



```
import org.springframework.stereotype.Component;

@Component
public class MyService {
    public void doSomething() {
        System.out.println("Doing something!");
    }
}
```

## الآن نكتب كود استخدام

مع @Bean. نحدد و @Configuration مع كود فئة أنشئ

مثال:

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {
    @Bean
    public MyService myService() {
        return new MyService();
    }
}
```

## قائمة طرق استخدام

beans.xml: مثال ملف في

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="myService" class="com.example.MyService"/>
</beans>
```

## 4. إطلاق

لإدارة الحاوية بدء عمليك

## مع Spring

مع `@SpringBootApplication`: ذلك يعني فقط تلقائيًا. لذلك فعل

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
public class MyApp {
    public static void main(String[] args) {
        SpringApplication.run(MyApp.class, args);
    }
}
```

وتدبره. `@Component` فئات عن تبحت الحاوية

## تعليمات لتكوين العادي مع Spring

مع `AnnotationConfigApplicationContext`:

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context =
            new AnnotationConfigApplicationContext(AppConfig.class);
        MyService service = context.getBean(MyService.class);
        service.doSomething();
        context.close();
    }
}
```

## مع Spring

مع `ClassPathXmlApplicationContext`:

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

public class Main {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context =
            new ClassPathXmlApplicationContext("beans.xml");
    }
}
```

```

        MyService service = context.getBean("myService", MyService.class);
        service.doSomething();
        context.close();
    }
}

```

---

## 5. الاعتماديات حقن

الاعتماديات حقن في الـ `MyService` في الـ `MyController` للحاوية يمكن

به `MyService` الموصى البناء حقن

```

@Component
public class MyController {
    private final MyService myService;

    public MyController(MyService myService) {
        this.myService = myService;
    }

    public void execute() {
        myService.doSomething();
    }
}

```

المحدد حقن

```

@Component
public class MyController {
    private MyService myService;

    @Autowired
    public void setMyService(MyService myService) {
        this.myService = myService;
    }
}

```

## تفصيل أقل الحقول حقن

```
@Component
public class MyController {
    @Autowired
    private MyService myService;

    public void execute() {
        myService.doSomething();
    }
}
```

الحقن. نقاط لتحديد 330 مبرمجين @Inject أو مبرمجين @Autowired استخدم

## 6. استخدام تشغيّل

context.getBean() مبرمجين باستخدام مبرمجين استرجاع يمكن اعتمادياتها. و مبرمجين تربط الحاوية، تشغيّل بعد الحقن. عبر تلقائيًا ذلك بإدارة مبرمجين مبرمجين ل- السماح أو العادي

مبرمجين مبرمجين:

```
@SpringBootApplication
public class MyApp {
    public static void main(String[] args) {
        ApplicationContext context = SpringApplication.run(MyApp.class, args);
        MyController controller = context.getBean(MyController.class);
        controller.execute();
    }
}
```

## 7. إضافة نصائح

لكل جديدة نسخ على للحصول @Scope("prototype") استخدم. نوع من هي مبرمجين مبرمجين، الارتفاع طلب.

مختلفة. حزم في مبرمجين كانت إذا التكوين فئة إلى @ComponentScan أضف: المكونات فحص

مبرمجين، مبرمجين. مبرمجين مبرمجين مبرمجين مبرمجين @Profile استخدم: المملفات

الموارد. لتحديد context.close() استخدم، مبرمجين مبرمجين مبرمجين مبرمجين: السيق إغلاق

