Applets, Servlets, JSP & J2EE / EJB

IFT 6802 Commerce électronique

Par Laurent Magnin

Université de Montréal

HTML, le langage du WEB

- **Le Web se compose d'un maillage de pages, hébergées sur différents sites, reliées entre elles par des liens hypertexte.**
- **Ces pages sont décrites grâce à un langage appelé HTML**
- **Le langage HTML (*HyperText Markup Language*) tire son origine du SGML (*Standard Generalized Markup Language*) développé par Charles Golfarb et du concept de l'Hypertexte créé par Ted Nelson. C'est Tim Berners-lee, en mars 89, qui défini au CERN (Suisse) le principe de l'HTML**

HTML, fichier texte universel

- **Un document HTML n'est rien de plus qu'un fichier texte. Il peut donc être écrit et reconnu sans problème de conversion d'un environnement à un autre.**
- **Une page HTML peut également être lue et interprétée (pas forcément de façon identique) par n'importe quel navigateur sur n'importe quelle plate-forme.**

HTML, données et structure

- **Comme SGML, HTML différencie au sein d'un document, les données de la structure et du format.**
- **Il donne des règles de balisages ("marquage" de l'information avec des balises) qui décrivent une structure arborescente ou chaque noeud est identifié par une étiquette. Ces étiquettes sont reconnues et interprétées par le navigateur.**
- ***Exemple :* `<title>` "titre de la page "`</title>`**

HTML, structure générale

`<HTML>` "première ligne du document"
`<head>` "ouverture de la zone d'entête"
`<title>` "titre de la page "`</title>`
`</head>` "fermeture de la zone d'entête."
`<body>` "ouverture du corps du document"
"Mettre le texte et les images ici"
`</body>` "fin du corps du document"
`</HTML>` "fin du document HTML"

HTML, pour aller plus loin

- **Voir le source des sites Web dans les fureteurs**
- **http://web.ccr.jussieu.fr/urfist/urfistage/f.html**

Limitations de HTML

- **HTML permet uniquement de structurer du texte et des images**
  - Rien n'est prévu pour les graphiques, formules (math, chimie, etc.)
- **Les données des pages HTML sont figées**
  - Par exemple, les pages ne peuvent être paramétrables
- **Les pages HTML ne sont pas interactives**
  - Il n'est pas possible d'y entrer de l'information

Pour dépasser les limitations d'HTML

- **Générer à la demande des pages spécifiques (en fonction de la requête et de l'état du serveur - BD -)**
  - Par un langage de script : CGI, PERL, PHP, etc.
  - Par un programme : Servlet, JSP
- **Intégrer du code exécutable à l'intérieur du code HTML**
  - Javascript
  - Applets
  - Flash (pour multimédia)

Les Applets

Les Applets Java - introduction

- **Une Applet est un programme Java qui est exécuté dans un navigateur tel que Netscape ou Explorer**
- **Une Applet est intégrée dans une page au format HTML et est automatiquement téléchargée sur le poste client. Elle est ensuite exécutée par celui-ci.**

Les Applets Java - introduction

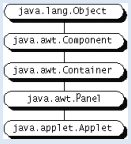
- **Le téléchargement transparent et l'exécution automatique posent des problèmes de sécurité**
- **C'est pour cela que les Applets Java sont limitées à certains domaines**
- **Typiquement, une Applet Java ne peut pas gérer de fichiers, ni ouvrir des connections réseau autres qu'avec son serveur d'origine**

Mise en œuvre

- **Tout programme Java fait appel à une classe donnée**
- **Une Applet n'échappe pas à cette règle. Si l 'on veut créer une Applet, on doit étendre la classe java.applet.Applet**
- **Cette classe contient les méthodes nécessaire à la gestion de l'Applet, et à l'interaction de celle-ci avec son environnement (navigateur)**
- **Une Applet est un objet graphique créé et contrôlé par le navigateur**

