

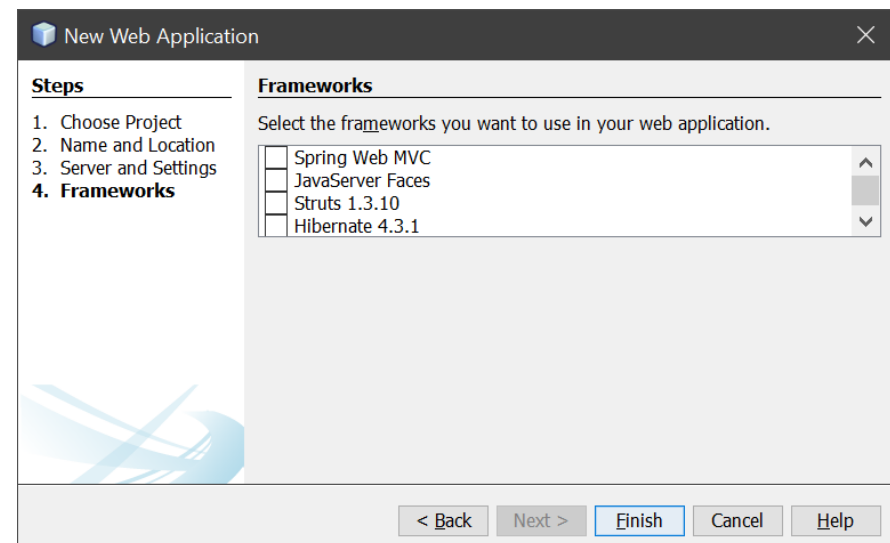
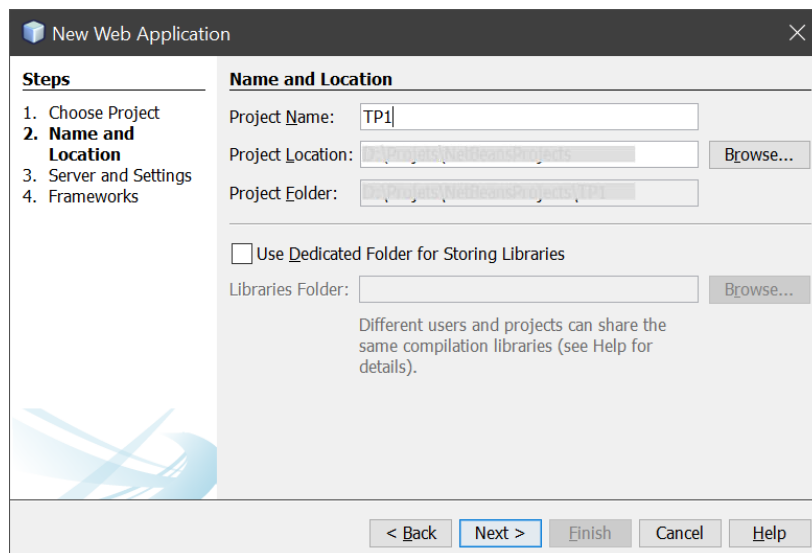
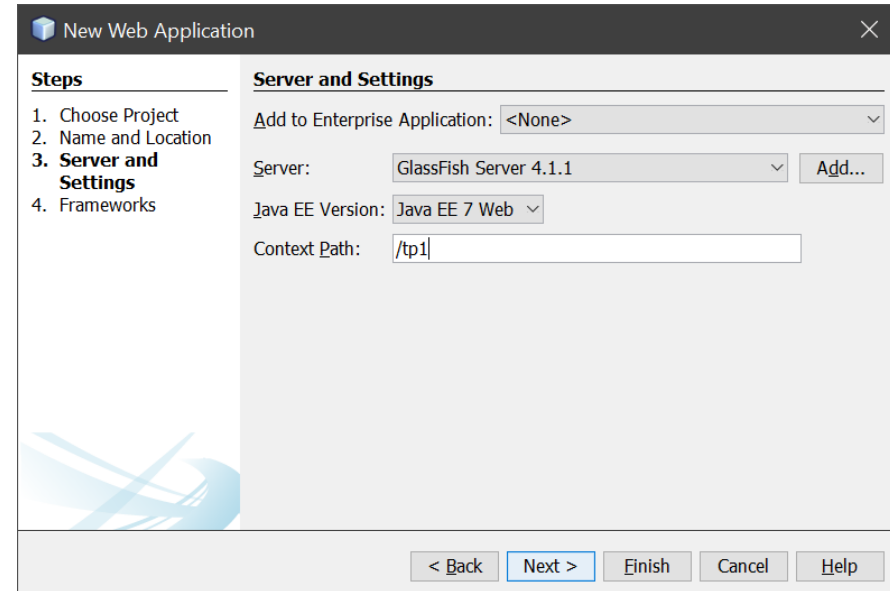
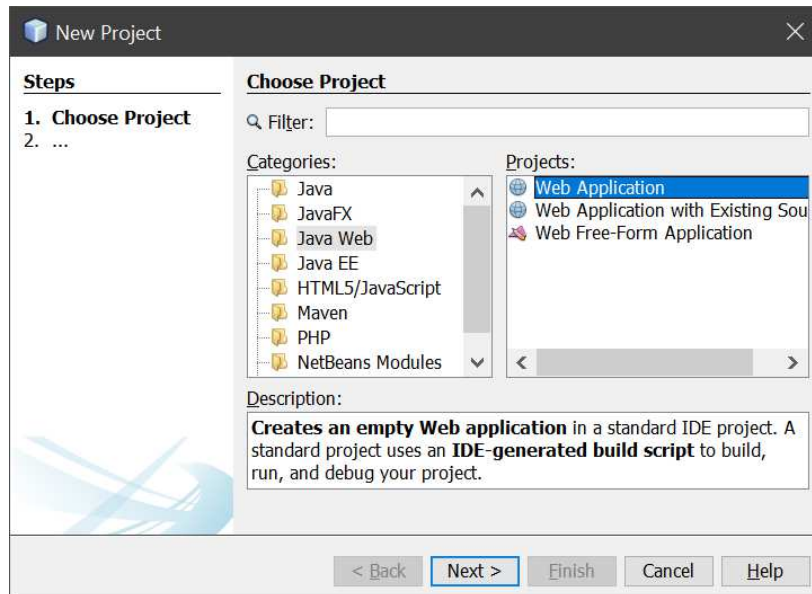
Servlet



