Technologie Web

Les Servlets

Alexandre Pauchet

INSA Rouen - Département ASI

BO.B.RC.18, pauchet@insa-rouen.fr

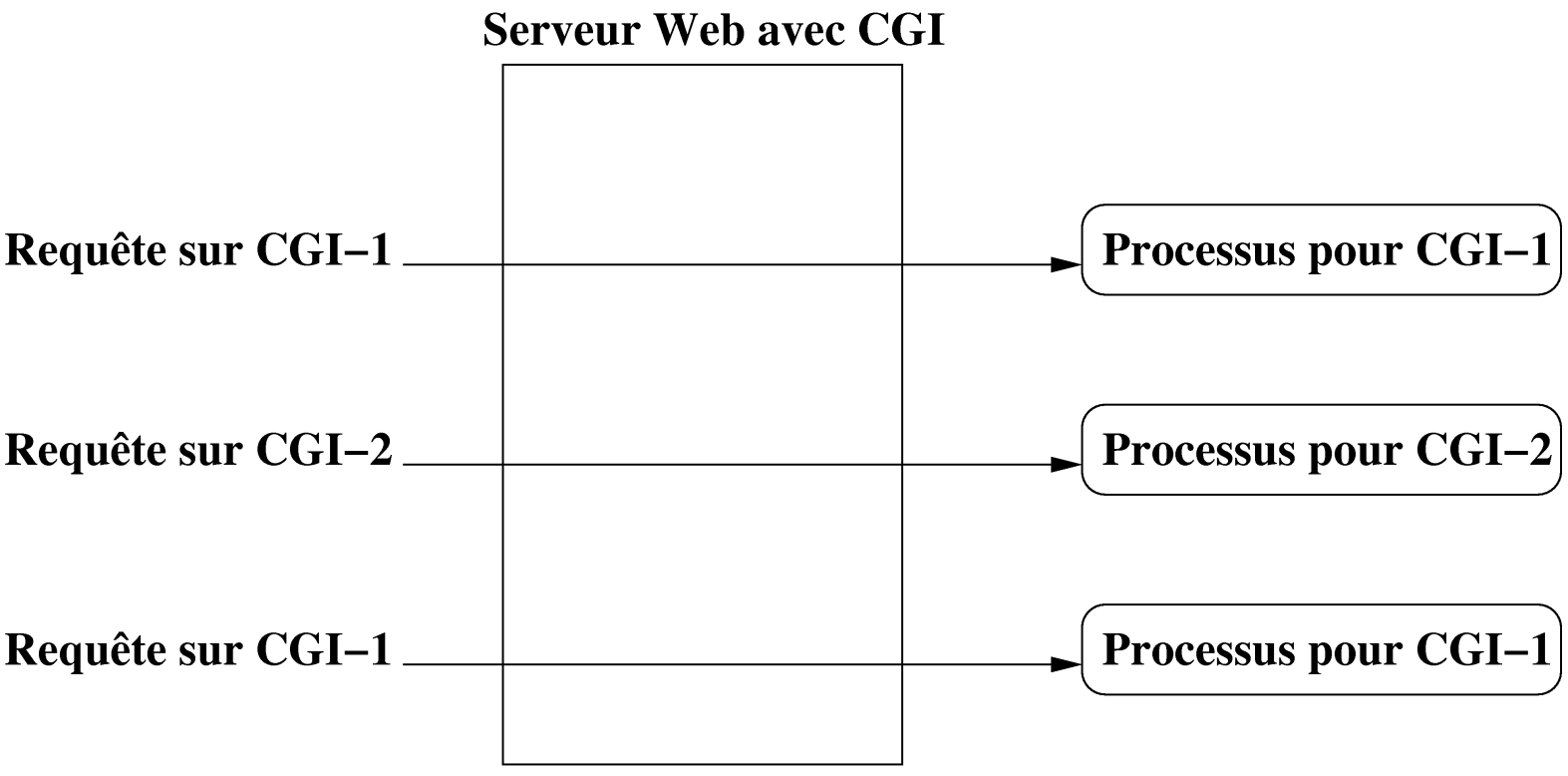
Plan

Introduction (1/10)
Description

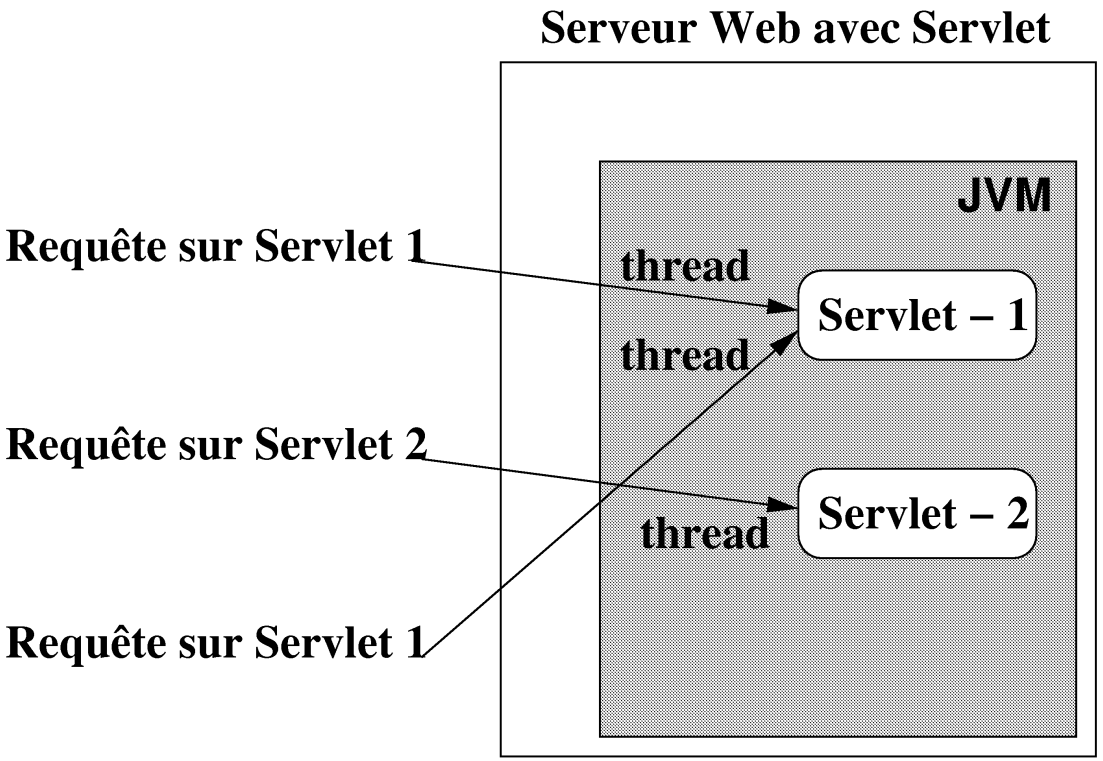
Que sont les Servlets ?

