**Field Injection**

**Field Injection**:

In field injection we can actually inject the dependency by setting the field values on our class directly, even for private fields and the nice thing is that happens behind the scenes using some Java technology called Reflection.

**Development Process**:

1. Configure the dependency injection with Autowired Annotation

* Applied directly to the field
* No need for setter methods

**Step 1: Configure the dependency injection with Autowired Annotation**:

**File: TennisCoach.java**:

public class TennisCoach implements Coach {

@Autowired

private FortuneService fortuneService;

public TennisCoach() {

}

// no need for setter methods

...

}

Behind the scenes when spring creates our object, then they’ll automatically set this field. So we don’t have to go through any setter methods or anything.

**Example**:

**File: Coach.java**:

**package** com.ruhul.odduu.fieldinjection;

**public** **interface** Coach {

**public** String getDailyWorout();

**public** String getDailyFortune();

}

**File: FortuneService.java**:

**package** com.ruhul.odduu.fieldinjection;

**public** **interface** FortuneService {

**public** String getFortune();

}

**File: HappyFortuneService.java**:

**package** com.ruhul.odduu.fieldinjection;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** HappyFortuneService **implements** FortuneService {

@Override

**public** String getFortune() {

**return** "Today is your lucky day!!!";

}

}

**File: TennisCoach.java**:

**package** com.ruhul.odduu.fieldinjection;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** TennisCoach **implements** Coach {

//field injection

@Autowired

**private** FortuneService fortuneService;

// default constructor

**public** TennisCoach() {

System.***out***.println(">> TennisCoach: inside default constructor");

}

@Override

**public** String getDailyWorout() {

**return** "Practice your backhand volley";

}

@Override

**public** String getDailyFortune() {

**return** fortuneService.getFortune();

}

}

**File: AnnotationDemoApp.java**:

**package** com.ruhul.odduu.fieldinjection;

**import** org.springframework.context.support.ClassPathXmlApplicationContext;

**public** **class** AnnotationDemoApp {

**public** **static** **void** main(String[] args) {

// read spring congig file

ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContextField.xml");

// get the bean from container

Coach theCoach = context.getBean("tennisCoach", Coach.**class**);

// call a method on the bean

System.***out***.println(theCoach.getDailyWorout());

// call method to get the daily fortune

System.***out***.println(theCoach.getDailyFortune());

// close the context

context.close();

}

}

**File: applicationContextField.xml**:

<?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: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.xsd"*>

<!-- add entry to enable component scanning -->

<context:component-scan

base-package=*"com.ruhul.odduu.fieldinjection"* />

</beans>

**Output**:

>> TennisCoach: inside default constructor

Practice your backhand volley

Today is your lucky day!!!

Field Injection