Chapitre 2 : Architectures des applications web MVC

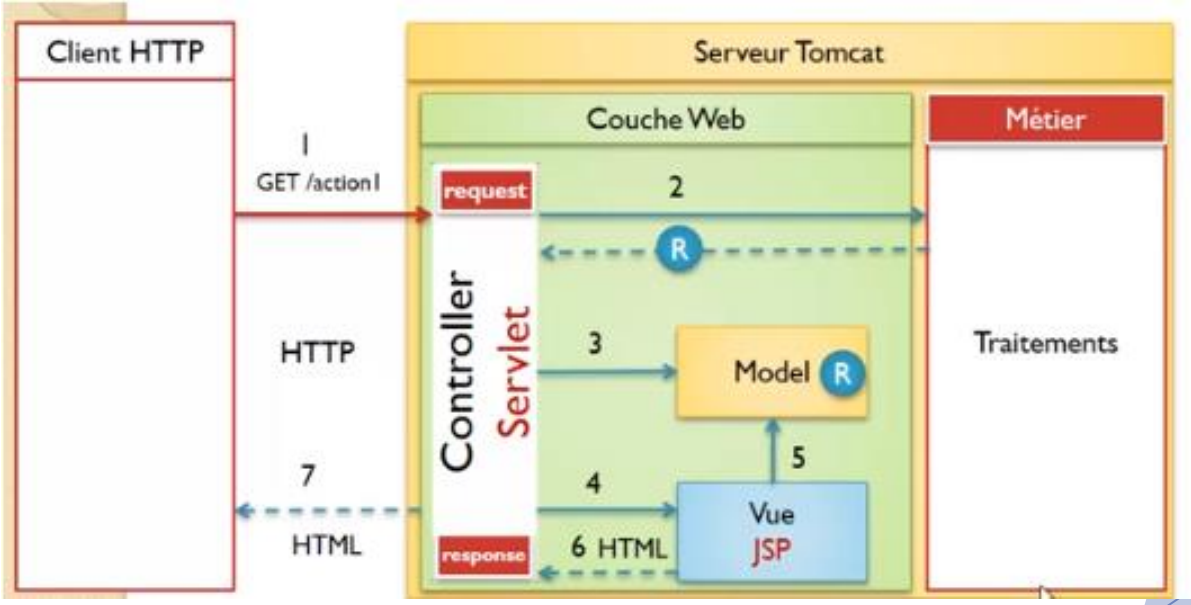
Sonia Kefi

3ème GI

Année Universitaire : 2019-2020

Architecture des applications web

- L'objectif de ce cours est de réaliser le **contrôleur** de l'applications Web en JEE en utilisant les **servlets**.



1 – Le client envoie une requête HTTP de type GET ou POST vers le contrôleur représenté par un composant Web JEE : SERVLET . Pour lire les données de la requête HTTP le contrôleur utilise l'objet **request** de type ***HttpServletRequest***

composants de la plate-forme JEE

- La plate-forme JEE regroupe un ensemble de composants qui facilitent la mise en œuvre d'applications Web en Java, dont les plus utilisés sont :

- **Servlets**

- Les Servlets forment l'un des composants JEE les plus utilisés. Elles permettent de gérer des **requêtes HTTP** et de fournir au client une **réponse HTTP** et forment ainsi la base de la programmation Web JEE.
- Les Servlets s'exécutent toujours dans un moteur de Servlet ou conteneur de Servlet permettant d'établir le lien entre la Servlet et le serveur Web. Dans notre projet, nous utiliserons Apache Tomcat.

Servlets

Retour sur HTTP

- Avant d'étudier le code d'une servlet, nous devons nous pencher un instant sur le fonctionnement du protocole HTTP. Pour le moment, nous avons simplement appris que c'était le langage qu'utilisaient le client et le serveur pour s'échanger des informations. Il nous faudrait idéalement un chapitre entier pour l'étudier en détail, mais nous ne sommes pas là pour ça ! Je vais donc tâcher de faire court…
- Si nous observions d'un peu plus près ce langage, nous remarquerions alors qu'il ne comprend que quelques mots, appelés **méthodes HTTP**. Ce sont les mots qu'utilise le navigateur pour poser des questions au serveur. En effet, nous ne nous intéresserons qu'à trois de ces méthodes : **GET**, **POST** et **HEAD**.

