JAVA ET LE WEB

Licence Devops
JEE
2019/2020

Fayçal BRAÏKI
etudiantspsp@free.fr

Sommaire

I. Rappels sur les notions d'architectures applicatives
II. Java et le web
III. Focus sur les différentes plateformes JAVA
IV. Les servlets
V. Les Java Server Pages (JSP)
VI. Créer des applications web avec Netbeans

Rappels sur les notions d'architectures applicatives

- Les systèmes d'information obéissent à une architecture applicative
- Il existe différents types d'architecture et ceux-ci ont été modelés dans le temps suivant le développement des réseaux et des serveurs à « bas coûts » (virtualisation par ex.), l'augmentation des performances, etc…
- Elles sont structurées en tiers : architectures 1-tiers, 2—tiers, 3-tiers, n-tiers
- Et chacun des *tiers joue un rôle suivant une logique dans le fonctionnement de l'application : logique de présentation, de traitement (ou métier), ou encore d'accès aux données*

Rappels sur les notions d'architectures applicatives

- Les architectures 1-tiers
  - *Les 3 logiques (présentation, traitement, données) sont confondues sur une seule machine*
  - *On parle alors d'informatique centralisée de type mainframe*
  - *Il s'agit historiquement de l'architecture la plus ancienne*
- Les architectures 2-tiers ou client/serveur
  - *La logique d'accès aux données est séparée du reste et est déplacée sur un serveur de données*
  - *Les autres couches (présentation et traitement) se trouvent sur une machine appelée le client (le + souvent un PC)*
    ➔importance des IHM
  - *Le(s) client(s) et le(s) serveur(s) communique(nt) par le biais des réseaux (locaux le + souvent)*

Le cours précédent s'inscrivait dans ce type d'architecture

Rappels sur les notions d'architectures applicatives

- Les architectures 3-tiers
  - *Les 3 couches sont distinguées et se situent chacune sur des machines différentes : client/serveur applicatif/serveur de données*
  - *Avec le développement du web, la couche cliente est implémentée le + souvent par un navigateur web (on parle alors d'« application web »)*
- Les architectures n-tiers
  - *Il s'agit d'une évolution de l'architecture 3-tiers dans laquelle la couche applicative est elle-même divisée en sous-composants nommés services*

Ce cours s'inscrit dans ce type d'architecture (distribuée ou répartie sur le web)

