1;

/**
 *  Expédié
 */
int ORDER_YIFAHUO=2;

/**
 *  Fini
 */
int ORDER_YIWANCHENG=3;
```

4 Découplage

Afin d'obtenir un code maintenable, réutilisable et extensible. Le découplage est nécessaire, c'est-à-dire abstrait du code qui apparaît à plusieurs reprises.

4.1 L'exemple: Hibernate de cadre SSH

Lorsque nous créons deux classes d'implémentation et choisissons la même interface. Par exemple, dans ProductServiceImpl.java, on code comme:

```
ProductDao pd = new ProductDaoImpl();
```

En suite, quand on cree un class s'appelle ProductDaoHibImpl.java qui a la même interface avec ProductServiceImpl.java. Puis on doit changer cette phrase par:

```
ProductDao pd = new ProductDaoHibImpl();
```

Cela montre que lorsque le volume du projet augmente, mais nous devons fondamentalement tout changer quand même. Cela a provoqué de nombreuses opérations répétitives, ce qui signifie que le code est couplé.

4.2 Mise en œuvre du découplage

Généralement, on peut creer un class qui s'appelle BeanFactory, dans lequel on peut mettre:

```
BeanFactory

public static Object getProductDao{
    return new productDaoImpl();

    return new productDaoHibImpl();
}
```

Et on change le code précédent par

```
ProductDao pd = beanfactory.getProductDao();
```

Comme ca, on doit changer le code quand même, mais la charge de travail est bien moindre qu'auparavant. Nous utilisons donc une méthode plus intelligente. Creer un class BeanFactory.java et Configurer dans bean.xml comme:

```java
public class BeanFactory {

    public static Object getBean(String id){
        try {
            //1.获取document对象
            Document doc=new SAXReader().read(BeanFactory.class.getClassLoader().getResourceAsStream("beans.xml"));

            //2.调用api selectSingleNode(表达式) Xpath
            Element beanEle=(Element) doc.selectSingleNode("//bean[@id='"+id+"']");

            //3.获取元素的class属性
            String classValue = beanEle.attributeValue("class");

            //4.通过反射返回实现类的对象
            Object newInstance = Class.forName(classValue).newInstance();
            return newInstance;
        }  catch (Exception e) {
            e.printStackTrace();
            System.out.println("Impossible d'obtenir le bean");
        }
        return null;
    }

    public static void main(String[] args) throws Exception {
        System.out.println(getBean("ProductDao1"));
    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans>
   <bean id="ProductDao" class="com.in.dao.impl.ProductDaoImpl"/>
   <bean id="UserDao" class="com.in.dao.impl.UserDaoImpl"/>
   <bean id="CategoryDao" class="com.in.dao.impl.CategoryDaoImpl"/>
   <bean id="OrderDao" class="com.in.dao.impl.OrderDaoImpl"/>

   <bean id="ProductService" class="com.in.service.impl.ProductServiceImpl"/>
   <bean id="UserService" class="com.in.service.impl.UserServiceImpl"/>
   <bean id="CategoryService" class="com.in.service.impl.CategoryServiceImpl"/>
   <bean id="OrderService" class="com.in.service.impl.OrderServiceImpl"/>
</beans>
```

Et on change le code précédent par

```java
ProductDao pd = (ProductDao)Beanfactory.getBean("ProductDao");
```

Comme ca, quand on va utiliser d'autre cadre, on peut juste changer url dans bean.xml.

5 Développement du module utilisateur

Pour le module d'utilisateur, il faut 4 fonctions basics: **Inscription, Activiation, Connexion et Déconnexion**. Les travaux sont:

5.1 UserDaoImpl

Après la connexion à la base de données, nous créons les methodes pour les opérations.

```java
public class UserDaoImpl implements UserDao{

    /*
     * Enregistrer l'utilisateur
     */
    public void save(User user) throws SQLException {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        /*
         *  `uid` varchar(32) NOT NULL,
         `username` varchar(20) DEFAULT NULL,
         `password` varchar(20) DEFAULT NULL,
         `name` varchar(20) DEFAULT NULL,
         `email` varchar(30) DEFAULT NULL,
         `telephone` varchar(20) DEFAULT NULL,
         `birthday` date DEFAULT NULL,
         `sex` varchar(10) DEFAULT NULL,
         `state` int(11) DEFAULT NULL,
         `code` varchar(64) DEFAULT NULL,
         */
        String sql = "insert into user values(?,?,?,?,?,?,?,?,?,?);";
        qr.update(sql, user.getUid(), user.getUsername(), user.getPassword(), user.getName(), user.getEmail(), user.getTelephone(), user.getBirthday(),
    }

    @Override
    /*
     * Trouver user par code d'activiation ou username
     */
    public User getByCode(String code) throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from user where code = ? limit 1";
        return qr.query(sql, new BeanHandler<>(User.class), code);
    }

    public User getByUsername(String username) throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from user where username = ? limit 1";
        return qr.query(sql, new BeanHandler<>(User.class), username);
    }
```

```java
    /*
     * Mettre à jour l'utilisateur
     */
    public void update(User user) throws Exception {
         QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
         /*
             *  `uid` varchar(32) NOT NULL,
             `username` varchar(20) DEFAULT NULL,
             `password` varchar(20) DEFAULT NULL,
             `name` varchar(20) DEFAULT NULL,
             `email` varchar(30) DEFAULT NULL,
             `telephone` varchar(20) DEFAULT NULL,
             `birthday` date DEFAULT NULL,
             `sex` varchar(10) DEFAULT NULL,
             `state` int(11) DEFAULT NULL,
             `code` varchar(64) DEFAULT NULL,
             */
         String sql = "update user set password = ?, state = ?, code = ? where uid = ? ";
         qr.update(sql, user.getPassword(), user.getState(), user.getCode(), user.getUid());

    }

    /*
     * Login
     */
    public User getByUsernameAndPwd(String username, String password) throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from user where username = ? and password = ? limit 1";

        return qr.query(sql, new BeanHandler<>(User.class),username,password);
    }

}
```

5.2 Inscription

5.2.1 Captcha

Captcha est le moyen le plus efficace de prévenir les attaques DDOS. Nous utilisons la méthode la plus courante: générer Captcha localement et produire le résultat, générer une image du Captcha et le soumettre en ligne. Pour le faire, on a besoin d'une classe utils(CaptchaUtil.java):

```
public class CaptchaUtil {

    public static String [] fuhao = {"+","*"};
    public static int num = 0;

    //Générer une formule
    public static String random(){

        List list = new ArrayList();
        Random ran = new Random();
        int n1 = ran.nextInt(10);
        int n2 = ran.nextInt(10);
        String f = fuhao[ran.nextInt(fuhao.length)];
        StringBuilder sb = new StringBuilder(4);

        switch (f){
            case "+":
                num = n1+n2;
                break;
            case "*":
                num = n1*n2;
                break;
        }
        sb.append(n1);
        sb.append(f);
        sb.append(n2);
        sb.append("=");

        return sb.toString();
    }
```

```java
    public static void outputImage(String str, OutputStream os){
        Random ran = new Random();
        //Créer un objet image
        BufferedImage img = new BufferedImage(100,40, BufferedImage.TYPE_INT_RGB);
        //Obtenir une toile
        Graphics g = img.getGraphics();
        //Définir l'arrière-plan
        g.setColor(Color.white);
        g.fillRect(0,0,100,40);
        //Ecrire
        g.setColor(Color.black);
        g.setFont(new Font("微软雅黑",Font.PLAIN,30));
        g.drawString(str,10,40);
        //Génère aléatoirement trois lignes
        g.drawLine(ran.nextInt(100),ran.nextInt(40),
                ran.nextInt(100),ran.nextInt(40));
        g.drawLine(ran.nextInt(100),ran.nextInt(40),
                ran.nextInt(100),ran.nextInt(40));
        g.drawLine(ran.nextInt(100),ran.nextInt(40),
                ran.nextInt(100),ran.nextInt(40));
        // Sortie
        // Formats de compression courants: jpeg (compression avec perte), png (compression sans perte)...
        try {
            ImageIO.write(img, "png", os);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Dans CaptchaServlet.java:

```java
@WebServlet(urlPatterns = "/captcha.png")
public class CaptchaServlet extends HttpServlet {

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

        response.setContentType("image/png");
        //Obtenir une chaîne aléatoire
        String random = CaptchaUtil.random();
        //Sortie
        CaptchaUtil.outputImage(random,response.getOutputStream());
        //Transmettre en ligne
        request.getSession().setAttribute("res",CaptchaUtil.num);

    }


}
```

Afin de réaliser la fonction de cliquer pour actualiser le Captcha, on cree une fonction par JavaScript qui s'appelle changeImg(img). Dans la page front-end:

```html
<label for="date" class="col-sm-2 control-label">Captcha</label>
<div class="col-sm-2">
<img src="${pageContext.request.contextPath }/captcha.png" onclick="changeImg(this)">${msg2 }
</div>
<div class="col-sm-3">
  <input type="text" class="form-control"  name="cap">

</div>
```

```
<script>
    function changeImg(img) {
        img.src = "${pageContext.request.contextPath }/captcha.png?t="+new Date().getTime();
    }
</script>
```

5.2.2 JSP

```
<div class="col-md-8" style="background:#fff;padding:40px 80px;margin:30px;border:7px solid #ccc;">
    <font>会员注册</font>USER REGISTER ${msg }
    <form class="form-horizontal" style="margin-top:5px;" method="post" action="${pageContext.request.contextPath }/user">
        <input type="hidden" name="method" value="regist">
         <div class="form-group">
            <label for="username" class="col-sm-2 control-label">Username</label>
            <div class="col-sm-6">
              <input type="text" class="form-control" id="username" placeholder="Veuillez saisir le nom d'utilisateur" name="username">
            </div>
          </div>
           <div class="form-group">
            <label for="inputPassword3" class="col-sm-2 control-label">Mot de passe</label>
            <div class="col-sm-6">
              <input type="password" class="form-control" id="inputPassword3" placeholder="Veuillez saisir le mot de passe" name="password">
            </div>
          </div>
           <div class="form-group">
            <label for="confirmpwd" class="col-sm-2 control-label">Confirmation de mot de passe</label>
            <div class="col-sm-6">
              <input type="password" class="form-control" id="confirmpwd" placeholder="Veuillez confirmer le mot de passe" name="conf">${msg1 }
            </div>
          </div>
          <div class="form-group">
            <label for="inputEmail3" class="col-sm-2 control-label">Email</label>
            <div class="col-sm-6">
              <input type="email" class="form-control" id="inputEmail3" placeholder="Email" name="email">
            </div>
          </div>
         <div class="form-group">
            <label for="usercaption" class="col-sm-2 control-label">Nom</label>
            <div class="col-sm-6">
              <input type="text" class="form-control" id="usercaption" placeholder="Veuillez saisir votre nom" name="name">
            </div>
          </div>
```

```
            <div class="form-group opt">
            <label for="inlineRadio1" class="col-sm-2 control-label">Sexe</label>
            <div class="col-sm-6">
              <label class="radio-inline">
            <input type="radio" name="sex" id="inlineRadio1" value="1"> M
          </label>
          <label class="radio-inline">
            <input type="radio" name="sex" id="inlineRadio2" value="0"> F
          </label>
          </div>
            </div>
            <div class="form-group">
              <label for="date" class="col-sm-2 control-label">Naissance</label>
              <div class="col-sm-6">
                <input type="date" class="form-control"  name="birthday">
              </div>
            </div>
            <div class="form-group">
              <label for="date" class="col-sm-2 control-label">Captcha</label>
              <div class="col-sm-2">
              <img src="${pageContext.request.contextPath }/captcha.png" onclick="changeImg(this)">${msg2 }
              </div>
              <div class="col-sm-3">
                <input type="text" class="form-control"  name="cap">
              </div>
            </div>
            <div class="form-group">
              <div class="col-sm-offset-2 col-sm-10">
                <input type="submit"  width="100" value="Je m'inscris" name="submit" border="0"
                  style="background: url('${pageContext.request.contextPath}/images/register.gif') no-repeat scroll 0 0 rgba(0, 0, 0, 0);
                  height:35px;width:120px;color:white;">
              </div>
            </div>
          </form>
```

En utilisant bootstrap et jquery, on a obtenu la page d'inscription pour les clients qui correspendant aux bases de données(register.jsp).
Comme ca, on peut lancer le projet. D'après sauter automatique à la page d'accueil, on peux cliquer sur 'Je m'inscris' et sauter à la page d'inscription:

