

Servlet

Chapitre 2

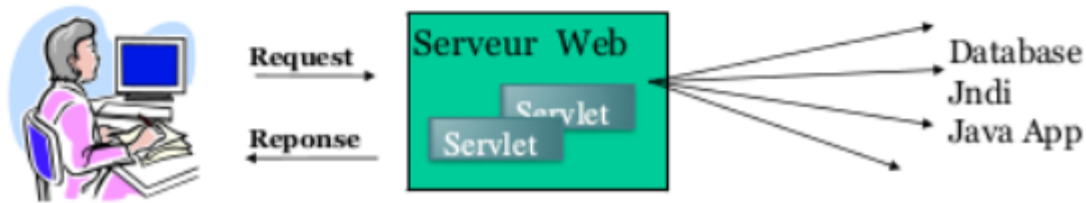
Introduction

Servlet : Server-side applet

- Permettre la programmation d'applications coté serveur
- Permettre l'extension d'un serveur Web en java
- Permettre la construction d'application Web dynamique
- Un serveur web héberge des classes Java Servlets qui sont exécutées à l'intérieur du [conteneur web](#).
- Une servlet est une classe implémentant l'interface [javax.servlet.Servlet](#). Elle fonctionne par l'intermédiaire du moteur de servlets, en réceptionnant les requêtes émises par les clients web, et en y répondant.
- Deux classes standards implémentant Servlet :
 - [GenericServlet](#) : une servlet indépendante de tout protocole
 - [HttpServlet](#) : une servlet spécifique au protocole HTTP

Fonctionnement d'une Servlet

- Une servlet lit les données envoyées par un client Web (transmises par le serveur)
 - Données explicites (Formulaire)
 - Données implicites (Request Header)
- Il génère un résultat
- Il envoie le résultat au client
 - Données explicites (Page HTML)
 - Données implicites (Response Header, Status code)



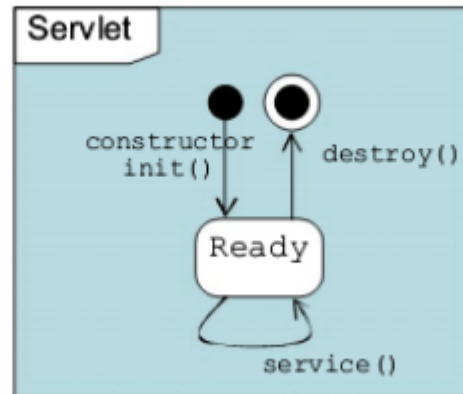
Servlet: Avantages coté serveur

- S'exécutent dans un moteur de Servlet (ou conteneur de Servlet) utilisé pour établir le lien entre la Servlet et le serveur web.
 - On ne se soucie plus de détails techniques tels que la connexion au réseau, la mise en forme de la réponse à la norme HTTP, ...,
- Une servlet, peut utiliser toutes les API Java afin de communiquer avec des applications extérieures, se connecter à des bases de données, accéder aux entrée- sorties (fichiers par exemple), ...

Servlet: Cycle de vie

Toutes les servlets ont le même cycle de vie

- Le conteneur charge la Servlet
 - Appel à la fonction : `init()`
- La Servlet répond aux requêtes des clients
 - Recevoir et répondre aux requêtes des clients par la fonction `service()`
- Le conteneur détruit la Servlet
 - La fonction `destroy()` est appelée (fermeture des connections)



Servlet: Cycle de vie

- Une servlet est un objet qui peut être manipulé par le conteneur via l'interface suivante :



- Lorsque le conteneur reçoit une requête, il la transmet à la servlet qui correspond à l'URL pour que la requête soit traitée effectivement