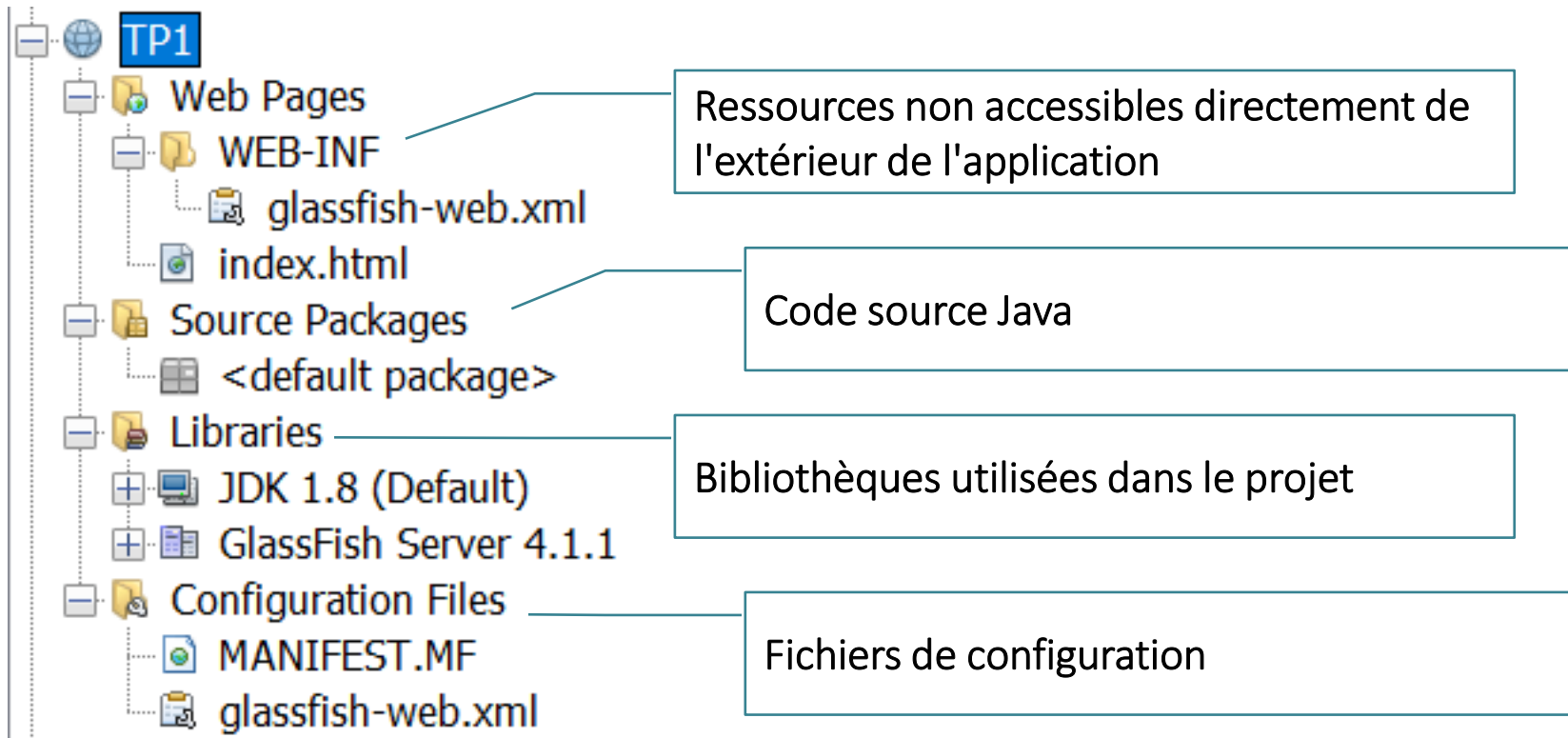


- ☐ Projet Web JEE
- ☐ Notion de Servlet
- ☐ Principe de fonctionnement
- ☐ Déclaration
- ☐ Création
- ☐ Gestion des cookies
- ☐ Gestion des sessions
- ☐ Communication entre Servlets

Création d'un projet JEE Web sous NetBeans



Structure d'un projet JEE Web sous NetBeans



Notion de Servlet

- ❑ Classe Java (POJO) qui s'exécute au sein d'un serveur HTTP permettant de traiter les requêtes et de préparer des réponses (des pages HTML par exemple).
- ❑ Se charge soit au démarrage du serveur soit lors de la première requête, puis reste chargée pour répondre à d'autres requêtes
- ❑ Implémente l'interface `Servlet` du package `javax.servlet` directement soit en héritant une des deux classes suivantes :
`javax.servlet.GenericServlet`
`javax.servlet.http.HttpServlet`

