Annotations dans Spring

Conteneur IoC

Cours n°2
ENSG - M2 TSI

Clément BOIN
clement.boin@gmail.com

Configuration par annotations

- Une alternative à la configuration par des fichiers XML
- Ajouter des noms clefs (tags) précédés d'un @ directement dans les classes pour les configurer.

Exemple

```
@Bean
public Service beanService() {
     return new ServiceImpl();
}
```

Avantages/Inconvénient

- Une classe particulière va inclure la configuration

@Configuration

+ Avantage : Plus besoin de fichiers XML

- Inconvénient : Configuration éparpillée dans les différentes classes

La classe de configuration

`@Configuration`

- Annotation à mettre au dessus du nom de la classe
- Indique que cette classe contient la configuration
- La classe contient une méthode par bean à injecter (voir suite)

Différence avec XML

- Fichier XML :

```
<?xml version="1.0" encoding="UTF-8"?>
   <beans>
   <bean id="typedeverif" class="tsi.ensg.RegularChecker"/>
   </beans>
</xml>
```

- Remplacé par :

```
@Configuration
public class Config {
        @Bean
        PasswordChecker typedeverif(){
                return new RegularChecker();
        }
}
```

Méthodes de la classe de configuration

@Bean

- équivalent à <bean>

- Une méthode par classe que l'on veut injecter
  - Son nom est l'id du bean

- Annotation à mettre au dessus de chacune des méthodes

- Chaque méthode va retourner une instance de la classe à injecter

package :

`org.springframework.context.annotation`

Exemple de classe de configuration

Classe de configuration

```
@Configuration
public class MaConfig {

        @Bean
        public HelloWord beanHello() {
                return new HelloWord();
        }

}
```

POJO

```
class HelloWord { public void affiche()
{ … } }
```

Application

```
ApplicationContext context = new AnnotationConfigApplicationContext(MaConfig.class);

//Le lien se fait sur le nom de la classe
HelloWord helloWorldBean = context.getBean( HelloWord.class);

helloWorldBean.affiche();
```

Injection de valeurs

@Value(….)

- Annotation permettant d’injecter une valeur à une propriété
- Se place dans la classe du bean au dessus de la déclaration de l’attribut (ou setter)

```
public class HelloWord {
        @Value("salut")
        String message;
        ...
}
```

Injection dans un constructeur

```
@Configuration
public class MaConfig {
        @Bean
        public Service beanService() {
                return new ServiceImpl();
        }

        @Bean
        public Application beanApplication() {
                return new Application(beanService());
        }
}
```

```
public class Application {
        private final Service serv;

        public Application(Service serv){
                this.serv=serv;
        }
}
```

Injection de dépendances autowired

@Autowired

- Pilote l'injection de dépendance des bean
- Peut se mettre au dessus de l'attribut ou au dessus du setter
- Par défaut, Spring résout les entrées @Autowired par type.

```
public class Personne {

        @Autowired
        private Adresse adresse;
        ...
}
```

```
//Si Personne est une interface
et Etudiant son implémentation
public class PersonneService {

        @Autowired
        @Qualifier("Etudiant")
        private Personne pers;
        ...
}
```

@Component

- On peut également annoter une classe pour créer un bean pour celle-ci

- Il faut annoter la classe de configuration avec :

```
@ComponentScan
```

- Spring va parcourir le package et les sous-packages à la recherche de classe annotée

```
@Component
```

- On peut aussi préciser le nom du package à parcourir :

```
@ComponentScan("fr.monpackage")
```

Exemple

```
@Component
public class Address {
...
 public Address(
 @Value("5") int streetNumber, @Value("Rue des lilas") String street,
 @Value("France") String country) {
 ...
 }
}

@Component
public class Client {
 private final String name;

 @Autowired
 private Address address;

 public Client(@Value("Bob") String name) { this.name = name;}
}
```

Variantes de @Component

- Nous verrons d’autres variantes de cette annotations dans la suite du cours

@Controller

@Service

@Repository

Conclusion
XML, Bean @Component

- L’utilisation d’une classe de configuration avec des @Bean se rapproche de l’assemblage réaliser avec le fichier XML
  - Mais on reste dans le code source Java

- Avec les @Component la configuration est répartie dans plusieurs fichiers

- La configuration avec @Bean est nécessaire si on n’a pas le code source de l’objet (par exemple si c’est dans une librairie).