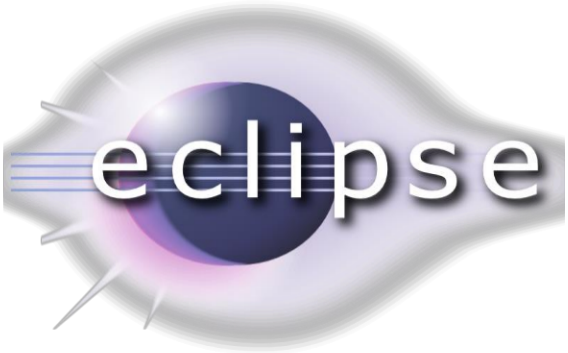
Université Hassan 1er
École Nationale des Sciences Appliquées – Berrechid

POLYCOPIE DES TRAVAUX PRATIQUES

PROGRAMMATION JEE



PROFESSEUR : LAHCEN MOUMOUN

Sommaire

Introduction

Java Enterprise Edition (JEE) est une technologie incontournable dans le monde du développement d'applications d'entreprise. Sa robustesse, sa scalabilité et son écosystème riche en font un choix privilégié pour les grandes applications web et systèmes d'information. Que ce soit pour des banques, assurances, administrations, ou autres grandes organisations, JEE offre la puissance nécessaire pour répondre aux besoins les plus complexes avec efficacité et sécurité.

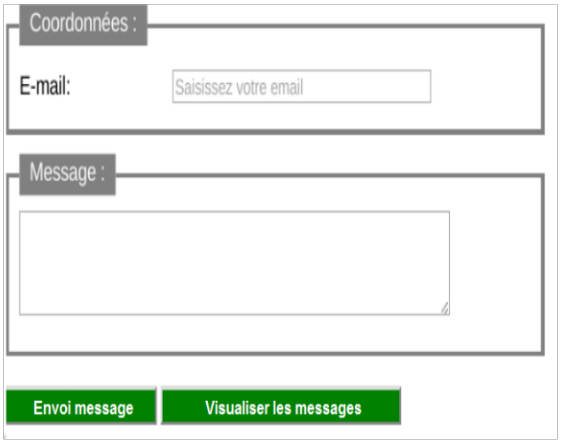
Ce polycopié de travaux pratiques est destiné aux étudiants souhaitant approfondir leurs compétences en développement d'applications web d'entreprise à l'aide d'une plateforme JEE. Grâce à une série progressifs de travaux pratiques, les étudiants seront en mesure de mettre en pratique les concepts théoriques abordés en cours et de développer une compréhension concrète des différents composants de l'écosystème Java EE.

TP N°1 : Servlet et JSP

On considère un forum de discussion permettant aux enseignants et aux étudiants. Les différentes informations de forum sont sauvegardées dans une base de données Mysql « BDForum ». l'application à réaliser doit assurer les tacjes suivantes :

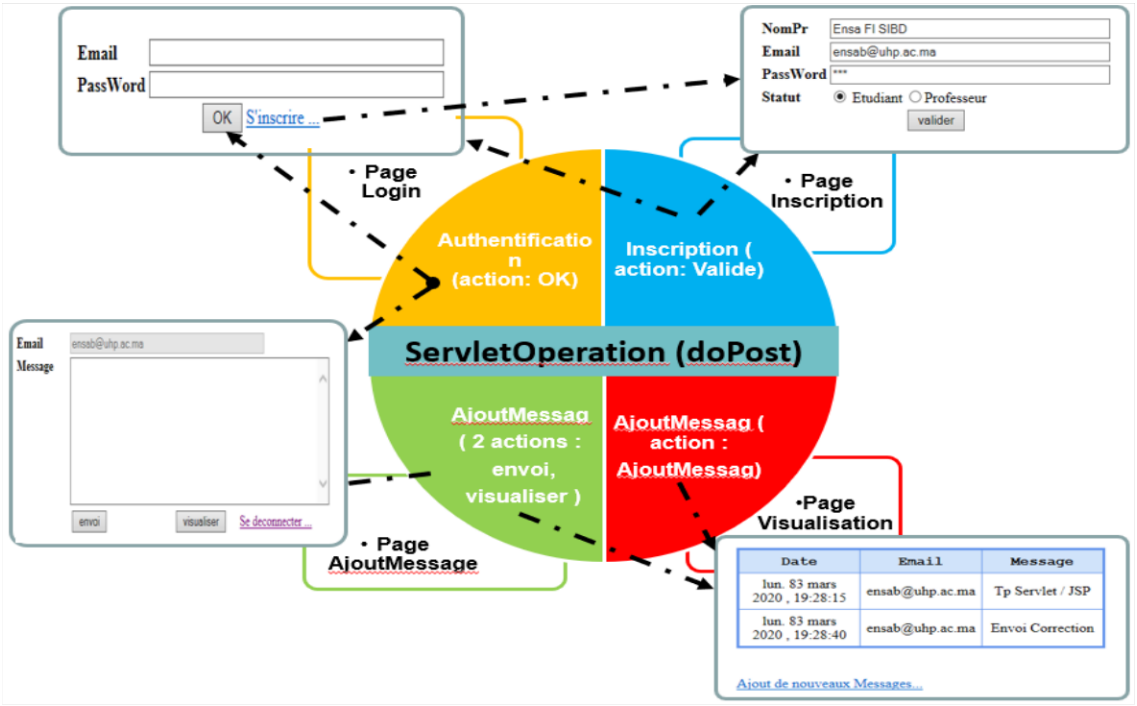
- Poster un message pour une personne loguée ;
- Visualiser l'ensemble des messages des étudiants pour un enseignant et uniquement ces propres messages dans le cas d'un étudiant.
- S'inscrire au forum.
- Ces taches sont réalisées à l'aide d'une servlete « servletMessage » qui fait appel aux pages JSP contenant les éléments présents dans les maquettes présentées ci-dessous.

Les informations sont enregistrées sur une base de données MYSQL. L'accès à cette dernière est réalisé en utilisant JDBC.



Elément de correction

Schéma d'exécution



Classes Metier

User : Metier
- email : String; - nomPr : String - password: String - statut : boolean // true pour etudiant
+ User ( ) + getXxxx():…; + setXxx(…):… + toString(): String; + estProf(): boolean + equals(obj: Object): boolean …

Message : Metier
- user : User - message : String - date : Date
+ Message (user : User, message : String) + getXxxx(): … ; + setXxx(…): … + toString(): String + equals(obj : Object): boolean …

NB. Pour la manipulation des objets de type « Date », nous utilisant un objet de type « SimpleDateFormat »

```
private static SimpleDateFormat formatter = new SimpleDateFormat ("E D MMM yyyy , H:m:s", Locale.FRANCE)

public String getDate() {return formatter.format ( date );}
```

Web.XML

```
<servlet>
   <servlet-name>Service </servlet-name>
   <servlet-class> Web.ServiceServlet </servlet-class>  </servlet>
<servlet-mapping>
   <servlet-name>Service</servlet-name>
   <url-pattern>/Service</url-pattern>   </servlet-mapping>
```

Package Dao : Classe Connexion

```
public class Connexion {
  private static Connection connection;
  static {
        try {
              Class.forName("com.mysql.jdbc.Driver");
              connection=DriverManager.getConnection
                ("jdbc:mysql://localhost:3306/bdforum","root","");
        } catch (ClassNotFoundException e) { e.printStackTrace();
        } catch (SQLException e) { e.printStackTrace(); }
        }
  public static Connection getConnection() { return connection; }
}
```

Package Dao : Classe ServiceDao

```
public class ServiceDao {
   private static  Connection connection = Connexion.getConnection();
   public static ArrayList<User> getAllUser( ) throws SQLException {
     ArrayList<User> users= new ArrayList<User>();
     ResultSet rs=connection.createStatement().executeQuery
                 ("Select * from user");
     while (rs.next()) {
           User user= new User(); user.setEmail(rs.getString("email"));
           user.setNomPr(rs.getString("nomPr"));
           user.setPassWord(rs.getString("password"));
           user.setStatut( rs.getBoolean("statut"));
           users.add(user);  }
     rs.close(); return users;  }
```

```
    public static void addUserDao(User user ) throws SQLException {
      PreparedStatement ps= connection.prepareStatement(
              "insert into user values(?,?,?,?) ");
      ps.setString(1, user.getEmail());
      ps.setString(2, user.getNomPr());
      ps.setString(3, user.getPassWord());
      ps.setBoolean(4, user.getStatut());
      ps.executeUpdate();  }
    public static ArrayList<Message> getAllMessage() throws Exception{
      ArrayList<Message> messages= new ArrayList<Message>();
      ResultSet rs = connection.createStatement().executeQuery(
                "SELECT * FROM message");
      while (rs.next()) {
           messages.add( new Message( ServiceMetier.getUserByEmail(
              rs.getString("emailUser") ), rs.getString("txtMessage"),
              rs.getDate("dtMessage") )   );   }
      rs.close();  return messages;  }
    public static void addMessageDao(Message message) throws SQLException{
      PreparedStatement ps=connection.prepareStatement(
         "INSERT INTO message (txtMessage, dtMessage, emailUser)"
         + " VALUES (?,?,?)");
       ps.setString(1, message.getMessage());
       ps.setDate(2, message.getDate2());
       ps.setString(3, message.getUser().getEmail());
       ps.executeUpdate(); }
}
```

Package Metier : Classe User

```
public class User implements Serializable{
      private String,email,passWord; private boolean statut=true
      public User() {   }
      public String getNomPr() {return nomPr;}
      public void setNomPr(String nomPr) {this.nomPr = nomPr;    }
      public String getEmail() {    return email;}
      public void setEmail(String email) {this.email = email;    }
```

```
	public String getPassWord() {return passWord;}
	public void setPassWord(String passWord){this.passWord=passWord;}
	public boolean getStatut() {return statut;}
	public void setStatut(Boolean statut) {  this.statut = statut;}
	public void setStatut(String statut) {
		if (statut.equals("Etudiant"))this.statut=true;
		else this.statut=false; }
	public boolean estProf() {return !statut;}
	public boolean estEtudiant() {return statut;}
	@Override
	public boolean equals(Object obj) {
		if (this == obj) return true;
		if (obj == null) return false;
		if (getClass() != obj.getClass()) return false;
		User other = (User) obj;
		if (email == null) {
			if (other.email != null) return false;
		} else if (!email.equals(other.email)) return false;
		return true;}
	@Override
	public String toString() {
		return "User [nomPr=" + nomPr + ", email=" + email + ",
			passWord=" + passWord + ", statut=" + statut + "]";}
}
```

Package Metier : Classe Message

```
public class Message implements Serializable {
	private User user ; private String message; private Date date;
	private static SimpleDateFormat formatter =
	  new SimpleDateFormat ("E D MMM yyyy , H:m:s", Locale.FRANCE);
	public Message() {
	  this.date = new Date( (new java.util.Date()).getTime() ); }
	public Message(User user, String message, Date date) {
		this.user = user; this.message = message; this.date = date;}
	public User getUser() { return user;}
```

```
	public void setUser(User user) {this.user = user;}
	public String getMessage() {return message;    }
	public void setMessage(String message) {this.message = message;}
	public String getDate() {
		if (date== null)  return "";
		return formatter.format(new java.util.Date(date.getTime()));}
	public Date getDate2() {     return (date );}
	public void setDate(Date date) {this.date = date;}
	@Override
	public String toString() {
		return "Message[user="+user+","+" message="+message+
				", date=" + getDate() + "]";  }
}
```

Package Metier : Classe ServiceMetier

```
public class ServiceMetier {
	public static ArrayList<User> getUsers() throws Exception {
		return (ArrayList<User>) ServiceDao.getAllUser();}
	public static ArrayList<Message> getMessages() throws Exception {
		return (ArrayList<Message>) ServiceDao.getAllMessage();}
	//Methodes metiers
	public static User getUserByEmail(String email) throws Exception{
		for(User user:getUsers()) {
			if(user.getEmail().equals(email))  return user;}
		return null; }
	public static User getUser(String email,String mPasse)throws Exception{
		for(User user:getUsers()) {
			if (user.getEmail().equals(email) &&
			  user.getPassWord().equals(mPasse)){return user;}
		}
		return null; }
	public static void addUser(User user )throws SQLException{
		 ServiceDao.addUserDao(user); }
	public static void addMessage(Message msg )throws SQLException{
		 ServiceDao.addMessageDao(msg);}
```

```
	public static ArrayList<Message> getMessagesByUser (User user)
                                        throws Exception{
		if (user.estProf()) return getMessages();
		ArrayList<Message> mes= new ArrayList<Message>();
		for (Message m: getMessages()) {
			if (m.getUser().equals(user)) mes.add(m);	}
		return mes; }
}
```

Package Web : Classe ServiceServlet

```
@WebServlet("/ServiceServlet")
public class ServiceServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;
	ServiceMetier serviceMetier;
	protected void doGet(HttpServletRequest request, HttpServletResponse
			response) throws ServletException, IOException {
		request.getSession().setAttribute("user", null);
		getServletContext().getRequestDispatcher("/login.jsp").
			forward(request,  response);  }
	protected void doPost(HttpServletRequest request, HttpServletResponse
			response) throws ServletException, IOException {
	    HttpSession session= request.getSession();
	    String page= (String) session.getAttribute("page");
	    User user;
	    if (page.equals("inscription")) {
	      String email=request.getParameter("email");
	      try {
			if (ServiceMetier.getUserByEmail(email)==null){
				user= new User();
				user.setEmail(email);
				user.setNomPr(request.getParameter("nomPr"));
				user.setPassWord(request.getParameter("passWord"));
				user.setStatut(request.getParameter("statut"));
				ServiceMetier.addUser(user);
				doGet(request,response);
```

```
        } else {
          request.setAttribute("erreur","email deja existant");
          getServletContext().getRequestDispatcher(
              "/inscription.jsp").forward(request,response) }
} catch (Exception e) {
        request.setAttribute("erreur", "Erreur");
        getServletContext().getRequestDispatcher(
           "/inscription.jsp").forward(request,  response); }
} else
if (page.equals("login")) {
  String email=request.getParameter("email");
  String passWord=request.getParameter("passWord");
  try {
        user=ServiceMetier.getUser(email, passWord);
        if (user!=null) {
          session.setAttribute("user", user);
          request.setAttribute("erreur", "Erreur");
          getServletContext().getRequestDispatcher(
           "/AjoutMessage.jsp").forward(request,  response);
        } else {
        request.setAttribute("erreur",
                         "email ou mot de passe incorrect");
        doGet(request,response);  }
  } catch (Exception e) {
        request.setAttribute("erreur", "Erreur Exception");
        doGet(request,response);
  }
} else
      if (page.equals("AjoutMessage")) {
        String action=request.getParameter("action");
        user=(User)session.getAttribute("user");
        if (action.equals("envoi")) {
              Message message=new Message();
              message.setUser(user);
```

```
                        message.setMessage(request.getParameter("message"));
                        try {
                              ServiceMetier.addMessage(message);
                        } catch (SQLException e) {    e.printStackTrace();}
                        getServletContext().getRequestDispatcher(
                         "/AjoutMessage.jsp").forward(request,response);
                    } else
                      if(action.equals("visualiser")){
                        getServletContext().getRequestDispatcher
                             ("/visualiserMessages.jsp").forward
                                                  (request,response); }
                  }
       }
}
```

