Spring avec annotations (sans Spring Boot)

Pourquoi passer aux annotations ?

- Moins de verbosité que le XML
- Configuration centralisée dans le code Java
- Meilleure lisibilité, meilleure navigation avec les IDE
- Permet l'autowiring, le scanning automatique des composants
- Préparation naturelle à l'utilisation de Spring Boot

Les annotations fondamentales

`@Component`

- Marque une classe pour qu'elle soit instanciée et gérée par Spring
- C'est l'annotation de base de tout composant

`@Service`, `@Repository`, `@Controller`

- Spécialisations de `@Component`
- Permettent une meilleure séparation des responsabilités

`@Service`, `@Repository`, `@Controller`

- Utilisées respectivement pour :
- La couche métier (`@Service`)
- La couche d'accès aux données (`@Repository`)
- La couche web (`@Controller`)

Injection de dépendances avec `@Autowired`

- Permet à Spring d'injecter automatiquement une dépendance
- Peut être utilisée :
- Sur un constructeur
- Sur un setter
- Sur un champ (non recommandé dans les projets testables)

Injection de dépendances avec `@Autowired`

Exemple

```
@Autowired
public void setService(MyService service) {
    this.service = service;
}
```

Configuration Java avec `@Configuration`

- Permet de définir une classe de configuration Spring (équivalent du `applicationContext.xml`)
- S'utilise avec `@ComponentScan` pour activer la détection automatique des composants

Exemple

```
@Configuration
@ComponentScan(basePackages = "com.example.app")
public class AppConfig {
}
```

Exemple complet

1. Interface

```
public interface MessageService {
    String getMessage();
}
```

2. Implémentation

```
@Service
public class SimpleMessageService implements MessageService {
    public String getMessage() {
        return "Hello from Spring!";
    }
}
```

Exemple complet (suite)

3. Utilisateur du service

```
@Component
public class MessagePrinter {
    private MessageService service;

    @Autowired
    public void setService(MessageService service) {
        this.service = service;
    }

    public void print() {
        System.out.println(service.getMessage());
    }
}
```

4. Classe main

```
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContext(AppConfig
.class);
        MessagePrinter printer = ctx.getBean(MessagePrinter.class);
        printer.print();
    }
}
```

Ce qu'il faut retenir

- Le XML n'est plus nécessaire
- Toute la configuration est possible via :
- `@Component`, `@Service`, `@Repository`, `@Controller`
- `@Autowired` pour l'injection
- `@Configuration` et `@ComponentScan` pour le démarrage
- C'est la base d'une configuration moderne avec Spring (avant Spring Boot)