Construction d'applications web

Chapitre 5

Applicatifs côté serveur en java
Architecture MVC

Architecture

- Servlet ou JSP ?
  - Deux approches pour produire du contenu
  - Servlet écrit à la main :
    - plus adapté pour l'écriture de code
  - JSP :
    - Se rapprocher d'une page HTML classique
    - Plus adapté présentation
    - => minimiser la portion de code embarqué
- Comment structurer son application ?

Exemple

- Boutique en ligne avec gestion d'un panier : site comprenant 3 pages

Boutique en ligne

Vous pouvez :

- Consulter la liste des produits et ajouter a son panier
- Voir votre facture

Page d'accueil : statique

Liste des produits disponibles

Ajout	Description	Prix
☐	Television ecran plat	1000 Euros
☐	Portable PC	1200 Euros
☐	Pizza surgelee	10 Euros
☐	Cours programmation internet	3 Euros
☐	Cafe moulu	8 Euros
☐	Smarties	5 Euros
☐	Lecteur MP3	80 Euros
☐	DVD d'un film quelconque	25 Euros
☐	Carte saison ski de fond	60 Euros
☐	Lecteur DVD	90 Euros

ajouter au panier

Retour à la page d'accueil

Consultation : générée à partir de la BD
"ajouter au panier" soumet le formulaire à une URL qui doit exécuter l'ajout
(le panier est un attribut de session)

Facture

Produit	Prix
Portable PC	1200
Cours programmation internet	3
Cafe moulu	8
Pizza surgelee	10
Cours programmation internet	3
Smarties	5
Lecteur MP3	80

Prix total : 1309 Euros

Facture : générée à partir de la BD et du panier

Architecture naïve

- un fichier (HTML, servlet ou JSP) par URL
  - il faut une URL par page + une URL pour l'ajout au panier
  - Choix logique : l'accueil (page statique) en HTML, l'URL d'ajout (pas d'affichage) est un servlet écrit à la main, les deux autres (beaucoup d'affichage) sont des JSP
- Problèmes :
  - les JSP doivent quand même contenir pas mal de code (dur à maintenir) : interrogation de la base de données, extraction des données du panier de la session...
  - Duplication : consultation.jsp et facture.jsp auront beaucoup de code en commun
  - Si on veut ajouter une fonction, p. ex. proposer une facture PDF :
    - impossible à faire en JSP, il faut définir un servlet en adaptant une grosse partie du code de facture.jsp
    - on aurait pu le prévoir et faire dès le départ un servlet pour la facture, mais alors beaucoup de HTML à écrire dans des out.println --> compliqué de modifier la présentation de la page

Défauts de l'architecture naïve

- Mélange des aspects analyse, calcul, présentation
- Complexité du code
- Difficulté de maintenance
- Peu de possibilités d'évolution (en particulier pour la présentation)

Architecture modèle-vue-contrôleur

- Architecture standard pour les applications web
  - Le *modèle* comprend la partie de l'application dite "métier" (le cœur de l'application, indépendant de l'interface web. On doit en principe pouvoir faire une autre interface à l'application, n'utilisant pas le web, sans modifier les classes du modèle).
  - Les *vues* génèrent les réponses (HTML ou autre) à renvoyer au client.
  - Le(s) *contrôleur(s)* reçoivent et traitent les requêtes HTTP. Ils exécutent les actions ou requêtes nécessaires en appelant des méthodes du modèle puis appellent une vue pour répondre au client.

Modèle + DAO

- Structure de données :
  - Concepts manipulés
  - Modèle
- Persistance des données :
  - Comment les sauvegarder et les restaurer
  - Isoler le code de sauvegarde de la structure
  - DAO (Data Access Object)

Modèle (concepts manipulés): Produit

```
public class Produit {

    protected String description = null ;
    protected int prix ;

    /** Crée un nouveau Produit */
    public Produit(String description, int prix) {
        this.description = description ;
        this.prix = prix ;
    }

    public String getDescription()
    {
        return description ;
    }

    public int getPrix()
    {
        return prix ;
    }
}
```

Cet objet est un “java-bean”

DAO : Produit

```
public class ProduitDAO {

      private DataSource ds;
      public ProduitDAO(DataSource d) {ds = d;}

      public Produit getProduit (int id) {

       // accès à la base de données et création du Produit d'identifiant id


      }

      /** Convertit une liste d'identifiants en une liste de produits */
      public List<Produit> getProduits (List<Integer> panier) {
          ...
      }
  ...

}
```