Page JSP Login

```
<body>
     <FORM action = "Service" method ="post">
     <TABLE >
      <TR>
            <TD> <b>Email </b>  </TD>
            <TD> <input type="text"  name="email"  style="width:300px">
      </TD> </TR>
      <TR>
            <TD> <b>Mot de passe </b>  </TD>
            <TD> <input type="text" name="passWord" style="width:300px">
      </TD> </TR>
      <TR>
            <TD  colspan =2 align=center >
            <INPUT type ="submit" name="login" value="OK">
            <a href="inscription.jsp">S'inscrire ... </a>
      </TD> </TR>
      </TABLE>
    </FORM>
    <% session.setAttribute("page", "login");
         String erreur= (String) request.getAttribute("erreur");
         if (erreur!=null ){
            out.print(erreur); }
    %>
</body>
```

Page JSP Inscription

```
<body>
<h2> <FORM action = "Service" method ="post">
     <TABLE >
      <TR>  <TD> <b>NomPr</b>  </TD>
            <TD> <input type="text"  name="nomPr" style="width:300px">
      </TD> </TR>
      <TR>  <TD> <b>Email </b>  </TD>
            <TD> <input type="text"  name="email" style="width:300px">
      </TD> </TR>
      <TR>
            <TD> <b>PassWord </b>  </TD>
            <TD> <input type="text" name="passWord" style="width:300px">
      </TD> </TR>
      <TR>  <TD> <b>Statut</b>  </TD>
      <TD><input type="radio" name="statut" value="Etudiant" checked>Etudiant
          <input type="radio" name="statut" value="Professeur" >Professeur
      </TD> </TR>
      <TR> <TD colspan =2 align=center >
            <INPUT type ="submit" name="Valider" value="valider">
      </TD> </TR>
      </TABLE>
    </FORM>
        <% session.setAttribute("page", "inscription");
         String erreur= (String) request.getAttribute("erreur");
         if (erreur!=null ){  out.print(erreur); }
    %>
</body>
```

Page JSP Ajout Message

```
<body>
     <% User user=(User) session.getAttribute("user"); %>
     <FORM action = "Service" method ="post">
     <TABLE >
      <TD width="100" > <b>Email </b>  </TD>
      <TD> <input type="text" name="email" style="width:300px" disabled
                  value= <%=user.getEmail()%>> </TD>
      </TR>
      <TR> <TD  valign="top" > <b>Message</b>  </TD>
      <TD> <textarea name="message" placeholder="Saisir un message"
            rows="10" cols="30" style="width:400px;height:150px">
      </textarea> </TD> </TR>
      </TABLE>
      <TABLE>
      <TR> <TD  align=center width="50%">
            <input type="submit" name="action" value="envoi" width="60"
                  height="60"> </TD>
```

```
      <TD  align=center width="25%">
            <input type="submit" name="action" value="visualiser"
                  width="60" height="60">      </TD>
      <TD  align=center valign= "middle">
            <a href="Service">Se deconnecter ... </a> </TD> </TR>
      </TABLE>
    </FORM>
    <% session.setAttribute("page", "AjoutMessage");  %>
</body>
```

Page JSP Affichage messages

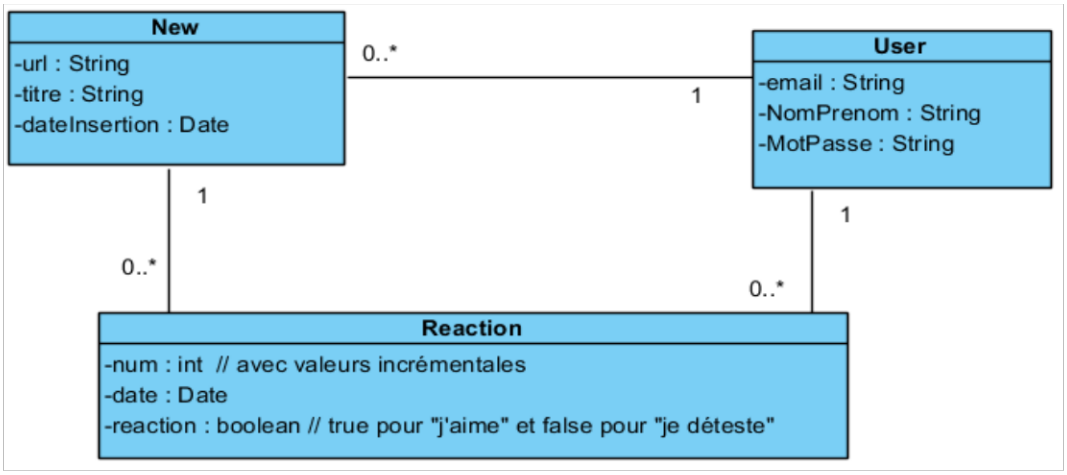
```
<body>
 <%
    User user=(User) session.getAttribute("user");
    ArrayList<Message> messages=ServiceMetier.getMessagesByUser(user);%>


  <TABLE >
      <TR> <TH >Date</TH> <TH >Email</TH> <TH >Message</TH> </TR>
      <%  for (Message message:messages ) { %>
          <TR> <TD > <%=message.getDate() %>       </TD>
          <TD > <%=message.getUser().getEmail() %>       </TD>
          <TD > <%=message.getMessage() %>    </TD>  </TR>
      <% } %>
</TABLE>
<BR><BR>
<a href="AjoutMessage.jsp">Ajout de nouveaux Messages... </a>
<% session.setAttribute("page", "visualiserMessages");%>
</body>
```

TP N°2 : Servlet , JSP et Hibernate

Notre objectif à travers ce TP est le développement d'un site Web simple de gestion des news. Un tel site Web est basé sur le principe suivant :

- Un utilisateur doit s'authentifier pour accéder aux différents news. Les nouveaux utilisateurs peuvent s'inscrire (en fournissant leur courriel « mail » considéré comme identifiant, le nom+prénom et le mot de passe).
- Un utilisateur peut ajouter une news au site si elle n'existe pas déjà dans le site, c'est à dire référencer une page Web (une URL) pour signaler son existence aux autres utilisateurs.
- Un utilisateur peut aimer ou détester une news ajouté précédemment par un utilisateur.

Nous considérons que l'analyse qui a été effectuée pour ce site a permis d'élaborer le diagramme suivant :



A. Dans l'implémentation JAVA, nous ajoutons à la classe « Reaction » un objet membre user et un objet membre news qui représentent respectivement l'utilisateur qui a réagi et la news associée à une réaction. Également, un objet membre user est ajouté à la classe « News » pour sauvegarder l'utilisateur qui a créé la news. Les classes métiers correspondantes au diagramme ci-dessus sont les suivantes :

Metier : User
- email : String - nomPrenom :String - motPasse : String
+ User () + getXxx():… ; + setXxx(…) : … + equals(Object obj) : boolean

Metier : News
- url : String ; - titre : String - dateNews : Date // Date courante - createur : User
+ News () + getXxx(): … ; + setXxx(…): … + equals(Object obj) : boolean

Metier : Reaction
- num // numéro avec valeurs incrémentales - dateReaction : Date // Date courante - reaction : boolean // true pour "j'aime" et false pour "je déteste" - user : User - news : News
+ Reaction () + getXxx(): … ; + setXxx(…): … // Xxx represente un attribute + equals(Object obj) : boolean

B. Les différentes informations des utilisateurs et des news sont enregistrées dans une base de données MySQL nommée « BdNews » dont le shéma est le suivant (une clé primaire est soulignée alors qu'une clé étrangère est précedée par #) :

- **User**(email :varchar(20),nomPrenom:varchar(30),motPasse:varchar(15))
- **News**(url: varchar(30), titre : varchar(50), dateNews :date , #email: varchar(20))
- **Reaction**(num: autoIncre, dateReaction: Date, reaction: boolean , #email: varchar(20), #url:varchar (30))

1- Créer la base de données « BdNews »

2- Créer un package dao contenant une classe singleton « HibernateUtil » qui assure la construction d'une unique SessionFactory , une classe « ServiceDao » qui regroupe les services Hibernate liés à la base de données « dbVentes ».

C. Nous considérons que les différents services métiers de notre site web sont définis dans la classe « ServiceMetier » suivante :

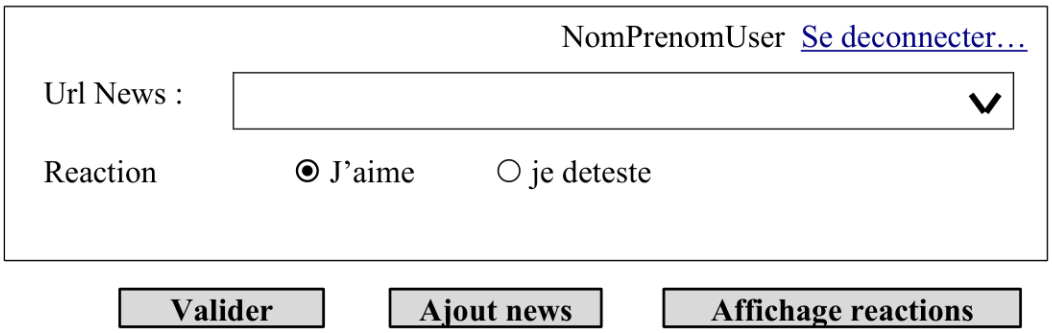
Metier : ServiceMetier
- listUsers : static ArrayList<User> - listNews : static ArrayList<News> - listReactions : static ArrayList<Reaction>
+ ServiceMetier () + getXxx(): … ; + setXxx(…) : … + getUser(email: String): User; + getUser(email: String, motPasse:String) + getNewsByUrl(url: String): News; + getAllUrl():ArrayList<String> + addNews (news: News) : boolean + addReaction (reaction: Reaction) : boolean + getNewsCreateByUser(user: User): ArrayList<News> + getReactionByUser(user: User): ArrayList<Reaction> + getReactionByNews(news: News): ArrayList<Reaction> + getNombreJaime(news: News): int; + getNombreJeDeteste(news: News): int …

D. Dans cette partie, nous nous intéressons à la couche Web et la couche présentation de notre site. Les différentes pages web de ce site sont des pages JSP alors que les traitements sont réalisés par une seule Servlette nommée « ServletService ». Nous nous limitons aux taches ci-dessous :

1- Ecrire le code de la page « login.jsp » permettant d'assurer l'authentification d'un utilisateur. Cette page comporte un formulaire avec les champs : login, mot de passe et un bouton connexion. Lorsqu'un utilisateur est logué, ses informations sont sauvegardées dans la scope session avant d'afficher la page d'ajout de réactions ;

2- Ecrire le code de la page « Inscription.jsp » permettant d'assurer l'inscription d'un nouvel utilisateur. Cette page comporte un formulaire avec les champs : email, nom prenom, mot de passe et un bouton validation. Lorsqu'un utilisateur est inscrit, ses informations sont sauvegardées dans la base de données « BdNews » ;

3- Donner le code de la page « ajoutReaction.jsp » dont l'aspect visuel est fourni par la figure ci-dessous. On suppose qu'un attribut nommé « urls » est placé dans la scope application, cet attribut contient la liste des urls des différents news du site. Ainsi, la liste de la page « ajoutReaction.jsp » permettra de sélectionner une news à partir de son URL.



1- Proposer une page JSP pour l'affichage des reaction des différentes news. Cette page devra permettre de rechercher les réctions selon un critère de news ou de date de réaction ;

2- Définir la servlet «ServiceServlet ».

Elément de correction

Package Dao : classe HibernateUtil

```
public class HibernateUtil {
    public static final SessionFactory sessionFactory;
    static {
        try {
            // Création de SessionFactory à partir de hibernate.cfg.xml
            sessionFactory = new Configuration().configure
                  ("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
           System.err.println("SessionFactory creation failed." + ex);
           throw new ExceptionInInitializerError(ex);
        }
    }
    public static SessionFactory getSessionFactory(){
        return sessionFactory;
    }
}
```

Package Dao : classe ServiceModel

```
public class ServiceModel {
      public static boolean add(Object obj) {
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
       session.beginTransaction();  session.save(obj);
       session.getTransaction().commit();
       return true;
      }
      public static boolean addUser(User user) throws Exception  {
            return add(user);    }
      public static boolean addNews(News news) throws Exception  {
            return add(news);  }
      public static boolean addReaction(Reaction reaction)throws Exception{
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
          session.beginTransaction(); System.out.println(reaction);
          session.saveOrUpdate(reaction);
          session.getTransaction().commit();
          return true;
    }
      public static List<User> getAllUsers() throws Exception{
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
            session.beginTransaction();
            Query req=session.createQuery("from User");
            return  req.list();
      }
```

```
	public static List<News> getAllNews() throws Exception{
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Query req=session.createQuery("from News");
		return  req.list();}
	public static List<Reaction> getAllReaction() throws Exception{
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Query req=session.createQuery("from Reaction");
		return  req.list();}
	public static User getUser(String email) throws Exception {
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		return ((User) session.load(User.class, email)); }
	public static News getNews(String url) throws Exception {
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Criteria criteria = session.createCriteria(News.class);
		Criterion critere = Restrictions.like("url", url);
		criteria.add(critere);
		if (criteria.list().size()==0) return (null);
		else return (News) criteria.list().get(0);}
	public static Reaction getReaction(int num) throws Exception {
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		return ((Reaction) session.load(Reaction.class, num));
	}
	public static User getUser(String email,String motPasse)throws Exception{
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Criteria criteria = session.createCriteria(User.class);
		Criterion critere = Restrictions.like("email", email);
		criteria.add(critere);
		critere = Restrictions.like("motPasse", motPasse);
		criteria.add(critere);
		if (criteria.list().size()==0) return (null);
		else return (User) criteria.list().get(0);
	}
	public static Reaction getReaction(User user,News news)throws Exception{
	 Session session=HibernateUtil.getSessionFactory().getCurrentSession();
		session.beginTransaction();
		Criteria criteria = session.createCriteria(Reaction.class);
		Criterion critere = Restrictions.like("user", user);
		criteria.add(critere);
		critere = Restrictions.like("news", news);
		criteria.add(critere);
		if (criteria.list().size()==0) return (null);
		else return (Reaction) criteria.list().get(0); }
}
```

Fichier de configuration hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
   <!-- Database connection settings -->
   <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
      name="connection.url">jdbc:mysql://localhost:3306/BdNews</property>
    <property name="connection.username">root</property>
    <property name="connection.password"></property>
    <!-- JDBC connection pool (use the built-in) -->
    <property name="connection.pool_size">1</property>
    <!-- SQL dialect -->
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.ejb.cfgfile">
                  /org/hibernate/ejb/test/hibernate.cfg.xml</property>
    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>
    <!-- Disable the second-level cache  -->
    <property name="cache.provider_class">
                       org.hibernate.cache.NoCacheProvider</property>
    <!-- Echo all executed SQL to stdout -->
    <property name="show_sql">false</property>
    <!-- Update the database schema on startup -->
    <property name="hbm2ddl.auto">update </property>
    <!-- mapping resource many to many -->
    <mapping resource="metier/User.hbm.xml"/>
    <mapping resource="metier/News.hbm.xml"/>
    <mapping resource="metier/Reaction.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Package Metier : Classe User

```
public class User implements Serializable{
      private String email, nomPrenom, motPasse;
      private Set<News> news= new HashSet<News>();
      private Set<Reaction> reactions= new HashSet<Reaction>();
      public User() {   }
      public User(String email, String nomPrenom, String motPasse) {
       this.email=email;this.nomPrenom=nomPrenom;    this.motPasse=motPasse;}
      public String getEmail() {return email;  }
      public void setEmail(String email) { this.email = email; }
      public String getNomPrenom() { return nomPrenom; }
      public void setNomPrenom(String nomPrenom) {
                  this.nomPrenom = nomPrenom;  }
      public String getMotPasse() { return motPasse;}
      public void setMotPasse(String motPasse) {this.motPasse = motPasse;}
      public Set<News> getNews() {  return news;}
```

```
	public void setNews(Set<News> news) { this.news = news;}
	public Set<Reaction> getReactions(){ return reactions; }
	public void setReactions(Set<Reaction> reactions) {
			this.reactions = reactions;}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (obj == null)  return false;
		if (getClass() != obj.getClass())  return false;
		User other = (User) obj;
		if (email == null) {
			if (other.email != null) return false;
		} else if (!email.equals(other.email))  return false;
		return true; }
	@Override
	public String toString() {
		return "User [email=" + email + ", nomPrenom=" + nomPrenom
			+ ", motPasse=" + motPasse + "]";}
}
```

