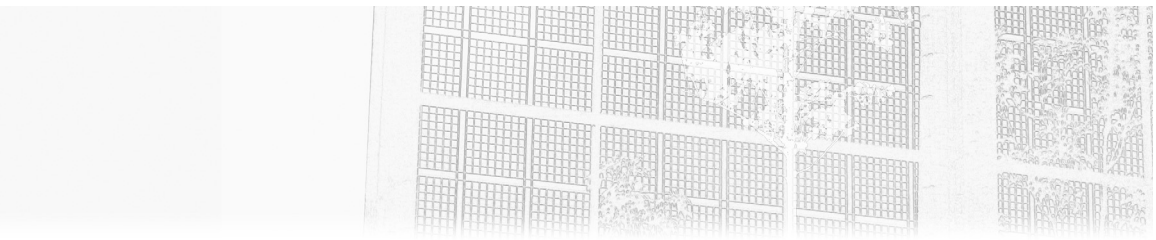


Chapitre 3
La présentation avec les JSP

1. Introduction

Les servlets ne sont pas adaptées pour gérer efficacement l'affichage comme vous avez pu le constater dans le chapitre précédent. La plateforme Jakarta EE propose une solution nommée **JSP** (*Jakarta Server Pages*). Cette technologie permet de créer facilement un contenu dynamique au format HTML ou XML. Elle correspond au V (vue) de l'architecture MVC. La servlet s'occupe de faire le traitement métier et lorsque celui-ci est terminé, elle délègue l'affichage à une JSP. La suite du chapitre prendra comme exemple un contenu HTML.

Les JSP sont, tout simplement, des pages HTML d'extension `.jsp` dans lesquelles il est possible d'ajouter différents types de contenus (non HTML) qui seront traités par le conteneur de servlets pour générer un rendu spécifique lié au contexte d'exécution de la requête. Ces types de contenus peuvent être :

- des scripts sous la forme de code Java,
- des scripts sous la forme d'EL (*Jakarta Expression Language*),
- des actions standards,
- des tags standards (JSTL - *Jakarta Standard Tag Library*),
- ou des tags personnalisés.

Le chapitre explique le principe de fonctionnement des JSP puis présente les différents types de contenus listés précédemment.

Différentes technologies sont abordées dans ce chapitre.

Tout d'abord les JSP. La version actuelle est la **3.0**. Vous pouvez consulter la spécification à l'adresse suivante :
https://jakarta.ee/specifications/pages/3.0/jakarta-server-pages-spec-3.0.pdf

Ensuite, l'EL (*Jakarta Expression Language*) est dans sa version 4.0. La spécification est consultable à l'adresse suivante :
https://jakarta.ee/specifications/expression-language/4.0/jakarta-expression-language-spec-4.0.pdf

Pour finir, la JSTL (*Jakarta Standard Tag Library*), dans sa version 2.0, dont la spécification est consultable à l'adresse suivante :
https://jakarta.ee/specifications/tags/2.0/jakarta-tags-spec-2.0.pdf

Aujourd'hui, la technologie des JSP n'est plus le standard promu par Oracle pour le développement des interfaces mais cette technologie est encore largement utilisée (et pour longtemps). Oracle préconise l'utilisation de la technologie JSF. Par ailleurs, l'avènement des services web et des frameworks JavaScript propose une autre alternative pour développer des applications. Les chapitres suivants aborderont ces autres possibilités.

2. Le projet

2.1 La création du projet

La suite du chapitre se base sur un projet exemple nommé **Projet_JSP**.

- Commencez par créer un projet de type Gradle Project avec les mêmes caractéristiques que dans le premier chapitre.
- Complétez l'arborescence de répertoires pour qu'elle ressemble à cela :

- Ajoutez un fichier nommé `web.xml` dans le répertoire `src/main/webapp/WEB-INF` avec un contenu équivalent au projet initial (*PremierProjetWeb*). Veillez simplement à renommer le nom du projet au sein de la balise `<display-name>`.
- Modifiez le fichier `build.gradle` avec le contenu suivant (observez la section `dependencies` permettant de référencer l'API des servlets et l'API des JSP) :

```
plugins {
    id 'java'
    id 'war'
}

repositories {
    jcenter()
}


dependencies {
    compileOnly "jakarta.servlet:jakarta.servlet-api:5.0.0"
compileOnly group: 'jakarta.servlet.jsp', name:
'jakarta.servlet.jsp-api', version: '3.0.0'
}
```

- Sélectionnez le menu contextuel **Gradle - Refresh Gradle Project** de votre projet pour télécharger les dépendances.

Le projet est prêt pour manipuler les JSP.