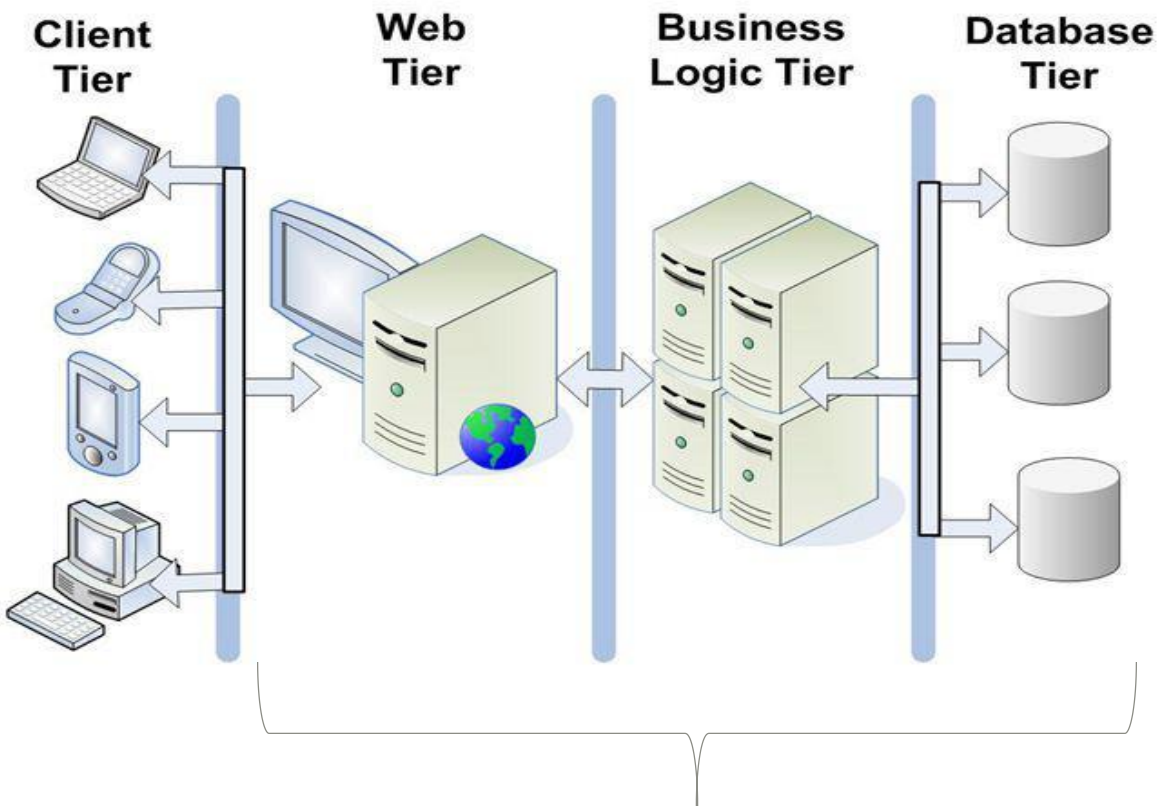
JAVA et le Web

- Java est un langage orienté objet pensé dès sa conception dans une logique de déploiement multi-plateformes par le biais du mécanisme de la Java Virtual Machine (JVM)
- Dans les architectures n-tiers, le fonctionnement de l'application est réparti sur plusieurs serveurs (présentation, application, données) et les communications entre composants sont supportées par un protocole réseau

Ce cours va être centré sur la partie serveur des applications

JAVA et le Web

Les architectures n-tiers (pour le WEB)



Partie serveurs

On associe à chaque couche des machines dédiés :
- Les clients gérent la présentation
- Le serveur WEB a en charge la gestion des requêtes HTTP
- La logique des traitements est assurée par une serveur d'application
- La gestion des données est assurée par un SGBD

Focus sur les différentes plateformes JAVA

- Une plate-forme java fournit au minimum une JVM et des librairies de classes
- Java PlatForm Standard Edition, JSE (anciennement Java 2 Standard Edition) destinée aux applications poste de travail (architectures 2-tiers) et contient les API de base
- JAVA Entreprise Edition, JEE (anciennement J2EE) étend JSE aux applications d'entreprise.
  Elle contient des composants serveurs et en particulier les servlets, un moteur de scripts JSP (Java Server Page), des API de connexion aux base de données (JDBC : Java DataBase Connectivity),….
- JAVA Platform Micro Edition, JME version destinée aux systèmes embarqués (voitures, imprimantes, …) et mobiles

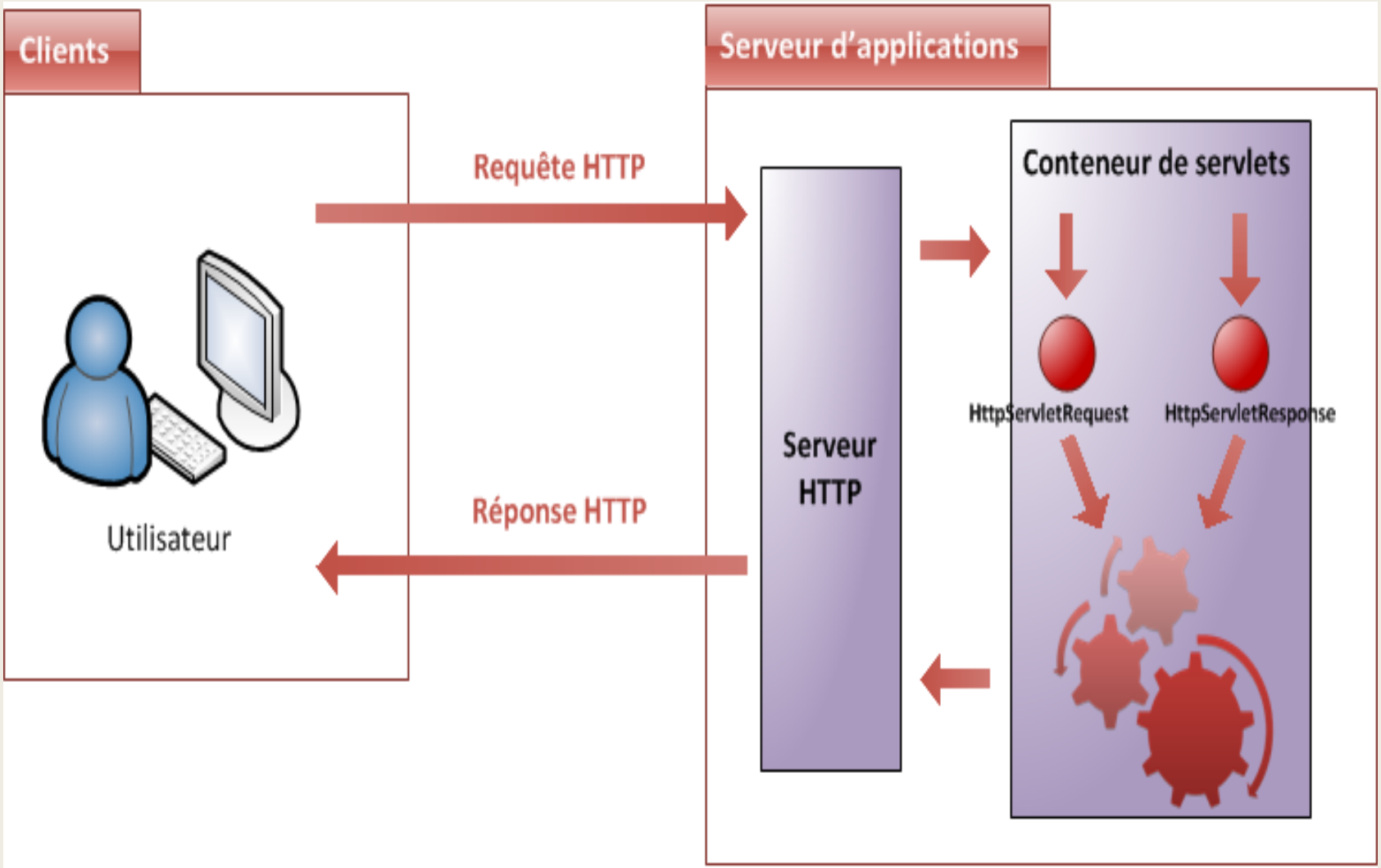
Focus sur les différentes plateformes JAVA

