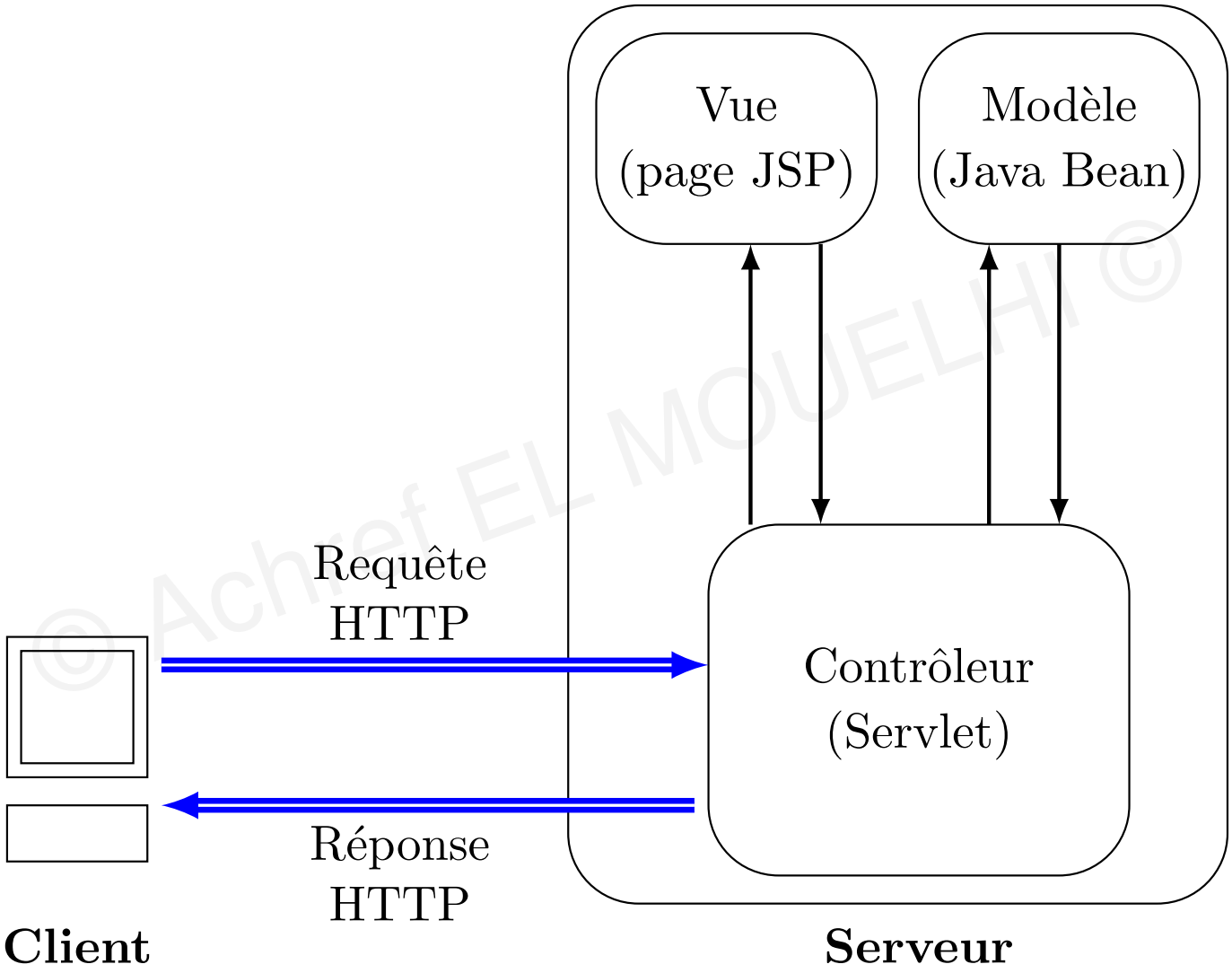
Plan

JEE

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 MV**C** dans une application **JEE**

JEE



JEE

Servlet : classe Java héritant de `HttpServlet`

```
package org.eclipse.config;

import javax.servlet.http.HttpServlet;

public class TestServlet extends HttpServlet {

}
```

JEE

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

JEE

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

```
package org.eclipse.config;

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) {

  }
}
```

JEE

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)
- Mais quelle Servlet ? je peux en avoir plusieurs
- Le serveur va chercher quelle Servlet est associée à cette route

JEE

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)
- Mais quelle Servlet ? je peux en avoir plusieurs
- Le serveur va chercher quelle Servlet est associée à cette route

Comment associer une route à une Servlet ?

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

JEE

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)
- Mais quelle Servlet ? je peux en avoir plusieurs
- Le serveur va chercher quelle Servlet est associée à cette route

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

JEE

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 example)
- Remplir le champ `Class name:` par un nom suffixé par le mot `Servlet` : TestServlet (par example)
- Cliquer sur `Next`

JEE

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`

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);
  }
}
```

Routage dans `web.xml`

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

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

Si le fichier n'existe pas

- Faire un clic droit sur `WEB-INF` de `WebContent` 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`

Contenu de web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://
   xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
   version="3.1">
  <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>
```

JEE

`<welcome-file-list>`

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

JEE

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>
```

JEE

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

JEE

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
- ...

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)

JEE

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

JEE

```
...
<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

JEE

Le contenu de web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://
   xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
   version="3.1">
<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>
```

Remarque

Dans la suite de ce cours, on utilise que le routage par annotation.

Format d'une requête utilisateur

```
http://localhost:8080/cours-jee/URLServlet
```

JEE

Format d'une requête utilisateur

```
http://localhost:8080/cours-jee/URLServlet
```


Comment récupérer ces informations

- `request.getContextPath()` : nom du projet défini par le serveur Apache Tomcat dans la requête
- `request.getServletPath()` : adresse de la Servlet demandée par l'utilisateur (définie soit dans `web.xml` ou dans l'annotation `@WebServlet`)
- `request.getServerPort()` : numéro de port utilisé par le serveur

JEE

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`

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)

JEE

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);
  }
}
```

JEE

Pour exécuter une deuxième fois

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

JEE

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

JEE

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>");
}
```

JEE

Constats

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

JEE

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).

JEE

Récupérer les paramètres d'une requête

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

JEE

Récupérer les paramètres d'une requête

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

Solution

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

JEE

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);
}
```

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)

JEE

Exercice 1

- Créez une Servlet `CalculServlet` accessible via la route `/calcul`
- En allant sur `localhost:8080/cours-jee/calcul?a=2&b=5`, le résultat de la somme de `a` et `b` sera affiché.

JEE

Exercice 2

- Modifiez `CalculServlet` pour qu'elle adapte le résultat affiché en fonction d'un troisième paramètre `op` : par exemple en saisissant `localhost:8080/cours-jee/calcul?a=2&b=5&op=plus`, le résultat de la somme de `a` et `b` sera affiché.
- Modifiez `op` peut avoir quatre valeurs possibles : `plus`, `moins`, `fois` et `div`.

JEE

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

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

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

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

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

JEE

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");
```

Pour avoir plusieurs routes qui permettent d'exécuter une Servlet

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

Dans la Servlet, pour récupérer la route qui a conduit à l'exécution de la Servlet

```
request.getServletPath()
```