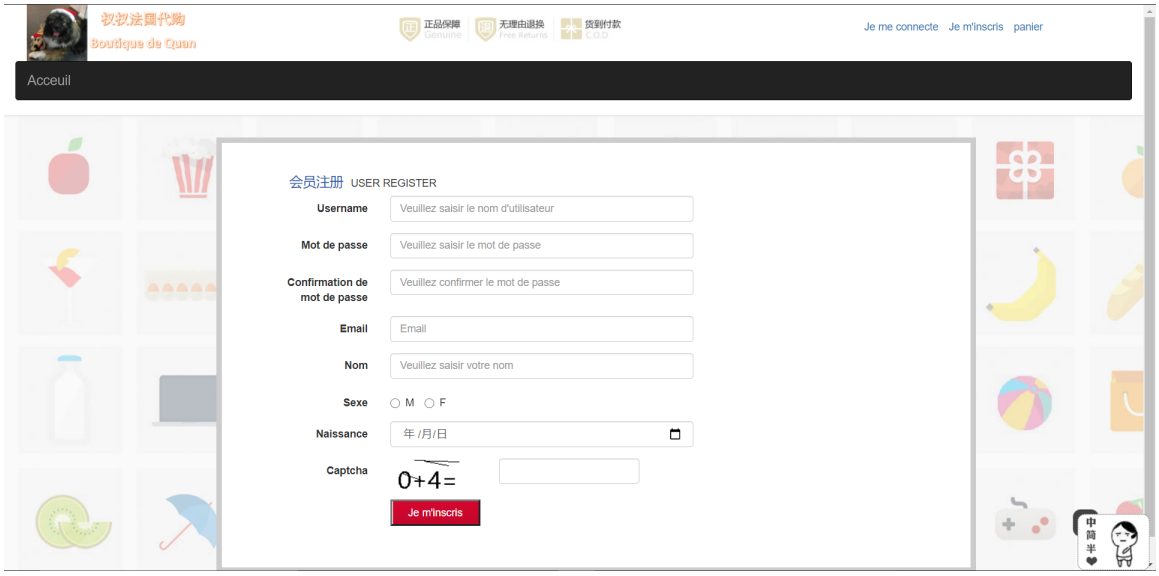


Fig. 9: La page d'inscription

5.2.3 La methode 'regist'

Creation de mothode 'registUI' dans UserServlet.java:

```
public String registUI(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
return "/jsp/register.jsp";
}
```

Ensuite, on cree la methode 'regist' dans UserServlet.java. Il y a des erreurs qu'on doit prévenir au début:

- L'objet user est vide
- La confirmation du mot de passe a échoué
- Erreur de code de Captcha
- Username existe déjà

```java
public String regist(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
  try {
    //1. Encapsulation d'objet
        User user = new User();
        BeanUtils.populate(user, request.getParameterMap());

        //1.1 Encapsulation uid
        user.setUid(UUIDUtils.getId());

        //1.2 Encapsulation state
        user.setState(Constant.USER_IS_NOT_ACTIVE);

        //1.3 Encapsulation la code d'activiation
        user.setCode(UUIDUtils.getCode());

        //2. Utiliser service pour finir l'inscription
        UserService us = new UserServiceImpl();

        // pour evider le double saisir
        UserDao ud = new UserDaoImpl();
        User usere = ud.getByUsername(user.getUsername());

        if(usere == null) {
        String cap = request.getParameter("cap");
        Integer res = (Integer)request.getSession().getAttribute("res");
        if(Integer.parseInt(cap) == res) {
            String conf = request.getParameter("conf");
            String passw =  request.getParameter("password");
            if(passw.equals(conf)) {
                us.regist(user);
                //3. Sauter et afficher les infors
                request.setAttribute("msg", "Felicitation, succès de l'inscription. Veuillez vous connecter votre mail pour terminer l'activation.");
            }else
            {
                request.setAttribute("msg1", "Veuillez conserver le même mot de passe.");
                return "/jsp/register.jsp";
            }
```

```java
        } else {
            request.setAttribute("msg2", "Captcha est incorrect.");
            return "/jsp/register.jsp";
        }

        }else {
            request.setAttribute("msg", "Vous vous avez deja inscrit, merci de l'activer ou bien re-inscrire.");
            return "/jsp/register.jsp";
        }
    } catch (Exception e) {
        e.printStackTrace();
        //sauter a msg.jsp
        request.setAttribute("msg", "L'inscription de l'utilisateur a échoué.");
        return "/jsp/msg.jsp";
    }

    return "/jsp/msg.jsp";
}
```

Pour générer Id et Code d'activiation aléatoirement, on doit creer les methodes dans UUIDUtils.java:

```java
public static String getId(){
return UUID.randomUUID().toString().replace("-", "").toUpperCase();
}

public static String getCode(){
return getId();
}

public static void main(String[] args) {
System.out.println(getId());
}
```

Ensuite, on configure Service. Creer la methode 'regist'

```java
public class UserServiceImpl implements UserService {

```

```
public void regist(User user) throws Exception {
//1. Enregistrement complet
UserDao ud = new UserDaoImpl();
ud.save(user);

//2.Envoyer un mail pour activer le compte
String emailMsg = "Felicitation"+user.getName()+":succes de inscription ,
<a href='http://localhost:8090/shop/user?method=active&code="
+user.getCode()+"'>
Cliquer ici pour activer votre compte.</a>";
MailUtils.sendMail(user.getEmail(), emailMsg);


}
```

5.3 Authentification par courriel

La realisation d'envoyer un mail automatique: Grace a qqmail (un type de mail tres connu en Chine, fini par @qq.com), on ouvre la service 'SMTP' et creer MailUtils.java pour realiser cette fonction.
Lorsque le nombre d'utilisateurs est très important, nous pouvons réduire la pression sur la base de données en supprimant périodiquement les utilisateurs inactifs. Pour le faire, on definit 2 constants dans Constant.java pour la realisation d'activiation:

```
int USER_IS_NOT_ACTIVE = 0;
int USER_IS_ACTIVE = 1;
```

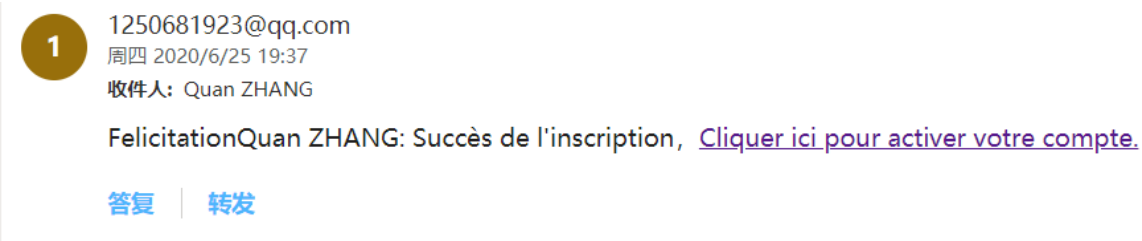
MailUtils.java:

```
public class MailUtils {

    public static void sendMail(String email, String emailMsg)
            throws AddressException, MessagingException {
        // 1.Créer un objet de session entre le code et le serveur de messagerie

        Properties props = new Properties();
        //le protocole d'envoi
        props.setProperty("mail.transport.protocol", "SMTP");

        //Configurer le serveur pour l'envoi de courrier
        props.setProperty("mail.host", "smtp.qq.com");
        props.setProperty("mail.smtp.auth", "true");

        // Créer un validateur
        Authenticator auth = new Authenticator() {
            public PasswordAuthentication getPasswordAuthentication() {
                //le mot de passe du compte de l'expéditeur
                return new PasswordAuthentication("1250681923@qq.com", "clvkajsyzqknjcba");
            }
        };
        Session session = Session.getInstance(props, auth);
```

```java
        // 2.Créer un message, qui est le contenu de l'e-mail
        Message message = new MimeMessage(session);

        //l'expéditeur
        message.setFrom(new InternetAddress("1250681923@qq.com"));

        //la méthode d'envoi et le destinataire
        message.setRecipient(RecipientType.TO, new InternetAddress(email));

        //le sujet
        message.setSubject("Activiation");

        //contenu
        message.setContent(emailMsg, "text/html;charset=utf-8");

        // 3.envoyer
        Transport.send(message);
    }
}
```

5.4 Activiation

Quand les utilisateurs se sont bien inscrit. Ils pouvent recevoir un mail pour activer le compte: (Comme 4.2, url de cette phrase permet de retour a la methode 'active')



Afin de realiser cette fonctionnalite, on doit creer une methode 'active' dans UserServlet.java:

```java
public String active(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        //1.La paramètre code
        String code = request.getParameter("code");

        //2.Finir activer en utilisant service, valeur de retour: user
        UserService us = new UserServiceImpl();
        User user = us.active(code);

        //3.Jugement
        if(user == null) {
            //échoué
            request.setAttribute("msg", "L'activation a échoué, veuillez réactiver ou vous réinscrire~");
            return "/jsp/msg.jsp";
        }
        //reussi
        request.setAttribute("msg", "Félicitations, l'activation est réussie, vous pouvez vous connecter~");
    } catch (Exception e) {
        request.setAttribute("msg", "L'activation a échoué, veuillez réactiver ou vous réinscrire~");
        return "/jsp/msg.jsp";
    }
    return "/jsp/msg.jsp";
}
```

UserServiceImpl.java:

```java
    public User active(String code) throws Exception {
        UserDao ud = (UserDao) BeanFactory.getBean("UserDao");
        //1.La paramètre code
        User user = ud.getByCode(code);

        //1.1 Aucun utilisateur trouvé par code d'activation
        if(user == null) {
            return null;
        }
        //2.Changer la state
        user.setState(Constant.USER_IS_ACTIVE);
        user.setCode(null);

        ud.update(user);
        return user;
    }
```

