Rapport de stage (première semaine)
MUNDUS LI YUANYUAN 03/07/2017 - 07/07/2017

Logiciel：eclipse + tomcat 7.0 + spring MVC

03/07/2017 - 04/07/2017 : étudier la structure de “Dao + Service + Controller”, programmer class - User

05/07/2017 : ajouter la deuxième classe - Book

06/07/2017 - 07/07/2017: ajouter deux classes: Demande, Delivery

Ⅰ、03/07/2017 - 0 4 /07/2017 : Class - User

Le premier jour, c'était un peu difficile pour moi， parce que je n'étudie pas JAVAWEB avant ce stage，presque，je ne comprends pas qu'est-ce que je dois faire．Donc，mon tuteur donne moi un programme comme un entraînement. Il configure les documents de 'config' pour connecter mysql，eclipse et serveur．

Mon travail est le programme de classe User（la table User）：ajouter un user à la table user，delete un user，update les informations de user，sélection les informations de user.

1.1 Les étapes：

1、programme test.xml pour mysql

2、programme UseDao，UserService，UserServiceImpl

3、run la classe Test comme junit test et run dans le serveur

1.2 explication de classe User

Il y a six attributs：userId，userName，password，entreprise，date de création，date de delete．User peut crée un compte avec les information personnelles，par exemple userName，password，entreprise．Après，user peut obtenir son userId comme la première clé． User doit utiliser userId et password pour log in．