Package Metier : Classe News

```
public class News implements Serializable {
	private static final long serialVersionUID= -5316888178005877528L;
	private static SimpleDateFormat formatter =
		new SimpleDateFormat ("yyyy-MM-dd", Locale.FRANCE);
	private static SimpleDateFormat formatter2 =
		new SimpleDateFormat ("dd-MM-yyyy", Locale.FRANCE);
	private String url; private String titre;
	private Date dateInsertion; private User createur;
	private Set<Reaction> reactions= new HashSet<Reaction>();
	public News() { dateInsertion =new Date(); }
	public News(String url, String titre, User createur) {
		this.url = url;  this.titre = titre;
		dateInsertion=new Date(); this.createur = createur;  }
	public String getDateNewsString() {
		return (String) formatter.format(dateInsertion.getTime()); }
	public String getDateNewsString2() {
	    return (String) formatter2.format (dateInsertion.getTime()); }
	public String getUrl() { return url; }
	public void setUrl(String url) { this.url = url; }
	public String getTitre() { return titre; }
	public void setTitre(String titre) { this.titre = titre; }
	public Date getDateInsertion() { return dateInsertion; }
	public void setDateInsertion(Date dateNews) {
		this.dateInsertion = dateNews; }
	public User getCreateur() { return createur; }
	public void setCreateur(User createur){
		this.createur = createur;}
```

```
	public Set<Reaction> getReactions() { return reactions;}
	public void setReactions(Set<Reaction> reactions) {
		this.reactions = reactions; }
	@Override
	public String toString() {
	  return "News [url=" + url +", titre="+titre +", dateNews=" +
	  getDateNewsString2()+",createur="+createur.getNomPrenom()+"]"; }
	@Override
	public boolean equals(Object obj) {
		if (this == obj) return true;
		if (obj == null)  return false;
		if (getClass() != obj.getClass())  return false;
		News other = (News) obj;
		if (url == null) {
			if (other.url != null)  return false;
		} else if (!url.equals(other.url)) return false;
		return true;
		}
	public  int getNombreJaime() {
		try {
			return ServiceMetier.getNombreJaime(this);
		} catch (Exception e) {
			System.out.println(e.getMessage());}
		return 0;
		}
	public  int getNombreJeDeteste() {
		try {
			return ServiceMetier.getNombreJeDeteste(this);
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
	return 0;
	}
}
```

Package Metier : Classe Reaction

```
public class Reaction implements Serializable{
	private static final long serialVersionUID = 1L;
	private static SimpleDateFormat formatter =
		new SimpleDateFormat ("yyyy-MM-dd", Locale.FRANCE);
	private static SimpleDateFormat formatter2 =
		new SimpleDateFormat ("dd-MM-yyyy", Locale.FRANCE);
	private int num; private Date dateReaction;
	private boolean reaction; private User user;
	private News news; public Reaction() {
		dateReaction =new Date(); }
	public Reaction(boolean reaction, User user, News news) {
		this.reaction = reaction; dateReaction =new Date();
		this.user = user;	this.news = news; }
```

```
	public void setNum(int num) { this.num = num;}
	public String getDateReactionString()    {
	return (String) formatter.format (dateReaction.getTime()); }
	public String getDateReactionString2() {
	return(String) formatter2.format (dateReaction.getTime()); }
	public Date getDateReaction() {return dateReaction;}
	public void setDateReaction(Date dateReaction) {
		this.dateReaction = dateReaction;}
	public boolean getReaction() {return reaction;}
	public String getReactionString() {
	if(reaction) return("J'aime");else return ("je Deteste");  }
	public void setReaction(boolean reaction){this.reaction=reaction;}
	public User getUser() { return user;}
	public void setUser(User user) {this.user = user;}
	public News getNews() { return news;}
	public void setNews(News news) {this.news = news;}
	public int getNum() {return num;}
	@Override
	public String toString() {
		return "Reaction [num=" + num + ", dateReaction=" +
			dateReaction + ", reaction=" + reaction + ", user=" +
			user + ", news=" + news + "]";}
	@Override
	public int hashCode() {
		final int prime = 31; int result = 1;
		result = prime * result + ((news == null) ? 0 : news.hashCode());
		result = prime * result + (reaction ? 1231 : 1237);
		result = prime * result + ((user == null) ? 0 : user.hashCode());
		return result;
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (obj == null) return false;
		if (getClass() != obj.getClass()) return false;
		Reaction other = (Reaction) obj;
		if (news == null) {
			if (other.news != null) return false;
		} else if (!news.equals(other.news)) return false;
		if (user == null) {
			if (other.user != null) return false;
			} else if (!user.equals(other.user)) return false;
		return true;
	}
}
```

Package Metier : Fichier de mapping User.hbm.xml

```
<hibernate-mapping>
  <class name="metier.User" table="users">
      <id name="email" column="email">
                  <generator class="assigned"/> </id>
      <property name="nomPrenom" column="nomPrenom" />
      <property name="motPasse" column="motPasse" />
      <set name="news">
            <key column="email"/>
            <one-to-many class="metier.News"/>
      </set>
      <set name="reactions">
            <key column="num"/>
            <one-to-many class="metier.Reaction"/>
      </set>
  </class>
</hibernate-mapping>
```

Package Metier : Fichier de mapping News.hbm.xml

```
<hibernate-mapping>
      <class name="metier.News" table="news">
            <id name="url" column="url">
                        <generator class="assigned"/> </id>
            <property name="titre" column="titre" />
            <property name="dateInsertion" column="dateInsertion" />
            <many-to-one name="createur" column="email"/>
            <set name="reactions">
                  <key column="num"/>
                  <one-to-many class="metier.Reaction"/>
            </set>
        </class>
</hibernate-mapping>
```

Package Metier : Fichier de mapping Reaction.hbm.xml

```
<hibernate-mapping>
  <class name="metier.Reaction" table="reactions">
      <id name="num" column="num">
            <generator class="increment"/> </id>
      <property name="dateReaction" column="dateReaction" />
      <property name="reaction" column="reaction" type="boolean" />
      <many-to-one name="user" column="email"/>
      <many-to-one name="news" column="url"/>
  </class>
</hibernate-mapping>
```

Package Metier : Classe ServiceMetier

```
public class ServiceMetier {
      public ServiceMetier ()  {}
      // Methodes metiers
      public static boolean addUser(User user) throws Exception {
            return ServiceModel.addUser(user); }
      public static User getUserByEmail(String email) throws Exception{
            return ServiceModel.getUser(email);}
       public static User getUser(String email,String motPasse)throws
            Exception{return ServiceModel.getUser(email,motPasse);}
       public static News getNewsByUrl(String url) throws Exception {
            return ServiceModel.getNews(url);}
      public static ArrayList<String> getAllUrl() throws Exception {
            ArrayList<String> urls= new ArrayList<String>();
            for(News news:ServiceModel.getAllNews()) {
                  urls.add(news.getUrl());}
            return urls;
      }
      public static ArrayList<News> getAllNews() throws Exception {
            return (ArrayList<News>)(ServiceModel.getAllNews()); }
      public static ArrayList<Reaction> getAllReaction() throws
                  Exception {
         return (ArrayList<Reaction>)(ServiceModel.getAllReaction());  }
       public static boolean addNews(News news) throws Exception {
            if ( getNewsByUrl( news.getUrl() )!= null)return false;
            ServiceModel.addNews(news);
            return true;}
       public static boolean addReaction(Reaction reaction) throws
            Exception {
            Reaction reaction2 = ServiceModel.getReaction(
                   reaction.getUser(),
            reaction.getNews() );
            if (reaction2==null) ServiceModel.addReaction(reaction);
            else {
                  reaction2.setReaction(reaction.getReaction());
                  ServiceModel.addReaction(reaction2);
            }
            return true;
      }
      public static ArrayList<News> getNewsCreateByUser(User user)
                  throws Exception{
            ArrayList<News> lesNews= new ArrayList<News>();
            for(News news:ServiceModel.getAllNews()) {
              if (news.getCreateur().equals(user)) lesNews.add(news);
            }
            return lesNews;
      }
```

```
      public static ArrayList<Reaction> getReactionByUser(User user)
                  throws Exception{
            ArrayList<Reaction> lesReaction= new ArrayList<Reaction>();
            for(Reaction reaction:ServiceModel.getAllReaction()) {
              if (reaction.getUser().equals(user))lesReaction.add(reaction);
            }
            return lesReaction;
      }
      public static ArrayList<Reaction> getReactionByNews(News news)
                                    throws Exception{
            ArrayList<Reaction> lesReaction= new ArrayList<Reaction>();
            for(Reaction reaction:ServiceModel.getAllReaction()) {
              if (reaction.getNews().equals(news))
                    lesReaction.add(reaction);
            }
            return lesReaction;}
      public static int getNombreJaime(News news) throws Exception{
            int nombreJaime=0;
            for(Reaction reaction:ServiceModel.getAllReaction()) {
              if (reaction.getNews().equals(news)
                  && reaction.getReaction()) nombreJaime++;
            }
            return nombreJaime;}
      public static int getNombreJeDeteste(News news) throws Exception{
            int nombreJeDeteste=0;
            for(Reaction reaction:ServiceModel.getAllReaction()) {
              if (reaction.getNews().equals(news)
                  && !reaction.getReaction()) nombreJeDeteste++;
            }
            return nombreJeDeteste; }
}
```

Page JSP Ajout News

```
<body>
      <% User user=(User) session.getAttribute("user"); %>
      <TABLE style="width:50%" >
            <TR >       <TD colspan =2 align="right" >
            <%=user.getNomPrenom()%>
                  <a href="ServletS">  Se deconnecter ... </a>
            </TD> </TR>
   </TABLE>
   <BR>
   <FORM action = "ServletS" method ="post">
     <TABLE style="width:50%">
      <TR> <TD width="10%" > <b>Email </b>  </TD>
        <TD><input type="text" name="email" style="width:300px" disabled
                  value= <%=user.getEmail()%>> </TD>
      </TR>
```

```
      <TR>  <TD width="10%" > <b>Url </b>  </TD>
        <TD> <input type="text"  name="url" style="width:300px" >
        </TD></TR>
      <TR>
        <TD width="10%"> <b>Titre </b>  </TD>
        <TD > <input type="text"  name="titre" style="width:300px" >
        </TD>
      </TR>
      </TABLE >
      <BR>
      <TABLE style="width:50%; align:center" >
       <TR> <TD>  <input type="submit" name="action" value="Valider"
                                        width="60" height="60" >
            </TD>
            <TD><input type="submit" name="action" value="Ajout reaction"
                                        width="60" height="60" >
            <input type="submit" name="action" value="Affichage news"
                                       width="60" height="60">
            </TD>
      </TR>
      </TABLE>
    </FORM>

    <% session.setAttribute("page", "AjoutNews");
            String erreur= (String) request.getAttribute("erreur");
               if (erreur!=null ){  out.print(erreur);  }
    %>
</body>Page JSP

Page JSP Ajout reaction

<body>
   <% User user=(User) session.getAttribute("user"); %>
   <% ArrayList<String> urls=(ArrayList<String>)
            application.getAttribute("urls"); %>
   <TABLE style="width:50%" >
      <TR > <TD colspan =2 align="right" >
      <%=user.getNomPrenom()%>
      <a href="ServletS">  Se deconnecter ...</a>
      </TD> </TR>
   </TABLE>
   <BR>
   <FORM action = "ServletS" method ="post">
     <TABLE style="width:50%; border-collapse: collapse;" >
      <TR >
      <TD style="width:10%; border: thin solid #6495ed; padding: 5px" >
      <b>News </b>  </TD>
            <TD style="border: thin solid #6495ed; padding: 5px" >
              <SELECT name="listUrl" size="1" >
```

```
                <% for(String url: urls){ %>
                     <OPTION> <%=url %> <%}%>
              </SELECT>
              </TD>
      </TR>
      <TR>
      <TD style="width:10%; border: thin solid #6495ed; padding: 5px">
            <b>Reaction </b>  </TD>
      <TD style="border: thin solid #6495ed; padding: 5px"><input type="radio"
            name="reaction"   value="Jaime" checked > J'aime
      <input type="radio" name="reaction" value="jedeteste" > je deteste
      </TD></TR>
      </TABLE>
       <BR>
       <TABLE style="width:50%; align:center" >
       <TR>
            <TD> <input type="submit" name="action" value="Valider"
                                        width="60" height="60" > </TD>
            <TD > <input type="submit" name="action" value="Ajout news"
                                        width="60" height="60" >
            <input type="submit" name="action" value="Affichage news"
                                       width="60" height="60">
            </TD>
      </TR>
      </TABLE>
     </FORM>
     <% session.setAttribute("page", "AjoutReaction");  %>
</body>
```

Page JSP Affichage News

```
<body>
   <% ArrayList<News> listNews=ServiceMetier.getAllNews(); %>
   <% User user=(User) session.getAttribute("user"); %>
   <p align="right">
      <%=user.getNomPrenom()%>
      <a href="ServletS">  Se deconnecter ... </a>
   </p>
   <TABLE >
      <TR><TH >Date</TH> <TH >Titre</TH> <TH >Url</TH>
      <TH >Nb J'aime</TH> <TH >Nb Je deteste</TH></TR>
      <% for (News news:listNews ) { %>
      <TR>
      <TD > <%=news.getDateNewsString2()%>       </TD>
      <TD>  <%=news.getTitre() %>   </TD>
      <TD > <%=news.getUrl() %>       </TD>
      <TD > <%=news.getNombreJaime() %>    </TD>
      <TD > <%=news.getNombreJeDeteste() %>    </TD>
      </TR>
      <% } %>
```

```
  </TABLE>
  <BR><BR>
  <a     href="AfficheReactions.jsp">Afficher     les     reactions...
     </a><BR><BR>
  <a href="AjoutNews.jsp">Envoyer de nouveaux news... </a><BR><BR>
  <a href="AjoutReaction.jsp">Envoyer de nouvelle reaction... </a>
  <% session.setAttribute("page", "AffichageNews");%>
</body>
```

