Cours sur les JSP et Servlets

Sommaire

Introduction aux concepts de base

- Qu’est-ce qu’un protocole ?
- Qu’est-ce qu’un port ?
- Qu’est-ce qu’HTTP ?
- Qu’est-ce qu’un serveur ?
- Quelle est la différence entre un serveur web et un serveur d’application ?

Qu’est-ce qu’un protocole ?

- Un protocole est un ensemble de règles qui définissent comment les données sont transmises et reçues sur un réseau.
- Les protocoles assurent que les ordinateurs et les réseaux peuvent communiquer efficacement entre eux.
- Les protocoles peuvent définir des détails tels que la manière dont la connexion est établie, le format des données envoyées, la manière dont les erreurs sont détectées et corrigées, etc.
- Des exemples de protocoles couramment utilisés incluent HTTP, HTTPS, FTP, SMTP, TCP/IP, UDP ...

Qu’est-ce qu’un port ?

- Un port est utilisé par le protocole de transport (comme TCP ou UDP) pour distinguer les différentes applications ou processus qui communiquent sur le réseau.
- C’est comme un numéro d’appartement pour les données - il aide le système à savoir où envoyer les données.

Qu’est-ce qu’HTTP ?

- HTTP signifie “Hypertext Transfer Protocol”.
- C’est le protocole utilisé pour transférer des données sur le web.
- Une communication HTTP est faite de requêtes envoyées par le client et de réponses renvoyées par le serveur.
- HTTP utilise plusieurs méthodes, comme GET pour récupérer des informations, POST pour envoyer des informations, PUT pour mettre à jour des informations, et DELETE pour supprimer des informations.

Qu’est-ce qu’un serveur ?

- Un serveur est un ordinateur ou un système qui fournit des ressources, des données, des services, ou des programmes à d’autres ordinateurs, connus sous le nom de clients, sur un réseau.
- En termes plus généraux, on peut aussi parler de “serveur” pour désigner le logiciel qui implémente une telle fonctionnalité, quel que soit le type de machine sur lequel il tourne.
- Les serveurs peuvent être utilisés pour gérer l’accès aux ressources, envoyer des messages, stocker des fichiers, fournir des sites web, etc.

Quelle est la différence entre un serveur web et un serveur d’application ?

- Un serveur web est un serveur qui stocke des pages web et les rend accessibles via le réseau (généralement Internet) aux clients qui en font la demande via leur navigateur web.
- Un serveur d’application est un serveur qui exécute des applications sur le côté serveur et envoie les résultats au client. Ces applications peuvent être très diverses, y compris des applications web, mais elles sont plus complexes et ont plus de fonctionnalités que les pages web statiques servies par un serveur web.
- En général, un serveur d’application peut faire tout ce qu’un serveur web peut faire, mais pas l’inverse.

Introduction aux Servlets et JSP

- Qu’est-ce qu’un Servlet ?
- Qu’est-ce qu’une JSP ?
- Pourquoi utiliser Servlets et JSP ?

Qu’est-ce qu’un Servlet ?

- Un Servlet est une classe Java qui peut gérer les requêtes HTTP sur un serveur web.
- Il est généralement utilisé pour générer des pages web dynamiques en réponse à des requêtes de clients.
- Les Servlets peuvent lire des données envoyées par les utilisateurs (par exemple, des données d’un formulaire), lire les cookies HTTP et les en-têtes HTTP, et envoyer des documents HTML au client.

Qu’est-ce qu’une JSP ?

- JSP signifie JavaServer Pages.
- C’est une technologie qui permet aux développeurs d’écrire du code HTML et du code Java dans le même fichier.
- Les JSP sont compilées en Servlets par le serveur web, ce qui signifie que vous pouvez utiliser toutes les fonctionnalités des Servlets dans vos JSP.
- En général, les JSP sont utilisées pour la présentation et les Servlets pour la logique métier.

Pourquoi utiliser Servlets et JSP ?