- Applications java
- Exécutées côté serveur
- Traitement de requêtes HTTP et génération de réponses dynamiques
  $\Rightarrow$ comparables aux scripts CGI et aux scripts PHP
- Permettent d'étendre les fonctionnalités de base d'un serveur HTTP
- Quelques avantages :
  - Portabilité (java)
  - Puissance avec l'accès à la totalité de l'API java
  - Rapidité (la Servlet est chargée une seule fois)

Introduction (2/10)
Fonctionnement des CGI



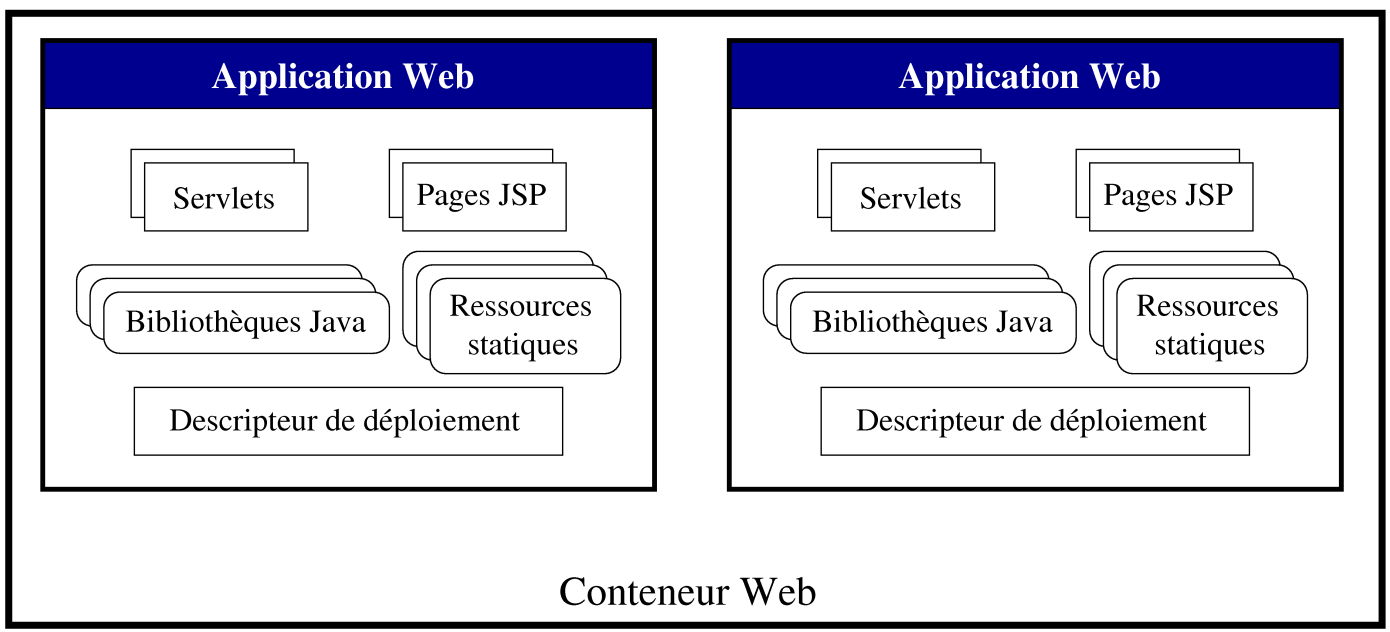
Introduction (3/10)
Fonctionnement des Servlets



Introduction (4/10)
Application Web JEE

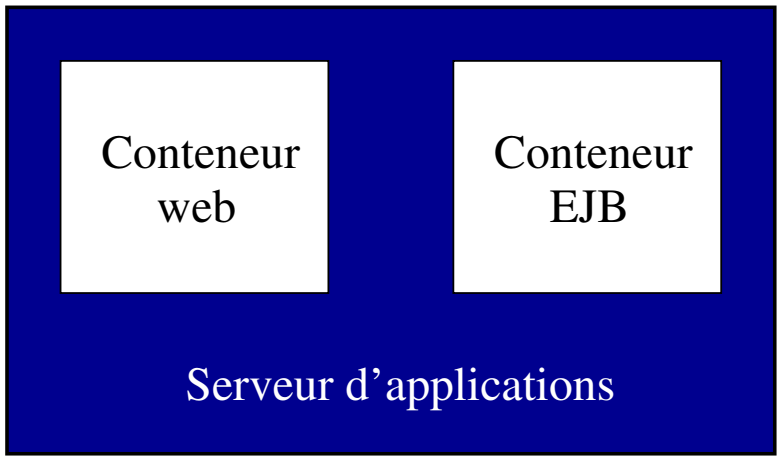
Une application web JEE est composée

- De Servlets/JSP
- De bibliothèques de classes annexes,
- De ressources statiques (images, HTML, XHTML, ...),
- Un descripteur de déploiement (fichier XML)



Introduction (5/10)
Serveur d'application JEE

- Une application JEE est composée d'une partie Web et d'un ensemble d'EJB



- Exemple de serveur web JEE : Tomcat
- Exemple de serveurs d'applications JEE : JBoss, WildFly

Introduction (6/10)
Vocabulaire JEE

JEE et les applications Web

- **Composant Web** : Servlets et JSP
- **Application Web** : ensemble de composants web, bibliothèques et ressources statiques dont l'organisation est décrite dans un descripteur de déploiement
- **Conteneur Web** :
  - Environnement d'exécution et de distribution des composants web et des ressources statiques
  - Gère le cycle de vie des Servlets (instanciation, initialisation, ...)

Introduction (7/10)
Exemple de Servlet : la Servlet

Hello.java

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

public class Hello extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
      ServletException, IOException {
    reponse.setContentType("text/plain");
    PrintWriter out = reponse.getWriter();
    out.println("Hello, hello !!!!");
  }
}
```

Introduction (8/10)
Exemple de Servlet : le fichier de déploiement

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>Bonjour</servlet-name>
    <servlet-class>Hello</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Bonjour</servlet-name>
    <url-pattern>/Servlet/Coucou</url-pattern>
  </servlet-mapping>
</web-app>
```

Introduction (9/10)
Exemple de Servlet : le fichier appelant

Appel sur JBOSS/WildFly :

hello.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première Servlet</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <meta http-equiv="REFRESH" content="0;http://localhost:8080/Hello/Servlet/Coucou">
  </head>
  <body>
  </body>
</html>
```

Introduction (10/10)
Exemple de Servlet : compilation et déploiement

compile.sh

```
#CLASSPATH=./src:$JBOSS_HOME/modules/system/layers/base/javax/servlet/api/main/jboss-
      servlet-api_3.1_spec-1.0.0.Final.jar
CLASSPATH=./src:$JBOSS_HOME/modules/system/layers/base/javax/servlet/api/main/jboss-
      servlet-api_4.0_spec-1.0.0.Final.jar

javac -cp $CLASSPATH -sourcepath src -d ./WEB-INF/classes src/*.java
jar cf Hello.war WEB-INF hello.html
cp Hello.war $JBOSS_HOME/standalone/deployments
```

- Dépôt de l'archive `Hello.war` dans `$JBOSS_HOME/standalone/deployments/`
- Appel : `http://localhost:8080/Hello/Servlet/Coucou`
- Ici la Servlet ne supporte que le GET

API (1/6)
Arborescence d'une archive WAR (Web ARchive)

