

# 第2章 Spring Boot 配置

● ————— ●  
《Spring Boot企业级开发教程（第2版）》



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌

# 学习目标/Target



掌握`application.properties`配置文件，能够在`application.properties`配置文件中正确配置数据

掌握`application.yml`配置文件，能够在`application.yml`配置文件中正确配置数据

掌握`@Value`注解，能够使用`@Value`注解为Bean的属性绑定配置数据

熟悉`Environment`对象，能够使用`Environment`对象获取全局配置文件中的属性

# 学习目标/Target



掌握@ConfigurationProperties注解，能够使用@ConfigurationProperties注解为Bean的属性绑定配置数据

了解@Value和@ConfigurationProperties对比分析，能够说出@Value和@ConfigurationProperties的主要区别

掌握引入配置文件，能够使用@PropertySource注解和@ImportResource注解引入配置文件

掌握定义配置类，能够使用@Configuration注解定义配置类

# 学习目标/Target

---



熟悉单一文件中配置Profile，能够在单一文件中配置Profile以实现多环境配置

掌握多文件中配置Profile，能够在多文件中配置Profile以实现多环境配置

熟悉@Profile注解，能够正确使用@Profile注解进行多环境配置



Spring Boot极大地简化了Spring应用的开发，尤其是Spring Boot的自动配置功能，该功能使项目即使不进行任何配置，也能顺利运行。当用户想要根据自身需求覆盖Spring Boot的默认配置时，需要使用配置文件修改Spring Boot的默认配置。本章将对Spring Boot的配置进行讲解。



2.1

全局配置文件

2.2

配置绑定

2.3

引入配置文件和定义配置类

2.4

Profile



## 2.1

# 全局配置文件



全局配置文件能够对一些默认配置值进行修改。Spring Boot默认使用的全局配置文件有application.properties和application.yml，Spring Boot启动时会自动读取这两个文件中的配置，如果文件中存在与默认自动配置相同的配置信息，则覆盖默认的配置信息。下面对全局配置文件进行讲解。



## >>> 2.1.1 application.properties配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌



掌握application.properties配置文件,  
能够在application.properties配置文  
件中正确配置数据

## 2.1.1 application.properties配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

`application.properties`文件中可以定义Spring Boot项目的相关属性，属性可采用键值对格式进行设置，表示形式为“`Key=Value`”，这些相关属性可以是系统属性、环境变量、命令参数等信息，也可以是自定义的属性。

```
address=beijing  
server.port=80  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

`application.properties`文件中的属性支持多种类型，常见的有字面量、数组和集合。



### 1. 字面量类型属性

字面量是指单个的不可拆分的值，例如：数字、字符串、布尔值等。在application.properties文件中配置字面量的属性时，直接将字面量作为Value写在键值对中即可，且默认情况下字符串是不需要使用单引号或双引号进行修饰的。

```
address=beijing  
age=13
```



### 1. 字面量类型属性

如果需要配置的属性为对象的属性，可以通过“对象名.属性名”的方式指定属性的键。对象中可能包含多个属性，在application.properties文件中为对象的属性赋值时，一个属性对应一对键值对。

```
user.username=lisi  
user.age=18
```



### 2.数组类型属性

在application.properties文件中配置数组类型属性时，可以将数组元素的值写在一行内，元素值之间使用逗号（,）间隔，也可以在多行分别根据索引赋值。

# 方式一

user.hobby=swim,travel,cook

# 方式二

user.hobby[0]=swim2

user.hobby[1]=travel2

user.hobby[2]=cook2



### 3.集合类型属性

在application.properties文件中也可以配置集合类型的属性，下面分别演示配置List、Set、Map的集合类型属性。

```
# 配置List:方式一
user.subject=Chinese,English,Math
# 配置List:方式二
user.subject[0]=Chinese
user.subject[1]=English
user.subject[2]=Math
```



### 3.集合类型属性

# 配置Set

user.salary=120,230

# 配置Map方式一

user.order.1001=cookie

user.order.1002=cake

# 配置Map方式二

user.order[1001]=cookie

user.order[1002]=cake



掌握application.yml配置文件，能够在application.yml配置文件中正确配置数据





`application.yml`配置文件是使用 **YAML**编写的文件，YAML是“YAML Ain't Markup Language”的递归缩写。YAML通常用于表示**数据结构**和**配置信息**，它使用**缩进**和**外观依赖**的方式表示**层级关系**，使得配置文件和数据结构的表达相对简洁和易于阅读。**YAML**支持的数据包括**列表**、**键值对**和**字符串**、**数字**等。

YAML文件的**后缀名**为**.yml**或**.yaml**，编写时需要遵循如下**规则**。

- 使用**缩进**表示**层级关系**。
- **缩进时不允许使用Tab 键**，只允许使用**空格**。
- 缩进的空格数不重要，但**同级元素**必须**左侧对齐**。
- **大小写敏感**。



### 1. 字面量类型属性

字面量是指单个的，不可拆分的值，例如：数字、字符串、布尔值等。在application.properties文YAML中，使用“Key: Value”的形式表示一对键值对，其中Value前面有一个空格，并且该空格不能省略。在配置字面量类型的属性时，直接将字面量作为Value直接写在键值对中即可，且默认情况下字符串是不需要使用单引号或双引号的。

```
address: beijing  
age: 13
```



### 1. 字面量类型属性

如果需要配置的属性为对象的属性，配置的方式有缩进式和行内式两种。

# 缩进式

consumer:

username: lisi

age: 18

# 行内式

consumer: {username: lisi, age: 18}



### 2.数组类型和单列集合属性

当YAML配置文件中配置的属性为数组类型或单列集合时，也可以使用缩进式写法和行内式写法。

```
consumer:
```

```
  hobby:
```

```
    - play
```

```
    - read
```

```
    - sleep
```

缩进式

or

```
consumer:
```

```
  hobby: [play,read,sleep]
```

行内式



### 3.Map集合属性

当YAML配置文件中配置的属性为Map集合时，可以使用缩进式写法和行内式写法。

