# 21 Resource总结

更新时间：2020-07-01 13:58:30

> 不安于小成，然后足以成大器；不诱于小利，然后可以立远功。——方孝孺

## 常见的 Spring 面试问题及解析


**在 Spring 中如何将一个 java.util.Properties 注入到 bean 中？**

两种方式，XML 和注解方式。

XML 方式：

```
<bean id="propertyConfigurer" class="org.springframework.context.support.PropertySourcesPlaceholderConfigurer">
<property name="location" value="/WEB-INF/application.properties" />
</bean>
<bean class="com.journaldev.spring.EmployeeDaoImpl">
<property name="maxReadResults" value="${results.read.max}"/>
</bean>
```

注解方式：

```
@Value("${maxReadResults}") private int maxReadResults;
```

**在 Spring 中如何读取静态文件？应用内部和应用外部有什么不同？**

Spring 提供了 Resource 实现，包括：

**UrlResource**：UrlResource 包装了 java.net.URL，可以用来访问通常可以通过 URL 访问的任何对象，比如文件、HTTP 目标、FTP 目标和其他。所有 URL 都有标准化的字符串表示，这样就可以使用适当的标准化前缀来表示不同类型的 URL。这包括 file：用于访问文件系统路径，http：用于通过 HTTP 协议访问资源，ftp：用于通过 ftp 访问资源，等等；

**ClassPathResource**：该类表示应从 classpath 获取的资源。实现支持 java.io 格式的解析；

**FileSystemResource**：实现了 java.io.File 和 java.nio.file.Path 的资源类. 支持 File 和 URL 格式的解析；

**ServletContextResource**:实现了 ServletContext 的资源类，它解析 web 相关应用程序根目录中的相对路径。它始终支持流访问和 URL 访问，但允许 java.io；

**InputStreamResource**：InputStream 的资源实现。解析 InpustStream 类型的资源；

**ByteArrayResource**：给定字节数组的资源实现。它为给定的字节数组创建 ByteArrayInputStream。它用于从任何给定的字节数组加载内容，而不必求助于一次性使用的 InputStreamResource；

**VfsResource**：资源以虚拟文件 VFS 的形式存在，可以使用该实现类。VFS 是一个虚拟文件系统，Linux 的系统中所有文件的顶层都设计为虚拟的 VFS，它能一致的访问物理文件系统、jar 资源、zip 资源、war 资源等，VFS 能把这些资源一致的映射到一个目录上，访问它们就像访问物理文件资源一样，而其实这些资源不存在于物理文件系统。该实现类，封装了一个 Object 对象，所有的操作都是通过这个包装的对象的反射来实现的。当然，内部具体实现细节，可以通过工具类 VfsUtils 调用。

从上面看出，Spring 读取静态文件的方式是多种多样的，应用内部一般使用 ClassPathResource，容器 ClassPathXmlApplicationContext 可以读取应用内部 classpath 下面的资源文件。

应用外部一般使用 FileSystemResource，容器 FileSystemXmlApplicationContext 可以读取应用外部任意路径的文件。

**Spring Bean 的定义是否支持 XML 和注解混合使用？**

支持，官方描述：

> Spring can accommodate both styles and even mix them together

**注意**：混合用的话，有个先后顺序，XML 配置会覆盖 annotation，官方描述：

> Annotation injection is performed before XML injection. Thus, the XML configuration overrides the annotations for properties wired through both approaches.

**如何读取 jar 或者 war 内部的资源文件？**

正确的方式：

```
input = XXXUtil.class.getClassLoader() .getResourceAsStream("resources" + File.separator + fileName);

reader = new InputStreamReader(input, Constant.UTF_8);
```

错误的方式：

```
ClassLoader loader = XXXUtil.class.getClassLoader();

String jsFileName = loader.getResource("") + "/resources/" + fileName;
```

初看两种方式，应该是一致的。

其实则不同：

使用文件路径，要求该路径下的文件在文件系统上是可以访问的。因为 jar 文件需要解压才能访问，直接访问不了；

使用流读文件，则不存在上述要求。

# Spring Resource 总结

**Resource 的不同资源实现**

Resouce 抽象了不同的计算机资源，下面是我们常用资源的形式及读取方式。

**UrlResource**：UrlResource 包装了 java.net.URL，可以用来访问通常可以通过 URL 访问的任何对象，比如文件、HTTP 目标、FTP 目标和其他。所有 URL 都有标准化的字符串表示，这样就可以使用适当的标准化前缀来表示不同类型的 URL。这包括 file：用于访问文件系统路径，http：用于通过 HTTP 协议访问资源，ftp：用于通过 ftp 访问资源，等等；

**ClassPathResource**：该类表示应从 classpath 获取的资源。实现支持 java.io 格式的解析；

**FileSystemResource**：实现了 java.io.File 和 java.nio.file.Path 的资源类. 支持 File 和 URL 格式的解析；

**ServletContextResource**:实现了 ServletContext 的资源类，它解析 web 相关应用程序根目录中的相对路径。它始终支持流访问和 URL 访问，但允许 java.io；

**InputStreamResource**：InputStream 的资源实现。解析 InpustStream 类型的资源；

**ByteArrayResource**：给定字节数组的资源实现。它为给定的字节数组创建 ByteArrayInputStream。它用于从任何给定的字节数组加载内容，而不必求助于一次性使用的 InputStreamResource。

**ApplicationContext 具有读取 Resource 的能力：**

- ClassPathXmlApplicationContext 则底层 Resource 是 ClassPathResource 实例；
- FileSystemXmlApplicationContext 则底层 Resource 是 FileSystemResource 实例；
- XmlWebApplicationContext 则底层 Resource 是 ServletContextResource 实例。

ApplicationContext 实现了 ResourceLoader 接口，可以通过 getResource() 方法实现资源的获取，示例如下：

```
Resource template = ClassPathXmlApplicationContext .getResource("some/resource/path/myTemplate.txt");

Resource template = ClassPathXmlApplicationContext .getResource("classpath:some/resource/path/myTemplate.txt");

Resource template = FileSystemXmlApplicationContext.getResource("file:///some/resource/path/myTemplate.txt");

Resource template = FileSystemXmlApplicationContext.getResource("https://myhost.com/resource/path/myTemplate.txt");
```

**ResourceLoaderAware 接口实现**

这是 Spring 为了方便我们获得 Resource 而提供的感知接口，Spring 会自动调用实现了 ResourceLoaderAware 接口类方法：serResourceLoader()，将 ApplicationContext 的 ResourceLoader 注入进去，之后对它 getResource()，就获取了系统的 Resource 了。

}

20 面试官，请不要再问我 @Resource和@Autowire注解的区别了

22 Spring Core参数校验之 validator