Servlet Web

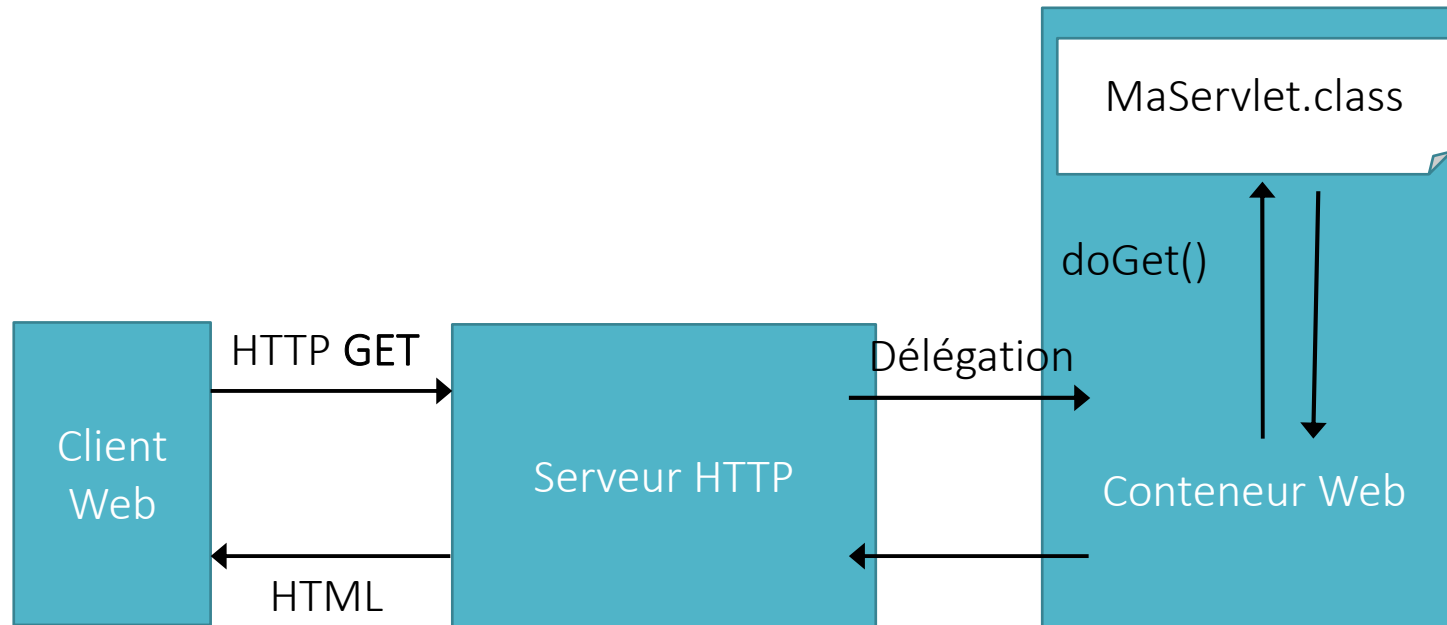
Le conteneur Web du Serveur Java EE réagit aux requêtes HTTP en appelant la méthode appropriée de la servlet qui va se charger de préparer la réponse HTTP.

Méthode HTTP	Méthode appelée par le moteur de Servlets
GET	doGet()
POST	doPost()
HEAD	doHead()
OPTIONS	doOptions()
PUT	doPut()
DELETE	doDelete()
TRACE	doTrace()

Deux objets sont passés à ces méthodes lorsqu'elles sont appelées :

- ☐ un objet de type [HttpServletRequest](#) : possède tous les renseignements sur les paramètres passés à la requête.
- ☐ un objet de type [HttpServletResponse](#) : permet d'obtenir un flux de sortie pour communiquer la réponse au client.

Principe de fonctionnement



Mais il faut une URL pour appeler une servlet !

Déclaration d'une Servlet

Première façon : via le descripteur de déploiement *web.xml*

`<web-app>`

`<servlet>`

`<servlet-name>` nom de servlet `</servlet-name>`

`<servlet-class>` nom de la classe de servlet `</servlet-class>`

`<init-param>`

`<param-name>` nom du par. d'initialisation `</param-name>`

`<param-value>` valeur du par. d'initialisation `</param-value>`

`</init-param>`

`</servlet>`

`<servlet-mapping>`

`<servlet-name>` nom de servlet `</servlet-name>`

`<url-pattern>` url d'invocation de servlet `</url-pattern>`

`</servlet-mapping>`

`</web-app>`

Nom attribué à la Servlet
(Exemple : MaServlet)

Nom de la classe qui
implémente la Servlet
(Exemple : TestServlet.class)

Paramètres d'initialisation récupérés via la méthode
`javax.servlet.ServletConfig.getInitParameter()`

URL(s) d'invocation de la servlet
(Exemple : /test)

La DTD contient plus de 50 balises !

https://docs.oracle.com/cd/E24329_01/web.1211/e21049/web_xml.htm#WBAPP525

Déclaration d'une Servlet

Deuxième façon : via les annotations

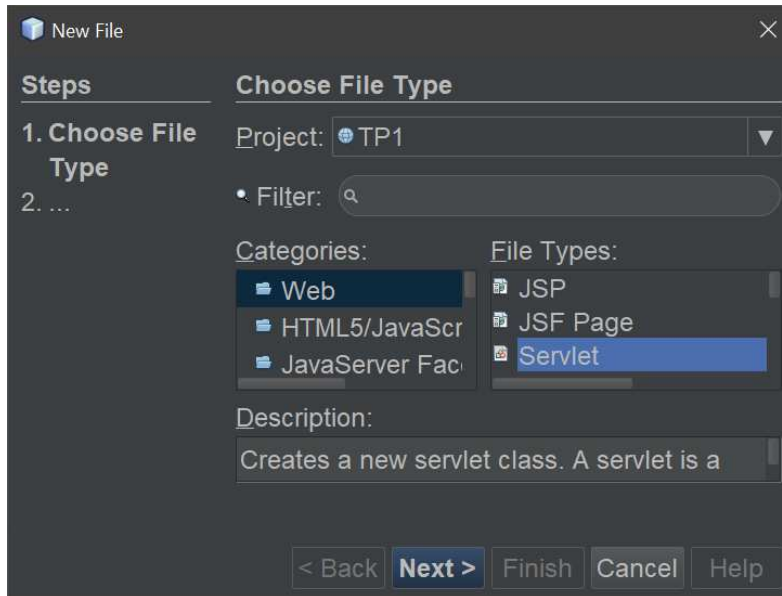
Avant la déclaration de la classe

```
@WebServlet(paramètre = valeur, ...)
```

Quelques paramètres :

name	Nom de la servlet dont valeur est une chaîne de caractères
urlPatterns ou value	patrons d'URL de la servlet dont la valeur est un tableau de chaînes de caractères { }
loadOnStartup	L'ordre de démarrage de la Servlet dont la valeur est un entier; si négatif le conteneur est libre de choisir le moment de démarrage, sinon 0 pour le plus prioritaire et le démarrage se fait avec le démarrage de l'application.