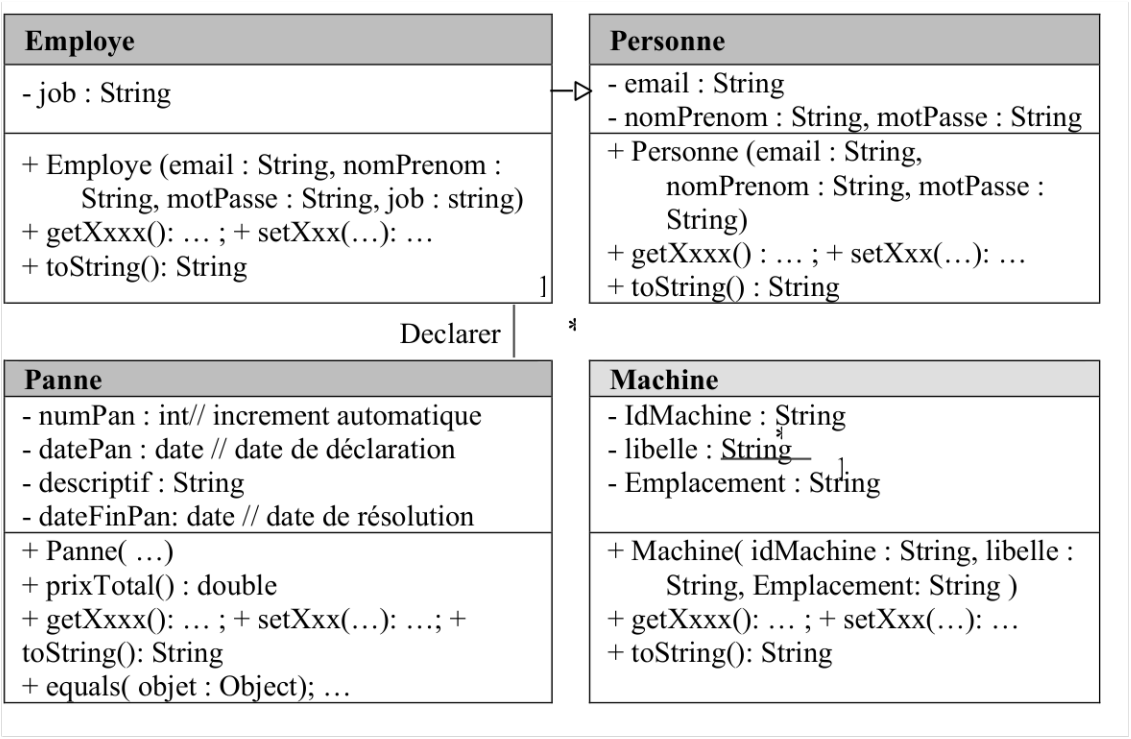
Page JSP Affichage des réactions

```
<body>
 <% ArrayList<Reaction> listReactions= ServiceMetier.getAllReaction();%>
 <% User user=(User) session.getAttribute("user"); %>
 <p align="right">
      <%=user.getNomPrenom()%>
      <a href="ServletS">  Se deconnecter ... </a>
 </p>
 <TABLE >
      <TR>  <TH >Num</TH> <TH >DateReaction </TH><TH >Titre</TH>
      <TH >User</TH> <TH >reaction</TH> </TR>
      <% for (Reaction react:listReactions ) { %>
      <TR>
      <TD > <%=react.getNum()%>       </TD>
      <TD>  <%=react.getNum() %>   </TD>
      <TD > <%=react.getNews().getTitre() %>       </TD>
      <TD > <%=react.getUser().getNomPrenom() %>    </TD>
      <TD > <% if (react.getReaction()) out.print("J'aime");
                  else out.print("je deteste");%>    </TD>


      </TR>
      <% } %>
</TABLE>
<BR><BR>
<a href="AjoutNews.jsp">Envoyer de nouveaux news... </a><BR><BR>
<a href="AjoutReaction.jsp">Envoyer de nouvelle reaction... </a>

<% session.setAttribute("page", "AffichageReaction");%>
</body>
```

TP N°3 : Hibernate et Servlets

On s'intéresse dans ce TP à une application de gestion des déclarations des pannes dans une entreprise de production industrielle.

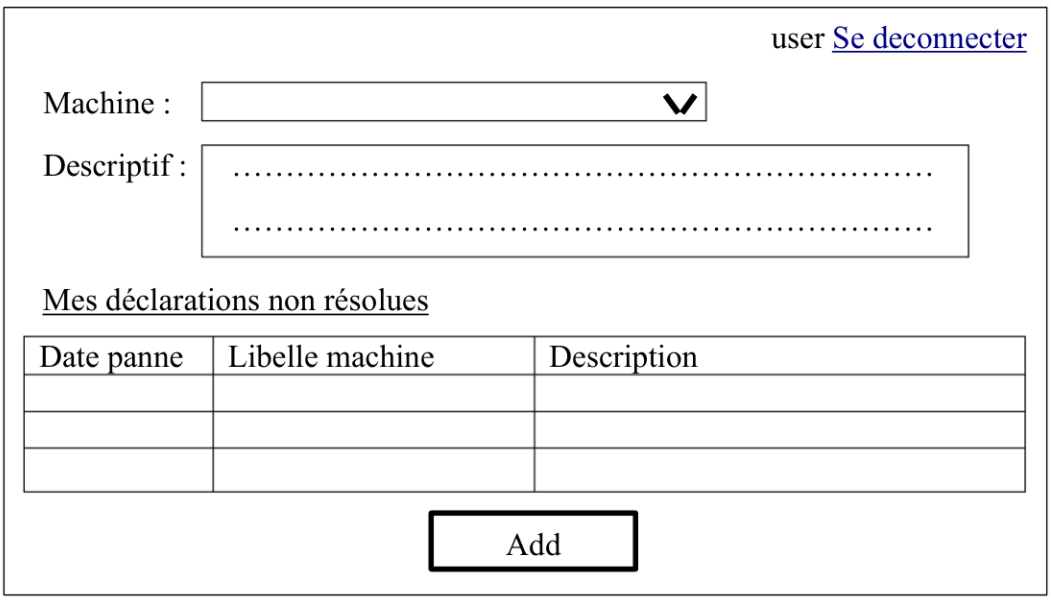
A. Pour des raisons de simplification, on considère que la modélisation des classes métiers représentée par le diagramme suivant :



B. On considère que les informations des pannes sont enregistrées dans une base de données mySQL nommée « GestPanne ». L'accès à cette dernière effectué en utilisant Hibernate. Ainsi, une classe « ServiceDAO » regroupe tous les services liés à la base de données. Nous disposons d'une classe singleton « HibernatUtil » qui assure une unique connexion à la base de données. La structure de la classe « ServiceDAO » est donnée par le diagramme suivant (toutes les méthodes sont considérées static) .

ServiceDAO
+ getAllPersonnes(): List<Personne> DAO « getPanneParMachine » et « addEmploye ».
+ getAllEmployes(): List<Personne> // //liste de tous les employés
+ getPersonne(email :String): Personne //test exitance d'une Personne
+ getPersonne(email :String, motPass : String) : Personne
+ addEmploye(employe : Employe) : boolean
+ getAllMachine(): List<Machine> ; + getMachine(IdMachine : String): Machine
+ addMachine(machine : Machine) : boolean
+ getAllPannes(): List<Panne> // //liste de tous les pannes
+ getPanneParDate(datePan : Date): List<Panne>
+ getPanneDeclareParPersonne(email : String): List<Panne>
+ getPannePersonneNonResolu (email : String): List<Panne> // dateFinPan est non null
+ getPanneParMachine(IdMachine : String): List<Panne>
+ addPanne(panne: Panne) : boolean
+ …

C. Dans cette partie, nous nous intéressons à la couche Web et la couche présentation de notre application web. Nous disposons d'une classe « ServiceMetier » dont la structure est similaire à celle de la classe « ServiceDAO » (on suppose qu'méthode de la classe « ServiceDAO » est appelée par la méthode de la clase « ServiceMetier » qui lui correspond et ayant la même signature). Les différents traitements sont réalisés par Struts. Pour des considérations de simplification, nous nous limitons à la page JSP « declarationPanne.jsp » suivante (on suppose que le déclarant de la panne est défini par un objet employe placé dans la scope « session ») :



TP N°4 : Hibernate et Struts

Dans ce TP nous nous intéressons à une application de gestion de la bibliothèque d'un établissement universitaire en se basant sur l'utilisation d'Hibernate et Struts. Cette application permet aux personnes de l'établissement (étudiants, professeurs et administrateurs) de réaliser les opérations (inscription, consultation, emprunt, …) relatives à la gestion de la bibliothèque. Nous nous limitons, dans ce qui suit, à certaines taches réalisées par l'étudiant.

L'application de gestion de la bibliothèque offre aux étudiants une interaction facile avec les livres disponibles. Premièrement chaque étudiant ne peut pas consulter les livres disponibles dans la bibliothèque sauf s'il est déjà inscrit, pour cela l'étudiant doit remplir un formulaire qui contient les champs suivants :

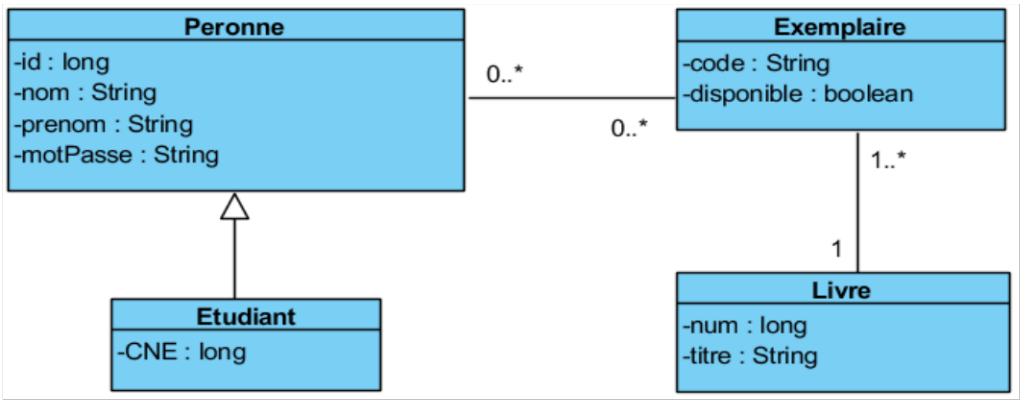
- CNE
- Prénom
- Nom
- Mot de passe

Après l'étape d'inscription, chaque étudiant peut s'authentifier en utilisant son CNE comme login et son mot de passe. L'authentification permettra aux étudiants de consulter la liste des livres disponibles. Nous considérons que chaque étudiant n'a pas le droit de supprimer ni d'ajouter un livre à la base de données, ces deux rôles importants sont réservés à l'administrateur de l'application (la partie administration de l'application ne fait pas l'objet de cet examen). L'étudiant peut consulter les livres disponibles et ceci par:

- Code de l'exemplaire
- Titre

L'emprunt des exemplaires de livres s'effectue en fournissant uniquement le code de l'exemplaire alors que le numéro CNE de l'étudiant est affiché automatiquement (un attribut « etudiant » est conservé dans la scope session de l'application). Pour des raisons de simplification on se limite à ces deux éléments (aucune information telle que la date de l'emprunt ou autre n'est pris en considération).

La figure suivante représente une partie du diagramme de classe de ce système :



Nous nous limitons dans cet examen aux pages jsp suivantes :

authentification.jsp

Login
Mot de passe
Statut ◉ Etudiant ○ professeur
S'inscrire …

inscription.jsp

CNE
Nom
Prénom
Mot de passe
Valider

consultDisponible.jsp

nomPrenom : se deconnecter

Code	Titre
…	…
…	…

Emprunter un exemplaire de livre …

emprunt.jsp

nomPrenom : se deconnecter
CNE
Code Exemplaire
Valider

Après la validation d'un emprunt, la page «emprunt.jsp» est réaffichée pour permettre l'ajout d'un nouvel emprunt.

A. Nous considérons que le projet à réaliser contient un package « metier » comportant les classes métiers « pojo », les fichiers de mapping correspondants et une classe « ServiceMetier » dont la structure est donnée par le diagramme suivant (toutes les méthodes sont considérées static) :

metier : ServiceMetier
+ getEtudiant(CNE: long, motPasse:String) : Etudiant //test d'existance d'un etudiant + addEtudiant(etudiant : Etudiant) : boolean + getExemplaireDispo() : list<Object> // liste des codes et titre des exemplaires disponibles + getCodeExemplaireDispo() : list<String> // liste des codes des exemplaires disponibles + addEmprunt(CNE : String, codeExemplaire: String) : boolean + …

1- En se basant sur le diagramme de classe du système (voir ci-dessus), définir les classes métiers (pojo) nécessaires à l'application.

2- Ecrire les fichiers de mapping correspondants.

B. Soit un package dao contenant la classe singleton «HibernateUtil» qui assure la construction d'une unique SessionFactory et la classe «ServiceModel» regroupant tous les services Hibernate(HQL, Criteria,…) liés à la base de données. Le diagramme de cette 2ème classe est le suivant (les méthodes sont considérées static):

dao : ServiceModel
+ getEtudiant(CNE: long, motPasse:String) : Etudiant //Test d'existance + addEtudiant(etudiant : Etudiant) : boolean // ajout d'un nouveau etudiant + getExemplaireDispo() : list<Object> // liste des codes et titres des exemplaires disponibles + getCodeExemplaireDispo() : list<String> // liste des codes des exemplaires disponibles + addEmprunt(CNE : String, codeExemplaire: String) : boolean + …

C. Nous supposons que les différentes actions de l'application (authentification, inscription, emprunt,…) sont décrites par le fichier de configuration « struts.xml » et que leur méthodes associées sont définies dans la classe « ServiceAction » du package web.

1- En se limitant aux pages jsp décrites précédemment, écrire le fichier de configuration «struts.xml».

2- Définir les méthodes de la classe « ServiceAction » à exécuter pour assurer les différents actions liées aux différents page JSP et définir ces pagves

Elément de correction

Fichier Web.xml

```
<filter>
  <filter-name>struts2</filter-name>
  <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Package DAO : Classe

```
public class HibernateUtil {
    public static final SessionFactory sessionFactory;
    static {
        try {
           sessionFactory = new Configuration().configure
                  ("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex); }
    }
    public static SessionFactory getSessionFactory(){return sessionFactory;}
}
```

Package DAO : Classe ServiceModel

```
public class ServiceModel {
     public static boolean add(Object obj) {
      Session session=HibernateUtil.getSessionFactory().getCurrentSession();
      session.beginTransaction(); session.save(obj);
      session.getTransaction().commit(); return true;}
     public static boolean addEmprunt(long idPersonne,StringcodeExemplaire){
      try {
      Session session=HibernateUtil.getSessionFactory().getCurrentSession();
      session.beginTransaction();
      // Charger la personne et l'exemplaire de l'emprunt
      Personne personne=(Personne) session.load(Personne.class,idPersonne);
      Exemplaire exemplaire=(Exemplaire)session.load(
                Exemplaire.class,codeExemplaire);
      //Enregistrer l'emprunt
      exemplaire.getPersonnes().add(personne);
      exemplaire.setDisponible(false); session.getTransaction().commit();
      } catch (Exception e) {  return false; }
      return true;
     }
```

```
      public static Etudiant getEtudiant(String cne) {
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
       session.beginTransaction();
       Query req=session.createQuery ("from Personne where cne like :cne");
       req.setString("cne", cne);
       if (req.list().size()==0) return (null);
       else return (Etudiant) req.list().get(0);
      }
      public static Etudiant getEtudiant(String cne,String motPasse)
                              throws Exception{
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
       session.beginTransaction();
       Criteria criteria = session.createCriteria(Etudiant.class);
       Criterion critere = Restrictions.like("cne", cne);
       criteria.add(critere);
       critere = Restrictions.like("motPasse", motPasse);
       criteria.add(critere);
       if (criteria.list().size()==0) return (null);
       else return (Etudiant) criteria.list().get(0);
      }
      public static boolean addEtudiant(Etudiant etudiant){
                  return add(etudiant);}
      public static boolean addLivre(Livre livre) { return add(livre); }
      public static boolean addExemplaire(Exemplaire exemplaire) {
                  return add(exemplaire); }
      public static List<Object> getExemplaireDispo() throws Exception{
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
       session.beginTransaction();
       Query req=session.createQuery(
            "Select ex.code,liv.titre FROM Exemplaire ex" + " JOIN ex.livre liv
             where ex.disponible=true");
       return (req.list()) ; }
      public static List<String> getCodeExemplaireDispo() throws Exception{
       Session session=HibernateUtil.getSessionFactory().getCurrentSession();
       session.beginTransaction();
       Query req=session.createQuery(
                     "select code from Exemplaire where disponible=true");
       return ( req.list()) ;  }
}
```

Package Metier : Classe Personne

