CSI 3540

Structures, techniques et normes du Web

Programmation côté serveur : Servlets

Objectif:

- Introduction à la programmation côté serveur
- Introduction aux Servlets

Lectures:

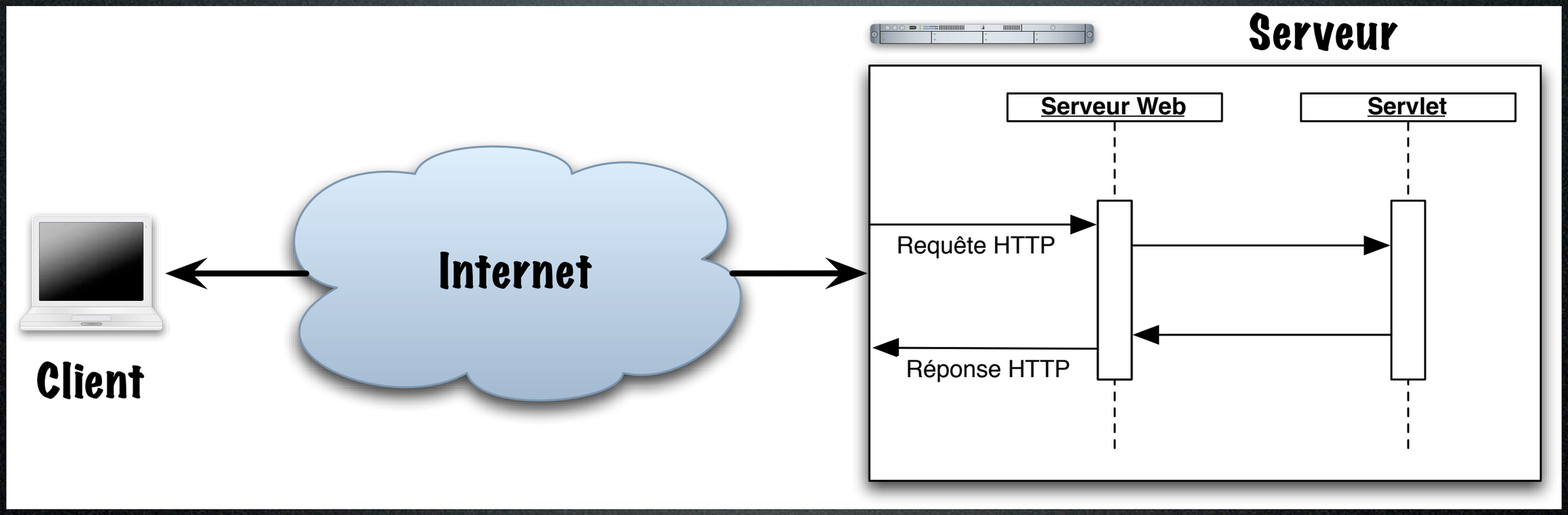
- Web Technologies (2007) § 6 Pages 307-322

Plan

1. Survol des Servlet
2. Redéfinir doGet() et doPost()
3. Développement d'une application Web

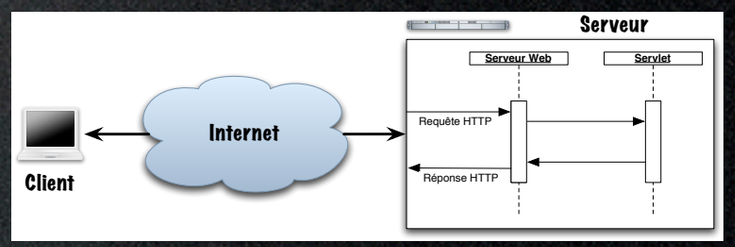
1. Ant (asant)
2. War
3. GlassFish

Servlet


Client
Internet
Serveur
Serveur Web
Servlet
Requête HTTP
Réponse HTTP

Séquence 1/4

- Le serveur Web reçoit une requête d'un agent utilisateur
  - Le **décodage de l'URI** : exemple, le chemin débute par **/servlet**



Configuration

- httpd.conf

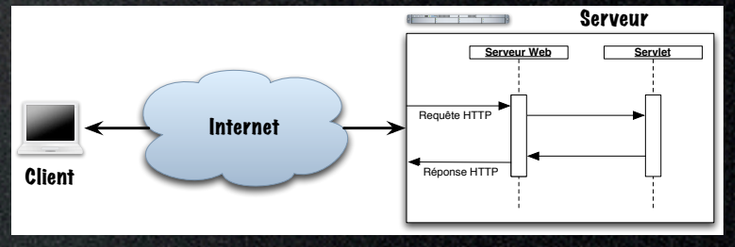
  LoadModule jk2_module libexec/mod_jk2.so

- workers2.properities

  [channel.socket:localhost:8009]
  port=8009
  host=127.0.0.1
  [ajp13:localhost:8009]channel=channel.socket:localhost:8009
  [uri:/servlet/*]
  worker=ajp13:localhost:8009

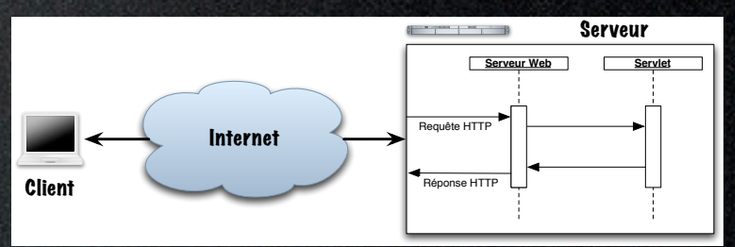
Séquence 2/4

- Le serveur Web fait un appel de méthode du Servlet
  - Passe en paramètre deux objets modélisant la requête et la réponse : **HttpServletRequest** et **HttpServletResponse**
- Ces objets sont définis par **Java Servlet API**



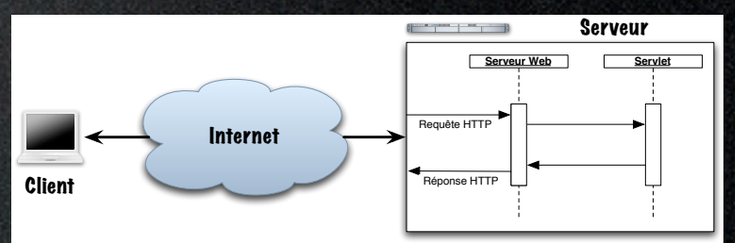
Séquence 3/4

- Le Servlet exécute un traitement
  - Typiquement, le Servlet **construira une page HTML qu'il sauvegardera dans l'objet HttpServletResponse**
- Le Servlet peut aussi ajouter à l'objet réponse des informations qu'il souhaite transmettre au client (via le serveur Web)



Séquence 4/4

- Le serveur Web génère **un message HTTP réponse**, à partir des informations sauvegardées dans l'objet **HttpServletResponse**, qu'il envoie à l'agent utilisateur



Quelle heure est-il ?

- **Problème**
  - Concevoir une application (**Servlet**) qui affiche la **date** et l'**heure** à chaque appel
  - Les pages seront certainement générées **dynamiquement**

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class GetTime extends HttpServlet {

    public void doGet( HttpServletRequest requete, HttpServletResponse reponse )
        throws ServletException, IOException {

        reponse.setContentType( "text/html; charset=\"UTF-8\"" );
        PrintWriter doc = reponse.getWriter();

        doc.println( "<!DOCTYPE html" );
          // ...
        doc.println( "</html>" );

        doc.close();
    }
}
```

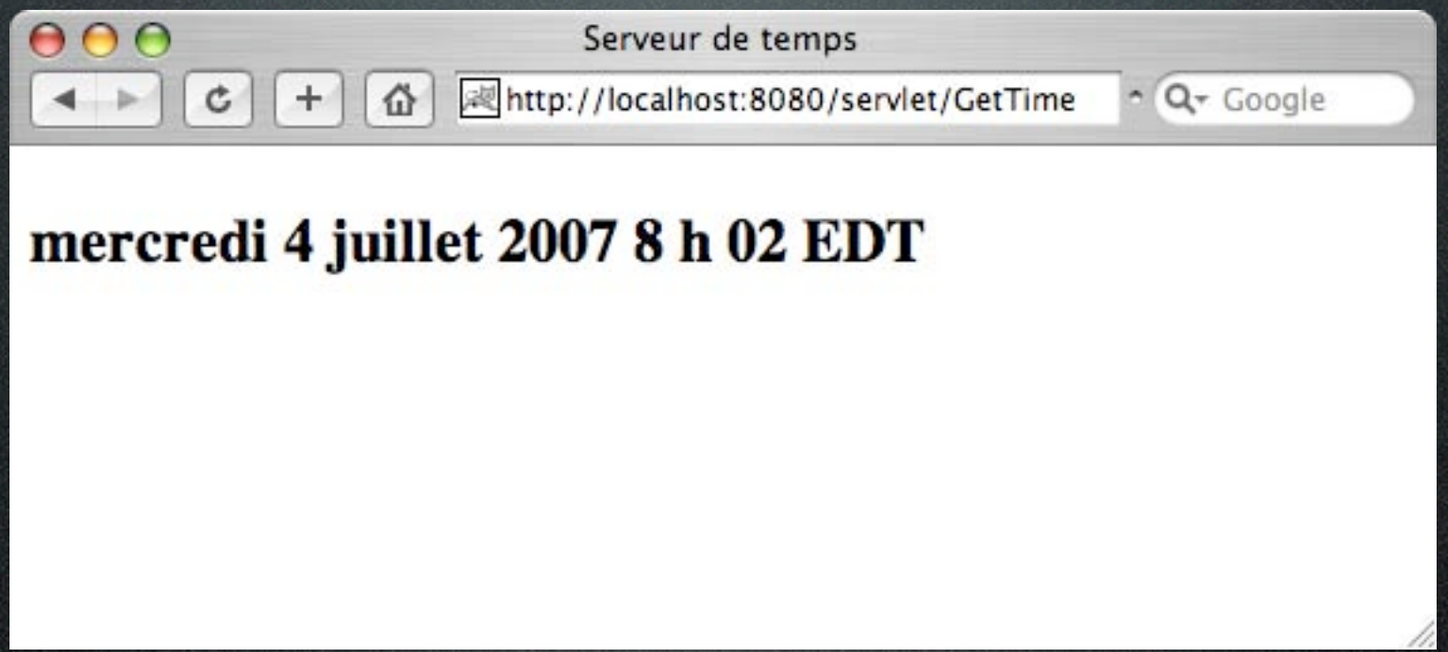
```
DateFormat df = DateFormat.getDateTimeInstance( DateFormat.FULL, DateFormat.FULL, Locale.CANADA_FRENCH );
String resultat = df.format( new Date() );

