Technologies Java

Houssem MAHMOUDI
houssem.mahmoudi@ensicaen.fr

Chapitre 4 : Servlets

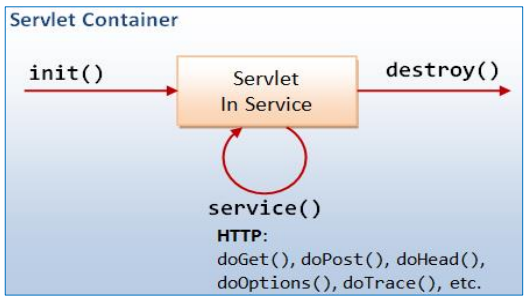
1. Définition
2. Cycle de vie d'une Servlet
3. Création d'une Servlet
4. Fonctionnement d'une Servlet
5. L'objet request
6. L'objet response
7. Les méthodes doGet() & doPost()
8. Interfaces ServletContext & ServletConfig
9. Navigation entre les Servlets
10. Mécanismes de filtres
11. Les sessions

I. Définition

- Une Servlet est une classe Java qui permet de :
  - Traiter les requêtes **HTTP** (Get et Post).
  - Générer des réponses dynamiques (HTML, json).
  - Gérer les sessions utilisateurs.
  - Contrôler le fonctionnement de l'application Web.
- Les servlets fonctionnent dans un **conteneur Web**, comme *Apache Tomcat* ou *WildFly*, qui gère leurs **cycle de vie** et fournit des services comme la gestion des connexions et des threads.

II. Cycle de vie d'une servlet

1. **Initialisation :**
   - La servlet est chargée par le conteneur au moment de son premier appel ou au démarrage du serveur.
   - La méthode **init()** est appelée une fois pour effectuer des initialisations nécessaires.



II. Cycle de vie d'une servlet

2. Traitement des requêtes :

- Pour chaque requête, la méthode suivante sera appelée :

service(**HttpServletRequest** *req*, **HttpServletResponse** *res*)

- Cette méthode délègue le traitement à doGet(), doPost(), ou d'autres méthodes selon le type de requête.

II. Cycle de vie d'une servlet

3. Destruction :

- Lorsque le serveur est arrêté ou que la servlet est retirée, la méthode **destroy()** est appelée pour libérer les ressources.

III. Création d'une servlet

- Pour créer une servlet, il faut suivre les étapes suivantes :
  1. Créer une classe qui étend **HttpServlet**.
  2. Redéfinir les méthodes **doGet()** ou **doPost()** pour traiter les requêtes.
  3. Configurer la Servlet dans le fichier **web.xml** ou via des annotations.
  4. Déployer une Servlet sur un serveur (Conteneur Web).

Remarque : Dans le cours/TP, on va utiliser le serveur Apache Tomcat v9.0 avec Intellij.

III. Création d'une servlet – Application

➔ Créer une Servlet qui permet de retourner le message :

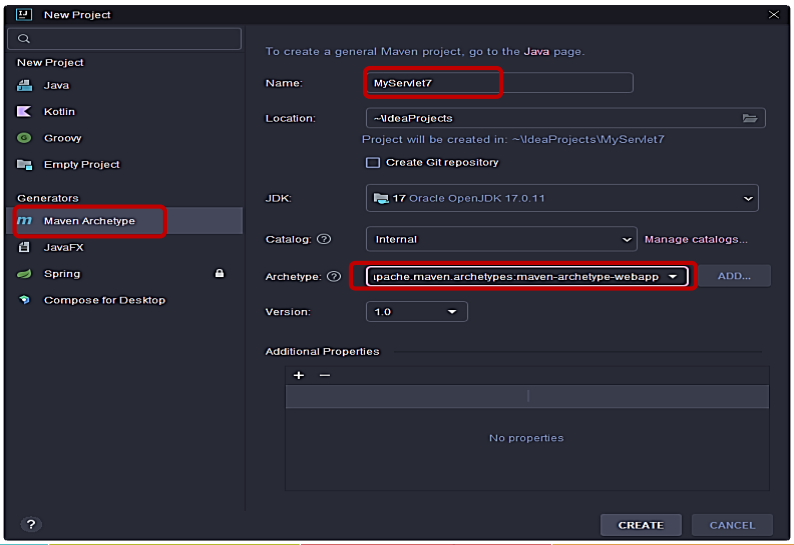
« Bienvenue dans la programmation des Servlets »