```
ApplicationWeb.war
  |_ fichiers.html
  |_ fichiers.jsp
  |_ répertoires/fichiers
  |_ META-INF
  |    |_ MANIFEST.MF
  |_ WEB-INF
       |_ web.xml
       |_ classes
            |_ Servlets.class
            |_ lib
                 |_ bibliotheques.jar
```

API (2/6)
Compilation et déploiement

- Compilation
  La compilation de Servlets nécessite l'ajout de bibliothèques JAR dans le classpath ; pour JBOSS :
  `$JBOSS_HOME/modules/system/layers/base/javax/servlet/api/main/wildfly-14.0.1.Final.jar`

- Déploiement
  - Le répertoire `ApplicationWeb` est placé dans le répertoire d'applications du conteneur web
    - `webapps` pour Tomcat
    - `$JBOSS_HOME/standalone/deployments/` pour Jboss
  - Ou un fichier `ApplicationWeb.war` (fichier archive web) ayant pour contenu `ApplicationWeb` :

```
cd ApplicationWeb ; jar cf ApplicationWeb.war *
```

API (3/6)

Descripteur de déploiement : `web.xml`

Le fichier `web.xml` contient la description du déploiement. Il doit être valide par rapport à une DTD.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>Alias de la Servlet</Servlet-name>
    <servlet-class>Nom de la classe Servlet</Servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Alias de la Servlet</Servlet-name>
    <url-pattern>chemin (URL) déclenchant le chargement de la Servlet</url-pattern>
    <!--  le joker * est autorisé dans le chemin -->
  </servlet-mapping>
</web-app>
```

Attention à bien respecter la DTD (ordre des balises)

API (4/6)
Cycle de vie

Gestion du cycle de vie des Servlets :

- Le conteneur Web doit se conformer à la gestion des Servlets :
  - Création et initialisation de Servlets
  - Traitement des services demandés par les éventuels clients
  - Destruction des Servlets et libération de l'espace mémoire (ramasse-miettes)
- Java propose une API permettant de construire des Servlets :
  - L'API permet au développeur de gérer la création, les requêtes et la destruction des Servlets
  - Le conteneur fait appel à cette API pour gérer les Servlets

API (5/6)
API : classe GenericServlet

API JEE SDK

Nécessite le JEE SDK

Packages : `javax.Servlet.*` et `javax.Servlet.http.*`

La classe `GenericServlet` :

```
public abtract class GenericServlet {
  GenericServlet()
    // méthodes gérant l'initialisation et la destruction
  void  destroy()
  void  init()
  void  init(ServletConfig config)
    // méthodes pour la récupération des paramètres passés
  String        getInitParameter(String name)
  Enumeration   getInitParameterNames()
    // récupération d'informations concernant la Servlet
  ServletConfig          getServletConfig()
  ServletContext         getServletContext()
  String        getServletInfo()
  String        getServletName()
    // écriture de logs, pratique pour le débugage
  void  log(String msg)
  void  log(String message, Throwable t)
    // traite les requête (doit être redéfinie dans la sous-classe)
  abstract  void  service(ServletRequest req, ServletResponse res)
}
```

API (6/6)
API : classe HttpServlet

- Classe `HttpServlet`

```
public abtract class HttpServlet extends GenericServlet {
  HttpServlet()
    // méthodes répondant aux différents types de requêtes
  protected  void doDelete(HttpServletRequest req, HttpServletResponse resp)
  protected  void doGet(HttpServletRequest req, HttpServletResponse resp)
  protected  void doHead(HttpServletRequest req, HttpServletResponse resp)
  protected  void doOptions(HttpServletRequest req, HttpServletResponse resp)
  protected  void doPost(HttpServletRequest req, HttpServletResponse resp)
  protected  void doPut(HttpServletRequest req, HttpServletResponse resp)
  protected  void doTrace(HttpServletRequest req, HttpServletResponse resp)

    // permet d'utiliser le cache côté client
  protected  long getLastModified(HttpServletRequest req)

    // gérent le dispatching en fontion du type de requête}
  protected  void service(HttpServletRequest req, HttpServetResponse resp)
  protected  void service(ServletRequest req, ServletResponse res)
}
```

- Classes à utiliser pour créer ses propres Servlets
  - Dériver `HttpServlet` comme support pour votre Servlet
  - `HttpServletRequest` : toutes les données de la requête
  - `HttpServletResponse` : réponse à la requête

Encodage (1/4)
Génération de texte accentué

Requêtes HTTP transmises aux Servlets :

- `doGet(ServletRequest req, ServletResponse rep)`
- `doPost(ServletRequest req, ServletResponse rep)`
- ...

⇒ **Toutes les entrées et sorties peuvent être (re)traitées :**

- `ServletRequest` définit la méthode
  `void setCharacterEncoding(String env)`
- `ServletResponse` définit les méthodes
  - `void setContentType(java.lang.String type)`
  - `void setLocale(java.util.Locale loc)`

Encodage (2/4)
Exemple de génération de texte accentué

Encodage.java

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

public class Encodage extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    reponse.setContentType("text/plain; charset=UTF-8");
    reponse.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry
     ()));
    PrintWriter out = reponse.getWriter();
    out.println("Texte accentué : àéèùöüï");
  }
}
```

Remarque

- L'encodage de la sortie et celui de l'affichage doivent être cohérents.
- Par défaut, les navigateurs choisissent ISO-8859-1 ou ISO-8859-15.

$\Rightarrow$ **Il est conseillé de la fixer à UTF-8**

Encodage (3/4)

Traitement de requêtes contenant des accents

CodageGet.java

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

public class CodageGet extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    reponse.setContentType("text/plain; charset=UTF-8");
    reponse.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry
     ()));
    PrintWriter out = reponse.getWriter();

    String texte;
    requete.setCharacterEncoding("UTF-8");
    texte = requete.getParameter("texte");
    out.println("Texte après traitement : " + texte);
  }
}
```

Encodage (4/4)

Traitement de requêtes contenant des accents

codageGet.html

```
<!DOCTYPE>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Saisissez du texte avec accents</title>
  </head>
  <body>
    <h1>Bonjour</h1>
    <form action="/CodageGet/Servlet/CodageGet" method="get">
      <label>Texte avec accent : </label><input type="text" name="texte" size="30">
      <input type="submit" value="envoyer">
    </form>
  </body>
</html>
```

Remarque

- L'encodage de l'entrée et celui du fichier émetteur doivent être cohérents.
- **Il est conseillé de la fixer à UTF-8.**

Paramètres (1/15)

