



Spring Core - annotation

DI – @Autowired on setter

Autowiring on setter method

- On va pouvoir injecter des dépendances en appelant une méthode setter sur votre classe.
- On va continuer d'utiliser cet exemple déjà vu :
 - Injecter PrepareService dans une implémentation de Musicien
 - Spring scanne les packages , et parmi les classes annotées @Component il en cherche une qui implémente PrepareService.
 - S'il en trouve une , Spring l'injecte automatiquement (par exemple ZenPreparationService)

Démo – autowire on setter

- À partir de la démo précédente , commenter le constructeur et l'injection (invisible) réalisée sur celui-ci
- Créer une méthode set... dans la classe qui héberge la dépendance
- Configurer la dépendance avec l'annotation @Autowired sur ce setter

Démo - TD

```
@Component
public class Trompetiste implements Musicien {

    private PrepareService prepareService;

    // public Trompetiste(PrepareService prepareService) {
    //     System.out.println(">> Trompetiste: à l'intérieur constructeur ");
    //     this.prepareService = prepareService;
    // }

    @Autowired
    public void setPrepareService(PrepareService prepareService) {
        System.out.println(">> Trompetiste : à l'intérieur du setter se prepareService ");
        this.prepareService = prepareService;
    }

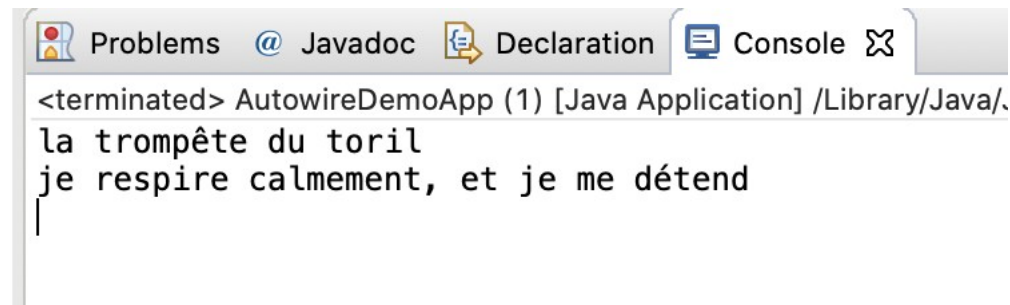
    @Override
    public String joueTaPartition() {
        return "la trompête du toril";
    }

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

}
```

Exécutons le code

- Run as > Java Application =>



The screenshot shows an IDE console window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of a Java application named 'AutowireDemoApp (1) [Java Application]'. The output text is:

```
<terminated> AutowireDemoApp (1) [Java Application] /Library/Java/.  
la trompête du toril  
je respire calmement, et je me détend  
|
```

- Nous pouvons en faite injecter notre dépendance en appelant n'importe quelle méthode sur notre classe , simplement en fournissant une @Autowired annotation sur cette méthode. Quand le contexte est chargé, l'annotation @Autowired est traitée, l'injection se réalise à l'instanciation du bean (essayez pour le tester)