Méthode **GET**

- C'est la méthode utilisée par le client pour récupérer une ressource web du serveur via une URL. Par exemple, lorsque vous tapez **www.polytecsousse.tn** dans la barre d'adresses de votre navigateur, le navigateur envoie une requête **GET** pour récupérer la page correspondant à cette adresse et le serveur la lui renvoie. La même chose se passe lorsque vous cliquez sur un lien.
- Lorsqu'il reçoit une telle demande, le serveur ne fait pas que retourner la ressource demandée, il en profite pour l'accompagner d'informations diverses à son sujet, dans ce qui s'appelle les **en-têtes** ou ***headers*** HTTP, typiquement, on y trouve des informations comme la longueur des données renvoyées ou encore la date d'envoi.
- Enfin, sachez qu'il est possible de transmettre des données au serveur lorsque l'on effectue une requête GET, au travers de paramètres directement placés après l'URL (paramètres nommés *query strings*) ou de cookies placés dans les en-têtes de la requête : nous reviendrons en temps voulu sur ces deux manières de faire. La limite de ce système est que, comme la taille d'une URL est limitée, **on ne peut pas utiliser cette méthode pour envoyer des données volumineuses au serveur**, par exemple un fichier.

Méthode **POST**

- La taille du corps du message d'une requête **POST** n'est pas limitée, c'est donc cette méthode qu'il faut utiliser pour soumettre au serveur des données de tailles variables, ou que l'on sait volumineuses. C'est parfait pour envoyer des fichiers par exemple.
- Toujours selon les recommandations d'usage, cette méthode doit être utilisée pour réaliser les opérations qui ont un effet sur la ressource, et qui ne peuvent par conséquent pas être répétées sans l'autorisation explicite de l'utilisateur. Vous avez probablement déjà reçu de votre navigateur un message d'alerte après avoir actualisé une page web, vous prévenant qu'un rafraîchissement de la page entraînera un renvoi des informations : eh bien c'est simplement parce que la page que vous souhaitez recharger a été récupérée via la méthode **POST**, et que le navigateur vous demande confirmation avant de renvoyer à nouveau la requête.

