Serveurs d'application Web en Java
-
Servlet et JSP

LP ESSIG
2010

Plan

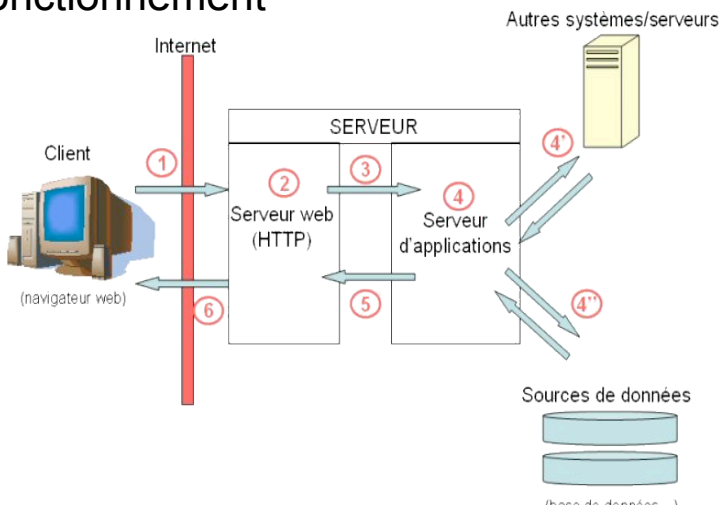
- Serveur d'application Web
  - Les containers
  - Tomcat : un exemple de container
- Les applications Web
  - Servlets
  - JSP
- Développer votre application Web

Applications sur serveur

- L'approche Java : les servlets
  - Java : langage de programmation "généraliste"
  - Permet le développement de grosses applications
  - Objectif : offrir un accès Web à ces applications
  - Alternatives :
    - CGI (pour Common Gateway Interface) - C, Perl, Python sont des langages permettant d'écrire des scripts CGI
    - PHP (pour Pretty Hypertext Processor) : libre, efficace et répandu
    - ASP (Active Server Pages) : Microsoft, plateforme .NET

Serveur d'application

- Fonctionnement



Serveur d'application

- Fonctionnement

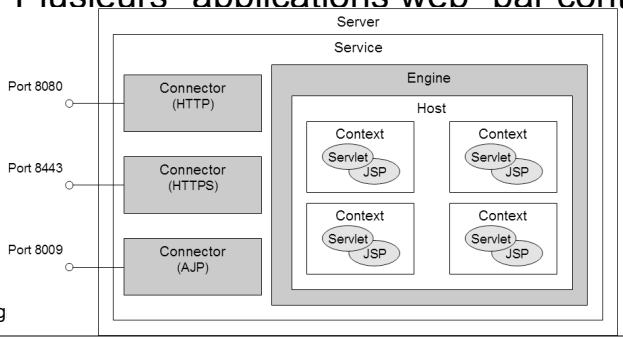
1) Réception d'une requète sur le serveur Web http://leserver.com/welcome
2) Tri des requêtes : renvoie le contenu statique (HTML/images) et délègue le traitement du contenu dynamique (3)
3) Requête pour le serveur d'application (ou *container*), transmise par le *connecteur*
4) Traitement de la requête (les objets *HttpRequest* et *HttpResponse*) sont passés en paramètres par le container à la *servlet : consultation de BD (4'') ou d'autres serveurs (4')*
5) *Renvoie de la réponse complétée (page HTML) au serveur Web*
6) *Envoie de la réponse devenue statique au client*

Encapsulation et généricité

- Il existe plusieurs *containers* d'application
- Ils respectent tous l'API des servlets
  - Donc on peut coder une servlet
  - Et la faire exécuter dans le container de son choix
    - Glassfish
    - Tomcat
    - Jboss
    - Jonas
    - Jrun
    - …

Container de Servlet

- Application Java, normalisation Sun (J2EE)
- Prise en charge du protocole Http
- Permet d'utiliser vos propres objets pour le traitement des requêtes
- Plusieurs "applications web" par container



