

Spring注解驱动开发第19讲——使用@PropertySource加载配置文件，我只看这一篇！！

简单介绍一下@PropertySource和@PropertySources这两注解

@PropertySource注解概述

@PropertySource注解是Spring 3.1开始引入的配置类注解。通过@PropertySource注解可以将 **properties配置文件** 中的key/value存储到Spring的Environment中，Environment接口提供了方法去读取配置文件中的值，参数是properties配置文件中定义的key值。当然了，也可以使用@Value注解用 **\${}** 占位符为bean的属性注入值。

我们来看一下@PropertySource注解的源代码，如下所示。

```
Person.java | MainConfigOfPropertyValues.java | IOCTest_PropertyValue.java | PropertySource.class
20 import java.lang.annotation.ElementType;
21 import java.lang.annotation.Repeatable;
22 import java.lang.annotation.Retention;
23 import java.lang.annotation.RetentionPolicy;
24 import java.lang.annotation.Target;
25
26 import org.springframework.core.io.support.PropertySourceFactory;
27
29+ * Annotation providing a convenient and declarative mechanism for adding a
152 @Target(ElementType.TYPE)
153 @Retention(RetentionPolicy.RUNTIME)
154 @Documented
155 @Repeatable(PropertySources.class)
156 public @interface PropertySource {
157
158+ * Indicate the name of this property source. If omitted, a name will
164 String name() default "";
165
167+ * Indicate the resource location(s) of the properties file to be loaded.
178 String[] value();
179
181+ * Indicate if failure to find the a {@link #value()} property resource} should be
187 boolean ignoreResourceNotFound() default false;
188
190+ * A specific character encoding for the given resources, e.g. "UTF-8".
193 String encoding() default "";
194
196+ * Specify a custom {@link PropertySourceFactory}, if any.
202 Class< extends PropertySourceFactory> factory() default PropertySourceFactory.class;
203
204 }
205
```

从@PropertySource的源码中可以看出，我们可以通过@PropertySource注解指定多个properties文件，使用的形式如下所示。

```
1 | @PropertySource(value={"classpath:/person.properties", "classpath:/car.properties"})
   AI写代码java运行
```

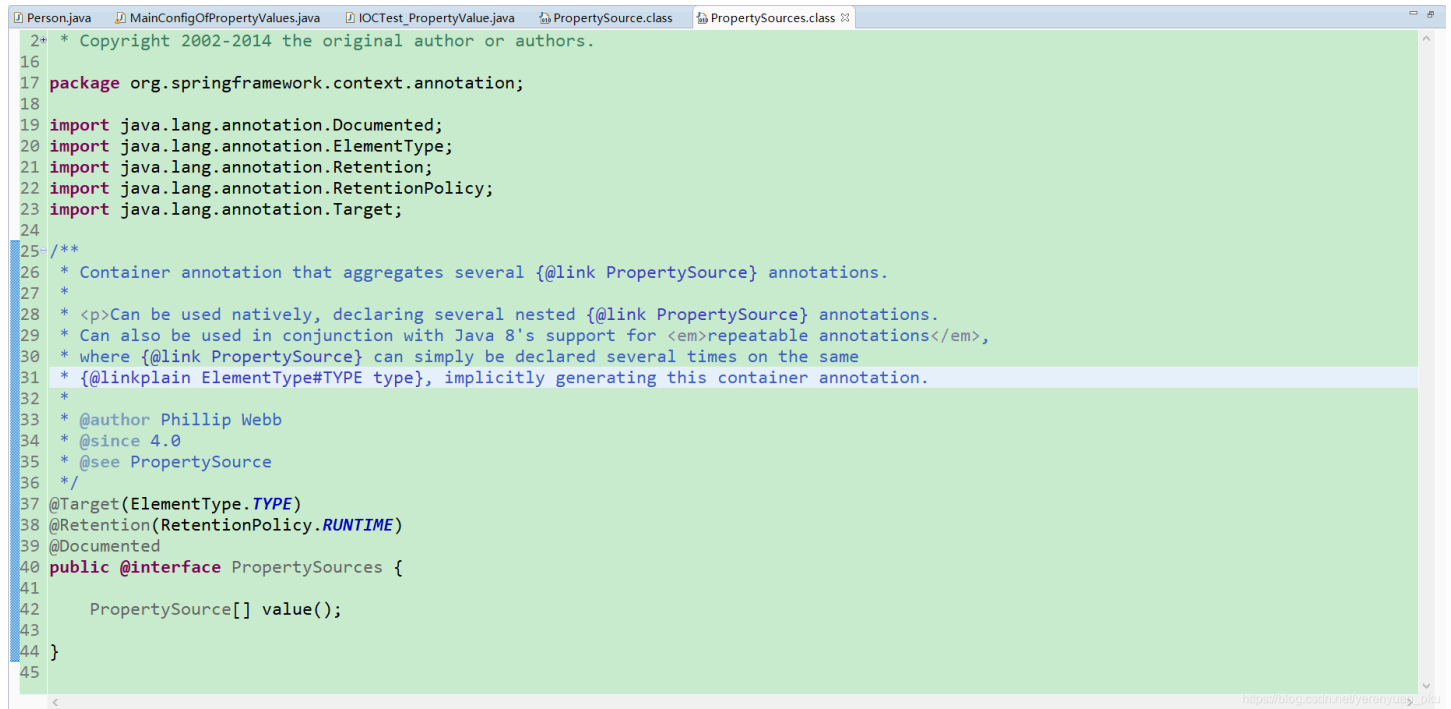
细心的读者可以看到，在@PropertySource注解的上面标注了如下的注解信息。

```
1 | @Repeatable(PropertySources.class)
   AI写代码java运行
```