Programmation des Applets

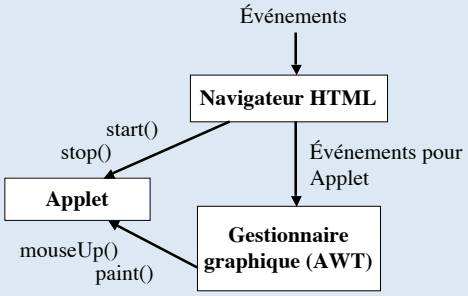
- **Basée sur la bibliothèque graphique AWT (peut également utiliser Swing, uniquement sur Java 1.2) :**



Les différentes méthodes de la classe Applet

- **Les méthodes d'interface graphique**
  - Public void paint(Graphics g) {dessiner le contenu actuel}
  - Public boolean mouseDown(Event evt, int x, int y)
  - Public boolean action(Event evt, Object what)
  - Etc.
- **Les méthodes de contrôle d'exécution**
  - Public void init() {initialisation, démarrage de threads}
  - Public void start() {démarrer l'Applet, la page Web est visitée}
  - Public void stop() {arrêter l'Applet, la page Web est quittée}
  - Public void destroy() {relâcher les ressources, libère la mémoire}

Interactions d'une Applet



Exemple d'une Applet

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {

  public void paint (Graphics g) {
   g.drawString("Hello world !", 50, 20);
  }
}
```


Intégration d'Applet au sein d'une page Web

- **Il existe des balises HTML spécifiques pour indiquer la présence d'une Applet**

```
<APPLET CODE=ClasseApplet WIDTH=largeur
HEIGHT=hauteur CODEBASE="repertoire" ALT="Ca ne
marche pas" NAME="NomApplet" ALIGN=alignement
ARCHIVE="fichier.jar">

<PARAM NAME="appletAttributei"  VALUE="valuei">

</APPLET>.
```


Les différents paramètres

- **L'argument de CODE, ClasseApplet, doit correspondre à un fichier .class qui est la classe de l'applet.**
- **WIDTH et HEIGHT définisse la largeur et la hauteur de la zone où sera affichée l'applet.**
- **CODEBASE (optionnel) permet de définir le chemin d'accès aux classes utilisées par l'applet. Par défaut le chemin d'accès est le répertoire d'où provient le fichier HTML. Le chemin spécifié par CODEBASE peut être relatif au répertoire courant du fichier HTML (qui se trouve sur le serveur), ou être une URL désignant un chemin sur un serveur différent.**

Les différents paramètres (suite)

- **ALT (optionnel) définit la chaîne de caractères à afficher quand l'applet ne peut être exécutée (si par exemple, un navigateur n'autorise pas Java, il écrira cette chaîne).**
- **NAME (optionnel) définit un nom pour l'applet (utilisé quand vous recherchez une applet par son nom).**
- **ALIGN (optionnel) permet de définir l'alignement horizontal de l'applet dans la page HTML (LEFT, RIGHT ou MIDDLE).**
- **ARCHIVE (optionnel) est un attribut apparu à partir de Java 1.1 pour définir le fichier JAR qui rassemble les classes, les images et autres fichiers qu'utilise l'applet.**

Récupération des paramètres

- **L'Applet peut lire les paramètre avec la méthode**
  - public String getParameter(String name)
  - Retourne null si le paramètre n'existe pas
- **Comme les paramètres sont des Strings, il faut les convertir selon ce qu'ils représentent (int, float, etc.)**

Exemples

```
Public void init() {
String s;

s = getParameter("niveau");
if (s != null) niveau = Integer.parseInt(s);

s = getParameter("incremental");
if (s != null) incremental = s.equals("true");

s = getParameter("angle");
if (s != null) angle = Float.valueOf(s).floatValue();
```


Exemple d'intégration d'Applet au sein d'une page Web

