Editions ENI

Java EE
Concevez et développez une application web responsive

Collection
Expert IT

Extrait

Chapitre 5
Les frameworks JSF et Struts

1. Présentation

L'utilisation des servlets et des JSP dans un projet d'envergure n'est pas chose aisée. Il est préférable d'utiliser un framework proposant un cadre d'architecture bien défini et un ensemble de composants limitant le code redondant et fastidieux. Le présent chapitre a pour objectif de découvrir les rudiments des frameworks MVC **JSF 2.2** (*JavaServer Faces*) et **Struts 2.5**. Ces deux frameworks ont été retenus car le premier est un framework orienté composants alors que le second est un framework web orienté actions.

Un framework orienté composants est un framework manipulant des composants visuels et métiers en limitant au maximum l'usage direct des technologies web. Dans ce sens, ce type de framework s'approche du développement d'une application client/serveur. Les développeurs novices en développement web peuvent donc obtenir des résultats rapidement.

Un framework orienté actions est un framework manipulant des requêtes et des réponses HTTP. Un framework de ce type est beaucoup plus proche du protocole HTTP et des technologies satellites. Il se base clairement sur les technologies présentées dans les chapitres précédents.

Ces deux frameworks respectent le pattern MVC (modèle-vue-contrôleur). Ces frameworks proposent donc une architecture permettant la séparation entre le modèle, la vue et le contrôleur.

2. JSF

2.1 Présentation

2.1.1 Généralités

JSF est une technologie Java décrite par la **JSR 344** dans sa version 2.2. La documentation officielle est disponible à l'adresse suivante :
https://www.jcp.org/en/jsr/detail?id=344

L'implémentation de référence, nommée **Mojarra 2.2**, est disponible à l'adresse suivante : https://javaserverfaces.java.net.

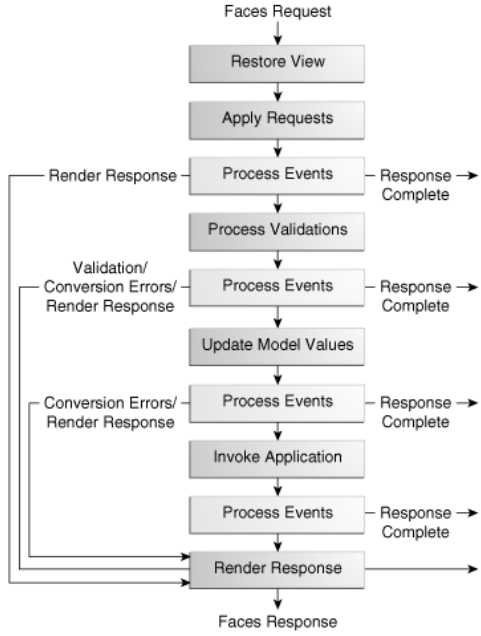
JSF est constitué des éléments suivants :

- Une API permettant de représenter les composants et de gérer leur état. Ces composants sont nommés ***managedBeans*** (on parle aussi de *backingBeans*).
- Une API permettant de gérer les événements, les validations côté serveur, la conversion des données, la navigation, l'internationalisation.
- Des librairies de tags à l'image de ceux disponibles pour les JSP (les balises JSTL) afin de proposer un rendu adapté des *managedBeans*. Les pages JSF sont appelées facelets.

2.1.2 Principes de fonctionnement

Le fonctionnement repose sur un ensemble d'étapes (cycle de vie) entre l'arrivée de la requête HTTP jusqu'à la restitution de la réponse.

La documentation officielle disponible à l'adresse suivante : http://docs.oracle.com/javaee/7/tutorial/jsf-intro006.htm, expose le schéma suivant :



Le cycle de vie est composé de six étapes entre lesquelles s'intercalent des phases de gestion d'événements (*Process Events*) non abordées dans cet ouvrage :

– ***Restore View*** : c'est la première étape exécutée lorsqu'une requête vers une ressource JSF est exécutée. Cette étape permet de reconstituer en mémoire un arbre de composants (*components tree*) aussi appelé vue (*view*) représentant la page et les éléments manipulés. Lorsque la page est accédée pour la première fois, cet arbre est tout simplement créé.

– ***Apply Requests*** : cette seconde étape consiste en la lecture des paramètres de la requête pour mettre à jour la vue avec les informations saisies par l'utilisateur.

