

JEE : Servlet

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`



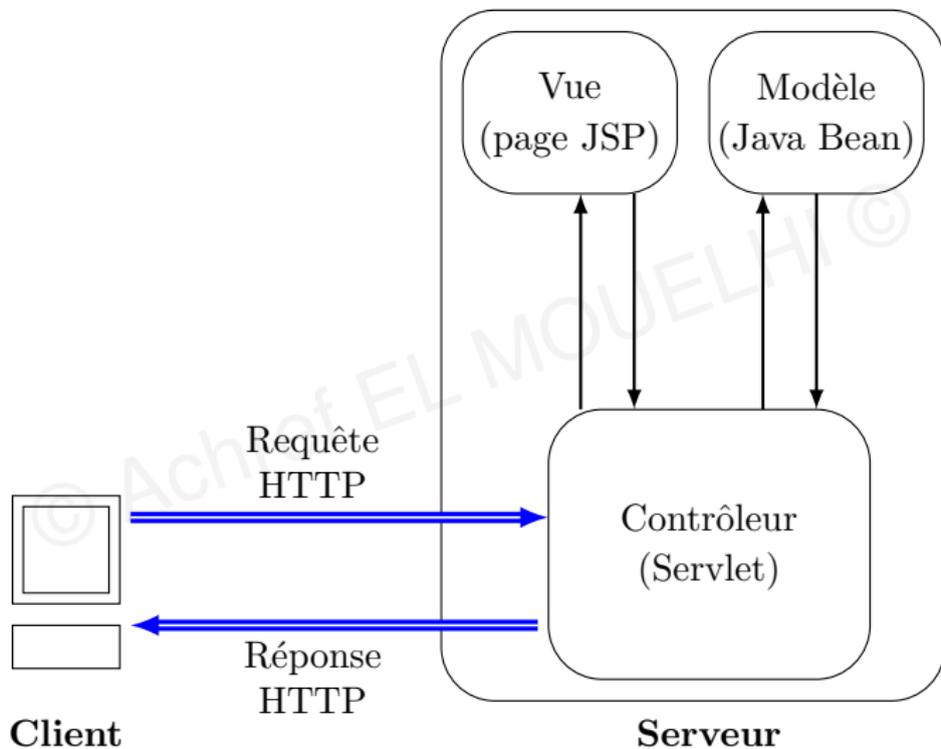
Plan

- 1 Introduction
- 2 Structure d'une Servlet
- 3 Première Servlet avec Eclipse
 - Routage par annotation
 - Routage dans `web.xml`
- 4 Tester la Servlet
- 5 Servlet multi-routes
- 6 Objet `HttpServletRequest`
- 7 Paramètres de requête
 - `getParameter()`
 - `getParameterValues()`
 - `getParameterMap()`
- 8 Rediriger vers une autre Servlet

Servlet : le cœur d'une application **JEE**

- Classe **Java** héritant de la classe `HttpServlet`
- Recevant une requête **HTTP** (de type `GET`, `POST`...) et retournant une réponse **HTTP**
- Contrôleur du modèle **MVC** dans une application **JEE**

Jakarta EE



Servlet : classe Java héritant de `HttpServlet`

```
package org.eclipse.controller;  
  
import javax.servlet.http.HttpServlet;  
  
public class TestServlet extends HttpServlet {  
  
}
```

Jakarta EE

Explication

- `HttpServlet` contient des méthodes abstraites, préfixées par `do()`, associées aux différentes méthodes (verbes) **HTTP**.
 - `doGet()` : s'exécute quand l'utilisateur demande une page (via la barre d'adresse, un lien hypertexte...)
 - `doPost()` : s'exécute quand l'utilisateur envoie des données via un formulaire par exemple
 - ...
- Chaque méthode prend en paramètre :
 - `HttpServletRequest` : contenant des informations sur la requête utilisateur
 - `HttpServletResponse` : permettant de personnaliser la réponse à retourner à l'utilisateur

Jakarta EE

Ajoutons les méthodes `doGet ()` et `doPost ()` à `TestServlet`

```
package org.eclipse.controller;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) {
    }
}
```

Mais quand cette Servlet sera exécutée ?

Quand l'utilisateur saisit une **URL** dans le navigateur, il envoie une requête **HTTP** à notre contrôleur (qui est en vrai une Servlet)

© Achref EL MOUADIB

Jakarta EE

Mais quand cette Servlet sera exécutée ?