```
<HTML>
<HEAD>
<TITLE> Un programme simple </TITLE>
</HEAD>
<BODY>
    Voici le résultat que vous devriez voir apparaître :
<APPLET CODE="HelloWorld.class" WIDTH=150
   HEIGHT=25>
    Mettre ici le texte en cas d'absence de Java
</APPLET>
</BODY>
</HTML>
```


Références

- **Ce cours est inspiré de celui du site http://groucho.univ-lemans.fr/~jacoboni/fichiers/java/cours/index.htm**
- **De nombreux exemples sur http://www.eteks.com/coursjava/tdm.html#AppletsSite**
- **Et bien entendu, sur http://java.sun.com/applets/**

Les Servlets

Les Servlets : introduction

- **Les servlets sont au serveur ce que les applets sont au client.**
- **Les servlets sont des composants de serveur, indépendants du protocole et écrit en Java qui peuvent enrichir dynamiquement un serveur.**
- **Les servlets doivent respecter l'interface *javax.servlet.Servlet* en général en héritant des classes *javax.servlet.GenericServlet* ou *javax.servlet.http.HttpServlet*.**

Cycle de vie des servlets

- **Le cycle de vie d'une servlet est le suivant :**
  1. la méthode **init()** est appelée après le chargement (éventuellement via le réseau);
  2. une méthode **service()** est appelé à chaque requête dans une nouvelle thread.
  3. la méthode **destroy()** est appelée pour le déchargement.
- **C'est le programmeur de la servlet qui doit gérer la concurrence.**

En héritant de GenericServlet

- **Une première méthode pour écrire une servlet est :**
  - D'hériter de *javax.servlet.GenericServlet*
  - De masquer la méthode *public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, IOException* pour définir le comportement de la servlet
  - De masquer la méthode *public String getServletInfo()* pour permettre de décrire le comportement de la servlet.

Exemple

```
import java.io.*;
import javax.servlet.*;

public class HelloServlet extends GenericServlet  {
public void service(ServletRequest req, ServletResponse res)
      throws ServletException, IOException {
 PrintStream out = new PrintStream(res.getOutputStream());
 out.println("Hello World!");
 }
public String getServletInfo() {
 return "Hello World Servlet";
 }
}
```


En héritant de HttpServlet

- **Une deuxième option consiste à hériter de la classe javax.servlet.http.HttpServlet dans le cas où l'on désire écrire une servlet pour le protocole HTTP1.0.**
- **Dans ce cas il faut masquer les méthodes :**
  - *protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException*
  - *protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException*
  - À la place de la méthode *service()*.

En héritant de HttpServlet (suite)

- **Les interfaces *HttpServletRequest* et *HttpServletResponse* héritent de *ServletRequest* et de *ServletResponse*. Elle ne font qu'ajouter des méthodes.**
- **Exemple de servlet retournant toutes les informations liées à une connexion : http://www-igm.univ-mlv.fr/~roussel/RESEAUJAVA/java.servlet.html**

JSP (JavaServer Pages)

Dynamically Generated Web Content

Université de Montréal

Introductions aux JSP

- **Une page JSP est une page HTML qui contient des parties de code exécutant la logique de l'application afin de générer des pages dynamiques**
- **Ce code peut être :**
  - JavaBeans
  - Objets JDBC
  - Entreprise Java Beans (EJB)
  - Objets RMI
- **JSP est un élément clef de J2EE (Java 2 Platform, Enterprise Edition)**

Servlets & JSP

- **JSP est une extention des Servlets**
- **À la différence des Servlets, les pages JSP sont compilées à la volée, ce qui permet une bien plus grande souplesse et dynamicité**
- **Les classes Servlets contiennent du code HTML, alors que les pages JSP sont du code HTML contenant des appels Java**
- **JSP : séparation entre la partie statique et la partie dynamique, entre la présentation et la logique d'une page Web**

Composition d'une page JSP

- **Composants HTML/XML**
- **Tags JSP**
- **Morceaux de code écrit en Java ("scriptlets")**

Exécution

- **Les fichiers .jsp sont stockés sur le serveur.**
- **Ils sont désignés par une URL: http://hanoi.crim.ca/app1/hello.jsp**
- **Le chargement de l'URL provoque la compilation du fichier selon le principe suivant:**
  - **Un fichier .jsp est compilé automatiquement en servlet pour la première exécution. Le résultat (servlet source et classe) est placé dans un répertoire de travail du serveur**
  - **Après chaque modification d'un fichier *.jsp, il est automatiquement recompilé**
- **La servlet générée est ensuite exécutée**

Premier exemple