- ***Process Validations*** : cette troisième étape permet la conversion et la validation des informations. La conversion permet de passer d'une chaîne de caractères saisie par l'utilisateur en un type adapté au contexte de la requête. Ce type peut être classique comme un numérique, une date... mais ce type peut être plus complexe comme un objet de type `Sport`, `Terrain`... Dans ce cas de figure, l'utilisation d'un convertisseur (*converter*) personnalisé est nécessaire. Lorsque la conversion s'est déroulée avec succès, la validation est réalisée. La validation passe par l'utilisation de validateurs (*validator*) standards ou personnalisés. Une section dédiée pour chacune de ces deux activités est disponible plus loin dans le chapitre.
- ***Update Model Values*** : cette quatrième étape permet la mise à jour du modèle à partir de la vue. Dans le cadre d'un formulaire de saisie d'un nouveau sport, si les saisies respectent les règles métiers, alors les propriétés d'un objet de type `Sport` sont valorisées avec les informations saisies par l'utilisateur. Cet objet est situé dans une classe Java que l'on appelle un *managedBean*. Une section dédiée aux *managedBeans* est disponible plus loin dans le chapitre.
- ***Invoke Application*** : cette cinquième étape est cruciale car elle permet de déclencher le traitement métier en lien avec la requête HTTP. Ce traitement est écrit dans un *managedBean*. Le *managedBean* fait office de contrôleur.
- ***Render Response*** : cette sixième et dernière étape permet d'établir le rendu correspondant à la réponse à partir de la vue. Ce rendu inclut notamment les éventuels messages d'erreurs générés par les étapes précédentes.

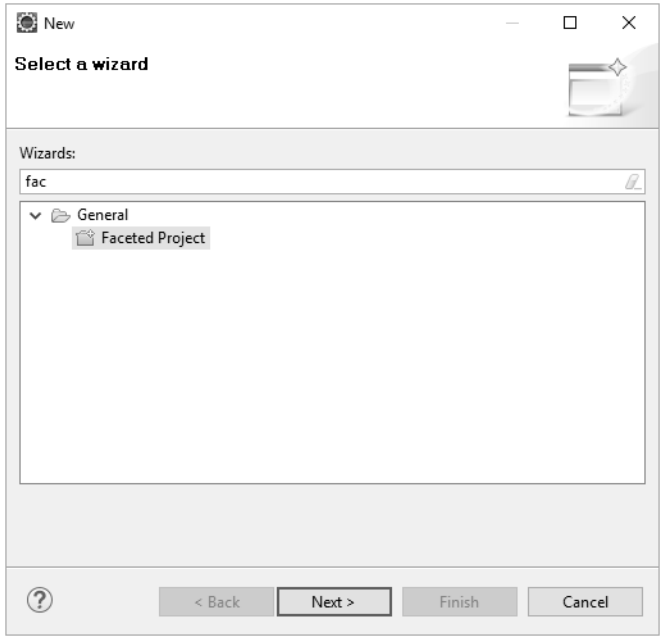
Lors de la première requête vers une ressource, seules les étapes *Restore View et Render Response* sont exécutées.

Ces différentes étapes sont pilotées par une servlet servant de point d'entrée aux différentes requêtes. Cette servlet se nomme `javax.faces.webapp.FacesServlet`. C'est la seule servlet dans un projet JSF. Elle fait office de *Front Controller* car son rôle est de réceptionner les requêtes HTTP entrantes, de les exploiter et de restituer une réponse adaptée. Les sections qui suivent ont pour objectif de présenter les éléments principaux pour débuter avec JSF.

