

# Le développement Web côté serveur – BackEnd avec Java EE

## Module 2 – Les servlets



1

Les servlets

## Objectifs

- Comprendre le rôle d'une servlet
- Comprendre le cycle de vie d'une servlet
- Savoir exploiter une requête HTTP
- Savoir générer une réponse HTTP

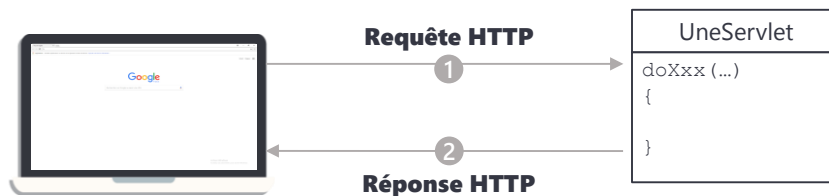


2

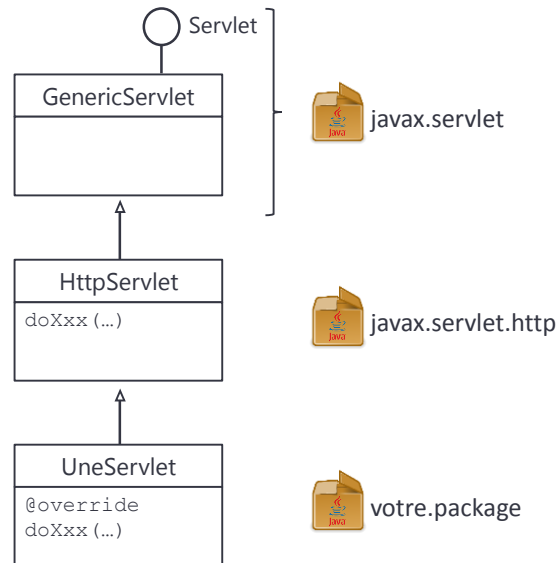
# Java Servlet 3.1



## Qu'est-ce que c'est ?



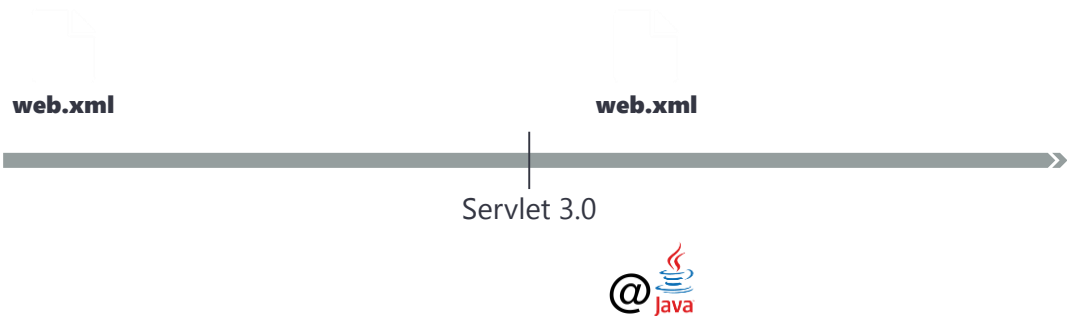
## Modèle objet



## La création d'une servlet

```
public class UneServlet extends HttpServlet
{
    @Override
    protected void doGet(...) throws ServletException, IOException
    {
        //Générer la réponse à une requête de type GET
    }
    @Override
    protected void doPost(...) throws ServletException, IOException
    {
        //Générer la réponse à une requête de type POST
    }
}
```

## Le paramétrage de l'URL



## Le paramétrage dans le web.xml

```
<!-- Déclaration d'une servlet -->
<servlet>
    <servlet-name>UneServlet</servlet-name>
    <servlet-class>fr.eni.demo.servlets.UnesServlet</servlet-class>
</servlet>

<!-- Association de la servlet à une ou des URL -->
<servlet-mapping>
    <servlet-name>UneServlet</servlet-name>
    <url-pattern>/url/de/la/servlet</url-pattern>*
    <url-pattern>...</url-pattern>
</servlet-mapping>
```



\* utilisation du caractère joker

```
<url-pattern>/debut/url/*</url-pattern>
```



## Le paramétrage par annotation

```
@WebServlet
(
    name="UneServlet",
    urlPatterns={"/url/de/la/servlet","..."}
)
public class UneServlet extends HttpServlet
{
    ...
}

@WebServlet("/debut/url/*")
public class UneServlet extends HttpServlet
{
    ...
}
```