```
<HTML>
<BODY>
<H1> Hello </H1>
<ul>
<% for (int i = 0; i < 5; i++)
     out.println ("<li>" + i); %>
</ul>
</BODY>
</HTML>
```

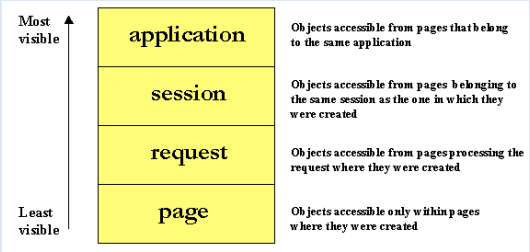
Syntaxe

- Directives: messages pour le moteur JSP
  - Page directive: souvent mis au début de chaque page
  ```
  <%@ page import="java.util.*" buffer="16k" %>
  ```
  - Include directive: inclut une autre ressource dans page
  ```
  <%@ include file="copyright.html" %>
  ```
- Déclaration: définit les variables et méthodes
  ```
  <%! int i=0 %>
  <%! public void salut() {out.print("Bonjour");} %>
  ```
- Expression: l'expression est évaluée, le résultat est converti en String et mis directement dans la page HTML générée.
  ```
  <%= totoBean.titi() %>
  ```
- Commentaire: `<%-- Voici est commentaire --%>`
- *JSP Standard Tag Library* (JSTL)

JSP Standard Tag Library (JSTL)

- **Ensemble d'actions permettant de faire des appels directs à diverses fonctionnalités**
- **Sont réparties en 4 groupes (à déclarer au début du fichier JSP) :**
  - De base < %@ taglib prefix="c" uri="http://java.sun.com/jstl/ea/core" %>
  - Traitements XML
  - I18N, gestion des localisations
  - Accès à des bases de données (SQL)
- **http://www.onjava.com/pub/a/onjava/2002/03/13/jsp.html**
- **http://www.jspin.com/home/tutorials/tags/taglibra**

Portée des objets (Tutorial de Sun)



Objets implicites

Objet	Type	Portée
application	ServletContext	application
session	HttpSession	session
request	HttpServletRequest	request
out	JSPWriter	page

Utilisation des Beans

- Déclaration de Bean:
  ```
  <jsp:useBean id= "myBean" class= "myBean"
    scope="session" />
  ```
- Modification d'un attribut:
  ```
  <jsp:setProperty name= "myBean" property="name"
    value= "dift3880" />
  ```
- Récupération d'un attribut:
  ```
  <jsp:getProperty name= "myBean" property="name" />
  ```


Référence

- **http://java.sun.com/products/jsp/**

J2EE & EJB

Java 2 Platform, Enterprise Edition &
Enterprise JavaBeans

Université de Montréal

J2SE, J2EE, J2ME ou Java Card ?

- **Java 2 Platform, Standard Edition (J2SE)**
  - Fourni l'environnement de base de Java
- **Java 2 Platform, Enterprise Edition (J2EE)**
  - Défini les standards pour le développement d'applications industrielles à base de composants
- **Java 2 Platform, Micro Edition (J2ME)**
  - Constitue un ensemble de technologies et de spécifications destinées à des applications embarquées (PDA, téléphone, imprimantes, etc.)
- **Java Card technology**
  - Destiné aux « cartes à puces » (mémoire et CPU limités)

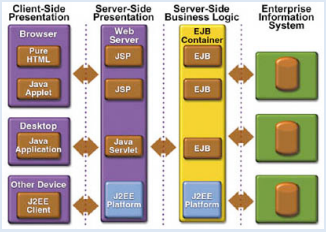
Java 2 Platform, Enterprise Edition (J2EE)

- « ***J2EE defines the standard for developing* multitier *enterprise applications. J2EE simplifies enterprise applications by basing them on* standardized, modular components, *by providing a complete* set of services *to those components, and by handling many details of application behavior automatically, without complex programming.*** »
