UNIVERSITE DE SOUSSE
Institut Supérieur d'Informatique
et des Techniques de Communication
Hammam Sousse

جـامعة سوسة
المعهد العالي للإعلامية وتقنيات الاتصال
بحمام سوسة

Cours Développement d'applications Web et Multimédia

Chap3 : Composant Web Java EE : Les JavaServer Pages (JSP)

Filière : 3ème année Licence Multimédia

Amen Ajroud
amen.ajroud.isitcom@gmail.com

Année Universitaire : 2021-2022

Plan

- Introduction
- Pourquoi les pages JSP
- Les éléments de base
- Objets implicites
- Les directives
- Collaboration avec les JSP
- Le modèle MVC

Introduction

- Une page JSP (JavaServer Page) est une page web contenant des balises HTML.
- A l'intérieur de ces balises, on trouve des éléments spéciaux, propres à JSP (scriptlets contenant du code Java, balises personnalisés).

Pourquoi les pages JSP

- Les **Servlets** sont des composants web fiables et répondent aux besoins des programmeurs : ce sont l'endroit idéal pour écrire du code Java.
- Cependant, Une Servlet présente un inconvénient pendant qu'il génère la réponse HTML au client : tout le code HTML doit être généré par le flot d'écriture out avec la méthode `println()`.

→ Ceci parait lourd d'un point de vu écriture d'instructions.

- Les pages JSP offrent par défaut ces balises HTML représentant la réponse, il suffit d'y ajouter seulement le code Java qui **affiche le contenu dynamique** de la réponse.

Les éléments de base

- Scriptlet : sert à englober un code Java dans une page JSP. Syntaxe :
  - `<% instructions jsp %>`
- Expressions : sert a afficher dans une page la valeur d'une expression. Syntaxe :
  - `<%= expression %>`
  - Exemple : Afficher la date :

```
<%= new java.util.Date() %>
```

Objets implicites

- Certains objets implicites sont accessibles sans instanciation :
  - `request : HttpServletRequest`
  - `response : HttpServletResponse`
  - `session : HttpSession`
  - `out : PrintWriter`

Les directives JSP

- Permet de spécifier des informations utiles lors de l'exécution des pages JSP
- Ces directives sont placées entre <%@ %>
- Il existe 2 types de directives :
  - <%@ page … %>
  - <%@ include … %>
- Les directives contiennent des attributs de la forme :
  - attribut = "valeur"
- Les directives page sont placées en tête de page JSP

La directive page

- `<%@ page att1 = "val" att2 = "val" %>`
- les attributs possibles avec leurs valeurs sont :
  - `language = "java"` seule valeur permise
  - `import = "package"`

    exemple : `import="java.util.*, java.io.*"`
  - `session = "true" ou "false"` (défaut : `true`)
  - `contentType` (défaut : `"text/html"`)

La directive include

- `<%@ include file="urlRelatif" %>`
- Elle permet d'inclure un fichier (`html ou jsp`) à l'endroit où est placée la directive.
- Exemple : inclure un header et un footer

```
<html>
<head>…</head>
<body>
<%@ include file="entete.jsp" %>
<%
   instructions jsp
%>
<%@ include file="pied.jsp" %>
</body>
</html>
```

Collaboration entre JSP et Servlet

- La collaboration entre JSP a pour but :
  - partage d'information : un état ou une ressource
    - Utilisation du contexte (ex : session) pour transmettre des attributs entre pages
    - Méthodes `setAttribute(…)` et `getAttribute(…)`
  - partage du contrôle : redirection de la requête d'une servlet vers une page jsp.
    - Utilisation de l'objet `RequestDispatcher` et l'objet implicite `request`
    - Utilisation de la méthode `request.setAttribute(…)` pour transférer un objet
    - Utilisation de la méthode `forward()`

Partage de contrôle

- Un objet de type `RequestDispatcher` est obtenu en invoquant la méthode `getRequestDispatcher(String)` de la requête. Le paramètre indique la ressource de redirection.

```
[HttpServletRequest].getRequestDispatcher(url)
```

- La redirection se fait en invoquant avec l'objet `RequestDispatcher` la méthode `forward` avec la requête et la réponse:

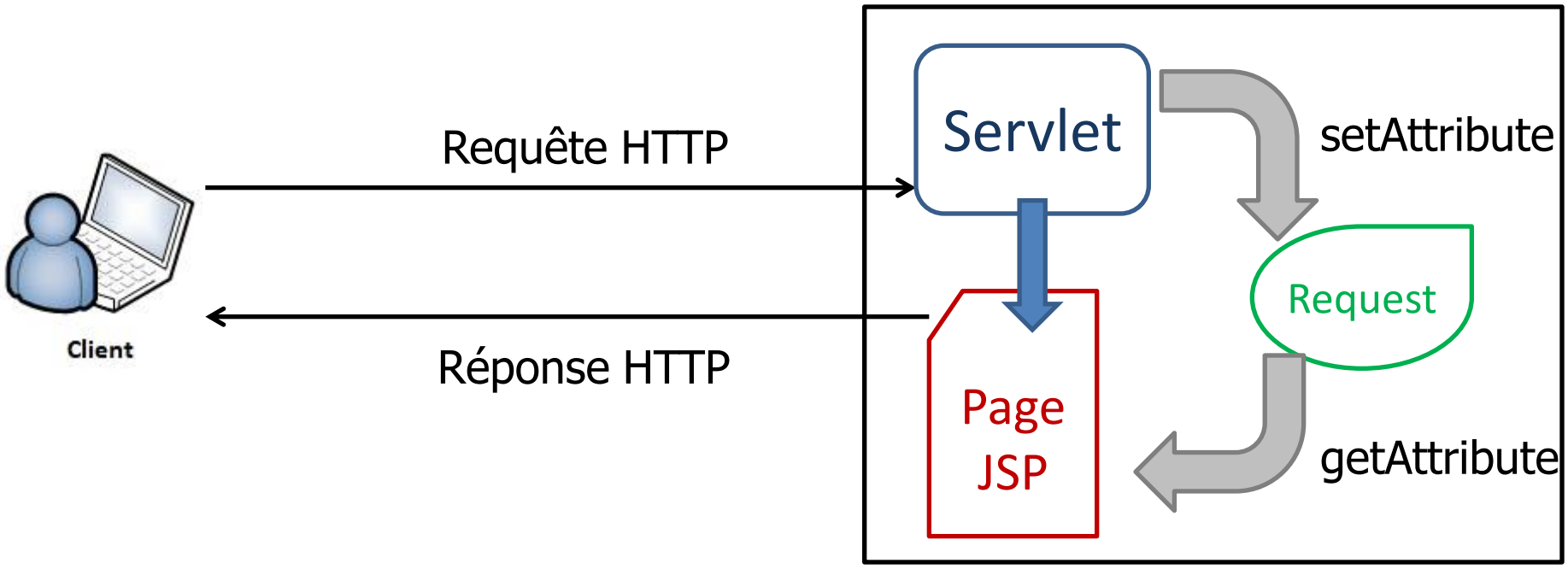
```
[RequestDispatcher].forward(request, response)
```

Partage de contrôle

- Exemple : redirection à partir d'une servlet vers une page fichier.jsp en ajoutant dans la requête un attribut.

```
RequestDispatcher dispatch =
request.getRequestDispatcher("fichier.jsp");
request.setAttribute("attribut1","bonjour");
dispatch.forward(request,response);
```

Collaboration entre Servlet & page JSP



Le pattern MVC

- Le MVC est un patron de conception qui consiste à séparer les données, les traitements et la présentation. Ainsi le code de l'application se retrouve segmentée en 3 composants essentiels :
  - le modèle
  - la vue
  - le contrôleur
- Chacun de ces 3 composants a un rôle bien défini.

Le pattern MVC

- Le **modèle** représente les données et les règles métiers.
- La **vue** correspond à l'IHM. Elle présente les données et interagit avec l'utilisateur.
- Le **contrôleur** se charge d'intercepter les requêtes de l'utilisateur, d'appeler le modèle puis de rediriger vers la vue adéquate.

Le pattern MVC

- La collaboration entre servlet, JSP et JavaBean permet de mettre en place le pattern MVC:
  - Le **contrôleur**, qui est une Servlet reçoit la requête HTTP **(1)**.
  - La Servlet instancie le **modèle** **(2)** et utilise ses méthodes pour accéder aux données de la BD **(3)**. La Servlet récupère les données du modèle **(4)** puis transfert le contrôle à la vue en incluant les données à afficher **(5)**.
  - La **vue** est une page JSP. Elle affiche les données comme réponse HTTP **(6)**.
  - Le **modèle** s'interface avec la BD à travers ses méthodes pour manipuler les données.