Quand l'utilisateur saisit une **URL** dans le navigateur, il envoie une requête **HTTP** à notre contrôleur (qui est en vrai une Servlet)

Et si on avait plusieurs Servlets, laquelle sera exécutée ?

- Chaque Servlet aura sa propre route (uniques)
- La Servlet ayant la route demandée sera exécutée

Comment associer une route à une Servlet ?

- soit avec l'annotation `@WebServlet`
- soit dans le fichier `web.xml`

© Achref EE

Comment associer une route à une Servlet ?

- soit avec l'annotation `@WebServlet`
- soit dans le fichier `web.xml`

Commençons par créer une Servlet avec **Eclipse**

Pour créer une **Servlet** sous **Eclipse**

- Faire un clic droit sur `src` situé dans `Java Resources` de notre projet
- Aller dans `New` et choisir `Servlet`
- Remplir le champ `Java package :` par `org.eclipse.controller` (par exemple)
- Remplir le champ `Class name :` par un nom suffixé par le mot `Servlet` : `TestServlet` (par exemple)
- Cliquer sur `Next`

Routage par annotation (par défaut)

- On peut modifier ou supprimer l'URL Mappings. Remplaçons la chaîne existante (`/TestServlet`) par `/mapage`
- Cliquer sur `Next`
- Décocher la case `Constructors from superclass`
- Vérifier que les cases correspondantes aux deux méthodes `doGet()` et `doPost` sont cochées
- Valider en cliquant sur `Finish`

Jakarta EE

Le contenu généré par Eclipse

```
package org.eclipse.controller;

// les imports

@WebServlet("/mapage")
public class TestServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

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

Jakarta EE

Routage dans `web.xml`

Le fichier `web.xml` situé dans `WEB-INF` de `webapp` permet de :

- déclarer la Servlet
- associer une URL à une Servlet (Mapping URL/Servlet)

Jakarta EE

Si le fichier n'existe pas

- Faire un clic droit sur `WEB-INF` de `webapp` de notre projet
- Aller dans `New` et choisir `Other`
- Saisir `xml` dans la zone de recherche
- Choisir `XML File`
- Cliquer sur `Next` et choisir le nom `web.xml`

Jakarta EE

Contenu de web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
  https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd" id="
  WebApp_ID" version="5.0">

  <display-name>cours-jee</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Jakarta EE

`<welcome-file-list>`

- Contient les différents formats de fichiers qui peuvent être utilisés comme page d'accueil de l'application
- Ces fichiers seront directement dans `webapp`
- Pas besoin d'une Servlet pour les afficher
- Accessibles via la route `/` ou directement via leur nom

Jakarta EE

Dans web.xml, on déclare la Servlet avant </web-app>

```
...  
<servlet>  
  <servlet-name>TestServlet</servlet-name>  
  <servlet-class>org.eclipse.controller.TestServlet</servlet-class>  
</servlet>
```

© Achref EL MOULI

Jakarta EE

Dans web.xml, on déclare la Servlet avant `</web-app>`

```
...  
<servlet>  
  <servlet-name>TestServlet</servlet-name>  
  <servlet-class>org.eclipse.controller.TestServlet</servlet-class>  
</servlet>
```

Explication

- `<servlet>` et `</servlet>` : déclaration de la Servlet
- `<servlet-name>` et `</servlet-name>` : permet d'attribuer un nom à la Servlet qu'on utilisera plus tard
- `<servlet-class>` et `</servlet-class>` : indique le chemin de la classe de la Servlet

Jakarta EE

Autres sous balises disponibles pour Servlet

- `<description>` **et** `</description>` : ajouter une description sur le fonctionnement de la Servlet (comme un commentaire)
- `<load-on-startup>` **et** `</load-on-startup>` : permet de forcer le chargement de la Servlet lors de démarrage
- ...

Jakarta EE

N'oublions pas, le rôle du `web.xml` :

- déclarer la Servlet (**c'est fait**)
- faire le mapping (assurer le routage si cela n'a pas été fait avec les annotations)

Jakarta EE

```
...  
<servlet-mapping>  
  <servlet-name>TestServlet</servlet-name>  
  <url-pattern>/mapage</url-pattern>  
</servlet-mapping>  
</web-app>
```

© Achref EL MOUËL

Jakarta EE