- JEE vise à permettre le développement d'applications :
  - *Portables*
  - *Sécurisées*
  - *Standardisées*
- 2 composants web de JEE vont particulièrement nous intéresser :
  - *Les servlets*
  - *Les JSP*
- Ils permettent tous 2 de générer dynamiquement des pages HTML de manière différente du fait de leur composition :
  - *Les servlets sont écrites en JAVA et génèrent dynamiquement du code HTML*
  - *Les JSP sont écrites en HTML et intègrent du code JAVA*
  - *Attention : NE PAS CONFONDRE AVEC JAVASCRIPT*

Les Servlets

- Composant faisant partie de JEE
- Classe compilée implémentée par le moteur de servlet d'un serveur d'application (GlassFish par ex.)
- Elles exécutent des traitements transmis par le serveur web et renvoient les réponses en générant dynamiquement la page web
- Elles jouent le rôle de contrôleur dans le design pattern MVC

Les Servlets



Les Servlets

- Elles font appel aux packages :
  - *Java.io*
  - *Javax.servlet*
  - *Javax.servlet.http*
- Et héritent de la classe HttpServlet qui permet d'utiliser les méthodes doGet et doPost en fonction de la requête HTML envoyée au serveur (GET ou POST)
- À noter que la méthode doPost n'est utilisable qu'avec un formulaire
- La classe HttpServlet hérite de la classe GenericServlet qui elle-même implémente l'interface Servlet
- La classe Servlet contient des méthodes permettant d'assurer le cycle de vie de la servlet :
  - *Init() //fait office de constructeur*
  - *Service()//assure le traitement des requêtes*
  - *Destroy()//fait office de destructeur*

Les Servlets

- La classe HttpServlet
  - *Fournit une implémentation spécifique au HTTP pour la classe Servlet*
  - *Fournit une surcharge de la méthode service() qui lit la méthode HTTP (GET, POST,…) de la requête et transmet la requête à la méthode appropriée pour traitement (doGet ou doPost)*
  - public void **doGet**(HttpServletRequest req, HttpServletResponse rep)
  - public void **doPost**(HttpServletRequest req, HttpServletResponse rep)
- Les objets de type HttpServletRequest et HttpServletResponse contiennent respectivement le flux de données de la requête et celui de la réponse.
- Ils permettent d'avoir accès à des informations sur la requête : méthode HTTP (GET, POST), adresse IP du requérant, nom de la machine, etc…

Les Servlets

- HttpServletResponse
  - *Classe utilisée pour envoyer une réponse HTTP au client*
- Méthodes liées
  - *void setStatus(int statusCode): définit le code de retour de la réponse*
  - *void setHeader(String name, String value): définit la valeur de l'entête name*
  - *void setContentType(String type): définit le type MIME du contenu*
  - *PrintWriter getWriter(): pour envoyer des données texte au client*
  - *ServletOutputStream getoutputStream(): flux pour envoyer des données binaires*
  - *void sendRedirect(String url): redirige le client vers l'URL*