1.3 Diagramme de classe

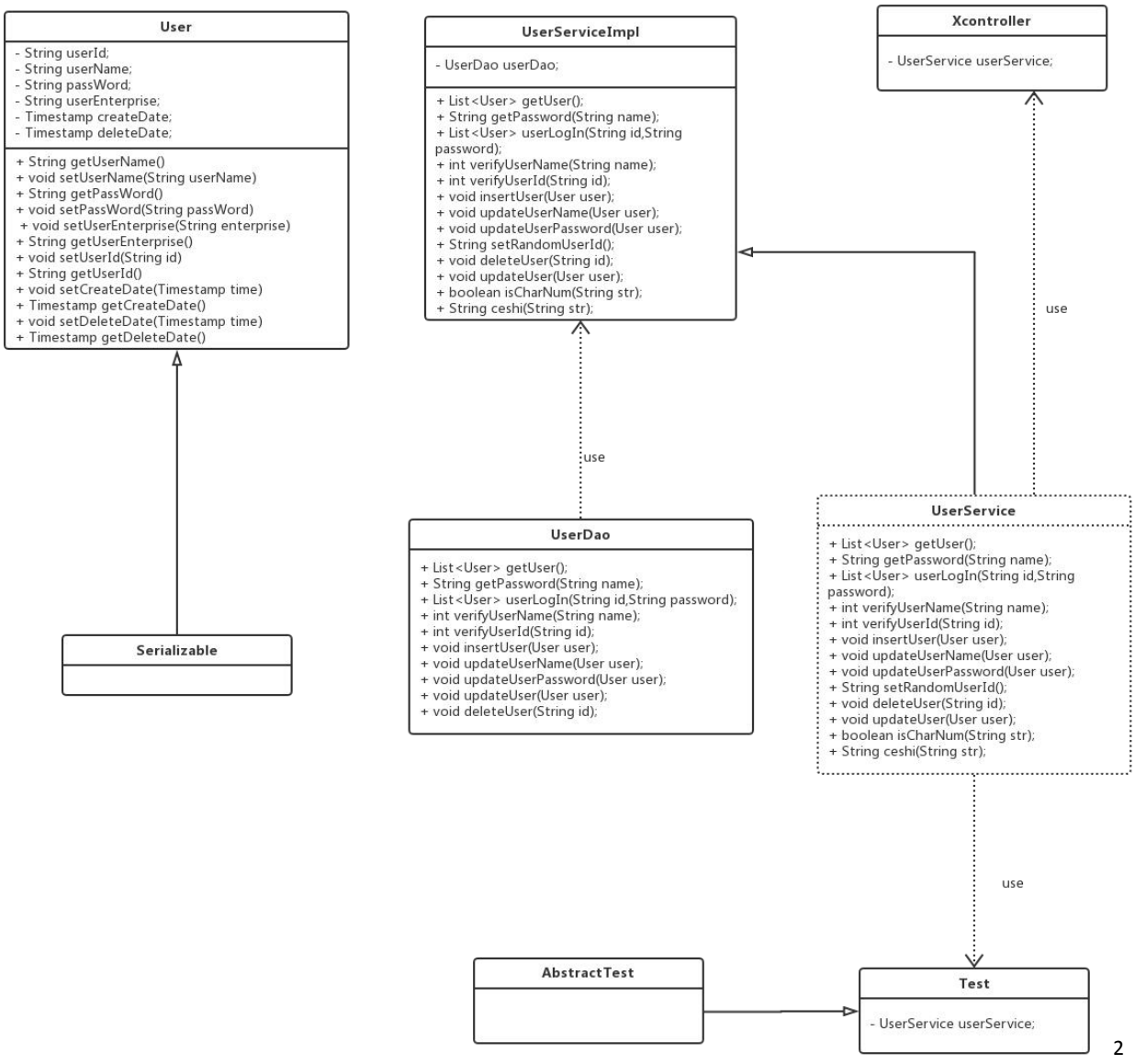


Diagramme de classe (1-1)

1.4、 Diagramme de séquence

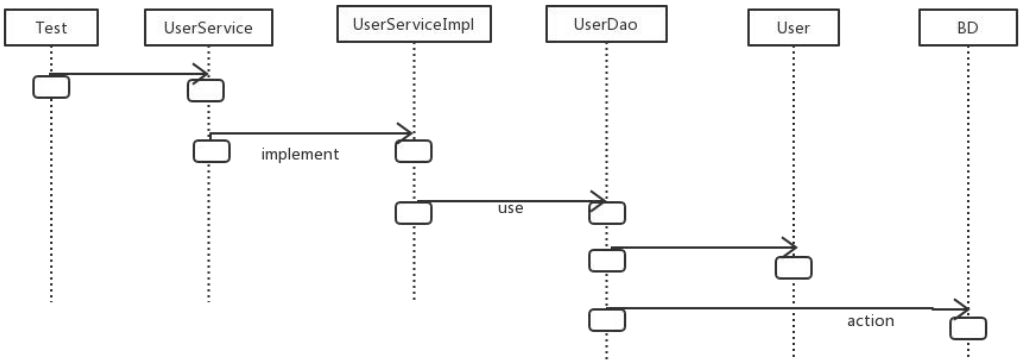


Diagramme de séquence (1-2)

Pendant ces deux jour ，

03/07/2017：je peux juste réussir deux fonctions－update et delete

0 4 /07/2017：je complète classe User

1.5、 le code

1.5.1 le document - test.xml

```
<mapper namespace="com.dao.UserDao">
<resultMap id="BaseUserMap" type="com.entity.User">
        <id property="userId" column="user_userId"/>
</resultMap>
        <select id="getUser" resultType="com.entity.User">
                select * from user
        </select>
        <select id="userLogIn" resultType="com.entity.User">
                select * from user where USERID=#{0} and PASSWORD=#{1}
        </select>
        <select id="verifyUserName" resultType="Int">
                        select count(1) from user where USERNAME=#{name}
        </select>
        <select id="verifyUserId" resultType="Int">
                        select count(1) from user where USERID=#{id}
        </select>
        <insert id="insertUser">
                insert into
user(USERID,USERNAME,PASSWORD,USERENTERPRISE,CREATEDATE,DELETEDATE) values
(#{userId,jdbcType=VARCHAR},#{userName,jdbcType=VARCHAR},#{passWord,jdbcType=VARCHAR},#{use
rEnterprise,jdbcType=VARCHAR},#{createDate,jdbcType=DATE},#{deleteDate,jdbcType=DATE})
        </insert>
        <update id="updateUser">
                update user set
```

```
USERNAME=#{userName,jdbcType=VARCHAR},PASSWORD=#{passWord,jdbcType=VARCHAR},USEREN
TERPRISE=#{userEnterprise,jdbcType=VARCHAR},CREATEDATE=#{createDate,jdbcType=DATE},DELETE
DATE=#{deleteDate,jdbcType=DATE} WHERE USERID=#{userId,jdbcType=VARCHAR}
    </update>
    <delete id="deleteUser">
        delete from user where USERID=#{id}
    </delete>
</mapper>
```