```
...  
<servlet-mapping>  
  <servlet-name>TestServlet</servlet-name>  
  <url-pattern>/mapage</url-pattern>  
</servlet-mapping>  
</web-app>
```

Explication

- `<servlet-mapping>` et `</servlet-mapping>` : pour faire le mapping Servlet/url
- `<servlet-name>` et `</servlet-name>` : permet d'indiquer le nom de la Servlet à appeler
- `<url-pattern>` et `</url-pattern>` : indique l'URL qui provoquera l'appel de la Servlet indiquée dans la la sous-balise précédente

Jakarta EE

Contenu de web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="
  https://jakarta.ee/xml/ns/jakartaee" xsi:schemaLocation="https://
  jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-
  app_5_0.xsd" id="WebApp_ID" version="5.0">

<servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-class>org.eclipse.controller.TestServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/mapage</url-pattern>
</servlet-mapping>
</web-app>
```

Jakarta EE

Remarque

Dans la suite de ce cours, on n'utilisera que le routage par annotation.

Une seule étape à faire

- Cliquer sur Run
- Une page blanche affichée ayant comme
 - `adresse` : `http://localhost:8080/cours-jee/mapage`
 - `contenu` : `Served at: /cours-jee`

Si on teste une autre URL inexistante

- Écrire dans la zone d'adresse
`http://localhost:8080/cours-jee/tapage`
- Une page HTTP 404 sera affichée

Comment afficher le `Hello World`

Il faut modifier la `Servlet` (l'objet `HttpServletResponse` qui est responsable de la réponse)

Jakarta EE

Nouveau contenu de la Servlet

```
public class TestServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {

        response.getWriter().print("Hello World");
    }

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

Pour exécuter une deuxième fois

- Cliquer sur Run
- Choisir `Continue without restarting` (pas besoin de redémarrer le serveur)

Jakarta EE

On peut indiquer l'encodage et le type du contenu de la réponse

```
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

    // pour indiquer le type de réponse
    response.setContentType("text/html");

    // indiquer l'encodage UTF-8 pour éviter les problèmes avec les
    accents
    response.setCharacterEncoding("UTF-8");

    PrintWriter out = response.getWriter();
    out.println("Hello World");
}
```

L'objet `PrintWriter`

- s'obtient de l'objet `response`
- permet d'envoyer un (ou des) message(s) à l'utilisateur

Pour retourner une page HTML complète

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException{
    response.setContentType("text/html");
    response.setCharacterEncoding("UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<meta charset=\"utf-8\" >");
    out.println("<title>Projet JEE</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("Hello World");
    out.println("</body>");
    out.println("</html>");
}
```

Constats

- Beaucoup de code dans la **Servlet** (trop long) pour un affichage simple
- Violation du modèle **MVC** : le contrôleur n'affiche pas de résultat.

© Achref EE

Constats

- Beaucoup de code dans la **Servlet** (trop long) pour un affichage simple
- Violation du modèle **MVC** : le contrôleur n'affiche pas de résultat.

Solution

Utiliser des vues pour l'affichage (chapitre suivant).

A une Servlet peuvent être associées plusieurs routes

```
@WebServlet ({"/route1", "/route2", ... /routeN})
```

Remarque

Pour récupérer la route qui a permis d'exécuter la Servlet, on utilise l'objet de type `HttpServletRequest`.