- **Prérequis :**
  1. Créer un nouveau projet Java Web
  2. Installer le plugins Tomcat dans Intellij
  3. Télécharger la version 9 Apache Tomcat*
  4. Ajouter le serveur Apache Tomcat dans Intellij
  5. Configurer le serveur avec le projet

*https://tomcat.apache.org/download-90.cgi

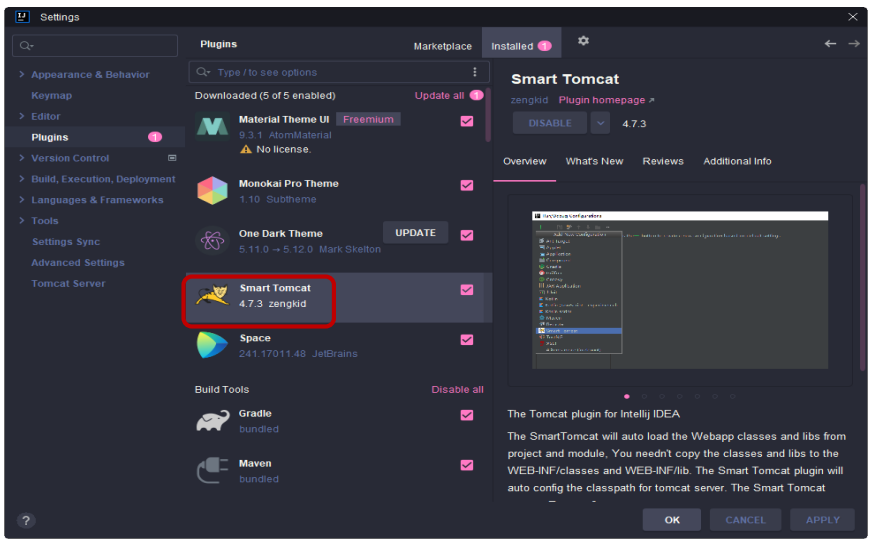
III. Création d'une servlet – Application

1. Créer un nouveau projet :



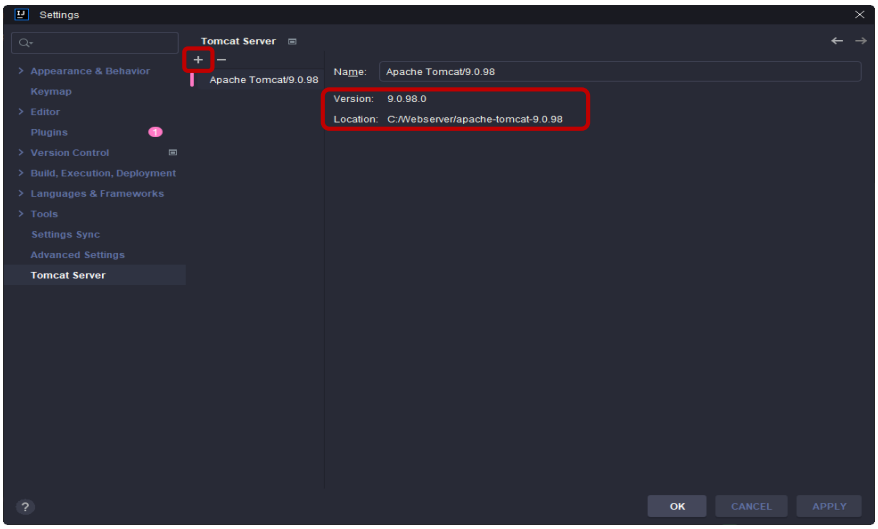
III. Création d'une servlet – Application