Création d'une Servlet (NetBeans)



New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project: TP1

Filter:

Categories:

- Web
- HTML5/JavaScript
- JavaServer Faces

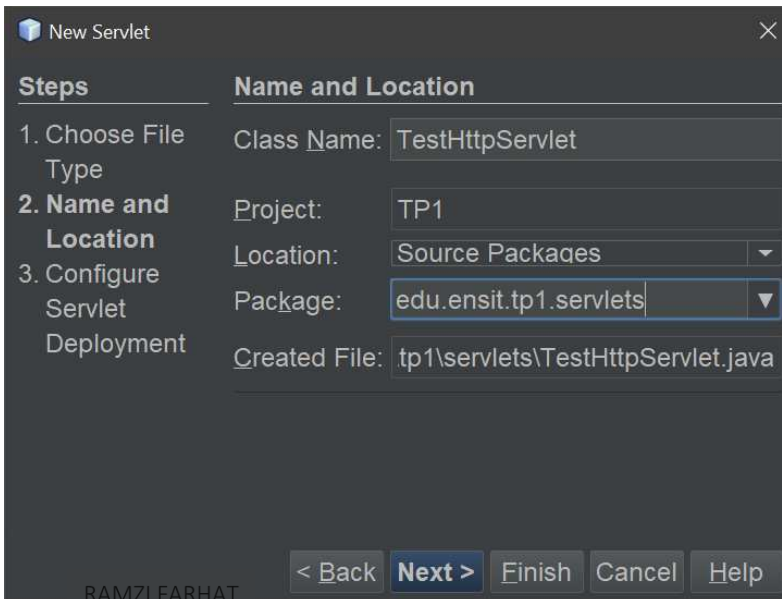
File Types:

- JSP
- JSF Page
- Servlet

Description:

Creates a new servlet class. A servlet is a

< Back **Next >** Finish Cancel Help



New Servlet

Steps

1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment

Name and Location

Class Name: TestHttpServlet

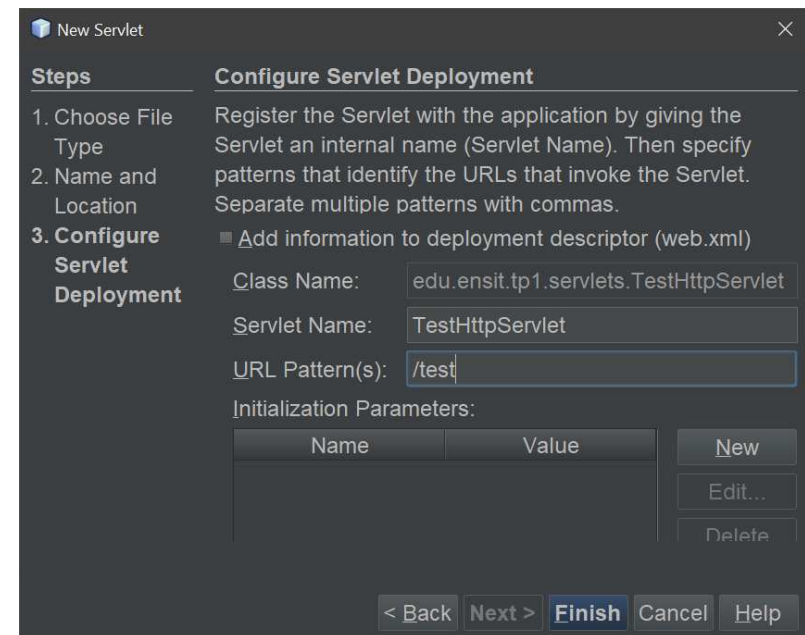
Project: TP1

Location: Source Packages

Package: edu.ensit.tp1.servlets

Created File: tp1\servlets\TestHttpServlet.java

< Back **Next >** Finish Cancel Help



New Servlet

Steps

1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☒ Add information to deployment descriptor (web.xml)

Class Name: edu.ensit.tp1.servlets.TestHttpServlet

Servlet Name: TestHttpServlet

URL Pattern(s): /test

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

< Back **Next >** **Finish** Cancel Help

Création d'une Servlet

```
package edu.ensit.tp1.servlets;
```

```
+ import ...7 lines
```

```
@WebServlet(name = "TestHttpServlet", urlPatterns = {"/test"})
```

```
public class TestHttpServlet extends HttpServlet {
```

```
+     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {...15 lines }
```

```
@Override
```

```
-     protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
```

```
@Override
```

```
-     protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
```

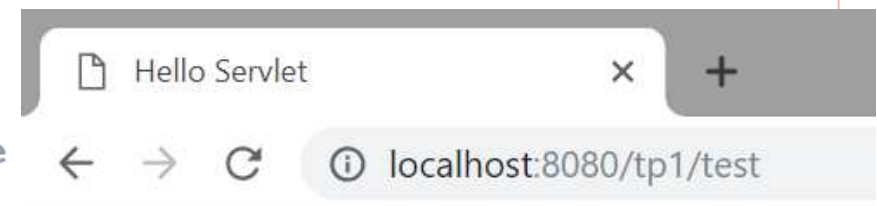
```
@Override
```

```
+     public String getServletInfo() {...3 lines } // </editor-fold>
```