1.5.2 interface UserDao

```
public interface UserDao
{
    public List<User> getUser();            //sélection les informations de tous
    public List<User> userLogIn(String id,String password);    //chercher user avec userId et password
    public int verifyUserName(String name);      //chercher    userName
    public int verifyUserId(String id);      //chercher userId
    public void insertUser(User user);      //ajouter User à la table user
    public void updateUserName(User user);      //update userName
    public void updateUserPassword(User user);      // update userPassword
    public void updateUser(User user);      // update user
    public void deleteUser(String id);      //delete user
}
```

1.5.3 interface UserService

```
public interface UserService
{
    List<User> getUser();
    public List<User> userLogIn(String id,String password);
    int verifyUserName(String name);
     public int verifyUserId(String id);
     void insertUser(User user);
    void updateUserName(User user);
     void updateUserPassword(User user);
    public String setRandomUserId();
    public void deleteUser(String id);
     public void updateUser(User user);
     boolean isCharNum(String str);
}
```

1.5.4 classe UserServiceImpl

```
@Service
public class UserServiceImpl implements UserService
{
    @Resource
    private UserDao userDao;
    private List<Book> books;
```

```
	public List<User> getUser() {
		return userDao.getUser();
	}
	public int verifyUserName(String name)
	{
		return userDao.verifyUserName(name);
	}
	public List<User> userLogIn(String id,String password)
	{
		return userDao.userLogIn(id,password);
	}
	public void insertUser(User user) {
		userDao.insertUser(user);
	}
	public void updateUserName(User user) {
		userDao.updateUserName(user);
	}
	public void updateUserPassword(User user)
	{
		userDao.updateUserPassword(user);
	}
	public void updateUser(User user)
	{
		userDao.updateUser(user);
	}
	public void deleteUser(String id)
	{
		userDao.deleteUser(id);
	}
	public boolean isCharNum(String str)
	{
		String pattern="^[a-z0-9A-Z]+";
		return str.matches(pattern);
	}
	public int verifyUserId(String id)
	{
		return userDao.verifyUserId(id);
	}
	public String setRandomUserId() // quand user crée un compte le système doit donner user son userId au
hasard
	{
		Random ra=new Random();
		int length=ra.nextInt(10)+1;
		String base="0123456789";
```

```
            Random random=new Random();
            StringBuffer buffer=new StringBuffer();
            for(int i=0;i<length;i++)
            {
                int eNumber=random.nextInt(base.length());
                buffer.append(base.charAt(eNumber));
            }
            String string=buffer.toString();
            if(verifyUserId(string)==0)
            {
                return string;
            }
            else
            {
                return setRandomUserId();
            }
        }
        public List<User> userGetDemandes(String id){
            return userDao.userGetDemandes(id);
        }
}
```

Ⅱ、 05/07/2017：class Book

Pour bien manier ce structure ，J’ ajoute classe book.

2.1、 explication de classe Book

User peut vendre des livres en ligne. Pour Book, il y a cinq attributs：bookId，bookName，quantity，unitprice，bookSellerId．bookId est la première clé，bookSellerId est la clé étrangère (table:user, ligne:userId).