2. Installer le plugin Tomcat dans Intellij : File -> Settings -> Plugins



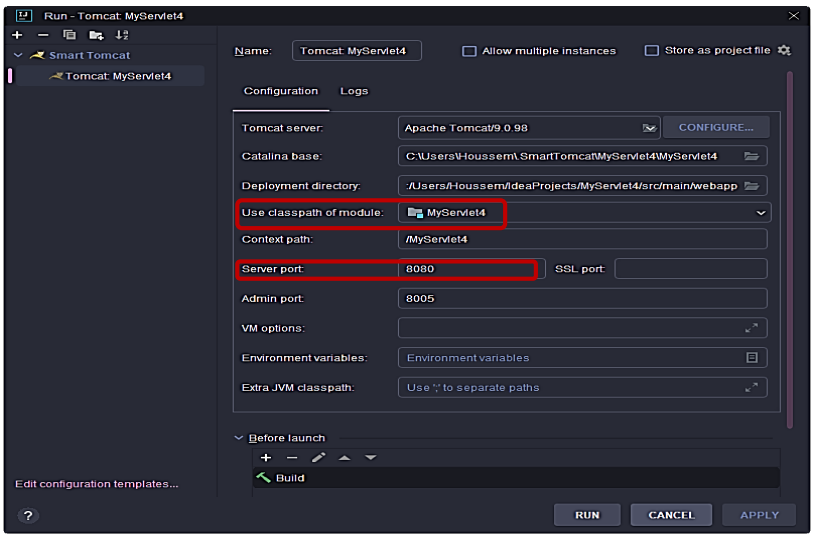
III. Création d'une servlet – Application

3. Ajouter le serveur Apache dans Intellij: File -> Settings -> Tomcat Server



III. Création d'une servlet – Application

4. Configurer le serveur avec le projet : Run -> Run -> Tomcat:nomDuProjet



III. Création d'une servlet – Application

Etapes :

- Créer un dossier intitulé java sous le dossier main
- Créer un nouveau fichier Java
- Ajouter cette dépendance dans le fichier **pom.xml** pour utiliser Javax.servlet.

```
<dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
</dependency>
```

