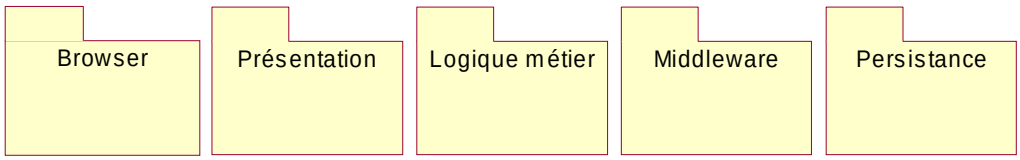
Introduction

Nous allons parler, dans ce document consacré aux architectures multi-tiers en environnement J2EE, de 3 des cinq couches les plus représentatives. Faisons tout d'abord un petit rappel sur ces cinq couches.



Browser

Cette partie est bien souvent non représentative de l'architecture mais je me permets de la située car il n'est pas exclu que celle-ci contienne une partie applicative communément appelée « Tests de premier niveau ». Les tests de premier niveau consiste principalement en la vérification du contenu des formulaires de saisie. Ils permettent de s'assurer que l'ensemble des champs obligatoires a bien été renseigné par exemple.
Cependant, et nous le verrons dans la suite, cette série de test DOIT faire partie de la couche de présentation. En effet, il n'est pas exclu que l'utilisateur final décide de désactiver les fonctionnalités JavaScript de son browser.
Une autre utilisation de cette couche est la représentation des pages dynamiques grâce, entre autre, au format DHTML.
Cette partie est généralement prise en charge par des « Web Agency » et sort donc du contexte de ce document.

Présentation

Cette couche prend en charge la logique de navigation Elle met souvent en œuvre les technologies JSP/Servlets.

Logique métier

Implémentée sous forme de JavaBean ou EJB, c'est dans cette couche que l'on retrouve l'ensemble des traitements d'une application.

Middleware

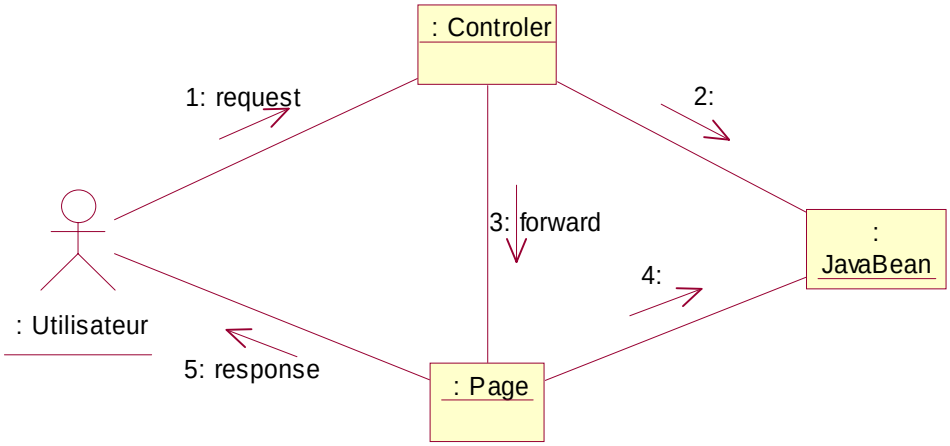
Cette partie de l'architecture couvre les connexions avec les autres systèmes ou les bases de données (Connector, JMS, JDBC).

Persistance

Partie sur laquelle nous ne nous étendrons pas non plus, la persistance. Elle se compose souvent d'une ou de plusieurs bases de données de type SGBDR, SGBDO, LDAP, …

Présentation

La couche de présentation assure la logique de navigation mais aussi la gestion des droits de l'utilisateur (authentification, droits). Elle se compose en règle générale de composants de type JSP pour la représentation graphiques des informations et de Servlets pour les contrôles.
Aucun traitement n'est à implémenter dans cette couche.
Le diagramme de collaboration ci-dessous donne un aperçu rapide de l'architecture technique a mettre en œuvre afin de respecter le Modèle Vue Contrôleur (MVC)



1. L'utilisateur émet une requête auprès du serveur en utilisant la méthode POST ou GET.
2. Le contrôleur accède aux informations demandées au travers d'un JavaBean.
3. Le contrôleur transmet le JavaBean à la page JSP chargée de la mise en forme du document.
4. La page JSP utilise les informations du JavaBean transmit.
5. La page JSP met en forme les informations et retourne la réponse à l'utilisateur.

Afin de mieux comprendre cette architecture et surtout la manière de la mettre en œuvre, nous allons réaliser un premier exemple qui a pour objectif d'afficher une page d'accueil après que l'utilisateur se soit authentifier sur le site.