## La première servlet

# Démonstration



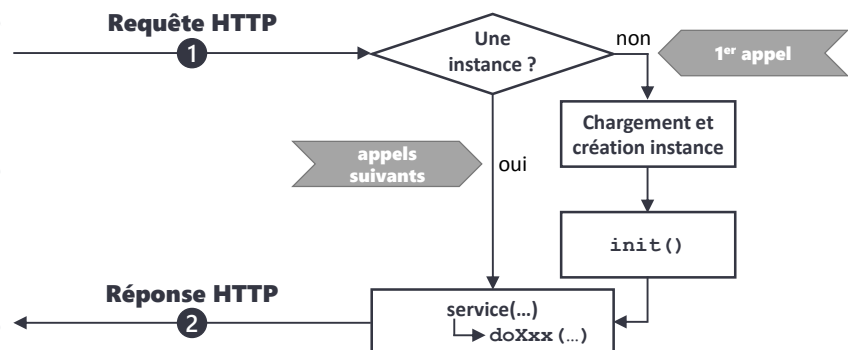
## Démonstration



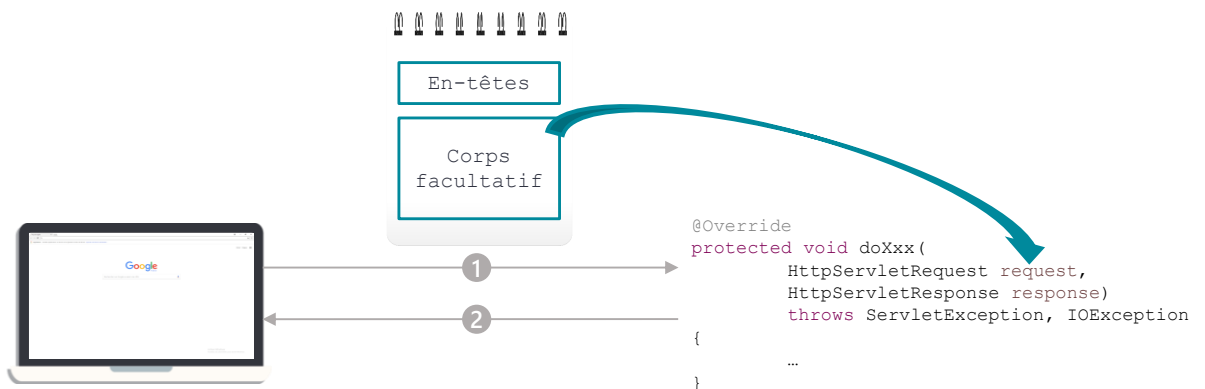
Gérée par le  
conteneur web

Une seule  
instance

Pool de  
threads



## Démonstration



## La lecture de la requête : l'URL

`http://www.exemples.fr:8080/DemoJavaEE/url/de/la/servlet`

HttpServletRequest <<interface>>
<code>getScheme():String</code> <code>getServerName():String</code> <code>getServerPort():int</code> <code>getContextPath():String</code> <code>getServletPath():String</code>



## La lecture de la requête : les en-têtes principaux



```
POST /docs/ouvrage HTTP/1.1
Host: www.exemples.fr
Accept-Language: en-US
...
```

```
nom=Java%20EE&auteur=ENI%20
Ecole
```

HttpServletRequest <<interface>>
<code>getCharacterEncoding():String</code> <code>getContentLength():int</code> <code>getContentType():String</code> <code>getLocale():Locale</code> <code>getMethod():String</code> ...





## La lecture de la requête : tous les en-têtes



```
POST /docs/ouvrage HTTP/1.1
Host: www.exemples.fr
Accept-Language: en-US
...
```

```
nom=Java%20EE&auteur=ENI%20
Ecole
```

### HttpServletRequest <<interface>>

```
getHeader(String name):String
getDateHeader(String name):Date
getIntHeader(String name):int
getHeaders(String name)
    :Enumeration<String>
getHeaderNames()
    :Enumeration<String>
...
```



## La lecture de la requête : les paramètres



```
POST /docs/ouvrage HTTP/1.1
Host: www.exemples.fr
Accept-Language: en-US
...
```

```
nom=Java%20EE&auteur=ENI%20
Ecole
```