- Ils permettent de créer des applications web dynamiques avec Java, un langage qui offre de nombreuses fonctionnalités et une grande flexibilité.
- Ils sont intégrés dans la plateforme Java EE, qui fournit de nombreux outils et services pour les applications web.
- Ils sont supportés par de nombreux serveurs web et d'applications, ce qui facilite le déploiement et la mise à l'échelle de vos applications.

Environnement de développement

- Qu'est-ce qu'Eclipse ?
- Comment configurer Eclipse pour le développement web Java ?
- Qu'est-ce que Tomcat ?
- Comment configurer Tomcat dans Eclipse ?

Qu’est-ce qu’Eclipse ?

- Eclipse est un environnement de développement intégré (IDE) largement utilisé pour le développement en Java.
- Il offre une multitude de fonctionnalités, comme l’achèvement automatique du code, la refactorisation intelligente, et le débogage facile.
- Il a également un support intégré pour Maven, qui est un outil de gestion des dépendances pour Java.

Comment configurer Eclipse pour le développement web Java ?

- Vous pouvez télécharger Eclipse IDE for Enterprise Java and Web Developers sur le site web officiel d'Eclipse.
- Une fois installé, vous pouvez créer un nouveau projet Maven avec support web.

Qu’est-ce que Tomcat ?

- Apache Tomcat est un serveur d’applications Java qui peut exécuter des applications web écrites en Java, y compris les Servlets et les JSP.
- C’est un choix populaire pour le développement et le déploiement d’applications web Java.

Comment configurer Tomcat dans Eclipse ?

- Dans Eclipse, vous pouvez configurer Tomcat comme serveur d’applications pour votre projet.
- Vous devrez d’abord installer Tomcat, puis le configurer dans Eclipse.

