

# JEE - Tomcat - L'API Servlet



P.Mathieu

IUT de Lille

<http://www.iut-a.univ-lille.fr>

prenom.nom@univ-lille.fr

1. l'API Servlet – rappels – redirections ....
2. Session - Filter - Listener ....
3. Maven ....
4. JSP, Beans auto-managés - EL - JSTL
5. JPA
6. Spring Core + Spring Data JPA
7. Spring MVC + validation
8. Spring Rest
9. Spring Security
10. OAuth2 + JWT + Authentification par tiers

compil à la main  
compil avec un IDE  
compil avec Maven et Cargo  
utilisation Postgres  
utilisation de H2

## Web dynamique

## Fabriquer une URL

## Objectif

- ▶ Génération de pages HTML à la volée
- ▶ Génération de pages spécifiques à la personne se connectant
- ▶ Génération de pages issues d'un calcul ou fonction de données
- ▶ Ex : Commerce électronique, Annuaire, Intranet, etc ...

## Principe :

- ▶ Le code est exécuté sur le serveur
- ▶ Le résultat est envoyé au client
- ▶ **Le client ne reçoit que le résultat, pas le code de génération !**

**CGI** technique classique et "ancestrale"

**ASP** (VB) ancienne technique Microsoft

**PHP**

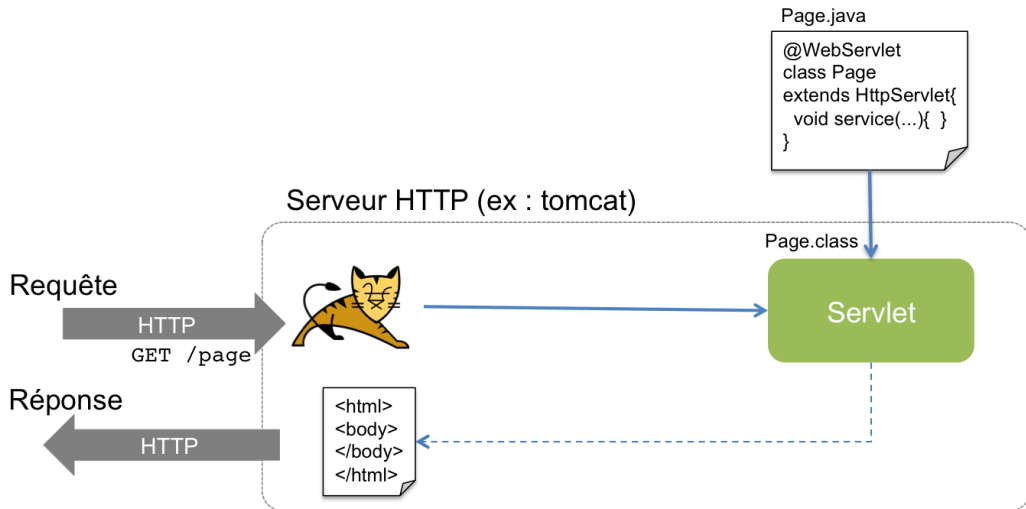
**Zope** (python)

**Rails** (Ruby)

**Java EE** (Java)

**.NET** "dot net" (C#) copie de Java EE par Microsoft

...



**Java 2 Enterprise Edition** : architecture de composants multi-plateformes proposée par SUN bâtie sur le langage JAVA.

JEE = J2SE + Servlets + JDBC + EJB

Le composant de base de l'architecture Java EE est un objet spécial : **La Servlet**

Définie dans deux paquetages : `javax.servlet` et `javax.servlet.http`

Interfaces	Classes
<code>HttpServletMapping</code> <b><code>HttpServletRequest</code></b> <b><code>HttpServletResponse</code></b> <b><code>HttpSession</code></b> <code>HttpSessionActivationListener</code> <code>HttpSessionAttributeListener</code> <code>HttpSessionBindingListener</code> <code>HttpSessionContext</code> <code>HttpSessionListener</code> ...	<code>Cookie</code> <code>HttpFilter</code> <b><code>HttpServlet</code></b> <code>HttpServletRequestWrapper</code> <code>HttpServletResponseWrapper</code> <code>HttpSessionBindingEvent</code> <code>HttpSessionEvent</code> <code>HttpUtils</code>

Contenu dans `servlet-api.jar` fourni par les serveurs WEB.