- Reloader la configuration maven

III. Création d'une servlet – Application

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class WelcomeServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<html>");
      out.println("<head>");
      out.println("<title> Bienvenue dans la programmation des Servlets </title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h2> Bienvenue dans notre Servlet");
      out.println("</body>");
      out.println("</html>");
      out.close();
   }
   protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
      doGet(req,res);
   }
```

III. Création d'une servlet – Application

Appel du Servlet depuis une URL :

- Pour utiliser une Servlet, il faut la déclarer dans le fichier « **web.xml** » :

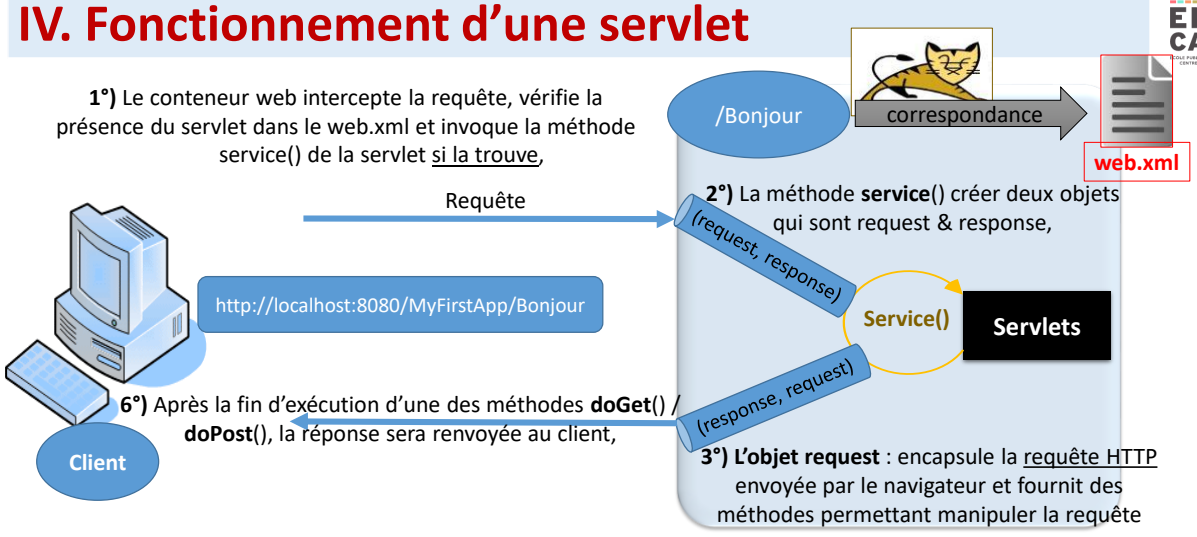
```
<!-- Déclaration de la servlet -->
  <servlet>
    <servlet-name>WelcomeServlet</servlet-name>
    <servlet-class>WelcomeServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>WelcomeServlet</servlet-name>
    <url-pattern>/welcome</url-pattern>
  </servlet-mapping>
```

Nom du Servlet à utiliser depuis l'adresse URL

- On peut utiliser les annotations :
- **@WebServlet**(urlPatterns={"/nomURL"}, name="NomServlet")

IV. Fonctionnement d'une servlet

1°) Le conteneur web intercepte la requête, vérifie la présence du servlet dans le web.xml et invoque la méthode service() de la servlet si la trouve,



2°) La méthode **service**() créer deux objets qui sont request & response,

3°) L'objet request : encapsule la requête HTTP envoyée par le navigateur et fournit des méthodes permettant manipuler la requête

6°) Après la fin d'exécution d'une des méthodes **doGet**() / **doPost**(), la réponse sera renvoyée au client,

Web container

4°) L'objet response : encapsule la réponse HTTP qui sera renvoyer au navigateur et fournit des méthodes pour insérer des données dans la réponse

5°) La méthode **service**() prévoie la nature de la requête HTTP utilisé par le client : Get / Post et appelle doGet() / doPost()

V. L'objet request

- La requête HTTP du client est encapsulée dans un objet « **request** » qui implémente l'interface *HttpServletRequest*. Cet objet contient les données de la requête et les informations sur le client.
- L'objet request permet de gérer les données envoyées avec la requête à la servlet.

Méthodes de récupération d'informations à partir de la requête :

Méthodes	Rôles
String getScheme()	Renvoie le protocole utilisé par la requête (exemple : http, ftp ...)
String getMethod()	Retourne le nom de la méthode HTTP utilisé (Get, POST,…)
String getProtocol()	Renvoie le protocole utilisé par la requête et sa version
String getParameter(‡nom_param‡)	Récupérer la valeur du paramètre passé en URL ou en formulaire

VI. L'objet response

- La réponse de la servlet est encapsulée dans un objet « **response** » qui implémente l'interface *HttpServletResponse*.
- L'objet response permet d'envoyer des données au navigateur du client,

Méthodes de récupération d'information à partir de la réponse :

Méthodes	Rôles
setContentType("arg")	Précise le type de la réponse. Ex "text/html" ; "text/plain" ; "application/pdf".
setHeader("action", "arg")	Renvoie une action au navigateur. Ex : setHeader("Refresh", "5"); setHeader("Refresh", "3;URL=http://www.google.com")
PrintWriter obj = response.getWriter()	Renvoie le protocole utilisé par la requête et sa version
sendRedirect("arg")	Rediriger la réponse au navigateur vers une autre ressource.

VII. Les méthodes doGet() / doPost()

- La méthode doGet() d'une servlet est invoquée lors de la réception d'une requête envoyée par GET, qui la méthode par défaut utilisée par le protocole HTTP. Utilisé pour un envoi des données limitées.
- Lors de l'envoie des données par POST (une formulaire/données binaire), la méthode doPost() sera invoquée et le traitement sera implémenter dedans. Utilisé pour un envoi des données non limitées.
- Il est judicieux que l'une de ces méthodes appelle l'autre pour que la Servlet fonctionne correctement dans les deux cas.

VIII. Interfaces ServletContext & ServletConfig

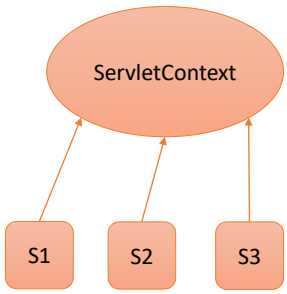
- Un objet de type javax.servlet.**ServletContext** représente les informations de configurations d'une application web globale, ces informations sont accessibles par toutes les Servlets.

1°) Déclaration dans le fichier web.xml :

