Bộ môn: Kỹ Thuật Phần Mềm

Lập trình WWW *Java*

Ioc – DI Configuration with Java Code

Java Configuration

- Instead of configuring **Spring container using XML**
- Configure the **Spring container** with **Java code**

Three Ways configuring Spring Container

1 Full XML config

```
<!-- Define your beans here -->
<bean id="myCoach"
    class="com.se.springdemo.libs.TrackCoach"
    init-method="doMyStartupStuff"
    destroy-method="doMyCleanupStuff">
    <!-- set up constructor injection -->
    <constructor-arg ref="myFortune" />
</bean>
<!-- define the dependency -->
<bean id="myFortune"
    class="com.se.springdemo.libs.HappyFortuneService">
</bean>
```

2 Annotation - XML Component Scan

```
<context:component-scan base-package="com.se.annotation.libs" />
```

3 Annotation - Java Configuration Class

```
@Configuration
@ComponentScan("com.se.javaconfiguration.libs")
public class SportConfig {
```

No XML

Development Process

- Create a Java class and annotate as **@Configuration**
- Add component scanning support **@ComponentScan** (optional)
- Read **Spring java configuration class**
- Retrieve **Bean** from **Spring container**

Step 1: Create a Java class and annotate as @Configuration

File*: SportConfig.java*

```
@Configuration
public class SportConfig {


}
```

Step 2: Add component scanning support @ComponentScan

File*: SportConfig.java*

```
@ComponentScan("com.se.javaconfiguration.libs")
@Configuration
public class SportConfig { }
```

Step 3: Read Spring java configuration class

File*: JavaConfigDemoApp.java*

```
public class JavaConfigDemoApp {
    public static void main(String[] args) {
        // read spring config java class
        AnnotationConfigApplicationContext context =
            new AnnotationConfigApplicationContext(SportConfig.class);
```

Step 4: Retrieve Bean from Spring container

File: *JavaConfigDemoApp.java*

```
// get the bean from spring container
Coach theCoach = context.getBean("tennisCoach", Coach.class);
```

File: *TennisCoach.java*

```
@Component
public class TennisCoach implements Coach {
    @Autowired
    private FortuneService fortuneService;
    public TennisCoach(){
        System.out.println(">>tennisCoach: inside dafault constructor"); }
    @Override
    public String getDailyFortune() {
        return fortuneService.getFortune();}
    @Override
    public String getDailyWorkout() {
        return "Pratice your backhand volley";   }
}
```

Since we didn't define bean in config file, we need to use some annotation in TennisCoach class

Define Bean with Java Code

File: *SwimSportConfig.java*

```
@Bean
public FortuneService happyFortuneService() {
    return new HappyFortuneService();}
// define bean for our swim coach AND inject de
@Bean
public Coach swimCoach() {
    SwimCoach mySwimCoach = new SwimCoach(happyFortuneService());
    return mySwimCoach;}
```

- Define method for expose bean
- Inject bean dependency

File: *SwimCoach.java*

```
public class SwimCoach implements Coach{
    private FortuneService fortuneService;
    public SwimCoach(FortuneService theFortuneService) {
        fortuneService = theFortuneService;}
    @Override
     public String getDailyWorkout() {
         return "Swim 1000 meters as a warm up.";}
     @Override
     public String getDailyFortune() {
         return fortuneService.getFortune();}
}
```

No special annotations

Define Bean with Java Code

File: *SwimJavaConfigDemoApp.java*

```
@Bean
public Coach swimCoach() {
    SwimCoach mySwimCoach = new SwimCoach(happyFortuneService());
    return mySwimCoach;}
```

```
public static void main(String[] args) {
    // read spring config java class
    AnnotationConfigApplicationContext context =
        new AnnotationConfigApplicationContext(SportConfig.class);
    // get the bean from spring container
    SwimCoach theCoach = context.getBean("swimCoach", SwimCoach.class);
    // call a method on the bean
    System.out.println(theCoach.getDailyWorkout());
    // call method to get the daily fortune
    System.out.println(theCoach.getDailyFortune());
```

Injecting values from Properties file

File: *SwimCoach.java*

1

```
@Value("$foo.email")
private String email;
public String getEmail()
        return email;}
@Value("$foo.team")
private String team;
public String getTeam()
        return team;}
```

Field Injection

File: *sport.properties*

2

```
foo.email=my_coach@mail.com
foo.team=Silly Java Coders
```

File: *swimsportconfig.java*

3

```
@Configuration
@PropertySource("classpath:swimsport.properties")
public class SwimSportConfig {
```

QUESTIONS