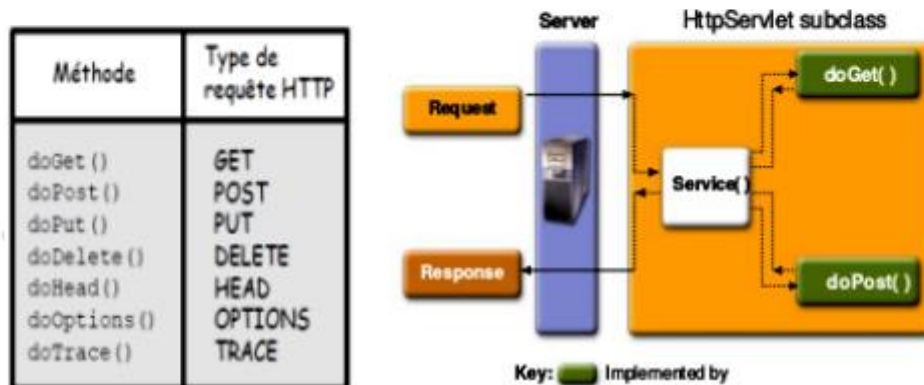
Servlet: Cycle de vie

Les servlets suivent un modèle de programmation
requête-service-response

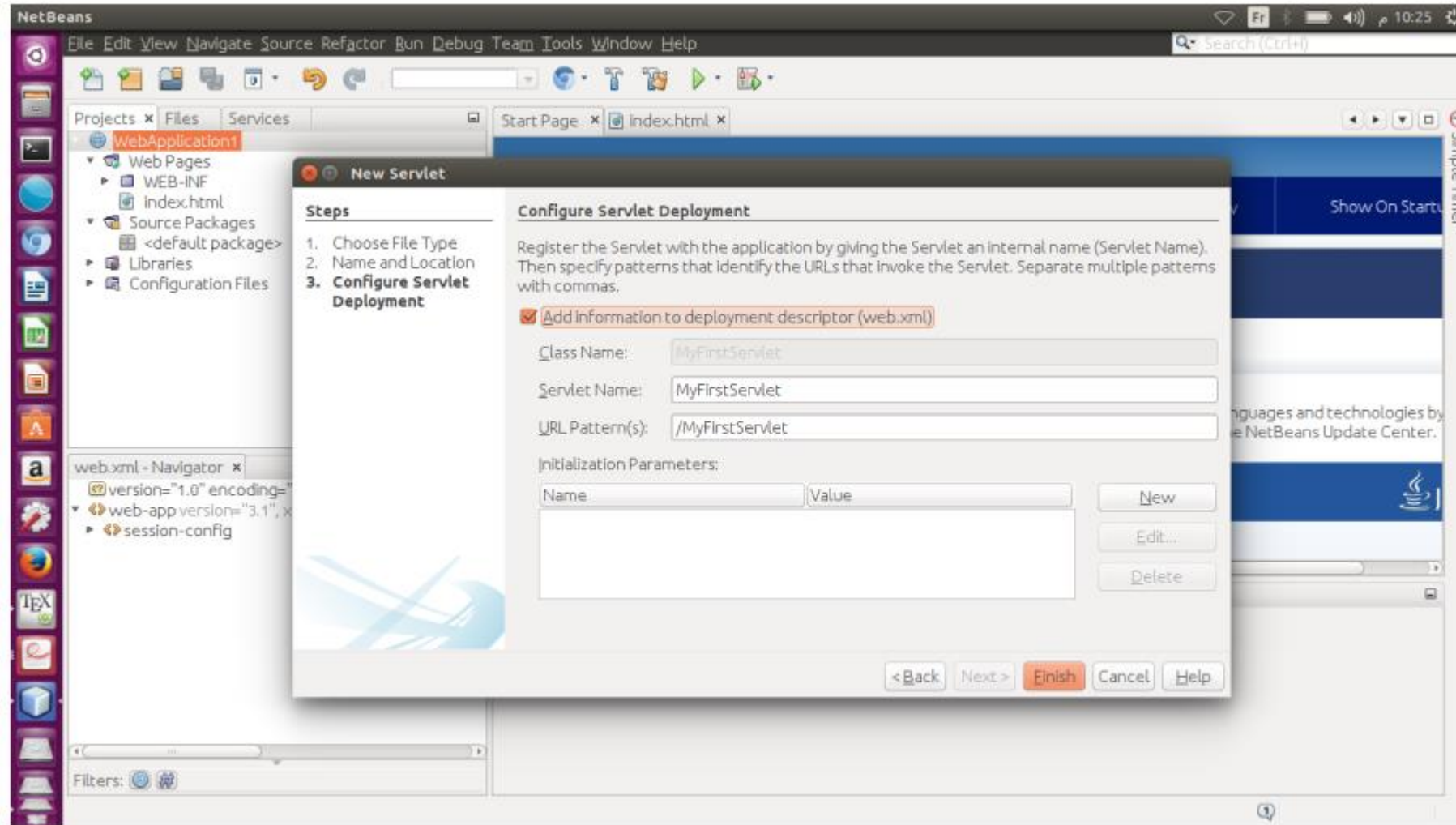
- Requête :
 - Objet : `javax.servlet.HttpServletRequest`
 - contient les informations nécessaires pour une communication du client vers le serveur
- Service : méthode `service()` est invoquée
- Réponse :
 - Objet : `javax.servlet.ServletResponse`
 - contient les informations nécessaires pour une communication du serveur vers le client