5.5 Login

5.5.1 JSP

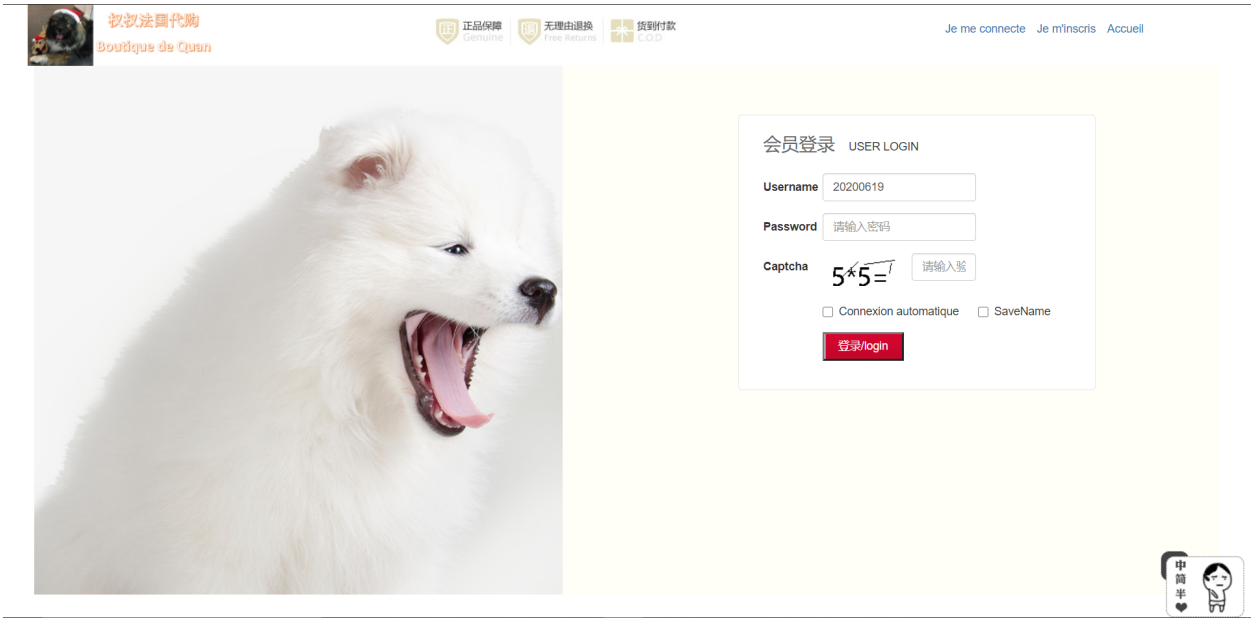
Dans cette étape, nous devons implémenter une fonction: mémoriser le nom d'utilisateur. Par conséquent, nous devons utiliser des cookies.

```jsp
<div class="container"  style="width:95%;height:650px;background:#fffff7 url('${pageContext.request.contextPath}/images/denglu.jpg') no-repeat;">
<div class="row">
    <div class="col-md-7">
        <!--<img src="${pageContext.request.contextPath}/image/login.jpg" width="500" height="330" alt="会员登录" title="会员登录">-->
    </div>

    <div class="col-md-5">
        <div style="width:440px;border:1px solid #E7E7E7;padding:20px 0 20px 30px;border-radius:5px;margin-top:60px;background:#fff;">
        <font>会员登录</font>USER LOGIN ${msg }

        <div> </div>
<form class="form-horizontal" action="${pageContext.request.contextPath }/user" method="post">
    <input type="hidden" name="method" value="login">

 <div class="form-group">
    <label for="username" class="col-sm-2 control-label">Username</label>
    <div class="col-sm-6">
      <input type="text" class="form-control" id="username" placeholder="请输入用户名" name="username">
    </div>
  </div>
   <div class="form-group">
    <label for="inputPassword3" class="col-sm-2 control-label">Password</label>
    <div class="col-sm-6">
      <input type="password" class="form-control" id="inputPassword3" placeholder="请输入密码" name="password">
    </div>
  </div>
   <div class="form-group">
        <label for="cap" class="col-sm-2 control-label">Captcha</label>
     <div class="col-sm-3">
      <img src="${pageContext.request.contextPath }/captcha.png" onclick="changeImg(this)">
    </div>
    <div class="col-sm-3">
      <input type="text" class="form-control" id="cap" placeholder="请输入验证码" name="cap">
    </div>
```

```
  </div>
   <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Connexion automatique
        </label>     
        <label>
          <input type="checkbox" name="savename" value="ok"> SaveName
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
    <input type="submit"  width="100" value="登录/login" name="submit" border="0"
    style="background: url('${pageContext.request.contextPath}/images/login.gif') no-repeat scroll 0 0 rgba(0, 0, 0, 0);
    height:35px;width:100px;color:white;">
    </div>
  </div>
</form>
</div>
    </div>
</div>
</div>
```

```
<script type="text/javascript">
          var val = "${cookie.saveName.value}";
          document.getElementById("username").value=decodeURI(val);
</script>
```

On peut lancer ce projet et aller a la page de login.



5.5.2 La methode login

Ensuite, on crée la methode 'login' dans UserServlet.java. Il y a des erreurs qu'on doit prévenir au début:

- Nom d'utilisateur et mot de passe ne correspondent pas
- S'il est activé
- Erreur de code de Captcha

- Déterminer s'il faut se souvenir du nom d'utilisateur

```java
public String login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        //Obtenir les parametres
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        //Login par service  valeur retrour: user
        UserService us = new UserServiceImpl();
        User user  = us.login(username,password);

        //Jugements
        if(user == null) {
            //Nom d'utilisateur et mot de passe ne correspondent pas
            request.setAttribute("msg", "username ou password incorrect.");
            return "/jsp/login.jsp";
        }
        //Déterminer s'il faut activer
        if(Constant.USER_IS_ACTIVE != user.getState()) {
            //non activé
            request.setAttribute("msg", "Veuillez vous activer avant la connexion.");
            return "/jsp/msg.jsp";
        }
        //Captcha
          String cap = request.getParameter("cap");
          Integer res = (Integer)request.getSession().getAttribute("res");
        if(Integer.parseInt(cap) == res) {
            //  réussi
            request.getSession().removeAttribute("res");
            //Enregistrer le statut de connexion
            request.getSession().setAttribute("user", user);

        } else {
            request.setAttribute("msg", "Captcha est incorrect.");
            return "/jsp/login.jsp";
        }
```

```java
          //////Save name
          //Déterminer s'il faut se souvenir du nom d'utilisateur
          if(Constant.SAVE_NAME.equals(request.getParameter("savename"))) {
              Cookie c = new Cookie("saveName", URLEncoder.encode(username, "utf-8"));

              c.setMaxAge(Integer.MAX_VALUE);
              c.setPath(request.getContextPath()+"/");

              response.addCookie(c);
          }
          ///////////
          //index.jsp
          response.sendRedirect(request.getContextPath());
      } catch (Exception e) {

          e.printStackTrace();
          request.setAttribute("msg", "Échec de la connexion.");
          return "/jsp/msg.jsp";
      }

      return null;
  }
```

5.6 Exit

UserServlet.java:

```java
  public String logout(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
      request.getSession().invalidate();

      response.sendRedirect(request.getContextPath());
      return null;
  }
```

6 Affichage des produit et catégories

6.1 CategoryDaoImpl

Après la connexion à la base de données, nous créons les methodes pour les opérations.

```
public class CategoryDaoImpl implements CategoryDao {

    @Override
    /**
     * Rechercher toutes les catégories
     */
    public List<Category> findAll() throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from category";
        return qr.query(sql, new BeanListHandler<>(Category.class));
    }
    @Override
    /**
     * Ajouter les catégories
     */
    public void save(Category c) throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "insert into category values (?,?);";
        qr.update(sql, c.getCid(),c.getCname());
    }


}
```

6.2 ProduitDaoImpl

Après la connexion à la base de données, nous créons les methodes pour les opérations.

```java
public class ProductDaoImpl implements ProductDao {

    @Override
    /**
     * Le plus vendu
     */
    public List<Product> findHot() throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from product where is_hot = ? and pflag = ? order by pdate desc limit 9";
        return qr.query(sql, new BeanListHandler<>(Product.class), Constant.PRODUCT_IS_HOT,Constant.PRODUCT_IS_UP);
    }

    @Override
    /**
     * Nouveau produit
     */
    public List<Product> findNew() throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from product where pflag = ? order by pdate desc limit 9";
        return qr.query(sql, new BeanListHandler<>(Product.class),Constant.PRODUCT_IS_UP);
    }

    @Override
    /**
     * Chercher un produit par ID
     */
    public Product getById(String pid) throws Exception {
        QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
        String sql = "select * from product where pid = ? limit 1";
        return qr.query(sql, new BeanHandler<>(Product.class), pid);

    }
```

```java
/**
 * Interroger les données de la page actuelle
 */
public List<Product> findByPage(PageBean<Product> pb, String cid) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    String sql = "select * from product where cid = ? and pflag = ? order by pdate desc limit ?,?";
    return qr.query(sql, new BeanListHandler<>(Product.class), cid,Constant.PRODUCT_IS_UP,pb.getStartIndex(),pb.getPageSize());
}

@Override
/**
 * Obtenir le nombre total d'enregistrements
 */
public int getTotalRecord(String cid) throws Exception {
    return ((Long)new QueryRunner(DataSourceUtils.getDataSource()).query("select count(*) from product where cid = ? and pflag = ?", new ScalarHandler(
}

@Override
/**
 * Trouver les produits existes
 */
public List<Product> findAll() throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    String sql = "select * from product where pflag = ? order by pdate desc";
    return qr.query(sql, new BeanListHandler<>(Product.class), Constant.PRODUCT_IS_UP);
}
```

```java
/**
 * Enregistrer les produits
 */
public void save(Product p) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());

    String sql="insert into product values(?,?,?,?,?,?,?,?,?,?);";
    qr.update(sql, p.getPid(),p.getPname(),p.getMarket_price(),
            p.getShop_price(),p.getPimage(),p.getPdate(),
            p.getIs_hot(),p.getPdesc(),p.getPflag(),
            p.getCategory().getCid());
}

public void update(Product product) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());

    String sql="update product set pflag = ? where pid = ?";
    qr.update(sql,product.getPflag(),product.getPid());

}
```

6.3 Introduction de Redis

De nombreux facteurs limitent la vitesse de réponse des pages Web. Les requêtes de base de données fréquentes en font partie. Redis est actuellement une technologie de mise en cache privilégiée par les ingénieurs. Ses caractéristiques de persistance et de réponse rapide peuvent réduire la pression des requêtes répétées sur la base de données et améliorer la vitesse de réponse des pages Web.

Dans ce projet, nous utiliserons **Jedis** pour les opération sur **Redis**. D'abord, nous configurons JedisUtils.java:

```java
public class JedisUtils {
    //Créer un pool de connexions
    private static final JedisPoolConfig config;
    private static final JedisPool pool;

    static{
        config=new JedisPoolConfig();
        config.setMaxTotal(30);
        config.setMaxIdle(2);

        pool=new JedisPool(config, Constant.REDIS_HOST, Constant.REDIS_PORT);
    }


    //Obtenir la connexion
    public static Jedis getJedis(){
        return pool.getResource();
    }


    //Perde la connexion
    public static void closeJedis(Jedis j){
        if(j!=null){
            j.close();
        }
    }
}
```

Pour l'utiliser, on cree une methode dans CategoryServiceImpl.java (On va juger si'l y a des donnes dans le cache ou bien Jedis. S'il y en a pas, on doit les obtenir dans Mysql en utilisant la methode findAll().):

```java
public String findAllFromRedis() throws Exception {
    Jedis j =null;
    String value=null;
    try {
        //1.Obtenez les informations de catagory de redis
        try {
            //1.1 Connexion
            j = JedisUtils.getJedis();

            //1.2 Obtenir les donnees
            value = j.get(Constant.STORE_CATEGORY_LIST);

            //1.3 Si non vide, renvoyer les données
            if(value!=null){
                System.out.println("Il y a des données dans le cache");
                return value;
            }
        } catch (Exception e) {
        }
        //2 S'il y n'a pas de donnees dans Redis, on prend les donnees dans Mysql et le met dans Redis
        value = findAll();

        //3.Mettre les values dans Redis
        try {
            j.set(Constant.STORE_CATEGORY_LIST, value);
            System.out.println("Les données ont été placées dans le cache");
        } catch (Exception e) {
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw e;
    } finally {
        //Fermer Jedis
        JedisUtils.closeJedis(j);
    }
    return value;
}
```

6.4 Afficher des catégories

6.4.1 Introduction de Ajax

Ajax, ou "Javascript asynchrone et XML" (JavaScript asynchrone et XML), fait référence à une technologie de développement Web pour créer des applications Web interactives.
Grâce à un petit échange de données avec le serveur en arrière-plan, Ajax peut utiliser des pages Web pour réaliser des mises à jour asynchrones.
Cela signifie que certaines parties de la page Web peuvent être mises à jour sans recharger la page Web entière (pas de technologie de rafraîchissement). Si une page Web traditionnelle (sans Ajax) doit être mise à jour, la page Web entière doit être rechargée.
Dans ce projet, on utilise Ajax pour Afficher tous les catégories et le faire dans head.jsp.

```html
<script type="text/javascript">
    $(function(){
        //发送ajax 查询所有分类
        $.post("${pageContext.request.contextPath}/category",{"method":"findAll"},function(obj){
            //alert(obj);
            //遍历json列表,获取每一个分类,包装成li标签,插入到ul内部
            //$.each($(obj),function(){});
            $(obj).each(function(){
                //alert(this.cname);
                $("#c_ul").append("<li><a href='${pageContext.request.contextPath}/product?method=findByPage&pageNumber=1&cid="+this.cid+"'>"+this.cname+"<
            });
        },"json");
    })
</script>
```

Attention! Afin d'utiliser Ajax, on doit importer Jquery(Ce qui a été importé au début)

6.4.2 Configuration des methodes JSON : JsonUtils

Le format de données utilisé dans ce projet qui doit être converti en String est le suivant:

- Array
- List
- Map
- Objet
- XML

La configuration JsonUtils.java:

```
/**
 *
 *
 * @param array
 * @return
 */
public static String array2json(Object[] objects){

    JSONArray jsonArray = JSONArray.fromObject(objects);
    return jsonArray.toString();

}

/**
 *
 *
 * @param list
 * @return
 */
public static String list2json(List list){

    JSONArray jsonArray = JSONArray.fromObject(list);
    return jsonArray.toString();

}
```

```java
/**
 *
 *
 * @param map
 * @return
 */
public static String map2json(Map map){

    JSONObject jsonObject = JSONObject.fromObject(map);
    return jsonObject.toString();

}

/**
 *
 *
 * @param object
 * @return
 */
public static String object2json(Object object){

    JSONObject jsonObject = JSONObject.fromObject(object);
    return jsonObject.toString();

}
```

```java
/**
 *
 *
 * @param xml
 * @return
 */
public static String xml2json(String xml){

    JSONArray jsonArray = (JSONArray) new XMLSerializer().read(xml);
    return jsonArray.toString();

}

/**
  * Supprimez les champs indésirables (particulièrement adaptés pour supprimer les objets en cascade)
  *
  * @param excludes
  * @return
*/
public static JsonConfig configJson(String[] excludes) {
    JsonConfig jsonConfig = new JsonConfig();
    jsonConfig.setExcludes(excludes);
    jsonConfig.setIgnoreDefaultExcludes(true);
    jsonConfig.setCycleDetectionStrategy(CycleDetectionStrategy.LENIENT);
    return jsonConfig;
}
```

6.4.3 Réaliser un affichage catégories

On cree une methode 'findAll() ' dans CategoryServlet.java:

```
public String findAll(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        //0.Définir le code de réponse
        response.setContentType("text/html;charset=utf-8");

        //1.interroger toutes les catégories, renvoyer la valeur json string
        CategoryService cs = new CategoryServiceImpl();
        //Récupérer la liste de mysql
        String value = cs.findAll();

        //Obtenir la liste de redis
        //String value = cs.findAllFromRedis();
        //2.Réécriver la chaîne dans le navigateur
        response.getWriter().println(value);
    } catch (Exception e) {
    }
    return null;
}
```

CategoryServiceImpl :

```
@Override
/**
 * Rechercher toutes les catégories
 */
public String findAll() throws Exception {
    /*//1.调用dao 查询所有分类
    CategoryDao cd = new CategoryDaoImpl();
    List<Category> list = cd.findAll();*/

    List<Category> list=findList();

    //2.Convertir la liste en chaîne json
    if(list!=null && list.size()>0){
        return JsonUtil.list2json(list);
    }
    return null;
}
```

6.5 Affichage les produit les plus vendus ou Nouveaux

Configuration dans index.jsp

```
<jsp:forward page="index"></jsp:forward>
```

Configuration dans IndexServlet.java:

```java
public class IndexServlet extends BaseServlet {
    private static final long serialVersionUID = 1L;

    @Override
    public String index(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        try {
            //1.Chercher les nouveaux produits et produits les plus vendus
            ProductService ps = new ProductServiceImpl();
            List<Product> hotList=ps.findHot();
            List<Product> newList=ps.findNew();

            //2.Mettre les 2 listes
            request.setAttribute("hList", hotList);
            request.setAttribute("nList", newList);
        } catch (Exception e) {
        }

        return "/jsp/index.jsp";
    }
}
```

Mettre en œuvre deux méthodes dans ProduitServiceImpl.java:

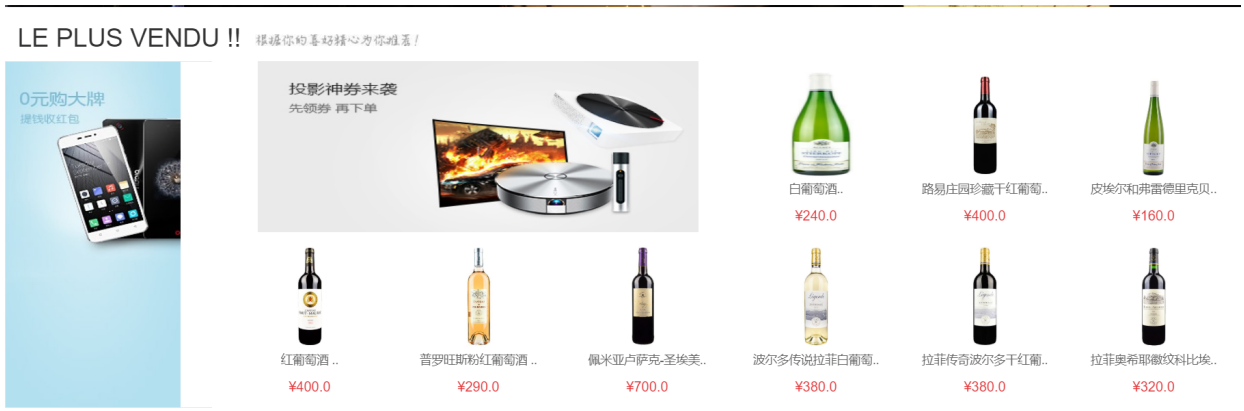
```java
/**
 * Trouver les produits les plus vendus
 */
public List<Product> findHot() throws Exception {
    ProductDao pd= (ProductDao) BeanFactory.getBean("ProductDao");
    return pd.findHot();
}

@Override
/**
 * Trouver les nouveaux produits
 */
public List<Product> findNew() throws Exception {
    //ProductDao pd= (ProductDao) BeanFactory.getBean("ProductDao");
    ProductDao pd= (ProductDao) BeanFactory.getBean("ProductDao");
    return pd.findNew();
}
```

Ajouter 'taglib 'fn" pour afficher les produits dans index.jsp:

```jsp
<%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
```

```
<div class="container-fluid">
    <div class="col-md-12">
        <h2>LE PLUS VENDU !!  <img src="${pageContext.request.contextPath}/img/title2.jpg"/></h2>
    </div>
    <div class="col-md-2" style="border:1px solid #E7E7E7;border-right:0;padding:0;">
        <img src="${pageContext.request.contextPath}/products/hao/big01.jpg" width="205" height="404" style="display: inline-block;"/>
    </div>
    <div class="col-md-10">
        <div class="col-md-6" style="text-align:center;height:200px;padding:0px;">
            <a href="product_info.htm">
                <img src="${pageContext.request.contextPath}/products/hao/middle01.jpg" width="516px" height="200px" style="display: inline-block;">
            </a>
        </div>
        <c:forEach items="${hList }" var="p">
            <div class="col-md-2" style="text-align:center;height:200px;padding:10px 0px;">
                <a href="${pageContext.request.contextPath }/product?method=getById&pid=${p.pid}">
                    <img src="${pageContext.request.contextPath}/${p.pimage}" width="130" height="130" style="display: inline-block;">
                </a>
                <p><a href="${pageContext.request.contextPath }/product?method=getById&pid=${p.pid}" style='color:#666'>${fn:substring(p.pname,0,10) }..
                <p><font color="#E4393C" style="font-size:16px">&yen;${p.shop_price }</font></p>
            </div>
        </c:forEach>


    </div>
</div>
```

Afficher le résultat front-end:



6.6 Affichage des détails du produit

Comme indiqué précédemment, nous cliquons sur l'image du produit dans index.jsp, puis nous passerons à la page d'informations sur le produit.
Pour le faire, on cree une methode getById() dans ProductServlet.java:

```java
/**
 * Info de produit
 * @param request
 * @param response
 * @return
 * @throws ServletException
 * @throws IOException
 */
public String getById(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        //1.Obtenir pid
        String pid = request.getParameter("pid");

        //2.Paramètre :pid     valeur retrour:product
        ProductService ps =new ProductServiceImpl();
        Product pro=ps.getById(pid);

        //3.Aller a la page  /jsp/product_info.jsp
        request.setAttribute("bean", pro);
    } catch (Exception e) {
        request.setAttribute("msg", "Délai d'expiration du service, veuillez mettre à jour");
        return "/jsp/msg.jsp";
    }

    return "/jsp/product_info.jsp";
}
```

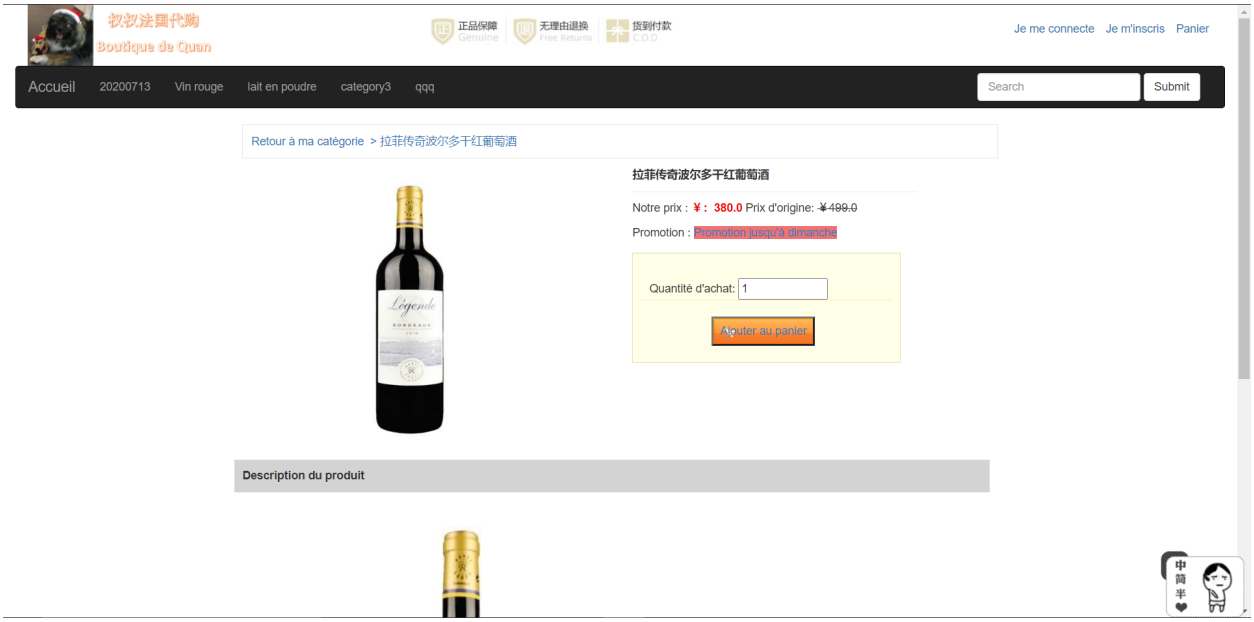
et dans ProductServiceImpl.java:

```java
public Product getById(String pid) throws Exception {
    ProductDao pd=(ProductDao) BeanFactory.getBean("ProductDao");

    return pd.getById(pid);
}
```

et dans product_info.jsp, on va afficher tous les infos de Attribute 'bean':

```jsp
<div style="margin:0 auto;width:950px;">
    <div class="col-md-6">
        <img style="opacity: 1;width:400px;height:350px;" title="" class="medium" src="${pageContext.request.contextPath}/${bean.pimage}">
    </div>

    <div class="col-md-6">
        <div><strong>${bean.pname }</strong></div>
        <div style="border-bottom: 1px dotted #dddddd;width:350px;margin:10px 0 10px 0;">
            <div></div>
        </div>

        <div style="margin:10px 0 10px 0;">Notre prix : <strong style="color:#ef0101;">￥ ${bean.shop_price }</strong> Prix d'origine: <del>￥${bea
        </div>

        <div style="margin:10px 0 10px 0;">Promotion : <a target="_blank" title="De 01/01 ~ 07/01" style="background-color: #f07373;">Promotion jus

        <div style="padding:10px;border:1px solid #e7dbb1;width:330px;margin:15px 0 10px 0;;background-color: #fffee6;">
            <div style="margin:5px 0 10px 0;"></div>


            <form action="${pageContext.request.contextPath }/cart" id="form1" method="get">
                <!-- 提交的方法 -->
                <input type="hidden" name="method" value="add2cart">

                <!-- 商品的pid -->
                <input type="hidden" name="pid" value="${bean.pid }">

                <div style="border-bottom: 1px solid #faeac7;margin-top:20px;padding-left: 10px;">Quantité d'achat:
                    <input id="quantity" name="count" value="1" maxlength="4" size="10" type="text"> </div>

                <div style="margin:20px 0 10px 0;;text-align: center;">
            </form>
                <a href="javascript:void(0)" onclick="subForm()">
                    <input style="background: url('${pageContext.request.contextPath}/images/product.gif') no-repeat scroll 0 -600px rgba(0, 0, 0,
                </a>  </div>
        </div>
    </div>
```

Afficher le résultat front-end:



6.7 Affichage les produits par catégories

Comme 5.4.1, nous avons ajouté un url vers le produit en ajax. On cree une methode 'findByPage()' dans ProductServlet.java

```
public String findByPage(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    try {
        //1.Obtenir les paramètres    Définir la taille de la page
        /*String parameter = request.getParameter("pageNumber");*/
        int pageNumber = 1;

        try {
            pageNumber = Integer.parseInt(request.getParameter("pageNumber"));
        } catch (NumberFormatException e) {
        }

        int pageSize = 12;
        String cid = request.getParameter("cid");

        //2. Trouver les produits par page   valeur retour:pagebean
        ProductService ps = new ProductServiceImpl();
        PageBean<Product> bean=ps.findByPage(pageNumber,pageSize,cid);

        //3.le transfert de la demande  product_list.jsp
        request.setAttribute("pb", bean);
    } catch (Exception e) {
        request.setAttribute("msg", "Délai d'expiration du service, veuillez mettre à jour");
        return "/jsp/msg.jsp";
    }
    return "/jsp/product_list.jsp";
}
```

Dans ProductServiceImpl.java:

```java
public PageBean<Product> findByPage(int pageNumber, int pageSize, String cid) throws Exception {
    ProductDao pDao= (ProductDao) BeanFactory.getBean("ProductDao");
    //1.Creer pagebean
    PageBean<Product> pb = new PageBean<>(pageNumber, pageSize);

    //2.Définir les données de la page actuelle
    List<Product> data = pDao.findByPage(pb,cid);
    pb.setData(data);

    //3.Définir le nombre total d'enregistrements
    int totalRecord = pDao.getTotalRecord(cid);
    pb.setTotalRecord(totalRecord);

    return pb;
}
```

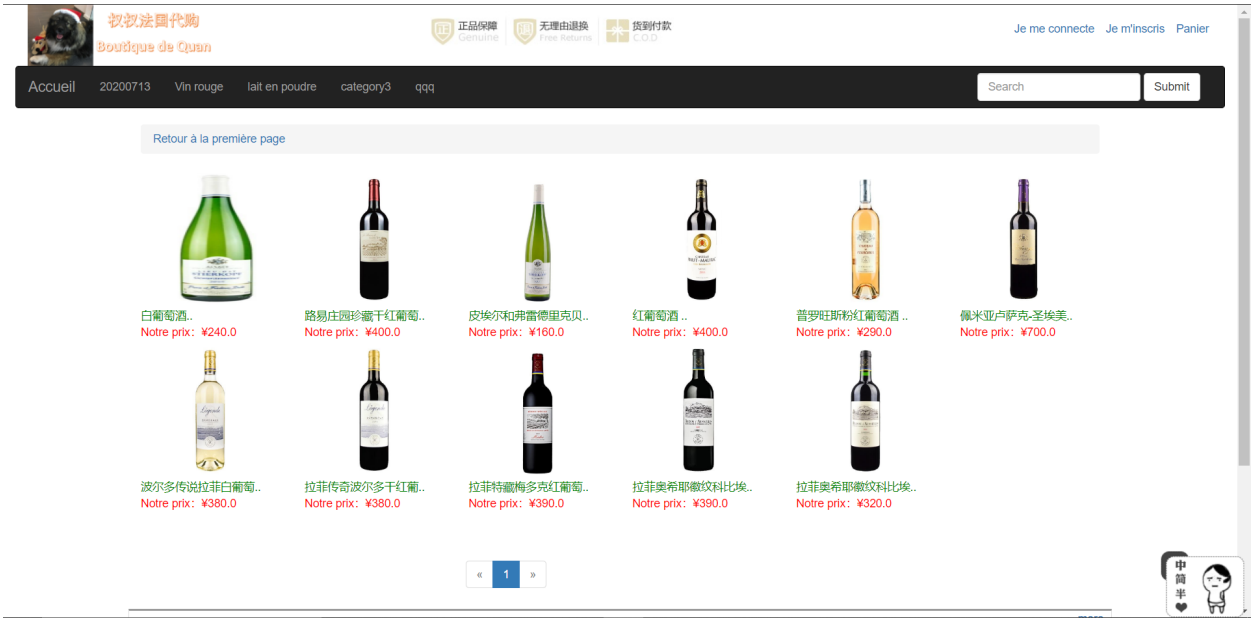
Implémentation front-end (dans product_list.jsp) en utilisant taglib 'fn':

```html
<div class="row" style="width:1210px;margin:0 auto;">
    <div class="col-md-12">
        <ol class="breadcrumb">
            <li><a href="${pageContext.request.contextPath }/product?method=findByPage&pageNumber=${1}&cid=${param.cid}">Retour à la première page</a><
        </ol>
    </div>
    <c:forEach items="${pb.data }" var="p">
        <div class="col-md-2">
            <a href="${pageContext.request.contextPath }/product?method=getById&pid=${p.pid}">
                <img src="${pageContext.request.contextPath}/${p.pimage}" width="170" height="170" style="display: inline-block;">
            </a>
            <p><a href="${pageContext.request.contextPath }/product?method=getById&pid=${p.pid}" style='color:green'>${fn:substring(p.pname,0,10) }..</
            <p><font color="#FF0000">Notre prix: &yen;${p.shop_price }</font></p>
        </div>
    </c:forEach>

</div>
```

Mise en œuvre de la division des numéros de page:

```html
<div style="width:380px;margin:0 auto;margin-top:50px;">
    <ul class="pagination" style="text-align:center; margin-top:10px;">

        <!-- 判断是否是第一页 -->
        <c:if test="${pb.pageNumber == 1 }">
            <li class="disabled">
                <a href="javascript:void(0)" aria-label="Previous">
                    <span aria-hidden="true">&laquo;</span>
                </a>
            </li>
        </c:if>

        <!-- 不是第一页 -->
        <c:if test="${pb.pageNumber != 1 }">
            <li>
                <a href="${pageContext.request.contextPath }/product?method=findByPage&pageNumber=${pb.pageNumber-1}&cid=${param.cid}" aria-label="Previ
                    <span aria-hidden="true">&laquo;</span>
                </a>
            </li>
        </c:if>

        <!-- 展示所有页码 -->
        <c:forEach begin="1" end="${pb.totalPage }" var = "n">


        <!--跳转功能-->
        <!-- 判断是否是当前页 -->
            <c:if test="${pb.pageNumber == n }">
                <li class="active"><a href="javascript:void(0)">${n }</a></li>
            </c:if>
            <c:if test="${pb.pageNumber != n }">
                <li><a href="${pageContext.request.contextPath }/product?method=findByPage&cid=${param.cid}&pageNumber=${n}">${n }</a></li>
            </c:if>
        </c:forEach>

        <!-- 判断是否是最后一页 -->
        <c:if test="${pb.pageNumber == pb.totalPage }">
            <li  class="disabled">
```

Afficher le résultat front-end:


权权法国代购
Boutique de Quan
正品保障 Genuine
无理由退换 Free Returns
货到付款 C.O.D
Je me connecte
Je m'inscris
Panier
Accueil
20200713
Vin rouge
lait en poudre
category3
qqq
Search
Submit
Retour à la première page
白葡萄酒..
Notre prix: ¥240.0
路易庄园珍藏干红葡萄..
Notre prix: ¥400.0
皮埃尔和弗雷德里克贝..
Notre prix: ¥160.0
红葡萄酒 ..
Notre prix: ¥400.0
普罗旺斯粉红葡萄酒 ..
Notre prix: ¥290.0
佩米亚卢萨克-圣埃美..
Notre prix: ¥700.0
波尔多传说拉菲白葡萄..
Notre prix: ¥380.0
拉菲传奇波尔多干红葡..
Notre prix: ¥380.0
拉菲特藏梅多克红葡萄..
Notre prix: ¥390.0
拉菲奥希耶徽纹科比埃..
Notre prix: ¥390.0
拉菲奥希耶徽纹科比埃..
Notre prix: ¥320.0
«
1
»

7 Mise en œuvre du panier d'achat

7.1 OrderDaoImpl

Dans OrderDaoImpl.java, on va réaliser les fonctionnalités suivantes:

- Enregistrer orders

```
public void save(Order o) throws Exception {
    QueryRunner qr = new QueryRunner();

    String sql = "insert into orders values(?,?,?,?,?,?,?,?)";
    qr.update(DataSourceUtils.getConnection(), sql, o.getOid(),o.getOrdertime(),o.getTotal(),
            o.getState(),o.getAddress(),o.getName(),
            o.getTelephone(),o.getUser().getUid());
}
```

- Enregistrer orderitems

```
public void saveItem(OrderItem oi) throws Exception {
    QueryRunner qr = new QueryRunner();

    String sql = "insert into orderitem values(?,?,?,?,?)";
    qr.update(DataSourceUtils.getConnection(), sql, oi.getItemid(),oi.getCount(),oi.getSubtotal(),
            oi.getProduct().getPid(),oi.getOrder().getOid());
}
```

- Obtenir le nombre total de mes commandes

```
public int getTotalRecord(String uid) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    String sql = "select count(*) from orders where uid = ?";
    return ((Long)qr.query(sql, new ScalarHandler(), uid)).intValue();
}
```

- Obtenir ma commande (Données de la page actuelle)

```java
public List<Order> findMyOrdersByPage(PageBean<Order> pb, String uid) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    //查询所有订单(基本信息)
    String sql="select * from orders where uid = ? order by ordertime desc limit ?,?";
    List<Order> list = qr.query(sql, new BeanListHandler<>(Order.class), uid,pb.getStartIndex(),pb.getPageSize());

    //遍历订单集合 获取每一个订单,查询每个订单订单项
    for (Order order : list) {
        sql="SELECT * FROM orderitem oi,product p WHERE oi.pid = p.pid AND oi.oid = ?";
        List<Map<String, Object>> maplist = qr.query(sql, new MapListHandler(), order.getOid());

        //遍历maplist 获取每一个订单项详情,封装成orderitem,将其加入当前订单的订单项列表中
        for (Map<String, Object> map : maplist) {
            //1.封装成orderitem
            //a.创建orderitem
            OrderItem oi = new OrderItem();

            //b.封装orderitem
            BeanUtils.populate(oi, map);

            //c.手动封装product
            Product p = new Product();

            BeanUtils.populate(p, map);

            oi.setProduct(p);

            //2.将orderitem放入order的订单项列表
            order.getItems().add(oi);
        }
    }
    return list;
}
```

- Obtenir détails de la commande par ID

```java
public Order getById(String oid) throws Exception {
    //1.查询订单基本信息
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    String sql ="select * from orders where oid = ?";
    Order order = qr.query(sql, new BeanHandler<>(Order.class), oid);

    //2.查询订单项
    sql ="SELECT * FROM orderitem oi,product p WHERE oi.pid = p.pid AND oi.oid = ?";
    //所有的订单项详情
    List<Map<String, Object>> maplist = qr.query(sql, new MapListHandler(), oid);

    //遍历 获取每一个订单项详情 封装成orderitem 加入到当前订单的items中
    for (Map<String, Object> map : maplist) {
        //创建ordreitem
        OrderItem oi = new OrderItem();

        //封装
        BeanUtils.populate(oi, map);

        //手动封装product
        Product p = new Product();
        BeanUtils.populate(p, map);

        oi.setProduct(p);
        //将orderitem加入到订单的items中
        order.getItems().add(oi);
    }
    return order;
}
```

- Update

```
public void update(Order order) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    /**
     * `oid` varchar(32) NOT NULL,
          `ordertime` datetime DEFAULT NULL,
         `total` double DEFAULT NULL,

         `state` int(11) DEFAULT NULL,
         `address` varchar(30) DEFAULT NULL,
         `name` varchar(20) DEFAULT NULL,

         `telephone` varchar(20) DEFAULT NULL,
         `uid` varchar(32) DEFAULT NULL,
     */
    String sql="update orders set state = ?,address = ?,name =?,telephone = ? where oid = ?";
    qr.update(sql,order.getState(),order.getAddress(),order.getName(),
            order.getTelephone(),order.getOid());

}
```

- Obtenir la liste de commande en arrière-plan

```
public List<Order> findAllByState(String state) throws Exception {
    QueryRunner qr = new QueryRunner(DataSourceUtils.getDataSource());
    String sql = "select * from orders ";

    //判断state是否为空
    if(null==state || state.trim().length()==0){
        sql +=" order by ordertime desc";
        return qr.query(sql, new BeanListHandler<>(Order.class));
    }

    sql += " where state = ? order by ordertime desc";
    return qr.query(sql, new BeanListHandler<>(Order.class),state);
}
```

