



当前位置: Java 技术驿站 (<http://cmsblogs.com>) > 死磕Java (<http://cmsblogs.com/?cat=189>) > 死磕 Spring (<http://cmsblogs.com/?cat=206>) > 正文

【死磕 Spring】—— IOC 之 PropertyPlaceholderConfigurer 的应用 (<http://cmsblogs.com/?p=3839>)

2019-01-05 分类: 死磕 Spring (<http://cmsblogs.com/?cat=206>) 阅读(5934) 评论(0)

在博客【死磕 Spring】----- IOC 之 深入分析 PropertyPlaceholderConfigurer (<http://cmsblogs.com/?p=3837>) 中了解了 PropertyPlaceholderConfigurer 内部实现原理，她允许我们在 XML 配置文件中使用占位符并将这些占位符所代表的资源单独配置到简单的 properties 文件中来加载。这个特性非常重要，因为它我们对 Bean 实例属性的配置变得非常容易控制了，主要使用场景有：

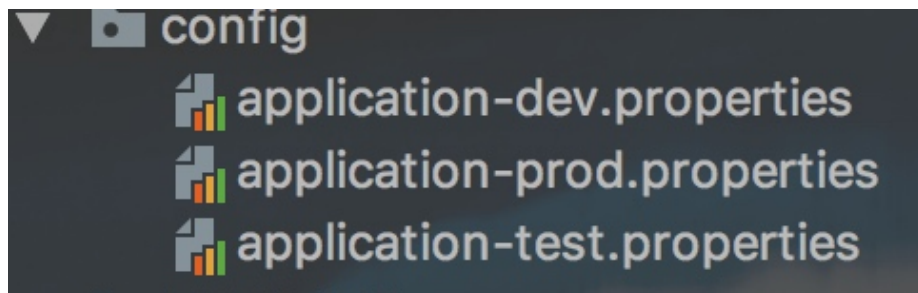
1. 动态加载配置文件，多环境切换
2. 属性加解密

下面我们就第一个应用场景来做说明。

利用 PropertyPlaceholderConfigurer 实现多环境切换

在我们项目开发过程中，都会存在多个环境，如 dev、test、prod 等等，各个环境的配置都会不一样，在传统的开发过程中我们都是在进行打包的时候进行人工干预，或者将配置文件放在系统外部，加载的时候指定加载目录，这种方式容易出错，那么有没有一种比较好的方式来解决这种情况呢？有，利用 PropertyPlaceholderConfigurer 的特性来动态加载配置文件，实现多环境切换。

首先我们定义四个 Properties 文件，如下：



(<https://gitee.com/chenssy/blog-home/raw/master/image/201811/15374242055683.jpg>)

内容如下：

```
- application-dev.properties
student.name=chenssy-dev

- application-test.properties
student.name=chenssy-test

- application-prod.properties
student.name=chenssy-prod
```

然后实现一个类，该类继承 PropertyPlaceholderConfigurer，实现 loadProperties()，根据环境的不同加载不同的配置文件，如下：



```

public class CustomPropertyConfig extends PropertyPlaceholderConfigurer {

    private Resource[] locations;

    private PropertiesPersister propertiesPersister = new DefaultPropertiesPersister();

    @Override
    public void setLocations(Resource[] locations) {
        this.locations = locations;
    }

    @Override
    public void setLocalOverride(boolean localOverride) {
        this.localOverride = localOverride;
    }

    /**
     * 覆盖这个方法，根据启动参数，动态读取配置文件
     * @param props
     * @throws IOException
     */
    @Override
    protected void loadProperties(Properties props) throws IOException {
        if (locations != null) {
            // locations 里面就已经包含了那三个定义的文件
            for (Resource location : this.locations) {
                InputStream is = null;
                try {
                    String filename = location.getFilename();
                    String env = "application-" + System.getProperty("spring.profiles.active", "dev") +
                        ".properties";

                    // 找到我们需要的文件，加载
                    if (filename.contains(env)) {
                        logger.info("Loading properties file from " + location);
                        is = location.getInputStream();
                        this.propertiesPersister.load(props, is);
                    }
                } catch (IOException ex) {
                    logger.info("读取配置文件失败.....");
                    throw ex;
                } finally {
                    if (is != null) {
                        is.close();
                    }
                }
            }
        }
    }
}

```

配置文件:



```
<bean id="PropertyPlaceholderConfigurer" class="org.springframework.core.custom.CustomPropertyConfig"
>
    <property name="locations">
        <list>
            <value>classpath:config/application-dev.properties</value>
            <value>classpath:config/application-test.properties</value>
            <value>classpath:config/application-prod.properties</value>
        </list>
    </property>
</bean>

<bean id="studentService" class="org.springframework.core.service.StudentService">
    <property name="name" value="${student.name}"/>
</bean>
```

在 idea 的 VM options 里面增加 `-Dspring.profiles.active=dev`，标志当前环境为 dev 环境。测试代码如下：

```
ApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");


StudentService studentService = (StudentService) context.getBean("studentService");
System.out.println("student name:" + studentService.getName());
```

运行结果:

```
student name:chenssy-dev
```

当将 `-Dspring.profiles.active` 调整为 `test`，则打印结果则是 `chenssy-test`，这样就完全实现了根据不同的环境加载不同的配置，如果各位用过 Spring Boot 的话，这个就完全是 Spring Boot 里面的 `profiles.active`。

PropertyPlaceholderConfigurer 对于属性的配置非常灵活，就看怎么玩了。

 赞(3) 打赏

【公告】版权声明 (http://cmsblogs.com/?page_id=1908)

标签: Spring 源码解析 (<http://cmsblogs.com/?tag=spring-%e6%ba%90%e7%a0%81%e8%a7%a3%e6%9e%90>)

死磕Java (<http://cmsblogs.com/?tag=%e6%ad%bb%e7%a3%95java>)

死磕Spring (<http://cmsblogs.com/?tag=%e6%ad%bb%e7%a3%95spring>)

 chenssy (<http://cmsblogs.com/?author=1>)