Modèle de programmation web

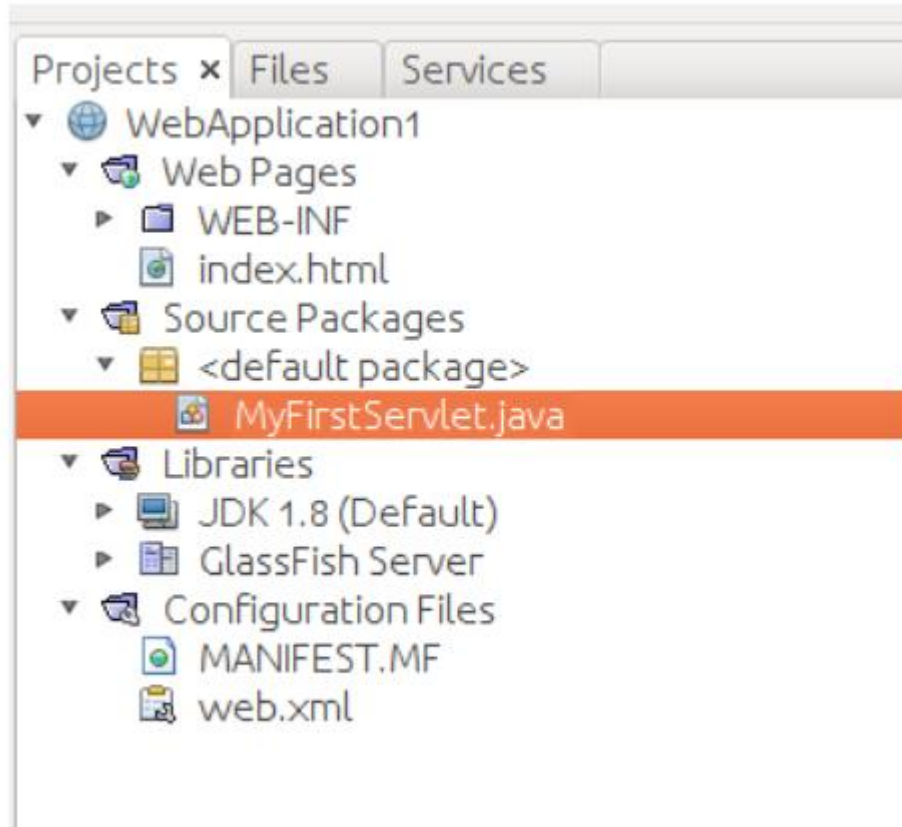
- Une Servlet Web étend la classe **javax.servlet.http.HttpServlet** (elle implémente **javax.servlet.Servlet**)
- Plusieurs méthodes spécifiques au protocole HTTP remplacent la méthode `service()`, qui appelle la méthode correspondant au type de requête.
- Une Servlet Web doit au moins redéfinir une de ces méthodes.



