Formation Spring Framework

Configuration par annotations

Activer les annotations

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:p="http://www.springframework.org/schema/p"
      xmlns:context="http://www.springframework.org/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans http://
www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context http://www.springframework.org/schema/
context/spring-context-4.1.xsd">

      <context:component-scan base-package="fr.pizzeria.spring"></context:component-scan>

</beans>
```

Sans XML

- Via l'annotation @Configuration

```java
@Configuration
@ComponentScan("fr.pizzeria.spring")
public class AppConfig {

}
```

```java
@Component
public class PizzaService {
	public String getVersion() {
		return "1.0";
	}
}
```

```java
public class BeanSpringTest {

	@Test
	public void test_creation_bean_spring() {
		try (AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class)) {
			PizzaService pizzaService = context.getBean(PizzaService.class);
			assertNotNull(pizzaService);
			assertEquals("1.0", pizzaService.getVersion());
		}
	}

}
```

Injection de dépendance

- L'annotation **@Component** définit un bean.
- L'annotation **@Autowired** permet d'injecter une dépendance. Elle peut être placée sur un champ ou un constructeur.

```
@Component
public class PizzaService {

	@Autowired
	private PizzaRepository pizzaRepository;

	public String findAll() {
		return pizzaRepository.findAll();
	}

}
```

Injection de dépendance

- L'annotation **@Qualifier** définit le nom du bean à injecter. Par défaut, l'injection se fait par type.

```
@Component
public class PizzaService {

	@Autowired
	@Qualifier("pizzaRepo1")
	private PizzaRepository pizzaRepository;

	public String findAll() {
		return pizzaRepository.findAll();
	}

}
```

Injection de dépendance

- L'annotation **@Value** définit une valeur pour un bean. Il est possible d'utiliser une variable.

```
@Component
public class PizzaService {

	@Autowired
	private PizzaRepository pizzaRepository;

	@Value("${jdbc.user}")
	public void setUser(String user) {
		this.user = user;
	}

}
```

JSR-330 et Spring

- La JSR-330 spécifie le standard en matière d'annotations pour l'injection de dépendances
  - @Inject
  - Google et SpringSource mène conjointement le lead
- Spring implémente la spécification
  - @Inject couvre 80% des besoins
  - Les autres besoins sont couverts par @Autowired

Les annotations JSR-330

- Spring supporte les annotations de la JSR-330

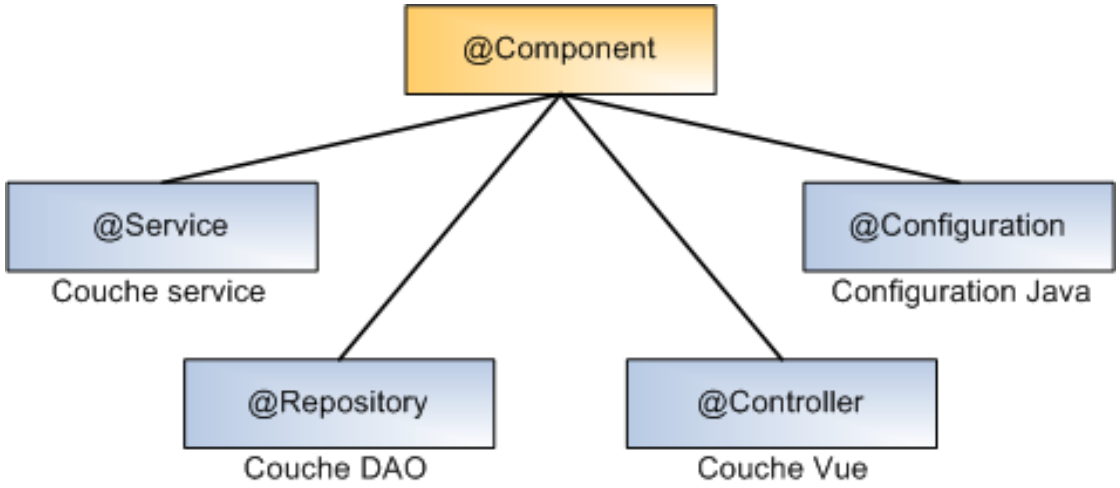
```
@Named
public class PizzaService {

	@Inject
	private PizzaRepository pizzaRepository;

	public String findAll() {
		return pizzaRepository.findAll();
	}

}
```

Annotations Spring vs JSR-330

Spring	javax.inject.*	Commentaires
@Autowired	@Inject	@Inject ne prend pas d⁹attributs.
@Component	@Named	Spring prend en compte cette annotation lors du scan.
@Scope	@Scope	
@Scope("singleton")	@Singleton	En JSR-330, le scope par défaut correspond au « prototype » de Spring.
@Qualifier	@Named	
@Value	N/A	
@Required	N/A	
@Lazy	N/A	

Les stéréotypes

- Un stereotype permet de caractériser le rôle d'un bean au sein du contexte
- Spring propose "en standard" quelques stéréotypes
- D'autres briques de Spring viennent compléter la liste de ces stereotypes



Définir plusieurs beans

- @Bean définit un bean
- L'identifiant du bean est déterminé par le nom de la méthode
- L'injection est effectué dans le corps de la méthode

```
@Configuration
public class AppConfig {
  @Bean
  public PizzaService pizzaService() {
    PizzaService service = new PizzaService();
    return service;
  }
  @Bean
  public PizzaRepository pizzaRepo() {
    return new PizzaRepository();
  }
}
```

Traitements post-processeurs

```
@Component("pizzaService")
@Scope("prototype")
public class PizzaServiceImpl implements IPizzaService {

  @PostConstruct
  public void init() {
        // du code
  }

  @PreDestroy
  public void destroy() {
       // du code
 }
}
```