7.2 Réalisation d'entité Panier et Articles du panier

On crée 2 class s'appellent Cart.java et CartItem.java:

- Dans CartItem.java, on met les infos de produits: produit, prix total et quantite:

```
public class CartItem {
    //Produit
    private Product product;

    //Subtotal
    private Double subtotal;

    //quantite
    private Integer count;
```

- Dans Cart.java, on met un Hashmap pour enregistrer les produits:

```java
public class Cart {
    private Map<String, CartItem> itemMap=new HashMap<String, CartItem>();
    private Double total=0.0;
```

- Dans Cart.java, on ajoute aussi les methodes:

```java
    /**
     * Ajouter au panier
     * @param item
     */
    public void add2cart(CartItem item){
        //Obtenir id
        String pid = item.getProduct().getPid();

        //Déterminer s'il y en a dans le panier
        if(itemMap.containsKey(pid)){
            //oui, on change la quantité
            CartItem oItem = itemMap.get(pid);

            oItem.setCount(oItem.getCount()+item.getCount());
        }else{
            //non
            itemMap.put(pid, item);
        }

        //Modifier le montant total
        total += item.getSubtotal();
    }

    /**
     * Supprimer un produit du panier
     * @param pid
     */
    public void removeFromCart(String pid){
        //1.Supprimer le produit du panier
        CartItem item = itemMap.remove(pid);

        //2.Modifier le montant total
        total -= item.getSubtotal();
    }
```

```java
    /**
     * Vider panier
     */
    public void clearCart(){
        //1.Vider map
        itemMap.clear();

        //2.Modifier le montant total = 0
        total=0.0;
    }
```

7.3 Ajouter les produits

On configure dans CartServlet.java et cree methodes 'add2cart(request,response)' et 'getCart(request)':

```java
public String add2cart(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    try {
        //1.Parametre pid count
        String pid = request.getParameter("pid");
        int count = Integer.parseInt(request.getParameter("count"));


        // Obtenir product
        ProductService ps = (ProductService) BeanFactory.getBean("ProductService");
        Product product = ps.getById(pid);

        //Creer cartitem
        CartItem cartItem = new CartItem(product, count);

        //3 Mettre cartitem dans cart
        Cart cart=getCart(request);

        cart.add2cart(cartItem);

        response.sendRedirect(request.getContextPath()+"/jsp/cart.jsp");
    } catch (Exception e) {
        e.printStackTrace();
        request.setAttribute("msg", "ajouter au panier est échec");
        return "/jsp/msg.jsp";
    }

    return null;
}
```

```java
private Cart getCart(HttpServletRequest request) {
    Cart cart = (Cart) request.getSession().getAttribute("cart");
    if(cart == null){
        cart = new Cart();

        //mettre cart dans session
        request.getSession().setAttribute("cart", cart);
    }
    return cart;
}
```

Dans cart.jsp, on va afficher tous les items de Order:

```jsp
<c:forEach items="${cart.cartItems }" var="ci">
    <tr class="active">
        <td width="60" width="40%">
            <input type="hidden" name="id" value="22">
            <img src="${pageContext.request.contextPath}/${ci.product.pimage}" width="70" height="60">
        </td>
        <td width="30%">
            <a target="_blank">${ci.product.pname}</a>
        </td>
        <td width="20%">
            ￥${ci.product.shop_price}
        </td>
        <td width="10%">
            <input type="text" readonly="readonly" name="quantity" value="${ci.count }" maxlength="4" size="10">
        </td>
        <td width="15%">
            <span class="subtotal">￥${ci.subtotal }</span>
        </td>
        <td>
            <a href="javascript:void(0);" onclick="removeFromCart('${ci.product.pid}')" class="delete">Supprimer</a>
        </td>
    </tr>
</c:forEach>
```

Exemple:

Mon panier

Images	Products	Prix	Quantite	Total	Operation
	路易庄园珍藏干红葡萄酒	¥400.0	2	¥800.0	Supprimer

Montant total: ¥800.0

Vider mon panier Soumettre

7.4 Enlever les produits

Dans cart.jsp:

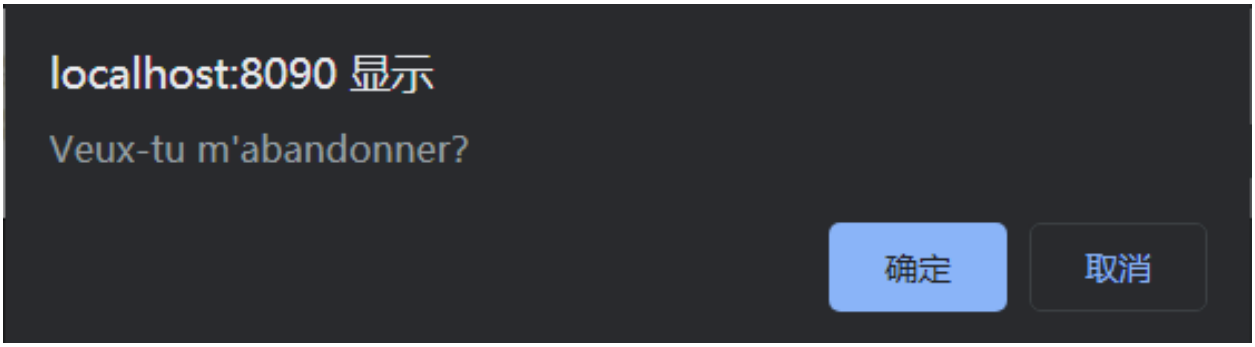
```
<td>
    <a href="javascript:void(0);" onclick="removeFromCart('${ci.product.pid}')" class="delete">Supprimer</a>
</td>
```

```
<script type="text/javascript">
    function removeFromCart(pid){
        if(confirm("Veux-tu m'abandonner？")){
            location.href="${pageContext.request.contextPath}/cart?method=remove&pid="+pid;
        }
    }
</script>
```

On configure la methode 'remove(request, response)' dans CartServlet.java:

```
public String remove(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //1.Obtenir pid
    String pid = request.getParameter("pid");

    //2.Supprimer
    getCart(request).removeFromCart(pid);

    //3.Retour
    response.sendRedirect(request.getContextPath()+"/jsp/cart.jsp");
    return null;
}
```

Quand on cliquer sur 'supprimer', on a :



7.5 Vider le panier

Configurer la méthode 'clear(request, response)' dans CartServlet.java:

```
public String clear(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    //1.Vider panier
    getCart(request).clearCart();

    //2.retour
    response.sendRedirect(request.getContextPath()+"/jsp/cart.jsp");

    return null;
}
```

7.6 Générer des commandes

On crée OrderServlet.java et une methode 'save(request, response)':

```
public String save(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    try {
        //-1.Obtenir user dans session()
        User user=(User) request.getSession().getAttribute("user");
        if(user == null){
            // faut login d'abord
            request.setAttribute("msg", "S'il vous plait Connectez-vous d'abord!");
            return "/jsp/msg.jsp";
        }

        //0.obtenir panier
        Cart cart=(Cart) request.getSession().getAttribute("cart");

        //1.Encapsulation
        //1.1Creer un order
        Order order = new Order();

        //1.2oid
        order.setOid(UUIDUtils.getId());

        //1.3ordertime

        SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        order.setOrdertime(df.format(new Date()));

        //1.4total
        order.setTotal(cart.getTotal());

        //1.5state
        order.setState(Constant.ORDER_WEIFUKUAN);

        //1.6user
        order.setUser(user);
```

```java
        //1.7 Parcourer la liste des articles d'achat
        for (CartItem   ci : cart.getCartItems()) {
            //a.Creer orderitem
            OrderItem oi = new OrderItem();

            //b.itemid uuid
            oi.setItemid(UUIDUtils.getId());

            //c.count
            oi.setCount(ci.getCount());

            //d.subtotal
            oi.setSubtotal(ci.getSubtotal());

            //e.product
            oi.setProduct(ci.getProduct());

            //f.order
            oi.setOrder(order);

            //1.7.2 Mettre orderitem dans order
            order.getItems().add(oi);
        }


        //2.Finir l'operation
        OrderService os = (OrderService) BeanFactory.getBean("OrderService");
        os.save(order);

        //3.Sauter a order_info.jsp
        request.setAttribute("bean", order);
    } catch (Exception e) {
    }
    return "/jsp/order_info.jsp";
}
```

Dans OrderServiceImpl.java:

```java
public void save(Order order) throws Exception{
    try {
        //dao
        OrderDao od  = (OrderDao) BeanFactory.getBean("OrderDao");
        //1.Lancer la transaction
        DataSourceUtils.startTransaction();

        //2.Enregistrer
        od.save(order);

        //3.Enregistrer plusieurs items
        for (OrderItem oi : order.getItems()) {
            od.saveItem(oi);
        }

        //4.Fermer
        DataSourceUtils.commitAndClose();
    } catch (Exception e) {
        e.printStackTrace();
        DataSourceUtils.rollbackAndClose();
        throw e;
    }
}
```

Dans order_info.jsp, on va afficher cet order:

```jsp
<c:forEach items="${bean.items }" var="oi">
    <tr class="active">
        <td width="60" width="40%">
            <input type="hidden" name="id" value="22">
            <img src="${pageContext.request.contextPath}/${oi.product.pimage}" width="70" height="60">
        </td>
        <td width="30%">
            <a target="_blank">${oi.product.pname}</a>
        </td>
        <td width="20%">
            ￥${oi.product.shop_price}
        </td>
        <td width="10%">
            ${oi.count}
        </td>
        <td width="15%">
            <span class="subtotal">￥${oi.subtotal}</span>
        </td>
    </tr>
</c:forEach>
```

Exemple:

Mes commandes

		A payer	Temps: 2020-07-06 06:15:54.0	
Images	**Produits**	**Prix**	**Quantite**	**Total**
	皮埃尔和弗雷德里克贝希特 阿尔萨斯 白葡萄酒	￥160.0	2	￥320.0
	红葡萄酒	￥400.0	1	￥400.0

Montant total: **￥720.0**

Adresse 请输入收货地址

Nom 请输收货人

Tel 请输入联系方式

确认订单

8 Payement

Pour des raisons de sécurité, nous n'implémentons temporairement pas la fonction de paiement. Dans ce cas, nous implémentons uniquement la fonction : "soumission de la commande et modification du statut de la commande".

8.1 La liste de commandes

Sur n'importe quelle page, cliquez sur <Mes commandes> pour afficher les commandes de l'utilisateur actuel (y compris la liste des produits de la commande).

Analyse technique:

- Ajouter le lien "Ma commande" sur head.jsp

$$/store/order?method = findMyOrdersByPage\&pageNumber = 1$$

- Créer findMyOrdersByPage dans orderservlet
  Obtenir le numéro de page et définir la taille de la page et obtenir l'ID utilisateur actuel
  Appeler le service pour terminer l'opération de requête de pagination: return pagebean
  Placer le pagebean dans le champ de requête et transférer la requête à order_list.jsp

- Dans service:
  Créer un pagebean
  Définir le nombre total
  Définir les données de la page actuelle

- Opération dans DAO:
  Obtenir les données de commande des pages actuelles
  sql: Seules les informations de base de la commande peuvent être interrogées, il n'y a pas d'article de commande

$$select * from\ orders\ where\ uid\ = ?\ order\ by\ ordertime\ desc\ limit?, ?$$

  Exécuter le sql ci-dessus, la liste ¡Order¿ renvoyée
  Parcourer la collection de la liste de commandes, obtenir chaque commande, interroger les produits de commande et la table de produits, et rechercher toutes les informations de la commande. Encapsuler-le dans orderItem et placer chaque article de commande dans la liste des produits de commande.

$$select * from\ orderitem\ oi, product\ p\ where\ oi.pid\ =\ p.pid\ and\ oi.oid\ =\ '...'$$

  Utiliser l'encapsulation maplisthandler pour encapsuler chaque enregistrement du résultat de la requête dans une carte.
  Placer toutes les cartes dans map et renvoyer
  Parcourer la liste dans map pour obtenir les détails de chaque commande, utiliser beanutils pour encapsuler orderitems et produit, et enfin placer orderitem dans la liste de commande:

$$[\{itemid = xxxx, pname = ..., ...\}, \{itemid = uyyyyx, pname = ..., ...\}]$$

8.2 Les infors des commandes

Cliquer sur <A payer> sur la page de la liste des commandes et afficher les détails de la commande en cours.