doc.println( "<!DOCTYPE html" );
doc.println( "          PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\"" );
doc.println( "          \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
doc.println( "<html xmlns=\"http://www.w3.org/1999/xhtml\" encoding=\"text/html; charset=\"UTF-8\"\" lang=\"fr-CA\">" );
doc.println( "  <head>" );
doc.println( "    <title>GetTime</title>" );
doc.println( "  </head>" );
doc.println( "  <body style=\"font-size:x-large\">" );
doc.println( "    <p>" );
doc.println( "      <b>" + resultat + "</b>" );
doc.println( "    </p>" );
doc.println( "  </body>" );
doc.println( "</html>" );
```

Quelle heure est-il ?

1. Compiler l'application
2. Déployer l'application
3. Visiter

http://localhost:8080/servlet/GetTime

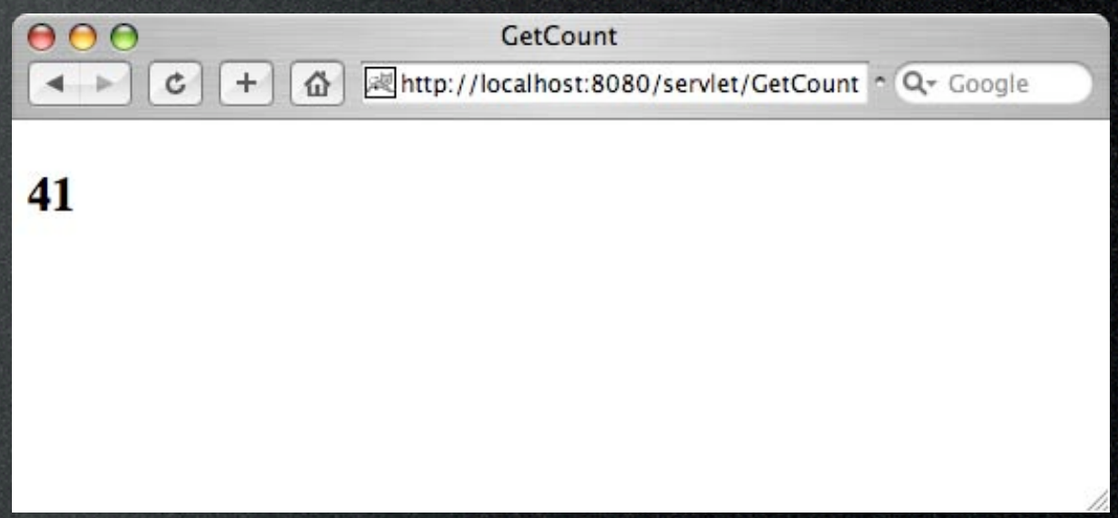
Quelle heure est-il ?



Savez-vous compter ?

- **Problème**
  - Concevoir une application (**Servlet**) qui affiche le **nombre de visites** depuis son démarrage

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class GetCount extends HttpServlet {

  private int count = 0;

    public void doGet( HttpServletRequest requete, HttpServletResponse reponse )
      throws ServletException, IOException {

        reponse.setContentType( "text/html; charset=\"UTF-8\"" );
        PrintWriter doc = reponse.getWriter();

        count++;

        doc.println( "<!DOCTYPE html" );
         // ...
        doc.println( "      <b>" + count + "</b>" );
         // ...
        doc.println( "</html>" );

        doc.close();
    }
}
```



Cycle de vide d'un servlet

1. **init**()
2. **service**()
3. **destroy**()

Cycle de vide d'un servlet : init()

- Le conteneur de Servlets (Tomcat ou GlassFish, par exemple) invoque cette méthode au démarrage du Servlet
- Par défaut, init() ne fait rien!
- On redéfinit cette méthode afin de lire le contenu d'un fichier (lire la valeur initiale du compteur), établir une connexion avec une base de données, etc.

Cycle de vide d'un servlet : service()

- Pour chaque requête HTTP, le conteneur fait un appel à la méthode **service**()
- La méthode **service**() appel la méthode **doGet**() ou **doPost**() selon le cas (type de requête)

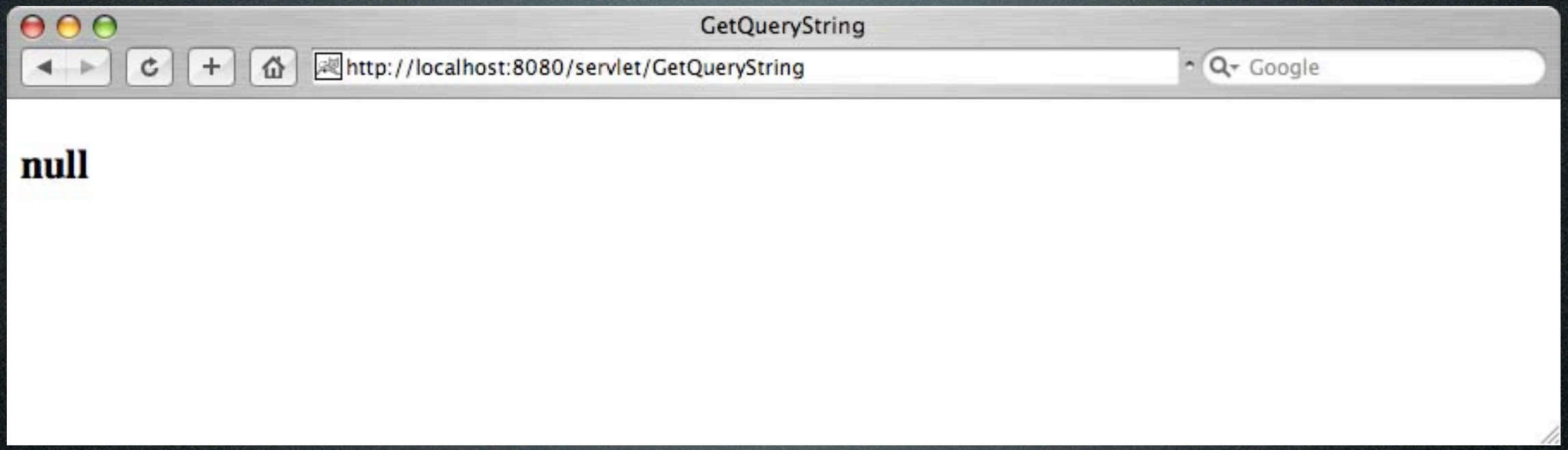
Cycle de vide d'un servlet : destroy()

- Cette méthode est appelée lors de l'arrêt du Servlet (et aussi donc lors de l'arrêt du conteneur)
- Par défaut, elle ne fait rien!
- On redéfinit cette méthode afin de sauvegarder des informations pour les exécutions subséquentes (la valeur finale du compteur), fermer une connexion avec une base de données, etc.

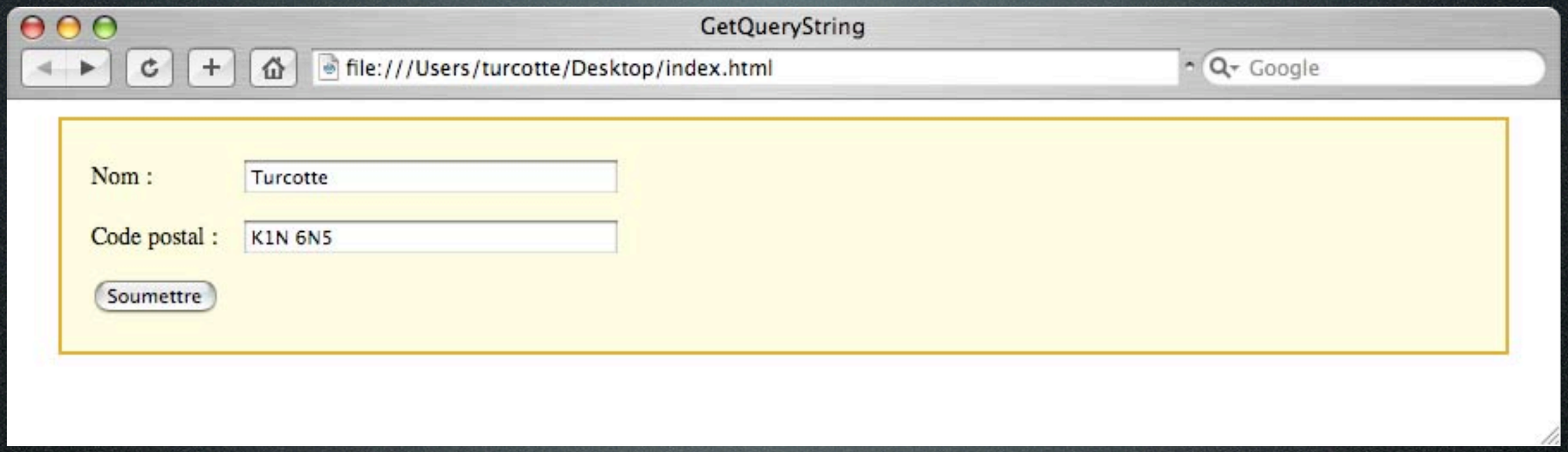
Accès aux données : doGet()

- Les données sont sauvegardées dans l'objet **HttpServletRequest**
- Dans le cas d'une requête GET, on se souviendra que les informations sont encodées à même l'URL
- La méthode **getQueryString**() retourne la portion de l'URL qui suit le symbole ? ou **null** si cette portion est absente

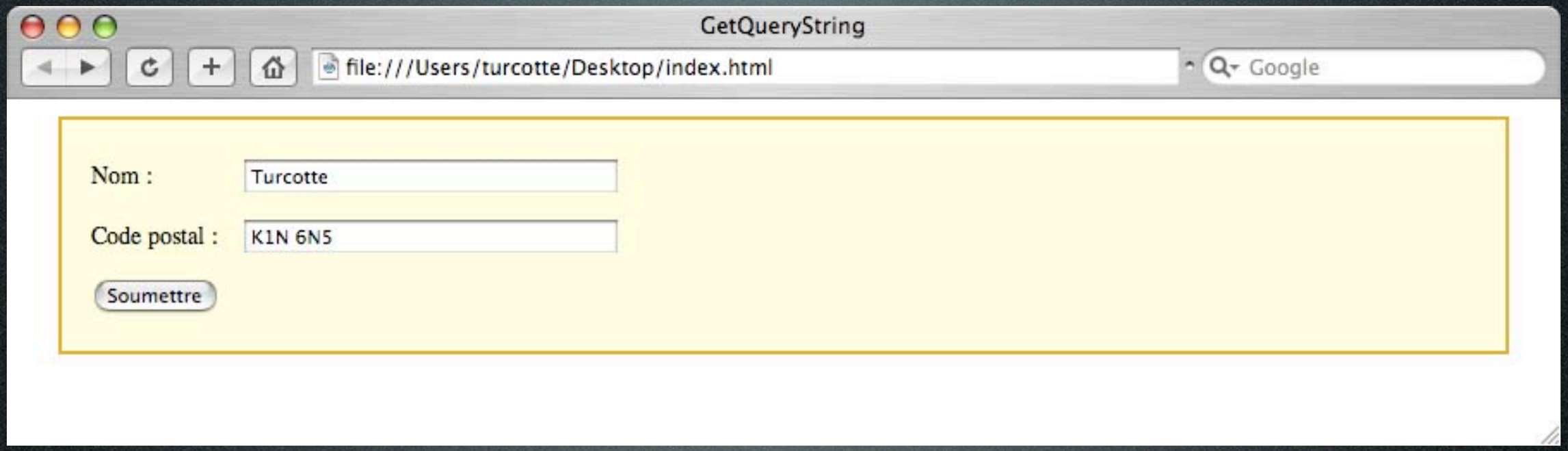
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class GetQueryString extends HttpServlet {

   public void doGet( HttpServletRequest requete, HttpServletResponse reponse )
      throws ServletException, IOException {

      reponse.setContentType( "text/html; charset=\"UTF-8\"" );
      PrintWriter doc = reponse.getWriter();

      doc.println( "<!DOCTYPE html" );
       // ...
      doc.println( "      <b>" + requete.getQueryString() + "</b>" );
       // ...
      doc.println( "</html>" );