2.2、 Diagramme de classe

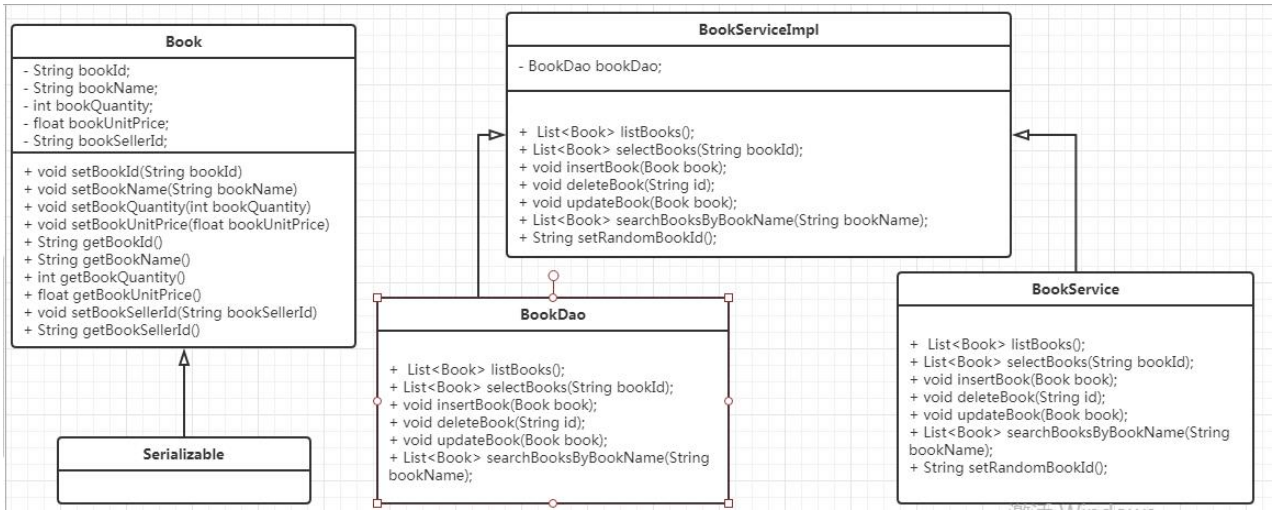


Diagramme de classe (2-1)

Ⅲ、 06/07/2017 - 07/07/2017 class Demande et Delivery

Pour bien compléter，j’ajoute deux autre classes Demande et Delivery.

Il y a huit attributs dans la classe Demande：demandeId，boobkId，quantity，userId，totalPrice，payOrNot，createDate，deliveryAdress.

Il y a neuf attributs dans la classe Demande： deliveryId， demandeId， userId， signOrNot， createDate，deliveryAdress，arriveDate，nowArriveAt，sellerId.

Donc dans la classe user， j′ ajoute un attribut “ private List<Demande>demandes ” et une fonction “ userGetDemandes”.

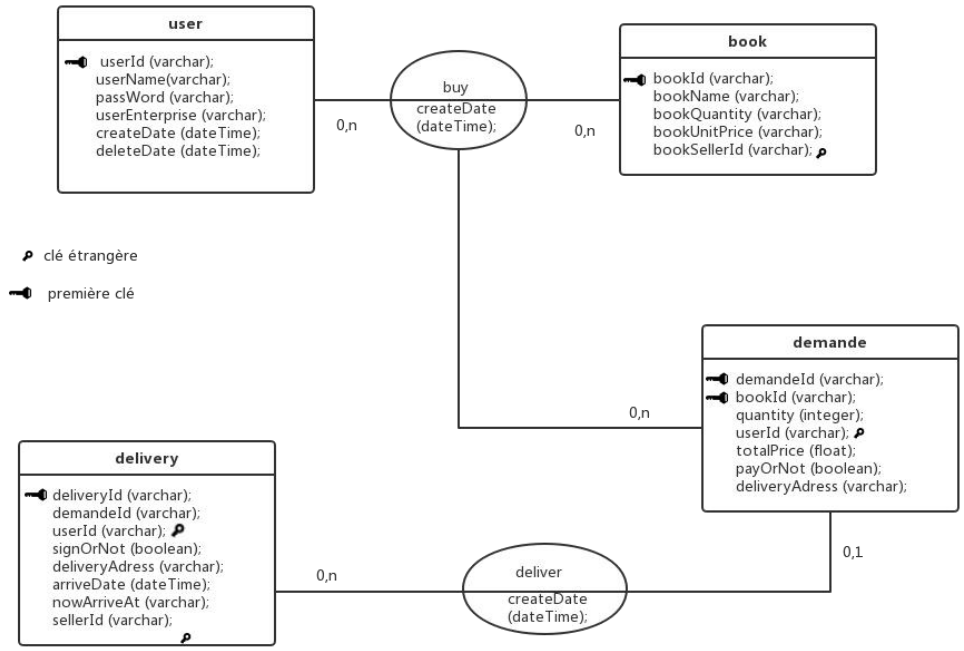
3.1、 le code de fonction userGetDemandes

```
<select id="userGetDemandes" resultMap="BaseUserMap">
        select* from user left join demande on user.userId=demande.userId where user.userId=#{0}
</select>
```