Paramètres provenant des formulaires : principe

- **Les paramètres de formulaires sont récupérables *via*** `HttpServletRequest`
- **La récupération se fait par les méthodes suivantes :**
  - `Enumeration getParameterNames()`
  - `String getParameter(String name)`
  - `String[] getParameterValues(String name)` :
    pour un champ de formulaire ayant plusieurs valeurs (liste à choix multiples, cases à cocher, *etc.*),

Paramètres (2/15)

Paramètres provenant des formulaires : exemple

helloPost.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ma première Servlet</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <h1>Bonjour</h1>
    <form action="/HelloPost/Servlet/Coucou" method="POST">
      <label>Prénom : </label><input type="text" name="prenom" size="30">
      <input type="submit" value="envoyer">
    </form>
    <hr/>
    <address><a href="mailto:Alexandre.Pauchet@insa-rouen.fr">Alexandre Pauchet</a></
     address>
  </body>
</html>
```

Paramètres (3/15)

Paramètres provenant des formulaires : exemple

HelloPost.java

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

public class HelloPost extends HttpServlet {
  public void doPost(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    reponse.setContentType("text/html; charset=UTF-8");
    reponse.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry
     ()));
    PrintWriter out = reponse.getWriter();
    requete.setCharacterEncoding("UTF-8");

    String prenom = requete.getParameter("prenom");

    out.println("<html>");
    out.println("<body>");
    out.println("<h1>Bonjour " + prenom + " ! </h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

Paramètres (4/15)
Passage en GET

helloGet.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Ma première Servlet</title>
  </head>
  <body>
    <h1>Bonjour</h1>
    <form action="/HelloGet/Servlet/HelloGet" method="GET">
      <label>Prenom : </label><input type="text" name="prenom" size="30">
      <input type="submit" value="envoyer">
    </form>
    <hr />
    <address><a href="mailto:Alexandre.Pauchet@insa-rouen.fr">Alexandre Pauchet</a></
     address>
  </body>
</html>
```

Paramètres (5/15)
Passage en GET

HelloGet.java

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

public class HelloGet extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
      ServletException, IOException {
    reponse.setContentType("text/html; charset=UTF-8");
    reponse.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry
      ()));
    PrintWriter out = reponse.getWriter();
    requete.setCharacterEncoding("UTF-8");

    String prenom = requete.getParameter("prenom");

    out.println("<html>");
    out.println("<body>");
    out.println("<h1>Bonjour " + prenom + " ! </h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

Paramètres (6/15)
Paramètres d'initialisation

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>ParamInit</servlet-name>
    <servlet-class>ParamInit</servlet-class>
    <init-param>
      <param-name>parametre1</param-name>
      <param-value>valeur1</param-value>
      <description>test de parametre</description>
    </init-param>
    <init-param>
      <param-name>parametre2</param-name>
      <param-value>valeur2</param-value>
      <description>test de parametre</description>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>ParamInit</servlet-name>
    <url-pattern>/Servlet/ParamInit</url-pattern>
  </servlet-mapping>
</web-app>
```

Paramètres (7/15)
Paramètres d'initialisation

ParamInit.java

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

public class ParamInit extends HttpServlet {
  String param1;
  String param2;

  public void init() {
    this.param1=getInitParameter("parametre1");
    this.param2=getInitParameter("parametre2");
  }

  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
      ServletException, IOException {
    reponse.setContentType("text/plain");
    PrintWriter out= reponse.getWriter();
    out.println("parametre1 = "+this.param1);
    out.println("parametre2 = "+this.param2);
  }
}
```

Paramètres (8/15)

Paramètres de contexte : principe

- **Association de paramètres à un ensemble de Servlets** : contrairement aux paramètres d'initialisation de Servlet précédents, ceux-ci sont commun à toutes les Servlets de l'application web.
- **Déclaration** : dans `web.xml`

```
<web-app>
  <context-param>
    <param-name>parametrecontext</param-name>
    <param-value>valeurcontext</param-value>
  </context-param>
  <!-- déclaration des Servlets -->
</web-app>
```

- **Récupération dans la Servlet** :

```
ServletContext context = getServletContext();
String chainecontext = context.getInitParameter("parametrecontext");
```

Paramètres (9/15)

Paramètres de contexte : exemple

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <context-param>
    <param-name>parametreDeContexte</param-name>
    <param-value>valeurContextuelle</param-value>
    <description>test de parametre</description>
  </context-param>
  <servlet>
    <servlet-name>ParamContext</servlet-name>
    <servlet-class>ParamContext</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ParamContext</servlet-name>
    <url-pattern>/Servlet/ParamContext</url-pattern>
  </servlet-mapping>
</web-app>
```

Paramètres (10/15)

Paramètres de contexte : exemple

ParamContext.java

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

public class ParamContext extends HttpServlet {
  String chainecontext;

  public void init() {
    ServletContext context = getServletContext();
    chainecontext = context.getInitParameter("parametreDeContexte");
  }

  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    reponse.setContentType("text/plain");
    PrintWriter out = reponse.getWriter();
    out.println("Parametre de contexte = "+chainecontext);
  }
}
```

Paramètres (11/15)
Attributs de contexte : principe

- **Partager des informations entre plusieurs Servlet** :
  - le partage est effectué à travers le contexte d'exécution géré par `ServletContext`
- **Obtention du contexte (ServletContext) d'une Servlet :**
  - `getServletContext().getContext(String uripath)`
- **Manipulation d'attributs du contexte par** `ServletContext` :
  - `void setAttribute(String name, Object o)`
  - `Object getAttribute(String name)`
  - `Enumeration getAttributeNames()`
  - `void removeAttribute(String name)`

Paramètres (12/15)
Attributs de contexte : exemple

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>Increment</servlet-name>
    <servlet-class>AttributContextIncrement</servlet-class>
  </servlet>

  <servlet>
    <servlet-name>Lecteur</servlet-name>
    <servlet-class>AttributContextLecteur</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Increment</servlet-name>
    <url-pattern>/Servlet/Increment</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Lecteur</servlet-name>
    <url-pattern>/Servlet/Lecteur</url-pattern>
  </servlet-mapping>
</web-app>
```

Paramètres (13/15)
Attributs de contexte : exemple

AttributContextLecteur.java

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

public class AttributContextLecteur extends HttpServlet {
  ServletContext context;

  public void init() {
    context = getServletContext().getContext("/AttributContext");
    if (context.getAttribute("compteur") == null)
      context.setAttribute("compteur","0");
  }

  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    reponse.setContentType("text/html");
    PrintWriter out= reponse.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("<h1> Lecteur valeur de compteur partage : "+context.getAttribute("
     compteur")+"</h1>");
    out.println("<a href='/AttributContext/Servlet/Increment'>incremente</a>");
    out.println("<a href='/AttributContext/Servlet/Lecteur'>lecteur</a>");
    out.println("</body>");
    out.println("</html>");
  }
}
```