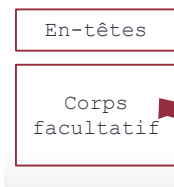
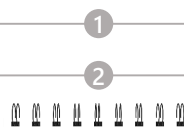
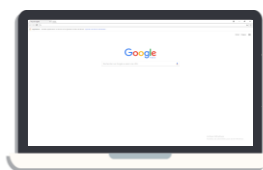
### HttpServletRequest <<interface>>

```
getParameter(String name)
    :String
getParameterValues(String name)
    :String[]
getParameterNames()
    :Enumeration<String>
getParameterMap()
    :Map<String, String[]>
```

*Quel que soit  
le type  
de la requête*



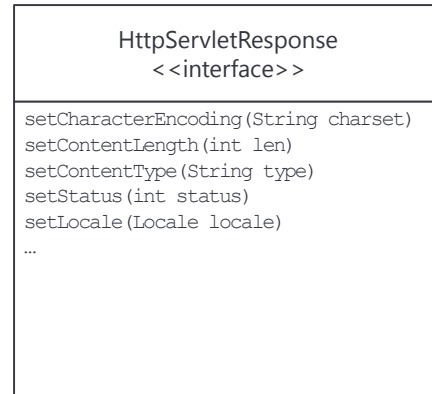
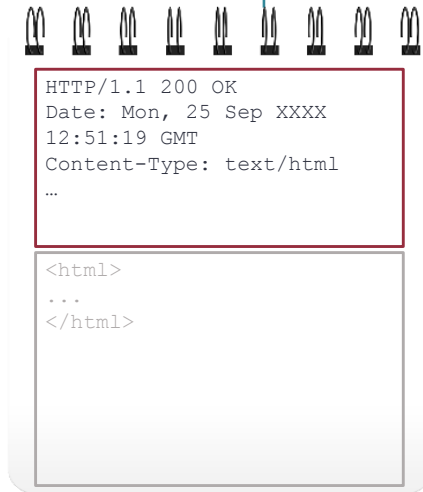
## Démonstration



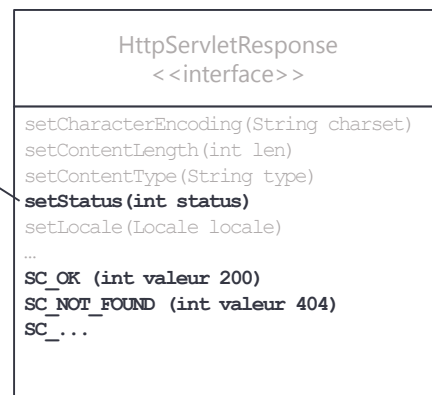
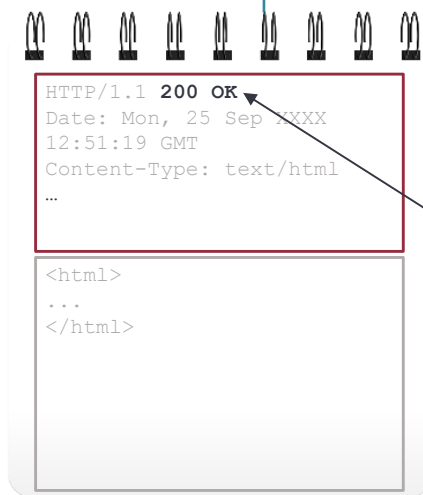
```
@Override
protected void doXXX(
    HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException
{
    ...
}
```



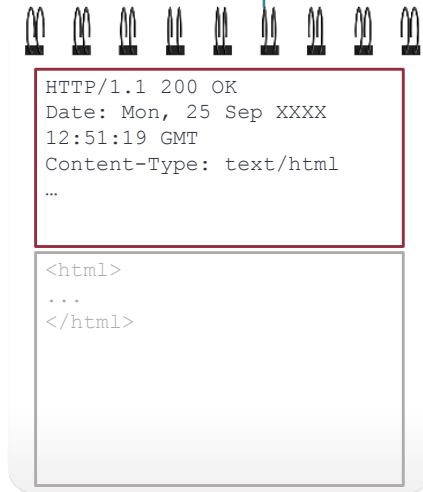
## L'écriture de la réponse : les en-têtes principaux