看到这里，小伙伴们是不是有种恍然大悟的感觉呢？没错，我们也可以使用@PropertySources注解来指定properties配置文件。

@PropertySources注解概述

首先，我们也看下@PropertySources注解的源码，如下所示。



@PropertySources注解的源码比较简单，只有一个PropertySource[]数组类型的value属性，那我们如何使用@PropertySources注解指定 **配置文件** 呢？其实也很简单，使用如下所示的方式就可以了。

```
1 @PropertySources(value={
2     @PropertySource(value={"classpath:/person.properties"}),
3     @PropertySource(value={"classpath:/car.properties"}),
4 })
    AI写代码java运行
```

是不是很简单呢？接下来，我们就以一个小案例来说明@PropertySource注解的用法。

一个小案例来说明@PropertySource注解的用法

准备工作

首先，我们在工程的src/main/resources目录下创建一个配置文件，例如person.properties，该文件的内容如下所示。

```
1 person.nickName=小甜甜
    AI写代码xml
```

然后，我们在Person类中新增一个nickName字段，如下所示。

```
1 package com.meimeixia.bean;
2
3 import org.springframework.beans.factory.annotation.Value;
4
5 public class Person {
6
7     @Value("李阿的")
8     private String name;
9     @Value("#{20-2}")
10    private Integer age;
11
12    private String nickName; // 昵称
13
14    public String getNickName() {
15        return nickName;
16    }
17    public void setNickName(String nickName) {
18        this.nickName = nickName;
19    }
20    public String getName() {
21        return name;
22    }
23    public void setName(String name) {
24        this.name = name;
25    }
26    public Integer getAge() {
27        return age;
28    }
29 }
```

```

28     }
29     public void setAge(Integer age) {
30         this.age = age;
31     }
32     public Person(String name, Integer age) {
33         super();
34         this.name = name;
35         this.age = age;
36     }
37     public Person() {
38         super();
39         // TODO Auto-generated constructor stub
40     }
41     @Override
42     public String toString() {
43         return "Person [name=" + name + ", age=" + age + ", nickName=" + nickName + "];
44     }
45 }
46 }