Paramètres (14/15)
Attributs de contexte : exemple

AttributContextIncrement.java

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

public class AttributContextIncrement extends HttpServlet {
  ServletContext context;

  public void init() {
    context = getServletContext().getContext("/AttributContext");
    if (context.getAttribute("compteur") == null)
      context.setAttribute("compteur","0");
  }

  public int incremente() {
    try {
      int compteur = Integer.parseInt(""+context.getAttribute("compteur"))+1;
      context.setAttribute("compteur",""+compteur);
      return compteur;
    }
    catch (NumberFormatException exception) {
      log("TestAttributContext1 : probleme de conversion en int");
      log(""+exception);
    }
    return -1;
  }
```

Paramètres (15/15)
Attributs de contexte : exemple

AttributContextIncrement.java

```
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
      ServletException, IOException {
    reponse.setContentType("text/html");
    PrintWriter out= reponse.getWriter();

    out.println("<html>");
    out.println("<body>");
    out.println("<h1> Increment valeur de compteur partage : "+incremente()+"</h1>");
    out.println("<a href='/AttributContext/Servlet/Increment'>incremente</a>");
    out.println("<a href='/AttributContext/Servlet/Lecteur'>lecteur</a>");

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

Servlets et Threads (1/9)

Traitement des requêtes

Remarque

Le conteneur WEB ne contient qu'UNE instance de chaque Servlet

Les requêtes sont traitées dans des threads séparés :

- persistance de la Servlet et de son thread d'une requête à une autre,
- taille mémoire réduite,
- pas de surcoût pour traiter une nouvelle requête (déjà chargée),
- mémoire partagée (exemple pratique : la connexion à une BD est initialisée au chargement),
- il faut gérer les problèmes de synchronisation (utilisation de l'exclusion mutuelle).

Servlets et Threads (2/9)

Exemple : Compteur simple

CompteurSimple.java

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

public class CompteurSimple extends HttpServlet {
  private int compteur = 0;

  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    compteur++;
    out.println("Compteur : "+compteur);
  }
}
```

Servlets et Threads (3/9)

Exemple : Compteur simple

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>CompteurSimple</servlet-name>
    <servlet-class>CompteurSimple</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CompteurSimple</servlet-name>
    <url-pattern>/Servlet/CompteurSimple</url-pattern>
  </servlet-mapping>
</web-app>
```

Servlets et Threads (4/9)

Exemple : Compteur avec synchronisation

CompteurSynchro1.java

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

public class CompteurSynchro1 extends HttpServlet {
  private int compteur = 0;

  public synchronized void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    // pb la servlet ne traite qu'une requete a la fois

    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    compteur++;
    out.println("Compteur : "+compteur);
  }
}
```

Servlets et Threads (5/9)

Exemple : Compteur avec synchronisation

CompteurSynchro2.java

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

public class CompteurSynchro2 extends HttpServlet {
  private int compteur = 0;

  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    // seule la partie necessitant l'exclusion mutuelle est prise en compte

    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    int valCompteur;

    synchronized(this) {
      valCompteur = ++compteur;
    }
    out.println("Compteur : "+valCompteur);
  }
}
```

Servlets et Threads (6/9)

Exemple : Compteur d'instances de Servlet

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
          "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
          "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>
    <servlet-name>CompteurA</servlet-name>
    <servlet-class>CompteurInstances</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>CompteurB</servlet-name>
    <servlet-class>CompteurInstances</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CompteurA</servlet-name>
    <url-pattern>/Servlet/CompteurA</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>CompteurB</servlet-name>
    <url-pattern>/Servlet/CompteurB</url-pattern>
  </servlet-mapping>
</web-app>
```

Servlets et Threads (7/9)

Exemple : Compteur d'instances de Servlet

CompteurInstances.java

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

public class CompteurInstances extends HttpServlet {
  private static int nbInstances = 0;
  private static int compteurClasse = 0;
  private int compteur = 0;

  public void init() throws ServletException {
    nbInstances++;
  }

  public void destroy() {
    nbInstances--;
  }

  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    compteur++;
    compteurClasse++;
    out.println("Compteur pour cette instance : "+compteur);
    out.println("Nombre d'instances de la servlet : "+nbInstances);
    out.println("Compteur totale sur l'ensemble des instances de la servlet : "+
     compteurClasse);
  }
}
```

Servlets et Threads (8/9)

Exemple : Compteur persistant

CompteurPersistant.java

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

public class CompteurPersistant extends HttpServlet {
  private int compteur = 0;

  public synchronized void init() throws ServletException {
    try {
      FileReader file = new FileReader("/tmp/compteur.txt");
      BufferedReader reader = new BufferedReader(file);
      compteur = Integer.parseInt(reader.readLine());
      reader.close();
    }
    catch (FileNotFoundException exception) {}
    catch (IOException exception)           {}
    catch (NumberFormatException exception) {}
  }

  public void destroy() {
    super.destroy();
    enregistrerEtat();
  }
```

Servlets et Threads (9/9)

Exemple : Compteur persistant

CompteurPersistant.java

```
  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    compteur++;
    out.println("Compteur : "+compteur);
    if (compteur % 20 == 0)
      enregistrerEtat();
  }

  public synchronized void enregistrerEtat() {
    try {
      FileWriter file = new FileWriter("/tmp/compteur.txt");
      PrintWriter writer = new PrintWriter(file);
      writer.println(compteur);
      writer.close();
    }
    catch (IOException exception) {
      log("ErreurServlet : enregistrement de l'etat impossible pour CompteurPersistant"
       );
      log(""+exception);
    }
  }
}
```

Partage du contrôle (1/15)
Principe d'invocation

Appel d'une ressource à partir d'une Servlet

- Deux types d'invocations possibles :
  - la délégation
  - l'inclusion
- L'interface `ServletRequest` met à disposition un distributeur de requêtes par URL absolue ou relative :
  `RequestDispatcher getRequestDispatcher(String path)`

Remarque

La délégation à une autre Servlet doit se faire avec la méthode HTTP correspondante (doGet() ou doPost()) !

Partage du contrôle (2/15)
RequestDispatcher

Deux méthodes importantes

- `void forward(ServletRequest req, ServletResponse rep)` : délègue totalement à la ressource appellée, la réponse sera celle de la ressource
- `void include(ServletRequest req, ServletResponse rep)` : invoque et inclut la réponse de la ressource à la réponse de la servlet

Transmission des paramètres

- Il y a transmission des objets requête (`ServletRequest`) et réponse (`ServletResponse`)
- Il est possible d'ajouter de nouveaux attributs à la requête

Partage du contrôle (3/15)
Délégation

Utilité

Couramment utilisée pour déléguer l'affichage, le traitement étant fait dans la servlet appelante. Le résultat est transmis à une autre servlet *via* des attributs puis est affiché.

delegation.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Compteur de mots</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>

  <body>
    <h1>Saisissez une phrase</h1>
    <form action="/Delegation/Servlet/Analyse" method="GET">
      <input type="text" name="phrase" size="80">
      <input type="submit" value="Calcul">
    </form>
  </body>
</html>
```