```
consumer:  
  order:  
    1001: cookie  
    1002: cake
```

缩进式

or

```
consumer:  
  order: {1001: cookie,1002: cake}
```

行内式



## 默认配置文件

Spring Boot项目将application.properties或application.yml作为项目的默认配置文件。Spring Boot项目中可以存在多个application.properties或application.yml，Spring Boot 启动时会扫描以下5个位置的application.properties和application.yml文件，并将扫描到的文件作为Spring Boot 的默认配置文件。

- ① file:./config/\*/
- ② file:./config/
- ③ file:./
- ④ classpath:/config/
- ⑤ classpath:/



## 默认配置文件

Spring Boot项目将application.properties或application.yml作为项目的默认配置文件。Spring 上述5个位置下如果存在application.properties和application.yml文件，在项目启动就都会被加载。加载多个application.properties或application.yml文件时，文件中的配置会根据文件所处的位置划分优先级，优先级规则如下。

- 上述位置1~位置5的优先级依次降低，序号越小优先级越高。
- 位于相同位置的application.properties的优先级高于application.yml，application.yml的优先级高于application.yaml。
- 存在相同的配置内容时，高优先级的内容会覆盖低优先级的内容。
- 存在不同的配置内容时，高优先级和低优先级的配置内容取并集。

2.2

## 配置绑定







使用Spring Boot全局配置文件配置属性时，如果配置的属性是Spring Boot内置的属性（如服务端口server.port），那么Spring Boot会自动扫描并读取配置文件中的属性值并覆盖原有默认的属性值。如果配置的属性是用户自定义的属性，可以通过Java代码去读取该配置属性，并且把属性绑定到Bean。在Spring Boot项目中可以通过 @Value、Environment对象和@ConfigurationProperties对配置属性进行绑定，下面分别对这三种方式实现配置绑定进行讲解。



掌握@Value注解，能够使用@Value  
注解为Bean的属性绑定配置数据



@Value注解是由Spring框架提供的，Spring Boot框架从Spring框架中对@Value注解进行了默认继承，通过@Value可以将配置文件中的属性绑定到Bean对象对应的属性。

```
@Component  
public class Person {  
    @Value("${person.id}")  
    private int id;  
}
```

将配置文件中属性person.id的值动态注入到id属性



下面通过案例演示在Spring Boot项目中使用@Value绑定全局配置文件中的属性。

(1) **创建实体类**。在IDEA中创建一个Spring Boot项目，在项目的java文件夹下创建类包com.itheima.domain，并在该类包下创建一个消费者**实体类Consumer**，在该类上使用@Component进行标注，并在属性上使用@Value注解注入配置文件中的属性，具体如文件2-1所示。



### 源代码

文件2-1

Consumer.java





## 2.2.1 @Value注解



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

(2) 添加配置信息。在项目的resource文件夹下创建配置文件application.yml，在配置文件中添加属性信息，具体如文件2-2所示。



源代码

文件2-2

application.yml





(3) **创建测试类**。在项目test文件夹下，创建类包com.itheima的类包，在该包下创建**测试类** **Chapter02ApplicationTests**，在该测试类中**注入Consumer对象**，并新增一个测试方法进行**输出测试**，具体代码如文件2-3所示。



### 源代码

文件2-3

[Chapter02ApplicationTests .java](#)





(4) 测试程序效果。运行测试方法wiredTest()。

The screenshot shows an IDE's test runner interface. At the top, a status bar indicates 'Tests passed: 1 of 1 test - 572ms'. Below this, a list of test results is displayed. The first result, 'Chapt', is highlighted in blue and shows a green checkmark and a duration of 572ms. To the right of this, two log entries are visible: '2022-10-12 18:23:15.091 INFO 10284 --- [main] com.itheima.Chapter02Appl.' and '2022-10-12 18:23:17.983 INFO 10284 --- [main] com.itheima.Chapter02Appl.'. Below these, a JSON-like object is printed: 'Consumer{username='lisi', age=23, hobby=[sing, read, sleep], subject=[100, 150]}'. The interface includes standard IDE icons for running and debugging tests.

```
>> ✓ Tests passed: 1 of 1 test - 572ms
✓ Chapt 572ms 2022-10-12 18:23:15.091 INFO 10284 --- [main] com.itheima.Chapter02Appl.
2022-10-12 18:23:17.983 INFO 10284 --- [main] com.itheima.Chapter02Appl.
Consumer{username='lisi', age=23, hobby=[sing, read, sleep], subject=[100, 150]}
```



熟悉Environment对象，能够使用Environment对象获取全局配置文件中的属性





使用@Value注解时，将该注解标注在Spring管控的Bean的属性名上方，就可以将某个数据绑定到Bean对象的属性。当Bean的属性比较多且这些属性都需要绑定配置的数据时，操作起来就比较烦琐。为此，Spring Boot提供了一个对象Environment，项目启动时能够将配置文件中的所有数据都封装到该对象中，这样就不需要手动对配置数据进行绑定。



使用Environment对象获取配置文件的数据时，不需要在提供其他实体类。在文件2-3的Chapter02ApplicationTests 类中，通过@Autowired注入Environment对象，并新增测试方法evnTest()，在测试方式中通过Environment对象获取配置文件中的属性。

```
@Autowired
```

```
private Environment env;
```

```
@Test
```

```
void evnTest() {
```

```
    System.out.println("consumer.username=" + env.getProperty("consumer.username"));
```

```
    System.out.println("consumer.age=" + env.getProperty("consumer.age"));
```

```
    System.out.println("consumer.hobby=" + env.getProperty("consumer.hobby"));
```

```
    System.out.println("consumer.subject=" + env.getProperty("consumer.subject"));
```

