

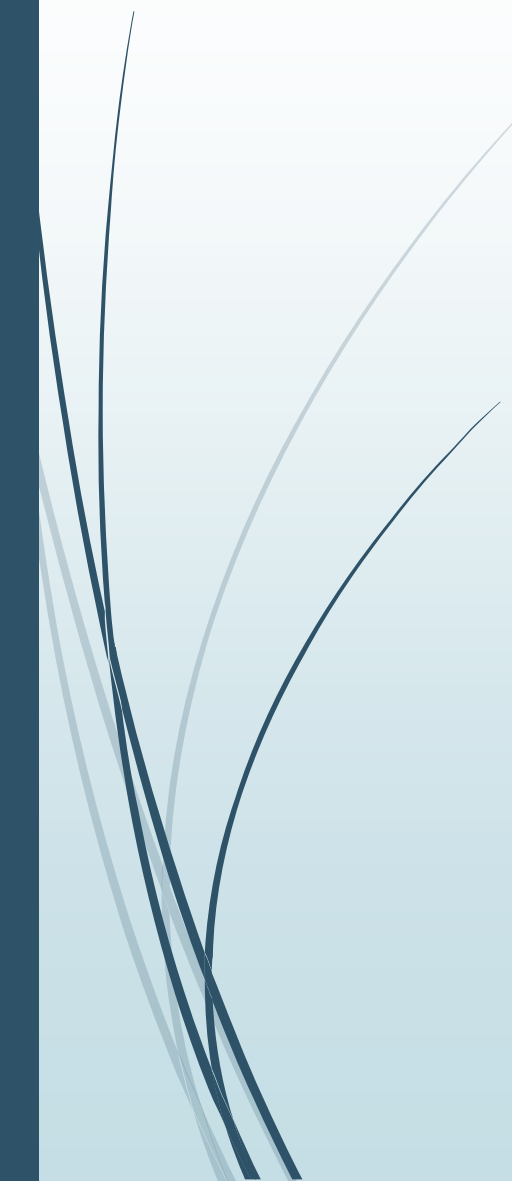
Spring annotations



François Supervielle-Brouquès



Sommaire

- Introduction
 - @Bean
 - @Autowired
 - @Component
 - @Controller
 - @Service
 - @RequestMapping
 - La suite
- 



Introduction



Les annotations Spring permettent d'ajouter des fonctionnalités ou des comportements aux objets sans ajouter directement du code. Les annotations s'adresse au bean qui est une sorte de surcouche (proxy) d'un objet, qui possède notamment les méthodes permettant à l'objet d'exister dans l'environnement Spring.

Cela aide aussi à la lisibilité du code. Attention cependant à ne pas ajouter d'annotations qui pourrait s'avérer inutile et alourdir le code à la compilation.



@Bean

- S'applique au niveau de la methode. Demande à la methode de produire un bean qui sera géré par le Spring Container

```
@Bean  
public BeanExample beanExample()  
{  
  return new BeanExample ();  
}
```

@Autowired

- L'annotation @Autowired permet d'activer l'injection automatique de dépendance. Cette annotation peut être placée sur un constructeur, une méthode setter ou directement sur un attribut (même privé). Le Spring Framework va chercher le bean du contexte d'application dont le type est applicable à chaque paramètre du constructeur, aux paramètres de la méthode ou à l'attribut.

```
@Component
public class Customer
{
    private Person person;
    @Autowired
    public Customer(Person person)
    {
        this.person=person;
    }
}
```

@Component

- est un stéréotype générique puisqu'il indique simplement que cette classe doit être utilisée pour instancier un bean dans le contexte Spring. Les autres stéréotypes fournis par le Spring Framework sont : @Service, @Repository, @Configuration, @Controller et @RestController

```
@Component
public class Student
{
    .....
}
```

@Controller

- Spécialisation de @Component. Il marque une classe comme étant en charge des requêtes web. Souvent utilisé avec @RequestMapping.

```
@Controller
@RequestMapping("books")
public class BooksController
{
    @RequestMapping(value = "/{name}", method = RequestMethod.GET)
    public Employee getBooksByName()
    {
        return booksTemplate;
    }
}
```

@Service

- @Service est purement descriptif.
Un service est un composant qui remplit une fonctionnalité centrale dans l'architecture d'une application. Il renvoie aux classes qui ont la charge de réaliser les fonctionnalités principales. Il s'agit normalement d'une classe qui ne maintient pas d'état conversationnel. (Une classe qui contient une logique de type business « business »).

```
package com.javatpoint;  
  
@Service  
  
public class TestService  
{  
  
    public void service1()  
    {  
        //business code  
    }  
}
```


@RequestMapping

- Utilisé pour diriger les requêtes web vers les méthodes définies et peut être autant utilisé sur les méthodes que sur les classes Il possède de nombreuses options ainsi que des versions abrégées tels que :
 - @GetMapping (abréviation de @RequestMapping(method = RequestMethod.GET) redirige une requête HTTP GET défini en dessous. Et utilisé pour créer un point de terminaison de service d'extraction
 - Existe aussi les bien connus
@PostMapping
@PutMapping
@DeleteMapping

```
@Controller
public class BooksController
{
    @RequestMapping("/computer-science/books")
    public String getAllBooks(Model model)
    {
        //application code
        return "bookList";
    }
}
```



La suite

- @Repository
- @Configuration
- @SpringBootApplication
- @EnableAutoConfiguration
- [...]



ressources

- <https://gayerie.dev/docs/spring/spring/annotations.html>
- <https://www.javatpoint.com/spring-boot-annotations>
- Livre Spring



Merci pour votre écoute :)

Bonne journée !