Comparatif

Editeur	A.S.F.	redhat.	Adobe	OW2 Consortium	A.S.F.	SAP	Sun	IBM
Licence	OpenSource	OpenSource	Payant	OpenSource	OpenSource	Payant	OpenSource	Payant
Dernière version stable	6.0.26 Mars 2010	5.1.0 GA Mai 2009	4.7 Nov. 2007	5.1.0 Septembre 2009	2.2 Decembre. 2009	7.1 *NC*	v3 Decembre 2009	7.0 Sept. 2008
Certif. Java	« Impl. 5 »	5	3	5	5	5	6	5
Servlet **JSP**	2.5 2.1	2.5 2.1	2.3 1.2	2.5 2.1	2.5 2.1	2.5 2.1	3.0 2.2	2.5 2.1
EJB	*A intégrer*	3.0	2.0	2.1	3.0	3.0	3.0	3.0

http://en.wikipedia.org/wiki/Comparison_of_application_servers

Pour plus d'info

- **Comparatif** (sources supplémentaires)
  - http://www.application-servers.com/
  - http://java.sun.com/javaee/community/glassfish/
  - http://www.sap.com/platform/netweaver/index.epx
  - http://www.itrportal.com/
  - http://java.sun.com/javase/technologies/security/
  - http://en.wikipedia.org/wiki/Comparison_of_application_servers
  - http://en.wikipedia.org/wiki/Comparison_of_web_servers
  - http://java.sun.com/javaee/overview/compatibility.jsp
  - http://java.sun.com/j2ee/compatibility_1.4.html
  - http://www-01.ibm.com/software/webservers/appserv/was/
  - http://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf
  - http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf

Plan

- Serveur d'application Web
  - Les containers
  - Tomcat : un exemple
- Les applications Web
  - Servlets
  - JSP
- Développer votre application Web

Tomcat
un exemple de container

Tomcat : un des plus célèbre container de servlet

- Conçu par Apache (open-source JAKARTA)
- Reconnu implémentation de référence par Sun de la norme Servlet
  http://jarkata.apache.org/tomcat/index.html
- Plusieurs versions :

– 5.0 : fonctionne sous jdk1.4

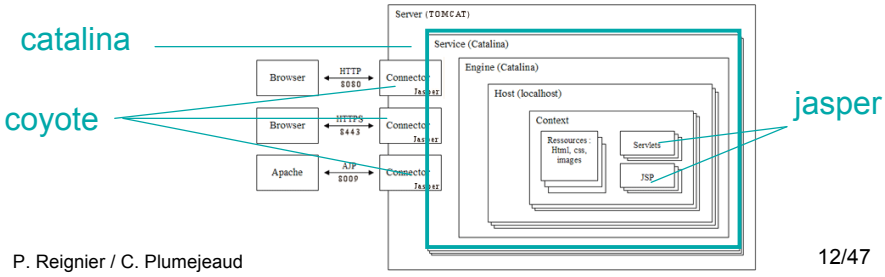
– 5.5 : fonctionne sous jdk1.5

– 6.0.26 : fonctionne sous jdk 1.6 (11 Mars 2010)

Howto : http://www.coreservlets.com/Apache-Tomcat-Tutorial/

Les composants de Tomcat

- **Catalina** est le container Servlets, et implémente les spécifications de Sun pour les Servlets et les jsp
- **Coyote** est le connecteur http: il route le trafic entrant, dirige les requètes au moteur de Tomcat, processe la requète et renvoie la réponse au client
- **Jasper** est le moteur jsp. Il parse les fichiers jsp pour les compiler en tant que Servlets (globale par Catalina).



