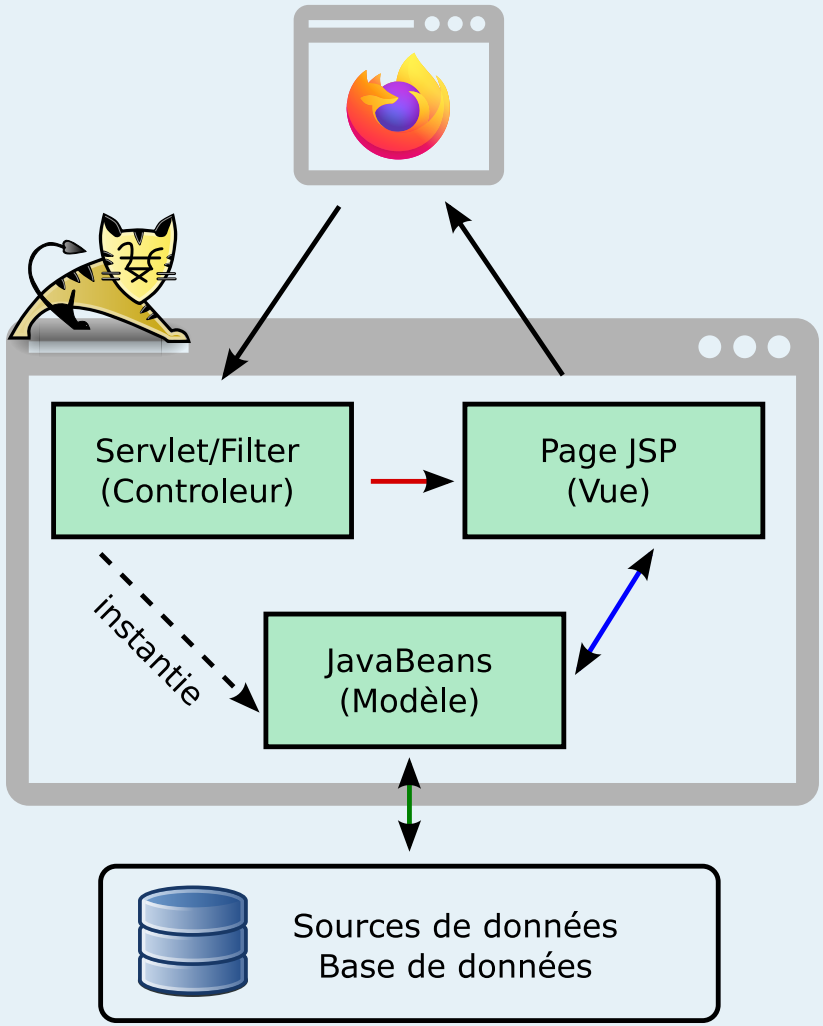
Introduction aux pages jsp

Java EE

Nicolas Delanoue

Université d'Angers - Polytech Angers

Le patron de conception Modèle-Vue-Controleur avec JEE



Avantages et inconvénients de MVC

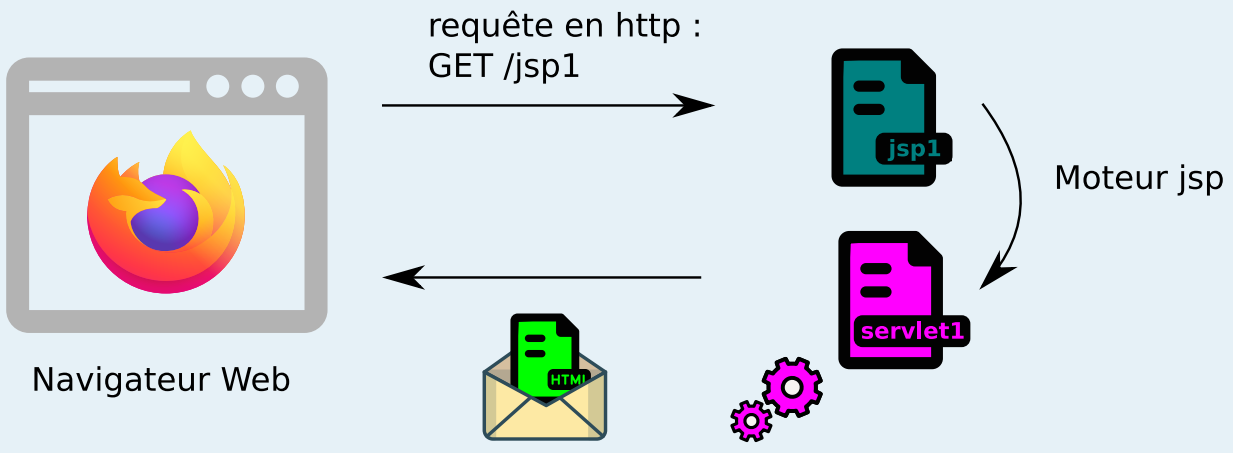
- Séparer le code chargé de créer le contenu de celui chargé de le présenter.
- Séparer les tâches de développement :
  - une équipe JSP,
  - une équipe Servlet,
  - une équipe SGBD.