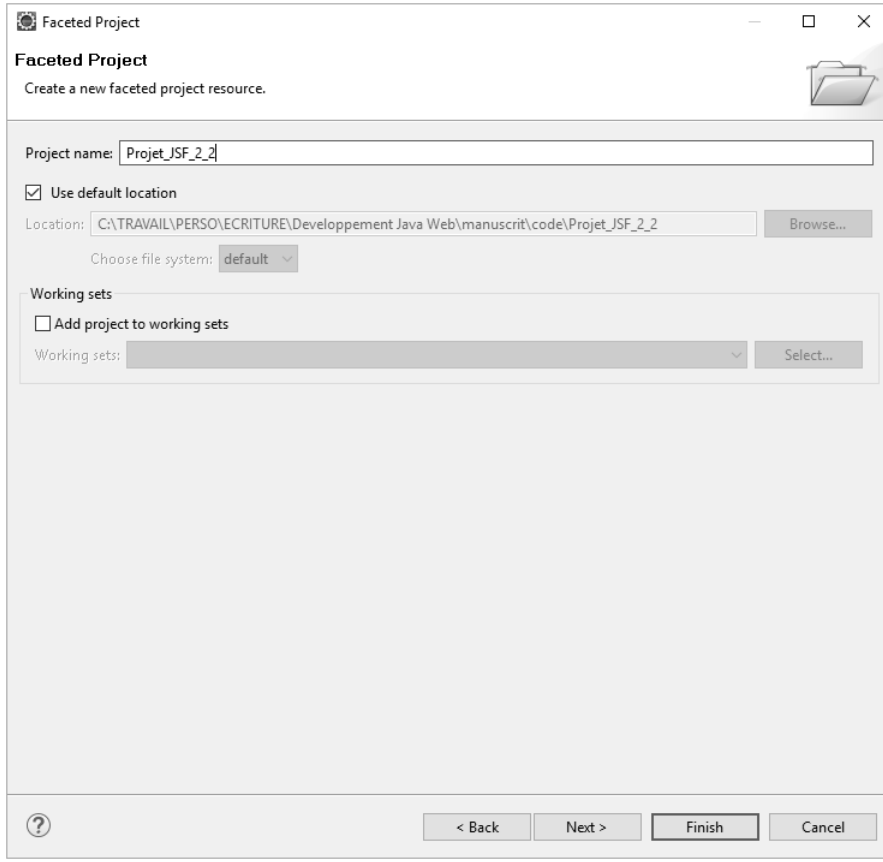
2.2 Le projet

L'objectif de ce chapitre est d'appréhender les rouages de base de JSF. La création d'un projet de type **Faceted Project** facilite la mise en place. Au final, c'est un projet de type **Dynamic Web Project** qui est créé mais préparamétré pour l'utilisation de JSF. Voici les étapes à suivre pour sa mise en place :

- Commencez par cliquer sur le menu **File** - **New** - **Other**. L'écran suivant apparaît dans lequel vous appliquez un filtre pour sélectionner **Faceted Project**.



Cliquez sur le bouton **Next** pour obtenir l'écran suivant dans lequel vous donnez un nom à votre projet. Dans l'exemple, le projet se nomme **Projet_JSF_2_2** :



Editions ENI

Java EE

Développez des applications web en Java

(Nouvelle édition)

Collection
Epsilon

Extrait

Chapitre 5
Conception d'une application Java EE

1. HTML et HTML5

1.1 Les bases HTML

Pour concevoir une application Java web responsive, il faut tout d'abord maîtriser le langage HTML et connaître son fonctionnement.

1.1.1 Introduction à HTML

HTML (*HyperText Mark-up Language*) est un langage de balises qui permet de structurer les pages web. Les balises décrivent la façon dont les documents et leurs divers éléments devront être affichés.

Les balises sont interprétées par le navigateur afin de définir comment afficher la page pour l'utilisateur.

Tout document ou page HTML comporte une balise `<html>` et une balise HTML fermante `</html>`. Il s'agit de l'élément racine du document.

Un document HTML5 doit a minima suivre la structure minimale requise ci-dessous :

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Main Title</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js"></script>
</head>
<body>
  ...
  <!-- other html element -->
  ...
</body>
</html>
```

La balise `<!doctype html>` spécifie qu'il s'agit d'un document HTML et plus précisément, d'un document HTML5.

1.1.2 Les principales balises

Les principales balises relatives à la structure de base d'une page HTML sont :

- `!doctype`, qui indique les règles de mise en forme à appliquer par le navigateur.
- `html`, qui représente l'élément racine de la page.
- `head`, pour les informations d'en-tête utilisées soit par le navigateur (titre, balises meta…), soit par les éléments du corps (styles, scripts...).
- `title`, qui correspond au titre de la page affichée dans le navigateur.
- `meta`, pour les informations optionnelles relatives au descriptif du contenu de la page, à son auteur et de nombreuses autres informations.
- `link`, qui correspond au lien vers des fichiers externes. La feuille de style CSS est référencée de cette façon.
- `script`, qui contient du code JavaScript utilisé par la page (fonctions appelées lors de certains événements déclenchés par l'utilisateur, par exemple, le clic sur un bouton).

– `body`, qui correspond au corps de la page contenant tous les éléments relatifs au rendu et à la présentation du document.

Voici quelques balises parmi les plus utilisées et contenues entre les balises `<body>` et `</body>` :

– La balise `<div></div>` peut être utilisée pour diviser un document en plusieurs sections. Elle peut contenir plusieurs éléments HTML.
– La balise `<a></a>` permet de créer un lien hypertexte.
– Les balises de type `<h1></h1>`, `<h2></h2>` jusqu'à `<h6></h6>` servent à introduire le titre d'un contenu HTML.
– La balise `<br></br>` effectue un retour à la ligne.
– La balise `<p></p>` définit un paragraphe.
– La balise `<!-- -->` réalise la mise en commentaire afin que le HTML ne soit pas interprété par le navigateur.

Voici un exemple d'utilisation des balises `div`, `h1` et `p` :

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Main Title</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js"></script>
</head>
<body>
  <div id="myId" Class="myClass">
    <h1>div element</h1>
    <p> First paragraph</p>
    <p> Second paragraph</p>
    <p> Third paragraph</p>
  </div>
</body>
</html>
```