Utiliser tomcat

- Deux aspects :
  - Où doit on mettre ses fichiers .class pour que le container les trouve : le déploiement
  - A quelle URL sera associée les classes, quels sont les paramètres de lancement, quelle sécurité . . . : le descripteur de déploiement

  http://www.coreservlets.com/Apache-Tomcat-Tutorial/

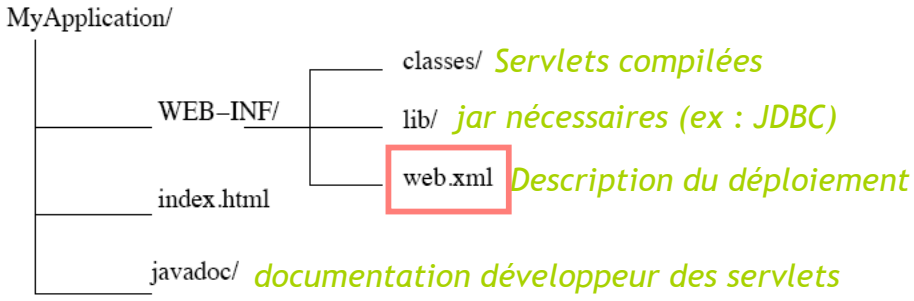
Arborescence de tomcat

- bin : lancement/arrêt du server
- common : packages java communs à l'ensemble des applications
  - classes
  - endorsed
  - lib
- conf
- server : packages java associés au serveur
  - classes
  - lib
- webapps ensemble des applications web

Arborescence de votre Webapp

- Normalisée !! (donc obligatoire…)

Sommet : fichiers HTML et JSP
- Des sous répertoires peuvent être créés pour structurer le stockage de ces fichiers.



3 solutions pour le déploiement

1. Copie de l'arborescence binaire dans le répertoire webapps
   - Redémarrer Tomcat pour prendre en compte la nouvelle application
2. Création d'une archive war déposée dans le répertoire webapps
   - Redémarrer Tomcat pour prendre en compte la nouvelle application
   - Décompression automatique du fichier war par Tomcat
3. Utilisation de l'interface *admin* de Tomcat
   - Permet d'indiquer le répertoire où se trouve l'application
   - Pas besoin de rédémarrer tomcat

La description du déploiement

- **web.xml** est le fichier pour description du déploiement de la servlet. Il permet de :
  - Mapper les URL d'accès avec des noms symboliques (nom de servlet)
  - Mapper des noms symboliques avec du code applicatif (classe de la servlet)
  - Définir des paramètres d'initialisation de l'application
  - Paramétrer les servlets
  - Définir des paramètres de durée de session, les fichiers d'accueil par défaut, etc.
- Toujours dans webappName/WEB-INF

Exemple

```
<web-app>
    <description>
      This is the webapp for the first TP on Servlet, March 2010
    </description>
    <display-name>TP1</display-name>

    <context-param>
      <param-name>webmaster</param-name>
      <param-value>christine.plumejeaud@imag.fr</param-value>
      <description>
        The EMAIL address of the administrator to whom questions
        and comments about this application should be addressed.
      </description>
    </context-param>

    <servlet>
      <servlet-name>first</servlet-name>
      <description>
         Repond à la requète
      </description>
      <servlet-class>TP1.FirstServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
      <init-param>
           <param-name>param1</param-name>
           <param-value>33</param-value></init-param>
    </servlet>

    <servlet-mapping>
      <servlet-name>first</servlet-name>
      <url-pattern>/first.form</url-pattern>
    </servlet-mapping>
</web-app>
```

String value = getServletContext().getInitParameter("webmaster");

String value = getServletConfig().getInitParameter("param1");

http://localhost:8080/TPServlet/first.form

Usage du mapping

- Associe une à n URL à nom symbolique

```
<servlet-mapping>
      <servlet-name>first</servlet-name>
      <url-pattern>/first.form</url-pattern>
 </servlet-mapping>
```

http://localhost:8080/TPServlet/first.form

```
<servlet-mapping>
      <servlet-name>first</servlet-name>
      <url-pattern>/first.form</url-pattern>
      <url-pattern>/autre.form</url-pattern>
 </servlet-mapping>
```

http://localhost:8080/TPServlet/first.form
http://localhost:8080/TPServlet/autre.form