Les Servlets

- HttpServletRequest
  - *Classe contenant la requête HTTP du client*
- Méthodes liées
  - *String getMethod(): retourne la méthode HTTP*
  - *String getHeader(String name): retourne le paramètre name de l'entête HTTP de la requête*
  - *String getRemoteHost(): retourne le nom d'hôte du client*
  - *String getRemoteAddr(): retourne l'adresse IP du client*
  - *String getParameter(String name): retourne la valeur d'un champ de formulaire*
  - *Enumeration getParameterNames(): retourne tous les noms des paramètres*
  - *String getServerName(): retourne le nom du serveur*
  - *String getServerPort(): retourne le port sur lequel le serveur écoute*

Les Servlets

Exemple de servlet

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

public class MaServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
 response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {

            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet NewServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Hello World ! </h1>");
            out.println("</body>");
            out.println("</html>");
  }

  }

}
```

Les Servlets

Les Java Server Pages (JSP)

- Technologie JAVA permettant de générer dynamiquement des pages web
- Il s'agit d'un équivalent à PHP et ASP
- Les JSP sont plus orientées présentation (le code HTML est + « facile » à gérer que le code JAVA alors que les servlets sont – adaptées à cet usage mais beaucoup + pour les traitements)
- Une JSP est habituellement constituée :
  - *de données et de tags HTML*
  - *de tags JSP*
  - *de scriptlets (code Java intégré à la JSP)*
- Elles portent l'extension .jsp et sont exécutées par le serveur après transformation transparente en servlet

Les Java Server Pages (JSP)

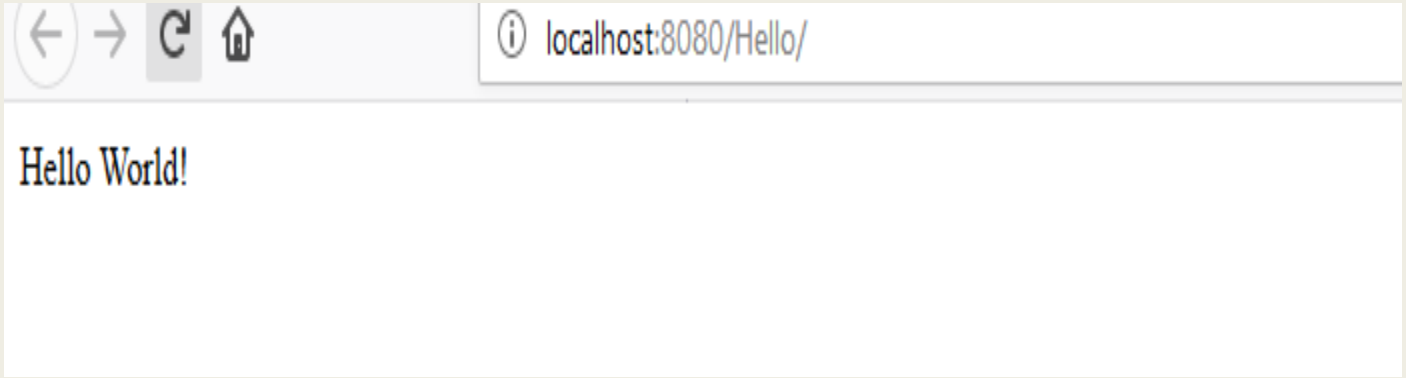
- Les éléments d'une JSP :
  - *Texte HTML : <H1>Le titre</H1>*
  - *Les commentaires HTML : <!-- commentaire HTML -->*
  - *Les commentaires JSP : <%-- commentaire JSP --%>*
  - *Expressions: <%=…%> : une expression est évaluée, transformée en chaîne et incluse dans la page*
  - *Scriplets: <%…%> : fragment de code Java exécuté dans la page*
  - *Déclarations: <%!…%> : déclaration de variables ou de méthodes utilisables dans la page*
  - *Directives: <%@…%> : instruction pour le moteur JSP (page et include)*

Les Java Server Pages (JSP)

- Exemple de JSP :

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
<% out.print("Hello World!");%>
    </body>
</html>
```

Les Java Server Pages (JSP)



Créer des applications web avec Netbeans 8.1

- L'IDE Netbeans intègre tous les éléments nécessaires au développement d'une application web
  - *Librairies JEE*
  - *Serveurs d'application (nous utiliserons GlassFish)*
  - *Base de données*
  - *Etc…*
- Pour développer nos applications web dynamiques, il faudra préparer netbeans à cet usage :
  - *Installation du serveur d'application GlassFish*
  - *Connexion à la base de données test nommée « APP »*

Créer des applications web avec Netbeans 8.1

- *Lancer la console des services*