Définition

Une Java Server Page (JSP) est une page html avec du *code* s'exécutant coté serveur.

- Le *code* peut être du java, ou bien une sur-couche (ex : JSTL),
- Une jsp permet de générer des pages web dynamiques,
- Une jsp est transformée par un moteur JSP en une servlet.

Jasper est un moteur de servlets



Exemple de JSP

```
<%-- Ceci est un commentaire JSP --%>
<%@ page contentType="text/html" %>
<%@ page errorPage="erreur.jsp" %>
<%@ page import="java.util.*" %>
<html>
 <head><title>Une page JSP</title></head>
 <body>
  <%! int nombreVisites = 0; %>
  <% //Il est possible d'écrire du code Java ici
    Date date = new Date();
    nombreVisites++; // portée de cette variable ?
  %>
  <h1>Exemple de page JSP</h1>
  Au moment de l'exécution de cette jsp, nous sommes le <%= date %>.
  Cette page a été affichée <%= nombreVisites %> fois !
 </body>
</html>
```

La servlet générée pour cette jps est disponible ici :
https://fr.wikipedia.org/wiki/JavaServer_Pages.

Equivalence

Tout ce qui peut être fait avec une servlet peut être fait par une jsp (et inversement).

Différences de présentation

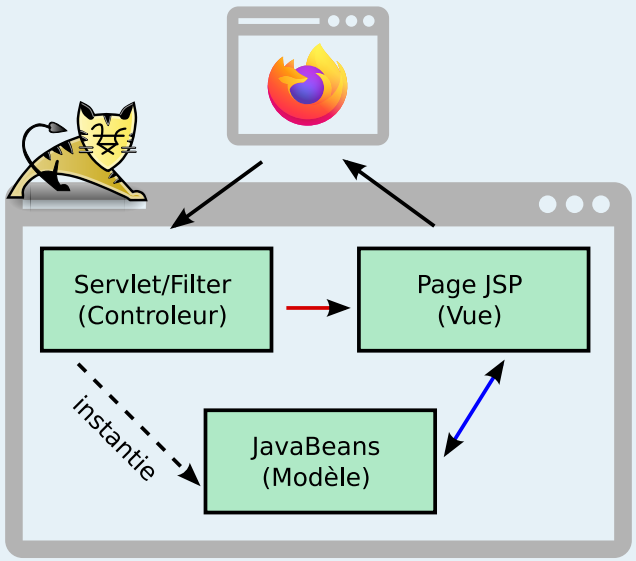
- Les servlets sont peu adaptées à la génération de contenu.
- Les pages JSP sont peu adaptées à la gestion de la logique de l'application et à l'extension de fonctions du serveur.

Request dispatcher

Le mécanisme de *Request dispatcher* permet d'automatiquement demander une autre ressource (servlet ou jsp) depuis une servlet.

