Chapitre 4 : Middlewares orientés composant

1 Introduction

1.1 Objectifs

- ✓ Arriver aux concepts d'étagères électroniques ce qui permet de stocker et retrouver des composants préconstruits
- ✓ Fournir des produits logiciels hautement fiables avec un coût réduit et un délai court

1.2 De plus en plus d'abstraction

Procédures et fonctions :

- Sous-programmes réalisant une action

Modules :

- Groupes de fonctions, de méthodes et de traitement, sous forme de bibliothèques

Objet :

- Brique de base logicielle, représentant une entité du monde physique, encapsulant un état et des traitements

Composant :

- Élément logiciel contenant du code compilé (donc opaque)
- Séparation des préoccupations techniques et fonctionnelles
- Définit une interface pour communiquer avec les autres composants, et une interface de configuration

Service

1.3 Approche Orientée Composants

- Construction d'applications à partir de l'assemblage de composants
- Principe : le développement et assemblage de composants peut être réalisé séparément par des acteurs différents à des endroits différents
- Utilisation de la composition comme mécanisme de réutilisation au lieu de l'héritage

2 Composant

2.1 Définitions

- Brique logicielle préfabriquée conçue pour être composée (assemblée) avec d'autres composants
- Réutilisable
- Son utilisation ne nécessite pas de connaissance sur l'implémentation
- Unité binaire : code source n'est pas forcément livré avec le composant