Partage du contrôle (4/15)
Délégation

AnalyseTexte.java

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

public class AnalyseTexte extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {

    String phrase = requete.getParameter("phrase");
    String url;

    if (phrase != null && !phrase.trim().equals(""))
      url = "/Servlet/Affiche";
    else
      url = "/delegation.html";

    requete.setAttribute("resultat", ""+calculNbMotif(" ", phrase));
    RequestDispatcher rd = requete.getRequestDispatcher(url);
    rd.forward(requete,reponse);
  }
```

Partage du contrôle (5/15)
Délégation

AnalyseTexte.java

```
  int calculNbMotif(String motif, String phrase) {
    int nbMotifs = 0;
    int indice = 0;
    while ((indice = phrase.indexOf(motif, indice)) != -1) {
      nbMotifs++;
      indice++;
    }
    return nbMotifs;
  }
}
```

Partage du contrôle (6/15)
Délégation

Affiche.java

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

public class Affiche extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse)
    throws ServletException, IOException {
    reponse.setContentType("text/html; charset=UTF-8");
    PrintWriter out = reponse.getWriter();

    out.println("<html>");
    out.println("<head>");
    out.println("<title> Exemple de délégation </title>");
    out.println("<meta http-equiv='content-type' content='text/html;charset=utf-8' />")
      ;
    out.println("</head>");
    out.println("<body>");

    out.println("<p> <b>Nombres d'espaces : </b>"+requete.getAttribute("resultat")+"</p
      >");
    out.println("<p><b> Phrase : </b>\""+requete.getParameter("phrase")+"\"</p>");
    out.println("</body></html>");
  }
}
```

Partage du contrôle (7/15)
Inclusion

inclusion.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Recherche de caractères</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>

  <body>
    <form action="/Inclusion/Servlet/Affiche" method="POST">
      <h1>Saisissez une phrase</h1>
      <input type="text" name="phrase" size="80"/><br/>
      <h1>Saisissez plusieurs caractères à rechercher</h1>
      <input type="text" name="caracteres" size="10"/><br/>
      <input type="submit" value="Recherche"/>
    </form>
  </body>
</html>
```

Partage du contrôle (8/15)
Inclusion

Affiche.java

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

public class Affiche extends HttpServlet {
  public void doPost(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    requete.setCharacterEncoding("UTF-8");
    reponse.setContentType("text/html; charset=UTF-8");
    reponse.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry
     ()));
    String phrase = requete.getParameter("phrase");
    String caracteres = requete.getParameter("caracteres");
    if (phrase == null || phrase.trim().equals("")) {
       RequestDispatcher rd = requete.getRequestDispatcher("/inclusion.html");
       rd.forward(requete,reponse);
    }


...
```

Partage du contrôle (9/15)
Inclusion

Affiche.java

```
...
    else {
       PrintWriter out= reponse.getWriter();
       out.println("<html>");
       out.println("<head>");
       out.println("<title> Exemple d'inclusion </title>");
       out.println("<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'
     />");
       out.println("</head>");
       out.println("<body>");
       out.println("<p><b> Phrase : </b>\""+phrase+"\"</p>");
       for(int i=0; i<caracteres.length(); i++) {
           requete.setAttribute("lettre", ""+caracteres.charAt(i));
           RequestDispatcher rd = requete.getRequestDispatcher("/Servlet/Analyse");
           rd.include(requete,reponse);
       }
       out.println("</body>");
    }
  }
}
```

Partage du contrôle (10/15)
Inclusion

AnalyseTexte.java

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

public class AnalyseTexte extends HttpServlet {
  public void doPost(HttpServletRequest requete, HttpServletResponse reponse)
    throws ServletException, IOException {
    String phrase   = requete.getParameter("phrase");
    String lettre    = (String)requete.getAttribute("lettre");
    PrintWriter out = reponse.getWriter();
    out.println();
    out.println("<p><b> Nombre de '" + lettre + "' : " + calculNbMotif(lettre,phrase) +
      "</b></p>");
  }

  int calculNbMotif(String lettre, String phrase) {
    int nbLettres = 0;
    int indice = 0;
    while ((indice = phrase.indexOf(lettre, indice)) != -1) {
      nbLettres++;
      indice++;
    }
    return nbLettres;
  }
}
```

Partage du contrôle (11/15)
Inclusion/Délégation de JSP et ressources statiques

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Recherche de caractères</title>
  </head>
  <body>
    <form action="Servlet/Analyse" method="POST">
      <h1>Saisissez une phrase avec des espaces</h1>
      <input type="text" name="phrase" size="80"/><br/>
      <input type="submit" value="Recherche"/>
    </form>
  </body>
</html>
```

Partage du contrôle (12/15)
Inclusion/Délégation de JSP et ressources statiques

AnalyseTexte.java

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

public class AnalyseTexte extends HttpServlet {
  public void doPost(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    String phrase   = requete.getParameter("phrase");
    requete.setAttribute("espaces", ""+calculNbMotif(" ",phrase));
    requete.setAttribute("phrase", phrase);
    (requete.getRequestDispatcher("/index.jsp")).forward(requete,reponse);
  }

  int calculNbMotif(String motif, String phrase) {
    int nbMotifs = 0;
    int indice = 0;
    while ((indice = phrase.indexOf(motif, indice)) != -1) {
      nbMotifs++;
      indice++;
    }
    return nbMotifs;
  }
}
```

Partage du contrôle (13/15)
Inclusion/Délégation de JSP et ressources statiques

index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html>
<html>
  <body>
    <%
      String nbEspaces = (String)request.getAttribute("espaces");
      if(nbEspaces!=null) {
    %>
      <p>Nombre d'espaces : <%=nbEspaces %></p>
    <% } else { %>
      <p>Traitement impossible !</p>
    <% } %>
    <a href="/DelegationServlet/index.html">Retour au formulaire ?</a>
  </body>
</html>
```

Partage du contrôle (14/15)
Inclusion/Délégation de JSP et ressources statiques

- La redirection vers une page interne est possible

Information.java

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

public class Information extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    reponse.sendRedirect(requete.getContextPath()+"/index.jsp" );
  }

  public void doPost(HttpServletRequest requete, HttpServletResponse reponse) throws
     ServletException, IOException {
    PrintWriter out = reponse.getWriter();
    if(requete.getParameter("nom")!=null)
      out.println("Ceci est une information !");
    else
      reponse.sendRedirect(requete.getContextPath()+"/index.jsp" );
  }
}
```

Partage du contrôle (15/15)
Inclusion/Délégation de JSP et ressources statiques

index.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html>
<html>
  <body>
    <form action="Servlet/Information" method="POST">
      <label>Nom : </label><input type="text" name="nom" size="30">
      <input type="submit" value="envoyer">
    </form>
  </body>
</html>
```

Sessions (1/4)
API de suivi de session

Chaque session d'un client est gérée *via* un objet `HttpSession`

- **Récupération d'une session par `HttpServletRequest` :**
  - `HttpSession getSession()`
- **Quelques méthodes de `HttpSession` :**
  - `void setAttribute(String name, Object value)`
  - `Object getAttribute(String name)`
  - `void removeAttribute(String name)`
  - `void setMaxInactiveInterval(int secs)`
  - `void invalidate()`

NB : les informations relatives au client sont stockées côté serveur, mais son identifiant de session l'est dans un cookie, côté client.