Cet objet est un “java-bean”

Modèle : Facture

```
public class Facture {

    ...

    /**
     * construit une facture à partir de la liste de produits.
     * @param produits liste des produits
     */
    public Facture(List<Produit> produits) {
        ...
    }

    public getProduit(int i) {
        ...
    }

    public getPrixTotal() {
        ...
    }


}
```

Présentation

- Classiquement : page jsp
- Peut être aussi une nouvelle servlet
  - création d'un document PDF par exemple
- Remarque :
  - Une vue est appelée par un contrôleur
  - Ne doit pas être accessible en direct via une URL

=> De préférence dans le répertoire WEB-INF (le standard des containers de servlets comme tomcat spécifie que le contenu de ce répertoire est caché)

Contrôleur

```
HttpSession session = request.getSession(true) ;
List<Integer> panier = (List<Integer>) session.getAttribute("panier") ;
if (panier == null)
    // il n'y a pas de paniers
{
    try {
        request.getRequestDispatcher("/WEB-INF/vide.html").forward(request, response);
    } catch (Exception e) {
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                "erreur : fichier vide.html introuvable") ;
        return ;
    }
} else {

    .....
}
}
```

ici le chemin n'est pas une URL mais un chemin interne au serveur

Transfère la requête et la réponse. Le navigateur n'est pas informé : il recevra le contenu de vide.html de façon transparente, l'URL de la barre d'adresse restera celle du contrôleur

Ne pas confondre avec response.sendRedirect(URL) qui indique au navigateur d'aller voir ailleurs lui-même (en faisant une nouvelle requête)

Contrôleur (suite)

```
ProduitDAO produitDAO = new ProduitDAO(datasource);
Facture facture = new Facture(produitDAO.getProduits(panier)) ;

request.setAttribute("facture", facture); // le modèle est rangé dans une zone associée à la requête

try {
    // on transfère la requête à la vue
    if (request.getParameter("type").equals("html"))
    {
        request.getRequestDispatcher("/WEB-INF/facture.jsp").forward(request, response);
    }
    else if (request.getParameter("type").equals("pdf"))
    {
        request.getRequestDispatcher("/WEB-INF/facturePDF").forward(request, response);
    }
    else
    {
        response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
                "erreur : format invalide") ;
    }
} catch (Exception e) {
    response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
            "erreur : facture.jsp introuvable") ;
    return ;
}
```

Présentation HTML en JSP simple

```
<%@page contentType="text/html"%>
<%@page import="model.Facture;"%>
<html>
   <head><title>JSP Page</title></head>
   <body>
      <% Facture facture = (Facture) request.getAttribute("facture"); %>
                .....
       <table>
       <tr><th>Produit</th><th>Prix</th></tr>
       <% for (int i=0; i<facture.getNbProduits(); i++)
       { %>

           <tr><td><%=facture.getProduit(i).getDescription()%></td>
                <td><%=facture.getProduit(i).getPrix()%></td>
           </tr>

       <%}%>
       </table>

       Prix total : <%=facture.getPrixTotal()%>
   </body>
</html>
```

Présentation HTML en JSP+EL+JSTL

```
<c:choose>
        <c:when test="${empty facture.produits}">
                    <c:import url="vide.html"/>
        </c:when>
 <c:otherwise>
        ...
       <table>
       <tr><th>Produit</th><th>Prix</th></tr>
          <c:forEach items="${facture.produits}" var="produit">
             <tr><td>${produit.description}</td>
                <td>${produit.prix}</td>
             </tr>
          </c:forEach>
        </table>

        Prix total : ${facture.prixTotal}
    </c:otherwise>
  </c:choose>
```

Présentation PDF

```
response.setContentType("application/pdf");
 OutputStream out = response.getOutputStream() ;
 Document document = new Document() ;

 HttpSession session = request.getSession(true) ;
 Facture facture = (Facture) request.getAttribute("facture") ;

 try {
    PdfWriter.getInstance(document,out) ;
    document.open() ;

       Paragraph titre = new Paragraph("Facture",...)) ;
       titre.setAlignment(Element.ALIGN_CENTER) ;
       titre.setSpacingAfter(30f) ;

       document.add(titre) ;
              .........
```