```
public abstract class Personne implements Serializable{
      private long id;
      private String nom,prenom, motPasse;
      private Set<Exemplaire> exemplaires= new HashSet<Exemplaire>();
      public Personne() {     }
      public Personne(String nom, String prenom, String motPasse) {
            this.nom = nom; this.prenom = prenom;this.motPasse = motPasse;}
```

```
	public long getId() { return id;}
	public void setId(long id) { this.id = id; }
	public String getNom() { return nom;}
	public void setNom(String nom) {    this.nom = nom; }
	public String getPrenom() { return prenom; }
	public void setPrenom(String prenom) {    this.prenom = prenom; }
	public String getMotPasse() { return motPasse; }
	public void setMotPasse(String motPasse) {this.motPasse = motPasse;}
	public String getNomPrenom() {return (nom + " "+ prenom) ; }
	public Set<Exemplaire> getExemplaires() {return exemplaires;}
	public void setExemplaires(Set<Exemplaire> exemplaires) {
		this.exemplaires = exemplaires;}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (obj == null)  return false;
		if (getClass() != obj.getClass())  return false;
		Personne other = (Personne) obj;
		if (id != other.id) return false;
		return true;
	}
	@Override
	public String toString() {
	 return " id="+i+",nom="+nom+",prenom="+prenom+",motPasse="+motPasse;}
}
```

Package Metier : Classe Etudiant

```
public class Etudiant extends Personne{
	private String cne;
	public Etudiant() { super();}
	public Etudiant(String cne, String nom, String prenom, String motPasse){
		super(nom, prenom, motPasse);	this.cne = cne;   }
	public String getCne() {	return cne; }
	public void setCne(String cne) {    this.cne = cne;   }
	@Override
	public String toString() {
		return "Etudiant [cne=" + cne + super.toString() + "]";}
@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (!super.equals(obj))	return false;
		if (getClass() != obj.getClass())  return false;
		Etudiant other = (Etudiant) obj;
		if (cne == null) {
			if (other.cne != null)  return false;
		} else if (!cne.equals(other.cne)) return false;
		return true;}
}
```

Package Metier : Classe Livre

```
public class Livre implements Serializable {
      private long num;
      private String titre;
      private Set<Exemplaire> exemplaires= new HashSet<Exemplaire>();
      public Livre() {  }
      public Livre(long num, String titre) {
            this.num = num;   this.titre = titre;}
      public long getNum() {  return num; }
      public void setNum(long num) {      this.num = num;   }
      public String getTitre() {    return titre;}
      public void setTitre(String titre) {      this.titre = titre;}
      public Set<Exemplaire> getExemplaires() {return exemplaires;}
      public void setExemplaires(Set<Exemplaire> exemplaires) {
            this.exemplaires = exemplaires;}
      public boolean equals(Object obj) {
            if (this == obj)  return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())  return false;
            Livre other = (Livre) obj;
            if (num != other.num)   return false;
            if (titre == null) {
                  if (other.titre != null)return false;
            } else if (!titre.equals(other.titre))   return false;
            return true;
      }
      @Override
      public String toString(){return "Livre[num="+num+",titre="+titre+"]";}
}
```

Package Metier : Classe Exemplaire

```
public class Exemplaire {
      private String code;private boolean disponible;
      private Livre livre;
      private Set<Personne> personnes= new HashSet<Personne>();
      public Exemplaire() {}
      public Exemplaire(String code, Livre livre ) {
            this.code = code; this.disponible = true; this.livre = livre;}
      public String getCode() {     return code;}
      public void setCode(String code) {this.code = code;}
      public boolean isDisponible() {     return disponible;      }
      public void setDisponible(boolean disp){this.disponible = disp;}
      public Livre getLivre() {return livre;}
      public void setLivre(Livre livre) { this.livre = livre;}
      public Set<Personne> getPersonnes() {return personnes;}
      public void setPersonnes(Set<Personne> p) {     this.personnes = p;}
```

```
      @Override
      public boolean equals(Object obj) {
            if (this == obj)return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())  return false;
            Exemplaire other = (Exemplaire) obj;
            if (code == null) {
                  if (other.code != null) return false;
            } else if (!code.equals(other.code))return false;
            return true;}
      @Override
      public String toString() {
        return "Exemplaire[code=" + code + ", disponible="+disponible+"]";}
}
```

Package de Metier : Fichier de mapping Personne.hbm.xml

```
<hibernate-mapping>
<class      name="metier.Personne"      table="personnes"      discriminator-
      value="Personne">
                  <id name="id" column="id">
                        <generator class="increment"/> </id>
            <discriminator column="TYPE" type="string" />
            <property name="nom" column="nom" />
            <property name="prenom" column="prenom" />
            <property name="motPasse" column="motPasse" />
            <set name="exemplaires" table="Emprunt" inverse="true">
                  <key column="IdPersonne"/>
                  <many-to-many column="codeExemplaire"
                              class="metier.Exemplaire"/>
            </set>
            <subclass name="metier.Etudiant" discriminator-value="Etudiant">
                  <property name="cne" column="cne" />
            </subclass>
</class> </hibernate-mapping>
```

Package Metier : Fichier de mapping Livre.hbm.xml

```
<hibernate-mapping>
      <class name="metier.Livre" table="livres">
            <id name="num" column="num">
                        <generator class="increment"/> </id>
            <property name="titre" column="titre" />
            <set name="exemplaires">
                  <key column="num"/>
                  <one-to-many class="metier.Exemplaire"/>
            </set>
        </class>
</hibernate-mapping>
```

Package Metier : Fichier de mapping Exemplaire.hbm.xml

```
<hibernate-mapping>
      <class name="metier.Exemplaire" table="exemplaires">
            <id name="code" column="code">
                        <generator class="assigned"/> </id>
            <property name="disponible" column="disponible" type="boolean" />
            <many-to-one name="livre" column="num"/>
            <set name= "personnes" table="Emprunt"  >
                  <key column="codeExemplaire"/>
                  <many-to-many column="idPersonne" class="metier.Personne"/>
            </set>
        </class>
</hibernate-mapping>
```

Package Metier : ServiceMetier

```
public class ServiceMetier {
      public ServiceMetier ()  {}
      // Methodes metiers login et inscription User
      public static Etudiant getEtudiantByCNE(String cne) throws Exception {
            return ServiceModel.getEtudiant(cne);}

      public static Etudiant getEtudiant(String cne, String motPasse) throws
                                    Exception {
            return ServiceModel.getEtudiant(cne, motPasse);}
      public static boolean addEtudiant(Etudiant etudiant) throws Exception {
            return ServiceModel.addEtudiant(etudiant);}
      public static List<Object> getExemplaireDispo() throws Exception{
            return ServiceModel.getExemplaireDispo();  }
      public static List<String> getCodeExemplaireDispo() throws Exception{
            return ( ServiceModel.getCodeExemplaireDispo()) ; }
      public static boolean addEmprunt(long idPersonne,String codeExemplaire){
            return (ServiceModel.addEmprunt(idPersonne,  codeExemplaire) );}
}
```

Fichier de configuration hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
    <!-- Database connection settings -->
    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="connection.url">
            jdbc:mysql://localhost:3306/bdbibleo </property>
    <property name="connection.username">root</property>
    <property name="connection.password"></property>
    <!-- JDBC connection pool (use the built-in) -->
    <property name="connection.pool_size">1</property>
    <!-- SQL dialect -->
```

```
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.ejb.cfgfile">
                  /org/hibernate/ejb/test/hibernate.cfg.xml</property>
    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>
    <!-- Disable the second-level cache  -->
   .<property
     name="cache.provider_class">org.hibernate.cache.NoCacheProvider</proper
     ty>
       <!-- Echo all executed SQL to stdout -->
       <property name="show_sql">true</property>
       <!-- Update the database schema on startup -->
       <property name="hbm2ddl.auto">create </property>
       <!-- mapping resource many to many -->
       <mapping resource="metier/Personne.hbm.xml"/>
       <mapping resource="metier/Livre.hbm.xml"/>
       <mapping resource="metier/Exemplaire.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Fichier struts.xml

```
<struts>
    <constant name="struts.enable.DynamicMethodInvocation" value="false" />
    <constant name="struts.devMode" value="false" />
     <package name="web"  namespace="/" extends="struts-default">
       <!-- Action de l'action de réference -->
       <default-action-ref name="login"/>
       <action name="login">
         <result>/login.jsp</result> </action>
       <action name="validerlogin" class="web.ServiceAction" method="login">
           <result name="success" >/AjoutEmprunt.jsp</result>
           <result name="input">/login.jsp</result>
           <result name="error">/login.jsp</result>
       </action>
       <action name="inscription"> <result>/inscription.jsp</result>
       </action>
       <action name="validerinscription" class="web.ServiceAction"
               method="inscription">
           <result name="success" >/login.jsp</result>
           <result name="input">/inscription.jsp</result>
           <result name="error">/inscription.jsp</result>
       </action>
       <action name="ajoutEmprunt" class="web.ServiceAction"
               method="GetCodeExemplaireDisponible">
           <result>/AjoutEmprunt.jsp</result>
       </action>
       <action name="validerAjoutEmprunt" class="web.ServiceAction"
               method="ajoutEmprunt">
           <result  >/AjoutEmprunt.jsp</result>
```

```
        </action>
        <action name="ConsultDisponible"  class="web.ServiceAction"
                method="GetExemplaireDisponible">
            <result>/ConsultDisponible.jsp</result>
        </action>
      </package>
 </struts>
```

Package Web : Classe ServiceAction

```
public class ServiceAction extends ActionSupport {
      static {
      HibernateUtil.getSessionFactory().openSession(); try {
            codeExemplairesDispo=ServiceMetier.getCodeExemplaireDispo();
            exemplairesDispo=ServiceMetier.getExemplaireDispo();
      } catch (Exception e) { e.printStackTrace();}
      }
      private static final long serialVersionUID = 1L;
      //parametttre de login et d'inscription
      private String ,nom, prenom, motPasse, codeExemplaire;
      private  static List<String>  codeExemplairesDispo;
      private  static List<Object>  exemplairesDispo;
      // Methodes d'actions
      public String inscription()  {
        if(nom.equals("") ) {
            addFieldError("nom", "le nom ne doit pas être vide");
            return "input";   }
        if(prenom.equals("")) {
            addFieldError("prenom", "le prenom ne doit pas être vide");
            return "input";   }
        try {
            if(ServiceMetier.getEtudiantByCNE(cne)!=null) {
              addFieldError("cne", "Ce CNE existe deja"); return "input";}
            Etudiant etudiant= new Etudiant(cne, nom, prenom, motPasse);
            ServiceMetier.addEtudiant(etudiant);
            cne=null;  nom=null; prenom=null;  motPasse=null;
            ServletActionContext.getRequest().getSession().setAttribute
                  ("etudiant", null);
            return "success";
        } catch (Exception e) {e.printStackTrace();}
      return "error";
      }
      public String login()  {
        try {
            Etudiant etudiant= ServiceMetier.getEtudiant(cne, motPasse);
            if (etudiant!=null) {
            ServletActionContext.getRequest().getSession().setAttribute
                  ("etudiant", etudiant);
            return "success";  }
```

```
        } catch (Exception e) {addFieldError("erreur", e.getMessage());}
        addFieldError("cne", "Ce CNE n'existe pas");  return "error";
      }
      public String ajoutEmprunt() {
        try {
            HttpServletRequest request=ServletActionContext.getRequest();
            HttpSession session=request.getSession();
            Etudiant etudiant= (Etudiant) session.getAttribute("etudiant") ;
            ServiceMetier.addEmprunt(etudiant.getId(), codeExemplaire);
            codeExemplairesDispo=ServiceMetier.getCodeExemplaireDispo();
            exemplairesDispo=ServiceMetier.getExemplaireDispo();
            return "success";
        } catch (Exception e) { e.printStackTrace();}
        return "error"; }
      public String GetCodeExemplaireDisponible() {return "success";}
      public String GetExemplaireDisponible() { return "success"; }
      // getter et setter
      public String getCne() {return cne;}
      public void setCne(String cne) {this.cne = cne;}
      public String getNom() {return nom; }
      public void setNom(String nom) {this.nom = nom;}
      public String getPrenom() {return prenom;}
      public void setPrenom(String prenom) {this.prenom = prenom;}
      public String getMotPasse() { return motPasse;}
      public void setMotPasse(String motPasse) {this.motPasse = motPasse;}
      public List<String> getCodeExemplairesDispo() {
            return codeExemplairesDispo;  }
      public List<Object> getExemplairesDispo() {return exemplairesDispo;}
      public String getCodeExemplaire() {return codeExemplaire;}
      public void setCodeExemplaire(String codeExemplaire) {
            this.codeExemplaire = codeExemplaire;}
}
```

Page Login.jsp

```
<%@ taglib prefix ="s" uri="/struts-tags" %>
<body> <center> <div id="formulaire">
      <s:form method ="post" action="validerlogin">
        <s:textfield name="cne" id="cne" label="Login" labelposition="left">
        </s:textfield>
        <s:textfield name="motPasse" id="motPasse"
            label="Mot de Passe" labelposition="left"> </s:textfield>
            <s:submit value = "OK"> </s:submit>
      </s:form>
      <a href="inscription.action">S'inscrire ... </a></a><br/
</div> </center> </body>
```

Page Inscription.jsp

```
<body> <center><div >
      <s:form method ="post" action="validerinscription">
        <s:textfield name="cne" id="cne" label="CNE" labelposition="left">
        </s:textfield>
        <s:textfield name="nom" id="nom" label="Nom " labelposition="left">
        </s:textfield>
        <s:textfield name="prenom" id="prenom"
            label="Prenom" labelposition="left"> </s:textfield>
            <s:textfield name="motPasse" id="motPasse"
                  label="Mot de Passe" labelposition="left">
            </s:textfield>
            <s:submit value = "OK"> </s:submit>
      </s:form>
</div> </center>  </body>
```

Page AJoutEmprunt.jsp