1.2 HTML5 : les apports

Les apports de HTML5 par rapport à HTML sont nombreux : du point de vue de la structuration des documents qui est plus cohérente, du point de vue de l'ergonomie avec de nouveaux éléments multimédias qui, graphiquement, offrent des possibilités se rapprochant de Flash ou de Silverlight, du point de vue du stockage des données avec le mode persistant hors connexion, et enfin du point de vue des nouvelles interfaces de programmation qui permettent d'utiliser nativement la géolocalisation et de glisser-déposer des objets.

1.2.1 Nouvelles balises pour structurer un document

HTML5 apporte de nouvelles balises qui permettent de structurer un document de façon plus cohérente et plus lisible.

En effet, auparavant, pour subdiviser un document, on utilisait principalement des balises . HTML5 apporte en plus les nouvelles balises suivantes qui vont permettre de mieux structurer un document :

- La balise `<header>` se place en début de page. Elle correspond à l'en-tête de la page. Cette balise permet de structurer un document avec une meilleure sémantique pour les en-têtes de sections ou de pages.
- La balise `<nav>` correspond à une section possédant des liens de navigation principaux (généralement, elle désigne le menu principal de l'application).
- La balise `<section>` correspond à une section générique qui regroupe des sections traitant d'un même sujet.
- La balise `<article>` correspond à une section avec un contenu indépendant relative au traitement d'un sujet spécifique. Cette balise peut contenir une section d'en-tête, un pied de page et différentes sections.
- La balise `<aside>` correspond à une section dont le contenu apporte un complément d'information par rapport aux autres éléments les plus proches.
- La balise `<footer>` correspond à une section qui se place en fin de document, de section ou d'article. Le plus souvent, cette balise correspond au pied de page du document.
- La balise `<main>` spécifie le contenu principal d'un document.

Voici un exemple de structure du document et d'imbrication des différentes balises HTML5 évoquées ci-dessus :

```
<header>
header
  <nav>
    nav
  </nav>
</header>

<section>
 <article>
    article 1
 </article>
 <article>
    article 2
 </article>
</section>

<aside>
    aside
</aside>
<footer>
    footer
</footer>
```

Par défaut, les éléments HTML se placent les uns sous les autres. Nous verrons dans les chapitres suivants l'importance du CSS pour agir sur la disposition de ces éléments dans une page web.

Graphiquement, l'objectif sera de subdiviser le document HTML comme le décrit le schéma ci-dessous :

<table>
<tr><td colspan="2">HEADER

NAV</td></tr>
<tr><td>SECTION

ARTICLE
ARTICLE</td><td>ASIDE</td></tr>
<tr><td colspan="2">FOOTER</td></tr>
</table>

1.2.2 Nouveaux éléments pour le multimédia

HTML5 fournit de façon native des outils multimédias par le biais de l'utilisation des balises `<video>`, `<audio>` et surtout, `<canvas>`.

Il devient possible de lire des vidéos en HTML5 sans recourir à l'installation d'un plug-in.

L'élément `<canvas>` permet de réaliser des dessins et des animations interactives. Il fournit une surface de dessin.

Les éléments `<video>` et `<audio>` permettent d'insérer des vidéos et des sons dans les pages web.

Voici un exemple d'utilisation des éléments `<video>` et `<canvas>` :

```
<video src="myVideo.mp4" width="400" height="300"></video>
<canvas id="myCanvas"></canvas>
```

Pour l'élément `<canvas>`, par le biais d'instructions JavaScript, on pourra créer des formes et des couleurs, et manipuler les pixels dans la zone `myCanvas` disponible avec la balise HTML5 `<canvas>`.