```
}
```

getProperty()方法获取封装到Environment对象  
中封装的配置文件的数据

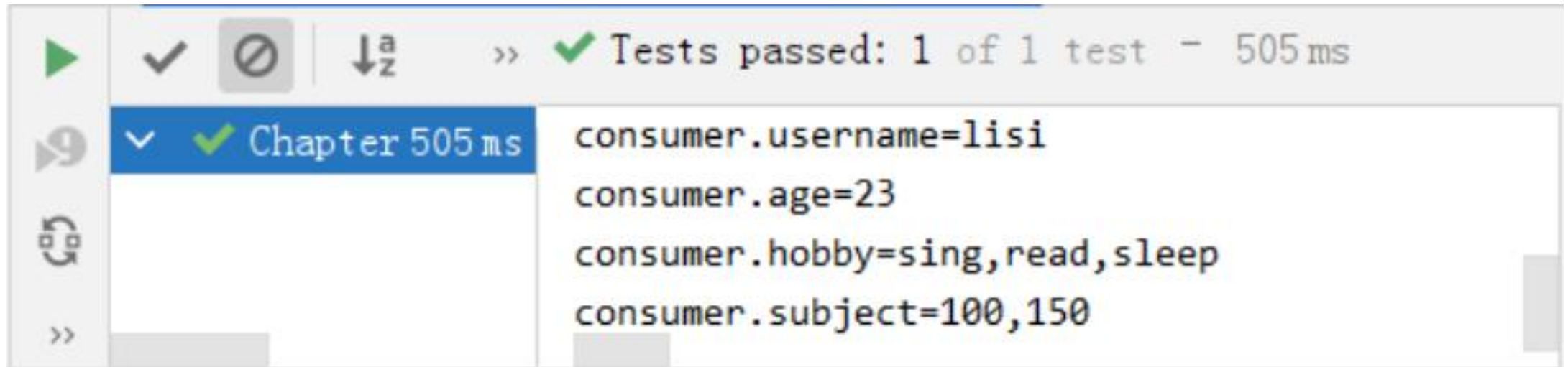
## >>> 2.2.2 Environment对象



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

运行测试方法evnTest(), 输出封装到Environment对象中的配置文件的数据。





掌握@ConfigurationProperties注解，  
能够使用@ConfigurationProperties  
注解为Bean的属性绑定配置数据



Java是面向对象的语言，很多情况下，人们习惯将具有相同特性的一组数据封装到一个对象中，Spring Boot中就提供了这样的注解。Spring Boot的@ConfigurationProperties注解可以 可以将配置文件中的一组配置数据同时绑定到Bean中。

下面通过案例演示在Spring Boot项目中使用@ConfigurationProperties注解绑定全局配置文件中的数据。

## >>> 2.2.3 @ConfigurationProperties注解



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

(1) 修改实体类属性的绑定方式。在文件2-1的Consumer类上使用@ConfigurationProperties注解进行标注，并且去除属性上方标注的@Value注解。

```
@Component
@ConfigurationProperties(prefix = "consumer")
public class Consumer {
    private String username;
    private int age;
    private String[] hobby;
    private List subject;
    //.....setter/getter方法, 以及toString()方法
}
```

类中的属性名需要和绑定的配置文件中属性名保持一致。



(2) 新增测试方法。在文件2-3的Chapter02ApplicationTests 类中，新增测试方法confTest()，在测试方式中输出Consumer对象。

```
@Test
void confTest() {
    System.out.println(consumer);
}
```

## >>> 2.2.3 @ConfigurationProperties注解



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

(3) 测试程序效果。运行测试方法confTest()。

```
>> ✓ Tests passed: 1 of 1 test - 548ms  
✓ Chapter02Ap 548 ms  
2022-10-13 11:28:16.893 INFO 11420 --- [main] com.itheima.Chapter02Appl  
2022-10-13 11:28:19.993 INFO 11420 --- [main] com.itheima.Chapter02Appl  
Consumer{username='lisi', age=23, hobby=[sing, read, sleep], subject=[100, 150]}
```





## 2.2.4 @Value和@ConfigurationProperties对比分析



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌



了解@Value和@ConfigurationProperties对比分析，能够说出@Value和@ConfigurationProperties的主要区别

使用@Value注解时，将该注解标注在Spring管控的Bean的属性名上方，就可以将某个数通过前面的学习可以了解到，@Value注解和@ConfigurationProperties注解都可以对配置文件中的属性进行绑定，但两者在使用过程中还是有一些差异的。为了进一步了解两者的不同，下面对@Value注解和@ConfigurationProperties注解进行对比分析。

对比项	@Value	@ConfigurationProperties
底层框架	Spring	Spring Boot
功能	单个注入配置文件中的属性	批量注入配置文件中的属性
为属性设置setter方法	不需要	需要
复杂类型属性注入	不支持	支持
松散绑定	不支持	支持
JSR303数据校验	不支持	支持
SpEL表达式	支持	不支持



### 1. 底层框架

@Value注解：由Spring框架提供。

@ConfigurationProperties注解：由Spring Boot框架提供。

### 2. 功能

@Value注解：只在需要注入属性值的单个属性上进行注入配置。

@ConfigurationProperties注解：主要用于将配置文件中某一类属性整体批量读取并注入到Bean的属性中。



### 3. 为属性设置setter方法

@Value注解：不需要为属性设置setter方法。

@ConfigurationProperties注解：必须为每一个属性设置setter方法。

### 4. 复杂类型属性注入

@Value注解：无法解析，导致注入失败。

@ConfigurationProperties注解：支持任意数据类型的属性注入。



### 5. 松散绑定

@Value注解：不支持松散绑定语法。

@ConfigurationProperties注解：支持松散绑定语法。例如Person类有一个字符串类型的属性firstName，可以绑定配置文件中的如下属性。

```
person.firstName=james    // 标准写法，对应Person类属性名
person.first-name=james   // 使用横线-分隔多个单词
person.first_name=james   // 使用下划线_分隔多个单词
PERSON.FIRST_NAME=james  // 使用大小写格式，推荐常量属性配置
```



### 6. JSR303数据校验

JSR303数据校验的主要作用是校验配置文件中注入到对应Bean属性的值是否符合相关值的规则。

@Value注解：不支持JSR303数据校验。

@ConfigurationProperties注解：支持JSR303数据校验，示例如下。