Création d'une Servlet

@Override

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Préciser le Type du contenu de la réponse et l'encodage
    response.setContentType("text/html");
    response.setCharacterEncoding("UTF-8");
    // Récupérer un objet PrintWriter pour écrire le contenu de la réponse
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Hello Servlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Hello Servlet !</h1>");
    out.println("</body>");
    out.println("</html>");
    // Fermer le PrintWriter de la réponse
    out.close();
}
```



Hello Servlet !

Quelques méthodes utiles

HttpServletRequest

<code>String getParameter(String param)</code>	Retourne la valeur du champ param transmis dans les données du formulaire
<code>java.util.Enumeration getParameterNames()</code>	retourne l'ensemble des noms de paramètres transmis à la servlet
<code>String getMethod()</code>	retourne la méthode HTTP (GET ou POST) utilisée pour invoquer la servlet
<code>Map<String, String[]> getParameterMap()</code>	Retourne un MAP dont les clés sont les noms des paramètres et les valeurs c'est les valeurs des paramètres

Quelques méthodes utiles

HttpServletResponse

void <code>setContentType</code> (String type)	Définit le type MIME (Multipurpose Internet Mail Extensions) du document retourné par la servlet (Exemple : "text/html; charset=UTF-8")
PrintWriter <code>getWriter</code> ()	Retourne un flux de sortie permettant à la servlet de produire son résultat (la servlet écrit le code HTML sur ce flux de sortie)
void <code>sendRedirect</code> (String location)	Permet de rediriger vers une URL (le paramètre location) donnée en paramètre

Exercice

Créer une Servlet qui permet d'afficher les noms et les valeurs des paramètres envoyés dans une requête HTTP tout en spécifiant la méthode d'envoi.

← → ↻ ⓘ localhost:8080/tp1_2018/infosRequete?nom=Farhat&Prenom=Ramzi

Méthode d'envoi de la requête :

GET

Paramètres de la requête :

Nom	Valeur
nom	Farhat
Prenom	Ramzi

Gestion de cookies

❑ Cookie :

❑ *Contenu textuel envoyé par un serveur HTTP et stocké dans un client HTTP*

❑ *Envoyé par le client HTTP dans chaque requête envoyée au même serveur HTTP*

❑ *Dispose des attributs :*

❑ name (nom - obligatoire)

❑ value (valeur - obligatoire)

❑ expires (date d'expiration)

❑ path (restriction à un chemin sous le nom de domaine)

❑ domain (domaine)

❑ secure (accès restreint aux connexion https)

❑ httponly (accès interdit via les scripts côté client)

❑ options (tableau associatif contenant certains attributs)

Plus de détails sur :

<https://developer.mozilla.org/fr/docs/Web/HTTP/Cookies>

The screenshot shows the MDN web docs page for 'HTTP cookies'. The page is in French and provides a comprehensive overview of cookies, including their definition, how they are sent and received, and their various attributes. The left sidebar contains a navigation menu with links to 'Technologies web pour développeurs', 'Sujets associés', and a list of guides including 'HTTP', 'HTTP access control (CORS)', 'HTTP authentication', 'HTTP caching', 'HTTP compression', 'HTTP conditional requests', 'HTTP content negotiation', 'HTTP cookies', 'HTTP range requests', 'HTTP redirects', 'HTTP specifications', and 'Feature policy'. The main content area is divided into sections: 'Un cookie HTTP (cookie web, cookie de navigateur) est un petit ensemble de données qu'un serveur envoie au navigateur web de l'utilisateur...', 'Les cookies sont utilisés pour 3 raisons principales :', 'Gestion des sessions', 'Personnalisation', 'Suivi', and 'Création de cookies'. The 'Création de cookies' section includes a code example for a Set-Cookie header and a list of programming languages that support cookies. The 'Cookies de session' section explains that session cookies are deleted when the browser is closed. The 'Cookies permanents' section explains that permanent cookies have an expiration date. The 'Cookies Secure et HttpOnly' section explains the secure and httponly attributes.

MDN web docs Technologies Guides et références Votre avis Connexion

HTTP cookies

Langues Modifier

Aller à : Création de cookies Sécurité Suivi et confidentialité Voir aussi

Technologies web pour développeurs > HTTP > HTTP cookies

Sujets associés

HTTP

Guides:

- Resources and URIs
- HTTP guide
- HTTP security
- HTTP access control (CORS)
- HTTP authentication
- HTTP caching
- HTTP compression
- HTTP conditional requests
- HTTP content negotiation
- HTTP cookies
- HTTP range requests
- HTTP redirects
- HTTP specifications
- Feature policy

References:

- HTTP headers
- HTTP request methods
- HTTP response status codes
- CSP directives
- CORS errors
- Feature-Policy directives

Un cookie HTTP (cookie web, cookie de navigateur) est un petit ensemble de données qu'un serveur envoie au navigateur web de l'utilisateur. Le navigateur peut alors le stocker localement, puis le renvoyer à la prochaine requête vers le même serveur. Typiquement, cette méthode est utilisée par le serveur pour déterminer si deux requêtes proviennent du même navigateur — pour exemple pour garder un utilisateur connecté. Les cookies permettent de conserver de l'information en passant par le protocole HTTP qui est lui "sans état".

Les cookies sont utilisés pour 3 raisons principales :

Gestion des sessions

Logins, panier d'achat, score d'un jeu, ou tout autre chose dont le serveur doit se souvenir.

Personnalisation

Préférences utilisateur, thèmes, et autres paramètres.

Suivi

Enregistrement et analyse du comportement utilisateur.

Les cookies étaient auparavant utilisés pour le stockage côté client. C'était légitime lorsque les cookies étaient la seule manière de stocker des données côté client, mais il est aujourd'hui recommandé de préférer les APIs modernes de stockage. Les cookies sont envoyés avec chaque requête, ils peuvent donc avoir un impact négatif sur les performances (particulièrement pour des connexions mobiles). Les APIs modernes de stockage côté client sont l'API Web storage (localStorage et sessionStorage) et IndexedDB.

Pour voir les cookies stockés (et d'autres stockages que le navigateur peut conserver), vous ouvrez l'inspecteur de stockage des Outils Développeur et sélectionnez Cookies dans l'onglet stockage (pour Firefox).

Création de cookies

Après avoir reçu une requête HTTP, un serveur peut renvoyer sa réponse avec une ou des entêtes(s) Set-Cookie. Le cookie ou les cookies ainsi définis sont habituellement stockés par le navigateur, puis renvoyés lors des prochaines requêtes au même serveur, dans une entête HTTP cookie. Une date d'expiration ou une durée peut être spécifiée par cookie, après quoi le cookie ne sera plus envoyé. De plus, des restrictions à un domaine ou un chemin spécifiques peuvent être spécifiées, limitant quand le cookie est envoyé.

Les entêtes Set-Cookie et Cookie

L'entête de réponse HTTP Set-Cookie envoie un cookie depuis le serveur vers le navigateur. Un cookie simple est défini comme ceci:

```
Set-Cookie: <nom-du-cookie>=<valeur-du-cookie>
```

Note: Voici comment utiliser l'en-tête Set-Cookie dans divers langages de programmation:

- PHP
- Node.js
- Python
- Ruby on Rails

Exemple de réponse HTTP complète:

```
1 HTTP/1.0 200 OK
2 Content-type: text/html
3 Set-Cookie: yummy_cookie=choco
4 Set-Cookie: tasty_cookie=strawberry
5
6 [contenu de la page]
```

Maintenant, à chaque requête vers le serveur, le navigateur va renvoyer au serveur tous les cookies stockés, avec l'entête Cookie:

```
1 GET /sample_page.html HTTP/1.1
2 Host: www.example.org
3 Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

Cookies de session

Le cookie créé ci-dessus est un cookie de session : il est effacé quand le navigateur est fermé, puisqu'on a pas spécifié de directive Expires ou Max-Age. Notons cependant que les navigateurs web peuvent utiliser la **restauration de session**, ce qui fait de la plupart des cookies des cookies permanents, comme si le navigateur n'avait jamais été fermé.

Cookies permanents

Plutôt que d'expirer quand le client ferme, les cookies permanents expirent à une date spécifique (Expires) ou après un certain temps (Max-Age).

```
1 Set-Cookie: id=a3f9a; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
```

Note: Quand une date d'expiration est définie, le temps et l'heure définis sont relatifs au client auquel le cookie est envoyé, et non au serveur.

Cookies Secure et HttpOnly

Gestion des cookies

- ❑ Méthode de la classe `HttpRequest` pour lire les cookies d'une requête (return null si aucun cookie) :

```
Cookie[] getCookies();
```

- ❑ Méthode de la classe `HttpReponse` pour ajouter un cookie à une réponse :

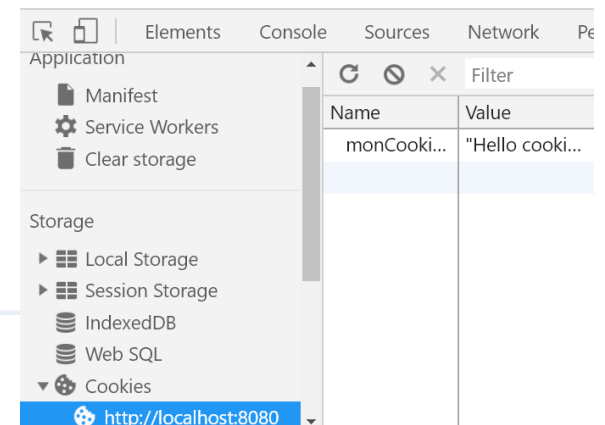
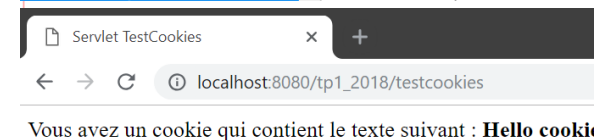
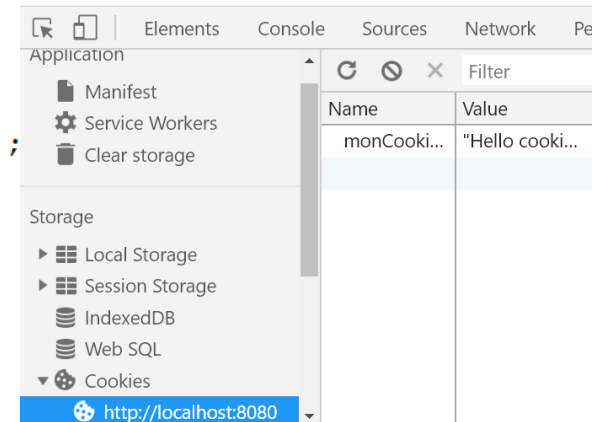
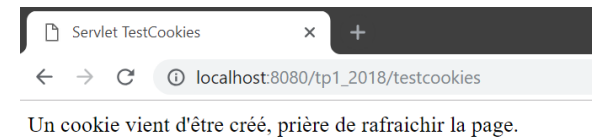
```
void addCookie(Cookie);
```

- ❑ Méthodes de la classe `javax.servlet.http.Cookie` pour

<code>Cookie("nom", "valeur");</code>	Construire un Cookie
<code>String getName();</code>	Récupérer le nom d'un Cookie
<code>String getValue();</code>	Récupérer la valeur d'un Cookie
<code>void maxAge(int secondes);</code>	Fixer la date d'expiration d'un cookie (0 secondes pour supprimer le cookie)