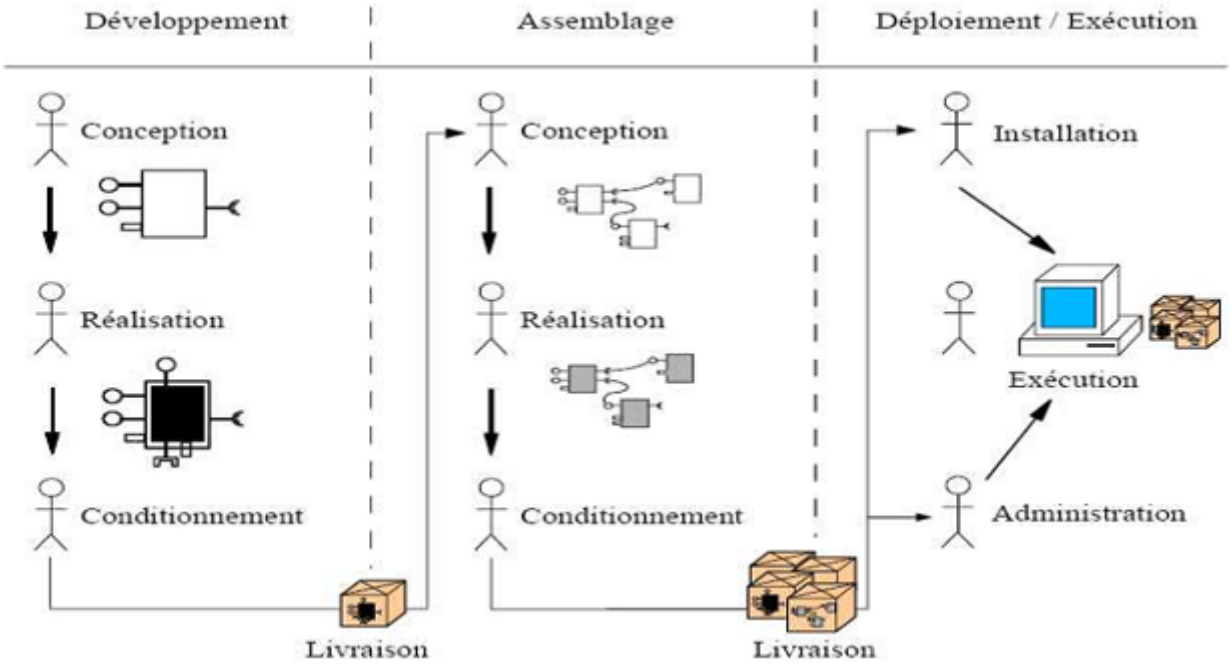

Figure 4.1- Cycle de vie d'une application à composant

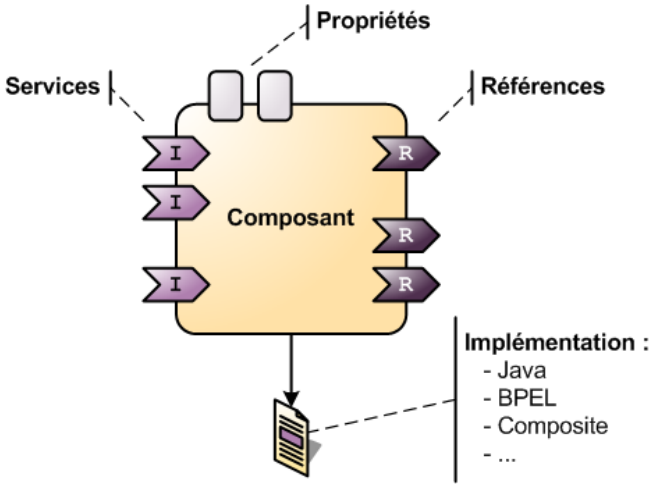

Figure 4.2- Un composant

2.2 Caractéristiques

2.2.1 Classe de composant (équivalente à la notion de classe en OO)

- Définit l'implémentation du composant (logique fonctionnelle)
- Peut être vue à partir :
  - D'une vue externe : vision des clients sur les instances du composant
  - D'une vue interne : vision de l'environnement d'exécution sur les instances du composant

2.2.2 Instance de composant (équivalente à la notion d'objet en OO)

- Obtenue à partir d'une classe de composants
- Fournit les fonctionnalités du composant à l'exécution
- Unique par rapport aux autres instances, peut avoir un état

2.2.3 Paquetage de composant

- Unité permettant de réaliser la livraison et le déploiement d'une classe de composant de manière indépendante
- Contient tout ce qui est nécessaire pour réaliser la création d'instances de la classe de composants
  - Code binaire du composant
  - Ressources nécessaires à son exécution (bibliothèques, images, fichiers de config)

2.2.4 Environnement d'exécution

- Fournit du support aux applications construites à partir du modèle à composants, lors de l'exécution
- Sorte de mini-système d'exploitation : gère les aspects divers (cycle de vie des instances, propriétés non fonctionnelles)

3 Modèles à Base de Composants Existants

- COM (Component Object Model, Microsoft)
  - ✓ Résout le problème d'interopérabilité
  - × Mais : ne propose pas une vue externe constante (les interfaces peuvent varier)
- JavaBeans (Sun)
  - ✓ Simplifier la construction d'applications grâce à la composition visuelle
  - × Vise les applications non distribuées avec interface utilisateur

- EJB[1] (Enterprise Java Beans, Sun)
  - ✓ Vise les applications réparties en trois tiers (MVC)[2]
  - × Ne supporte pas que la partie contrôleur soit distribuée
- CCM (CORBA Component Model, OMG)
  - Définir l'architecture d'une application distribuée sous forme de composition d'instances de composants
  - Utilisation d'un langage abstrait pour la description d'interfaces (IDL) + d'un langage CIDL pour décrire les implémentations) → Obligation de tout écrire en langage abstrait pour de compiler pour le langage cible
  - Supporte les applications distribuées

4 Frameworks Java

Avant de discuter des frameworks Java, discutons de ce que sont les frameworks logiciels. En programmation informatique, un ***framework*** (appelé aussi **infrastructure logicielle**[1], **infrastructure de développement**, **environnement de développement**, **socle d'applications**, **cadre d'applications** ou **cadriciel**) est un ensemble cohérent de composants logiciels structurels qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel, c'est-à-dire une architecture.

Un framework se distingue d'une simple bibliothèque logicielle principalement, d'une part par son caractère générique, faiblement spécialisé, contrairement à certaines bibliothèques ; un framework peut à ce titre être constitué de plusieurs bibliothèques, chacune spécialisée dans un domaine. Un framework peut néanmoins être spécialisé dans un langage particulier, une plateforme spécifique, ou un domaine particulier.

Les frameworks sont donc conçus et utilisés pour modeler l'architecture des logiciels applicatifs, des applications web, des middlewares et des composants logiciels.

4.1 Définition

Les frameworks Java ne sont rien d'autre que le corps ou la plate-forme de codes déjà écrits. Ils sont utilisés par les développeurs Java pour développer des applications Web. Les frameworks Java

[1] Enterprise JavaBeans (EJB) est une architecture de composants logiciels côté serveur pour la plateforme de développement Java EE. Nous aborderons EJB au chapitre suivant.
[2] Modèle-vue-contrôleur ou MVC est un motif d'architecture logicielle destiné aux interfaces graphiques. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs. Un modèle (Model) contient les données à afficher. Une vue (View) contient la présentation de l'interface graphique. Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur

consistent en une collection de classes et de fonctions prédéfinies, qui peuvent être utilisées pour traiter les entrées, gérer les périphériques matériels qui interagissent avec le logiciel du système.

5 Meilleurs Frameworks Java

Maintenant que nous avons une bonne compréhension de ce que sont les frameworks Java, allons-y et examinons les meilleurs frameworks Java actuellement utilisés par l'industrie du logiciel.

5.1 Java EE

Jakarta EE (anciennement Java 2 Platform, Enterprise Edition, ou J2EE, puis Java Platform, Enterprise Edition ou Java EE), est une spécification pour la plate-forme Java d'Oracle, destinée aux applications d'entreprise.

