Introduction à Spring Boot — Les bases et l'environnement

1. Qu'est-ce que le Spring Framework ?

Le **Spring Framework** est l'un des frameworks les plus puissants et les plus populaires pour le développement d'applications Java, en particulier les applications web et les applications d'entreprise (Enterprise Applications). Il a été créé pour pallier la complexité inhérente aux anciennes technologies Java (comme J2EE) en offrant une infrastructure complète et modulaire.

Le framework Spring repose fondamentalement sur deux principes d'architecture logicielle majeurs :

- **L'Inversion de Contrôle (Inversion of Control - IoC) :** Au lieu que le développeur instancie et gère manuellement les objets via le mot-clé `new`, c'est le framework Spring (spécifiquement le conteneur Spring ou *Spring Container*) qui prend en charge la création, la configuration et la gestion du cycle de vie de ces objets (appelés *Beans*).
- **L'Injection de Dépendances (Dependency Injection - DI) :** Il s'agit de l'implémentation pratique de l'IoC. Spring fournit automatiquement aux objets les dépendances (autres objets) dont ils ont besoin pour fonctionner. Cela réduit considérablement le couplage (Coupling) entre les composants, rendant le code plus modulaire, testable et facile à maintenir.
- **Programmation Orientée Aspect (Aspect-Oriented Programming - AOP) :** *(Notion avancée)* Spring permet de séparer les préoccupations transversales (comme la journalisation, la sécurité, ou la gestion des transactions) du code métier principal, rendant l'application beaucoup plus propre.

> **Remarque importante :** Spring n'est pas un simple outil monolithique, mais un vaste écosystème composé de plusieurs modules indépendants tels que : *Spring MVC* pour les API et interfaces web, *Spring Data* pour l'interaction simplifiée avec les bases de données, et *Spring Security* pour une gestion robuste de l'authentification et des autorisations.

2. Qu'est-ce que Spring Boot ?

Malgré la puissance incontestable de Spring, la configuration initiale d'un nouveau projet exigeait historiquement un travail fastidieux : fichiers XML interminables, classes de configuration complexes, et une gestion stricte de la compatibilité des versions des bibliothèques. C'est pour résoudre cette problématique qu'est né **Spring Boot**.

Spring Boot est une extension du framework Spring conçue pour accélérer et simplifier drastiquement le développement d'applications Java, les rendant prêtes pour la production (*Production-Ready*) avec un minimum de configuration manuelle. Sa philosophie repose sur le principe de "*Convention over Configuration*" (privilégier les conventions par défaut plutôt que la configuration explicite).

Les atouts majeurs de Spring Boot :

- **Auto-Configuration :** Spring Boot analyse les bibliothèques présentes dans votre projet (classpath) et configure automatiquement l'application en conséquence. Par exemple, s'il détecte une base de données en mémoire, il la configure sans que vous n'ayez à écrire une ligne de code.
- **Serveurs Intégrés (Embedded Servers) :** Spring Boot embarque des serveurs web comme Tomcat, Jetty ou Undertow directement dans l'application. Vous n'avez plus besoin de déployer des fichiers WAR sur des serveurs externes ; l'application s'exécute comme un simple programme Java (fichier `.jar`).
- **Les "Starters" (Dépendances de démarrage) :** Ce sont des descripteurs de dépendances pré-packagés. Si vous voulez créer une API web, il suffit d'inclure le `spring-boot-starter-web` pour obtenir instantanément toutes les bibliothèques nécessaires (Spring MVC, Jackson, Tomcat, etc.) avec des versions garanties compatibles.

3. La différence fondamentale entre Spring et Spring Boot

Pour clarifier ces concepts, voici une comparaison détaillée :

Critère de comparaison	Spring Framework	Spring Boot
Objectif principal	Offrir une infrastructure flexible et robuste pour construire des applications d'entreprise complexes.	Réduire le temps de développement (RAD) et livrer des applications Spring prêtes à l'emploi le plus vite possible.
Configuration	Nécessite une configuration manuelle lourde (fichiers XML ou annotations Java complexes).	S'appuie sur l'Auto-Configuration intelligente. La configuration manuelle est presque inexistante.
Gestion des dépendances	Le développeur doit spécifier chaque bibliothèque et gérer minutieusement les conflits de versions.	Utilise les `Starters` qui regroupent logiquement les dépendances avec des versions pré-testées et compatibles.