      doc.close();
   }
}
```


GetQueryString
http://localhost:8080/servlet/GetQueryString
Google
null

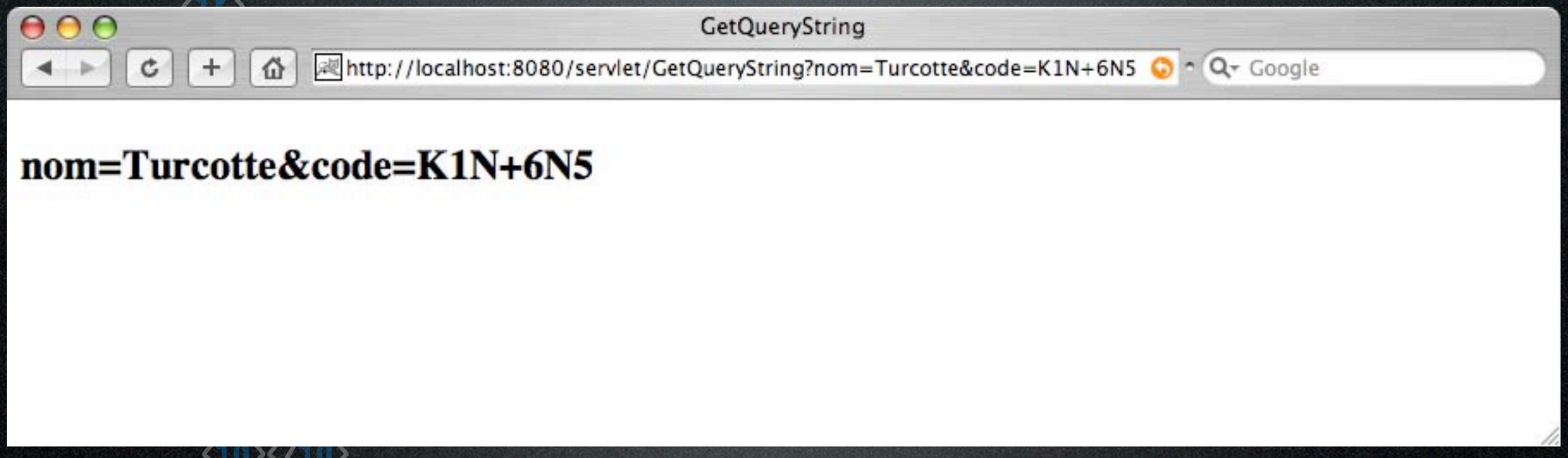
GetQueryString
http://localhost:8080/servlet/GetQueryString?nom=Turcotte&code=K1N+6N5
Google
nom=Turcotte&code=K1N+6N5




```
<form action="http://localhost:8080/servlet/GetQueryString" method="get">
 <table border="0" cellpadding="5">
  <tr>
   <td><label for="nom">Nom :</label></td>
   <td><input type="text" size="30" name="nom" /></td>
  </tr>
  <tr>
   <td><label for="code">Code postal :</label></td>
   <td><input type="text" size="30" name="code" /></td>
  </tr>
  <tr>
   <td><input type="submit" value="Soumettre" /></td>
   <td></td>
  </tr>
 </table>
</form>
```




```
<form action="http://localhost:8080/servlet/GetQueryString" method="get">
 <table border="0" cellpadding="5">
  <tr>
   <td></td>
  </tr>
 </table>
</form>
```



Google

http://www.google.ca/

Google

iGoogle | My Account | Sign out

Google™ Canada

Web Images Groups News Maps Scholar **more »**

servlet getQueryString

Advanced Search
Preferences
Language Tools

Google Search I'm Feeling Lucky

Search: the web pages from Canada

Google.ca offered in: Français

Advertising Programs - Business Solutions - About Google - Go to Google.com

servlet getQueryString - Google Search

http://www.google.ca/search?hl=en&q=servlet+getQueryString&btnG=Google+Search&meta=

Google

Web History | My Account | Sign out

Google

Web Images Groups News Maps Scholar **more »**

servlet getQueryString Search

Advanced Search
Preferences

Search: the web pages from Canada

Web

Personalized Results **1** - **10** of about **50,900** for **servlet getQueryString**. (**0.60** seconds)

: Interface HttpServletRequest
Returns any extra path information after the **servlet** name but before the query string, and translates it to a real path. java.lang.String, **getQueryString**() ...
java.sun.com/products/**servlet**/2.2/ javadoc/javax/**servlet**/http/Http**Servlet**Request.html - 38k - Cached - Similar pages

Interface javax.**servlet**.http.HttpServletRequest
An HTTP **servlet** request. This interface gets data from the client to the **servlet** for use in ....
o **getQueryString** public abstract String **getQueryString**() ...
java.sun.com/products/**servlet**/2.1/ api/javax.**servlet**.http.Http**Servlet**Request.html - 19k - Cached - Similar pages

Getting the Requesting URL in a **Servlet** (Java Developers Almanac ...
getServletPath(); // /**servlet**/MyServlet String pathInfo = req.getPathInfo(); // /a/b;c=123 String queryString = req.**getQueryString**(); // d=789 ...
www.exampledepot.com/egs/javax.**servlet**/GetReqUrl.html - 8k - Cached - Similar pages

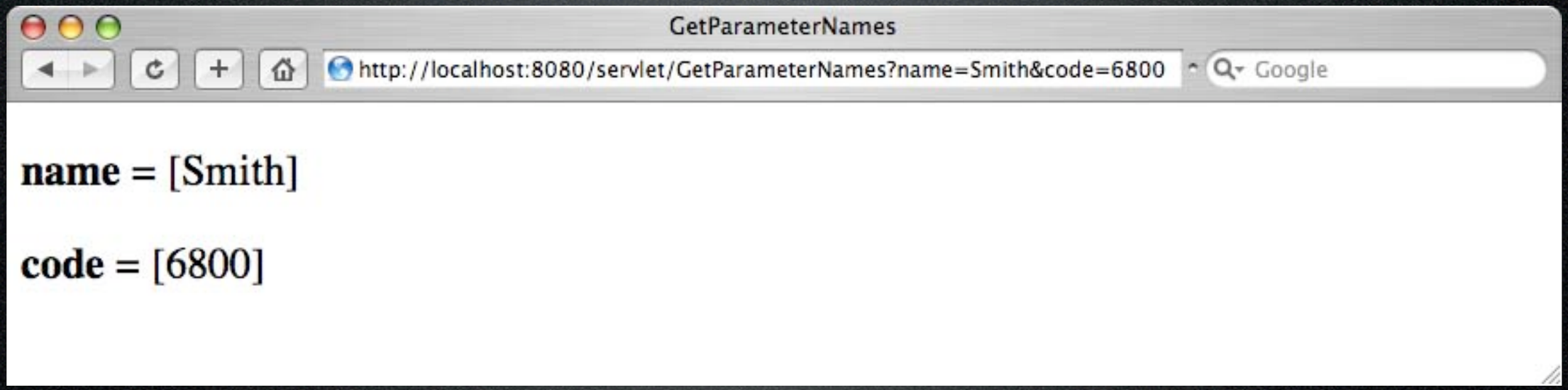
getQueryString()

- Chaque paramètre est une chaîne de la forme nom=valeur
- S'il y a plusieurs paramètres, il sont séparés par le symbole &
- L'ordre des paramètres est sans importance

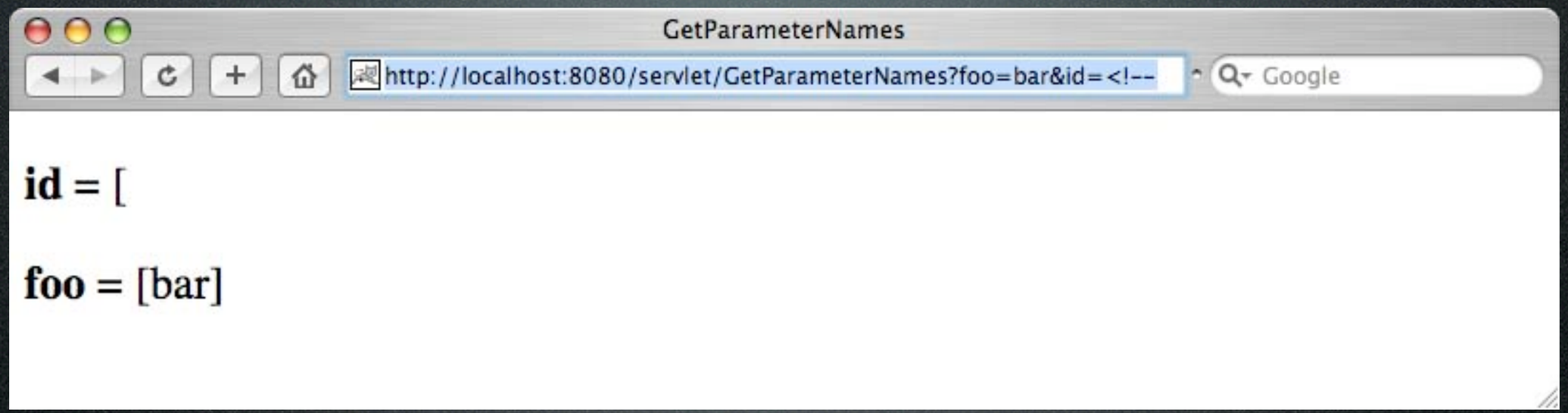
getQueryString()

- Les noms et valeurs sont composés de caractères 8 bits
- Les caractères non alphanumérique sont encodés (URL encoding)
- Si espace alors **+**
- Sinon, **% valeur hexadécimale**

```
Enumeration<String> names = requete.getParameterNames();
while ( names.hasMoreElements() ) {
   String name = names.nextElement();
   doc.println( "    <p>" );
   doc.println( "      <b>" + name + "</b> = " );
   String[] values = requete.getParameterValues( name );
   for ( int i=0; i<values.length; i++ ) {
     doc.println( "[" + values[ i ] + "]" );
   }
   doc.println( "    </p>" );
}
```



http://localhost:8080/servlet/GetParameterNames?foo=bar&id=<!--



- Il y a un problème, quel est-il ?
  - Certains caractères, tels que <, > et & ont une sémantique XHTML associée et doivent être remplacés par une entité référence (< > &)

GetParameterNames

file:///Users/turcotte/Desktop/index.html

Google

Nom : Christophe Salengro

Code postal : H1A 2A3

Pays : Principauté du Groland

Sexe : Homme : Femme :

Soumettre

GetParameterNames

?nom=Christophe+Salengro&code=H1A+2A3&pays=Groland&sexe=m

Google

nom=Christophe+Salengro&code=H1A+2A3&pays=Groland&sexe=m

nom = [Christophe Salengro]

sexe = [m]

code = [H1A 2A3]

pays = [Groland]

```
<form action="http://localhost:8080/servlet/GetParameterNames" method="get">
  <table border="0" cellpadding="5">
    <tr>
      <td>Nom :</td>
      <td><input type="text" size="30" name="nom" /></td>
    </tr>
    <tr>
      <td>Code postal :</td>
      <td><input type="text" size="30" name="code" /></td>
    </tr>
    <tr>
      <td>Pays :</td>
      <td>
        <select name="pays">
          <option value="none" selected="selected">
            Sélectionnez un pays :
          </option>
          <option value="Canada">Canada</option>
          <option value="Groland">Principauté du Groland</option>
          <option value="Syldavie">Syldavie</option>
        </select>
      </td>
    </tr>
    <tr>
      <td>Sexe :</td>
      <td>
        Homme : <input type="radio" checked="checked" name="sexe" value="m"/>
        Femme : <input type="radio" name="sexe" value="f"/>
      </td>
    </tr>
    <tr>
      <td><input type="submit" value="Soumettre" /></td>
      <td></td>
    </tr>
  </table>
</form>
```

doPost()

Les URLs ont généralement une taille d'au plus quelques milliers de caractères, ce qui limite la taille des paramètres

- <form … method="post">
- L'agent utilisateur passe les paramètres (nom-valeur) dans le corps du message (plutôt que l'URL)
- Virtuellement aucune limite sur la taille des paramètres
- Le fureteur alerte l'usager s'il tente accidentellement de soumettre à nouveau la requête (option préférée)

doGet()

- Les paramètres sont passés à même l'URL
- Les paramètres ont donc une taille limitée
- **On peut sauvegarder l'information sous forme de signet (bookmark) afin de la soumettre à nouveau**

Développement

Ant, War, GlassFish

Développement d'une application Web

- **Ant** pour automatiser les tâches connexes
- Fichier **WAR** pour un déploiement facile
- **GlassFish** comme conteneur des applications

Qu'est-ce que «**ant**» ?

- Un outil pour l'**automatisation des tâches** (répétitives) **liées au développement** de logiciels : par exemple, la compilation et l'installation des programmes

- Contrairement à **make**, un utilitaire populaire sous Unix :

  - Les fichiers de configuration sont écrits en format **XML**

  - Les extentions sont en **Java**

Qu'est-ce que «**Apache Ant**» ?

- Conçu par James Duncan Davidson lors du développement de **Apache Tomcat**
- Donc **bien adapté** au développement d'applications Web
- **Ant** = «**Another Neat Tool**»
- **Eclipse** et **NetBeans**, deux environnements de développement intégré (IDE) très populaires, utilisent tous deux ant

Apache Ant

- Universel
- Multiplateforme
- Capacité d'extension
- Code source libre

Hello World

- **Problématique** : automatiser les tâches répétitives liées au développement d'une application Java qui affiche la chaîne de caractères «Hello World!»

- En particulier, il faut **1)** créer la structure des fichiers et des répertoires, **2)** compiler les programmes, **3)** créer une archive (fichier jar) et **4)** exécuter le programme résultant

1. Préparation

- mkdir src
- mkdir src/bonjour
- mkdir build
- mkdir build/classes
- mkdir build/jar

src/bonjour/HelloWorld.java