Analyse technique:

- Modifier le lien "A Payer"

$$/store/order?method = getById\&oid = xxxx$$

- Créer la méthode getById dans orderservlet
  Obtenir oid
  Appeler le service pour interroger une seule commande et Valeur de retour: commande
  Mettre ordre dans request et Aller à order_info.jsp

- Opération dans Service

- Opération dans Dao

$$select\ *\ from\ orders\ where\ oid\ = ?$$

$$select\ *\ from\ orderitem\ oi, product\ p\ where\ oi.pid\ =\ p.pid\ and\ oi.oid =?$$

- Utiliser maplisthandler pour l'encapsulation

		A payer	Montant total :720.0	
Images	Produits	Prix	Quantite	Total
	皮埃尔和弗雷德里克贝希特 阿尔萨斯 白葡萄酒	¥160.0	2	¥320.0
	红葡萄酒	¥400.0	1	¥400.0

8.3 Filtrage des autorisations

Lors de l'accès au panier, il est nécessaire de déterminer si l'utilisateur est connecté. Sinon, il vous demandera.

Analyse technique:

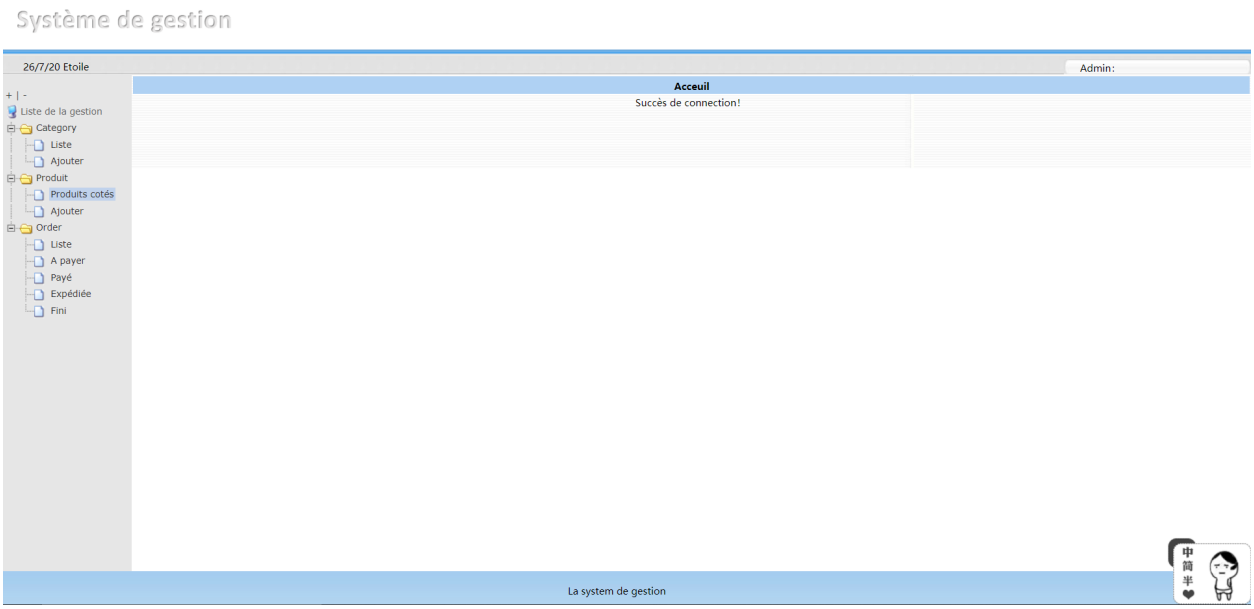
- Configuration du filtre:
  Créer une classe PrivilegeFilter
  Implémenter l'interface filtre
  Créer une methode doFilter

- fichier de configuration
  filter
  filter-mapping

- Obtenir des utilisateurs de la session
  Déterminer si l'utilisateur est vide S'il est vide: invite "Veuillez vous connecter", demande transmise à /jsp/msg.jsp

9 Système de gestion

Nous voulons créer un système de gestion pour aider à gérer les commandes, les catégories et les produits:



Pour le faire, on utilise dTree et le configure dans left.jsp:

```
<script type="text/javascript" src="${pageContext.request.contextPath}/js/dtree.js"></script>
<script type="text/javascript">

    d = new dTree('d');
    d.add('01',-1,'Liste de la gestion');
    d.add('0102','01','Category','','','mainFrame');
    d.add('010201','0102','Liste','${pageContext.request.contextPath}/adminCategory?method=findAll','','mainFrame');
    d.add('010202','0102','Ajouter','${pageContext.request.contextPath}/adminCategory?method=addUI','','mainFrame');
    d.add('0104','01','Produit');
    d.add('010401','0104','Produits cotés','${pageContext.request.contextPath}/adminProduct?method=findAll','','mainFrame');
    d.add('010402','0104','Ajouter','${pageContext.request.contextPath}/adminProduct?method=addUI','','mainFrame');
    d.add('0105','01','Order');
    d.add('010501','0105','Liste','${pageContext.request.contextPath}/adminOrder?method=findAllByState','','mainFrame');
    d.add('010502','0105','A payer','${pageContext.request.contextPath}/adminOrder?method=findAllByState&state=0','','mainFrame');
    d.add('010503','0105','Payé','${pageContext.request.contextPath}/adminOrder?method=findAllByState&state=1','','mainFrame');
    d.add('010504','0105','Expédiée','${pageContext.request.contextPath}/adminOrder?method=findAllByState&state=2','','mainFrame');
    d.add('010505','0105','Fini','${pageContext.request.contextPath}/adminOrder?method=findAllByState&state=3','','mainFrame');
    document.write(d);

</script>
```

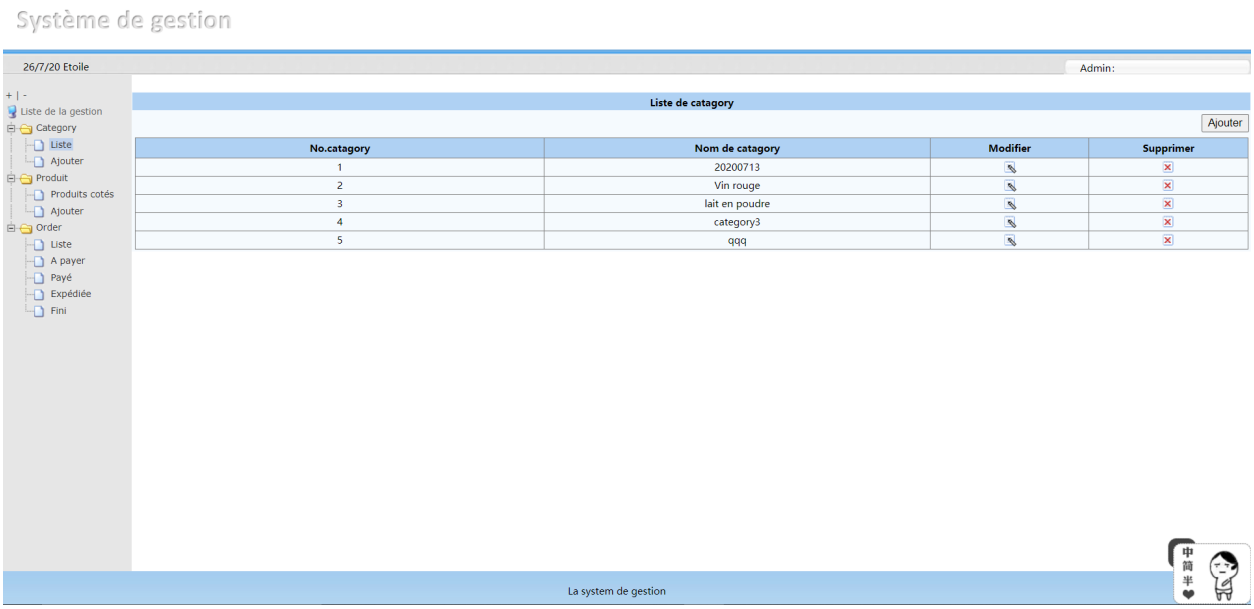
9.1 Liste de category

Ajouter un lien "Liste" à gauche, cliquer lequel il va afficher toutes les catégories sur la page principale à droite.

- Ajoutez un nœud parent de connexion "Category" sur left.jsp, ajouter un nœud enfant "Liste"

$$/store/adminCategory?method = findAll$$

- Configurer adminCategoryservlet qui hérite de baseservlet, Créer la méthode findAll
  Valeur retour: List<Category>
  Aller a:/*admin/category/list.jsp*



9.2 Ajouter les categories

Ajouter un lien "Ajouter" à gauche, cliquer lequel on peut ajouter les catégories sur la page principale à droite.

- Ajouter un lien "ajouter" dans gauche.jsp

$$/store/adminCategory?method = addUI$$

- Créer la méthode addUI dans admincategoryservlet

$$/admin/category/add.jsp$$

- Modifier add.jsp

$$action : /store/adminCategory?method = save$$

- Cliquer sur "confirmer" et envoyer une demande à adminCategory, methode :Save
  Obtenir cname, définir cid
  Encapsuler des objets de catégories
  Appeler le service pour terminer l'opération
  Rediriger vers la page de liste des catégories

- Service:
  Dao
  Mettre à jour les données dans redis

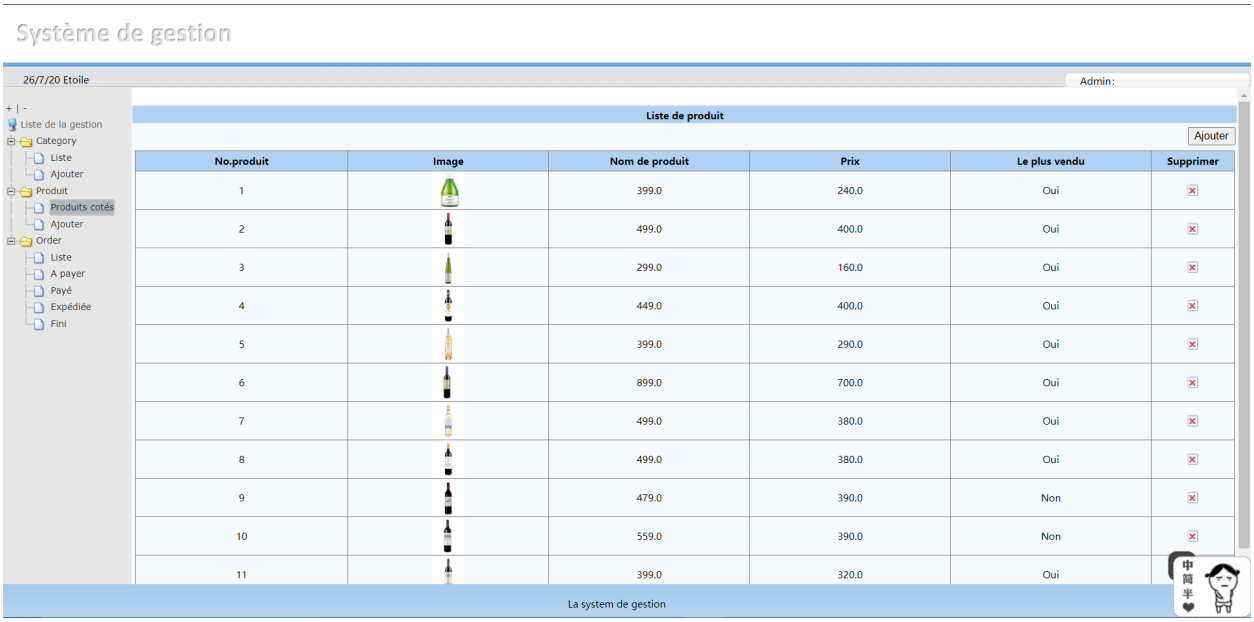
9.3 Afficher les produits

Cliquez sur un lien à gauche "Produits cotés" pour afficher tous les produits sur la page principale à droite.