- « ***J2EE adds [to J2SE] full support for* Enterprise JavaBeans *components,* Java Servlets *API,* JavaServer Pages *and* XML *technology. The J2EE standard includes complete* specifications and compliance tests *to ensure* portability *of applications across the wide range of existing enterprise systems capable of supporting J2EE.*** »
- **http://java.sun.com/j2ee/**

Standards vs. Implémentations

- **Standards de J2EE**
  - Collaboration entre plusieurs partenaires industriels
  - Définit des normes
  - Certifie des implémentations
- **Plates-formes compatibles**
  - *Inprise Appserver* (Borland)
  - *Weblogic* (BEA)
  - *Websphere* (IBM)
  - *JBOSS* (Open Source)
  - http://www.javaportal.co.uk/links/j2ee/page1.htm

Modèle d'applications industrielles



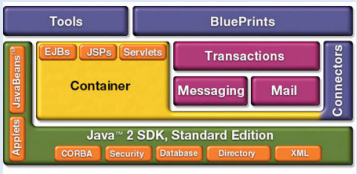
- **Modèle client / serveur, multi-tiers**

Caractéristiques des applications J2EE

- **Sont intégrés et gérés par les plates-formes :**
  - Gestion des transactions, du cycle de vie et des ressources
- **Support transparent de nombreuses technologies**
  - HTML, XML, HTTP, SSL, RMI,…
- **Logique d'affaire encapsulée dans des composants EJB**
- **Tout cela permet de se concentrer sur la logique d'affaire et les interfaces**

Services J2EE

- **Java Naming and Directory Interface API (JDNI)**
  - Standard de nommage et d'accès aux différents services et objets
- **Java Transaction API**
  - JTA permets de déclarer son propre modèle de transaction
- **Java Message Service**
  - Messages assynchrones entre composants



EJB ?

- ***« Enterprise JavaBeans (EJB) technology is the server-side component architecture for the Java 2 Platform, Enterprise Edition (J2EE) platform »***
- ***« EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology »***
- **Spécifications définies par *Java Community Process (JCP)***

EJB : *Entity Beans* ou *Session Beans* ?

- ***Entity Beans***
  - Sont permanents (e.g. sont sauvegardés par une BD)
  - Peuvent être utilisés à travers le réseau (exécution à distance)
  - Possèdent un identifiant unique (*primary key*)
- ***Session Beans***
  - Objets temporaires
  - Il est possible de les partager en utilisant leurs "*handles*"
  - Pas d'identifiants uniques

Les interfaces des EJB

- **L'interface `remote`**
  - Définit les services (logique d'affaire) fournis par l'EJB
  - Demande l'implémentation des méthodes par le programmeur
- **L'interface `home`**
  - Sert à la gestion des EJBs
  - Elle supporte la création et la découverte d'EJBs
  - Les *containers* procurent l'implémentation des méthodes

EJB : références

- **http://java.sun.com/products/ejb/index.jsp**
- **http://www.ejbtut.com/**
- **http://openejb.sourceforge.net/hello-world.html**

Référence pointant vers de nombreux cours Java :

- **http://java.developpez.com/cours/**