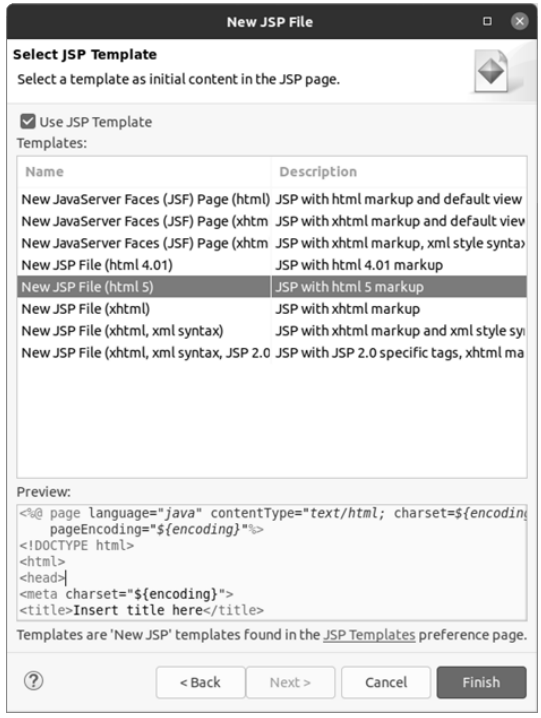
2.2 La création d'une JSP

Pour créer une JSP, suivez les étapes suivantes :

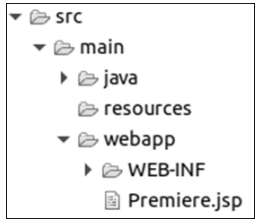
- Faites un clic droit sur le répertoire `webapp` de votre projet puis cliquez sur le menu **New** - **JSP File**.

L'écran suivant apparaît :

- Donnez un nom à votre JSP (`Premiere.jsp`) dans l'exemple et cliquez sur **Next** afin de pouvoir sélectionner le template à utiliser pour la création :



- Sélectionnez le template **New JSP File (html 5)** et cliquez sur **Finish**, votre première page JSP est créée et opérationnelle :



Le contenu de la JSP est le suivant :

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

C'est une page HTML classique mis à part la première ligne. C'est ce que l'on appelle une directive. Cette ligne permet de définir certaines caractéristiques de la JSP. Pour plus d'informations, veuillez vous référer à la section Les directives.

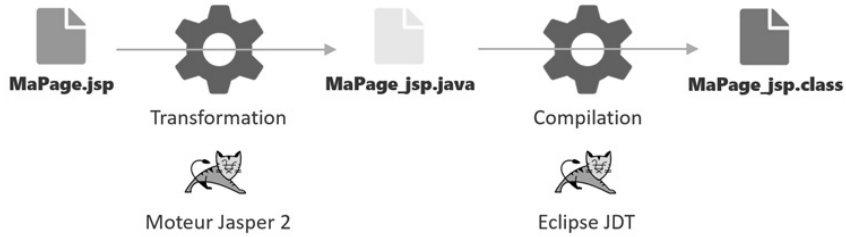
Cette page peut être appelée au travers de l'URL suivante :
http://localhost:8080/Projet_JSP/PremiereJSP.jsp

Le résultat n'est pas spectaculaire car aucun contenu dynamique n'est présent. Cependant, en affichant le code source de la page HTML sur le navigateur, vous pouvez noter l'absence de la directive ce qui indique que le fichier a été traité sur le serveur avant d'être remis au navigateur. La section suivante explique ce traitement.

3. Le principe d'exécution

Les JSP existent pour simplifier la création d'un contenu dynamique car les servlets ne sont pas adaptées pour cette tâche. Lorsqu'une requête HTTP implique l'exécution d'une JSP, voici les actions qui sont déclenchées :

– Si la JSP n'a encore jamais servi, celle-ci est transformée en classe Java (*Translation phase*) avant d'être compilée. La transformation est réalisée par le moteur **Jasper 2** et la compilation est réalisée par défaut par le compilateur Java **Eclipse JDT** (et non pas `javac`). Pour plus d'informations, veuillez vous référer à la documentation officielle à l'adresse suivante :
https://tomcat.apache.org/tomcat-10.0-doc/jasper-howto.html



Cette classe doit implémenter l'interface `jakarta.servlet.Servlet`. Dans l'environnement Tomcat, elle dérive indirectement de la classe `jakarta.servlet.http.HttpServlet`. Une JSP n'est donc ni plus ni moins qu'une servlet. La classe doit aussi implémenter l'interface `jakarta.servlet.jsp.HttpJspPage`.

Cette interface force l'écriture de la méthode `_jspService(...)`. Cette méthode est l'équivalent des méthodes `doXXX(...)` des servlets. Elle prend en paramètre un objet de type `HttpServletRequest` et un objet de type `HttpServletResponse`. Elle a pour rôle la création d'une réponse à l'utilisateur.

Ensuite, la méthode `service(...)` de cette nouvelle classe est appelée. Elle appellera la méthode `_jspService(...)`.

Le cycle de vie d'une JSP est le même que celui d'une servlet. La seule différence est le nom des méthodes pour l'initialisation et la destruction. Elles s'appellent `jspInit()` et `jspDestroy()`. Il est possible de les redéfinir dans la page JSP si nécessaire. Pour plus de détails, veuillez vous référer au chapitre traitant des servlets et à la section Le paramétrage d'une JSP un peu plus loin dans ce chapitre.

Pour bien comprendre cette étape, voici un extrait de la classe Java générée pour la JSP `premiereJSP.jsp`. La classe s'appelle `Premiere_jsp` et dérive de la classe `org.apache.jasper.runtime.HttpJspBase` qui dérive de la classe `HttpServlet` et qui implémente l'interface `HttpJspPage`.

```
/*
 * Generated by the Jasper component of Apache Tomcat
 * Version: Apache Tomcat/10.0.7
 * Generated at: 2021-09-06 19:7:26 UTC
 * Note: The last modified time of this file was set to
 *       the last modified time of the source file after
 *       generation to assist with modification tracking.
 */
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class Premiere_jsp
```