Remarque

Ne confondez pas paramètres de requête, attributs de contexte et attributs de session !

Sessions (2/4)
Exemple : Compteur & Session

Timeout

```
<web-app>
  ...
  <session-config>
    <session-timeout>10</session-timeout>
    <!-- temps en minutes -->
  </session-config>
  ...
</web-app>
```

CompteurSession.java

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

public class CompteurSession extends HttpServlet {
  private int compteur = 0;

  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String message;
    HttpSession session = req.getSession();


...
```

Sessions (3/4)
Exemple : Compteur & Session

CompteurSession.java

```
...
    if (session.getAttribute("DejaVenu") == null) {
      session.setAttribute("DejaVenu","dejavenu");
      message="Bienvenu";
      compteur++;
    } else {
      message="Vous êtes déjà venu";
    }
    res.setContentType("text/html; charset=UTF-8");
    res.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry()));
    PrintWriter out = res.getWriter();
    out.println("<head>");
    out.println("<title> Compteur de Session </title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>"+compteur+"</h1>");
    out.println("<h1>"+message+"</h1>");
    out.println("<a href='CompteurKiller'>The Session killer</a>");
    out.println("</body>");
  }
}
```

Sessions (4/4)
Compteur & Session

CompteurSessionKiller.java

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

public class CompteurSessionKiller extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String message="Vous n'avez pas de session en cours";
    HttpSession session = req.getSession();
    if (session.getAttribute("DejaVenu") != null) {
      message="C'est la morte session :)";
      session.invalidate();
    }
    res.setContentType("text/html charset=UTF-8");
    res.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry()));
    PrintWriter out = res.getWriter();
    out.println("<head>");
    out.println("<title> Compteur de Session </title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>"+message+"</h1>");
    out.println("<a href='Compteur'>Le retour du compteur !!!</a>");
    out.println("</body>");
  }
}
```

Authentification (1/6)
Méthodes d'authentification

2 méthodes d'identification

- **Utilisation des mécanismes d'authentification du serveur**
  La gestion des comptes et mots de passe dépend du serveur
  - pour Tomcat gestion des comptes dans `conf/tomcat-users.xml`
  - pour JBoss/WildFly : utilisation du script `$JBOSS_HOME/bin/add-user.sh`
    1. Ajouter un "Application User" pour "ApplicationRealm"
    2. Préciser Login, MdP et la liste des groupes d'appartenance (rôles)
    3. Dans `web.xml`, définir la restriction au rôle pour le domaine de sécurité et préciser la méthode d'authentification
- **Authentification personnalisée gérée par des servlets**
  La gestion des comptes et mots de passe est laissée entièrement au développeur

Authentification (2/6)

Authentification par le serveur : Servlet secrète

InformationSecrete.java

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

public class InformationSecrete extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws
      ServletException, IOException {
    res.setContentType("text/html; charset=UTF-8");
    PrintWriter out = res.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Information Secrète</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Contrepèterie belge :</h1>");
    out.println("<p>\"Il fait beau et chaud.\"</p>");
    out.println("<p><a href='" + req.getContextPath() + "/Servlet/Logout'>Logout</a></p
      >");
    out.println("</body>");
  }
}
```

Authentification (3/6)

Authentification par le serveur : `web.xml`

WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "
     http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <servlet>
    <servlet-name>InformationSecrete</servlet-name>
    <servlet-class>InformationSecrete</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>SessionKiller</servlet-name>
    <servlet-class>SessionKiller</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>InformationSecrete</servlet-name>
    <url-pattern>/Servlet/InformationSecrete</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>SessionKiller</servlet-name>
    <url-pattern>/Servlet/Logout</url-pattern>
  </servlet-mapping>


...
```

Authentification (4/6)

Authentification par le serveur : `web.xml`

WEB-INF/web.xml

```
...
 <security-constraint>
  <web-resource-collection>
   <web-resource-name>Authentification</web-resource-name>
   <description>Une contrepèterie (belge) securisée</description>
   <url-pattern>/Servlet/InformationSecrete</url-pattern>
   <http-method>GET</http-method>
   <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
   <role-name>prof</role-name>
  </auth-constraint>
 </security-constraint>

 <login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
   <form-login-page>/login.jsp</form-login-page>
   <form-error-page>/erreur.jsp</form-error-page>
  </form-login-config>
 </login-config>

</web-app>
```

Authentification (5/6)

Authentification par le serveur : login et erreur

login.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
<!DOCTYPE html>
<html>
  <head>
    <title>Authentification par formulaire</title>
  </head>
  <body>
    <hr><form method="POST" action="j_security_check">
      Login : <input type="text" name="j_username" id="user"><br/>
      Pass : <input type="password" name="j_password"><br/>
      <input type="submit" value="login">
    </form><hr/>
  </body>
</html>
```

erreur.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
<!DOCTYPE html>
<html>
  <head><title>Authentification par formulaire</title></head>
  <body>
    <hr><h1>Erreur d'authentification</h1><hr>
  </body>
</html>
```

