Web-MI4 et SIMO-MI7

Modèle-Vue-Contrôleur

Francis Brunet-Manquat – SIMO

Hervé Blanchon – AW

Basé sur le cours de Leila Kefi-Khelif et D. Enselme

Objectifs

- Découvrir le descripteur de déploiement (web.xml)

- Mettre en place le MVC – Modèle-Vue-Contrôleur

Descripteur de déploiement

- Le descripteur de déploiement d'une application web est un fichier nommé **web.xml** et situé dans le répertoire WEB-INF du répertoire racine de l'application web (webapp).
- Il contient les caractéristiques et paramètres de l'application. Cela inclut la **description des servlets utilisées**, ou les différents **paramètres d'initialisation**.

Description du contexte (1/2)

- Création de **paramètres de contexte** dans web.xml

```
<!-- Titre de l'application -->
  <context-param>
      <param-name>title</param-name>
      <param-value>pierre-papier-ciseaux</param-value>
  </context-param>

  <!-- URLs communes aux vues -->
  <context-param>
      <param-name>entetedepage</param-name>
      <param-value>/WEB-INF/JSP/commun/entetedepage.jsp</param-
  value>
  </context-param>
```

Description du contexte (2/2)

- **Accès** aux paramètres de contexte dans une Servlet

```
urlEntete = getInitParameter("entetedepage");
```

- **Accès** aux paramètres de contexte dans une JSP

```
<title><%= application.getInitParameter("title")%></title>
```

Description d'une servlet (1/2)

❑**Nom, class, paramètres de la Servlet** dans web.xml

```
<!-- Servlet controleur -->
<servlet>
    <servlet-name>controleur</servlet-name>
    <servlet-class>jeu.controleur.Controleur</servlet-class>
    <init-param>
        <param-name>urlJeu</param-name>
        <param-value>/WEB-INF/JSP/jeu.jsp</param-value>
    </init-param>
    <init-param>
        <param-name>urlResultat</param-name>
        <param-value>/WEB-INF/JSP/resultat.jsp</param-value>
    </init-param>
</servlet>
```

Dans la servlet (Controleur.java)

```
// Initialisation de la servlet
public void init() throws ServletException {
    urlJeu = getInitParameter("urlJeu");
    urlResultat = getInitParameter("urlResultat");
}
```

Description d'une servlet (2/2)

- **Mapping URL / Servlet** dans web.xml

  Exemple de mapping pour accéder à la servlet par l'url: http://localhost:8080/JeuMVC-1.0-SNAPSHOT/do/resultat

```
<servlet-mapping>
    <servlet-name>controleur</servlet-name>
    <url-pattern>/do/*</url-pattern>
</servlet-mapping>
```

IMPORTANT : Plus besoin de l'annotation `@WebServlet("/…")`

- **Accès** à la servlet depuis une URL dans une JSP

```
<a href="<%= application.getContextPath()%>/do/jeu">rejouer</a>
<form method="post" action="<%= application.getContextPath()%>/do/resultat">
```

Autres descriptions possibles

- Description de l'application
  `<display-name><description>`
- Fichiers index
  `<welcome-file-list><welcome-file>`
- Description des servlets
  `<servlet-class><description><load-on-startup>`
- Attributs de session
  `<session-config><session-timeout>`
- Sécurité, pages d'erreurs, références d'EJB, etc.
- Plus d'info:
  - https://docs.oracle.com/cd/E24329_01/web.1211/e21049/web_xml.htm#WBAPP502
  - http://www-igm.univ-mlv.fr/~dr/XPOSE2003/tomcat/tomcat.php?rub=16