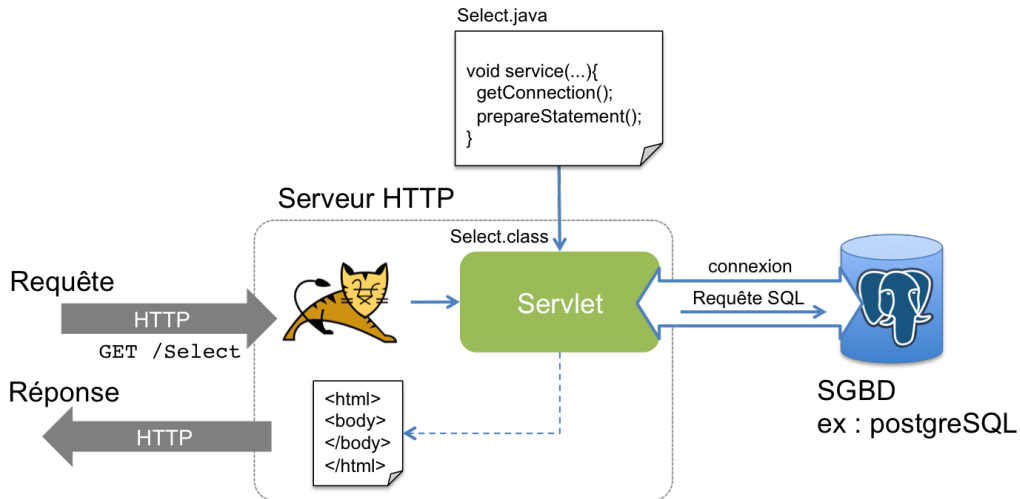


## Un exemple

```
import java.util.Date;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.WebServlet;

@WebServlet("/servlet-MyDate")
public class MyDate extends HttpServlet
{
    public void service( HttpServletRequest req, HttpServletResponse res )
        throws ServletException, IOException
    {
        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out = res.getWriter();
        out.println( "<html><body>" );
        out.println( "<h1>Nous sommes le "+new Date()+"</h1>" );
        out.println( "</body></html>" );
    }
}
```

`http://127.0.0.1/contexte/servlet-MyDate`



```
import java.sql.*;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebServlet("/Servlet-Select")
public class Select extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out = res.getWriter();
        out.println( "<html><body>");
        Connection con;
        try { // ... récupération de la connexion
            String query = "select NOM,PRENOM,AGE from CLIENTS";
            PreparedStatement ps = con.prepareStatement(query);
            ResultSet rs = ps.executeQuery();
            while (rs.next())
            {
                String nom = rs.getString("nom");
                String p = rs.getString("prenom");
                int a = rs.getInt("age");
                out.println(n + " " + p + " " + a + "<br>");
            }
            out.println( "</body></html>" );
        }
        // ... fermeture et gestion des exceptions
    }
}
```

## Tomcat

### Serveur Java EE de référence réalisé par le consortium Apache

```
Tomcat
  bin
  conf
  work
  lib
    servlet-api.jar
  webapps
    contextel
      mapage.html
      WEB-INF
        web.xml
        lib
          postgresql.jar
        classes
          First.class
    contexte2
      .....
```

<http://localhost/contextel/page1.html>

<http://localhost/contextel/servlet-First>

Web dynamique

**Fabriquer une URL**

# Fabriquer une URL

## Trois manières de fabriquer une URL

The screenshot shows the Facebook homepage in a web browser. Three red arrows point from text annotations to specific parts of the page:

- Saisie d'URL GET**: Points to the address bar showing the URL `https://fr-fr.facebook.com`.
- Formulaire HTML GET ou POST**: Points to the registration form titled "Inscription".
- Hyperlien GET**: Points to the link "Créer une Page" at the bottom of the page.

The registration form includes the following fields and options:

- Adresse électronique ou téléphone
- Mot de passe
- Garder ma session active
- Mot de passe oublié ?
- Connexion
- Prénom
- Nom de famille
- Adresse électronique
- Confirmer l'adresse électronique
- Nouveau mot de passe
- Date de naissance (Jour, Mois, Année)
- Pourquoi dois-je indiquer ma date de naissance ?
- Femme
- Homme
- Inscription
- Créer une Page pour une célébrité, un groupe ou une entreprise.

### ► Get

- Mode d'appel de la ligne d'URL du navigateur ou d'un hyperlien (tag `<A>`).
- Les paramètres sont concaténés à l'URI :

```
http://127.0.0.1/contexte/servlet/comptes?nom=Dupond&prenom=pierre
```

- L'url et ses paramètres sont visibles dans la barre d'adresses

### ► Post

- Les paramètres ne sont pas visibles de l'utilisateur.
- Uniquement utilisable à partir d'un formulaire (tag `<Form>`)
- Adapté à un grand nombre de paramètres