- Page par défaut

```
<welcome-file-list>
        <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

http://localhost:1977/TPServlet/ renvoie index.html

Plan

- Serveur d'application Web
  - Les containers
  - Tomcat : un exemple
- Les applications Web
  - Servlets
  - JSP
- Développer votre application Web

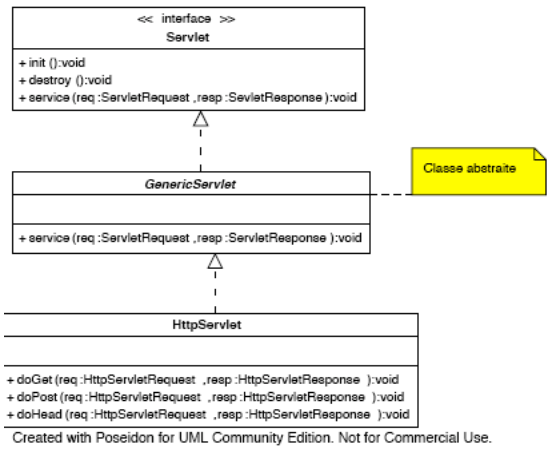
Les Servlets

- Une servlet est une application Java :
  - C'est une classe
  - Elle est compilée sous forme de bytecode
  - Elle est exécutée par une machine virtuelle Java (JVM)
- Elle possède néanmoins certaines particularités :
  - Elle réside dans un container
  - Elle est exécutée par le container quand une URL est mappée dessus
  - Elle construit alors une page HTML dynamiquement et la renvoie au container (qui transmet au server Web)
- Les servlets sont apparues avec la plate-forme Java 2 Enterprise Edition, et ne font donc pas partie de l'API Java standard. La norme actuelle de cette API est la 3.0.

http://jcp.org/aboutJava/communityprocess/final/jsr315/index.html

API des Servlet

- L'api



Cycle de vie d'une servlet

- Initialisation :
  - Chargement de la classe (au démarrage ou sur requête).
  - Instantiation d'un objet.
  - Appel de la méthode `init()` (par exemple : ouverture de lien JDBC)
- Utilisation :
  - Appel de la méthode `service()`
  - Dans le cas d'une HttpServlet : appel vers `doGet()`, `doPost()`.
- Destruction :
  - Peut-être demandée pour optimiser la mémoire
  - Appel de la méthode `destroy()`

Les paramètres de doGet(), doPost()

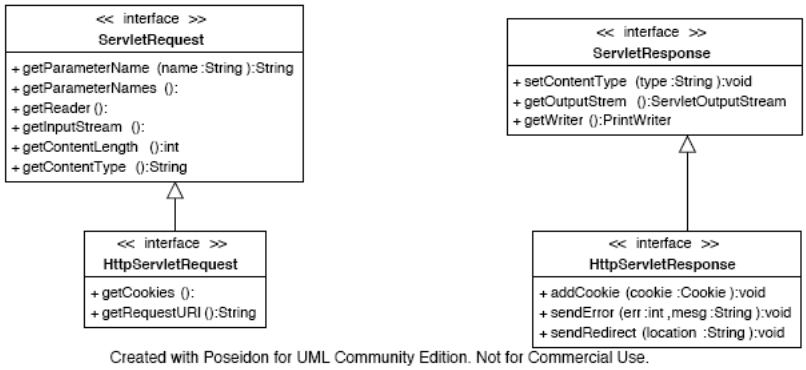
- HttpServletRequest :
  - Contexte de l'appel
  - Paramètres de formulaires
  - Cookies
  - Headers
- HttpServletResponse
  - Contrôle de la réponse
  - Type de données
  - Données
  - Cookies
  - Status

Ma première servlet

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class FirstServlet extends HttpServlet {

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

          response.setContentType("text/html");
          PrintWriter pw = response.getWriter();
          pw.print("<html>");
          pw.print("<body bgcolor=\"white\">");
          pw.print("<head>");
          pw.print("<title>Ma premiere servlet</title>");
          pw.print("</head>");
          pw.print("<body>");
          pw.print("<h1>Ca marche !</h1>");

          pw.print("<h2>context param : "+getServletContext().getInitParameter("webmaster")+"</h2
          pw.print("<h2>Servlet param : "+getServletConfig().getInitParameter("param1")+"</h2>");

          pw.print("</body>");
          pw.println("</html>");
     }
}
```