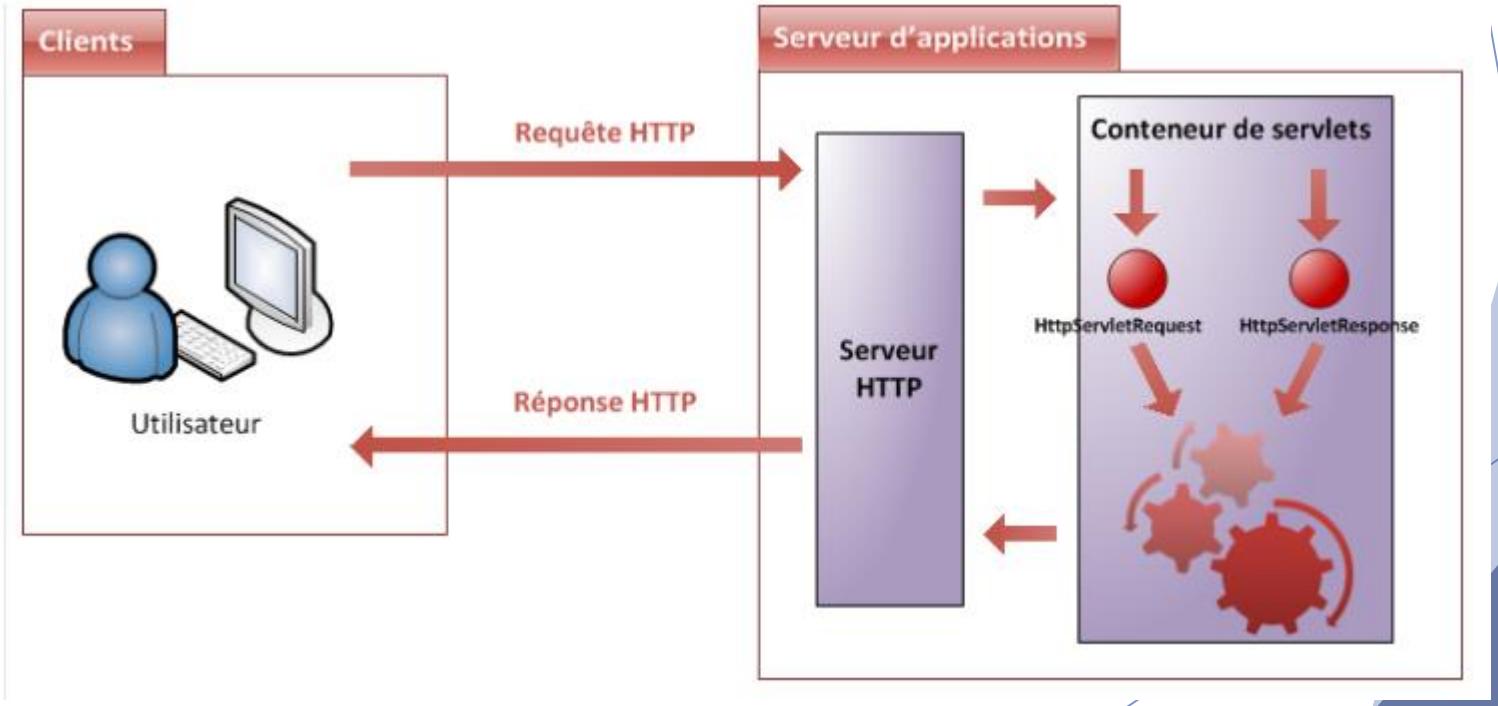
Méthode **HEAD**

- Cette méthode est identique à la méthode GET, à ceci près que le serveur n'y répondra pas en renvoyant la ressource accompagnée des informations la concernant, mais **seulement ces informations**.
- En d'autres termes, il renvoie seulement les en-têtes HTTP ! Il est ainsi possible par exemple de vérifier la validité d'une URL ou de vérifier si le contenu d'une page a changé ou non sans avoir à récupérer la ressource elle-même : il suffit de regarder ce que contiennent les différents champs des en-têtes.

Serveur web

- La requête HTTP part du client et arrive sur le serveur.
- L'élément qui entre en jeu est alors le **serveur HTTP** (on parle également de **serveur web**), qui ne fait qu'écouter les requêtes HTTP sur un certain port, en général le port 80.
- Nous savons déjà qu'il la transmet à un autre élément, que nous avons jusqu'à présent nommé conteneur : il s'agit en réalité d'un **conteneur de servlets**, également nommé **conteneur web** (voir la figure suivante). Celui-ci va alors créer deux nouveaux objets :

- `HttpServletRequest` : cet objet contient la requête HTTP, et donne accès à toutes ses informations, telles que les en-têtes (headers) et le corps de la requête.
- `HttpServletResponse` : cet objet initialise la réponse HTTP qui sera renvoyée au client, et permet de la personnaliser, en initialisant par exemple les en-têtes et le corps (nous verrons comment par la suite).



Conteneur et paire d'objets requête/réponse

- En effet, le conteneur de servlets va les transmettre à votre application, et plus précisément aux **servlets** et filtres que vous avez éventuellement mis en place. Le cheminement de la requête dans votre code commence à peine, et nous devons déjà nous arrêter :

qu'est-ce qu'une servlet ?

Création d'une Servlet

- Une **servlet** est en réalité une simple classe Java, qui a la particularité de **permettre le traitement de requêtes et la personnalisation de réponses**. Pour faire simple, dans la très grande majorité des cas une servlet n'est rien d'autre qu'une classe capable de recevoir une requête HTTP envoyée depuis le navigateur de l'utilisateur, et de lui renvoyer une réponse HTTP.
- En principe, une **servlet** dans son sens générique est capable de gérer n'importe quel type de requête, mais dans les faits il s'agit principalement de

Déploiement d'une servlet

Il existe deux manières pour déployer une servlet sur le serveur Tomcat :

1) On utilise les annotations web, exp :
   @WebServlet("/ServletUpdatePerson")

2)
- Pour que le serveur Tomcat reconnaisse une servlet, celle-ci doit être déclarée dans le fichier web.xml qui se trouve dan le dossier WEB-INF.
- Le fichier web.xml s'appelle le descripteur de déploiement de Servlet.
- Ce descripteur doit déclarer principalement les éléments suivant :
  - Le nom attribué à cette servlet
  - La classe de la servlet
  - Le nom URL à utiliser pour faire appel à cette servlet via le protocole HTTP.

Fin !