Conclusion des premières séances

- Les pages **JSP** **doivent** **contenir le moins de java possible**

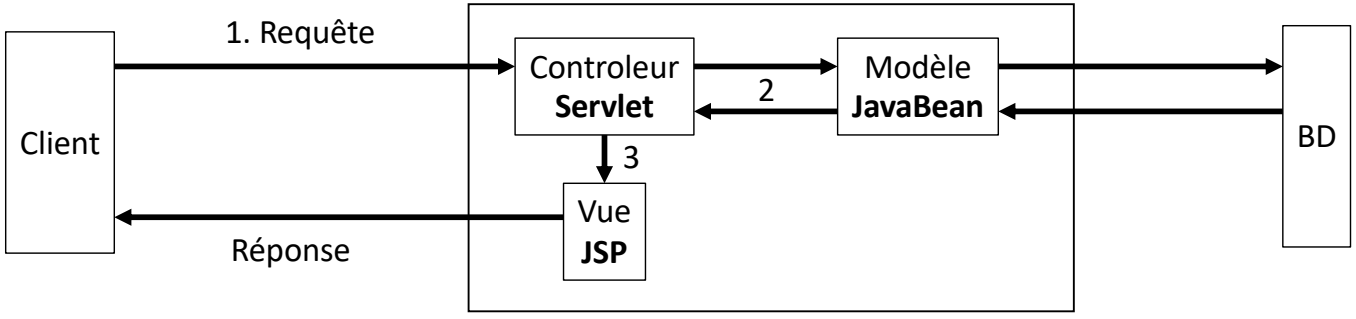
- Besoin de **structurer le code** plus finement **selon les compétences de chaque technologie** (JSP, Servlet, etc.)

⇒ Utilisation du modèle **MVC** : Modèle - Vue - Contrôleur

MVC (1/2)

- Architecture MVC (en règle générale)
  - Le **contrôle** est géré par les **Servlets**
    - Accède au modèle
    - Redirige vers la vue
  - Le **modèle** est géré par les **JavaBeans**
  - La **vue** est gérée par les **JSP**

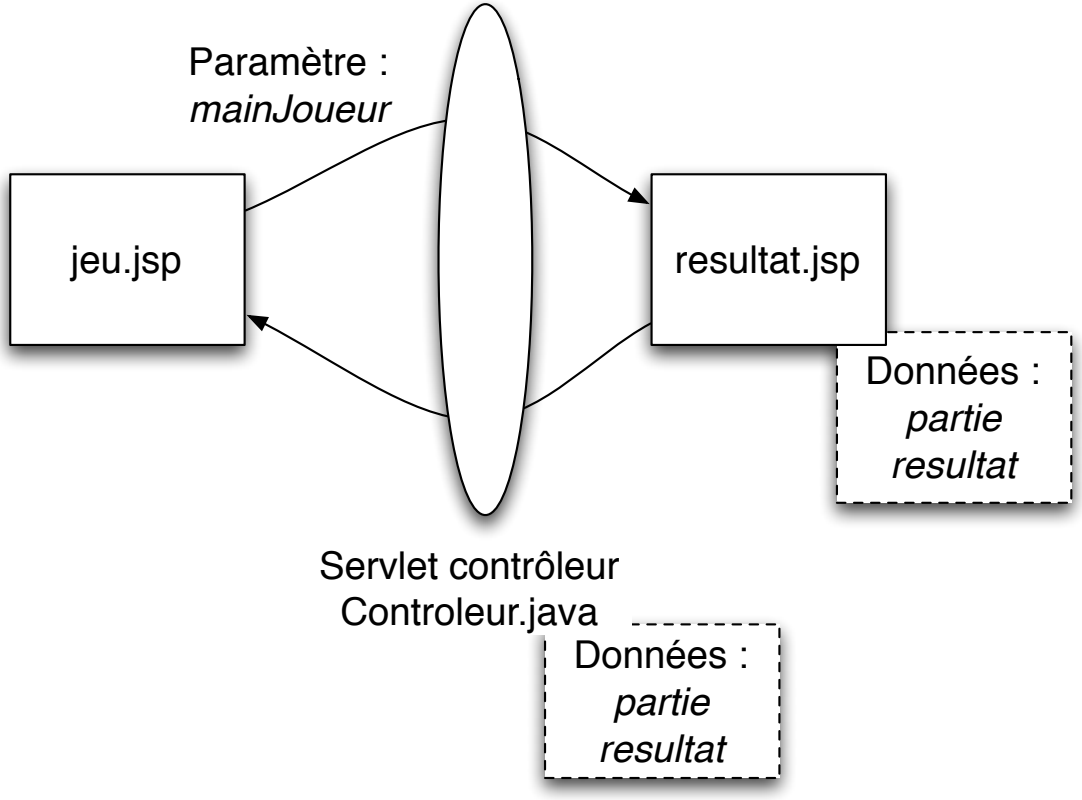
MVC (2/2)