```
package bonjour;

public class HelloWorld {
    public static void main( String[] args ) {
        System.out.println( "Howdy!" );
    }

}
```

2. Compilation

- javac -sourcepath src -d build/classes src/bonjour/HelloWorld.java

3. Créer l'archive

- echo "Main-Class: bonjour.HelloWorld" > MANIFEST.MF
- jar cvfm build/jar/HelloWorld.jar MANIFEST.MF -C build/classes .

4. Exécution

- java -cp build/classes bonjour.HelloWorld

  ou encore

- java -jar build/jar/HelloWorld.jar

Remarques

- Lors du développement de l'application, ces tâches seront répétées à plusieurs reprises
- Pour bien des projets, les tâches sont les mêmes

ant : build.xml

```
<project>

    <target name="clean">
        <delete dir="build"/>
    </target>

    <target name="compile">
        <mkdir dir="build/classes"/>
        <javac srcdir="src" destdir="build/classes"/>
    </target>

    <target name="jar">
        <mkdir dir="build/jar"/>
        <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
            <manifest>
                <attribute name="Main-Class" value="bonjour.HelloWorld"/>
            </manifest>
        </jar>
    </target>

    <target name="run">
        <java jar="build/jar/HelloWorld.jar" fork="true"/>
    </target>

</project>
```

ant : build.xml

- **build.xml** est le nom par défaut du fichier de configuration
- C'est un fichier XML et la racine est l'élément **project**
- Un projet est constitué d'un ensemble de cibles, éléments **target**
- Une cible est constituée d'un ensemble de **tâches**, par exemple : delete, mkdir, javac, jar, java, ...

ant : build.xml

```
<project>

   <target name="clean">
      <delete dir="build"/>
   </target>

   <target name="compile">
      <mkdir dir="build/classes"/>
      <javac srcdir="src" destdir="build/classes"/>
   </target>

   <target name="jar">
      <mkdir dir="build/jar"/>
      <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
         <manifest>
            <attribute name="Main-Class" value="bonjour.HelloWorld"/>
         </manifest>
      </jar>
   </target>

   <target name="run">
      <java jar="build/jar/HelloWorld.jar" fork="true"/>
   </target>

</project>
```

Cibles

Tâches

session

- ant compile
  Buildfile: build.xml

  compile:
      [mkdir] Created dir: build/classes
      [javac] Compiling 1 source file to build/classes

  BUILD SUCCESSFUL
  Total time: 4 seconds

- ant jar
  Buildfile: build.xml

  jar:
      [mkdir] Created dir: build/jar
        [jar] Building jar: jar/HelloWorld.jar

  BUILD SUCCESSFUL
  Total time: 1 second

- ant run
  Buildfile: build.xml

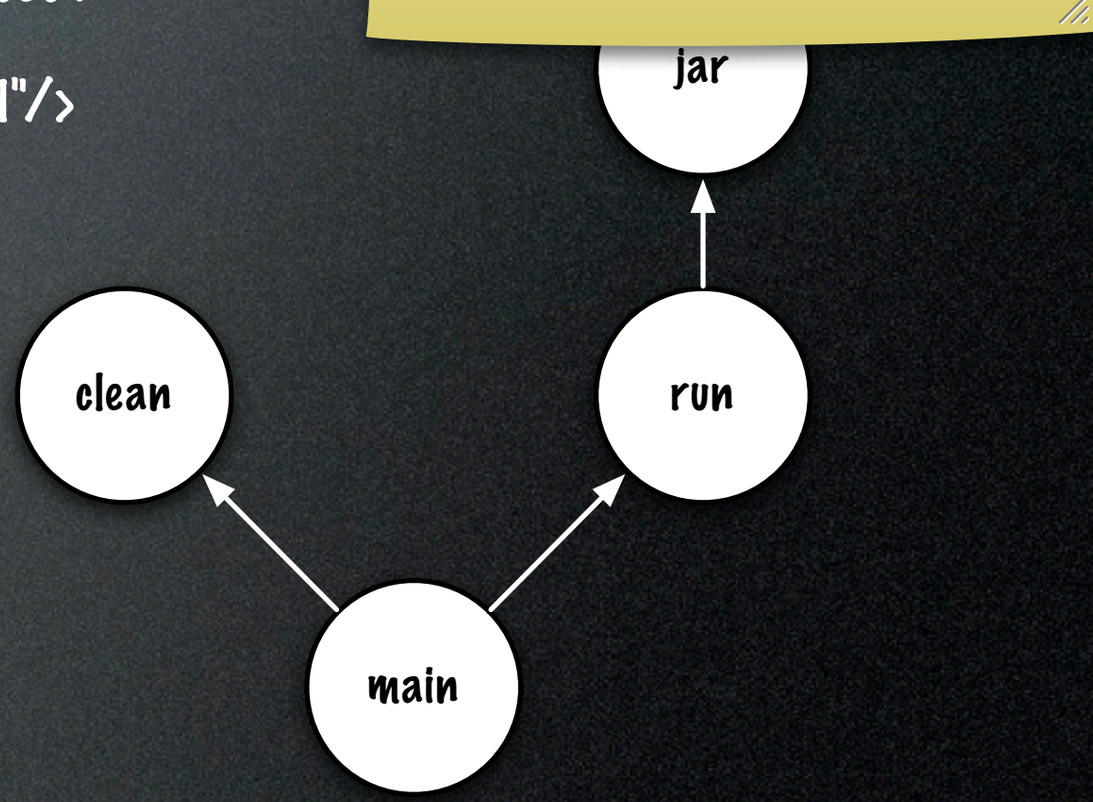
  run:
      [java] Howdy!

  BUILD SUCCESSFUL
  Total time: 0 seconds

dépendances

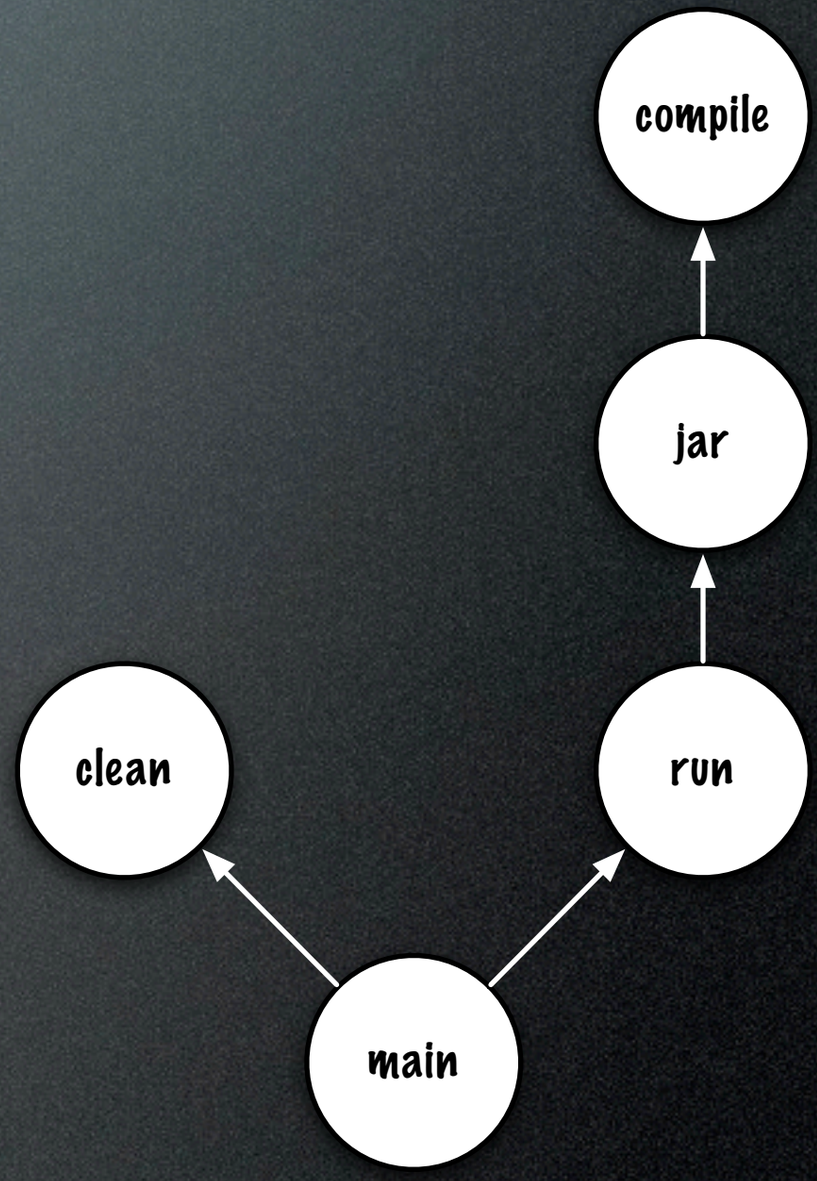
```
<project name="HelloWorld" basedir="." default="main">

  <target name="clean">
    <delete dir="build"/>
  </target>

  <target name="compile">
    <mkdir dir="build/classes"/>
    <javac srcdir="src" destdir="build/classes"/>
  </target>

  <target name="jar" depends="compile">
    <mkdir dir="build/jar"/>
    <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
      <manifest>
        <attribute name="Main-Class" value="bonjour.HelloWorld"/>
      </manifest>
    </jar>
  </target>

  <target name="run" depends="jar">
    <java jar="build/jar/HelloWorld.jar" fork="true"/>
  </target>

  <target name="clean-build" depends="clean,jar"/>
  <target name="main" depends="clean,run"/>

</project>
```

name = nom du projet
basedir = répertoire par rapport auquel les chemins relatifs sont exprimés
default = cible par défaut, i.e. la cible utilisée lorsqu'aucune cible n'est spécifiée sur la ligne de commande



dépendances

```
> ant
Buildfile: build.xml

clean:
   [delete] Deleting directory build

compile:
    [mkdir] Created dir: build/classes
    [javac] Compiling 1 source file to build/classes

jar:
    [mkdir] Created dir: build/jar
     [jar] Building jar: build/jar/HelloWorld.jar

run:
    [java] Howdy!

main:

BUILD SUCCESSFUL
Total time: 4 seconds
```



ant : build.properties

```
<project name="HelloWorld" basedir="." default="main">

  <property file="build.properties"/>

  <target name="clean">
    <delete dir="${build.dir}"/>
  </target>

  <target name="compile">
    <mkdir dir="${classes.dir}"/>
    <javac srcdir="${src.dir}" destdir="${classes.dir}"/>
  </target>
  ...
  <target name="main" depends="clean,run"/>