```

AI写代码java运行



目前，我们并没有为Person类的nickName字段赋值，所以，此时Person类的nickName字段的值为空。我们可以运行IOCTest_PropertyValue类中的test01()方法来看下输出结果，如下所示。

```

<terminated> IOCTest_PropertyValue.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月2日 下午3:56:49)
十二月 02, 2020 3:56:49 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext prepareRef
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup date
org.springframework.context.annotation.internalConfigurationAnnotationProcessor
org.springframework.context.annotation.internalAutowiredAnnotationProcessor
org.springframework.context.annotation.internalRequiredAnnotationProcessor
org.springframework.context.annotation.internalCommonAnnotationProcessor
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
mainConfigOfPropertyValues
person
=====
Person [name=李阿昀, age=18, nickName=null]
十二月 02, 2020 3:56:50 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup date

```

可以看到，Person类的nickName字段的值确实输出了null。

使用XML配置文件方式获取值

如果我们需要在XML配置文件中获取person.properties文件中的值，那么我们首先需要在Spring的XML配置文件中引入context名称空间，并且使用context命名空间导入person.properties文件，之后在bean的属性字段中使用如下方式将person.properties文件中的值注入到Person类的nickName字段上。

```

1 <context:property-placeholder location="classpath:person.properties" />
2
3 <!-- 注册组件 -->
4 <bean id="person" class="com.meimeixia.bean.Person">
5     <property name="age" value="18"></property>
6     <property name="name" value="liayun"></property>
7     <property name="nickName" value="${person.nickName}"></property>
8 </bean>

```

AI写代码xml

此时，整个beans.xml文件的内容如下所示。

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.springframework.org/schema/p"
4     xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"
5     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6     xsi:schemaLocation="http://www.springframework.org/schema/beans
7         http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
8         http://www.springframework.org/schema/context
9         http://www.springframework.org/schema/context/spring-context-4.2.xsd">
10
11     <context:property-placeholder location="classpath:person.properties" />
12

```

```

13     <!-- 注册组件 -->
14     <bean id="person" class="com.meimeixia.bean.Person">
15         <property name="age" value="18"></property>
16         <property name="name" value="liayun"></property>
17         <property name="nickName" value="${person.nickName}"></property>
18     </bean>
19
20 </beans>

```

AI写代码xml



这样就可以将person.properties文件中的值注入到Person类的nickName字段上了。

然后，我们在IOCTest_PropertyValue类中创建一个test02()测试方法，如下所示。

```

1  @Test
2  public void test02() {
3      ClassPathXmlApplicationContext applicationContext = new ClassPathXmlApplicationContext("classpath:beans.xml");
4      Person person = (Person) applicationContext.getBean("person");
5      System.out.println(person);
6  }

```

AI写代码java运行

接着，运行以上test02()方法，输出的结果信息如下所示。

```

<terminated> IOCTest_PropertyValue.test02 [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月2日 下午4:26:24)
十二月 02, 2020 4:26:24 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext prepareRefresh
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup date [Wed Dec 02, 2020 4:26:24 下午]
十二月 02, 2020 4:26:24 下午 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh
信息: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@3745e5c6: startup date [Wed Dec 02, 2020 4:26:24 下午]
十二月 02, 2020 4:26:24 下午 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
信息: Loading XML bean definitions from class path resource [beans.xml]
Person [name=liayun, age=18, nickName=小甜甜]

```

<https://blog.csdn.net/yerenyuan>

使用注解方式获取值

如果我们使用注解的方式，那么该如何做呢？首先，我们需要在MainConfigOfPropertyValues配置类上添加一个@PropertySource注解，如下所示。