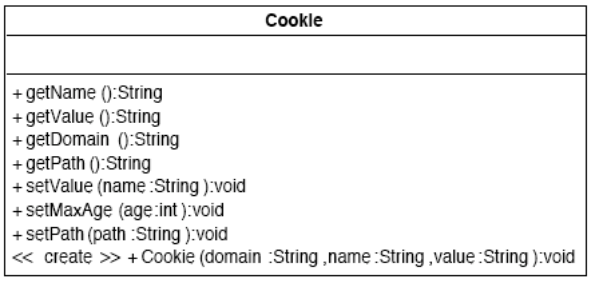

Request and Response

- L'API



Cookie

- Le nom d'un Cookie ne peut pas être changé après sa création.
- Il est conservé par le navigateur Web du client, si le client l'autorise !



Structure d'une URL

- http://serverName:serverPort/contextPath/servletPath?queryString

Exemple : http://localhost:1977/TPServlet/test

protocol	HTTP/1.1
serverName	localhost
serverPort	1977
contextPath	/TPServlet
servletPath	/test
uri	/TPServlet/test
queryString	null
Chemin sur le FS du fichier test.html	/Applications/apache-tomcat-6.0.20/webapps/TPServlet/test.html
contentType	application/x-www-form-urlencoded
pathInfo	null
pathTranslated	null

Structure d'une URL (bis)

- http://localhost:1977/TPServlet/test?classe=LPessig&cours=servlet&heure=8-10

protocol	HTTP/1.1
serverName	localhost
serverPort	1977
contextPath	/TPServlet
servletPath	/test
uri	/TPServlet/test
queryString	classe=LPessig&cours=servlet&heure=8-10
Chemin sur le FS du fichier test.html	/Applications/apache-tomcat-6.0.20/webapps/TPServlet/test.html
contentType	null
pathInfo	null
pathTranslated	null

- Paramètres de la requête : Lpessig, cours et heure

Exemple de récupération des paramètres

```
protected void doGet(HttpServletRequest req, HttpServletResponse res)
  throws java.io.IOException, ServletException {

  res.setContentType("text/html");
  PrintWriter pw = reponse.getWriter();
  Enumeration e = req.getParameterNames();

  pw.setContentType("text/html") ;
  ...
  while (e.hasMoreElements())
     { // récupération des paramètres
          String attribute = (String)e.nextElement();
          pw.print("Attribut :   " + attribute) ;
          pw.print("Valeur :   " + req.getParameter(attribute)) ;
     }
  ...
}
```

Exemple de récupération des cookies

```
cookie = req.getCookies() ;
if (cookie != null)
    {
        for (int i=0; i<cookie.length; i++)
            {
                pw.print("<TR><TD>\n") ;
                pw.print(cookie[i].getName()) ;
                pw.print("</TD><TD>\n") ;
                pw.print(cookie[i].getValue()+"</TD>\n") ;
                pw.print("</TR>\n") ;
            }
    }
```


Exécution d'une servlet

- Serveur web multi-threadé
  - Un thread par requête
- Attention à la gestion des ressources partagées
- Lien Instance, Thread. Par défaut :
  - Une seule instance
  - Les attributs de la classe deviennent des ressources partagées

Exemple avec factorielle

- nombre comme ressource partagée

```
public class Factorielle extends HttpServlet {

protected int nombre ;

protected int factorielle(int x) {
     int res=1 ;
     try {
          this.nombre = x ;
          res = 1 ;
          for (int i =1; i<=nombre; i++) {
          Thread.sleep(1000) ;
          res*= i ;
     }
     } catch (InterruptedException e) {
          e.printStackTrace() ;
     }
     return res ;
}
```