</project>
```

build.properties :

```
src.dir=src
build.dir=build
classes.dir=${build.dir}/classes
jar.dir=${build.dir}/jar
main-class=bonjour.HelloWorld
```

ant : documentation

```
<project name="HelloWorld" basedir="." default="main">

  <property file="build.properties"/>

  <target name="clean" description="Deletes all the build files" >
    <delete dir="${build.dir}"/>
  </target>

  <target name="compile" description="Compiles the source code" >
    <mkdir dir="${classes.dir}"/>
    <javac srcdir="${src.dir}" destdir="${classes.dir}"/>
  </target>
  ...
  <target name="main"
          description="Default target"
          depends="clean,run"/>

</project>
```

```
> ant -p
Buildfile: build.xml

Main targets:

 clean    Deletes all the build files
 compile  Compiles the source code
 jar      Packages the application's archive file
 main     Default target
 run      Executes the program
Default target: main
```

ant : résumé

- Fichiers : build.xml et build.properties
- Définition des cibles (targets) liées au projet
- Mais aussi des dépendances entre les cibles
- Plusieurs tâches prédéfinies, entre autres plusieurs tâches liées au développement d'applications Web, comme nous le verrons bientôt

Web Archive : WAR

- Un fichier **WAR** ce n'est que :
  - Un fichier **JAR** dont le suffixe est WAR
  - Une **structure de répertoires et de fichiers bien définie**
- C'est donc un fichier ZIP pour distribuer des classes, des Servlets, des documents XHTML statiques et dynamiques, ainsi que des métadonnées

WAR : Hello World

- Créer un fichier nommé **index.html** :

```
<!DOCTYPE html
          PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
                 "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr-CA">
  <head>
    <title>Ma première application Web</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body style="font-size:x-large">
    <p>
      Ma première application Web et son contenu statique.
    </p>
  </body>
</html>
```

WAR : Hello World

Façon 1

1. jar cf test.war index.html

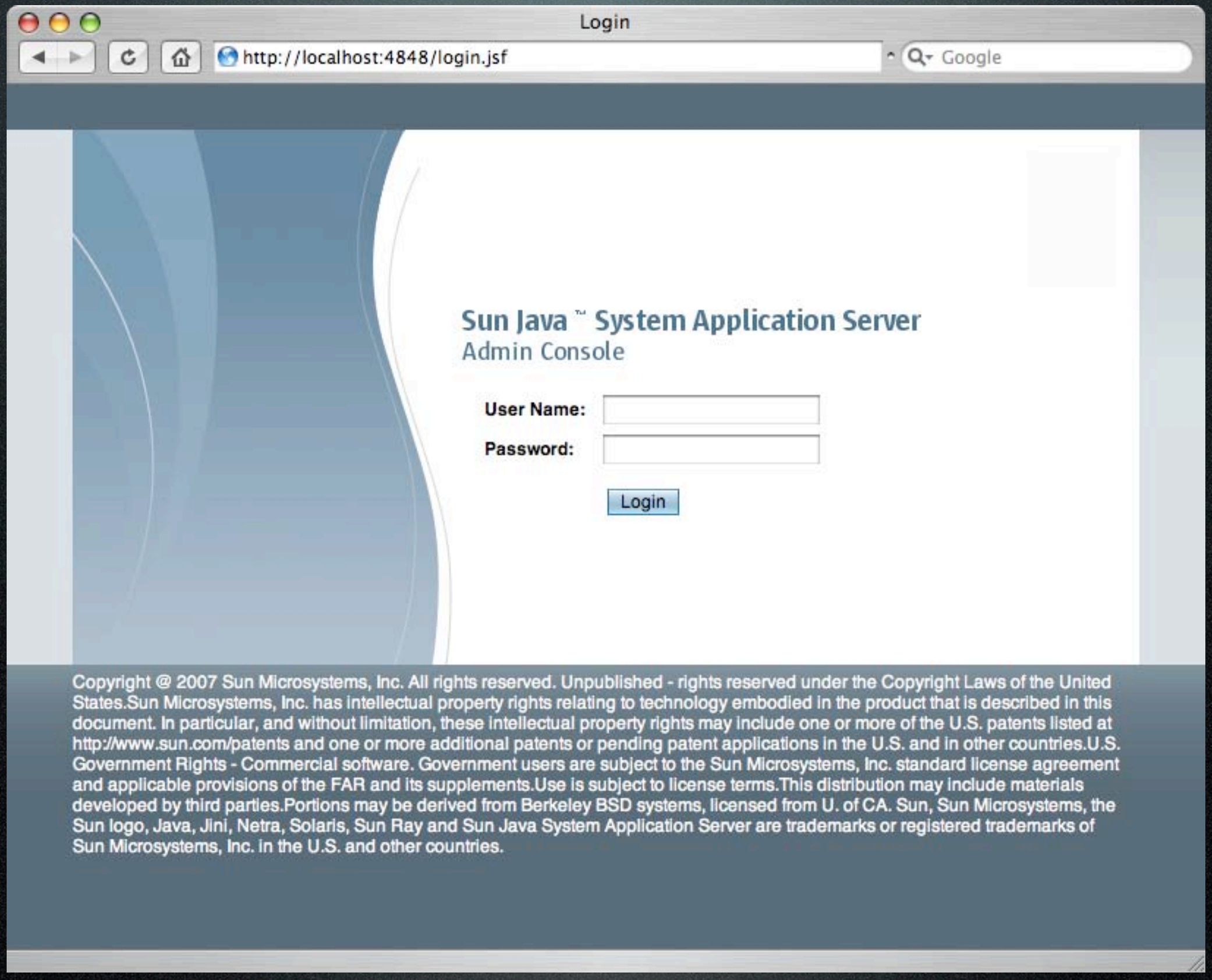
2. cp test.war $GLASSFISH_DIR/domains/domain1/autodeploy

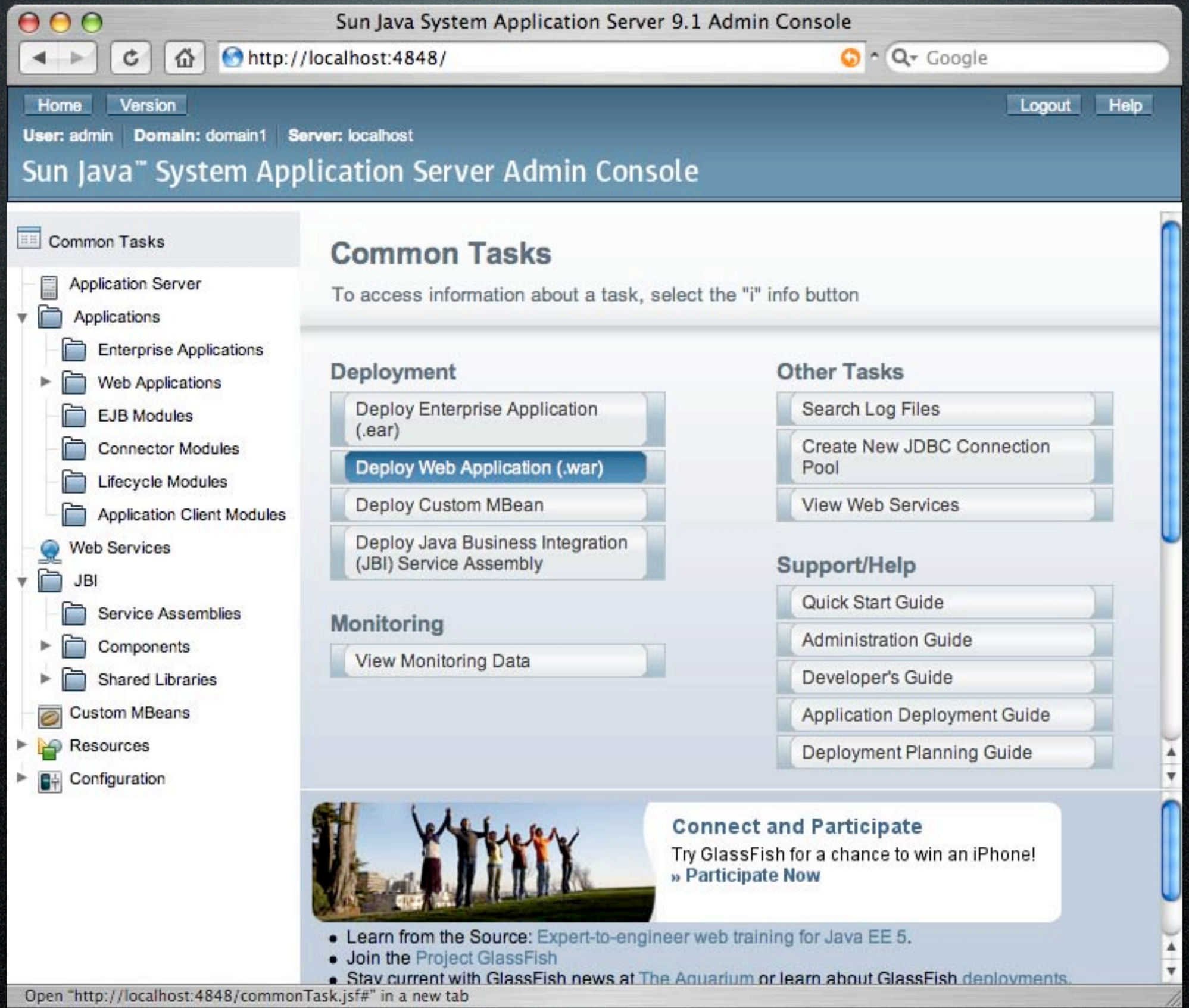
$GLASSFISH_DIR/bin/asadmin start-domain domain1

WAR : Hello World

Façon 2

1. jar cf test.war index.html
2. Visitez l'URL suivante : http://localhost:4848/


Login
http://localhost:4848/login.jsf
Google
Sun Java ™ System Application Server
Admin Console
User Name:
Password:
Login
Copyright @ 2007 Sun Microsystems, Inc. All rights reserved. Unpublished - rights reserved under the Copyright Laws of the United States.Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries.U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.Use is subject to license terms.This distribution may include materials developed by third parties.Portions may be derived from Berkeley BSD systems, licensed from U. of CA. Sun, Sun Microsystems, the Sun logo, Java, Jini, Netra, Solaris, Sun Ray and Sun Java System Application Server are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.


Sun Java System Application Server 9.1 Admin Console
http://localhost:4848/
Google
Home
Version
Logout
Help
User: admin
Domain: domain1
Server: localhost
Sun Java™ System Application Server Admin Console
Common Tasks
Application Server
Applications
Enterprise Applications
Web Applications
EJB Modules
Connector Modules
Lifecycle Modules
Application Client Modules
Web Services
JBI
Service Assemblies
Components
Shared Libraries
Custom MBeans
Resources
Configuration
Common Tasks
To access information about a task, select the "i" info button
Deployment
Deploy Enterprise Application (.ear)
Deploy Web Application (.war)
Deploy Custom MBean
Deploy Java Business Integration (JBI) Service Assembly
Monitoring
View Monitoring Data
Other Tasks
Search Log Files
Create New JDBC Connection Pool
View Web Services
Support/Help
Quick Start Guide
Administration Guide
Developer's Guide
Application Deployment Guide
Deployment Planning Guide
Connect and Participate
Try GlassFish for a chance to win an iPhone!
» Participate Now
Learn from the Source: Expert-to-engineer web training for Java EE 5.
Join the Project GlassFish
Stay current with GlassFish news at The Aquarium or learn about GlassFish deployments.
Open "http://localhost:4848/commonTask.jsf#" in a new tab

Sun Java System Application Server 9.1 Admin Console

http://localhost:4848/

Google

Home | Version | Logout | Help

User: admin | **Domain:** domain1 | **Server:** localhost

Sun Java™ System Application Server Admin Console

- Common Tasks
- Application Server
- Applications
  - Enterprise Applications
  - Web Applications
  - EJB Modules
  - Connector Modules
  - Lifecycle Modules
  - Application Client Modules
- Web Services
- JBI
  - Service Assemblies
  - Components
  - Shared Libraries
- Custom MBeans
- Resources
- Configuration

Applications > Web Applications

Deploy Enterprise Applications/Modules

OK | Cancel

Specify the location of an application to deploy. Applications can be in packaged files such .war, .ear, .jar, and .rar.

Type: Web Application (.war)

Location:

- (•) **Packaged file to be uploaded to the server**
  - Choose File | no file selected
- ( ) **Local packaged file or directory that is accessible from the Application Server**
  - Browse Files...
  - Browse Folders...

General

Application Name: *

Context Root:

Path relative to server's base URL

Virtual Servers: server

Associates an internet domain name with a physical server

Status: [x] **Enabled**

Run Verifier: [ ] **Enabled**

Sun Java System Application Server 9.1 Admin Console

http://localhost:4848/

Google

Home | Version | Logout | Help

User: admin | **Domain:** domain1 | **Server:** localhost

Sun Java™ System Application Server Admin Console

- Common Tasks
- Application Server
- Applications
  - Enterprise Applications
  - Web Applications
  - EJB Modules
  - Connector Modules
  - Lifecycle Modules
  - Application Client Modules
- Web Services
- JBI
  - Service Assemblies
  - Components
  - Shared Libraries
- Custom MBeans
- Resources
- Configuration

Applications > Web Applications

Deploy Enterprise Applications/Modules

OK | Cancel

Specify the location of an application to deploy. Applications can be in packaged files such .war, .ear, .jar, and .rar.

Type: Web Application (.war)

Location:

- (•) **Packaged file to be uploaded to the server**
  - Choose File | test.war
- ( ) **Local packaged file or directory that is accessible from the Application Server**
  - Browse Files...
  - Browse Folders...

General

Application Name: * test

Context Root: test

Path relative to server's base URL

Virtual Servers: server

Associates an internet domain name with a physical server

Status: [x] **Enabled**

Run Verifier: [ ] **Enabled**

Sun Java System Application Server 9.1 Admin Console

http://localhost:4848/

Google

Home | Version | Logout | Help

User: admin | **Domain:** domain1 | **Server:** localhost

Sun Java™ System Application Server Admin Console

- Common Tasks
- Application Server
- Applications
  - Enterprise Applications
  - Web Applications
    - Temperature
    - Test
    - converter
    - ConverterClient
    - Echo
    - motd
    - MotdClient
    - hello
    - snipsnap
    - test
  - EJB Modules
  - Connector Modules
  - Lifecycle Modules
  - Application Client Modules
- Web Services
- JBI
  - Service Assemblies
  - Components
  - Shared Libraries
- Custom MBeans

Applications > Web Applications

Web Applications

A Web application module consists of a collection of Web resources such as JavaServer Pages (JSPs), servlets, and HTML pages that are packaged in a WAR (Web Application Archive) file or directory.

Deployed Web Applications (10)

Deploy... | Undeploy | Enable | Disable

Name	Enabled	Context Root	Action
converter	true	converter	Launch | Redeploy
Temperature	true	Temperature	Launch | Redeploy
hello	true	/hello	Launch | Redeploy
snipsnap	true	snipsnap	Launch | Redeploy
ConverterClient	true	ConverterClient	Launch | Redeploy
Test	true	Test	Launch | Redeploy
motd	true	motd	Launch | Redeploy
Echo	true	Echo	Launch | Redeploy
MotdClient	true	MotdClient	Launch | Redeploy
test	true	test	Launch | Redeploy

WAR : Hello World
Façon 3

- Utilisez **asant** plutôt qu'**ant**
- C'est tout simplement une version d'**ant** livrée avec l'application **GlassFish** et ayant des **cibles** et des **tâches** prédéfinies pour **GlassFish**
- **$GLASSFISH_DIR/bin/asant**

WAR : Hello World
Façon 3

- Tous mes projets nécessitent (±) les mêmes cibles et tâches
- J'ai donc des fichiers
  build-common.properties et
  build-common.xml
  qui sont communs à tous mes projets

Voir polycopier
build-common.xml

WAR : Hello World
Façon 3

build.properties:

