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.

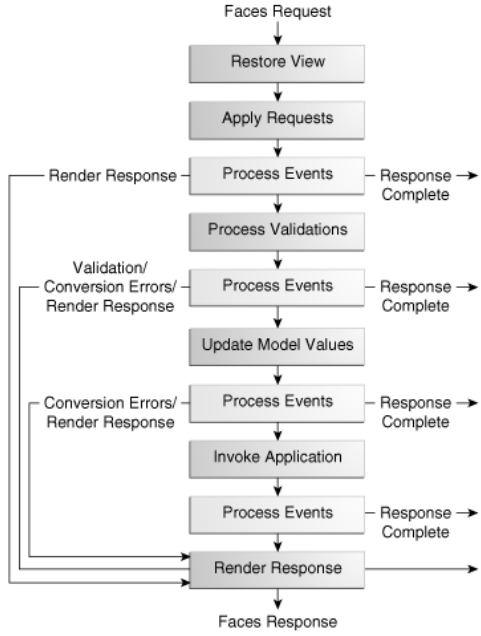
JSF est constitué des éléments suivants :

- 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.

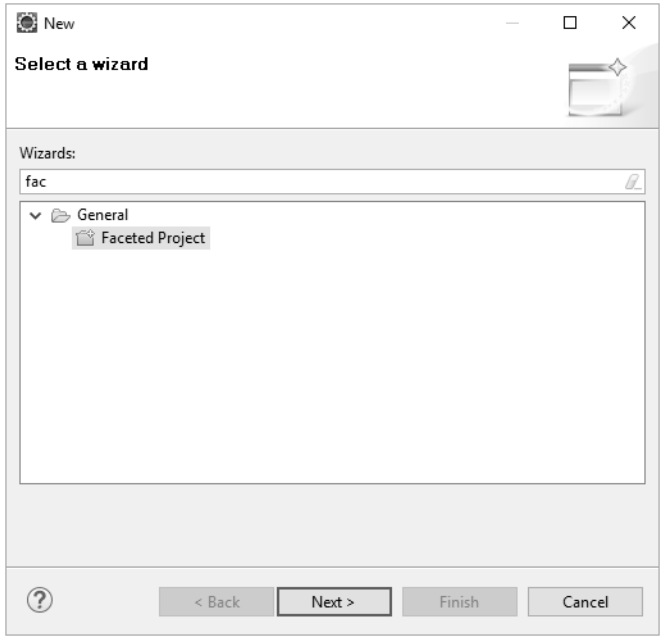
Lors de la première requête vers une ressource, seules les étapes *Restore View et Render Response* sont exécutées.

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** :