Commençons tout d'abord par le JavaBean.
Ce dernier acceptera les connexions des utilisateurs enregistrés dans le fichier « users.properties » dont voici le format.

```
# users.properties
#
# login = firstName ; lastName ; userId ; password
#

alwin=Valère;VIANDIER;124;a8b22
pat=Patricia;ROGER;584;patoune
```

La classe User est définie comme suit :

```
package exo1;

import java.util.Properties;
import java.io.FileInputStream;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;

public class User {

  private String firstName;
  private String lastName;
  private int userId;

  public User() {
  }

  public String getFirstName() {
    return firstName;
  }

  public String getLastName() {
    return lastName;
```

```
  }

  public int getUserId() {
    return userId;
  }

  public void connect(String user, String password) throws Exception {
    // Contrôle des paramètres en entrée
    if( user == "" || user == null || password == "" || password == null) {
      throw new Exception("Utilisateur ou mot de passe non renseigné !");
    }
    else {
      // Lecture du fichier des utilisateurs users.properties
      Properties prop = new Properties();
      FileInputStream  in = new FileInputStream("users.properties");
      prop.load(in);

      String pUser = prop.getProperty(user);
      if( pUser == null)
        throw new Exception("Utilisateur inconnus");
      StringTokenizer iUser = new StringTokenizer(pUser,";");

      String iFirstName = iUser.nextToken();
      String iLastName = iUser.nextToken();
      String iUserId = iUser.nextToken();
      String iPassword = iUser.nextToken();
      if (iPassword.equalsIgnoreCase(password)) {
        this.firstName = iFirstName;
        this.lastName = iLastName;
        this.userId = Integer.parseInt(iUserId);
      }
      else
        throw new Exception("Mot de passe invalide");
    }
  }



  /** Classe de test */
  public static void main(String[] args) {
    User user = new User();
    try {
      user.connect("pat","patoune");
      System.out.println("Utilisateur N°" + user.getUserId()+" connecté");
      System.out.println(user.getFirstName() + " " + user.getLastName());
    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Ce JavaBean est composé des attributs « `firstName`, `lastName` et `userId` ». De plus, il dispose d'une méthode de connexion « `connect` » qui a la charge de vérifier les informations transmises et de recherche l'utilisateur.
Nous constatons que ce JavaBean est en lecture seule car il ne dispose pas d'accesseurs de type « `setXxxx` ». Ainsi, nous nous assurons que seule la méthode « `connect` » est en mesure d'initialiser les propriétés du Bean.

Nous allons utiliser une page d'index (index.jsp) qui dispose d'un formulaire d'identification. Le bouton de soumission du formulaire donnera la main au servlet contrôleur (ExoLogin) qui auras la charge d'invoquer le Bean puis de router la requête vers une page d'accueil personnalisée (accueil.jsp).

Index.jsp

```
<HTML>
<BODY>
<H1>Identification</H1>

<FORM method="post" action="/servlet/exo1.ExoLogin">
  User name : <INPUT TYPE="text" NAME="userName" value="pat"><br>
  Password : <INPUT TYPE="password" NAME="password" value="patoune"><br>
  <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
  <INPUT TYPE="RESET" VALUE="Reset">
</FORM>
</BODY>
</HTML>
```

accueil.jsp

```
<jsp:useBean id="user" class="exo1.User" scope="request"/>
<HTML>
<HEAD><TITLE>accueil</TITLE></HEAD>
<BODY>
Bonjour <%= user.getFirstName() %> <%= user.getLastName() %>
</BODY>
</HTML>
```

ExoLogin.java

```
package exo1;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import exo1.*;

public class ExoLogin extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  private User user;
  /**Initialize global variables*/
  public void init(ServletConfig config) throws ServletException {
    super.init(config);
  }
  /**Process the HTTP Get request*/
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    doPost(request,response);
  }
  /**Process the HTTP Post request*/
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // récupération des paramètres pUserName et pPassword du formulaire
    String pUserName = request.getParameter("userName");
    String pPassword = request.getParameter("password");
    System.out.println("userName:" + pUserName + "\npassword:" + pPassword
);

    // on initialise le JavaBean
    user = new User();
    try {
      // utilisation de la méthode de connexion 'connect' du JavaBean
      user.connect(pUserName,pPassword);
      // On transmet le JavaBean au context afin d'être récupéré par la
page JSP
      request.setAttribute("user",user);
      // On re-route l'appel vers la page d'accueil JSP
      gotoPage("/src/exo1/accueil.jsp",request,response);
    }
    catch (Exception e) {
      response.setContentType(CONTENT_TYPE);
```

```
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<head><title>Erreur</title></head>");
      out.println("<body>");
      out.println("<p>Erreur : " + e.getMessage() +"</p>");
      out.println("</body></html>");
    }
  }
  /**Clean up resources*/
  public void destroy() {
  }

  private void gotoPage(String adress, HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    RequestDispatcher dispatcher =
getServletContext().getRequestDispatcher(adress);
    dispatcher.forward(request,response);
  }
}
```

Les sources du projet Jbuilder4 sont à votre disposition !
Une remarque cependant sur l'utilisation de Jbuilder4 dans l'exécution de ce type d'application via l'EDI !
Il arrive parfois que la connexion se perde entre deux appel, je préconise donc de désactiver le browser intégrer à Jbuilder4 en décochant l'option project/ project properties / Run / JSP/Servlet / IDE Options / Use integrated browser et en utilisant votre browser standard avec l'adresse http://localhost :8080