```
<body><center>    <div >
      <p style="text-align:right">
      <a href="login.action">
            <s:property value="#session.etudiant.getNomPrenom()"/>
                  : Se deconnecter ... </a>
      <s:form method ="post" action="validerAjoutEmprunt">
            <s:textfield name="cne" id="cne"
                 value= "%{#session.etudiant.cne}" readonly="true"
                  label="CNE" labelposition="left" >
            </s:textfield>
            <s:select name="codeExemplaire" label="Code Exemplaire"
                  list="codeExemplairesDispo"/>
            <s:submit value = "valider"> </s:submit>
      </s:form>
      <BR><BR>
      <a href="ConsultDisponible.action">
                  Consulter les exemplaires disponibles... </a>
</div> </center> </body>
```

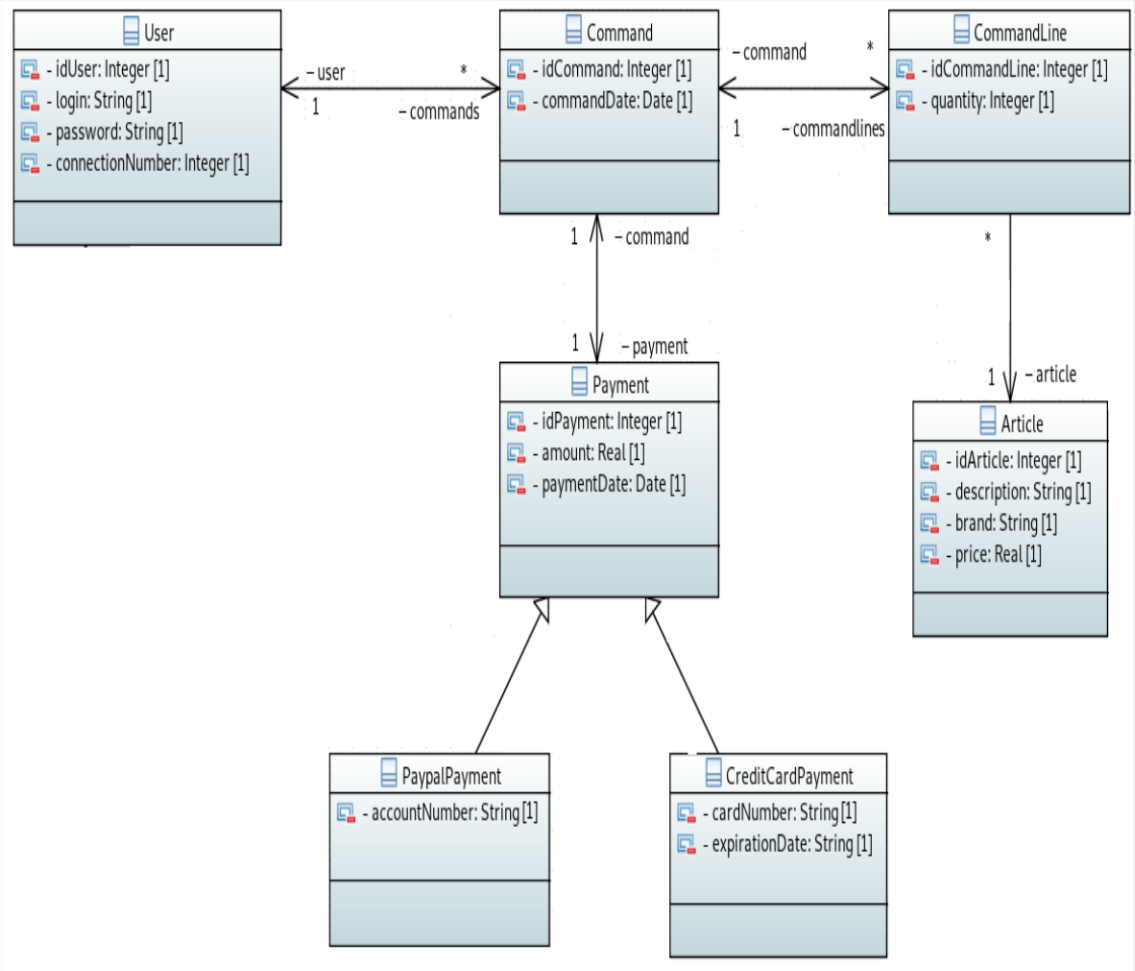
Page ConsultDiponible.jsp

```
<body>
      <p style="text-align:right">
      <a href="login.action">
         <s:property value="#session.etudiant.getNomPrenom()"  />
                  : Se deconnecter ... </a>
      <s:if test="%{exemplairesDispo.size()>0 }">
      <TABLE >
        <TR> <TH >Code</TH> <TH >Titre</TH> </TR>
        <s:iterator value="exemplairesDispo" var="exemplaire">
```

```
   <TR> <TD>  <s:property value="#exemplaire[0]"/>  </TD>
        <TD>  <s:property value="#exemplaire[1]"/>  </TD>
        <!--<TD>  <s:property value="titre"/>  </TD>  --> </TR>
  </s:iterator>
</TABLE>
</s:if>
<s:else >    Aucune Emprunt </s:else>
<BR><BR>
<a href="ajoutEmprunt.action">Emprunt un exemplaire de livre... </a>
</body>
```

TP N°5 : Spring MVC

Dans ce sujet nous nous intéressons à un site E-Commerce permettant aux utilisateurs de passer des commandes d'articles et d'effectuer les paiements correspondants (2 paiements sont supportés : le paiement par carte et le paiement PayPal). On suppose que le projet relatif à ce site sera réalisé en se basant sur les Frameworks Hibernate et Spring MVC. Les informations des ventes sont enregistrées dans une base de données MySQL nommée « dbVentes ». Le diagramme de classe du système étudié est le suivant :



1- Créer un package « entities » comportant les classes du diagramme ci-dessus et les fichiers de mapping correspondants. Puis définir le fichier de configuration Hibernate.cfg.xml

2- Créer un package dao contenant une classe singleton « HibernateUtil » qui assure la construction d'une unique SessionFactory , une interface « IServiceDao » et la classe « ServiceDao » qui l'implémente. La classe « ServiceDao » regroupe les services Hibernate liés à la base de données « dbVentes ».

dao : ServiceDao
+ getUser(login: String, password : String) : User //test d'existence d'authentification + addUser(user : User) : Boolean + addCommande(commande : Commande) : Boolean + addArticle(article : Article) : Boolean + addArticleCommande(idCommande:Integer,idArticle:Integer,quantite:Integer):Boolean + addPaiement(commande : Commande, paiement : Paiement) : Boolean + …

3- Créer un package métier contenant l'interface « IServiceMetier » et la classe « ServiceMetier » qui l'implémente. La classe « ServiceMetier » est similaire à la classe « ServiceDao » et que chaque méthode de cette dernière est appelée par la méthode de la classe « ServiceMetier » qui lui correspond et ayant la même signature (une dépendance de l'interface « IServiceDao » sera injectée par code dans la classe « ServiceMetier »).

4- Définir les fichiers de configuration du contexte de l'application, le fichier d'injection des dépendances et le fichier de déploiement « web.xml »

5- Définir les différentes page JSP permettant de réaliser les différentes fonctionnalités du projet.

6- Définir la classe « ServiceController » qui définit les différentes actions à exécuter et les méthodes qui leur sont associées.

Elément de correction

Package Etities: Classe User

```
public class User {
      private Integer idUser; private String login;private String password;
      private Integer ConnectionNumber; private Set<Command> listCommandes;
      public User() {}
      public User(Integer idUser,String login,String password,Integer
                  connectionNumber) {
            this.idUser = idUser;this.login = login;this.password = password;
            ConnectionNumber = connectionNumber;
            this.listCommandes= new HashSet<Command>();}
      public Integer getIdUser() {  return idUser;}
      public void setIdUser(Integer idUser) {this.idUser = idUser;}
      public String getLogin() {return login;}
      public void setLogin(String login) {this.login = login;}
      public String getPassword() { return password;}
      public void setPassword(String password){this.password=password  }
      public Integer getConnectionNumber() {return ConnectionNumber;}
      public void setConnectionNumber(Integer connectionNumber) {
            ConnectionNumber = connectionNumber;}
      public Set<Command> getListCommandes() {  return listCommandes;}
      @Override
      public String toString() {
        return "User[idUser="+idUser+",login="+login+",password="+password+
            ", ConnectionNumber="+ConnectionNumber + "]";}
      @Override
      public int hashCode() {
            final int prime = 31;   int result = 1;
            result = prime * result + ((login == null) ? 0 : login.hashCode());
            return result;
      }
      @Override
      public boolean equals(Object obj) {
            if (this == obj)  return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())  return false;
            User other = (User) obj;
            if (login == null) {
                  if (other.login != null) return false;
                  } else if (!login.equals(other.login))
            return false;
            return true;
      }
      public void setListCommandes(Set<Command> listCommandes) {
            this.listCommandes = listCommandes;}
}
```

Package Etities: Classe Article

```
public class Article implements Comparable<Article>{
      private Integer idArticle; private String description;
      private String brand;private Float price;
      private Set<CommandeLine> listCommandeLine;
      public Article(Integer idArticle,String description,String brand,
            Float price) {
            this.idArticle = idArticle;this.description = description;
            this.brand = brand;this.price = price;
            listCommandeLine=new HashSet<CommandeLine>();
      }
      public Article() {listCommandeLine=new HashSet<CommandeLine>();}
      public Integer getIdArticle() {return idArticle;}
      public void setIdArticle(Integer idArticle){this.idArticle = idArticle }
      public String getDescription() {return description;}
      public void setDescription(String descr){this.description = descr;}
      public String getBrand() {return brand;}
      public void setBrand(String brand) {this.brand = brand;}
      public Float getPrice() {return price;}
      public void setPrice(Float price) { this.price = price;}
      public Set<CommandeLine> getListCommandeLine(){
            return listCommandeLine;}
      @Override
      public String toString() {
            return "Article [idArticle=" + idArticle + ", description=" +
            description + ", brand=+ brand + ", price="+ price + "]";  }
      @Override
      public int hashCode() {
        final int prime = 31;int result = 1;
        result=prime*result+((idArticle==null)?0:idArticle.hashCode());
            return result;}
      @Override
      public boolean equals(Object obj) {
            if (this == obj)  return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())  return false;
            Article other = (Article) obj;
            if (idArticle == null) {
                  if (other.idArticle != null)  return false;
                  } else if (!idArticle.equals(other.idArticle))return false;
            return true;}
      @Override
      public int compareTo(Article o){return(int)(this.price-o.price); }
      public void setListCommandeLine(Set<CommandeLine> listCommandeLine) {
            this.listCommandeLine = listCommandeLine;}
}
```

Package Etities: Classe Command

```
public class Command {
      private Integer idCommand; private Date commandDate;
      private User user;  private Set<CommandeLine> listCommandeLine;
      public Command(Integer idCommand, Date commandDate, User user) {
        this.idCommand = idCommand;this.commandDate = commandDate;
        this.user=user;this.listCommandeLine=new HashSet<CommandeLine>();    }
      public Command(){this.listCommandeLine=new HashSet<CommandeLine>();}
      public Integer getIdCommand() {return idCommand;     }
      public void setIdCommand(Integer idCommand){this.idCommand=idCommand;}
      public Date getCommandDate() {return commandDate;}
      public void setCommandDate(Date commandDate){
                  this.commandDate=commandDate; }
      public User getUser() { return user;}
      public void setUser(User user) {    this.user = user; }
      public Set<CommandeLine> getListCommandeLine(){return listCommandeLine;}
      @Override
      public int hashCode() {
            final int prime = 31;   int result = 1;
            result=prime*result+((idCommand==null)?0:idCommand.hashCode());
            return result;}
      @Override
      public boolean equals(Object obj) {
            if (this == obj)  return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())  return false;
            Command other = (Command) obj;
            if (idCommand == null) {
                  if (other.idCommand != null) return false;
            } else if (!idCommand.equals(other.idCommand)) return false;
            return true; }
      @Override
      public String toString() {
            return "Command [idCommand=" + idCommand + ", commandDate=" +
                  commandDate + ", user=" + user.getLogin() + "]";}
      public void setListCommandeLine(Set<CommandeLine> listCommandeLine) {
                  this.listCommandeLine = listCommandeLine;  }
}
```

Package Etities: Classe CommandLine

```
public class CommandeLine {
      private Integer idCommandLine;private Integer quantite;
      private Command command; private Article article;
      public CommandeLine(Integer idCommandLine, Integer quantite,
            Command command, Article article) {
            this.idCommandLine = idCommandLine;this.quantite = quantite;
            this.command = command; this.article = article; }
```

```
	public CommandeLine(Integer quantite,Command command,Article article){
		this.quantite=quantite;this.command=command;
		this.article=article;   }
	public CommandeLine() {}
	public Integer getIdCommandLine() { return idCommandLine;}
	public void setIdCommandLine(Integer idCommandLine) {
		this.idCommandLine = idCommandLine;}
	public Integer getQuantite() {return quantite;}
	public void setQuantite(Integer quantite){this.quantite=quantite;}
	public Command getCommand() { return command;}
	public void setCommand(Command command) { this.command = command; }
	public Article getArticle() { return article;}
	public void setArticle(Article article) { this.article = article; }
	@Override
	public int hashCode() {
		final int prime = 31;   int result = 1;
		result=prime*result+((article==null)?0:article.hashCode());
		result=prime*result+((command==null)?0:command.hashCode());
		return result; }
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (obj == null)  return false;
		if (getClass() != obj.getClass())  return false;
		CommandeLine other = (CommandeLine) obj;
		if (article == null) {
			if (other.article != null)return false;
		} else if (!article.equals(other.article))return false;
		if (command == null) {
			if (other.command != null)return false;
		} else if (!command.equals(other.command))return false;
		return true;}
	@Override
	public String toString() {
	  return "CommandeLine[idCommandLine="+idCommandLine+",command=
		"+command.getIdCommand()+", article="+article.getDescription()
			+ ", quantite="+quantite+"]";      }
}
```

Package Etities: Classe Payment

```
public abstract class Payment {
	protected Integer idPayment;  protected Float amount;
	protected Date paymentDate; protected Command command;
	public  Payment(Integer  idPayment,  Float  amount,  Date  paymentDate,
	Command command) {
		this.idPayment = idPayment;this.amount = amount;
		this.paymentDate = paymentDate; this.command = command;}
	public Payment() {}
	public Integer getIdPayment() {return idPayment;}
```

```
	public void setIdPayment(Integer idPayment){this.idPayment=idPayment;}
	public Float getAmount() {return amount;}
	public void setAmount(Float amount) {this.amount = amount; }
	public Date getPaymentDate() {return paymentDate;}
	public void setPaymentDate(Date pDate){this.paymentDate= pDate;}
	@Override
	public int hashCode() {
		final int prime = 31;int result = 1;
		result=prime*result+((idPayment==null)?0:idPayment.hashCode());
		return result; }
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (obj == null)  return false;
		if (getClass() != obj.getClass())  return false;
		Payment other = (Payment) obj;
		if (idPayment == null) {
			if (other.idPayment != null) return false;
		} else if (!idPayment.equals(other.idPayment)) return false;
		return true; }
	@Override
	public String toString() {
		return "Payment [idPayment="+idPayment+", amount="+amount+",
			paymentDate="+paymentDate+"]";}
	public Command getCommand() { return command;}
	public void setCommand(Command command) { this.command = command; }
}
```

Package Etities: Classe CreditCardPayment

```
public class CreditCardPayment extends Payment {
	private String cardNumber;private Date expirationDate;
	public CreditCardPayment() {  }
	public    CreditCardPayment(Integer    idPayment,Float    amount,Date
	paymentDate,Command command,String cardNumber,Date expirationDate) {
	 super(idPayment, amount, paymentDate, command);
	 this.cardNumber=cardNumber; this.expirationDate=expirationDate; }
	public String getCardNumber() {return cardNumber;}
	public void setCardNumber(String cardNumber){
		this.cardNumber=cardNumber;}
	public Date getExpirationDate() {return expirationDate;}
	public void setExpirationDate(Date expirationDate) {
		this.expirationDate = expirationDate;}
	@Override
	public int hashCode() {
	  final int prime = 31; int result = super.hashCode();
	  result=prime*result+((cardNumber==null)?0:cardNumber.hashCode());
	  return result;
	}
```

```
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (!super.equals(obj)) return false;
		if (getClass() != obj.getClass())  return false;
		CreditCardPayment other = (CreditCardPayment) obj;
		if (cardNumber == null) {
			if (other.cardNumber != null) return false;
		} else if (!cardNumber.equals(other.cardNumber))return false;
		return true;}
	@Override
	public String toString() {
	  return "CreditCardPayment [cardNumber="+cardNumber+", Paiement="
		 +super.toString()+", expirationDate="+expirationDate+"]"; }
}
```

Package Etities: Classe PaypalPayment