```
@Component
@ConfigurationProperties(prefix = "person")
@Validated    // 引入Spring框架支持的数据校验规则
public class Example {
    @Email    // 对属性进行规则匹配
    private String email;
    public void setEmail(String email) {
        this.email = email;
    }
}
```



### 7. SpEL表达式

@Value注解：支持SpEL表达式语法，例如Person类有一个整数类型的属性id，直接使用SpEL表达式语法进行属性注入。

```
@Value("#{5*2}") // 使用@Value注解的SpEL表达式直接为属性注入值  
private int id;
```

@ConfigurationProperties注解：不支持SpEL表达式语法。

@Value和@ConfigurationProperties两种注解没有明显的优劣之分，它们只是适合的应用场景不同而已，不同场景下的使用推荐如下。

- 如果只是针对某一个业务需求，要引入配置文件中的个别属性，推荐使用@Value注解；
- 如果针对某个JavaBean类，需要批量注入属性，则推荐使用@ConfigurationProperties注解。



## 2.3

# 引入配置文件和定义配置类





## 2.3 引入配置文件和定义配置类



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

虽然Spring Boot免除了项目中大部分的手动配置，对于一些特定情况，可以通过修改全局配置文件以适应具体的开发或生产环境，但是有时候项目中不可避免地要使用默认配置文件之外的配置信息，这个时候就需要手动引入配置文件或配置类。下面分别对在Spring Boot项目中引入配置文件和配置类进行讲解。



掌握引入配置文件，能够使用  
`@PropertySource`注解和  
`@ImportResource`注解引入配置文件



Spring Boot项目中引入的配置文件通常有两类，第一类为YAML或properties的属性配置文件；第二类为XML配置文件。一般第一类配置文件可以使用@PropertySource引入，第二类配置文件可以使用@ImportResource引入，下面分别对使用这两个注解引入配置文件进行讲解。



## 2.3.1 引入配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

### 1.使用@PropertySource引入属性配置文件

@PropertySource注解标注在类上，可以指定引入的配置文件的位置和名称。如果需要将自定义配置文件中的属性值注入到对应类的属性中，可以使用@ConfigurationProperties或者@Value注解进行注入。

下面通过案例演示在Spring Boot项目中使用@PropertySource引入属性配置文件。



## 2.3.1 引入配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

### 1.使用@PropertySource引入属性配置文件

(1) **创建配置文件**。在项目chapter02的resources目录下创建自定义配置文件`user.properties`，在该配置文件中编写需要设置的**属性**，具体如文件2-4所示。



#### 源代码

文件2-4

`user.properties`





### 1.使用@PropertySource引入属性配置文件

(2) **创建实体类**。在com.itheima.domain包下创建**用户实体类User**，在类上使用@PropertySource引入配置文件user.properties，并使用@ConfigurationProperties注解将配置文件中的属性绑定到类的属性上，具体如文件2-5所示。



#### 源代码

文件2-5  
User.java





### 1.使用@PropertySource引入属性配置文件

(3) **新增测试方法**。在文件2-3的Chapter02ApplicationTests 类中注入User对象，并**新增测试方法 propTest()**，在测试方式中**输出User对象**，具体代码如下。

```
@Autowired
private User user;

@Test
void propTest() {
    System.out.println(user);
}
```



### 1.使用@PropertySource引入属性配置文件

(4) 测试程序效果。运行测试方法propTest()。

The screenshot shows an IDE's test runner interface. At the top, a status bar indicates 'Tests passed: 1 of 1 test - 559 ms'. Below this, a tree view shows a test class 'Chapter' with a duration of '559 ms'. The main output pane displays the following log messages:

```
2022-10-13 14:31:42.017 INFO 13824 --- |
2022-10-13 14:31:42.020 INFO 13824 --- |
2022-10-13 14:31:45.005 INFO 13824 --- |
User{id='1001', nickname='wangwu'}
```





## 2.3.1 引入配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

### 2. 使用@ImportResource引入XML配置文件

传统Spring框架大多采用XML文件作为配置文件，但Spring Boot推荐使用Java配置类进行配置，Spring Boot默认不能自动识别XML配置文件，想让Spring的XML配置文件生效，可以使用@ImportResource注解加载XML配置文件。

@ImportResource注解标注在一个配置类上，使用时需要指定引入XML配置文件的路径和名称。下面通过案例演示在Spring Boot项目中使用@ImportResource引入XML配置文件。



## 2.3.1 引入配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

### 2. 使用@ImportResource引入XML配置文件

(1) 创建**组件类**。在com.itheima.service包下创建类**MyService**，在类中**定义方法**用于后续测试，具体如文件2-6所示。



#### 源代码

文件2-6  
[MyService.java](#)





## 2.3.1 引入配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

### 2. 使用@ImportResource引入XML配置文件

(2) 创建XML配置文件。resources文件夹下创建配置文件，在该配置文件中声明Bean，具体如文件2-7所示。



#### 源代码

文件2-7  
[beans.xml](#)





### 2. 使用@ImportResource引入XML配置文件

(3) **添加@ImportResource注解**。编写完Spring的XML配置文件后，Spring Boot默认不会自动引入，为了保证XML配置文件生效，需要在项目启动类Chapter02Application上添加@ImportResource注解来指定XML文件的位置，内容如文件2-8所示。



#### 源代码

文件2-8

[Chapter02Application.java](#)





## 2.3.1 引入配置文件



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

### 2. 使用@ImportResource引入XML配置文件

(4) 新增测试方法。在文件2-3的Chapter02ApplicationTests类中注入MyService对象，并新增测试方法beanTest()，在测试方式中使用MyService对象调用getById()方法，具体代码如下。

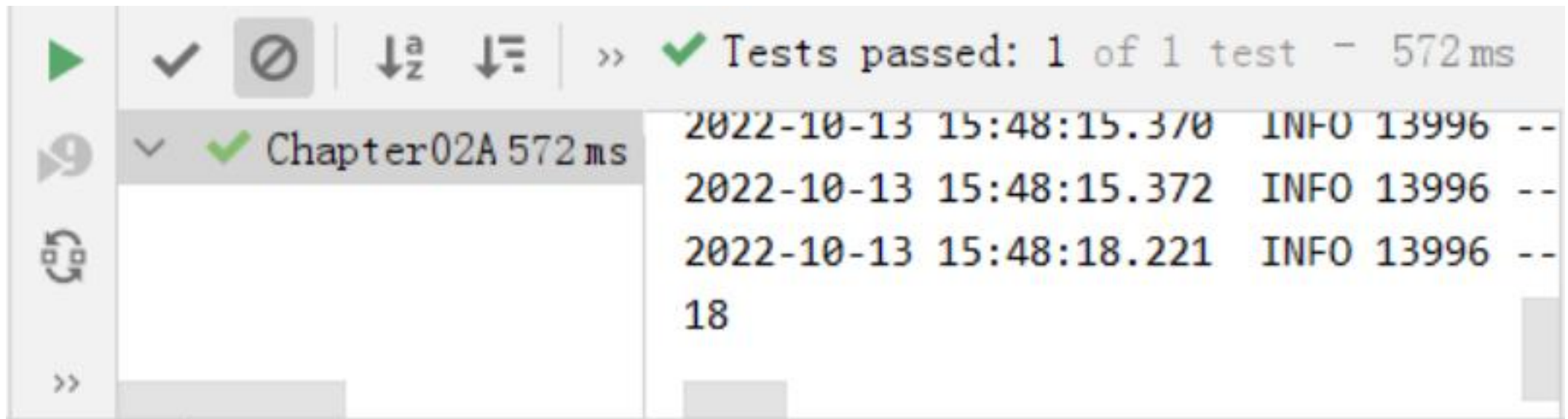
```
@Autowired
private MyService myService;

