Introduction au Framework Spring

Lahfari Bilal

December 22, 2024

1 Framework Spring

1.1 Qu'est-ce que le Framework Spring ?

Le **Framework Spring** est un framework open-source conçu pour développer des applications d'entreprise en Java. Il simplifie le processus de développement en offrant un modèle complet de programmation et de configuration, facilitant ainsi la création d'applications robustes et évolutives.

Figure 1: Framework Spring

1.2 Concepts de base de Spring

1.2.1 Principe de l'Inversion de Contrôle (IoC)

L'Inversion de Contrôle est un principe de conception où le contrôle de la création et de la gestion des objets est transféré du code applicatif à un conteneur ou framework—dans ce cas, **le conteneur IoC de Spring**. Cela signifie que, au lieu de créer leurs dépendances, les objets dépendent du conteneur IoC pour gérer leur cycle de vie et leurs dépendances. Cela permet un code plus

modulaire et maintenable, car les composants peuvent être facilement remplacés ou modifiés sans affecter les autres parties de l'application.

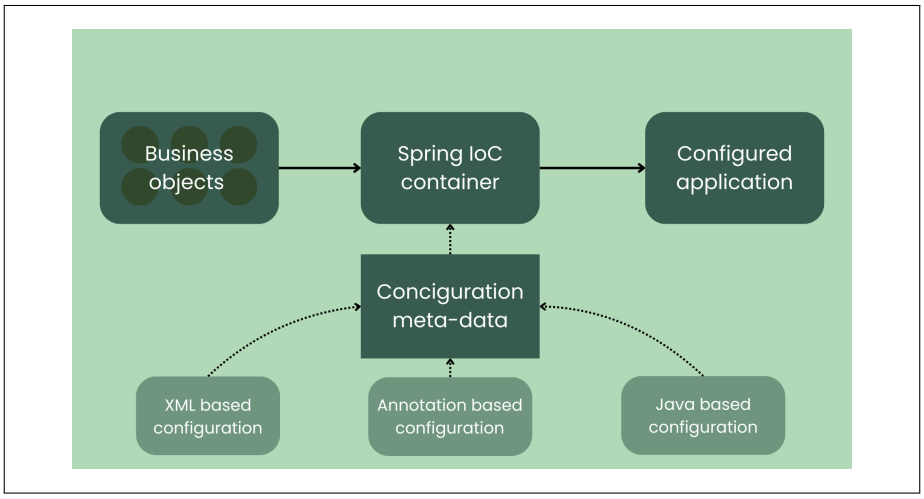


Figure 2: Concept du conteneur IoC de Spring

1.2.2 Exemple d'IoC

Pour démontrer l'Inversion de Contrôle, voici un exemple de code Java montrant une configuration Spring simple :

- **Objets métiers** : POJO signifie Plain Old Java Object. Ce sont des classes Java de base qui représentent la logique métier ou les données de votre application. Elles ne contiennent aucun code Spring spécifique—ce sont de simples objets Java définissant vos données et opérations.
- **Conteneur IoC de Spring** : Le conteneur IoC est la partie centrale de Spring. Il est responsable de la création, de la gestion et de l'assemblage de vos objets (les POJOs) et de leurs dépendances.
- **Métadonnées de configuration** : Les métadonnées de configuration définissent les paramètres ou instructions pour le conteneur IoC. Elles indiquent au conteneur quels objets (beans) créer et comment ils doivent être connectés.
  - **Basé sur XML** : Vous écrivez des fichiers XML pour spécifier quels beans (objets) créer et comment les connecter.
  - **Basé sur les annotations** : Vous utilisez des annotations Java directement dans votre code pour indiquer à Spring quelles classes utiliser et comment les connecter.

  - **Basé sur Java** : Vous utilisez du code Java pour configurer les beans et leurs dépendances, généralement via des classes annotées `@Configuration`.

- **Application** : Une fois que le conteneur IoC lit les métadonnées de configuration et assemble les beans (POJOs), il en résulte une application entièrement configurée.

1.2.3 Beans Spring et injection de dépendances

- **Beans Spring** Un bean Spring est un objet instancié, assemblé et géré par le conteneur IoC de Spring. Ces beans représentent l'épine dorsale d'une application Spring, encapsulant la logique et les fonctionnalités de l'application. La gestion des beans inclut leur cycle de vie, leur configuration et leurs dépendances, définies via des métadonnées fournies par le développeur.