```

1  package com.meimeixia.config;
2
3  import org.springframework.context.annotation.Bean;
4  import org.springframework.context.annotation.Configuration;
5  import org.springframework.context.annotation.PropertySource;
6
7  import com.meimeixia.bean.Person;
8
9  // 使用@PropertySource读取外部配置文件中的key/value保存到运行的环境变量中，加载完外部的配置文件以后，使用${}取出配置文件中的值
10 @PropertySource(value={"classpath:/person.properties"})
11 @Configuration
12 public class MainConfigOfPropertyValues {
13
14     @Bean
15     public Person person() {
16         return new Person();
17     }
18
19 }

```

AI写代码java运行



这里使用的 @PropertySource(value={"classpath:/person.properties"}) 注解就相当于XML配置文件中使用的 <context:property-placeholder location="classpath:person.properties" />。

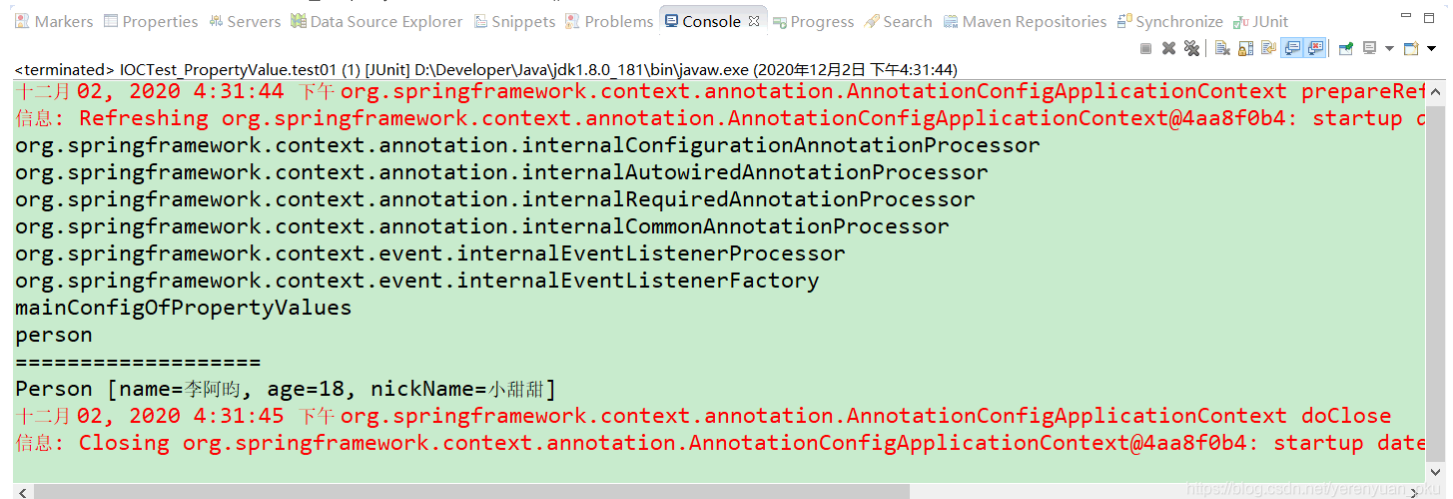
然后，我们就可以在Person类的nickName字段上使用@Value注解来获取person.properties文件中的值了，如下所示。

```

1  @Value("${person.nickName}")
2  private String nickName; // 昵称

```

配置完成后，我们再次运行IOCTest_PropertyValue类中的test01()方法来看下输出结果，如下所示。