@Test
void beanTest() {
    myService.getById("18");
}
```



### 2. 使用@ImportResource引入XML配置文件

(5) 测试程序效果。运行测试方法beanTest()。





掌握定义配置类，能够使用  
`@Configuration`注解定义配置类



当自动配置不能满足我们的需求的时候，通常会通过配置类来对自定义Bean进行Bean容器的装配工作。添加@Configuration注解的类称之为配置类，配置类主要用于替代原来的配置文件。

当定义一个配置类后，还需要在类中的方法上使用@Bean注解进行组件配置，将方法的返回对象注入到Spring容器中。组件名称默认使用的是方法名，也可以使用@Bean注解的name或value属性自定义组件的名称。引入配置类时，只需让配置类被Spring Boot自动扫描识别即可，该配置类中返回的组件会自动添加到Spring容器中。





下面通过案例演示在Spring Boot项目中定义配置类，具体如下。

(1) **创建配置类**。在com.itheima.config包下创建MyConfig类，并使用@Configuration注解将该类声明为一个配置类，具体如文件2-9所示。

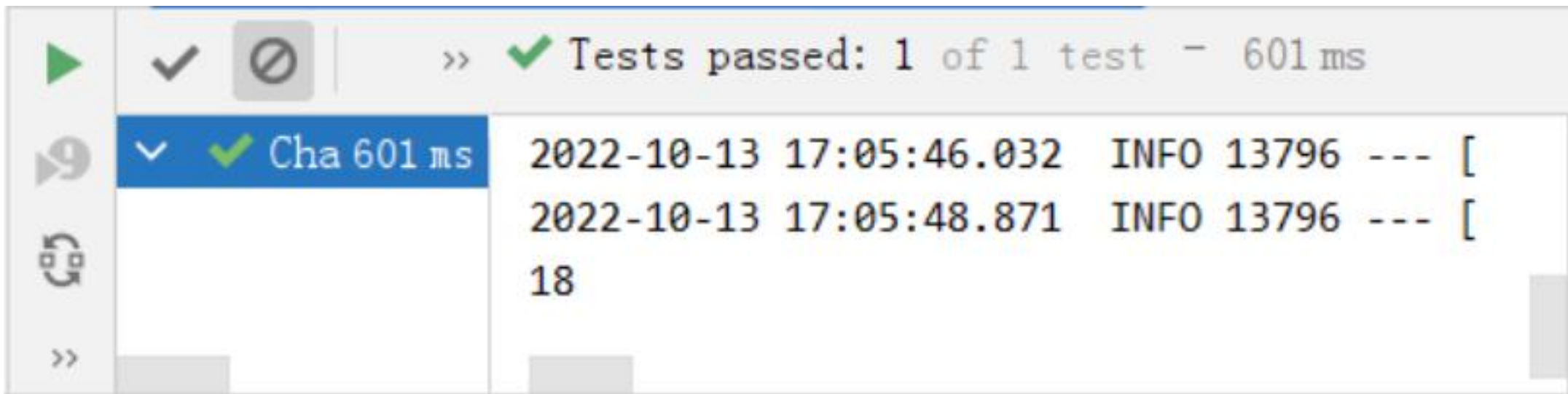


### 源代码

文件2-9  
MyConfig.java



(2) 测试程序效果。本案例使用@Configuration和@Bean将MyService对象添加到Spring容器中，对此可以将文件2-9中项目启动类Chapter02Application上添加的@ImportResource注解注释，运行测试方法beanTest()。





在实际开发中，根据项目的开发进度，项目经常需要在不同的部署环境间切换，常见部署的环境有开发环境、测试环境、生产环境。不同环境使用的配置信息往往不同，而且项目的配置信息往往有很多，如果每次变更项目部署的环境时，都采用手动方式更改配置信息会很麻烦。

针对这种情况，在Spring Boot中可以使用Profile解决这类问题，Profile使Spring Boot可以针对不同的环境提供不同的配置。在Spring Boot中可以将Profile配置在单一文件和多个文件中，也可以通过@Profile注解指定Bean的生效环境。下面对Profile的使用进行讲解。

## >>> 2.4.1 单一文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌



熟悉单一文件中配置Profile，能够在单一文件中配置Profile以实现多环境配置



Spring Boot中可以在配置文件使用 `spring.config.activate.on-profile`指定Profile的名称, 使用`spring.profiles.active`指定激活哪个Profile, 如果需要激活多个Profile, Profile名称之间使用逗号间隔即可。

每个Profile中的配置信息都对应于一个部署环境, 在单一YAML文件配置多个Profile时, 可以通过三个短横线号 (---) 将不同的Profile分隔开。

## >>> 2.4.1 单一文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

下面通过案例演示在单一文件中配置Profile。

(1) 配置Profile。在项目chapter02的application.yml配置文件中配置3个Profile，名称分别为dev、test、pro，表示开发环境、测试环境和生产环境，具体配置信息如文件2-10所示。



源代码

文件2-10  
[application.yml](#)



## >>> 2.4.1 单一文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

(2) 创建控制器类。在com.itheima.controller包下创建控制器类DefaultController，在类中注入Environment对象，并定义获取项目服务端口的方法，具体如文件2-11所示。



源代码

文件2-11

[DefaultController.java](#)



## >>> 2.4.1 单一文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

(3) 测试程序效果。启动项目，控制台输出对应的信息。