Servlet: Création de servlet



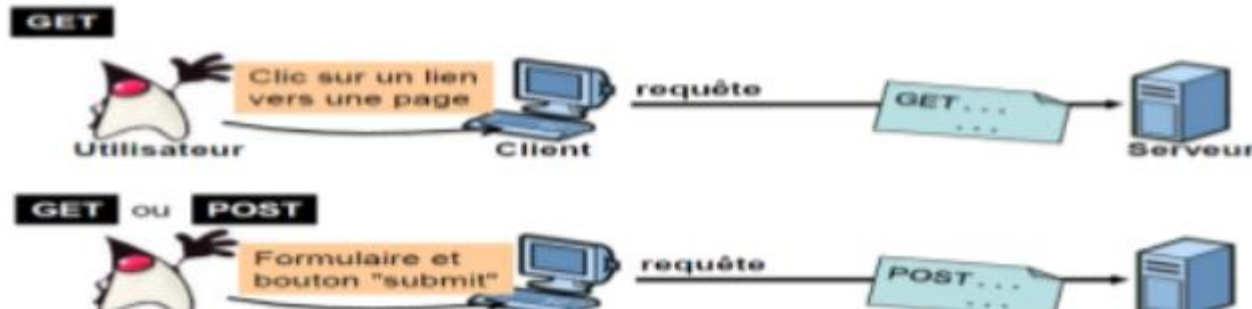
Servlet: structure de projet



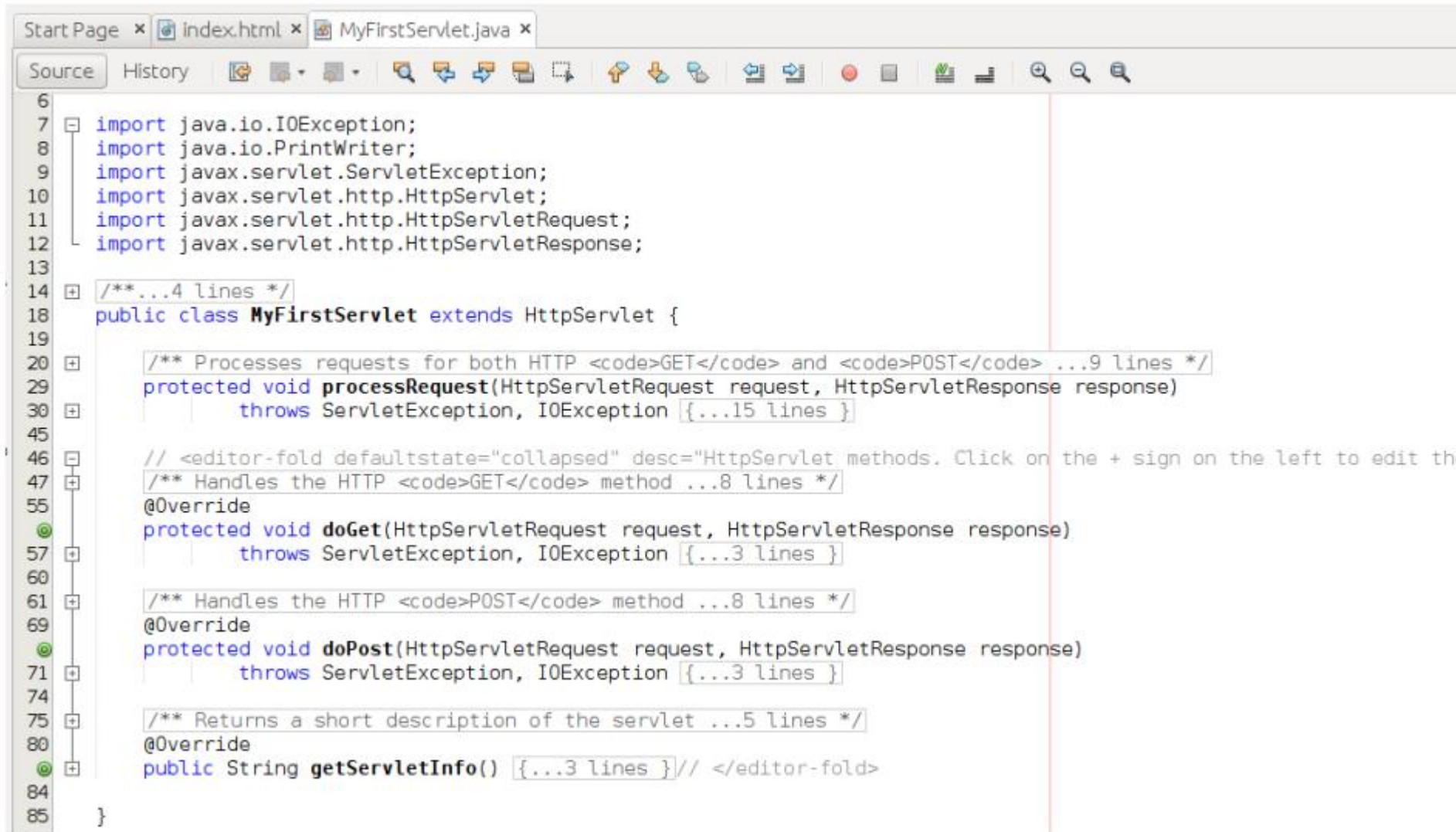
La classe HttpServlet

Utiliser les objets `HttpServletRequest` et `HttpServletResponse` passés en paramètres des méthodes `doGet()` ou `doPost()` pour implémenter le service :

- `HttpServletRequest` contient les renseignements sur le formulaire HTML initial (utile pour `doPost()`)
- `HttpServletResponse` contient le flux de sortie pour la génération de la page HTML résultat
 - ce flux de sortie est obtenu par les méthodes :
 - `getWriter()` : recommandé pour retourner du texte
 - `getOutputStream()` : recommandé pour des données binaires



Exemple d'une Servlet



```
Start Page x Index.html x MyFirstServlet.java x
Source History
6
7 import java.io.IOException;
8 import java.io.PrintWriter;
9 import javax.servlet.ServletException;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 /**...4 lines */
18 public class MyFirstServlet extends HttpServlet {
19
20     /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
29     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
30         throws ServletException, IOException {...15 lines }
45
46     // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the
47     /** Handles the HTTP <code>GET</code> method ...8 lines */
55     @Override
56     protected void doGet(HttpServletRequest request, HttpServletResponse response)
57         throws ServletException, IOException {...3 lines }
60
61     /** Handles the HTTP <code>POST</code> method ...8 lines */
69     @Override
70     protected void doPost(HttpServletRequest request, HttpServletResponse response)
71         throws ServletException, IOException {...3 lines }
74
75     /** Returns a short description of the servlet ...5 lines */
80     @Override
81     public String getServletInfo() {...3 lines }// </editor-fold>
84
85 }
```