`http://localhost:8080/nom-projet/route-servlet?param1=value1¶m2=value2`

Jakarta EE

Protocole

Scheme



<http://localhost:8080/nom-projet/route-servlet?param1=value1¶m2=value2>

Jakarta EE

Protocole
Scheme

ServerName

http://localhost:8080/nom-projet/route-servlet?param1=value1¶m2=value2

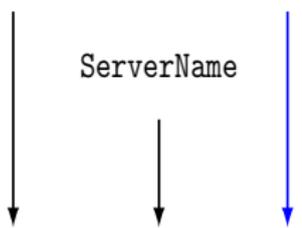
Jakarta EE

Protocole

Scheme

ServerPort

ServerName



http://localhost:8080/nom-projet/route-servlet?param1=value1¶m2=value2

Jakarta EE

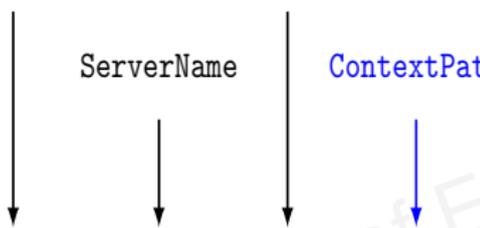
Protocole

Scheme

ServerPort

ServerName

ContextPath



http://localhost:8080/nom-projet/route-servlet?param1=value1¶m2=value2

Jakarta EE

Protocole

Scheme

ServerPort

route

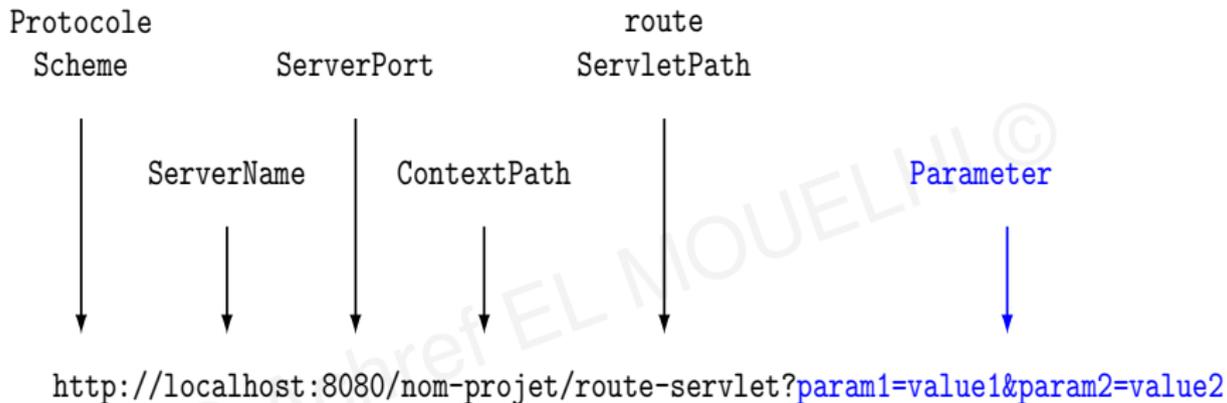
ServletPath

ServerName

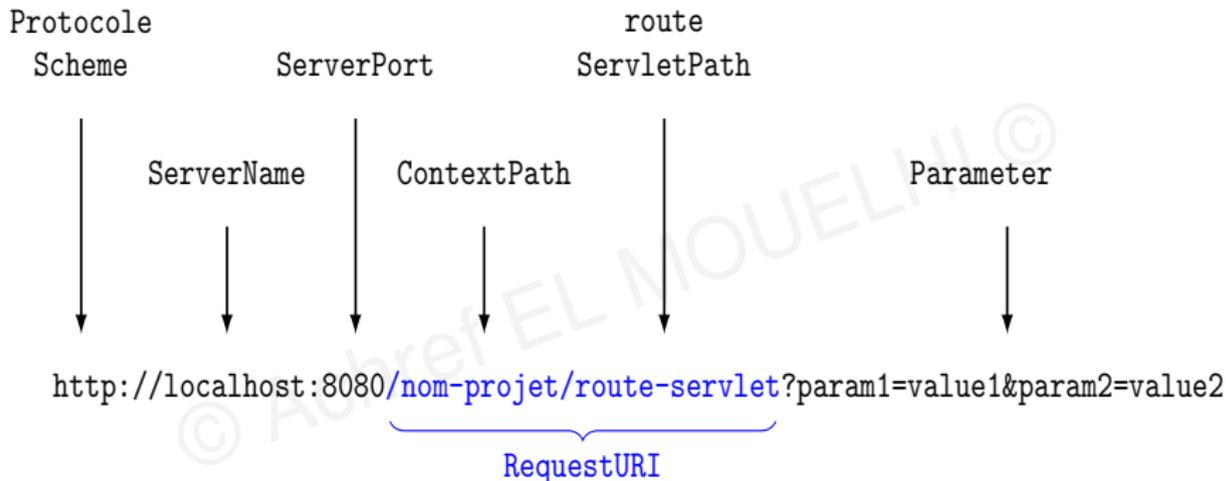
ContextPath

http://localhost:8080/nom-projet/**route-servlet**?param1=value1¶m2=value2

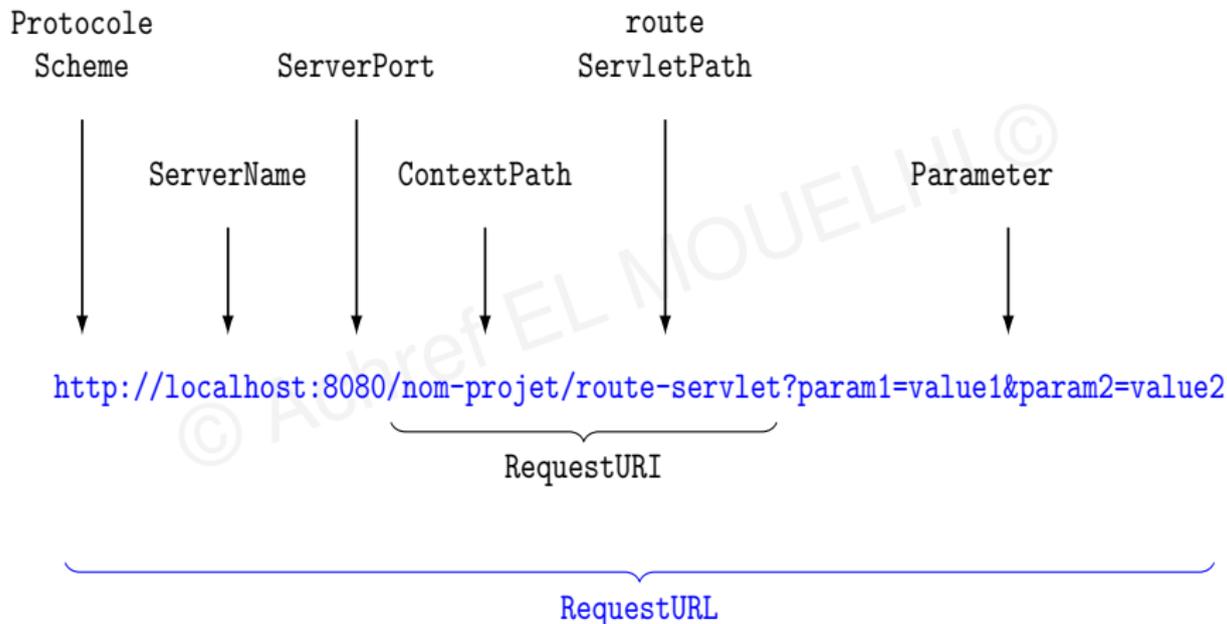
Jakarta EE



Jakarta EE



Jakarta EE



Jakarta EE

Comment récupérer toutes ces informations ?

Tout est défini dans l'objet de type `HttpServletRequest`.

© Achref EL MOUELHI ©

Jakarta EE

Comment récupérer toutes ces informations ?

Tout est défini dans l'objet de type `HttpServletRequest`.

Exemples

Il suffit de préfixer le nom des propriétés précédentes par `get`

- `request.getContextPath()`
- `request.getServletPath()`
- `request.getServerPort()`
- ...

Exercice 1

- Créez une Servlet `CalculServlet` accessible via une des quatre routes suivantes
 - `/calcul/plus`
 - `/calcul/moins`
 - `/calcul/fois`
 - `/calcul/div`
- La Servlet doit afficher chaque fois la deuxième partie de la route demandé
 - `plus`
 - `moins`
 - `fois`
 - `div`

Récupérer les paramètres d'une requête

- Mais, une requête peut avoir de paramètres (par exemple `/mapage?nom=Wick&prenom=John`)
- Comment, dans ce cas, récupérer les paramètres ?

© Achref L.

Récupérer les paramètres d'une requête

- Mais, une requête peut avoir de paramètres (par exemple /mapage?nom=Wick&prenom=John)
- Comment, dans ce cas, récupérer les paramètres ?