Exemple

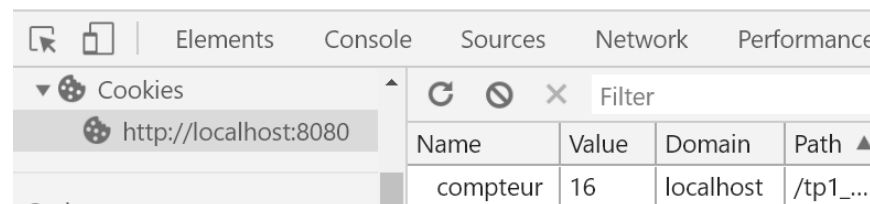
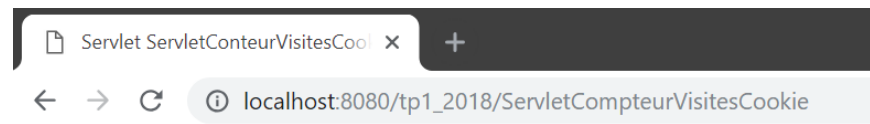
```
Cookie[] mesCookies= request.getCookies();
String contenuCookie, msg;
if(mesCookies == null){
    Cookie helloCookie= new Cookie("monCookieDeTest", "Hello cookie !");
    helloCookie.setMaxAge(5*60);
    response.addCookie(helloCookie);
    msg= "Un cookie vient d'être créé, prière de rafraichir la page.";
}
else{
    contenuCookie = mesCookies[0].getValue();
    msg= "Vous avez un cookie qui contient le texte suivant : "+
        "<strong>"+contenuCookie+"</strong>";
}
// Affichage
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head><title>Servlet TestCookies</title></head>");
    out.println("<body>");
    out.println(msg);
    out.println("</body>");
    out.println("</html>");
}
```



Gestion des cookies

Exercice : compteur de visites

- ☐ Lors de la première visite il faut :
 - ☐ *créer un nouveau cookie avec le nom "compteur", avec la valeur 1 et avec une date d'expiration d'une année.*
 - ☐ *afficher le nombre de visites (qui est 1)*
- ☐ Lors de la n^{ième} visite il faut :
 - ☐ *mettre à jour le cookie (valeur et date d'expiration)*
 - ☐ *afficher le nombre de visites*



Gestion des sessions

❑ Objectif : Garder des informations sur les interaction du client sur le serveur

❑ Problème : HTTP est un protocole sans état

❑ Solution : Objet Session

❑ *Objet côté serveur géré par le conteneur*

❑ *Permet de stocker des informations durant l'interaction avec un client particulier*

❑ *Disparition si destruction de l'objet ou expiration de la session*

❑ Définition de la durée d'expiration au niveau global : web.xml

`<web-app>`

`<session-config>`

`<session-timeout>10</session-timeout>`

`</session-config>`

`</web-app>`

Durée en minutes

❑ Définition de la durée d'expiration pour une session :

❑ *utiliser la méthode : `public void setMaxInactiveInterval(int secondes)` de la classe `javax.servlet.http.HttpSession` (0 secondes : pas d'expiration)*

Gestion des sessions

Quelques méthodes de la classe `HttpServletRequest`

<code>HttpSession getSession()</code>	Retourne la session (<code>HttpSession</code>) associée à la requête, sinon elle crée une nouvelle session .
<code>HttpSession getSession(boolean create)</code>	Fonctionne de la même façon que <code>getSession()</code> , en cas d'absence de session c'est le booléen en paramètre qui va préciser si on va créer une ou non.
<code>boolean isRequestedSessionIdFromCookie()</code>	Retourne vrai si l'ID de la session vient d'une cookie (<code>JSESSIONID</code>)
<code>boolean isRequestedSessionIdFromURL()</code>	Retourne vrai si l'ID est contenu dans l'URL

Remarque : l'ID de la session est stocké côté client dans un cookie qui porte le nom `JSESSIONID`

Gestion des sessions

Quelques méthodes de la classe `HttpSession` :

<code>void setMaxInactiveInterval(int)</code>	Permet de spécifier la durée de vie d'une Session en secondes
<code>void invalidate()</code>	Permet de terminer une session
<code>void setAttribute(String, Object)</code>	Permet d'ajouter un attribut à une session sous forme nom/objet
<code>Object getAttribute(String)</code>	Permet de récupérer un attribut à partir de son nom
<code>void removeAttribute(String)</code>	Permet de supprimer un attribut à partir de son nom
<code>java.util.Enumeration getAttributeNames()</code>	Permet de récupérer les noms des attributs dans l'objet session

Communication entre servlets

Renvoyer la requête et la réponse à une autre servlet (ou une autre ressource) :

Méthode de `HttpServletRequest` (ou de `ServletContext`)

`RequestDispatcher` `getRequestDispatcher(String)`

Création d'un `RequestDispatcher` avec comme paramètre un chemin d'une ressource (statique ou dynamique) interne à l'application pour lui transférer la requête ou l'inclure dans la réponse

Méthodes de `RequestDispatcher`

`void include(HttpServletRequest, HttpServletResponse)`

Inclure le contenu (statique ou dynamique) de la ressource dans la réponse dynamique retournée

`void forward(HttpServletRequest, HttpServletResponse)`

Transférer le traitement à une autre ressource dans le serveur

Exemple

```
response.setContentType("text/html");
response.setCharacterEncoding("UTF-8");
try (PrintWriter out = response.getWriter()) {
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<meta charset='UTF-8'>");
    out.println("<title>Servlet LoginServlet</title>");
    out.println("</head>");
    out.println("<body>");
    request.getRequestDispatcher("/WEB-INF/inc/monHeader.html").include(request, response);
    out.println("<h1>Contenu de la servlet principale </h1>");
    request.getRequestDispatcher("/monFooter").include(request, response);
    out.println("</body>");
    out.println("</html>");
}
```

```
<footer style="width: 100%; background-color: yellow;">
    <h1>Header de page </h1>
</footer>
```

monHeader.html

```
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    out.println("<footer style=\"width: 100%; background-color: yellow;\">\n" +
        "    <h1> Entête de page </h1>\n" +
        "</footer>");
}
```