```
Run: Chapter02Application x
Console
2022-10-14 14:48:32.676 INFO 15252 --- [ restartedMain] com.itheima.Chapter02Application : Starting Chapter02Application using Java 11.0.16 on t
2022-10-14 14:48:32.678 INFO 15252 --- [ restartedMain] com.itheima.Chapter02Application : The following 1 profile is active: "dev"
2022-10-14 14:48:32.791 INFO 15252 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devto
2022-10-14 14:48:32.791 INFO 15252 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting t
2022-10-14 14:48:33.932 INFO 15252 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 80 (http)
2022-10-14 14:48:33.942 INFO 15252 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-14 14:48:33.943 INFO 15252 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-10-14 14:48:34.004 INFO 15252 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-14 14:48:34.004 INFO 15252 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-10-14 14:48:34.401 INFO 15252 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-10-14 14:48:34.461 INFO 15252 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 80 (http) with context pat
2022-10-14 14:48:34.474 INFO 15252 --- [ restartedMain] com.itheima.Chapter02Application : Started Chapter02Application in 2.483 seconds (JVM ru
```



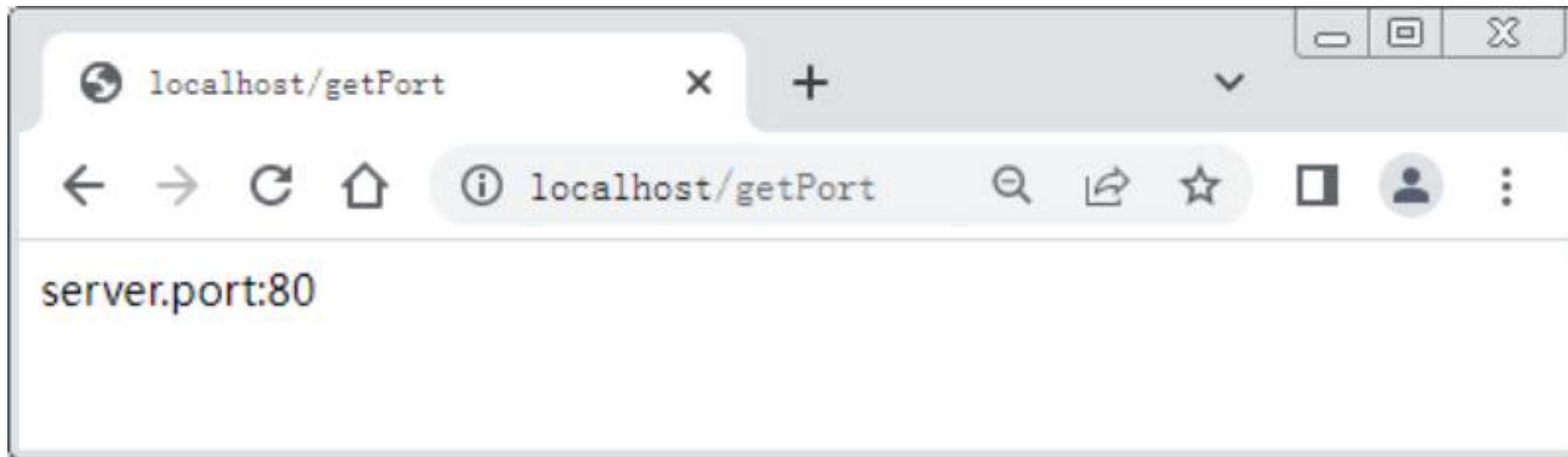
## 2.4.1 单一文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

在浏览器访问<http://localhost/getPort>。



## >>> 2.4.1 单一文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

如果将文件2-11中激活的Profile修改为pro，再次启动项目时，项目控制台输出pro对应的信息。