```
public class PaypalPayment extends Payment {
	private String accountNumber;
	public PaypalPayment() {}
	public PaypalPayment(Integer idPayment, Float amount, Date paymentDate,
	Command command, String accountNumber) {
		super(idPayment, amount, paymentDate, command);
		this.accountNumber = accountNumber; }
	public String getAccountNumber() {  return accountNumber;}
	public void setAccountNumber(String accountNumber) {
		this.accountNumber = accountNumber;}
	@Override
	public int hashCode() {
		final int prime = 31;   int result = super.hashCode();
		result=prime*result+((accountNumber==null)?
			0:accountNumber.hashCode());
		return result;}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)  return true;
		if (!super.equals(obj)) return false;
		if (getClass() != obj.getClass())  return false;
		PaypalPayment other = (PaypalPayment) obj;
		if (accountNumber == null) {
			if (other.accountNumber != null)   return false;
		} else if (!accountNumber.equals(other.accountNumber))
			return false;
		return true;	}
	@Override
	public String toString() {
		return "PaypalPayment [toString()="+super.toString()+",
			accountNumber="+accountNumber+"];}
}
```

Package Etities: Fichier de mapping User.hbm.xml

```
<hibernate-mapping>
<class name="entities.User" table="user" >
      <id name="idUser" column="idUser">
            <generator class="assigned"/> </id>
      <property name="login" column="login" />
      <property name="password" column="password" />
      <property name="ConnectionNumber" column="ConnectionNumber" />
      <set name="listCommandes" inverse="true">
            <key column="idUser"/> <one-to-many class="entities.Command"/>
      </set>
</class> </hibernate-mapping>
```

Package Etities: Fichier de mapping Article.hbm.xml

```
<hibernate-mapping>
<class name="entities.Article" table="article" >
      <id name="idArticle" column="idArticle">
            <generator class="assigned"/> </id>
      <property name="description" column="description" />
      <property name="brand" column="brand" />
      <property name="price" column="price" />
      <set name="listCommandeLine" inverse="true">
        <key column="idArticle"/>  <one-to-many
                  class="entities.CommandeLine"/>
      </set>
</class> </hibernate-mapping>
```

Package Etities: Fichier de mapping Commande.hbm.xml

```
<hibernate-mapping>
<class name="entities.Command" table="command" >
      <id name="idCommand" column="idCommand">
            <generator class="assigned"/> </id>
      <property name="commandDate" column="commandDate" />
      <many-to-one name="user" column="idUser"/>
      <set name="listCommandeLine" inverse="true">
            <key column="idCommand"/>
            <one-to-many class="entities.CommandeLine"/>
      </set>
</class> </hibernate-mapping>
```

Package Etities: Fichier de mapping CommandeLine.hbm.xml

```
<hibernate-mapping>
<class name="entities.CommandeLine" table="CommandeLine" >
      <id name="idCommandLine" column="idCommandLine">
            <generator class="assigned"/> </id>
      <property name="quantite" column="quantite" />
            <many-to-one name="command" column="idCommand"/>
            <many-to-one name="article" column="idArticle"/>
</class> </hibernate-mapping>
```

Package Etities: Fichier de mapping Payment.hbm.xml

```
<hibernate-mapping>
<class name="entities.Payment" table="payment"
            discriminator-value="Payment">
      <id name="idPayment" column="idPayment">
            generator class="increment"/> </id>
      <discriminator column="TYPE" type="string" />
      <property name="amount" column="amount" />
      <property name="paymentDate" column="paymentDate" />
      <many-to-one name="command" class="entities.Command" column="idCommand"
      cascade="all" unique="true" />
      <subclass name="entities.PaypalPayment"
            discriminator-value="PaypalPayment">
            <property name="accountNumber" column="accountNumber" />
      </subclass>
      <subclass name="entities.CreditCardPayment"
            discriminator-value="CreditCardPayment">
            <property name="cardNumber" column="cardNumber" />
           <property name="expirationDate" column="expirationDate" />
      </subclass>
</class> </hibernate-mapping>
```

Fichier de configuration Hibernate

```
<hibernate-configuration> <session-factory>
  <!-- Database connection settings -->
  <property name="connection.driver_class">com.mysql.jdbc.Driver
  </property>
  <property name="connection.url">jdbc:mysql://localhost:3306/dbGISPring2
  </property>
  <property name="connection.username">root</property>
  <property name="connection.password"></property>
  <!-- JDBC connection pool (use the built-in) -->
  <property name="connection.pool_size">1</property>
  <!-- SQL dialect -->
  <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
```

```
   <property name="hibernate.ejb.cfgfile">
             /org/hibernate/ejb/test/hibernate.cfg.xml</property>
   <!-- Enable Hibernate's automatic session context management -->
   <property name="current_session_context_class">thread</property>
   <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider
   </property>
   <property name="show_sql">false</property>
   <property name="hbm2ddl.auto">update </property>
   <!-- mapping resource many to many -->
   <mapping resource="entities/Article.hbm.xml"/>
   <mapping resource="entities/Command.hbm.xml"/>
   <mapping resource="entities/Payment.hbm.xml"/>
   <mapping resource="entities/User.hbm.xml"/>
   <mapping resource="entities/CommandLine.hbm.xml"/>
</session-factory> </hibernate-configuration>
```

Package Dao

```
public interface ServiceDao{
      public User  getUser(String login, String Password);}
public class ImpServiceDao implements ServiceDao{
  @Override
  public User getUser(String login, String password) {
      Session session=HibernateUtil.getSessionFactory().getCurrentSession();
      session.beginTransaction();
      Criteria criteria = session.createCriteria(User.class);
      Criterion critere = Restrictions.like("login", login);
      criteria.add(critere);
      critere = Restrictions.like("password", password);
      criteria.add(critere);
      if (criteria.list().size()==0) return (null);
      else return (User) criteria.list().get(0); }
}
Package Controller: classe ServiceController
  @Controller
  public class ServiceController {
      // injection de depandance
      @Autowired
      private ServiceMetier service;
      @RequestMapping(value="/index")
      public String accueil(Model m) {
            m.addAttribute("user",null); return "index";}
      @RequestMapping (value="/login")
      public String     login(Model m, String login, String password) {
            User user=service.getUser(login, password);
            if (user== null) return "index";m.addAttribute("user",user);
            m.addAttribute("commandsUser",user.getListCommandes());
            return "test"; }
}
```

Fichier application-servlet-config.xml

```
<beans xmlns="…">
  <context:component-scan base-package="controller"/>
  <bean class=
        "org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/vues/"/>
    <property name="suffix" value=".jsp"/>
  </bean>
</beans>
```

Fichier spring-beans.xml

```
<beans xmlns="…">
  <bean class="dao.ImpServiceDao" id="serviceDao" init-method=""> </bean>
  <bean class="metier.ImpServiceMetier" id="serviceMetier">
     <property name="dao" ref="serviceDao"></property>
  </bean>
</beans>
```

Fichier Web.xml

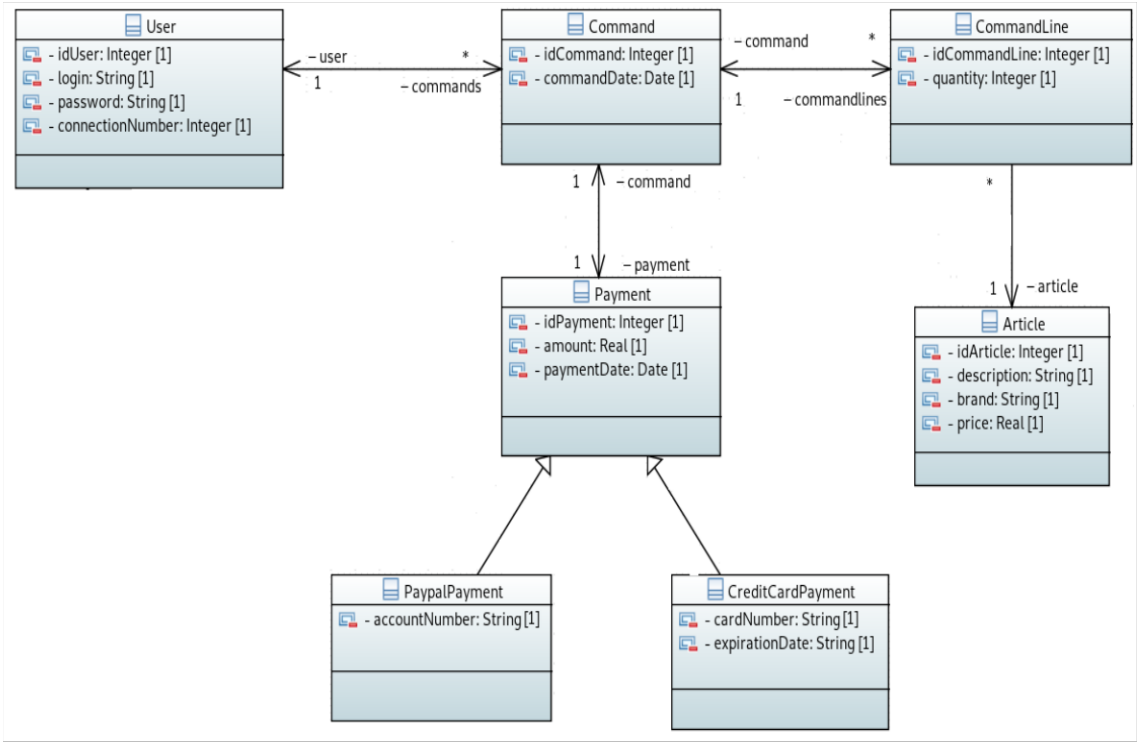
```
 <servlet>
   <servlet-name>action  </servlet-name>
   <servlet-
     class>org.springframework.web.servlet.DispatcherServlet</servlet-
     class>
   <init-param>
 <param-name>contextConfigLocation</param-name>
 <param-value>/WEB-INF/application-servlet-config.xml</param-value>
 ..</init-param>
 </servlet>
 <servlet-mapping>
   <servlet-name>action</servlet-name>
 ..<url-pattern>*.htm</url-pattern>
 </servlet-mapping>
 <context-param>
     <param-name>contextConfigLocation</param-name>
     <param-value>/WEB-INF/spring-beans.xml</param-value>
 </context-param>
 <listener>
 ..<listener-class>org.springframework.web.context.ContextLoaderListener
 ..</listener-class>
 </listener>
```

Page Login.jsp

```
<body>
<FORM action = "login.htm" method ="post">
..<TABLE >
    <TR><TD> <b>login</b>  </TD>
      <TD> <input type="text"  name="login" style="width:300px"> </TD>
    </TR>
    <TR> <TD> <b>Mot de passe</b>  </TD>
      <TD> <input type="text"  name="password"    style="width:300px"> </TD>
    </TR>
    <TR> <TD colspan =2 align=center >
       <INPUT type ="submit" name="OK" value="OK"> </TD>
    </TR>
  </TABLE>
</FORM>
<br>
</body>
```

TP N°6 : Spring Boot

Dans ce TP nous nous intéressons à un site E-Commerce permettant aux utilisateurs de passer des commandes d'articles et d'effectuer les paiements correspondants (2 paiements sont supportés : le paiement par carte et le paiement PayPal). On suppose que le projet relatif à ce site sera réalisé en se basant sur les Frameworks Spring Boot. Les informations des ventes sont enregistrées dans une base de données MySQL nommée « dbVentes ». Le diagramme de classe du système étudié est le suivant :



Les exigences techniques de l'application sont :

- Les données sont stockées dans une base de données MySQL. Un package « entities » regroupe les classes du diagramme ci-dessus
- L'application se compose de trois couches:
  - La couche Dao qui est basée sur Spring Data et JPA. Cette couche comporte principalement les interfaces JPA Repository basées sur Spring Data.
- La couche Metier (Interfaces et implémentations)

Elément de correction

Fichier « pom.xml »

```
<project …>
 …
 <dependencies>
     <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-data-jpa</artifactId>
     </dependency>
     <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-thymeleaf</artifactId>
     </dependency>
     <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-web</artifactId>
     </dependency>
     <dependency>
           <groupId>com.mysql</groupId>
           <artifactId>mysql-connector-j</artifactId>
           <scope>runtime</scope>
     </dependency>
     <dependency>
           <groupId>org.webjars</groupId>
           <artifactId>bootstrap</artifactId>
           <version>5.2.3</version>
     </dependency>
     <dependency>
           <groupId>nz.net.ultraq.thymeleaf</groupId>
           <artifactId>thymeleaf-layout-dialect</artifactId>
     </dependency>
     <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-validation</artifactId>
     </dependency>
     <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-starter-test</artifactId>
           <scope>test</scope>
     </dependency>
 </dependencies>
 …
</project>
```

Fichier application.properties

```
server.port=8081
spring.datasource.url= jdbc:mysql://localhost:3306/bdVentes
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.naming.physical-
    strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.properties.hibernate.dialect.storage_engine=innodb
spring.jpa.properties.hibernate.format_sql=true
spring.thymeleaf.cache=false
```

Package entities: Classe User

```
@Entity
@Table(name="Users")
public class User {
      @Id
      @GeneratedValue (strategy = GenerationType.IDENTITY)
      private Integer idUser;
      private String login;   private String password;
      private Integer ConnectionNumber;
      @OneToMany(mappedBy="user")
      private Set<Command> listCommandes;
      public Integer getIdUser() {  return idUser;}
      public void setIdUser(Integer idUser) {this.idUser = idUser;}
      public String getLogin() {return login;}
      public void setLogin(String login) {this.login = login;}
      public String getPassword() { return password;}
      public void setPassword(String password) {this.password = password;}
      public Integer getConnectionNumber() {return ConnectionNumber;}
      public void setConnectionNumber(Integer connectionNumber) {
                  ConnectionNumber = connectionNumber;}
      public Set<Command> getListCommandes() {  return listCommandes;}
      public void setListCommandes(Set<Command> listCommandes) {
                  this.listCommandes = listCommandes;}
      @Override
      public int hashCode() { return Objects.hash(login);}
      @Override
      public boolean equals(Object obj) {
            if (this == obj)  return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())return false;
            User other = (User) obj;
            return Objects.equals(login, other.login);}
```

```
@Override
      public String toString() {
          return "User [idUser=" + idUser + ", login=" + login + ", password="
              + password + ", ConnectionNumber="+ ConnectionNumber + "]";}
}
```

Package entities: Classe Article

```
@Entity
@Table(name="Articles")
public class Article{
      @Id
      @GeneratedValue (strategy = GenerationType.IDENTITY)
      private Integer idArticle;
      @NotEmpty
      @Size (min=5, max=50)
      private String description;
      @NotEmpty private String brand;
      @Positive private Float price=0.F;
      @OneToMany(mappedBy="article")
      private Set<CommandLine> listCommandeLine= new HashSet<CommandLine>();
      public Integer getIdArticle() {return idArticle; }
      public void setIdArticle(Integer idArticle){
            this.idArticle = idArticle;}
      public String getDescription() {return description;}
      public void setDescription(String description) {
            this.description = description;}
      public String getBrand() {return brand;}
      public void setBrand(String brand) { this.brand = brand;}
      public Float getPrice() {return price;}
      public void setPrice(Float price) {this.price = price; }
      public Set<CommandLine> getListCommandeLine() {
                  return listCommandeLine; }
      public void setListCommandeLine(Set<CommandLine> listCommandeLine) {
            this.listCommandeLine = listCommandeLine; }
      @Override
      public int hashCode() { return Objects.hash(description);    }
      @Override
      public String toString() {
        return "Article [idArticle="+idArticle+",description="+
            description+", brand=" + brand +", price="+price + "]";}
      @Override
      public boolean equals(Object obj) {
            if (this == obj)  return true;
            if (obj == null)  return false;
            if (getClass() != obj.getClass())    return false;
            Article other = (Article) obj;
            return Objects.equals(description, other.description);
      }
}
```

Package entities: Classe Commande