RAMZI FARHAT

monFooter

47

Header de page

Contenu de la servlet principale

Entête de page

Partage de données entre Servlets

Usage d'un objet `ServletContext` pour partager des couples nom/objet entre les servlets instanciées

Méthode de Servlet

`ServletContext` *getServletContext()*

Permet de récupérer l'objet `ServletContext` qui représente le contexte d'exécution de la Servlet

Méthodes de `ServletContext`

`void` *setAttribute(String, Object)*

`Object` *getAttribute(String)*

`void` *removeAttribute(String)*

`java.util.Enumeration` *getAttributeNames()*

`String` *getInitParameter(String)*

```
<context-param>
  <param-name> ...</param-name>
  <param-value> ...</param-value>
</context-param>
...
web.xml
```

TP Gestion de Sessions

Exercice

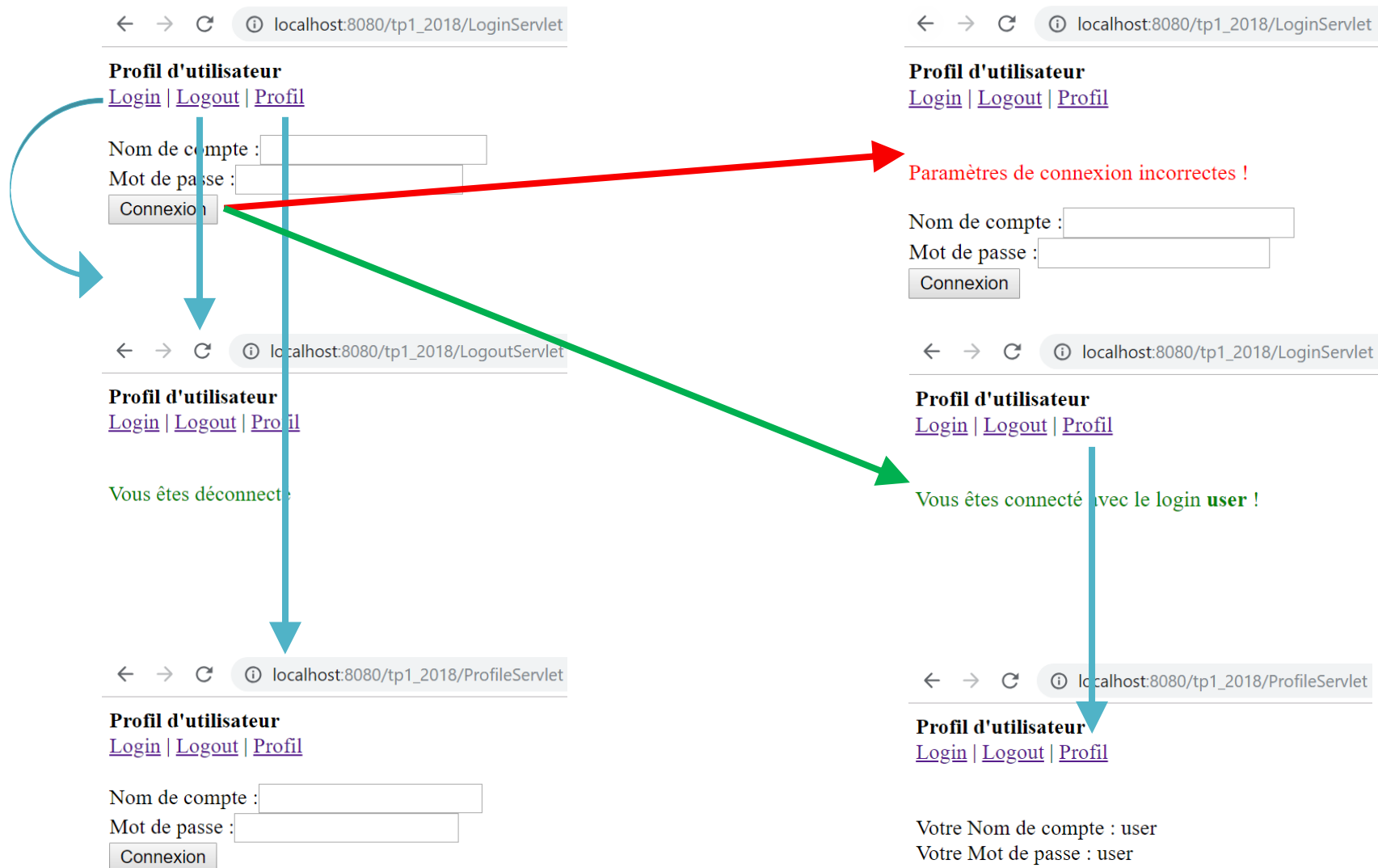
Créez une application Web Java EE permettant d'autoriser les utilisateurs à se connecter pour pouvoir consulter une page. L'application doit contenir trois pages :

LoginServlet : permettant la connexion via un nom d'utilisateur et un mot de passe

LogoutServlet : permettant la déconnexion de l'utilisateur

ProfilServlet : permettant d'afficher les informations sur l'utilisateurs sauvegardées dans la session s'il est connecté, sinon, redirection ver la page de Connexion.

TP Gestion de Sessions





- ☐ Projet Web JEE
- ☐ Notion de Servlet
- ☐ Principe de fonctionnement
- ☐ Déclaration
- ☐ Création
- ☐ Gestion des cookies
- ☐ Gestion des sessions
- ☐ Communication entre Servlets