Exemple d'une Servlet

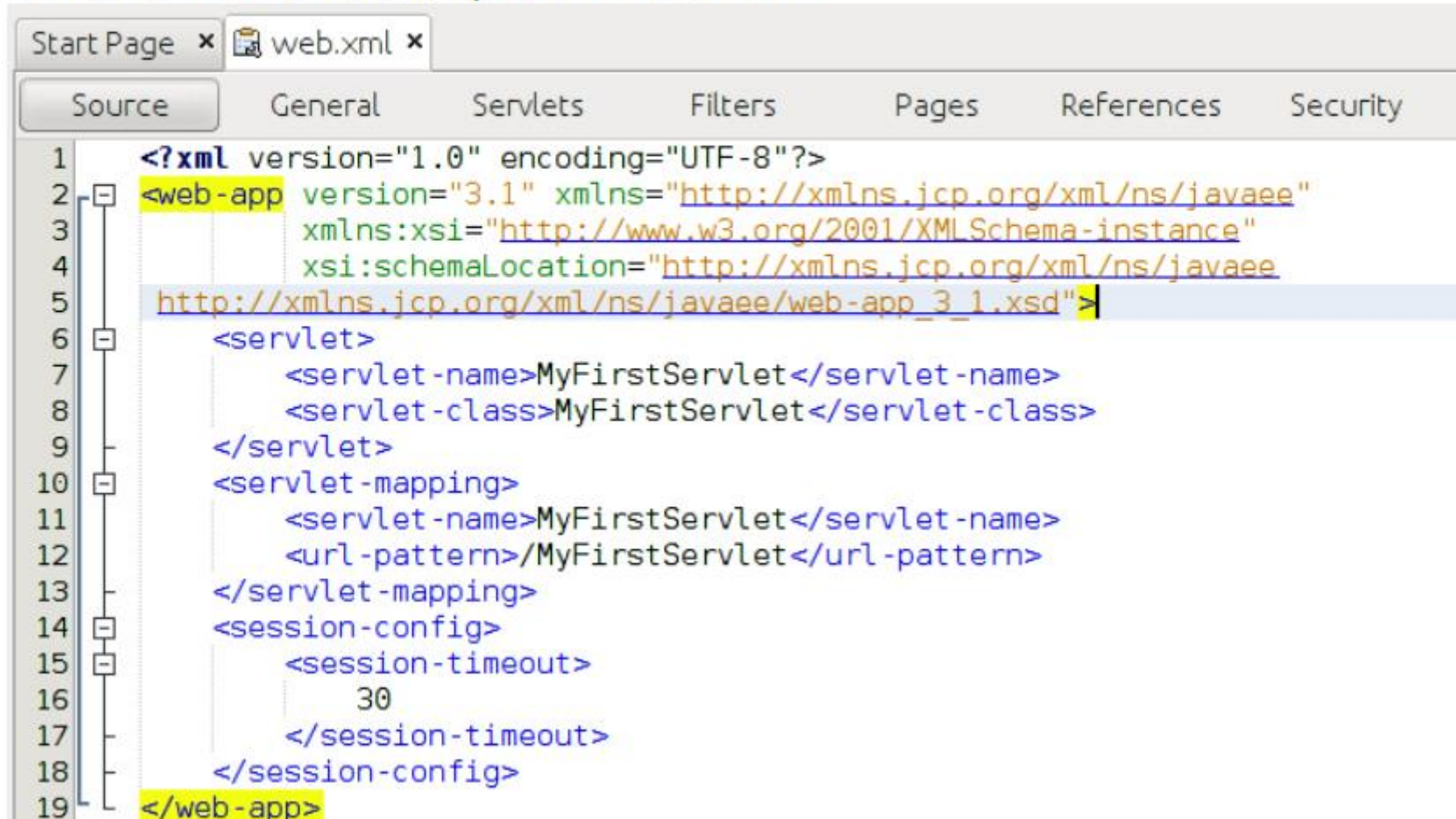
```
+ /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
- protected void processRequest(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-8");
  try (PrintWriter out = response.getWriter()) {
    /* TODO output your page here. You may use following sample code. */
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet MyFirstServlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Servlet MyFirstServlet at " + request.getContextPath() + "</h1>");
    out.println("</body>");
    out.println("</html>");
  }
}
```


Exemple d'une Servlet

- Lorsque la servlet est instanciée, il peut être intéressant d'effectuer des opérations qui seront utiles tout au long du cycle de vie de la servlet (se connecter à une base de données, ouvrir un fichier, ...). Pour ce faire, il s'agit de surcharger la méthode `init()` de la servlet. `public void init()`
- L'objet `HttpServletResponse` permet de renvoyer une page à l'utilisateur :
 - ① définir le type de données qui vont être envoyées au client : la méthode `setContentType()` de l'objet `HttpServletResponse`
 - ② envoyer du texte formaté au navigateur `PrintWriter out = res.getWriter()`
 - ③ utiliser la méthode `println()` de l'objet `PrintWriter` afin d'envoyer les données textuelles au navigateur,
 - ④ fermer l'objet `PrintWriter` lorsqu'il n'est plus utile avec sa méthode `close()`