```
app.name=test
```

build.xml:

```
<project name="Web Application" default="build"
basedir=".">

 <property file="build.properties"/>
 <property file="../build-common.properties"/>

 <import file="../build-common.xml"/>

</project>
```

WAR : Hello World
Façon 3

- **build-common.properties**
  build-common.xml
  projet-01/build.properties
  projet-01/build.xml
  projet-01/src
  projet-01/...
  projet-02/build.properties
  projet-02/build.xml
  ...

WAR : Hello World

Façon 3

```
> asant package
Buildfile: build.xml

init:
    [echo] Creating test's src and web directories ...

prepare:
    [echo] Creating test's war directory ...

build:
    [echo] Building test ...
    [copy] Copying 1 file to ./war

package:
    [echo] Packaging test's war file ...
     [jar] Building jar: ./test.war

BUILD SUCCESSFUL
Total time: 1 second
```

WAR : Hello World
Façon 3

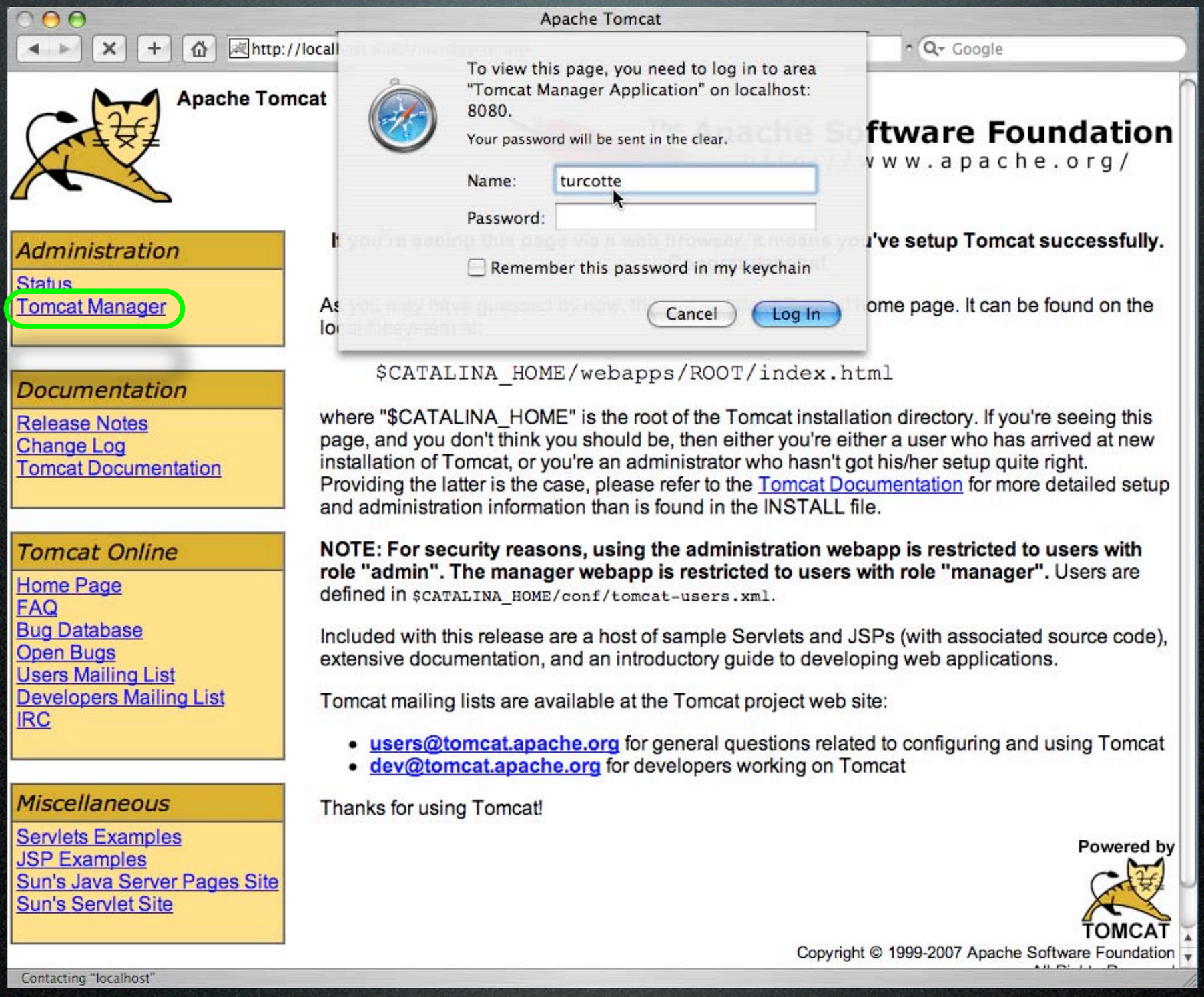
```
> asant undeploy
Buildfile: build.xml

undeploy:
[sun-appserv-undeploy] Executing: undeploy --user admin --passwordfile "../password.txt"
test
[sun-appserv-undeploy] Command undeploy executed successfully.

BUILD SUCCESSFUL
Total time: 1 second
```

```
> asant deploy
Buildfile: build.xml

init:
    [echo] Creating test's src and web directories ...

prepare:
    [echo] Creating test's war directory ...

build:
    [echo] Building test ...

package:
    [echo] Packaging test's war file ...
  [delete] Deleting: test.war
    [jar] Building jar: test.war

deploy:
[sun-appserv-deploy] Executing: deploy --user admin --passwordfile "../password.txt"
--force=true --enabled=true --name test --verify=false --precompilejsp=true --
upload=true "test.war"
[sun-appserv-deploy] Command deploy executed successfully.