```
Console  Actuator
2022-10-14 15:09:59.134 INFO 14732 --- [ restartedMain] com.itheima.Chapter02Application : Starting Chapter02Application using Java 11.0.16 on t
2022-10-14 15:09:59.136 INFO 14732 --- [ restartedMain] com.itheima.Chapter02Application : The following 1 profile is active: "pro"
2022-10-14 15:09:59.223 INFO 14732 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devto
2022-10-14 15:09:59.224 INFO 14732 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting t
2022-10-14 15:10:00.801 INFO 14732 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 82 (http)
2022-10-14 15:10:00.811 INFO 14732 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-14 15:10:00.811 INFO 14732 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-10-14 15:10:00.882 INFO 14732 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-14 15:10:00.882 INFO 14732 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-10-14 15:10:01.248 INFO 14732 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-10-14 15:10:01.306 INFO 14732 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 82 (http) with context pat
2022-10-14 15:10:01.320 INFO 14732 --- [ restartedMain] com.itheima.Chapter02Application : Started Chapter02Application in 2.9 seconds (JVM runn
```



掌握多文件中配置Profile，能够在多文件中配置Profile以实现多环境配置



实际开发中，项目中通常会包含多个组件或框架，如果将所有的配置信息都放在一个配置文件中，尤其是配置的部署环境都不一样时，配置文件会非常臃肿，不便与维护。针对此种情况，可以将一个配置文件拆分成多个配置文件。拆分后，可在不同的配置文件中不同环境的配置，主配置文件中指定激活的Profile。

拆分出的配置文件的名称格式为application-{profile}.yaml或application-{profile}.properties，其中{profile}对应具体环境标示的Profile名称。例如，YAML格式的开发环境、测试环境和生产环境配置文件命名如下。

application-dev.yml	// 开发环境配置文件
application-test.yml	// 测试环境配置文件
application-pro.yml	// 生产环境配置文件

## >>> 2.4.2 多文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

下面通过案例演示在多文件中配置Profile，具体如下。

(1) **拆分配置文件**。将文件2-11中开发环境、测试环境、生产环境的Profile拆分为3个文件，具体如文件2-12~文件2-14所示。



### 源代码

文件2-12 [application-dev.yml](#)

文件2-13 [application-test.yml](#)

文件2-14 [application-pro.yml](#)



## >>> 2.4.2 多文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

(2) 测试程序效果。此时项目`application.yml`配置文件中指定激活的Profile为`pro`，启动项目，控制台输出`pro`的信息。

```
Run: Chapter02Application x
Console
2022-10-14 15:47:39.490 INFO 14312 --- [ restartedMain] com.itheima.Chapter02Application : Starting Chapter02Application using Java 11.0.16 on t
2022-10-14 15:47:39.492 INFO 14312 --- [ restartedMain] com.itheima.Chapter02Application : The following 1 profile is active: "pro"
2022-10-14 15:47:39.579 INFO 14312 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devto
2022-10-14 15:47:39.579 INFO 14312 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting t
2022-10-14 15:47:40.848 INFO 14312 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 82 (http)
2022-10-14 15:47:40.861 INFO 14312 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-14 15:47:40.861 INFO 14312 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-10-14 15:47:40.937 INFO 14312 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-14 15:47:40.937 INFO 14312 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-10-14 15:47:41.388 INFO 14312 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-10-14 15:47:41.456 INFO 14312 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 82 (http) with context pat
2022-10-14 15:47:41.470 INFO 14312 --- [ restartedMain] com.itheima.Chapter02Application : Started Chapter02Application in 2.729 seconds (JVM ru
```



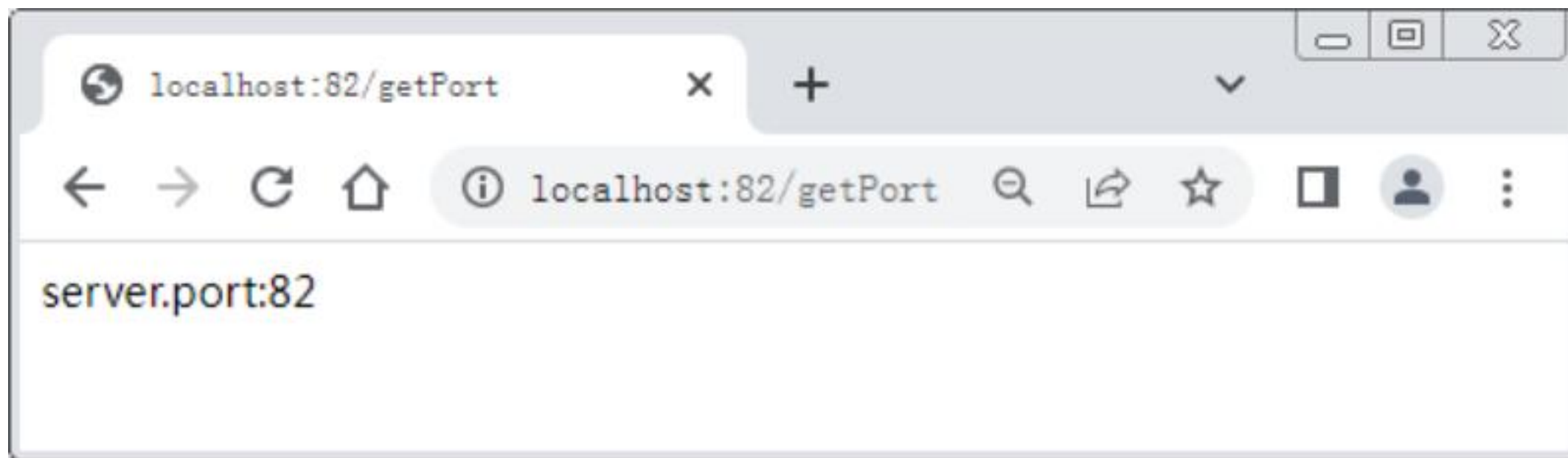
## >>> 2.4.2 多文件中配置Profile



黑马程序员  
www.itheima.com

传智教育旗下  
高端IT教育品牌

在浏览器访问<http://localhost:82/getPort>。





如果将`application.yml`中激活的Profile修改为`test`，再次启动项目时，项目控制台输出`test`的相关信息。