Remarques

- C'est une sorte d'aiguilleur.
- C'est la flèche rouge du schéma suivant :



Exemple de redirection

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
  throws ServletException, IOException {
 // construction d'un request dispatcher sur le chemin /process,
 // qui doit exister dans la web application courante
 RequestDispatcher requestDispatcher ;
 requestDispatcher = req.getRequestDispatcher("/process") ;

 // redirection de la requête vers cette ressource
 requestDispatcher.forward(req, res);
 // la servlet actuelle perd la main pour le traitement de la requête.
}
```

Remarques

- La chaîne de caractères en entrée de `getRequestDispatcher` peut être une jsp (sur le même serveur).
- Avant le forward, on peut modifier l'objet requête avec les méthodes `setAttribute()` ou `setParameter()`.

Echanges de données entre ressources

- Avec un attribut :
  - Coté émetteur :
    `void setAttribute(String name, Object o)`
  - Coté récepteur :
    `Object getAttribute(String name)`
- Avec un paramètre :
  - Coté émetteur :
    `void setParameter(String name, String value)`
  - Coté récepteur :
    `String getParameter(String name)`
- Avec les variables de session . . .

Question

Est ce que les valeurs de ces paramètres sont visibles depuis le navigateur ?

Exemple de JSP

```
<%-- Ceci est un commentaire JSP --%>
<%@ page contentType="text/html" %>
<%@ page errorPage="erreur.jsp" %>
<%@ page import="java.util.*" %>
<html>
 <head><title>Une page JSP</title></head>
 <body>
  <%! int nombreVisites = 0; %>
  <% //Il est possible d'écrire du code Java ici
    Date date = new Date();
    nombreVisites++; // portée de cette variable ?
  %>
  <h1>Exemple de page JSP</h1>
  Au moment de l'exécution de cette jsp, nous sommes le <%= date %>.
  Cette page a été affichée <%= nombreVisites %> fois !
 </body>
</html>
```

Type d'éléments de balise JSP

- Commentaires : `<%-- Ceci est un test --%>`.
- Directives : `<%@ page import="java.util.*" %>`
  instruction pour le moteur JSP (page et include).
- Scriplets : `<% Date date = new Date(); %>`
  fragment de code Java exécuté dans la page.
- Expressions : `<%= nombreVisites %>`
  une expression est évaluée, transformée en chaîne et incluse dans la page.
- Déclarations : `<%! int nombreVisites = 0; %>`
  déclaration de variables ou de méthodes utilisables dans la page (variables de classe).

Définition

Les *directives JSP* contrôlent la manière dont le compilateur doit créer la servlet.

Synthaxe(s) générale(s)

- Avant JSP 2.0 :
  ```
  <%@ NomDirective{attribut="valeur"}%>
  ```
- Depuis JSP 2.0 :
  ```
  <jsp:directive.NomDirective attribut="valeur" />
  ```

Rôles

- définir des données relatives à la page (directive page),
- inclure une autre page JSP (directive include),
- inclure des bibliothèques de balise (directive taglib),
- ...

Exemple

Une directive page avec l'attribut import pour importer une classe à utiliser dans la page :

- `<%@ page import="java.util.Date" %>`
- `<jsp:directive.page import="java.util.Date" />`

Exemple

Une directive include avec l'attribut file pour inclure le contenu d'une autre jsp :

- `<%@ include file="maPage.jsp" %>`
- `<jsp:directive.include file="maPage.jsp" />`

Remarques :

- Avec la basile `<jsp>`, le contenu de la page contenu est généré à l'exécution . . .et non pas à la compilation (`<%@ ... >`)
- Classique pour avoir un menu commun à plusieurs pages.

Définition

Une déclaration permet de créer une variable d'instance pour notre servlet.

Synthaxe(s) générales

- Avant JSP 2.0 :

```
<%! private classe objet = valeur %>
```

- Depuis JSP 2.0 :

```
<jsp:declaration> ... </jsp:declaration>
```

Exemples

- `<%! private int VariableDInstance = 5 %>`
- `<%! private void methodeBlabla() {...} %>`

Exemple - Page jsp avec une déclaration

```
<H1>Un titre</H1>
<jsp:declaration>
 private String randomHeading() {
   return("<H2>"+ Math.random() + "</H2>"); }
</jsp:declaration>
<%= randomHeading() %>
```

Servlet générée