Authentification (6/6)
Authentification par le serveur : logout

SessionKiller.java

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

public class SessionKiller extends HttpServlet {
  public void doGet(HttpServletRequest requete, HttpServletResponse reponse)
    throws ServletException, IOException {
    requete.getSession().invalidate();
    reponse.sendRedirect(requete.getContextPath()+"/information.html" );
  }
}
```

Comptes créés

- bob/leponge : prof
- samantha/lo : etudiant

Filtrage (1/8)
Principes du filtrage

Principes du filtrage

- **Application de filtres :**
  - aux requêtes faites aux servlets
  - aux réponses de servlets
- **Objectif :** modifier leur entête et/ou leur contenu
- **Utilisation :**
  - authentification,
  - conversion (ex : formats d'image),
  - journalisation (ex : ajout d'une date),
  - transformations XML,
  - ...

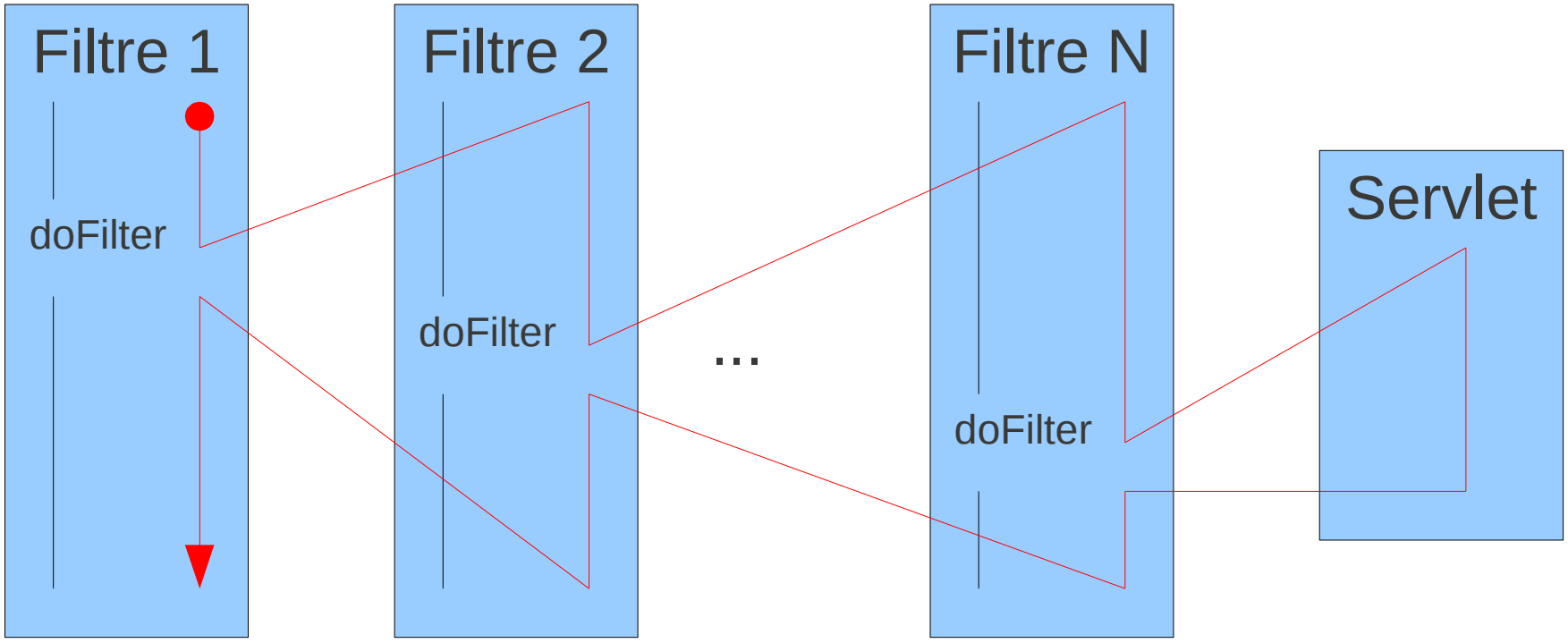
Filtrage (2/8)
Mise en œuvre

Implémentation de l'interface `Filter`

- Trois méthodes à redéfinir :
  - `void init(FilterConfig)` : permet de récupérer des informations sur le filtre mais aussi sur la servlet *via* le contexte
  - `void destroy()` : pour libérer des ressources à la destruction
  - `void doFilter(ServletRequest, ServletResponse, FilterChain)` :
    - examiner la requête et/ou la réponse (entête et corps)
    - modifier la requête et/ou la réponse (entête et corps)
    - appeler les autres filtres
- Les filtres sont appelés dans l'ordre de déclaration dans `web.xml`
- Le passage d'un filtre au suivant se fait par `.doFilter(...)`

Filtrage (3/8)
Parcours de filtres

- Les filtres sont appelés dans l'ordre de déclaration dans `web.xml`, par le filter-mapping
- Le passage d'un filtre au suivant se fait par un appel explicite à la méthode `((FilterChain)chain).doFilter(...)`



Filtrage (4/8)
Mapping d'URL

```
<filter-mapping>
  <filter-name>NomFiltre</filter-name>
  <url-pattern>URL à filtrer</url-pattern>
  ...
  <url-pattern>URL à filtrer</url-pattern>
</filter-mapping>
```

- L'ordre de déclaration dans `web.xml` des filter-mapping définit l'ordre d'appel des filtres
- Le mapping peut être :
  - Absolu (ex : /Servlet/Hello)
  - Relatif (ex : *.jsp ou /Servlet/*)
- Plusieurs motifs peuvent s'appliquer à un même filtre

Filtrage (5/8)
Exemple de filtre

Filtre.java

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

public class Filtre implements Filter {
  private SimpleDateFormat formatter = new SimpleDateFormat("E d MMM yyyy, H:m:s.S",
      Locale.FRANCE);
  private Date date = new Date();
  private FilterConfig fconfig;

  public void init(FilterConfig fconfig) {
    this.fconfig = fconfig;
  }

  public void destroy() {
    this.formatter = null;
    this.date      = null;
    this.fconfig   = null;
  }


...
```

Filtrage (6/8)
Exemple de filtre

Filtre.java

```
...
  public void doFilter(ServletRequest requete, ServletResponse reponse, FilterChain
      chain) throws ServletException, IOException {
    ServletContext context = this.fconfig.getServletContext();
    HttpServletRequest httpRequete = (HttpServletRequest)requete;
    PrintWriter writer = reponse.getWriter();

    this.date.setTime(System.currentTimeMillis());
    context.log("Debut: "+formatter.format(this.date)+" > "+httpRequete.getRequestURL()
     );
    reponse.setContentType("text/plain; charset=UTF-8");
    reponse.setLocale(new Locale(Locale.FRENCH.getLanguage(), Locale.FRANCE.getCountry
     ()));
    writer.println("En-tête ajouté par le filtre : " + this.fconfig.getInitParameter("
     EnTete"));
    chain.doFilter(requete, reponse);
    writer.println("Pied de page ajouté par le filtre : " + this.fconfig.
     getInitParameter("PiedDePage"));
    this.date.setTime(System.currentTimeMillis());
    context.log("Fin: "+formatter.format(this.date)+" > "+httpRequete.getRequestURL());
  }
}
```

Filtrage (7/8)
Exemple : déploiement

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "
     http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <filter>
     <filter-name>ServletFilter</filter-name>
     <filter-class>Filtre</filter-class>
     <init-param>
        <param-name>EnTete</param-name>
        <param-value>En-tête</param-value>
     </init-param>
     <init-param>
        <param-name>PiedDePage</param-name>
        <param-value>Pied de page</param-value>
     </init-param>
  </filter>
  <filter-mapping>
     <filter-name>ServletFilter</filter-name>
     <url-pattern>/Servlet/*</url-pattern>
  </filter-mapping>


...
```

Filtrage (8/8)
Exemple : déploiement

web.xml

```
...
  <servlet>
     <servlet-name>Hello</servlet-name>
     <servlet-class>Hello</servlet-class>
  </servlet>
  <servlet>
     <servlet-name>Compteur</servlet-name>
     <servlet-class>Compteur</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>Hello</servlet-name>
     <url-pattern>/Servlet/Hello</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
     <servlet-name>Hello</servlet-name>
     <url-pattern>/Hello</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
     <servlet-name>Compteur</servlet-name>
     <url-pattern>/Servlet/Compteur</url-pattern>
  </servlet-mapping>
</web-app>
```