Jakarta EE (anciennement Java 2 Platform, Enterprise Edition, ou J2EE, puis Java Platform, Enterprise Edition ou Java EE), est une spécification pour la plate-forme Java d'Oracle, destinée aux applications d'entreprise. La première version des spécifications de Java EE fut publiée en 1999, la version 1.3 apparut en 2001, puis la version 1.4 en 2003 (support XML et services Web) et la version 1.5 (renommée Java EE 5) en 2007. Depuis le mois d'aout 2017 la version en cours est Java EE 8.

En 2018, le projet est confié par Oracle à la Fondation Eclipse, et le nom Jakarta EE est choisi par la communauté des développeurs à la place de Java EE.

5.1.1 Outils de développement

Pour le développement des applications Java EE, Java EE définit les éléments suivants :

- Plate-forme (Java EE Platform), pour héberger et exécuter les applications, incluant outre Java SE des bibliothèques logicielles additionnelles du Java Development Kit (JDK).
- Suite de tests (Java EE Compatibility Test Suite) pour vérifier la compatibilité.
- Réalisation de référence (Java EE Reference Implementation), dénommée GlassFish.
- Catalogue de bonnes pratiques (Java EE BluePrints).
- Code script.

À chaque version de Java EE correspond notamment, comme toutes les éditions Java :

- Java Specification Requests (JSR), constituant les spécifications de la version considérée.
- Java Development Kit (JDK), contenant les bibliothèques logicielles.
- Java Runtime Environment (JRE), contenant le seul environnement d'exécution (compris de base dans le JDK).

- JavaFX est un framework et une bibliothèque d'interface utilisateur, qui permet aux développeurs Java de créer une interface graphique.

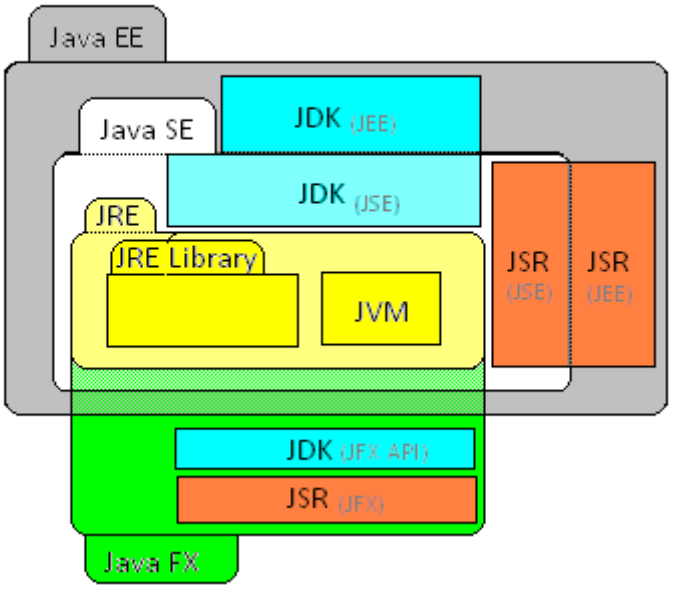

Figure 4.3- Positionnement de Java EE vs Java SE.

Les JDK spécifiques à Java EE sont conçus de façon qu'une application réalisée avec Java EE fonctionne sur le même JRE qu'une application écrite avec Java SE, mais nécessitera cependant qu'en complément, les bibliothèques exploitées soient fournies par un conteneur Java tel que *Payara, JBoss* ou *JOnAS.*

5.1.2 Interfaces de programmation

Ci-dessous, une liste de composants pouvant être contenus dans une implémentation Java EE :

- Enterprise JavaBean EJB : Composants distribués transactionnels.
- Java API pour XML RPC (JAX-RPC) : qui est la technologie principale qui fournit le support pour le développement des services web dans la plate-forme J2EE.
- Java Remote Method Invocatin (RMI) et RMI-IIOP : qui est le protocole de communication des objets distribués.
- Java Naming and Directory Interface (JNDI) : qui est une API pour l'accès aux systèmes de nommages de différents produits.
- Java DataBase Connectivity (JDBC) : qui est une API pour l'accès à n'importe qu'elle base de données relationnelles,
- Java Transaction API (JTA) et Java Transaction Service (JTS) : qui sont deux API permettant aux composant d'utiliser un support fiable de transaction,

- Java Messaging Service (JMS) : qui est une API permettant la communication par message.
- Java Servlet : technologie qui définit un modèle de composant coté serveur écrit en java déjà compilé pour les serveurs web pour la génération des pages web dynamiques.
- JavaServer Pages (JSP) : est un langage de script qui permet aux développeurs de créer dynamiquement du code HTML, XML ou tout autre type de page web.