## L'écriture de la réponse : focus - le code de statut



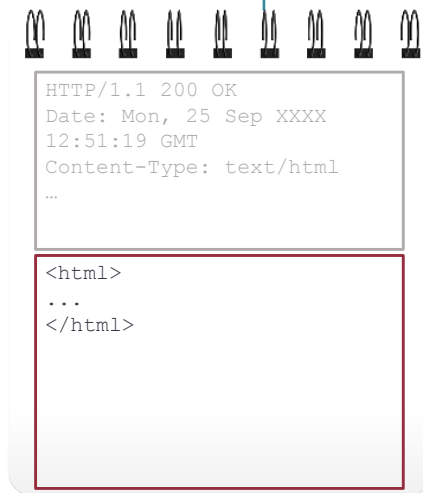
## L'écriture de la réponse : tous les en-têtes



HttpServletResponse <<interface>>
<code>setHeader(String name, String value)</code> <code>setDateHeader(String name, long date)</code> <code>setIntHeader(String name, int value)</code> <code>...</code>



## L'écriture de la réponse : le corps



HttpServletResponse <<interface>>
<code>getWriter() : PrintWriter</code>  <i>Flux texte</i>  <code>getOutputStream() : ServletOutputStream</code>  <i>Flux binaire</i>



## L'écriture de la réponse : le corps au format texte

### Utilisation

```
PrintWriter out = response.getWriter();  
out.println("Ecriture du corps de la réponse HTTP");  
//Suite écriture...  
out.close();
```

### Manipulation du tampon

```
//Taille du tampon  
response.setBufferSize(size);  
//Vidage du tampon  
response.reset();  
//Forcer l'envoi du tampon  
out.flush();
```



## L'écriture de la réponse

# Démonstration



## L'écriture de la réponse : la redirection permanente



**301**  
**Nous avons**  
**déménagé**

HttpServletResponse  
<<interface>>

```
setStatus(int status)  
setHeader(String name, String value)  
SC_MOVE_PERMANENTLY
```

```
response.setStatus(HttpServletResponse.SC_MOVED_PERMANENTLY);  
response.setHeader("Location", "Nouvelle URL");
```



## L'écriture de la réponse : la redirection temporaire



**302**  
**Nous revenons**  
**bientôt**

HttpServletResponse  
<<interface>>

```
setStatus(int status)  
setHeader(String name, String value)  
SC_MOVE_TEMPORARILY  
ou  
sendRedirect(String location)
```

```
response.sendRedirect("URL Temporaire");
```



## L'écriture de la réponse : répondre une erreur

HttpServletResponse <<interface>>
<pre>sendError(int status) sendError(int status, String message)</pre>

### web.xml

```
<error-page>
  <error-code>500</error-code>
  <location>/erreur500.html</location>
</error-page>
```

```
response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
```



## Rediriger la réponse

# Démonstration




## Rechercher le nombre tiré au sort

TP





## L'écriture de la réponse : `IllegalStateException`

1<sup>er</sup> CAS

```
//traitement
response.getWriter();
//traitement
response.getOutputStream(); 
//traitement
```

2<sup>nd</sup> CAS

```
PrintWriter out = response.getWriter();
out.println("Début réponse");
//...
//envoi réponse a débuté
//traitement
response.sendRedirect("URL");  *
//traitement
```

\*response.sendError(x); 







## Démonstration



**web.xml**

```
<servlet>
  <servlet-name>UneServlet</servlet-name>
  <servlet-class>...</servlet-class>
  <init-param>
    <description>...</description>
    <param-name>NOM_PARAMETRE</param-name>
    <param-value>VALEUR_PARAMETRE</param-value>
  </init-param>
  ...
</servlet>
```



## Les paramètres d'initialisation par annotation



```
@WebServlet
(
    name="UneServlet",
    urlPatterns="/url/de/la/servlet",
    initParams=
    {
        @WebInitParam(description="...",
            name="NOM_PARAMETRE",
            value="VALEUR_PARAMETRE"),
        ...
    }
)
```



## Utilisation des paramètres d'initialisation



```
public class UneServlet extends HttpServlet
{
    private String valeurParametre;

    @Override
    public void init() throws ServletException
    {
        this.valeurParametre=this.getInitParameter("NOM_PARAMETRE");
    }
}
```



Les servlets

## Les paramètres d'initialisation

# Démonstration



37

Les servlets

## Rechercher le nombre tiré au sort (évolution)

# TP



38

## Conclusion

- Vous savez maintenant générer une réponse adaptée avec la technologie des servlets

