

Spring avec annotations (sans Spring Boot)

Pourquoi passer aux annotations ?

- ¥ Moins de verbosit  que le XML
- ¥ Configuration centralis e dans le code Java
- ¥ Meilleure lisibilit , meilleure navigation avec les IDE
- ¥ Permet l' utowiring, le scanning automatique des composants
- ¥ Pr paration naturelle   l'utilisation de Spring Boot

Les annotations fondamentales

@Component

- ¥ Marque une classe pour qu'elle soit instanci e et g r e par Spring
- ¥ C'est l'annotation de base de tout composant

@Service, @Repository, @Controller

- ¥ Sp cialisations de @Component
- ¥ Permettent une meilleure s paration des responsabilit s

@Service, @Repository, @Controller

- ¥ Utilis es respectivement pour :
- ¥ La couche m tier (@Service)
- ¥ La couche d'acc s aux donn es (@Repository)
- ¥ La couche web (@Controller)

Injection de d pendances avec @Autowired

- ¥ Permet   Spring d'injecter automatiquement une d pendance
- ¥ Peut  tre utilis e :
- ¥ Sur un constructeur
- ¥ Sur un setter
- ¥ Sur un champ (non recommand  dans les projets testables)

Injection de dŽpendances avec @Autowired

Exemple

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

Configuration Java avec @Configuration

¥ Permet de dŽfinir une classe de configuration Spring (Žquivalent du `applicationContext.xml`)

¥ S'utilise avec `@ComponentScan` pour activer la dŽtection automatique des composants

Exemple

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

Exemple complet

1. Interface

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

2. ImplŽmentation

```
@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)