【死磕 Spring】—— IOC 之 获取验证模型 (http://cmsblogs.com/?p=2688)

2018-09-10 分类：死磕 Spring (http://cmsblogs.com/?cat=206) 阅读(13839) 评论(7)

> 原文出自：http://cmsblogs.com (http://cmsblogs.com)

在上篇博客【死磕Spring】----- IOC 之 加载 Bean (http://cmsblogs.com/?p=2658) 中提到，在核心逻辑方法 `doLoadBeanDefinitions()` 中主要是做三件事情。

1. 调用 `getValidationModeForResource()` 获取 xml 文件的验证模式
2. 调用 `loadDocument()` 根据 xml 文件获取相应的 Document 实例。
3. 调用 `registerBeanDefinitions()` 注册 Bean 实例。

这篇博客主要分析获取 xml 文件的验证模式。

> XML 文件的验证模式保证了 XML 文件的正确性

DTD 与 XSD 的区别

DTD(Document Type Definition)，即文档类型定义，为 XML 文件的验证机制，属于 XML 文件中组成的一部分。DTD 是一种保证 XML 文档格式正确的有效验证方式，它定义了相关 XML 文档的元素、属性、排列方式、元素的内容类型以及元素的层次结构。其实 DTD 就相当于 XML 中的 “词汇”和“语法”，我们可以通过比较 XML 文件和 DTD 文件 来看文档是否符合规范，元素和标签使用是否正确。 要在 Spring 中使用 DTD，需要在 Spring XML 文件头部声明：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC  "-//SPRING//DTD BEAN//EN"  "http://www.springframework.org/dtd/spring-beans.dtd">
```

DTD 在一定的阶段推动了 XML 的发展，但是它本身存在着一些缺陷：

1. 它没有使用 XML 格式，而是自己定义了一套格式，相对解析器的重用性较差；而且 DTD 的构建和访问没有标准的编程接口，因而解析器很难简单的解析 DTD 文档。
2. DTD 对元素的类型限制较少；同时其他的约束力也叫弱。
3. DTD 扩展能力较差。
4. 基于正则表达式的 DTD 文档的描述能力有限。

针对 DTD 的缺陷，W3C 在 2001 年推出 XSD。XSD（XML Schemas Definition）即 XML Schema 语言。XML Schema 本身就是一个 XML文档，使用的是 XML 语法，因此可以很方便的解析 XSD 文档。相对于 DTD，XSD 具有如下优势：

- XML Schema基于XML,没有专门的语法
- XML Schema可以象其他XML文件一样解析和处理
- XML Schema比DTD提供了更丰富的数据类型.
- XML Schema提供可扩充的数据模型。
- XML Schema支持综合命名空间
- XML Schema支持属性组。

getValidationModeForResource() 分析

```
protected int getValidationModeForResource(Resource resource) {
    // 获取指定的验证模式
    int validationModeToUse = getValidationMode();
    // 如果手动指定，则直接返回
    if (validationModeToUse != VALIDATION_AUTO) {
        return validationModeToUse;
    }
    // 通过程序检测
    int detectedMode = detectValidationMode(resource);
    if (detectedMode != VALIDATION_AUTO) {
        return detectedMode;
    }

    // 出现异常，返回 XSD
    return VALIDATION_XSD;
}
```

如果指定了 XML 文件的的验证模式（调用 `XmlBeanDefinitionReader.setValidating(boolean validating)`）则直接返回指定的验证模式，否则调用 `detectValidationMode()` 获取相应的验证模式，如下:

```
protected int detectValidationMode(Resource resource) {
    if (resource.isOpen()) {
        throw new BeanDefinitionStoreException(
                "Passed-in Resource [" + resource + "] contains an open stream: " +
                "cannot determine validation mode automatically. Either pass in a Resource " +
                "that is able to create fresh streams, or explicitly specify the validationMode " +
                "on your XmlBeanDefinitionReader instance.");
    }

    InputStream inputStream;
    try {
        inputStream = resource.getInputStream();
    }
    catch (IOException ex) {
        throw new BeanDefinitionStoreException(
                "Unable to determine validation mode for [" + resource + "]: cannot open InputStream.
" +
                "Did you attempt to load directly from a SAX InputSource without specifying the " +
                "validationMode on your XmlBeanDefinitionReader instance?", ex);
    }

    try {
      // 核心方法
        return this.validationModeDetector.detectValidationMode(inputStream);
    }
    catch (IOException ex) {
        throw new BeanDefinitionStoreException("Unable to determine validation mode for [" +
                resource + "]: an error occurred whilst reading from the InputStream.", ex);
    }
}
```

前面一大堆的代码，核心在于 `this.validationModeDetector.detectValidationMode(inputStream)`，validationModeDetector 定义为 `XmlValidationModeDetector`，所以验证模式的获取委托给 `XmlValidationModeDetector` 的 `detectValidationMode()` 方法。

```
public int detectValidationMode(InputStream inputStream) throws IOException {
    BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
    try {
        boolean isDtdValidated = false;
        String content;
        // 一行一行读取 xml 文件的内容
        while ((content = reader.readLine()) != null) {
            content = consumeCommentTokens(content);
            if (this.inComment || !StringUtils.hasText(content)) {
                continue;
            }
            // 包含 DOCTYPE 为 DTD 模式
            if (hasDoctype(content)) {
                isDtdValidated = true;
                break;
            }
            // 读取 < 开始符号，验证模式一定会在 < 符号之前
            if (hasOpeningTag(content)) {
                // End of meaningful data...
                break;
            }
        }
    // 为 true 返回 DTD，否则返回 XSD
        return (isDtdValidated ? VALIDATION_DTD : VALIDATION_XSD);
    }
    catch (CharConversionException ex) {
        // 出现异常，为 XSD
        return VALIDATION_AUTO;
    }
    finally {
        reader.close();
    }
}
```

从代码中看，主要是通过读取 XML 文件的内容，判断内容中是否包含有 DOCTYPE ，如果是 则为 DTD，否则为 XSD，当然只会读取到 第一个 "<" 处，因为 验证模式一定会在第一个 “<” 之前。如果当中出现了 CharConversionException 异常，则为 XSD模式。 好了，XML 文件的验证模式分析完毕，下篇分析 `doLoadBeanDefinitions()` 的第二个步骤：获取 Document 实例。

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)