```
@Entity
@Table(name="Commandes")
public class Command {
	@Id
	@GeneratedValue (strategy = GenerationType.IDENTITY)
	private Integer idCommand; private Date commandDate;
	@OneToOne (mappedBy ="command" )
	private Payment payment;
	@ManyToOne
	private User user;
	@OneToMany(mappedBy="command")
	private Set<CommandLine> listCommandeLine;
	public Integer getIdCommand() {	return idCommand;  }
	public void setIdCommand(Integer idCommand) {
		this.idCommand = idCommand;}
	public Date getCommandDate() { return commandDate;}
	public void setCommandDate(Date commandDate) {
		this.commandDate = commandDate;}
	public Payment getPayment() {  return payment;    }
	public void setPayment(Payment payment) {  this.payment = payment;}
	public User getUser() {  return user;}
	public void setUser(User user) {this.user = user;}
	public Set<CommandLine> getListCommandeLine(){return listCommandeLine;}
	public void setListCommandeLine(Set<CommandLine> listCommandeLine) {
		this.listCommandeLine = listCommandeLine;}
}
```

Package entities: Classe CommandeLine

```
@Entity @Table(name="DetailsCommande")
public class CommandLine {
	@Id
	@GeneratedValue (strategy = GenerationType.IDENTITY)
	private Integer idCommandLine; private Integer quantite;
	@ManyToOne
	private Command command;
	@ManyToOne @JoinColumn(name="idArticle")
	private Article article;
	public Integer getIdCommandLine() {return idCommandLine;}
	public void setIdCommandLine(Integer idCommandLine) {
		this.idCommandLine = idCommandLine;}
	public Integer getQuantite() { return quantite;}
	public void setQuantite(Integer quantite){ this.quantite = quantite;}
	public Command getCommand() {  return command;}
	public void setCommand(Command command){this.command=command; }
	public Article getArticle() {return article;}
	public void setArticle(Article article){this.article = article;}
}
```

Package entities: Classe Payment

```
@Entity @Table(name="Payments")
@Inheritance (strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn (name="Type")
public abstract class Payment {
      @Id
      @GeneratedValue (strategy = GenerationType.IDENTITY)
      protected Integer idPayment;   protected Float amount;
      protected Date paymentDate;
      @OneToOne
      protected Command command;
      public Integer getIdPayment() {return idPayment;}
      public void setIdPayment(Integer idPayment){this.idPayment=idPayment;}
      public Float getAmount() {return amount;}
      public void setAmount(Float amount) {this.amount = amount;}
      public Date getPaymentDate() { return paymentDate;}
      public void setPaymentDate(Date paymentDate) {
                  this.paymentDate = paymentDate;}
      public Command getCommand() {  return command;}
      public void setCommand(Command command) {this.command = command;}
}
```

Package entities: Classe CreditCardPayment

```
@Entity @DiscriminatorValue("CreditCard")
public class CreditCardPayment extends Payment {
      private String cardNumber; private Date expirationDate;
      public String getCardNumber() {return cardNumber; }
      public void setCardNumber(String cardNumber) {
            this.cardNumber = cardNumber;}
      public Date getExpirationDate() {return expirationDate; }
      public void setExpirationDate(Date expirationDate) {
            this.expirationDate = expirationDate;}
}
```

Package entities: Classe PaypalPayment

```
@Entity @DiscriminatorValue("Paypal")
public class PaypalPayment extends Payment {
      private String accountNumber;
      public String getAccountNumber() {return accountNumber; }
      public void setAccountNumber(String accountNumber) {
            this.accountNumber = accountNumber;}
}
```

Package Dao: exemple de Repository « Interface ArticleRepository »

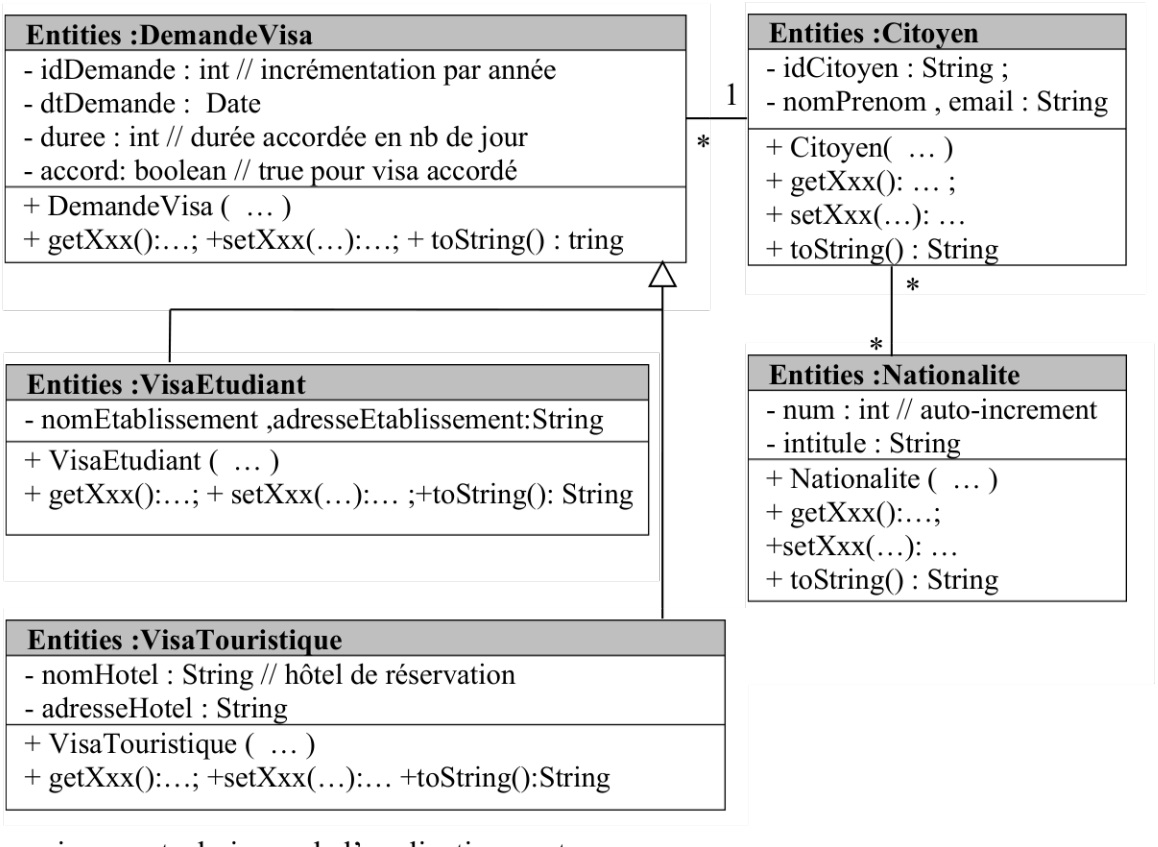
```
public interface ArticleRepository extends JpaRepository<Article, Integer>{
      public Page<Article> findByDescriptionContains(
            String key,Pageable pageable);
}
```

Package Web : Exemple de Controller « Classe ArticleController »

```
@Controller
public class ArticleController {
      @Autowired
      private ArticleRepository articleRepository;
      @RequestMapping(value="/", method = RequestMethod.GET)
      public String accueil() {return "redirect:/articles";}
      @RequestMapping(value="/articles", method = RequestMethod.GET)
      public String accueil(Model m,
            @RequestParam(name="page", defaultValue = "0") int page,
            @RequestParam(name="size", defaultValue = "8") int size,
            @RequestParam(name="key", defaultValue = "") String key) {
        Page <Article> pageArticles=articleRepository.findByDescriptionContains
        (key,PageRequest.of(page, size));
        m.addAttribute("pages", new int[pageArticles.getTotalPages()]);
        m.addAttribute("key", key); m.addAttribute("currentPage", page);
        m.addAttribute("listArticles",pageArticles.getContent());
        return "articles"; }
      @GetMapping(value="/deleteArticle")
      public String deleteArticle(Integer id,
         @RequestParam(name="page", defaultValue = "0") int page,
         @RequestParam(name="key", defaultValue = "") String key){
            articleRepository.deleteById(id);
            return "redirect:/articles?key="+key+"&page="+page;}
      @GetMapping(value="/addArticle")
      public String addArticle(Model m,Integer id, String key, int page) {
            m.addAttribute("key", key);    m.addAttribute("page", page);
            m.addAttribute("article", new Article());
            return "addArticle"; }
      @RequestMapping(value="/saveArticle", method=RequestMethod.POST)
      public String addArticle(Model m, @Valid Article article,
                BindingResult bindingResult, int page, String key) {
            if (bindingResult.hasErrors()) return "addArticle";
            articleRepository.save(article);
            return "redirect:/articles?key="+key+"&page="+page; }
      @GetMapping(value="/editArticle")
      public String editArticle(Model m,Integer id, String key, int page){
            Article article= articleRepository.findById(id).get();
            m.addAttribute("key", key);
            m.addAttribute("page", page);
            m.addAttribute("article",article);
            return "editArticle";
      }
      public ArticleRepository getArticleRepository(){
            return articleRepository;}
      public void setArticleRepository(ArticleRepository articleRepository) {
            this.articleRepository = articleRepository;}
}
```

TP N°7 : Spring Boot

On souhaite créer une application JEE qui permet de gérer les demandes de visas pour les citoyens désireux d'effectuer un séjour dans un pays étranger. On suppose que cette application sera réalisée en se basant sur les Frameworks Spring Boot. Les informations des demandes de visa sont enregistrées dans une base de données MySQL nommée « dbDemandesVisa ». Pour des considérations de simplification, on se limite au diagramme de classes ci-dessous. Ces classes sont placées dans un package « Entities »:

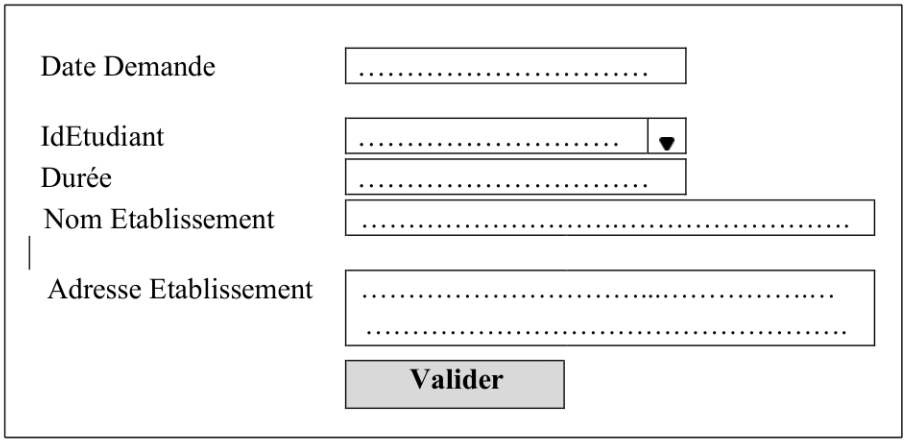


Les exigences techniques de l'application sont :

- Les données sont stockées dans une base de données MySQL
- L'application se compose de trois couches:
  - La couche Dao qui est basée sur Spring Data et JPA. Cette couche comporte principalement les interfaces JPA Repository basées sur Spring Data.
  - La couche Metier (Interfaces et implémentations)

Pour des considérations de TP, nous nous limitons à la page JSP « AddDemandeVisaEtudiant.jsp » suivante :

AddDemandeVisaEtudiant.jsp



Après la validation, la même page «AddDemandeVisaEtudiant.jsp» est réaffichée pour permettre l'ajout d'une nouvelle demande.

TP N°8 : Spring Boot

Une mutuelle désire créer une application JEE pour gérer les contrats d'assurance que les clients souscrivent. Pour offrir différents types de contrats, elle a mis au point des formules d'assurance. Chacune de ces formules est identifiée par un code et possède un libellé. Un client est identifié par un ID, il est caractérisé par le nomPrenom, l'émail et le numéro de téléphone.

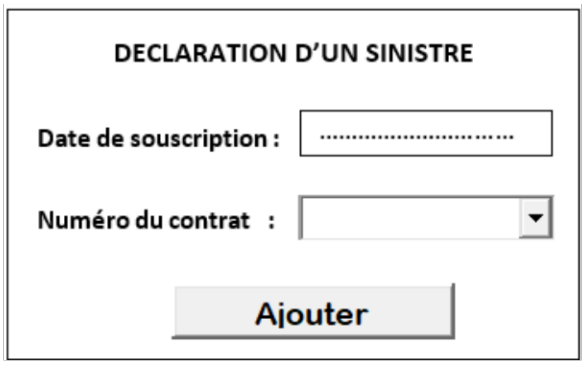
Les clients souscrivent des contrats en choisissant une formule d'assurance. Un contrat est défini par une formule. Un contrat est identifié par un numéro, il est signé à une date donnée (date de souscription) et pour une durée donnée, qui définit la date d'échéance du contrat. Mais un client peut souscrire plusieurs contrats d'assurance avec des formules différentes: voitures, logements...

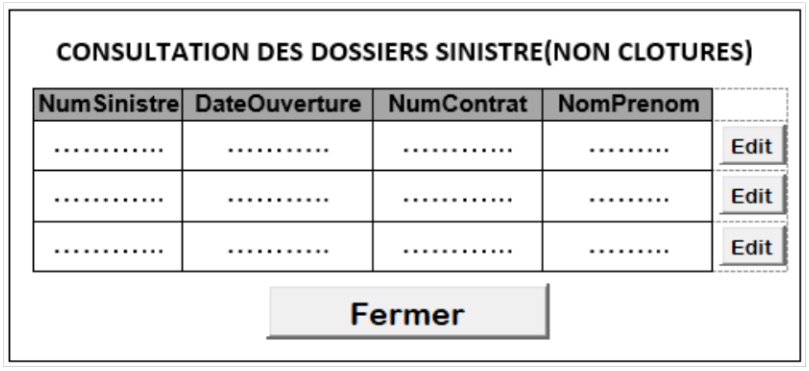
Lorsqu'un client est victime d'un sinistre (accident), il fait une déclaration auprès de sa mutuelle. Cette dernière ouvre, ainsi, un dossier de sinistre, dans le cadre du contrat correspondant. Un dossier de sinistre est identifié par un numéro unique, il contient les informations suivantes : date d'ouverture, date de clôture, et le montant des indemnités.

Les exigences techniques de l'application sont :

- Les données sont stockées dans une base de données MySQL (tous les numéros correspondant à des champs avec incrémentation automatique).
- L'application se compose de trois composants suivants:
    - La couche Dao qui est basée sur Spring Data et JPA.
    - La couche Entities (classes métiers : Formule, Contrat, Client et DossierSinistre)
    - La couche Web basée sur Spring Boot (coté serveur) en utilisant Thymeleaf comme moteur de templates.

Pour des raisons de simplification, on se limite aux opérations de déclaration et de consultation des informations des sinistres. Les pages web simplifiées correspondantes sont illustrées par la figure suivante :





Travail demandé :

1- **Package Entities**

Ecrire le code des classes métiers java (classe POJO).

2- **Package DAO**

Créer les interfaces JPA Repository basées sur Spring Data en leurs ajoutant les méthodes nécessaires pour l'ajout d'un nouveau sinistre et la consultation des sinistres non clôturés (voir la figure ci-dessus).

3- **Package Web**

Ecrire le code d'un contrôleur Spring Boot « SinistreController » qui permet de :

- Déclarer un nouveau sinistre (ajout d'un dossier de sinistre)
- Consulter les dossiers de sinistre non clôturés.