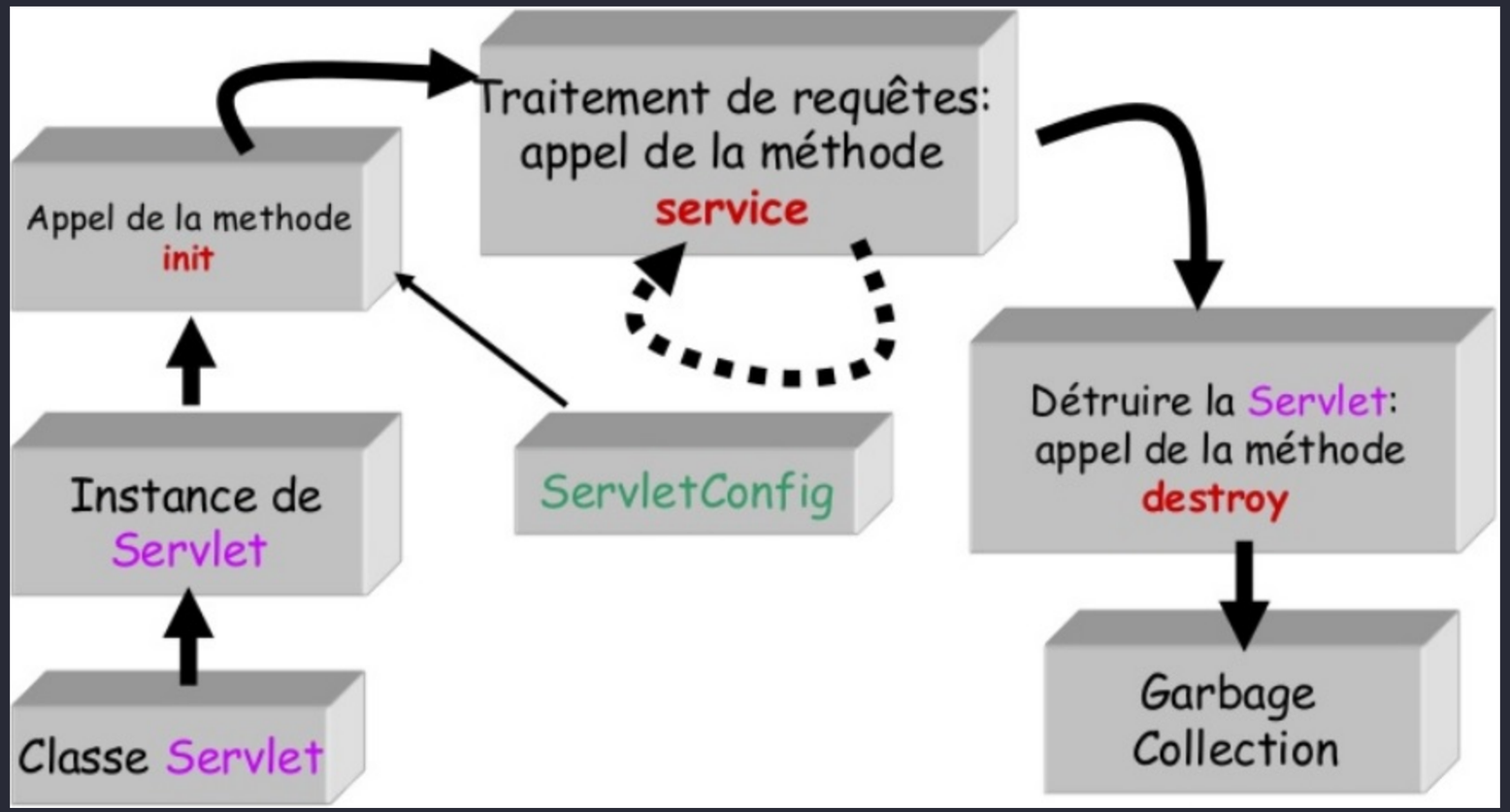
Les Servlets en détail

- Cycle de vie d’un Servlet
- Création et configuration d’un Servlet
- Gestion des requêtes HTTP
- Gestion des sessions
- Gestion des réponses HTTP
- Gestion des cookies

Cycle de vie d'un Servlet

1. Chargement du Servlet : Le serveur web charge le Servlet en mémoire lorsqu'il est nécessaire ou lors du démarrage.
2. Initialisation : Le Servlet est initialisé avec la méthode `init()`. Cette méthode est appelée une seule fois pendant le cycle de vie du Servlet.
3. Traitement des requêtes : Le Servlet traite les requêtes HTTP en utilisant la méthode `service()`. Cette méthode appelle généralement les méthodes `doGet()` ou `doPost()` selon la méthode HTTP utilisée.
4. Destruction : Lorsque le Servlet n'est plus nécessaire ou que le serveur web s'arrête, le Servlet est détruit avec la méthode `destroy()`.

Cycle de vie d'un Servlet -



Création et configuration d'un Servlet

- Créez une classe Java qui hérite de la classe `HttpServlet`.
- Annoter la classe avec `@WebServlet` pour indiquer au serveur qu'il s'agit d'un Servlet.
- Implémentez les méthodes nécessaires pour gérer les requêtes HTTP (par exemple, `doGet()` ou `doPost()`).

Création et configuration d'un Servlet - exemple

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

// @WebServlet déclare que c'est un Servlet et définit son URL d'accès
@WebServlet("/helloServlet")
// Déclaration de la classe MonServlet qui hérite de HttpServlet
public class HelloServlet extends HttpServlet {

    // Cette méthode sera appelée lorsque le serveur reçoit une requête GET pour ce Servlet
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        // Définition du type de contenu de la réponse
        resp.setContentType("text/html");
        // Obtenir l'objet PrintWriter de l'objet HttpServletResponse
```

Annotation @WebServlet

L'annotation `@WebServlet` est utilisée pour déclarer un Servlet en Java. Elle contient plusieurs attributs pour la configuration :

- `name` : Le nom du servlet. Si non fourni, le nom de la classe est utilisé.
- `value` ou `urlPatterns` : Les URL associées au servlet. Peut être une seule URL ou un tableau d'URL.
- `initParams` : Les paramètres d'initialisation du servlet.
- `loadOnStartup` : L'ordre de chargement du servlet lors du démarrage du serveur.

Les méthodes HTTP dans les Servlets

Les Servlets gèrent différentes méthodes HTTP via les méthodes correspondantes dans `HttpServlet` :

- `doGet(...)`: Utilisée pour demander une ressource ou une page Web.
- `doPost(...)`: Utilisée pour envoyer des données à traiter à une ressource spécifique.
- `doPut(...)`: Utilisée pour mettre à jour une ressource existante.
- `doDelete(...)`: Utilisée pour supprimer une ressource existante.
- `doPatch(...)`:Utilisée pour modifier partiellement une ressource existante.

Chaque méthode `doXXX` a la même signature : `(HttpServletRequest req, HttpServletResponse resp)`

Obtenir des informations sur une requête

Pour obtenir les informations sur la requête HTTP, nous utilisons les méthodes suivantes de l'objet HttpServletRequest :

Récupération des paramètres de requête

L'objet `HttpServletRequest` permet de récupérer les paramètres de la requête HTTP. Ces paramètres peuvent provenir de la chaîne de requête dans l'URL (pour les requêtes GET) ou du corps de la requête (pour les requêtes POST, PUT ...).