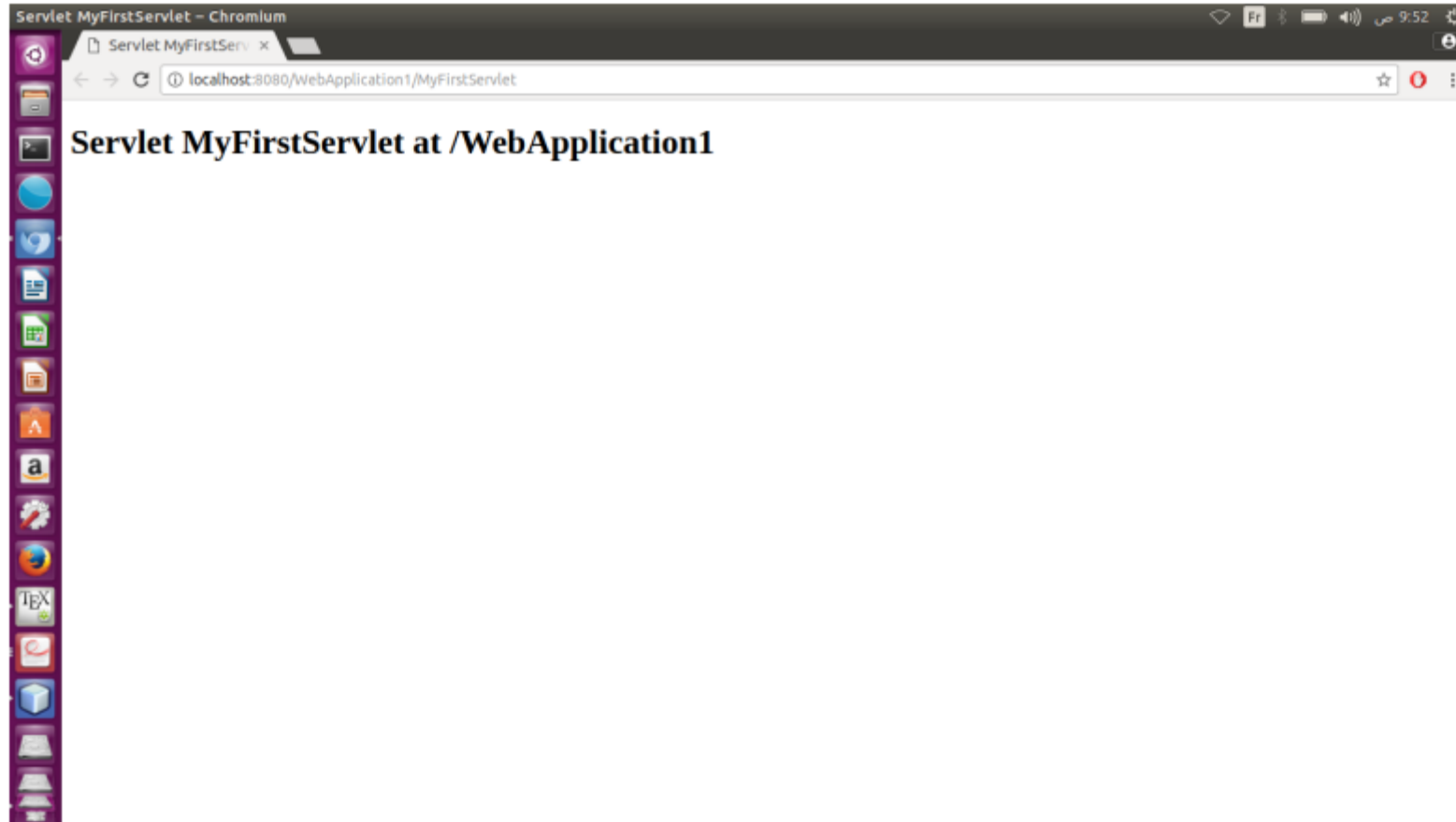
Le fichier descripteur: web.xml

Déclaration d'une Servlet au sein d'une application Web se fait dans le fichier descripteur Web.xml



```
Start Page x web.xml x
Source General Servlets Filters Pages References Security
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5     http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
6     <servlet>
7         <servlet-name>MyFirstServlet</servlet-name>
8         <servlet-class>MyFirstServlet</servlet-class>
9     </servlet>
10    <servlet-mapping>
11        <servlet-name>MyFirstServlet</servlet-name>
12        <url-pattern>/MyFirstServlet</url-pattern>
13    </servlet-mapping>
14    <session-config>
15        <session-timeout>
16            30
17        </session-timeout>
18    </session-config>
19 </web-app>
```

