Java EE

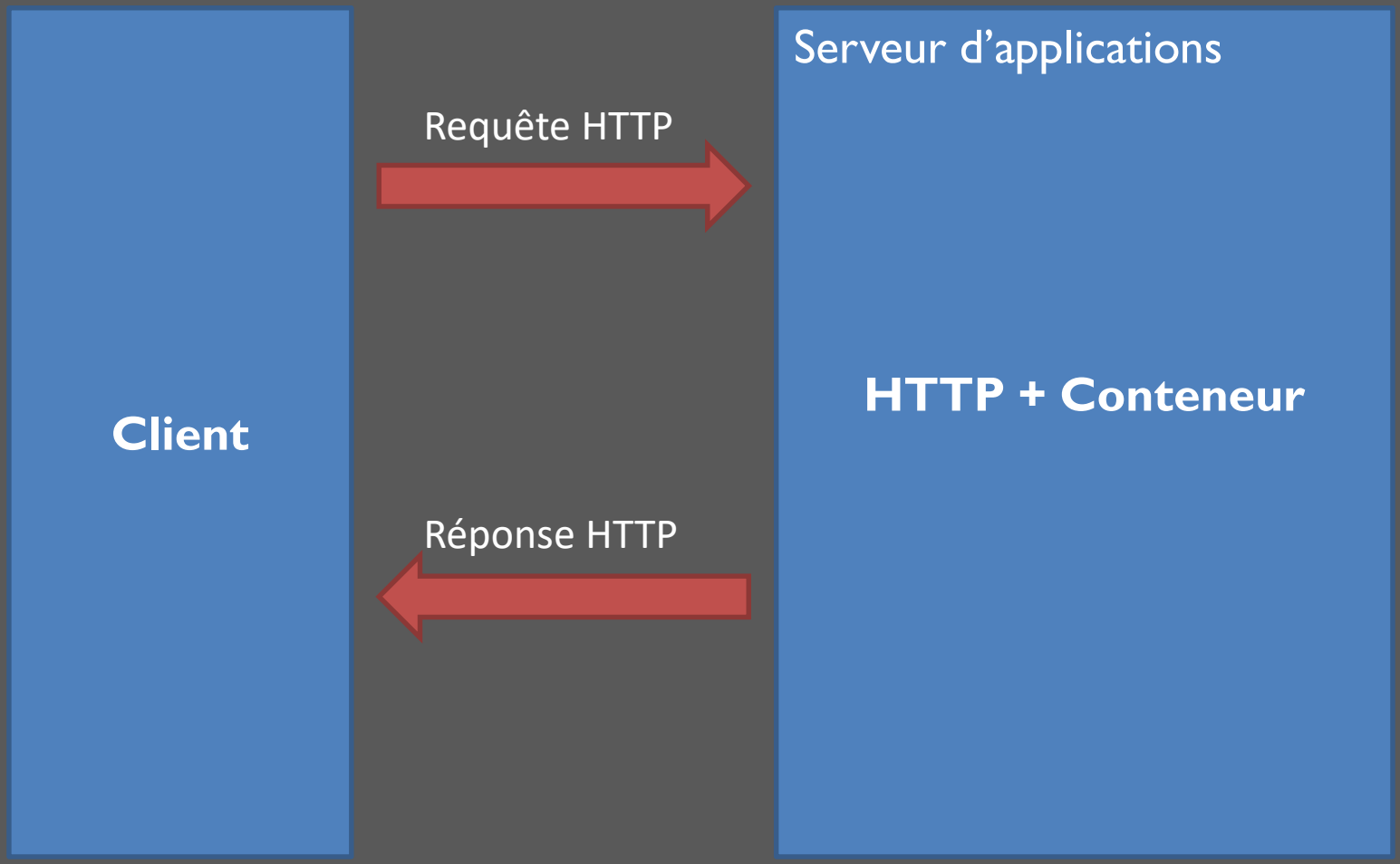
Romain SESSA

Java Entreprise Edition

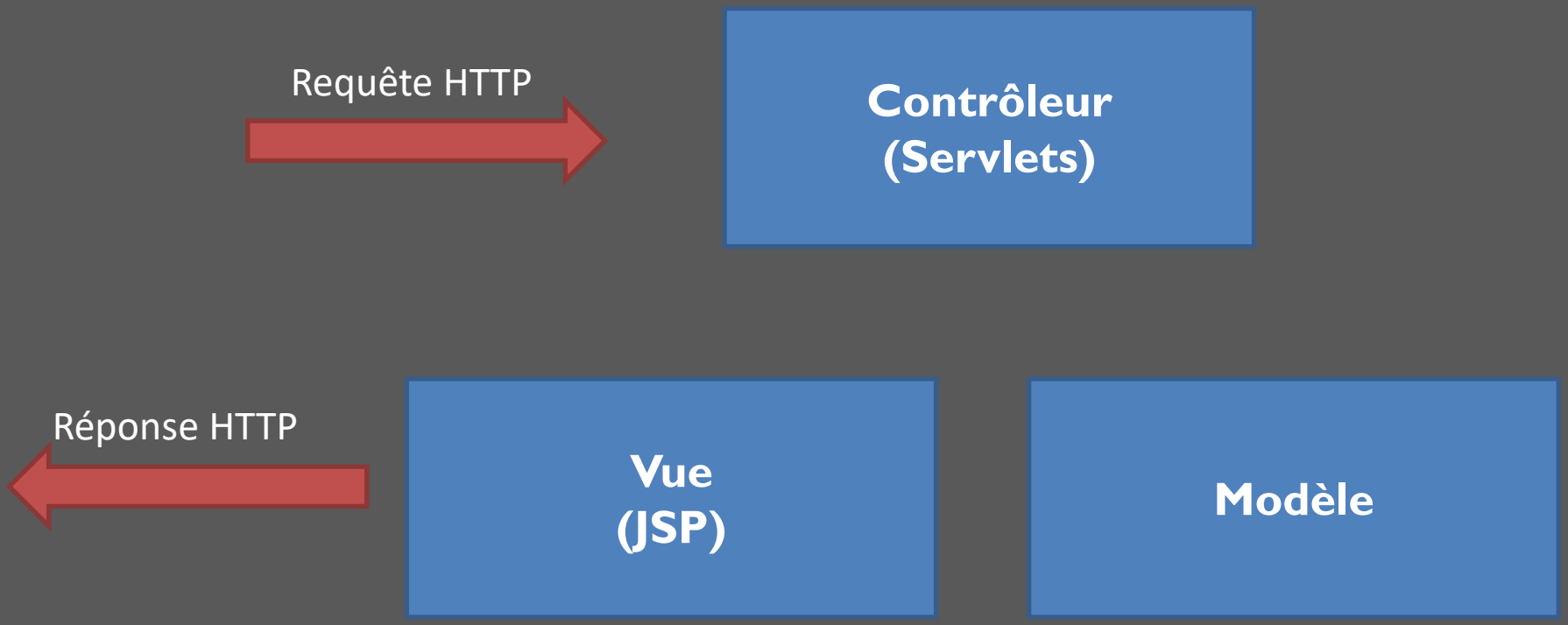
- Basé sur :
  - Java Standard Edition
  - Langage Java

- Ajout de fonctionnalités permettant le déploiement d'application web sur un serveur d'application.

Serveur d'application



MVC


Requête HTTP
Contrôleur
(Servlets)
Réponse HTTP
Vue
(JSP)
Modèle

IDE – Serveur d'application

Eclipse IDE for JAVA EE Developpers

Apache Tomcat Server

Dynamic Web Project

- Création du nouveau projet dans Eclipse :

1. “File” / “New” / “Dynamic Web Project”
2. Saisir le “Project Name”
3. Cliquer sur “New Runtime” (uniquement pour le 1er projet)
    1. Sélectionner la version de Tomcat installée
    2. Cocher “Create a local server”
    3. Selectionner la racine du repertoire Tomcat
4. Cocher “Generate web.xml”

Structure du projet

- **Java Resources** : contient le code JAVA.
- **WebContent** : contient les fichiers de l'application web qui seront déployés dans Tomcat.
  - **WEB-INF** : contient des fichiers qui ne seront pas accessibles à l'utilisateur de l'application.
  - **WEB-INF\web.xml** : fichier de configuration de l'application.

- Créer un fichier "test.html" dans le repertoire "WebContent".
- Le visualiser dans le navigateur via :

  http://localhost:8080/[nom de projet]/test.html

Servlet

- Une requête est redirigé au conteneur (de servlets).
- Le conteneur crée deux objets :
  - HttpServletRequest
  - HttpServletResponse
- Une servlet est une classe JAVA qui manipule ses objets.