Solution

```
request.getParameter("nomParameter");
```

Jakarta EE

Exemple de récupération et d'affichage de paramètres de la requête

```
protected void doGet (HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {

    String nom = request.getParameter("nom");
    String prenom = request.getParameter("prenom");
    PrintWriter out = response.getWriter();
    out.print("Hello " + nom + " " + prenom);
}
```

© ACP

Jakarta EE

Exemple de récupération et d'affichage de paramètres de la requête

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {

    String nom = request.getParameter("nom");
    String prenom = request.getParameter("prenom");
    PrintWriter out = response.getWriter();
    out.print("Hello " + nom + " " + prenom);
}
```

Remarque

Si le paramètre n'est pas présent, on peut avoir une exception.

Jakarta EE

Il est recommandé de vérifier si la valeur du paramètre est non nulle avant de l'utiliser

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {
    String nom = request.getParameter("nom");
    String prenom = request.getParameter("prenom");
    PrintWriter out = response.getWriter();
    if (nom != null && prenom != null) {

        out.print("Hello " + nom + " " + prenom);
    } else {
        out.print("Hello world");
    }
}
```

Récupérer les paramètres d'une requête

À ne pas confondre

- Les paramètres de requête : concept relatif aux requêtes **HTTP**
- Les attributs de requête : concept introduit dans **JEE** (à voir dans le prochain chapitre)

Exercice 2

Modifiez `CalculServlet` pour qu'elle

- accepte deux paramètres `a` et `b`
- Si la route demandée est
 - `/calcul/plus?a=2&b=5`, alors la Servlet affiche le résultat de l'addition.
 - `/calcul/moins?a=2&b=5`, alors la Servlet affiche le résultat de la soustraction.
 - `/calcul/fois?a=2&b=5`, alors la Servlet affiche le résultat de la multiplication.
 - `/calcul/div?a=2&b=5`, alors la Servlet affiche le résultat de la division.

Jakarta EE

Si le paramètre est présent plusieurs fois dans l'URL avec le même nom, alors on peut utiliser `getParameterValues()`

```
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

    PrintWriter out = response.getWriter();
    String[] noms = request.getParameterValues("nom");
    if (noms != null) {
        for (String nom : noms) {
            out.print("Hello " + nom);
        }
    }
}
```

Jakarta EE

Si le paramètre est présent plusieurs fois dans l'URL avec le même nom, alors on peut utiliser `getParameterValues()`

```
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {

    PrintWriter out = response.getWriter();
    String[] noms = request.getParameterValues("nom");
    if (noms != null) {
        for (String nom : noms) {
            out.print("Hello " + nom);
        }
    }
}
```

Exemple d'URL pour tester

```
http://localhost:8080/test-jee/home?nom=wick&nom=dalton
```

Exercice 3

- Créez une Servlet `MoyenneServlet` accessible via la route `/moyenne`.
- En allant sur `localhost:8080/cours-jee/moyenne?note=12¬e=15`, la moyenne des notes passées en paramètre sera affichée.
- Le nombre de notes est variable.

Jakarta EE

Pour récupérer tous les paramètres dans un `Map`, on peut utiliser `getParameterMap()`

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    PrintWriter out = response.getWriter();
    Map<String, String[]> params = request.getParameterMap();
    for (var param : params.entrySet()) {
        out.print(param.getKey());
        for (var value : param.getValue()) {
            out.print(value);
        }
    }
}
```

Jakarta EE

Pour récupérer tous les paramètres dans un `Map`, on peut utiliser `getParameterMap()`

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

    PrintWriter out = response.getWriter();
    Map<String, String[]> params = request.getParameterMap();
    for (var param : params.entrySet()) {
        out.print(param.getKey());
        for (var value : param.getValue()) {
            out.print(value);
        }
    }
}
```

Exemple d'URL pour tester

```
http://localhost:
8080/test-jee/home?nom=wick&nom=dalton&genre=homme&age=45
```

Rediriger vers une autre Servlet annotée par `@WebServlet ("/MaServlet")`

```
response.sendRedirect ("MaServlet");
```

© Achref EL MOUËLHANI

Rediriger vers une autre Servlet annotée par `@WebServlet ("/MaServlet")`

```
response.sendRedirect ("MaServlet");
```

Ne pas mettre `"/` avant `MaServlet`.

© Achref EL MOUËLHANI

Jakarta EE

Rediriger vers une autre Servlet annotée par `@WebServlet ("/MaServlet")`

```
response.sendRedirect ("MaServlet");
```

Ne pas mettre "/" avant `MaServlet`.

On peut aussi reconstruire l'URL depuis le `contextPath`

```
response.sendRedirect (request.getContextPath () + "/MaServlet");
```