Dans ce fonction, je utilise “ left join ”, donc, dans le document <test.xml>,il besoin “ collection”：

```
<resultMap id="BaseUserMap" type="com.entity.User">
        <id property="userId" column="user_userId"/>
        <collection property="demandes" ofType="com.entity.Demande"></collection>
</resultMap>
```

Ⅳ、 modèle conceptuel de données



Modèle conceptuel de données

Ⅴ、 Conclusion

Pendant cette semaine, J' étudie ：

5.1、 les annotation，par exemple：

@Service：quand on voudrait utiliser 'bean'，il peut produire un objet，le nom de ce objet commence avec une lettre minuscule；

@Resource：chercher Bean；

@Controller：couche — contrôle；

@org．Juint．Test：run application comme juint test；

@RequestMapping：la requête，url，adresse；

@ResponseBody：le type de paramètre de 'return'；

@PathVariable：ajouter les paramètres dans la requête；

5.2、 SQL - join

Quand on utilise " left join" ," right join"," nature join"...dans le document ".xml", l'efficacité de fonctionnement deminue, donc, on doit eviter d'utiliser " join ".

5.3、 le plan

Après, je comprends que cette semaine，les documents que je programme sont backend．Je trouve que les informations sont affichées sur le web page sans '＼n'，ce n'était pas très beau，donc je voudrais bien compléter mon web page - vendre des livres en ligne．Par exemple， user peut log in avec click un bouton，les informations peut bien afficher sur le web page．

Ⅵ、le code d'autre classes．

1.1、Book.xml:

```
<mapper namespace="com.dao.BookDao">
<resultMap id="BaseBookMap" type="com.entity.Book">
    <id property="bookId" column="book_bookId"/>
</resultMap>
    <select id="listBooks" resultType="com.entity.Book">
        select* from book
    </select>
    <select id="selectBooks" resultType="com.entity.Book">
        select* from book where BOOKID=#{bookId}
    </select>
    <select id="searchBooksByBookName" resultType="com.entity.Book">
        select* from book where BOOKNAME like #{0}
    </select>
    <insert id="insertBook">
        insert into book(BOOKID,BOOKNAME,BOOKQUANTITY,BOOKUNITPRICE,BOOKSELLERID)
values
(#{bookId,jdbcType=VARCHAR},#{bookName,jdbcType=VARCHAR},#{bookQuantity,jdbcType=NUMERIC},
#{bookUnitPrice,jdbcType=FLOAT},#{bookSellerId,jdbcType=VARCHAR})
    </insert>

    <update id="updateBook">
```

```
        update book set
BOOKNAME=#{bookName,jdbcType=VARCHAR},BOOKQUANTITY=#{bookQuantity,jdbcType=NUMERIC}
,BOOKUNITPRICE=#{bookUnitPrice,jdbcType=FLOAT} WHERE BOOKID=#{bookId,jdbcType=VARCHAR}
    </update>
    <delete id="deleteBook">
        delete from book where BOOKID=#{id}
    </delete>
</mapper>
```