```
<terminated> IOCTest_PropertyValue.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月2日 下午4:31:44)
十二月 02, 2020 4:31:44 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext prepareRef
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup c
org.springframework.context.annotation.internalConfigurationAnnotationProcessor
org.springframework.context.annotation.internalAutowiredAnnotationProcessor
org.springframework.context.annotation.internalRequiredAnnotationProcessor
org.springframework.context.annotation.internalCommonAnnotationProcessor
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
mainConfigOfPropertyValues
person
=====
Person [name=李阿昀, age=18, nickName=小甜甜]
十二月 02, 2020 4:31:45 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup date
```

可以看到，此时Person类的nickName字段已经注入 小甜甜 这个值了。

使用Environment获取值

上面我已经说过，使用@PropertySource注解读取外部配置文件中的key/value之后，是将其保存到运行的环境变量中了，所以我們也可以通过运行环境来获取外部配置文件中的值。

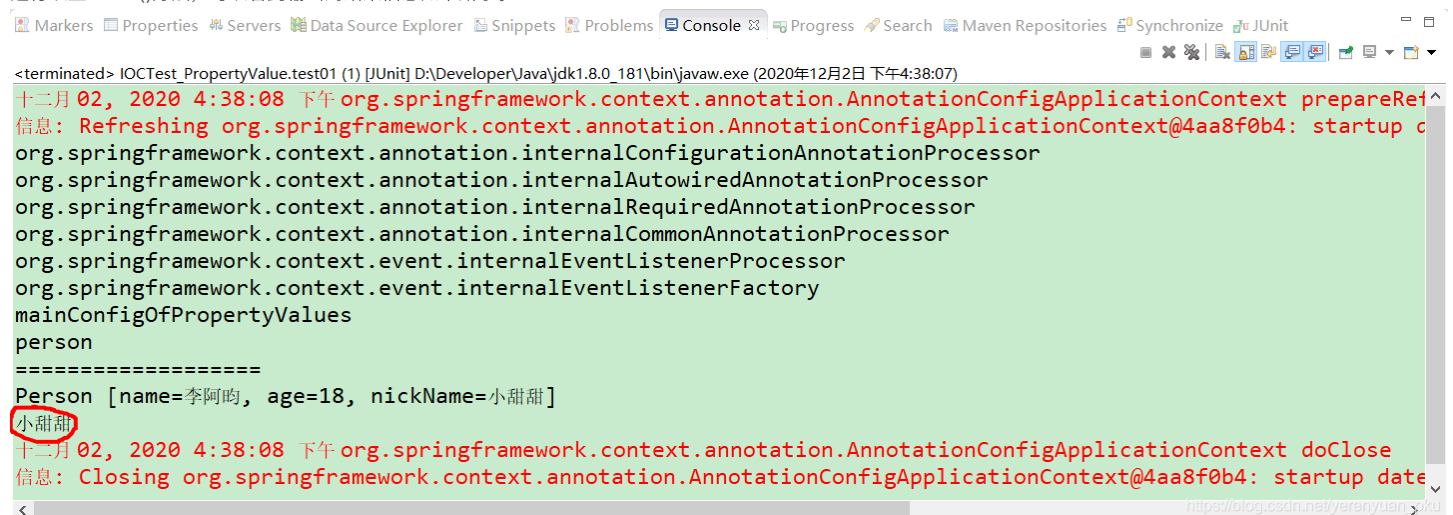
这里，我们可以稍微修改一下IOCTest_PropertyValue类中的test01()方法，即在其中添加一段使用Environment获取person.properties文件中的值的代码，如下所示。

```
1 @Test
2 public void test01() {
3     printBeans(applicationContext);
4     System.out.println("=====");
5
6     Person person = (Person) applicationContext.getBean("person");
7     System.out.println(person);
8
9     ConfigurableEnvironment environment = applicationContext.getEnvironment();
10    String property = environment.getProperty("person.nickName");
11    System.out.println(property);
12
13    // 关闭容器
14    applicationContext.close();
15 }
```

AI写代码java运行



运行以上test01()方法，可以看到输出的结果信息如下所示。



```
<terminated> IOCTest_PropertyValue.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月2日 下午4:38:07)
十二月 02, 2020 4:38:08 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext prepareRef
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup c
org.springframework.context.annotation.internalConfigurationAnnotationProcessor
org.springframework.context.annotation.internalAutowiredAnnotationProcessor
org.springframework.context.annotation.internalRequiredAnnotationProcessor
org.springframework.context.annotation.internalCommonAnnotationProcessor
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
mainConfigOfPropertyValues
person
=====
Person [name=李阿昀, age=18, nickName=小甜甜]
小甜甜
十二月 02, 2020 4:38:08 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@4aa8f0b4: startup date
```

可以看到，使用Environment确实能够获取到person.properties文件中的值。