Exemple avec factorielle

- L'affichage du résultat

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException {

     response.setContentType("text/html");
     PrintWriter pw = response.getWriter();
     pw.print("<html>");
     pw.print("<body bgcolor=¥"white¥">");
     pw.print("<head>");
     pw.print("<title>Factorielle</title>");
     pw.print("</head>");
     pw.print("<body>");
     Enumeration e = request.getParameterNames();
     while (e.hasMoreElements()){
          String param = e.nextElement().toString();
          pw.println("<h2>"+param+" vaut "+request.getParameter(param)+"</h2>");
     }
     String paramValue = request.getParameter("V1");
     int x = new Integer(paramValue).intValue();
     int factX = this.factorielle(x);
     pw.println("<h2>factorielle de "+paramValue+" vaut "+factX+"</h2>");
     pw.print("</body>");
     pw.println("</html>");
}
```

Solutions au partage de ressources

- Sections critiques :

Utilisation du mot clé synchronized

- Modèle d'exécution : 1 instance par thread

Implémenter l'interface SingleThreadModel

Le suivi du client : Session

- Permet de simuler une connexion continue entre un navigateur et le serveur.
- Sauvegarde d'objets (associés à un client) :
  - entre différentes pages
  - entre des appels successifs à une même page
- Identifiant unique transmis sous forme de cookie
- Utilisation de l'interface `javax.servlet.http.HttpSession`

Usage de Session

- Session : associé à HttpServletRequest
- Accès à la session :

```
HttpSession session = request.getSession(true)
```

  - Récupération de la session associée au navigateur
  - Création de la session si elle n'existe pas
- Sauvegarde d'objets : Association (clé, instance)
  - `Obj o ;`
  - `session.setAttribute("cle", o);`
- Extraction d'objets :
  - `Obj o ;`
  - `o = (Obj) session.getAttribute("cle") ;`
- Fin de session :
  - `session.invalidate() ;`

Exemple : un compteur

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException {

     HttpSession session = request.getSession(true) ;
     Compteur cpt = (Compteur) session.getAttribute("compteur") ;
     if (cpt == null) {
          cpt = new Compteur() ;
          session.setAttribute("compteur", cpt) ;
     }

     response.setContentType("text/html");
     PrintWriter pw = response.getWriter();
     pw.print("<html>");
     pw.print("<body bgcolor=¥"white¥">");
     pw.print("<head>");
     pw.print("<title>Le compteur</title>");
     pw.print("</head>");
     pw.print("<body>");
     pw.print("<h1>Nombre de visites : "+cpt.getCompteur()+"</h1>");
     pw.print("</body>");
     pw.println("</html>");

     cpt.setCompteur(cpt.getCompteur()+1);
     int max = new
     Integer(getServletConfig().getInitParameter("maxvisites")).intValue();
     if (cpt.getCompteur()==max){
          session.invalidate() ;
     }
}
```

Au sujet des beans

- La classe Compteur est un bean (stocké dans la session)

```
public class Compteur {
    private int compteur;

    public Compteur(){
        compteur = 0;
    }

    public int getCompteur() {
        return compteur;
    }
    public void setCompteur(int compteur) {
        this.compteur = compteur;
    }
}
```

attribut

Accesseur (getter)

Modifieur (setter)

- Les beans sont des *containers* pour les données : ils ne doivent pas faire de traitement applicatif.

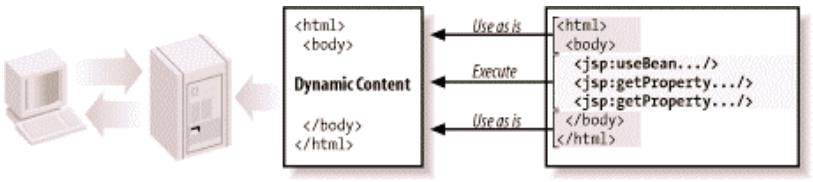
Java Server Pages (introduction)

- Les JSP sont des pages HTML comportant du code Java et des directives spécifiques
  - Avantage : plus besoin de coder le HTML avec out.println(« <html>…<table>…);
  - La page HTML contient directement les instructions pour une mise à jour dynamique
- qui sont compilées par le container : une fois compilée, c'est en fait le code d'une servlet (de même nom que la page JSP) qui s'exécute.

