



Spring Core - annotation

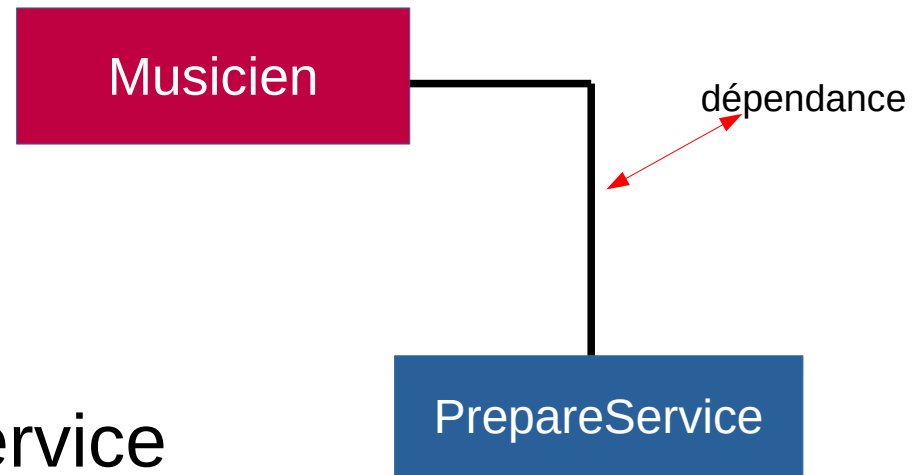
DI - Constructor

Exemple d'injection

Considérons qu'un musicien doit pouvoir se préparer avant une potentielle interprétation :

Je m'installe tranquillement
Je fais des gammes
Je me relaxe, en respirant profondément
Je me concentre
...etc

En utilisant un service : `prepareService`



Rappel : dépendance

Quand un objet contient une instance d'un autre objet et utilise les méthodes de cet autre objet pour accomplir ses propres tâches

```
class Dependante {  
    private T sousTraitante  
  
    public Dependante( ){  
        this.sousTraitante= new T() ;  
    }  
  
    public void soustraitance(){  
        sousTraitante.faitQlqChose() ;  
    }  
}
```

```
Class T {  
  
    public void faitQlqChose(){ ... }  
}
```

On veut pouvoir choisir la dépendance à injecter via la configuration.

Pour pouvoir faire évoluer le code par extension.

```
class Dependante {  
    private I sousTraitante  
  
    public Dependante( ){  
        this.sousTraitante= new T() ;  
    }  
  
    public void soustraitance(){  
        sousTraitante.faitQlqChose() ;  
    }  
}
```

```
Class T implements I{  
  
    public void faitQlqChose(){ ... }  
}
```

```
Interface I {  
  
    public void faitQlqChose() ;  
}
```

Spring Autowiring

- Pour l'injection de dependance Spring peut utiliser auto wiring
- Spring va rechercher une classe qui "matche", qui correspond , à la propriété récipiendaire (la correspondance se fait par Type : claase ou interface)
- Spring l'injecte automatiquement (=autowired)

Démo

dans une classe de Musicien, pour configurer l'injection de dépendance, nous annoterons avec @Autowired, au choix :

- un constructeur
- un setter
- un attribut
- Au démarrage Spring scannera les packages, créera des beans à chaque classe @Component
- Si Spring trouve une classe qui correspond pour notre configuration de dépendance, il l'injectera.

Démo : Autowire on Constructor

```
@Component
public class ZenPreparationService implements PrepareService {
    @Override
    public String preparesToi() {
        return "je respire calmement, et je me détend";
    }
}
```

```
public interface PrepareService {

    public String preparesToi();
}
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import com.springdemo.interfaces.Musicien;
import com.springdemo.interfaces.PrepareService;
```

```
@Component
public class Trompetiste implements Musicien {

    private PrepareService prepareService;

    @Autowired
    public Trompetiste(PrepareService prepareService) {
        this.prepareService = prepareService;
    }

    @Override
    public String joueTaPartition() {
        return "la trompète du toril";
    }

    @Override
    public String preparesToi() {
        return prepareService.preparesToi();
    }
}
```

```
public class AutowireDemoApp {

    public static void main(String[] args) {

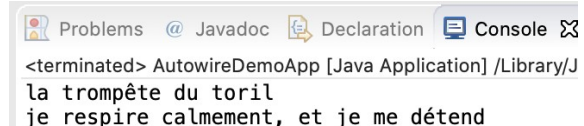
        //charger le fichier de configuration xml
        ClassPathXmlApplicationContext context = new
        ClassPathXmlApplicationContext("applicationContext.xml");

        // accéder au bean géré par le spring container
        Musicien musicien=
            context.getBean("trompetiste", Musicien.class);

        // utiliser les methode de musicien
        System.out.println(musicien.joueTaPartition());
        System.out.println(musicien.preparesToi());

        // fermer le context
        context.close();
    }
}
```

Run As > Java Application =>



```
Problems @ Javadoc Declaration Console
<terminated> AutowireDemoApp [Java Application] /Library/J
la trompète du toril
je respire calmement, et je me détend
```

Rappel du schéma directeur:

Ioc & DI