BUILD SUCCESSFUL
Total time: 1 second
```

Déploiement avec Tomcat

- C'est très semblable
- En fait **GlassFish** est le successeur de **Tomcat**
- **GlassFish** ajoute les concepts de services Web, que nous verrons d'ici la fin du semestre

http://localhost:8080/manager/html



/manager

http://localhost:8080/manager/html/list

Google

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications				
Path	**Display Name**	**Running**	**Sessions**	**Commands**
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

/manager

http://localhost:8080/manager/html/list

Google

/sample	Hello, World Application	true	0	Expire sessions with idle ≥ 30 minutes
/snipsnap	SnipSnap 1.0b3-uttoxeter	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

WAR or Directory URL:

Deploy

WAR file to deploy

Select WAR file to upload Choose File no file selected

Deploy

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/6.0.13	1.5.0_07-164	"Apple Computer, Inc."	Mac OS X	10.4.10	ppc

/manager

http://localhost:8080/manager/html/upload

Google

/sample	Hello, World Application	true	0	Expire sessions with idle ≥ 30 minutes
/snipsnap	SnipSnap 1.0b3-uttoxeter	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (optional):

XML Configuration file URL:

WAR or Directory URL:

Deploy

WAR file to deploy

Select WAR file to upload Choose File Test.war

Deploy

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/6.0.13	1.5.0_07-164	"Apple Computer, Inc."	Mac OS X	10.4.10	ppc

/manager

http://localhost:8080/manager/html/upload

Google

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

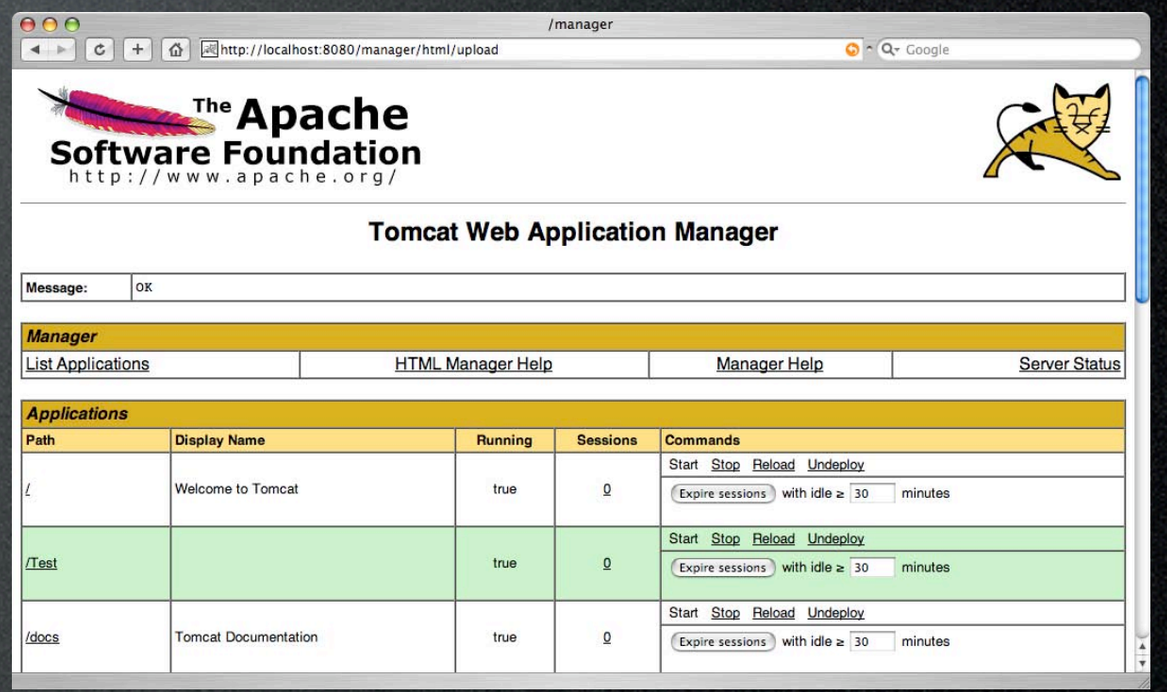
Applications				
Path	**Display Name**	**Running**	**Sessions**	**Commands**
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/Test		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes


Ma première application Web
http://localhost:8080/Test/
Google
Ma première application Web et son contenu statique.

Remarques

- Comme plusieurs serveurs Web, **Tomcat** affiche le document index.html s'il existe
- Le champ «Display Name» est vide

- Le nom du fichier WAR sert de contexte



métadonnées

- WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5">

  <display-name>Ma première application Web</display-name>

  <description>
     L'avenir appartient à ceux qui ...
  </description>

</web-app>
```

Descripteur de déploiement d'une application Web
Web Application Deployment Descriptor

métadonnées

/manager

http://localhost:8080/manager/html

Google

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications				
Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/Test	Ma première application Web	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

manager : html/text

- Faites glisser le curseur au dessus des liens «Stop», «Reload» et «Undeploy»

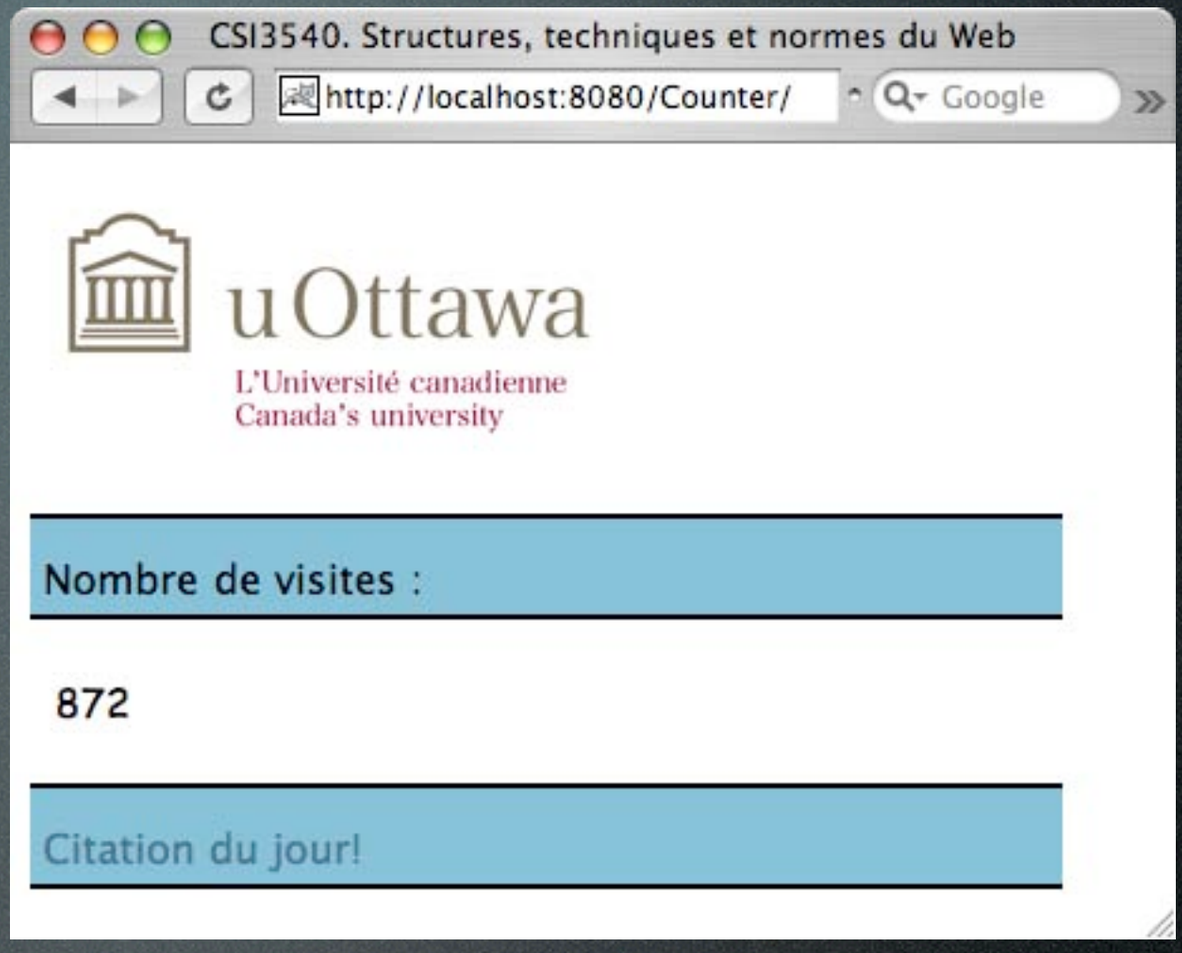
http://localhost:8080/manager/html/start?path=/Test
http://localhost:8080/manager/html/stop?path=/Test
http://localhost:8080/manager/html/reload?path=/Test
http://localhost:8080/manager/html/undeploy?path=/Test

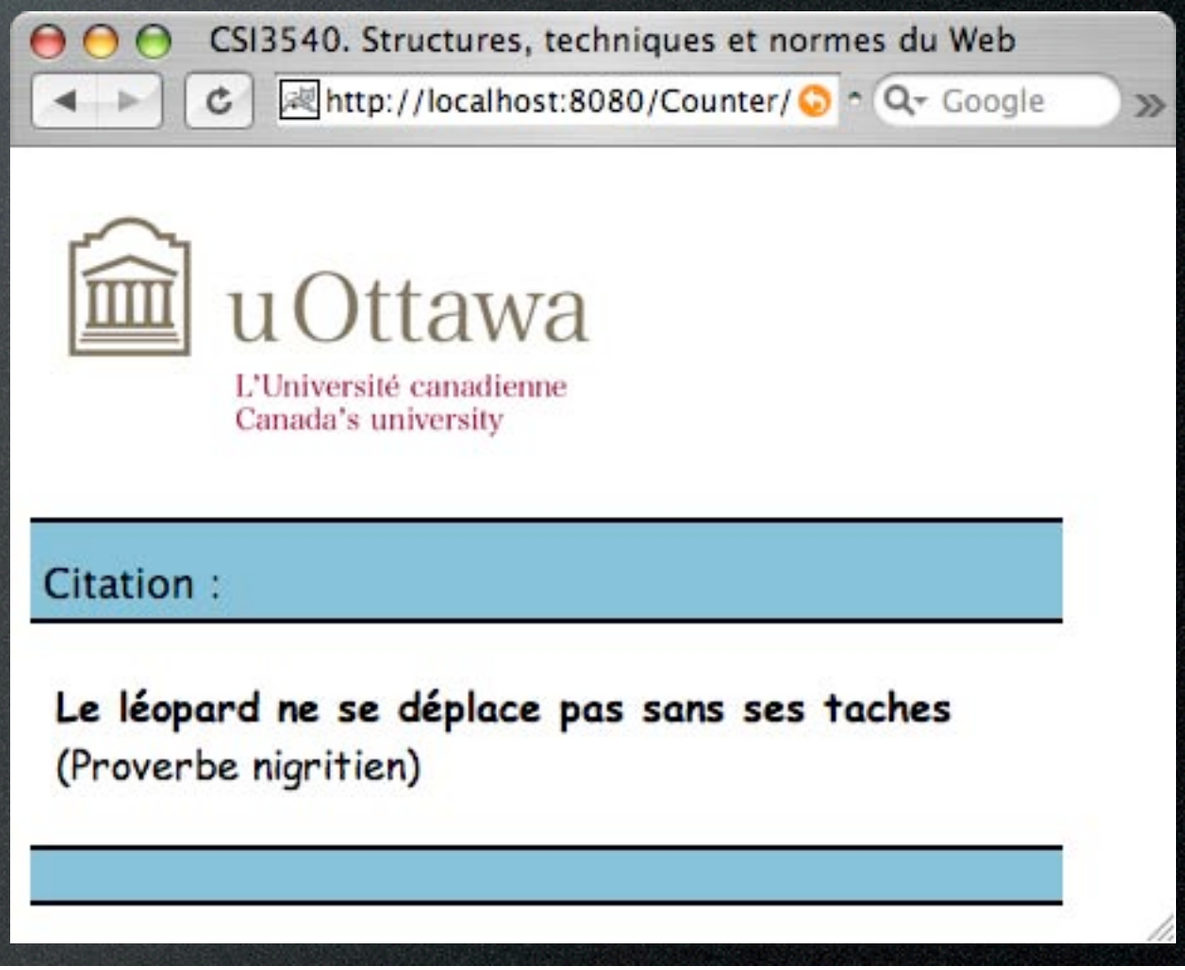
http://localhost:8080/manager/start?path=/Test
http://localhost:8080/manager/stop?path=/Test
http://localhost:8080/manager/reload?path=/Test
http://localhost:8080/manager/undeploy?path=/Test