Exécution de la Servlet



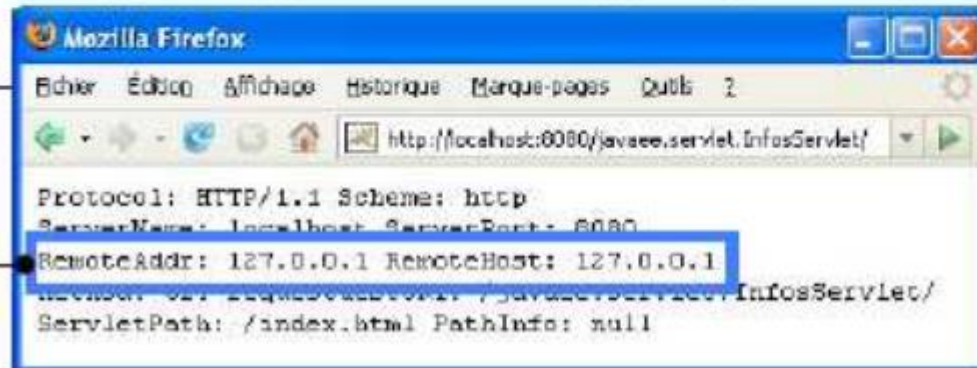
- **Une servlet** ne peut pas être déployé directement dans un conteneur, elle doit faire partie d'un module Web.
- **Un module Web** est un ensemble de librairies, de fichiers de configurations, de code Java (bytecode des servlets...), ...
- Pour déployer une application dans un conteneur, il faut lui fournir deux éléments :
 - L'application avec tous les composants (classes compilées, ressources ...) regroupée dans une archive ou module. Chaque conteneur possède son propre format d'archive.
 - Un fichier descripteur de déploiement contenu dans le module qui précise au conteneur des options pour exécuter l'application

L'interface HttpServletRequest

- Fournit les informations sur la requête du client au serveur :
 - `String getParameter(String name)`
 - Permet de retourner la valeur d'un champ dont on a passé le nom.
 - `public String[] getParameterValues(String Key)`
 - à utiliser Lorsque'un champ d'un formulaire peut avoir plusieurs valeurs (liste à choix multiples, cases à cocher,...)
 - `Enumeration getParameterNames()`
 - Pour connaître l'ensemble des noms des champs du formulaire passé à la servlet.
 - `String getHeader(String name)`
 - `Enumeration getHeaderNames()`
 - Retourne une énumération de tous les noms des propriétés du header
 - `String[] getHeaderValues()`
 - Retourne un tableau de toutes les valeurs du header

L'interface HttpServletRequest

```
public class InfosServlet extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        response.setContentType("text/plain");  
        PrintWriter out= response.getWriter();  
        out.println("Protocol: " + request.getProtocol());  
        out.println("Scheme: " + request.getScheme());  
        out.println("ServerName: " + request.getServerName());  
        out.println("ServerPort: " + request.getServerPort());  
        out.println("RemoteAddr: " + request.getRemoteAddr());  
        out.println("Method: " + request.getMethod());  
    }  
}
```