```
public class ______ implements HttpJspPage {
 private String randomHeading() {
  return("<H2>" + Math.random() + "</H2>");
 }
 public void _jspService(...) throws ... {
  response.setContentType("text/html");
  HttpSession session = request.getSession(true);
  JspWriter out = response.getWriter();
  out.println("<H1>Some Heading</H1>");
  out.println(randomHeading());
   ...
 }
}
```

Définition

Une *expression* est une instruction java qui est évaluée, transformée en chaîne et incluse dans la page.

Synthaxe(s) générales

- Avant JSP 2.0 : `<%= CodeGenerantString %>`
- Depuis JSP 2.0 :
  `<jsp:expression> CodeGenerantString </jsp:expression>`

Exemples

- Avant JSP 2.0 : `<%= a.ToString() %>`
- Depuis JSP 2.0 :
  `<jsp:expression> a.ToString() </jsp:expression>`

Les variables suivantes sont toujours disponibles dans une JSP :

- out : JSPWriter utilisé pour envoyer la réponse au client.
- request : HttpServletRequest,
- response : HttpServletResponse,
- session : javax.servlet.http.HttpSession,
- application : javax.servlet.ServletContext
  stocke des informations uniques pour toute la vie de l'application.
- config : javax.servlet.ServletConfig
  contient des paramètres de la servlet.
- page : la servlet elle-même.
- pageContext : javax.servlet.jsp.PageContext
  contient les données associées à la page entière.
- exception, cookie, param, ...

EL : Expression Language

- Proposée par JSTL (Java Standard Tag Library)
- Disponible depuis la version 2.4 de l'API Servlet
- Permettant d'optimiser les pages JSP (simplifier le code)
- Forme générale : `${ expression }`

Exemples

```
${ 5 + 0.2 } <!-- affiche 5.2 -->
${ "nicolas" } <!-- affiche nicolas -->
${ 2+1 < 5 ? "oui" : "non"}> <!-- affiche oui -->
```

Sans Expression Language

```
<%
String nom = request.getParameter("nom");
String prenom = request.getParameter("prenom");
out.println("Hello" + nom + " " + prenom );
%>
```

Avec Expression Language

```
Hello ${param.prenom} ${param.nom}
```

Définition

La JavaServer Pages Standard Tag Library (JSTL) ajoute une bibliothèque de balises à la spécification JSP pour

- l'exécution conditionnelle,
- les boucles,
- l'internationalisation,
- ...

Directive à ajouter au fichier jsp

```
<%@ taglib
      uri="http://java.sun.com/jsp/jstl/core"
      prefix="c" %>
```

Objectif

Faire disparaître le code java des pages jsp.

Exemple de fichier jsp avec JSTL

```
<%@ page language="java" pageEncoding="UTF-8"
         contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix ="c"%>
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type"
        content="text/html; charset=UTF-8">
    <title>Projet JEE</title>
  </head>
 <body>
  <c:out value="Hello World" />
 </body>
</html>
```

Principales balises de la librairie Core

- out : afficher un message ou le contenu d'une variable,
- set : déclarer ou modifier la valeur d'une variable,
- if, choose et when : effectuer un traitement conditionnel,
- forEach : structure de contrôle itérative.

Ces balises s'utilisent avec le préfixe c:

```
<c:out value="coucou"> <-- affiche coucou -->
<c:set var="myUser" value="nicolas" scope="request" />
<c:set var="x" value="${ 0 }" />
<c:if test="${ 1 < 2 and 2 < 3 }" > affiché </c:if>
<c:import url="header.jsp"></c:import>
...
```

4 portées pour les variables

- page : la variable est accessible seulement dans cette page,
- request : la variable est accessible seulement entre la servlet et vue appelée
- session : la variable est accessible dans toutes les pages de l'application A pour un utilisateur donné.
- application : la variable est accessible dans toutes les pages de l'application et est partagée par tous les utilisateurs.

Avantages d'une JSP

- Facile à maintenir et à coder,
- Technologie performante et scalable (comme les servlets),
- Basée sur les technologie Java donc plateforme indépendante (en principe).