- **Injection de dépendances** : est un modèle de conception utilisé pour implémenter l'IoC. Cela permet au conteneur Spring de fournir les dépendances d'un objet au lieu que l'objet les crée lui-même. Cela mène à un couplage lâche entre les composants et améliore la testabilité.

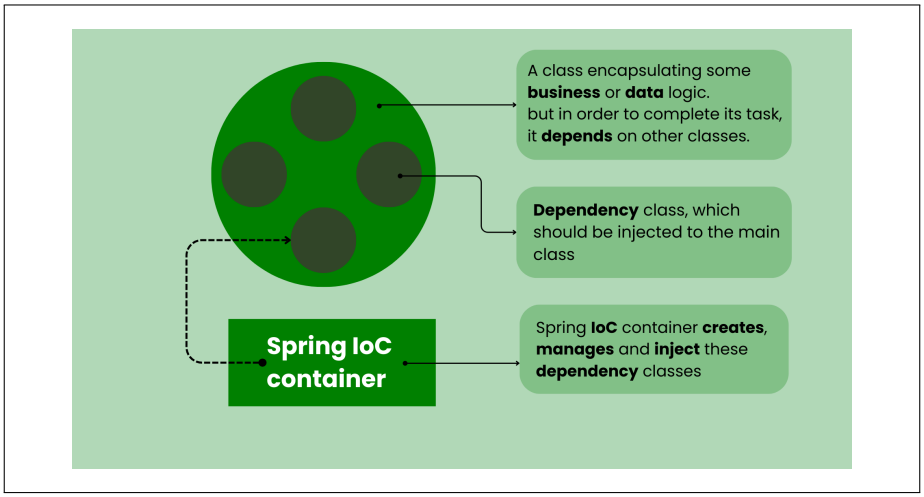


Figure 3: Injection de dépendances des beans

1.2.4 Exemples d'Injection de Dépendances

1. **Configuration basée sur XML**
2. **Configuration basée sur les annotations**
3. **Configuration basée sur Java**

2 Spring Boot

Spring Boot est une extension du Framework Spring qui simplifie le processus de développement des applications Spring. Pour les débutants, comprendre comment Spring Boot se rapporte au Framework Spring peut aider à clarifier son objectif et ses avantages.

Figure 4: Spring Boot

2.1 Caractéristiques clés de Spring Boot

- **Configuration automatique** : Spring Boot configure automatiquement votre application en fonction des bibliothèques incluses. Par exemple, si vous ajoutez une bibliothèque pour le développement web, il configurera un serveur web sans nécessiter une configuration extensive.
- **Applications autonomes** : Avec Spring Boot, vous pouvez créer des applications autonomes qui fonctionnent indépendamment. Il inclut un serveur web embarqué (comme Tomcat), ce qui élimine le besoin de déployer votre application sur un serveur externe.
- **Dépendances de démarrage** : Au lieu de spécifier manuellement chaque dépendance dans votre fichier de configuration de projet, Spring Boot fournit des `starters` qui regroupent les bibliothèques couramment utilisées. Par exemple, `spring-boot-starter-web` inclut tout ce dont vous avez besoin pour une application web.

3 Mise en place d'une application Spring Boot

3.1 Prérequis

3.1.1 Kit de développement Java (JDK)

Vous devez avoir Java installé sur votre système. Spring Boot 3.x nécessite au moins **Java 17 ou une version supérieure**. Assurez-vous que votre version de Java est compatible avec la version de Spring Boot que vous souhaitez utiliser.

3.1.2 Outils de build

Vous avez besoin d'un outil de build pour gérer les dépendances et les processus de build de votre projet. Les deux outils les plus couramment utilisés avec Spring Boot sont :

- **Maven** : La version 3.6.3 ou une version ultérieure est requise.
- **Gradle** : Les versions 7.x (7.5 ou ultérieures) et 8.x sont prises en charge.