Counter.war

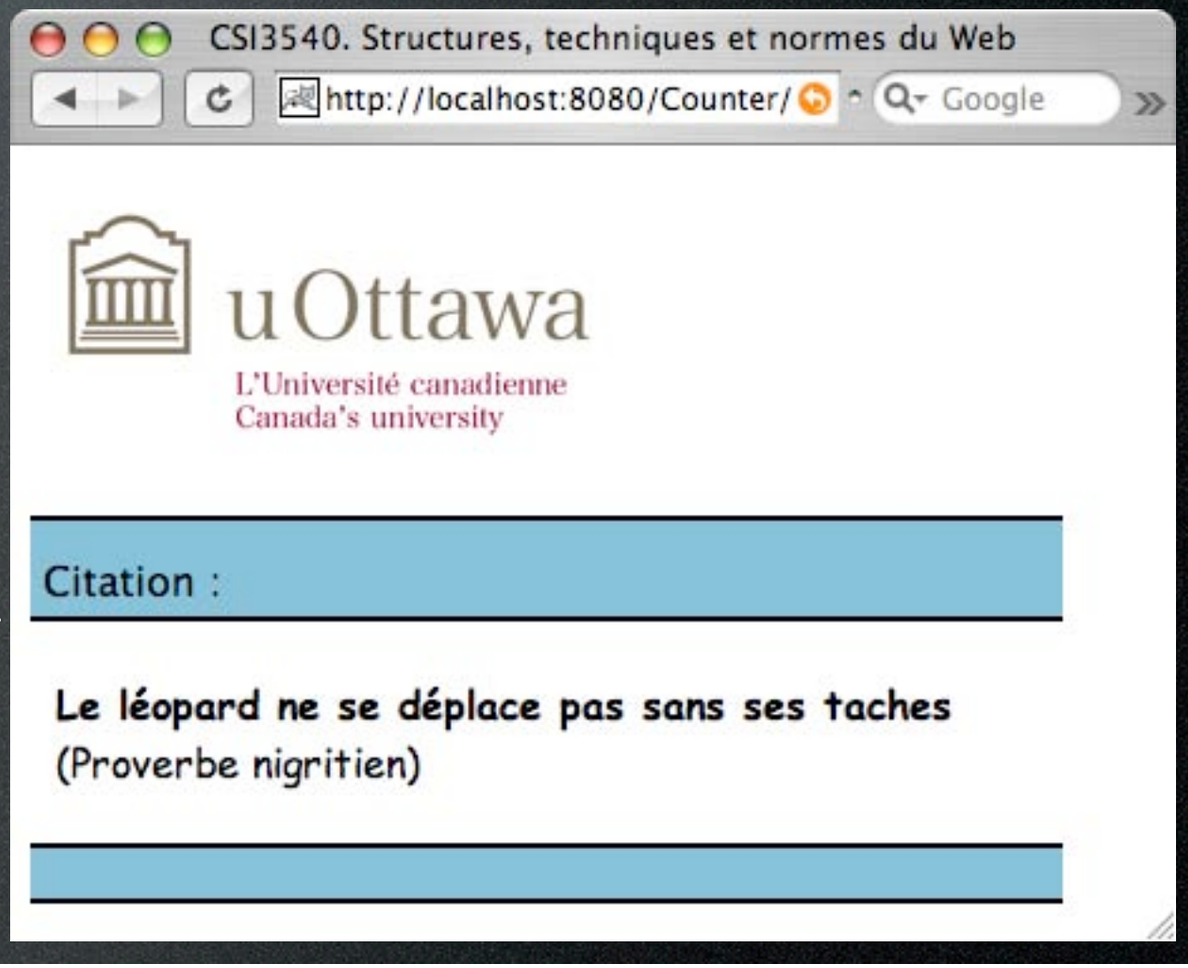
- Prototype d'une application Web complète :
  - **Page**(s) **statique**(s) (index.html)
  - **Feuille**(s) **de styles** (css/default.css)
  - **Image**(s) (images/uOttawa.png)
  - **Servlet**(s) (WEB-INF/classes/counter/GetCount.class)

```
./
  Counter.war
  build.properties
  build.xml
  ./src/counter/
          GetCount.java
  ./war/
      WEB-INF/
          classes/counter/
                  GetCount.class
          lib/
          web.xml
      css/
          default.css
      images/
          uOttawa.png
      index.html

./web/
      WEB-INF/
          lib/
          web.xml
      css/
          default.css
      images/
          uOttawa.png
      index.html
```



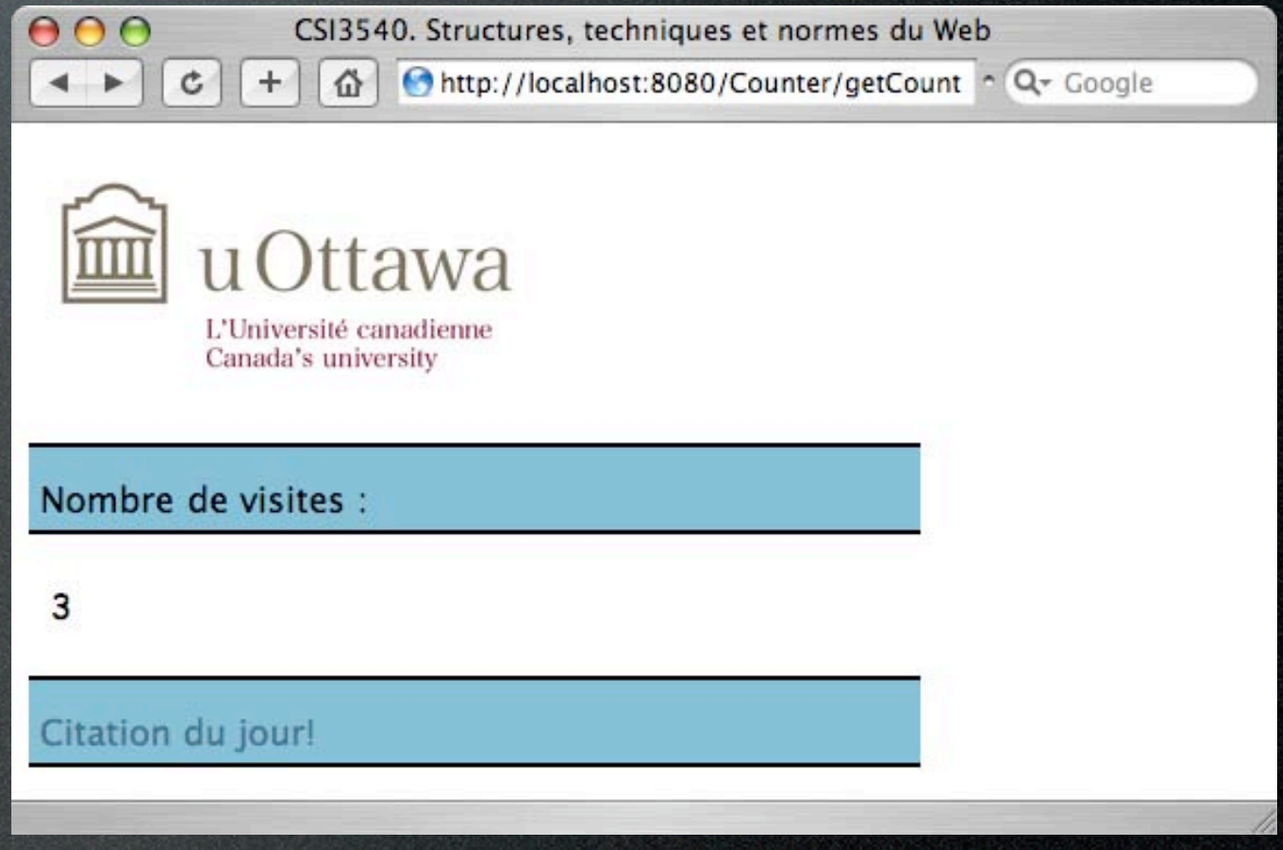


```
> ant
Buildfile: build.xml

init:
    [echo] Creating Counter's web directory ...

prepare:
    [echo] Creating Counter's war directory ...
   [mkdir] Created dir: war
   [mkdir] Created dir: war/WEB-INF
   [mkdir] Created dir: war/WEB-INF/classes
   [mkdir] Created dir: war/WEB-INF/lib

build:
    [echo] Building Counter ...
   [javac] Compiling 1 source file to war/WEB-INF/classes
    [copy] Copying 1 file to war/WEB-INF
    [copy] Copying 3 files to war

package:
    [echo] Packaging Counter's war file ...
     [jar] Building jar: Counter.war

deploy:
    [echo] Deploying Counter ...
  [deploy] OK - Deployed application at context path /Counter

BUILD SUCCESSFUL
Total time: 5 seconds
```



web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                             http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5">

  <display-name>Compteur d'accès</display-name>

  <description>
    Modèle d'une application Web.
  </description>

  <servlet>
    <servlet-name>Counter</servlet-name>
    <servlet-class>counter.GetCount</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Counter</servlet-name>
    <url-pattern>/getCount/*</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>getCount</welcome-file>
  </welcome-file-list>

</web-app>
```



web.xml

- Table 8.2, page 443 (principaux éléments)
- Page 445, URL pattern
- Table 8.3

Appendix

Tomcat

ant et tomcat

- La distribution de **tomcat** fournit plusieurs tâches **ant** liées au développement d'applications Web : deploy, undeploy, start, stop, install, remove, reload, list, resources, roles

ant et tomcat

- Plusieurs documents suggèrent de copier la bibliothèque
  ${tomcat.home}/lib/catalina-ant.jar
  soit dans
  ${ant.home}/lib
  ou encore
  ${user.home}/.ant/lib

- Ce qui peut nécessiter les privilèges de l'administrateur, et vous devrez le refaire pour chaque environnement

ant et tomcat

- Je vous suggère d'ajouter ceci à votre fichier build.xml :

```
<import file="${tomcat.home}/bin/catalina-tasks.xml"/>
```

ant : deploy

```
> ant deploy
Buildfile: build.xml

prepare:
    [echo] Creating Test's war directory ...
   [mkdir] Created dir: war

build:
    [echo] Building Test ...
    [copy] Copying 1 file to war/WEB-INF
    [copy] Copying 1 file to war

package:
    [echo] Packaging Test's war file ...
     [jar] Building jar: Test.war

deploy:
    [echo] Deploying Test ...
  [deploy] OK - Deployed application at context path /Test

BUILD SUCCESSFUL
Total time: 1 second
```

Ressources (suite)

- Apache Ant [http://ant.apache.org] 2007

Appendice

Configurer Tomcat

Déploiement

C'est l'approche du bouquin, la notre sera différente!

1. Compiler la classe GetTime

   > javac -cp ../../../Apache-Tomcat/6.0.13/lib/servlet-api.jar GetTime.java

2. Copier le .class dans le répertoire :

   webapps/ROOT/WEB-INF/classes

3. (Re)démarrer le serveur

4. Visiter

   http://localhost:8080/servlet/GetTime

Configurer Tomcat

- Ce n'est pas le déploiement habituel
- Mais c'est la façon simple et rapide
- **Un serveur de production ne devrait pas être configuré ainsi**
- Nous verrons plus tard une méthode de déploiement qui évite le redémarrage du serveur

Configurer Tomcat

- Éditer le fichier conf/context.xml
  - Remplacer <Context>
  - Par <Context privileged="true">
- Faites d’abord une copie sous le nom conf/context.xml.ori, ainsi vous pourrez facilement revenir à la configuration initiale

Configurer Tomcat

- Éditez conf/web.xml
  - Retirez les commentaires autour des lignes suivantes :

```
<servlet>
  <servlet-name>invoker</servlet-name>
  <servlet-class>
   org.apache.catalina.servlets.InvokerServlet
  </servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

Configurer Tomcat

- Éditez conf/web.xml
  - Retirez les commentaires autour des lignes suivantes :

```
<servlet-mapping>
   <servlet-name>invoker</servlet-name>
   <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

Configurer Tomcat

- Créer le répertoire :

  webapps/ROOT/WEB-INF/classes

Configurer Tomcat

- L'application tourne sur un seul serveur
- Il y a un seul filin d'exécution
- Nous devrons tout de même développer des solutions aux problèmes de synchronisation

Resources :

- Modèle build.xml :

  http://tomcat.apache.org/tomcat-6.0-doc/appdev/build.xml.txt

- Modèle web.xml :

  http://tomcat.apache.org/tomcat-6.0-doc/appdev/web.xml.txt

- Organisation du code source :

  http://tomcat.apache.org/tomcat-6.0-doc/appdev/source.html