5.2 Spring

Spring est un framework open source pour construire et définir l'infrastructure d'une application Java, dont il facilite le développement et les tests.

Spring est considéré comme un conteneur dit « léger ». La raison de ce nommage est expliquée par Erik Gollot dans l'introduction du document *Introduction au framework Spring : «* Spring est effectivement un conteneur dit « léger », c'est-à-dire une infrastructure similaire à un serveur d'applications J2EE. Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets. Le gros avantage par rapport aux serveurs d'application est qu'avec Spring, les classes n'ont pas besoin d'implémenter une quelconque interface pour être prises en charge par le framework (au contraire des serveurs d'applications J2EE et des EJBs). C'est en ce sens que Spring est qualifié de conteneur *« léger »*

5.2.1 Concepts

Spring s'appuie principalement sur l'intégration de trois concepts clés :

- L'inversion de contrôle est assurée de deux façons différentes : la recherche de dépendances et l'injection de dépendances.
- La programmation orientée aspect.
- Une couche d'abstraction.

La couche d'abstraction permet d'intégrer d'autres frameworks et bibliothèques avec une plus grande facilité. Cela se fait par l'apport ou non de couches d'abstraction spécifiques à des frameworks particuliers. Il est ainsi possible d'intégrer un module d'envoi de mails plus facilement.

- La recherche de dépendance : consiste pour un objet à interroger le conteneur, afin de trouver ses dépendances avec les autres objets. C'est un cas de fonctionnement similaire aux EJBs.
- L'injection de dépendances : cette injection peut être effectuée de trois manières possibles.
  - l'injection de dépendance via le constructeur,
  - l'injection de dépendance via les modificateurs (setters),

- l'injection de dépendance via une interface.

5.2.2 Composition de Spring

A. **Noyau de base :** Le noyau de Spring est basé sur :

- Fabrique générique de composants informatiques, composants nommés beans.
- Conteneur capable de stocker ces beans.

De plus, le noyau de Spring permet de forcer le contrôle de ces composants de leur extérieur, par l'inversion de contrôle.

Le principal avantage est de composer les beans de façon plus déclarative plutôt que de façon impérative dans le programme. Les beans peuvent être définis par le biais de fichiers de configuration en Java ou XML.

B. **Compléments :** Divers modules Spring Framework viennent en complément du noyau de base pour permettre l'intégration avec les autres bibliothèques et framework.

5.3 Apache Struts

Apache Struts est un framework libre servant au développement d'applications web Java EE. Il utilise et étend l'API Servlet Java[3] afin d'encourager les développeurs à adopter l'architecture Modèle-Vue-Contrôleur (MVC).

Apache Struts 2 C'est le nouveau framework de présentation de la communauté Open Source Apache. C'est un framework Java EE développé à partir de deux autres framework Java EE : Struts 1 (Apache Struts) et WebWork. Il devait initialement être publié sous le nom de WebWork2, avant d'être finalement publié sous son nom actuel

Apache Struts 2 est un framework Java, pour le développement d'applications Web. Ce n'est pas une extension de *Apache Struts 1*. Struts 2 regroupe les avantages de deux précédents outils, WebWork et Struts 1, mais c'est une refonte complète ; Cette seconde génération de framework MVC (Modèle-vue-contrôleur) utilise les notions suivantes : intercepteurs, annotations, langage d'expression OGNL , l'intégration d'outils comme JSTL (JavaServer Pages Standard Tag Library) ou Spring framework.

[3] Un ou une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs web. Les servlets utilisent l'API Java Servlet (package javax.servlet).

5.3.1 Utilisation

Cette infrastructure permet la conception et l'implémentation d'applications Web de taille importante par différents groupes de personnes. En d'autres termes, les designers, développeurs de composants logiciels peuvent gérer leur propre part du projet de manière découplée.

Struts permet la structuration d'une application Java sous forme d'un ensemble d'actions représentant des événements déclenchés par les utilisateurs de l'application. Ces actions sont décrites dans un fichier de configuration de type XML décrivant les cheminements possibles entre les différentes actions. En plus de cela, Struts permet d'automatiser la gestion de certains aspects comme la validation des données entrées par les utilisateurs via l'interface de l'application. Plus besoin de venir coder le contrôle de chaque donnée fournie par un utilisateur, il suffit de décrire les vérifications à effectuer dans un fichier XML affecté à cette tâche.

5.3.2 Limites

En utilisant Struts, le développeur simplifie son travail au niveau des vues et des contrôleurs du modèle MVC. Mais il serait inadapté d'utiliser ce framework dans des projets de petite taille car il introduit une certaine complexité. Struts montre toute sa puissance dans des applications d'une certaine envergure.

Struts est une application mature et correctement documentée. Elle fait face à de nouvelles infrastructures MVC plus légères telles que « Apache Tapestry » ou « JSF ».