```
<context-param>
    <param-name>variable_application</param-name>
    <param-value>valeur</param-value>
</context-param>
```

2°) Récupération de la valeur depuis la Servlet :

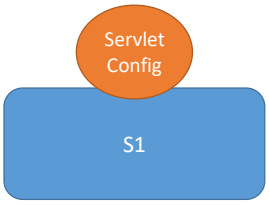
```
ServletContext sc = getServletContext();
String ch = sc.getInitParameter("variable");
```



Les deux méthodes : sc.getInitParameter("arg") et sc. setInitParameter("arg", object) permettent respectivement de récupérer et d'ajouter des objets dans le contexte globale de l'application par programmation.

VIII. Interfaces ServletContext & ServletConfig

- Un objet de type javax.servlet.**ServletConfig** représente les informations propre à une Servlet.



1°) Déclaration dans le fichier web.xml :

```
<servlet>
    <servlet-name>ok</servlet-name>
    <servlet-class>com.sources.Accepter</servlet-class>
    <init-param>
        <param-name>variable_locale</param-name>
        <param-value>valeur</param-value>
    </init-param>
</servlet>
```

2°) Récupération de la valeur depuis la Servlet :

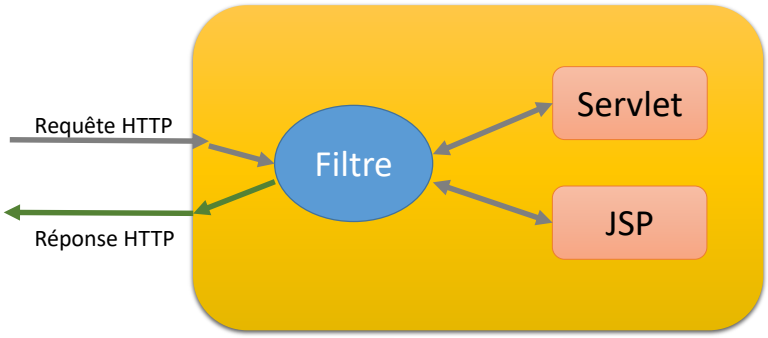
```
ServletConfig scf = getServletConfig();
String ch = sc.getInitParameter("variable");
```


IX. Navigation entre les pages

- **sendRedirect()** : permet d'envoyer une réponse au client en indiquant le nom de la ressource.
- Exemple :
  - response.senRedirect("nom_de_la_ressource") ;
- Inconvénient : les paramètres de la requête sont perdus.
- **RequestDispatcher** : permet de rediriger la requête vers une autre ressource (servlet ou page jsp).
- Exemple :
  - RequestDispatcher rd = request.getRequestDispatcher("/ressource");
  - rd.forward(request, response);
  - Ou on peut aussi utiliser :
  - rd.include(request, response);

X. Mécanismes de filtres

- Les filtres permettent d'intercepter et de traiter :
  - Les requêtes envoyées par les clients,
  - Les réponses générées par les servlets.



X. Mécanismes de filtres

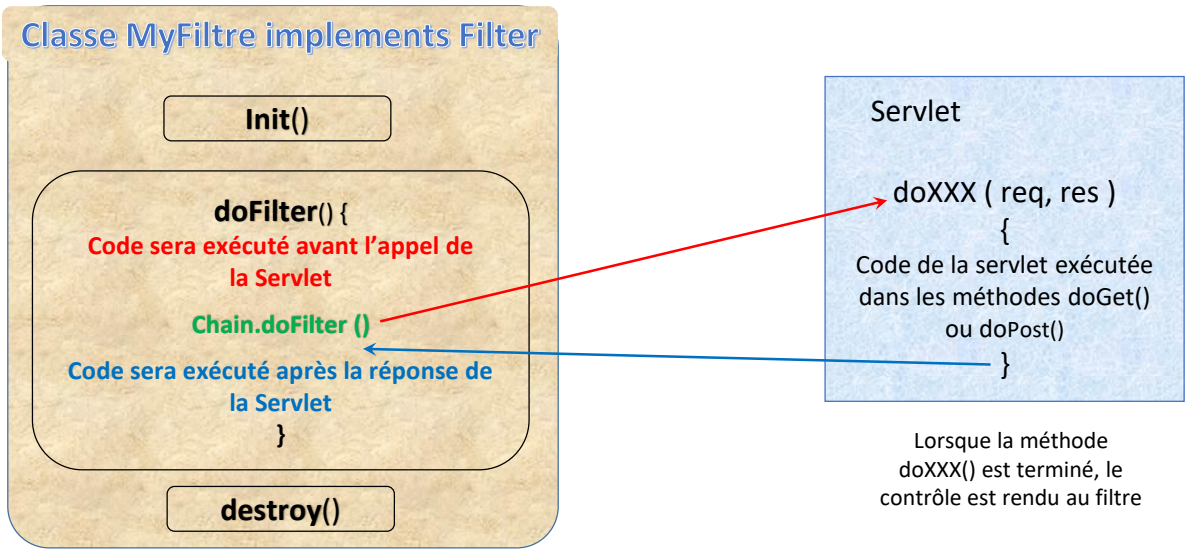
Exemple pratique des filtres :

- Décrypter les données d'une requête aux servlets,
- Crypter les réponses aux clients,
- Gérer l'authentification des clients (admin pour les Servlets protégées),
- Convertir les formats des images / vidéos,

Utilisation d'un filtre :

- Réaliser une classe qui implémente l'interface Filter.
- Modifier le descripteur de déploiement (web.xml) pour indiquer au conteneur d'utiliser le filtre.

X. Mécanismes de filtres



X. Mécanismes de filtres

Déclaration d'un filtre : La déclaration d'un filtre se fait dans le fichier web.xml