JSP

- Passage du HTML statique au HTML dynamique

```
<html>
<head>
<title>Greetings !</title>
</head>
<body>
<% for (int i=0;i<5;i++) { %>
      <h1>Hello World !</h1>
<% } %>
</body>
</html>
```



Exemple

- http://localhost:1977/TPServlet/FirstJSP.jsp

JSP

```
<%@page contentType="text/html"%>
<%@page import="java.util.Date"%>
<html>
    <head><title>Commentaires dans une page JSP</title></head>
    <%-- page pour montrer l'utilisation des commentaires --%>
    <!-- cette page a été générée le <%= new Date()%> -->
    <body>
    <H1>BONJOUR MONDE CRUEL</H1>
    <H1>Il est <%= new Date() %></H1>
    </body>
</html>
```

HTML

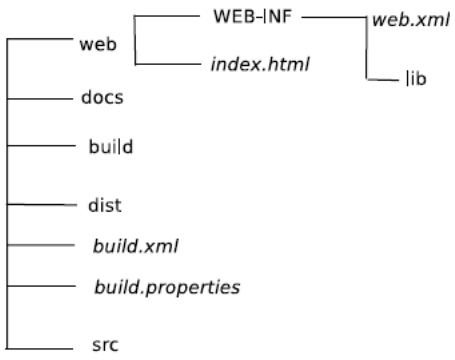
```
<html>
   <head><title>Commentaires dans une page JSP</title></head>

   <!-- cette page a été générée le Mon Mar 15 15:47:13 CET 2010 -->
   <body>
   <H1>BONJOUR MONDE CRUEL</H1>
   <H1>Il est Mon Mar 15 15:47:13 CET 2010</H1>
   </body>

</html>
```

Plan

- Serveur d'application Web
  - Les containers
  - Tomcat : un exemple
- Les applications Web
  - Servlets
  - JSP
- Développer votre application Web

Organiser les sources

- Arborescence



Cycle de développement

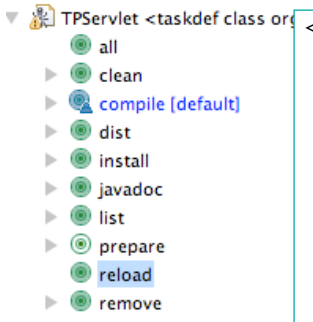
1. Ecriture du code source
2. Compilation vers le répertoire build
3. Déploiement sous tomcat (soit par recopie vers webapps, soit en utilisant directement le répertoire build) (cf slide 16)
4. Test
5. Retour si nécessaire au point 1
6. Une fois terminé, création de l'archive war sous dist.

Ant

- Objectif : automatiser les tâches.
- build.xml : liste de tâches (*target*) associées à un projet.
- Sélectionnées parmi des tâches prédéfinies :
  - Compilation, copie de fichiers, SVN, ftp, . . .
- Gestion des dépendances entre tâches.
- Ne fait que ce qui est nécessaire.
- Possibilité d'ajouter de nouveaux types de tâches
- Multi-plateforme (car basé sur Java)

Exemple avec ant

- Appel de la tâche : ant <nom de la target>
- Vue sous eclipse




```
<project default="compile" basedir=".">
      <property name="bin.dir" location="../classes"/>

      <target name="init">
            <mkdir dir="${bin.dir}"/>
      </target>
      <target name="compile"  depends="init ">
            <javac destdir="${bin.dir}"  srcdir="."/>
      </target>
      ...
      <target name="clean">
            <delete>
            <fileset dir=".">
                  <include name="*.class"/>
            </fileset>
            </delete>
      </target>
</project>
```