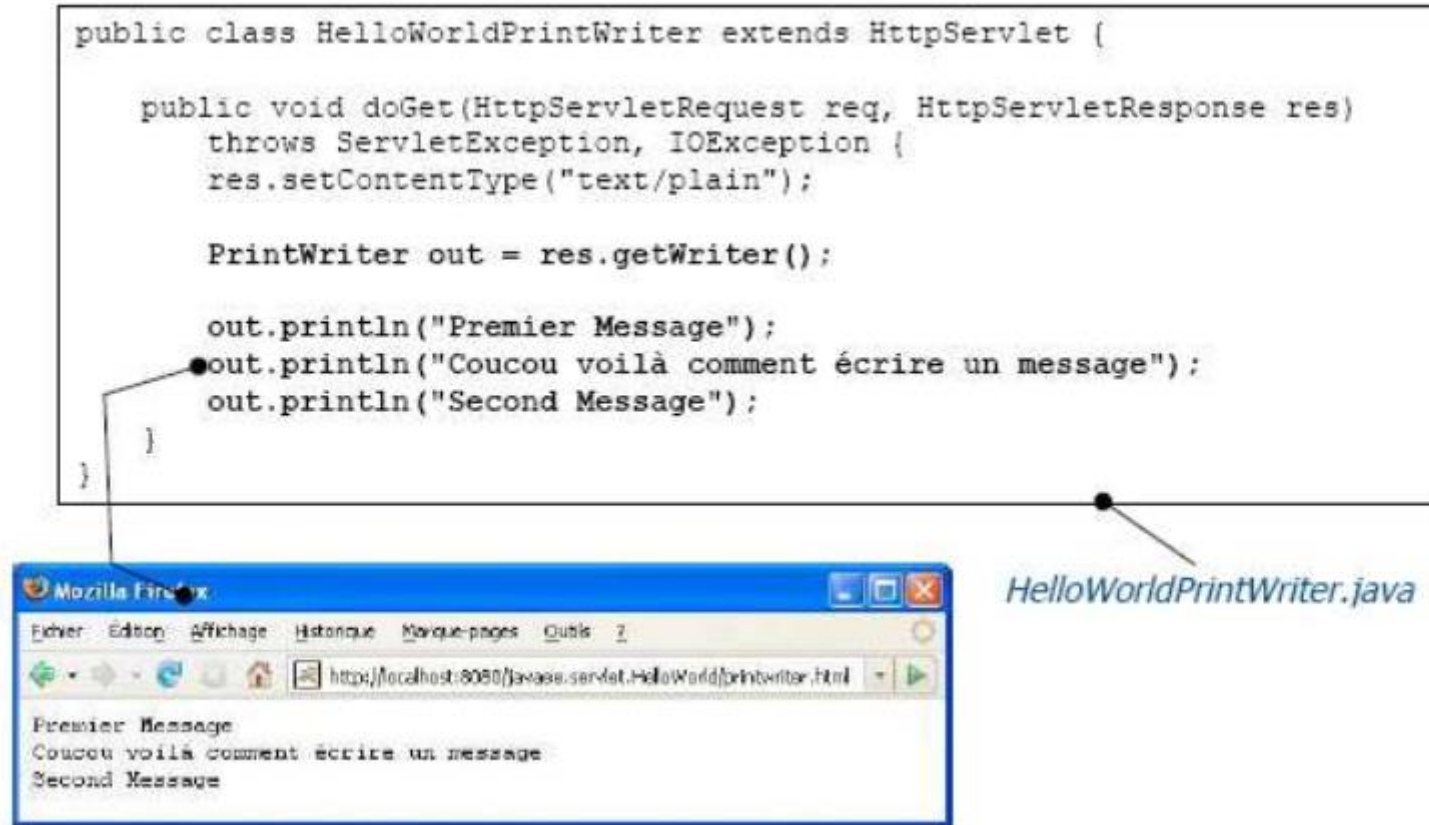


InfosServlet.java

L'interface HttpServletResponse

- Objet fournissant les services pour envoyer la réponse à un client
 - `java.io.PrintWriter getWriter()`
 - Pour récupérer un `printWriter` qui permet d'envoyer du texte au client
 - `public void setContentLength(int len)`
 - Positionne le Content-Length Header
 - `public void setContentType(java.lang.String type)`
 - Position le Content-Type header (exemple text/html)
 - `public void sendError(int sc, java.lang.String msg) throws java.io.IOException`
 - Envoi un message d'erreur au client (cf code dans l'API)
 - `public void setHeader(java.lang.String name, java.lang.String value)`
 - Ajoute une propriété à la requête

L'interface HttpServletResponse



Le RequestDispatcher

- L'objet RequestDispatcher est utilisé pour
 - Transférer la requête à un autre programme (jsp, servlet)
 - Inclure la réponse d'un autre programme

- Pour obtenir le request dispatcher

```
RequestDispatcher rd=getServletContext().getRequestDispatcher("/index.jsp");  
rd.forward(request, response);
```

- La requête peut être transférée
 - rd.forward(request,response)
 - L'utilisation classique consiste à avoir un servlet contrôleur qui transmet les commandes à des servlets spécialisés
 - Le contrôle ne revient jamais au servlet
- La réponse peut être incluse dans la réponse en cours de construction
 - rd.include(request,response)

Exemple d'utilisation de RequestDispatcher

```
RequestDispatcher rd;  
String path=request.getPathInfo();  
if (path.equals("/test")) {  
    rd=getServletContext().getRequestDispatcher("/index.jsp");  
} else {  
    rd=getServletContext().getRequestDispatcher("/error.jsp");  
}  
rd.forward(request, response);  
out.println("</body>");  
out.println("</html>");
```

Conclusion

- Servlets : étendent le comportement des serveurs Web avec des programme Java
 - Portabilité, facilité d'écriture (Java)
 - Définition du code, du packaging, du déploiement
 - Persistance des données dans les servlets
 - Exécutée en // avec des processus légers (threads)
- Mais :
 - Difficile d'écrire du code HTML dans du code Java
 - ⇒ Introduction de la technologie Java Server Pages (JSP)
 - Pas de mécanisme intégré de distribution
 - ⇒ Introduction de la technologie Enterprise Java Beans(EJB)