1.2 BookDao

```
public interface BookDao
{
        public    List<Book> listBooks();   // afficher tous les livres
        public List<Book> selectBooks(String bookId);       // chercher le livre avec BookId
        public void insertBook(Book book);       //ajoute un livre à la table book
        public void deleteBook(String id);          //delete un livre
        public void updateBook(Book book);     //update les informations de livre
        public List<Book> searchBooksByBookName(String bookName);//chercher le livre avec BookName
}
```

2.1 Demande.xml

```
<resultMap id="BaseDemandeMap" type="com.entity.Demande">
        <id property="demandeId" column="demande_demandeId"/>
        <id property="bookId" column="demande_bookId"/>
</resultMap>
        <select id="listDemande" resultType="com.entity.Demande">
        select* from demande
        </select>
        <select id="listDemandeOrderByDate" resultType="com.entity.Demande">
        select* from demande order by CREATEDATE,DEMANDEID
        </select>
        <select id="getDemande" resultType="com.entity.Demande">
        select* from demande where DEMANDEID=#{0}
        </select>
        <select id="getDemandeByBookId" resultType="com.entity.Demande">
        select* from demande where DEMANDEID=#{0} and BOOKID=#{1}
        </select>
        <select id="getDemandeByUserOrderByDate" resultType="com.entity.Demande">
        select* from demande where USERID=#{0}
        </select>
        <select id="getDemandeNotPay" resultType="com.entity.Demande">
        select* from demande where PAYORNOT='0' and USERID=#{0}
        </select>
      <insert id="insertDemande">
        insert into
demande(DEMANDEID,USERID,BOOKID,QUANTITY,TOTALPRICE,PAYORNOT,CREATEDATE,DELIVER
```

```
YADRESS) values
(#{demandeId,jdbcType=VARCHAR},#{userId,jdbcType=VARCHAR},#{bookId,jdbcType=VARCHAR},#{quan
tity,jdbcType=NUMERIC},#{totalPrice,jdbcType=FLOAT},#{payOrNot,jdbcType=NUMERIC},#{createDate,jdb
cType=TIMESTAMP},#{deliveryAdress,jdbcType=VARCHAR})
    </insert>
    <update id="updateDemande">
            update demande set
USERID=#{userId,jdbcType=VARCHAR},QUANTITY=#{quantity,jdbcType=NUMERIC},TOTALPRICE=#{tot
alPrice,jdbcType=FLOAT},PAYORNOT=#{payOrNot,jdbcType=NUMERIC},CREATEDATE=#{createDate,jdbc
Type=TIMESTAMP},DELIVERYADRESS=#{deliveryAdress,jdbcType=VARCHAR} where
DEMANDEID=#{demandeId,jdbcType=VARCHAR} and BOOKID=#{bookId,jdbcType=VARCHAR}
    </update>
    <delete id="deleteDemande">
        delete from demande where DEMANDEID=#{0} and BOOKID=#{1}
    </delete>
</mapper>
```

2.2 DemandeDao

```
public interface DemandeDao
{
    public List<Demande> listDemande();
    public List<Demande> listDemandeOrderByDate();
    public List<Demande> getDemande(String demandeId);
    public void insertDemande(Demande demande);
    public void updateDemande(Demande demande);
    public void deleteDemande(String demandeId,String bookId);
    public List<Demande> getDemandeByUserOrderByDate(String userId);
    public List<Demande> getDemandeByBookId(String demandeId,String bookId);
    public List<Demande> getDemandeNotPay(String id);
}
```