```
2022-10-14 15:51:09.125 INFO 11444 --- [ restartedMain] com.itheima.Chapter02Application : Starting Chapter02Application using Java 11.0.16 on t
2022-10-14 15:51:09.127 INFO 11444 --- [ restartedMain] com.itheima.Chapter02Application : The following 1 profile is active: "test"
2022-10-14 15:51:09.199 INFO 11444 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtool
2022-10-14 15:51:09.199 INFO 11444 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting t
2022-10-14 15:51:10.425 INFO 11444 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 81 (http)
2022-10-14 15:51:10.436 INFO 11444 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [tomcat]
2022-10-14 15:51:10.436 INFO 11444 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-10-14 15:51:10.495 INFO 11444 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-14 15:51:10.496 INFO 11444 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-10-14 15:51:10.875 INFO 11444 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-10-14 15:51:10.928 INFO 11444 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 81 (http) with context pat
2022-10-14 15:51:10.946 INFO 11444 --- [ restartedMain] com.itheima.Chapter02Application : Started Chapter02Application in 2.495 seconds (JVM ru
```





熟悉@Profile注解，能够正确使用  
@Profile注解进行多环境配置



默认情况下，项目启动后，所有的Bean在任何环境下都可以生效，如果想要指定某个Bean只在特定的配置环境下生效，可以使用@Profile注解实现。@Profile可以标注在类上和方法上。标注在类上时，通常类的上方需要被@Component标注，以指定创建的Bean的生效环境。标注在方法上时；通常为配置类中被@Bean标注的方法，以指定返回的Bean的生效环境。



下面通过案例来演示@Profile注解进行多环境配置的使用方法。

(1) **创建数据库连接接口**。在chapter02项目的com.itheima.config包下，创建**数据库配置**的接口DBConnector，并在该接口中声明一个**数据库配置连接方法**，具体如文件2-15所示。



### 源代码

文件2-15  
[DBConnector.java](#)





(2) **创建数据库连接实现类**。在chapter02项目的com.itheima.config包下，根据数据库连接接口DBConnector，创建**不同环境的数据库连接实现类DevDBConnector和ProDBConnector**，并**重写configure()方法**模拟进行不同数据库环境的连接配置，具体如文件2-16和文件2-17所示。



### 源代码

文件2-16 [DevDBConnector.java](#)

文件2-17 [ProdDBConnector.java](#)





(3) **新增控制器方法**。在文件2-11的DefaultController类中**注入DBConnector对象**，并新增方法**showDB()**，在该方法中执行数据库连接配置方法，具体代码如下所示。

```
@Autowired
private DBConnector connector;
@RequestMapping("/showDB")
public void showDB() {
    connector.configure();
}
```

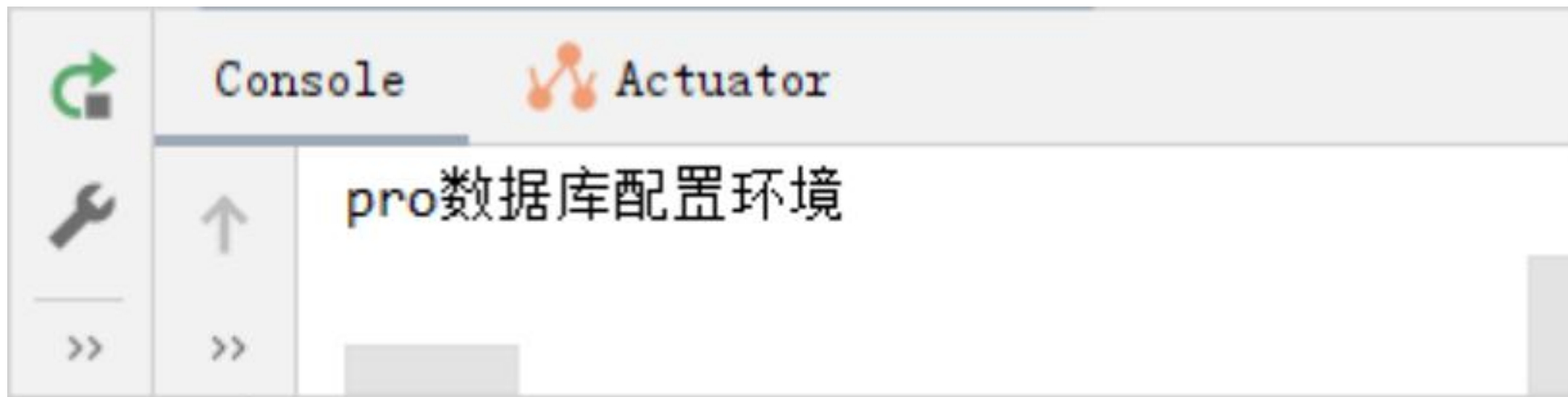


(4) 测试程序效果。在项目application.yml配置文件中指定激活的Profile为pro，启动项目，控制台会输出pro的相关信息。

```
Run: Chapter02Application x
Console Actuator
2022-10-14 17:37:28.074 INFO 12420 --- [ restartedMain] com.itheima.Chapter02Application : Starting Chapter02Application using Java 11.0.16 on t
2022-10-14 17:37:28.076 INFO 12420 --- [ restartedMain] com.itheima.Chapter02Application : The following 1 profile is active: "pro"
2022-10-14 17:37:28.169 INFO 12420 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devto
2022-10-14 17:37:28.169 INFO 12420 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting t
2022-10-14 17:37:29.336 INFO 12420 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 82 (http)
2022-10-14 17:37:29.347 INFO 12420 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-14 17:37:29.347 INFO 12420 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-10-14 17:37:29.411 INFO 12420 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-14 17:37:29.411 INFO 12420 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
2022-10-14 17:37:29.806 INFO 12420 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-10-14 17:37:29.859 INFO 12420 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 82 (http) with context pat
2022-10-14 17:37:29.878 INFO 12420 --- [ restartedMain] com.itheima.Chapter02Application : Started Chapter02Application in 2.424 seconds (JVM ru
```

## >>> 2.4.3 @Profile注解

在浏览器访问<http://localhost:82/showDB>





### 本章小结

本章主要对Spring Boot配置进行了讲解。首先讲解了全局配置文件；然后讲解了配置绑定；接着讲解了引入配置文件和定义配置类；最后讲解了Profile。通过本章的学习，希望大家可以掌握对Spring Boot的基本配置进行配置的方法，为后续更深入学习Spring Boot做好铺垫。



為千萬學生少走彎路而著書



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