Critère de comparaison	Spring Framework	Spring Boot
Déploiement & Serveur	Génération d'un fichier `.war` nécessitant un serveur d'application externe configuré (ex: Tomcat standalone).	Génération d'un fichier `.jar` exécutable autonome avec un serveur web intégré.

4. Configuration de l'environnement de développement (Setup)

Avant de plonger dans le code, vous devez préparer votre machine avec les outils indispensables :

1. **Java Development Kit (JDK) :** Installez la version Java 17 (version LTS fortement recommandée) ou supérieure. Vérifiez l'installation via le terminal avec la commande : `java -version`.
2. **Environnement de Développement Intégré (IDE) : IntelliJ IDEA** (la version Community gratuite est excellente) est le standard de l'industrie pour Spring. Vous pouvez également opter pour **Eclipse** (avec le plugin Spring Tools) ou Visual Studio Code (avec l'Extension Pack for Java).
3. **Outil de build :** Spring Boot s'appuie sur **Maven** ou **Gradle** pour télécharger les dépendances et compiler le projet. Ces outils sont généralement intégrés nativement dans votre IDE moderne.

5. Création de votre premier projet Spring Boot

L'approche la plus standard et efficace pour générer l'architecture d'un projet Spring Boot consiste à utiliser le générateur officiel **Spring Initializr**:

- Rendez-vous sur le site officiel : start.spring.io.
- **Project :** Sélectionnez *Maven* ou *Gradle* (Maven est recommandé pour débuter).
- **Language :** Sélectionnez *Java*.
- **Spring Boot :** Gardez la version stable par défaut (évitez les versions contenant SNAPSHOT ou M/RC).
- **Project Metadata :** C'est ici que vous définissez l'identité de votre projet.
  - Group : `com.studiogames` (votre domaine ou entreprise)
  - Artifact : `mystery-backend` (le nom du module)
  - Packaging : *Jar*
  - Java : *17*

apcpedagogie.com

- **Dependencies :** Cliquez sur "Add Dependencies" et ajoutez **Spring Web** (indispensable pour exposer des API REST).
- Cliquez sur le bouton **GENERATE** pour télécharger votre projet sous forme d'archive (.zip).

Extrayez l'archive et ouvrez le dossier résultant directement dans votre IDE.

6. Écriture et exécution de votre première API REST

Laissez à l'IDE quelques instants pour indexer le projet et télécharger les dépendances Maven. Naviguez dans l'arborescence (`src/main/java`) pour trouver la classe principale.

1. La classe de démarrage (MysteryBackendApplication.java) :

```
package com.studiogames.mysterybackend;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

// Cette annotation est le cœur de la magie Spring Boot.
// Elle active l'Auto-Configuration, le balayage des composants (Component Scan) et la
configuration Java.
@SpringBootApplication
public class MysteryBackendApplication {

    public static void main(String[] args) {
        // Démarre l'application Spring et initialise le serveur Tomcat embarqué
        SpringApplication.run(MysteryBackendApplication.class, args);
    }
}
```

2. Création d'un contrôleur (REST Controller) :

Créons un endpoint d'API pour un projet d'investigation. Dans le même package, créez une nouvelle classe nommée `InvestigationController.java`:

```
package com.studiogames.mysterybackend;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

// @RestController indique à Spring que cette classe va intercepter des requêtes HTTP
// et retourner directement des données (généralement du JSON), et non une page HTML.
@RestController
@RequestMapping("/api/cases") // Préfixe pour toutes les routes de ce contrôleur
public class InvestigationController {

    // On associe cette méthode à une requête HTTP GET spécifique
    @GetMapping("/vanishing-diplomat/status")
    public String getCaseStatus() {
        // Simule le renvoi d'un indice ou du statut du dossier en cours
```

```
        return "Dossier : The Vanishing Diplomat | Statut : En cours d'investigation. 2
suspects identifiés.";
    }
}
```

3. Exécution de l'application :

Exécutez votre projet depuis l'IDE (souvent un bouton "Play" vert à côté de votre méthode `main`).

Regardez la console (logs) : dès que vous apercevez la ligne `Tomcat started on port(s): 8080 (http)`, cela signifie que votre serveur est prêt.

Ouvrez votre navigateur web ou un outil comme Postman et accédez à l'URL suivante : **`http://localhost:8080/api/cases/vanishing-diplomat/status`**.

Vous verrez apparaître la réponse de votre toute première API Spring Boot !

*Rédigé et structuré par : **Med Khalil Kribi***

Source : https://apcpedagogie.com/introduction-spring-boot-bases · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com