3.1 Delivery.xml

```
<resultMap id="BaseDeliveryMap" type="com.entity.Delivery">
    <id property="deliveryId" column="delivery_delivery_deliveryId"/>
    <collection property="demandesForDelivery" ofType="com.entity.Demande"></collection>
</resultMap>
        <select id="listDelivery" resultType="com.entity.Delivery">
        select* from delivery
        </select>
        <select id="searchDeliveryByUserId" resultType="com.entity.Delivery">
        select* from delivery where USERID=#{0} order by CREATEDATE DESC
        </select>
        <select id="searchDeliveryByDemandeId" resultType="com.entity.Delivery">
        select* from delivery where DEMANDEID=#{0}
        </select>
        <select id="demandeWaitingForDelivery" resultType="com.entity.Demande">
```

```
        select * from demande    where demande.bookId in (SELECT DISTINCT bookId from book where
bookSellerId=#{0}) AND demande.demandeId not in(select distinct delivery.demandeId from delivery) and
demande.payOrNot=1 order by demande.createDate,demande.demandeId
        </select>
        <select id="searchDelivery" resultType="com.entity.Delivery">
         select* from delivery where DELIVERYID=#{0}
        </select>
        <insert id="insertDelivery">
        insert into
delivery(DELIVERYID,DEMANDEID,USERID,SIGNORNOT,DELIVERYADRESS,CREATEDATE,ARRIVED
ATE,NOWARRIVEAT,SELLERID) values
(#{deliveryId,jdbcType=VARCHAR},#{demandeId,jdbcType=VARCHAR},#{userId,jdbcType=VARCHAR},#{si
gnOrNot,jdbcType=NUMERIC},#{deliveryAdress,jdbcType=VARCHAR},#{createDate,jdbcType=TIMESTAMP}
,#{arriveDate,jdbcType=TIMESTAMP},#{nowArriveAt,jdbcType=VARCHAR},#{sellerId,jdbcType=VARCHAR}
)
        </insert>
        <update id="updateDelivery">
                update delivery set
DEMANDEID=#{demandeId,jdbcType=VARCHAR},USERID=#{userId,jdbcType=VARCHAR},SIGNORNOT=
#{signOrNot,jdbcType=NUMERIC},DELIVERYADRESS=#{deliveryAdress,jdbcType=TIMESTAMP},CREATE
DATE=#{createDate,jdbcType=TIMESTAMP},ARRIVEDATE=#{arriveDate,jdbcType=TIMESTAMP},NOWAR
RIVEAT=#{nowArriveAt,jdbcType=VARCHAR},SELLERID=#{sellerId,jdbcType=VARCHAR} where
DELIVERYID=#{deliveryId,jdbcType=VARCHAR}
         </update>
         <delete id="deleteDelivery">
         delete from delivery where DELIVERYID=#{0}
         </delete>
</mapper>
```

3.2 DeliveryDao

```
public interface DeliveryDao
{
        public List<Delivery> listDelivery();
        public void insertDelivery(Delivery delivery);
        public List<Delivery> searchDelivery(String deliveryId);
        public void updateDelivery(Delivery delivery);
        public void deleteDelivery(String deliveryId);
        public List<Demande> demandeWaitingForDelivery(String sellerId);
        public List<Delivery>searchDeliveryByUserId(String userId);
        public List<Delivery>searchDeliveryByDemandeId(String demandeId);
}
```

4. Calsse Xcontroller

Il y a plan de fonctions dans ce classe ， par exemple ， userLogIn()

```
@Controller
@RequestMapping("/user")
```

```
public class Xcontroller
{
	@Resource
	private UserService userService;
	@Resource
	private BookService bookService;
	@Resource
	private DemandeService demandeService;
	@Resource
	private DeliveryService deliveryService;

@ResponseBody
@RequestMapping("/{userId}/{userPassword}/logIn") //user log in
public String users(@PathVariable String userId, @PathVariable String userPassword,HttpServletRequest request)
{
		session=request.getSession(); // set session
		//session.setMaxInactiveInterval(10*60);
		String sessionId=session.getId();
		User u=new User();
		u.setUserId(userId);
		u.setPassWord(userPassword);
		if(userService.userLogIn(userId,userPassword).size() ==1)
		{
			u=userService.userLogIn(userId, userPassword).get(0);
			session.setAttribute("userName", u.getUserName());
			session.setAttribute("userId", userId);
			session.setAttribute("userPassword", userPassword);
			System.out.println(u.getUserName());
			return u.getUserName()+ ", welcome to your homepage! your sessionID:"+sessionId;
		}
		else
		{
			return "wrong id or password.";
		}
}
}
```

Le résultat：