1. Requête envoyée par le client
2. Récupération des données du modèle par le contrôleur
3. Transmission des données traitées à la vue
4. La vue est envoyée en réponse au client

→ *à voir dans le projet JeuMVC*

Jeu version MVC



Un contrôleur pour les gérer tous

```
public class Controleur extends HttpServlet {

        …

        // GET
        public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException {


            // On récupère la méthode d'envoi de la requête
            String methode = request.getMethod().toLowerCase();

            // On récupère l'action à exécuter
            String action = request.getPathInfo();
            if (action == null) {
                action = "/jeu";
            }

            // Exécution action
            if (methode.equals("get") && action.equals("/jeu")) {
                doJeu(request, response);

            } else if (methode.equals("post") && action.equals("/resultat")) {
                doResultat(request, response);

            } else {
                // Autres cas
                doJeu(request, response);
            }
        }
```

Récupération d'informations de la requête

Traitement des informations

Rappel sur la session (1/2)

- Enregistrer les données d'un utilisateur pendant un temps donné
- Une session par utilisateur (par instance de navigateur)
- Une session est commune à toutes les servlets

Rappel sur la session (2/2)

- Dans la servlet contrôlant le jeu :
  request.getSession().setAttribute(*RESULTAT_KEY*, resultat);

- Dans la JSP affichant le résultat du jeu

```
<jsp:useBean id="resultat" class="jeu.data.Resultat" scope="session"/>

…

<p>nombre de victoires :
      <jsp:getProperty name="resultat" property="nombreVictoire"/>
</p>
```

MVC : attention à la sécurité

- Le contrôleur vérifie les données en entrée
  - **ATTENTION :** toutes les requêtes doivent passer par lui !

- Les JSP sont à placer dans le dossier WEB-INF
  - Accès public aux JSP interdit
  - Seuls les servlet auront accès aux JSP

- Décrire les JSP en paramètre de la servlet contrôleur
  - A faire dans le descripteur de déploiement (web.xml)
  - Seules ces JSP seront utilisées par la servlet

Pour aller plus loin

- Les bases du développement web MVC en Java par Serge Tahé

http://tahe.developpez.com/java/baseswebmvc/

Votre projet *projet_SIL4.pdf sur Chamilo*

- Etape 3 : **mise en place du MVC** (cours 3)
  - Création d'un contrôleur pour gérer l'application, intégration des pages réalisées dans la première étape, création des pages JSP de consultation : absences, notes
  - Rappel : vous pouvez utiliser le framework CSS bootstrap pour une mise en page HTML rapide (http://getbootstrap.com).
- Etape 4 : mise en place de la persistance (cours 4)

Détails de l'étape 3

- Créer les pages suivantes :

1. Accueil contenant les url d'accès aux autres pages
2. Liste des étudiants
   - Aide : envoie d'une liste d'étudiants du contrôleur vers la vue
3. Fiche détaillée d'un étudiant (via la liste d'étudiants)
4. Tableau des absences pour tous les étudiants
   - Aide : envoie d'une HashMap du contrôleur vers la vue

- Améliorations : modifier le nombre d'absences, ajouter une moyenne, etc.

IMPORTANT : La classe GestionFactory ne doit plus être utilisée dans les JSP. Pas d'accès au modèle dans une JSP.