# Fabriquer une URL

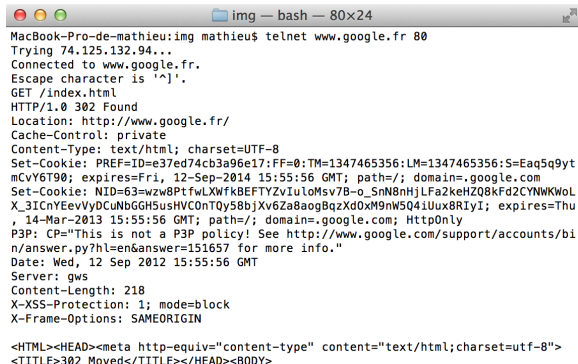
## Derrière le navigateur

- Le serveur WEB reçoit une requête (GET, POST) de fourniture d'un fichier et renvoie ce fichier

```
> telnet www.google.fr 80  
> GET /index.html HTTP/1.1
```

ou

```
> curl -i www.google.fr
```



```
MacBook-Pro-de-mathieu:img mathieu$ telnet www.google.fr 80  
Trying 74.125.132.94...  
Connected to www.google.fr.  
Escape character is '^J'.  
GET /index.html  
HTTP/1.0 302 Found  
Location: http://www.google.fr/  
Cache-Control: private  
Content-Type: text/html; charset=UTF-8  
Set-Cookie: PREF=ID=e37ed74cb3a96e17:FF=0:TM=1347465356:LM=1347465356:S=Eaq5q9yt  
mCvY6T90; expires=Fri, 12-Sep-2014 15:55:56 GMT; path=/; domain=.google.com  
Set-Cookie: NID=63=wzw8PtfwLXWfkBEFTYZvIuloMsv7B-o_SnN8nHjLFa2keHZQ8kFd2CYNWkWoL  
X_3ICnYEEvVyDCuNbGGHSusHVCOnTQy58bjXv6Za8aogBqzXd0xM9nW5Q4iUux8RIyI; expires=Thu  
, 14-Mar-2013 15:55:56 GMT; path=/; domain=.google.com; HttpOnly  
P3P: CP="This is not a P3P policy! See http://www.google.com/support/accounts/bi  
n/answer.py?hl=en&answer=151657 for more info."  
Date: Wed, 12 Sep 2012 15:55:56 GMT  
Server: gws  
Content-Length: 218  
X-XSS-Protection: 1; mode=block  
X-Frame-Options: SAMEORIGIN  
  
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">  
<TITLE>302 Moved</TITLE></HEAD><BODY>
```

- Le client WEB (Navigateur) interprète ce fichier en fonction d'un type MIME.



# Fabriquer une URL

## Un formulaire

```
<html>
<form action=Servlet-Bonjour method=get>
  Saisir votre nom <input name=n type=text> <br/>
  <input type=submit>
</form>
</html>
```

Ce formulaire génère une requête

`http://localhost/contexte/Servlet-Bonjour?n=paul`

```
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.WebServlet;

@WebServlet("/Servlet-Bonjour")
public class Bonjour extends HttpServlet
{
    public void service( HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out = res.getWriter();
        out.println( "<html><body>" );
        out.println( "<h1>Bonjour " + req.getParameter("n") + "</h1>" );
        out.println( "</body></html>" );
    }
}
```

Si le paramètre n'est pas présent dans la requête, `getParameter` renvoie `null`

## Trois méthodes de traitement

- ▶ `public void doGet(...)`  
répond aux requêtes HTTP de type GET
- ▶ `public void doPost(...)`  
répond aux requêtes HTTP de type POST
- ▶ `public void service(...)`  
qui appelle `doGet` ou `doPost` en fonction de la requête HTTP reçue

Redéfinir `service` empêche d'utiliser `doGet` ou `doPost`

### Méthodes principales

- ▶ `getContentType()` Type MIME de requete
- ▶ `getRemoteAddr()` IP du client
- ▶ `getMethod()` type de requete réseau
- ▶ `getRemoteUser()` login de l'utilisateur sur le poste client
- ▶ `getParameter(String)` lecture d'un paramètre
- ▶ `getHeader()` entêtes HTTP
- ▶ `getParameterNames()` lecture des noms de tous les paramètres
- ▶ `getParameterValues(String)`
- ▶ ...

### Méthodes principales :

- ▶ `getWriter()` pointeur vers un flux texte `PrintWriter`
- ▶ `getOutputStream()` pointeur vers un flux binaire
- ▶ `sendRedirect(url)` renvoie l'ordre au navigateur de recharger la page passée en paramètre (en Get)
- ▶ .....

Parfois une Servlet n'exécute que du code, sans rien afficher

Deux méthodes de redirection

- ▶ Via le client

```
response.sendRedirect(url)
```

- ▶ Interne au serveur

```
RequestDispatcher.forward(req, res)
```

```
RequestDispatcher rd=request.getRequestDispatcher("chemin");  
rd.forward(request, res);
```

- ▶ Tomcat 10 – Java  $\geq 8$  – Servlet 5.0  
(remplacer `javax` en `jakarta`)
- ▶ Serveur Postgres
  - ▶ Machine : `psqlserv`
  - ▶ Database : `but3`
  - ▶ Login : `prenomnetu` (sans les points)
  - ▶ Password : `moi`
- ▶ IDE : aux choix (VSCode ou IntelliJ)