- Ajoutez un nœud parent de connexion "Produits" sur left.jsp, ajouter un nœud enfant "Produits cotés"

$$/store/adminProduct?method = findAll$$

- Configurer adminCategoryservlet qui hérite de baseservlet, Créer la méthode findAll
  Valeur retour:list Aller a:$/admin/product/list.jsp$



9.4 Ajouter les produits

9.4.1 Upload/Transfert des photos

Une classe d'outils produite par apache pour gérer les téléchargements de fichiers:
Étapes d'utilisation:

- Importez le package jar

- Créer une fabrique d'éléments de fichier de disque:

$$DiskFileItemFactory\ factory = new\ DiskFileItemFactory();$$

- Créer un objet de téléchargement principal:

$$ServletFileUpload\ upload = new\ ServletFileUpload(factory);$$

- Analyser la demande et obtenir tous les composants de téléchargement List <FileItem>:

$$List < FileItem >\ \ list = upload.parseRequest(request);$$

- Parcourer la liste pour obtenir chaque élément téléchargé
  getFiledName(): Récupèrer la valeur de l'attribut "name" de balise
  isFormFiled (): Détermine s'il s'agit d'un composant de téléchargement normal
  true: composant de téléchargement normal
  false: composant de téléchargement de fichier <input type = "file">

- Composant de téléchargement normal
  getString ("utf-8"): Récupèrer le contenu saisi par l'utilisateur

- Composant de téléchargement de fichiers
  getName (): nom de fichier
  getInputStream (): Récupèrer le flux du fichier
  delete (): Supprimer le fichier temporaire généré lors du téléchargement du fichier

Remarque lors du téléchargement:

- Nom du fichier:
  Différents navigateurs peuvent obtenir des noms différents.
  Ex: 1.txt ou G:\1.txt

- Nom de fichier en double:
  Solution:
  - Donnez au fichier un nom aléatoire:
    Comme ce qu'on a fait pour uuid
  - Plusieurs champs peuvent être stockés dans la base de données:
    Nom du fichier et path du fichier
  - Télécharger le document:

$$response.setHeader("content - disposition","attachment; filename = " + realname);$$

- Sécurité des fichiers (fastDFS):
  Répertoire sécurité: un répertoire en dehors du projet web-inf et meta-inf

- Séparation des répertoires:
  Par utilisateur, Par temps, Par quantité ou bien Division aléatoire

Analyse des étapes:

- Ajouter un lien "ajouter" dans gauche.jsp sous "produit"

$$/store/adminProduct?method = addUI$$

- Créer la méthode addUI dans adminproductserlvet
  Appeler categoryservice pour interroger toutes les catégories: findList
  demande transmise a:
  $$/admin/category/add.jsp$$

- Modifiez la page du formulaire:
  action: /store/addProduct
  method:post
  entype:multipart/form-data
  Cliquer sur Enregistrer pour envoyer une demande à addProductServlet

- Créer addProductServlet:
  Encapsuler un objet de produit
  Enregistrez l'image du produit sur le serveur
  Avant, on utilisait $beanUtils.populate(Productbean, Mapmap)$;
  Mais maintenant $request.getParameterMap$ ne fonctionne pas.Si nous voulons utiliser la méthode ci-dessus pour encapsuler l'objet, nous devons créer une carte et stocker les informations de l'objet produit dans la carte.
  $Map < StringXXX, ObjectXXX > map$ Enregistrez l'image sur le serveur via commons-fileupload
  Mettez le chemin de l'image dans map
  Appeler le service pour enregistrer le produit
  Rediriger vers la page de liste de produits

- service dao

9.5 Interroger la liste des commandes par statut

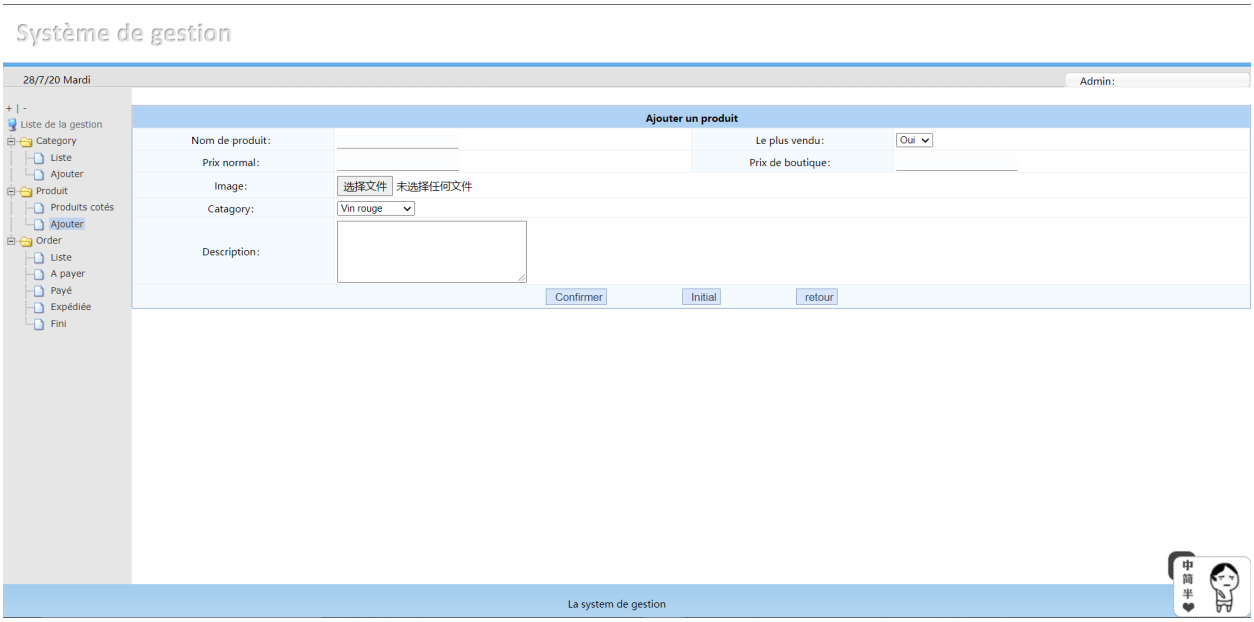
Il y a une gestion des commandes dans left.jsp qui a 5 connexions ci-dessous. Cliquer sur chaque connexion pour afficher toutes les commandes par l'état actuel. **Analyse des étapes:**

- Modifier left.jsp et ajouter 5 connexions

$$/store/adminOrder?method = findAllByState\&state = x$$

- Créer adminorderservlet qui hérite Baseservlet et créer la méthode findAllByState
  Obtenir State
  Appeler le service pour obtenir la liste
  La demande est transmise à $/admin/order/list.jsp$

- Dao



9.6 Interroger les détails de la commande

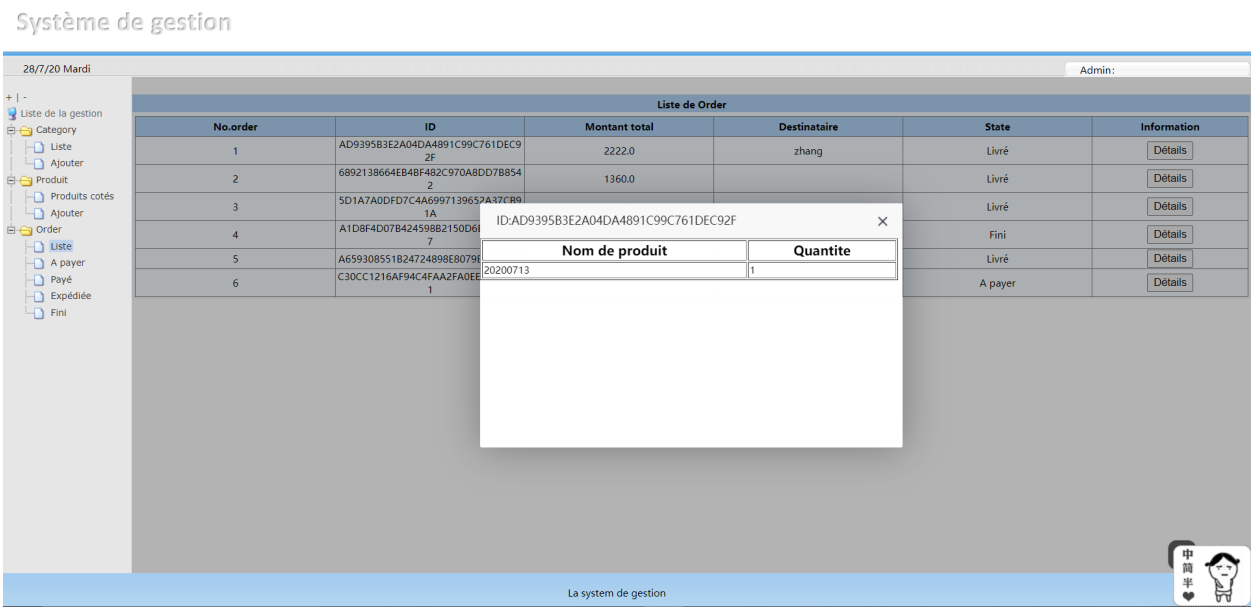
Cliquez sur "Détails" de chaque commande sur list.jsp pour afficher la liste de "Orderitems" de ce "Order".

Analyse technique:

On va utiliser Ajax et Layer. (https://layer.layui.com/)

Analyse des étapes:

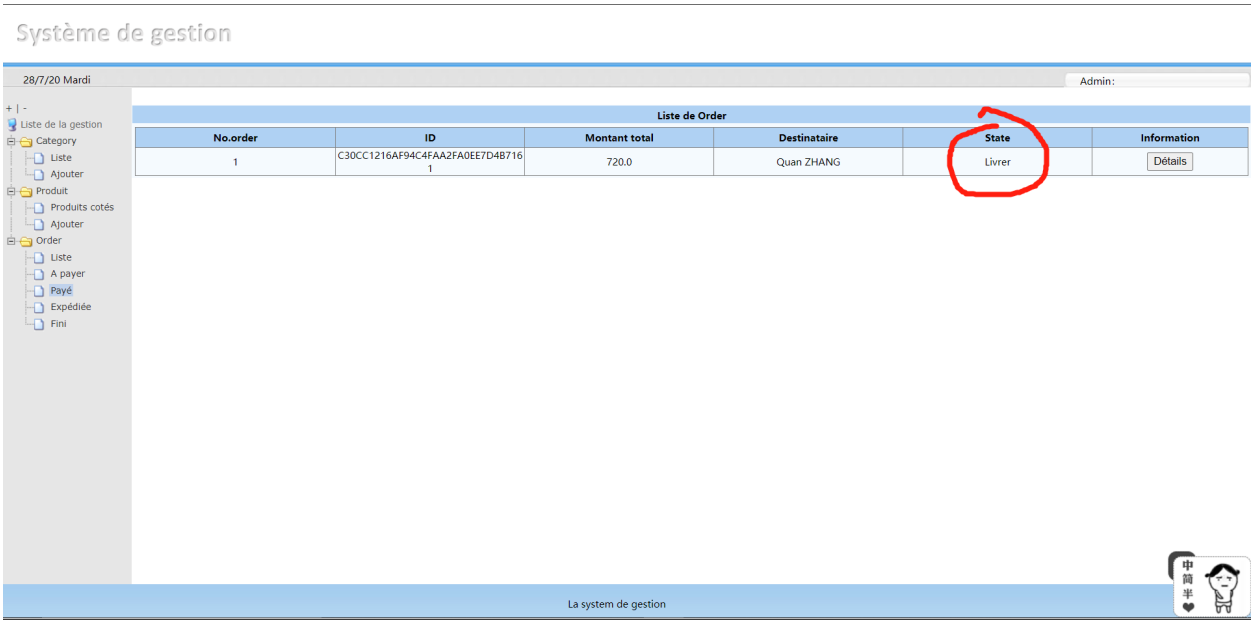
- Ajouter un événement de clic à "Détails" en obtenant Oid (On doit utiliser Layer)
- Créer une fonction:
  $.post(url,params,fn,type);
  $url : /store/adminOrder$
  $params : \{method : showDetail, oid : xxx\}$
  type:json
- Créer la méthode showDetail:
  Obtenir oid
  Utiliser getById() dans service. Obtenir order, Obtenir la liste de orderitems de order
  Convertir la liste en json et revenir au navigateur



9.7 Modifier le statut de la commande

Dans la liste des commandes qui sont payées, cliquer sur "Livrer" et modifier le statut de la commande.
Analyse des étapes:

- Modifier list.jsp et ajouter une connexion à "Livrer"

$$/store/adminOrder?method = updateState\&oid = xxx$$

- Créer la methode updateState dans adminorderservlet
  Obtenir oid
  Appeler le service pour obtenir la commande, modifier $state = 2$
  Rediriger vers la liste des commandes qui sont payées