XML et Programmation Internet

Cours 8

kn@lri.fr

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir®

Plan

Programmation Web coté serveur

(rappel) génération de pages-web dynamiques (*i.e.* dont le contenu est calculé en fonction de la requête HTTP). Plusieurs choix de langage côté serveur.

- PHP (déploiement de site très simple, langage ~~merdique~~ particulier)
- Python, Ruby (manque de standardisation, plusieurs framework concurrents, problèmes de performances)
- ASP .NET (microsoft)
- Java/JSP (langage raisonnable, déploiement complexe)

JSP

JSP (Java Server Pages) est un *framework* permettant de créer des pages Web dynamiques en Java. Il fait partie de la suite Java EE (Entreprise Edition). Rappel :

- Java Card (Java pour cartes de crédit, très peu de choses, pas de GC)
- Java ME (Micro Edition, pour les périphériques embarqués, mobiles, etc.)
- Java SE (Standard Edition, java « normal »)
- Java EE (Entreprise Edition, SE + packages pour JSP, et autres)

Architecture

Nécessite un serveur Web particulier. Le standard est Apache Tomcat.

- **Le programmeur écrit des fichiers** `.jsp`**, contenant du HTML + du java dans des balises spéciales**
- **(Le programmeur déploie les fichiers sur le serveur Tomcat)**
- **L'utilisateur navigue vers une page** `foo.jsp`
- **Le serveur Tomcat génère** `fooServlet.class`
- **La classe est chargée dans la JVM java et (sa métode principale) est exécutée, produisant une page HTML**
- **La page HTML est envoyée au navigateur**

Exemple

```
<%@ page contentType="text/html; charset=UTF-8" %>

<!DOCTYPE html>
<html>
<head><title>test JSP</title>
      <meta charset="UTF-8"/>
</head>
<body>
Page created on
<%
       java.util.Date d = new java.util.Date();
       out.println(d.toString());
%>
</body>
</html>
```

balises spéciales JSP

JSP introduit 4 balises spéciales qui sont interprétée par le serveur Tomcat.

- **Balise de configuration :** `<%@ ... %>` **(options HTML, import de packages, ...)**
- **Balise de déclarations :** `<%! ... %>` **(déclarer des attributs et des méthodes)**
- **Balises d'instructions :** `<% ... %>` **(permet de mettre une suite d'instructions)**
- **Balises d'expressions :** `<%= ... %>` **(permet de mettre une expression dont le résultat est converti en** `String`**)**

Exemple complet

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ page import="java.util.Date" %>
<!DOCTYPE html>
<html>
<head><title>test JSP</title>
      <meta charset="UTF-8"/>
</head>
<%!
       Date maDate;

       Date maMethode() {
           return new Date();
       }
%>
 <body>
<%
   maDate = maMethode();
%>
 Page created on <%= maDate %>
</body>
</html>
```

Objets par défaut

Le code placé dans les balises spéciales a accès à certains objets automatiquement déclarés. Parmi les principaux :

- `out` **de type** `JspWrite` **(comme** `System.out` **mais écrit dans la page Web générée**
- `session` **de type** `HttpSession` **: permet de définir et récupérer des variables de sessions (i.e. Objets que l'on stocke « globalement » et que l'on peut récupérer d'une page à l'autre)**
- `request` **de type** `HttpServletRequest` **: permet de récupérer les paramètres passés dans une requête HTTP (par exemple les valeurs d'un formulaire)**
- `response` **de type** `HttpServletResponse` **: permet de spécialiser la réponse envoyée au client (en-têtes HTTP, cookies, ...)**

Classe JspWriter

Fonctionne comme `System.out` **(on peut donc appeler** `.print/.println`**) mais correspond a un endroit particulier de la page HTML**

Classe HttpSession

Propose plusieurs méthodes :

```
//Renvoie la valeur stockée sous le nom name
Object getAttribute(String name)

//Stocke l'objet value sous le nom name
void setAttribute(String name, Object value)

//Supprime l'association name value
void removeAttribute(String name)

//Définit la durée (en secondes) d'inactivité d'une session
void setMaxInactiveInterval(int interval)

//Renvoie la durée (en secondes) d'inactivité d'une session
int getMaxInactiveInterval()

//Renvoie la date (en mili-secondes depuis EPOCH) de dernière utilisation
long getLastAccessedTime()

//Détruit la session en cours
void invalidate()
```

Classe HttpServletRequest

Propose plusieurs méthodes :

```
//Récupère la valeur des cookies:
String[] getCookies()

//Récupère les paramètres passés par un formulaire :
Map<String, String[]>getParameterMap()

//Récupère un paramètre particulier
String[]getParameter(String name)
```

Classe HttpServletResponse

Propose plusieurs méthodes :

```
//Renvoie une erreur HTTP (404 par exemple)
void sendError(int code)

//Ajoute un cookie au site
void addCookie(Cookie c)

//Effectue une redirection temporaire
void sendRedirect(String url)
```

Classe Cookie

Propose plusieurs méthodes :

```
//Constructeur
Cookie(String name, String value)

//Expiration en secondes
void setMaxAge(int a)
```

Plan

Chemins par défaut

Par défaut le serveur Tomcat tourne sur le port 8080 (configurable)

- `/var/lib/tomcat/webapps/toto/` **correspond à l'URL** `http://domaine:8080/toto`
- `/var/lib/tomcat/webapps/toto/WEB-INF/` **contient des fichiers de configuration**
- `/var/lib/tomcat/webapps/toto/WEB-INF/classes/` **contient des fichiers** `.class` **auxiliaires**

(ce sera configuré légèrement différement au PUIO)

Plan

Il faut avoir le fichier `jsp-api.jar` **dans le classpath**

```
javac -cp /chemin/vers/jsp-api.jar MaClasse.java
```

On peut ensuite copier le `.class` **dans** `WEB-INF/classes`