```
<filter>
        <filter-name>nom_filtre</filter-name>
        <filter-class>nom_complet_du_filtre</filter-class>
</filter>
<filter-mapping>
        <filter-name>nom_filtre</filter-name>
        <servlet-name>nom_complet_du_Servlet_à_filtrer</servlet-name>
</filter-mapping>
```

- Filtre appliquer à toute l'application web :
  <servlet-name>/*</servlet-name>
- Filtre appliquer à un dossier particulier :
  <servlet-name>/protected/*</servlet-name>

XI. Les sessions

- Le protocole HTTP est un protocole sans état (stateless),il ne conserve aucune information entre les transactions.
- Pour suivre l'activité des clients dans une application web, il faut utiliser les sessions.
- Une session permet d'identifier un utilisateur tout au long de sa navigation dans l'application.
- Une session permet de stocker des chaînes de caractères ainsi que des objets

XI. Les sessions

Création d'une session « HttpSession » :

- L'interface « javax.servlet.http.HttpServletRequest » définit deux méthodes qui permettent de créer une session HTTP :
  - La méthode getSession(); : retourne la session courante, sinon null.
  - La méthode getSession(true); : retourne la session courante, sinon elle crée une nouvelle.

XI. Les sessions

Travailler avec une session :

- L'interface « javax.servlet.http.HttpSession » définit des méthodes pour manipuler les sessions :
  - setAttribute(nom,objet) : permet de stocker un objet identifié par nom, dans la session.
  - getAttribute(nom) : permet de récupérer, à partir de la session, la valeur de l'attribut.
  - removeAttribute(nom) : permet de supprimer un attribut de la session.
  - invalidate() : permet de détruire la session courante et l'ensemble de ses attributs.
  - isNew() : permet de savoir si la session est nouvelle ou non.