3.1.3 Environnement de développement intégré (IDE)

- **Eclipse IDE**
- **IntelliJ IDEA**
- **Visual Studio Code**

3.2 Créer notre premier projet

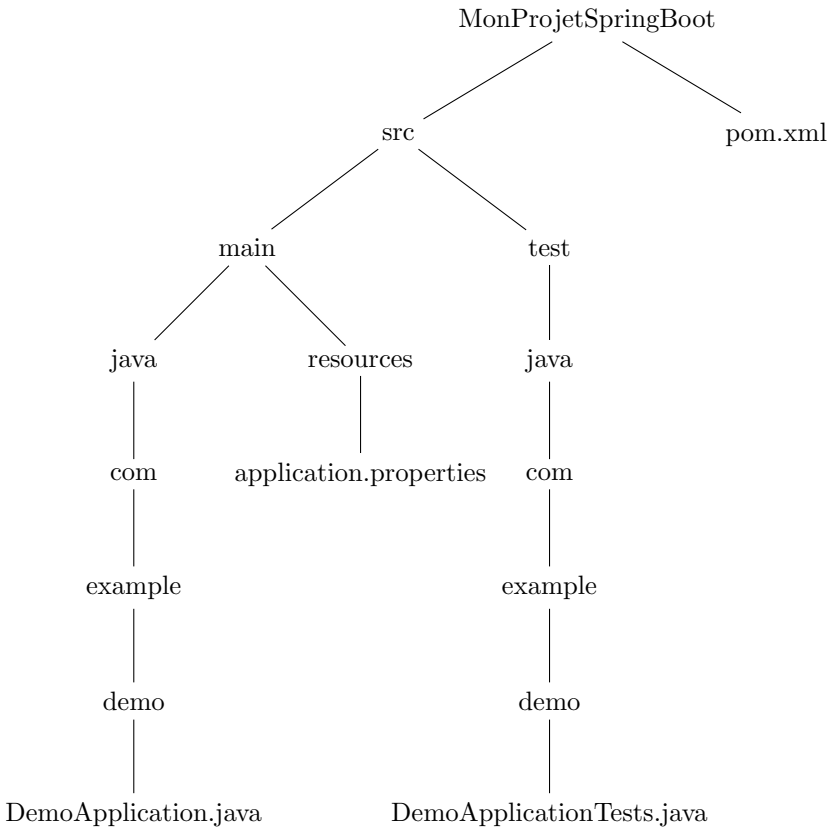
3.2.1 Initialisation du projet

Il est recommandé d'utiliser Spring Initializr (https://start.spring.io/) pour configurer un projet Spring Boot pour les raisons suivantes :

- Génère rapidement la structure du projet et les dépendances.
- Permet de sélectionner les modules nécessaires (par exemple, Spring Web, Spring Data JPA).
- Fournit des builds préconfigurés pour Maven/Gradle.
- Économise du temps et simplifie le processus de configuration.

Figure 5: Spring Initializr

3.2.2 Structure du projet



1. **src/main/java** :

- Ce répertoire contient le code source principal de votre application.
- Il est organisé par package, suivant généralement la convention du domaine inversé (par exemple, `com.example.votreapplication`).
- La classe principale (par exemple, `VotreApplication.java`) contient la méthode `main` qui démarre l'application Spring Boot.

2. **src/main/resources** :
   - Ce dossier est destiné aux fichiers de ressources dont votre application a besoin, comme les fichiers de configuration et les ressources statiques.
     - `static` : Contient des fichiers statiques comme des fichiers CSS, JavaScript et images pouvant être servis directement.
     - `templates` : Contient les templates côté serveur (par exemple, Thymeleaf ou FreeMarker) utilisés pour générer des pages web dynamiques.
     - `application.properties` : Un fichier de configuration clé où vous définissez divers paramètres pour votre application, tels que le port du serveur, les détails de connexion à la base de données, et d'autres propriétés.
3. **src/test/java** :
   - Ce répertoire contient les classes de tests correspondant à votre code d'application principal.
   - Il suit la même structure de package que `src/main/java`, permettant de garder les tests organisés à côté du code qu'ils testent (par exemple, `VotreApplicationTests.java`).
4. **pom.xml** :
   - Il s'agit du fichier de build Maven qui gère les dépendances, les plugins et d'autres configurations du projet.
   - Il spécifie les dépendances starter de Spring Boot que vous utilisez (comme `spring-boot-starter-web` pour les applications web) et d'autres bibliothèques nécessaires à votre projet.

4 Conclusion

Le Framework Spring et Spring Boot fournissent des outils puissants pour construire des applications d'entreprise évolutives et maintenables. En exploitant des concepts tels que l'Inversion de Contrôle (IoC) et l'Injection de Dépendances, les développeurs peuvent créer des systèmes modulaires, testables et facilement configurables. Spring Boot simplifie davantage le processus de développement grâce à sa configuration automatique, ses applications autonomes et ses dépendances préconfigurées, en faisant un choix idéal pour les

applications modernes basées sur Java. Comprendre les caractéristiques clés et le processus de configuration de Spring et Spring Boot aidera les développeurs à rationaliser le développement de leurs applications et à améliorer leur productivité.