```
// Récupération du paramètre "nom"
String nom = servletRequest.getParameter("nom");
// ...
```

Utilisation des attributs de requête

Les attributs de requête sont des données qui sont associées à une requête HTTP spécifique. Ils sont stockés dans l'objet `HttpServletRequest` et peuvent être utilisés pour passer des informations entre différentes parties de votre application.

```
// Ajout d'un attribut à la requête
servletRequest.setAttribute("nom", "John Doe");
// Récupération d'un attribut de la requête
String nom = (String) servletRequest.getAttribute("nom");
```

Gestion des sessions

- Les sessions permettent de stocker des informations sur un utilisateur entre plusieurs requêtes HTTP.
- Vous pouvez créer et accéder aux sessions avec l'objet `HttpServletRequest`.
- Utilisez la méthode `getSession()` pour obtenir l'objet `HttpSession` associé à la requête.

```
// Création d'une nouvelle session ou récupération de la session existante
HttpSession session = req.getSession();
```

- Si vous voulez vérifier s'il existe déjà une session sans en créer une nouvelle, vous pouvez utiliser `request.getSession(false)` elle renvoie null s'il n'y a pas de session.

Utilisation des attributs de session

Les attributs de session sont des données qui sont associées à une session utilisateur spécifique. lls sont stockés dans l'objet `HttpSession` et peuvent être utilisés pour passer des informations entre différentes requêtes d'un même utilisateur.

```java
// Obtention de l'objet HttpSession
HttpSession session = req.getSession();
// Ajout d'un attribut à la session
session.setAttribute("nom", "John Doe");
// Récupération d'un attribut de la session
String nom = (String) session.getAttribute("nom");
```

Destruction de la Session

La session est détruite quand :

1. Expiration de la session : Si aucune nouvelle requête n'est reçue du client pendant la durée du timeout, la session est détruite automatiquement.
2. Invalidation de la session : Vous pouvez explicitement détruire une session en utilisant `session.invalidate()`.
3. Arrêt du serveur : Si le serveur s'arrête ou redémarre, toutes les sessions en cours sont détruites.

Gestion des réponses HTTP

- Utilisez l'objet `HttpServletResponse` pour envoyer des réponses HTTP aux clients.
- Vous pouvez définir le code de statut, les en-têtes et le contenu de la réponse.
- Pour générer du contenu HTML, utilisez un `PrintWriter` pour écrire directement dans la réponse.

Gestion des cookies

- Les cookies sont de petits morceaux de données stockés par le navigateur et renvoyés au serveur avec chaque requête.
- Vous pouvez créer des cookies avec la classe `Cookie` et les envoyer au client avec l'objet `HttpServletResponse`.
- Vous pouvez lire les cookies envoyés par le client avec l'objet `HttpServletRequest`.

```java
// Création d'un nouveau cookie
Cookie cookie = new Cookie("nom", "John Doe");
// Envoi du cookie au client
servletResponse.addCookie(cookie);

// Récupération des cookies
Cookie[] cookies = servletRequest.getCookies();
// Parcours des cookies
for (Cookie cookie : cookies)
  if (cookie.getName().equals("nom"))
    String nom = cookie.getValue();
    // ...
```