Servlet

- Une servlet est une classe JAVA qui étend la classe : *javax.servlet.http.HttpServlet*

- Une servlet doit surcharger au moins une des methodes de sa classe mère.
- Exemple :

```
protected void doGet(
        HttpServletRequest req,
        HttpServletResponse resp)
```

Servlet

- Une servlet doit être déclarée dans le fichier web.xml :

```
<servlet>
        <servlet-name>ServletName</servlet-name>
        <servlet-class>
                com.package.ServletName
        </servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>ServletName</servlet-name>
        <url-pattern>/name</url-pattern>
</servlet-mapping>
```

Transmission des données

- Attribut de rêquete :

  req.setAttribute(“key”,”value”);

- Paramètre de rêquete :

  req.getParamater(“name”);

Servlet

Bonnes pratiques :

- Nom de classe et nom de servlet dans web.xml identiques.
- La servlet n'est pas chargé de l'affichage.

JSP

- Extension : .jsp
- Exécuté côté serveur et page générée puis renvoyée au client.
- Contient du HTML mais aussi des balises JSP.

JSP

- Appel d'un JSP via une servlet :

```
this.getServletContext().getRequestDispatcher(
        "/WEB-INF/test.jsp").forward(
                request, response);
```

- JSP sont placées dans le repertoire WEB-INF et donc non accessible en direct.

JSP

- Ecrire du code JAVA dans une JSP : <% %>
- Ecrire un commentaire : <%-- --%>
- Afficher une expression : <%= %>
- Importer : <%@ include file="uneAutreJSP.jsp" %>

JSP

- Des objets implicites sont instanciés par le conteneur et accessibles les pages JSP :
  pageContext, application, session, request, response, exception, out, config, page

- Les objets peuvent avoir une portée :
  page, request, session, application

Application

- Créer un “Hello World” en J2EE. Cela nécessite :

- Un projet Dynamice Web.
- Une servlet.
- Une JSP
- Un fichier web.xml mis à jour.

Expression Language

- ${param.xxxxx} : Récupère la valeur de xxxxx transmit par le get.
- ${paramValues.xxxxx[0]} : Plusieurs valeurs associées à la même clé.

- ${yyyyyy} : Récupère directement une valeur transmise via requete.setAttribute.

Expression Language

Catégorie	Identifiant	Description
JSP	**pageContext**	Objet contenant des informations sur l'environnement du serveur.
Portées	**pageScope**	Une `Map` qui associe les noms et valeurs des attributs ayant pour portée la page.
	requestScope	Une `Map` qui associe les noms et valeurs des attributs ayant pour portée la requête.
	sessionScope	Une `Map` qui associe les noms et valeurs des attributs ayant pour portée la session.
	applicationScope	Une `Map` qui associe les noms et valeurs des attributs ayant pour portée l'application.
Paramètres de requête	**param**	Une `Map` qui associe les noms et valeurs des paramètres de la requête.
	paramValues	Une `Map` qui associe les noms et multiples valeurs ** des paramètres de la requête sous forme de tableaux de String.
En-têtes de requête	**header**	Une `Map` qui associe les noms et valeurs des paramètres des en-têtes HTTP.
	headerValues	Une `Map` qui associe les noms et multiples valeurs ** des paramètres des en-têtes HTTP sous forme de tableaux de String.
Cookies	**cookie**	Une `Map` qui associe les noms et instances des cookies.
Paramètres d'initialisation	**initParam**	Une `Map` qui associe les données contenues dans les champs `<param-name>` et `<param-value>` de la section `<init-param>` du fichier web.xml

Source : openclassrooms

JSTL

- Bibliothèque de balises JSP.

- Ajouter la bibliothèque :

```
<%@ taglib
uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
```

- Utiliser la bibliothèque :

```
<c:out value="test">
```

JSTL

- Quelques possibilités :

<c:out : Permet d'échapper les caractères spéciaux.
<c:set var= value= scope= /> = request.setAttribute
<:out value="${ requestScope.maVar }"/>
<c:remove var= scope= />
<c:url />
<c:redirect />
<c:import />
<c:if />
<c:forEach />

Questions

?