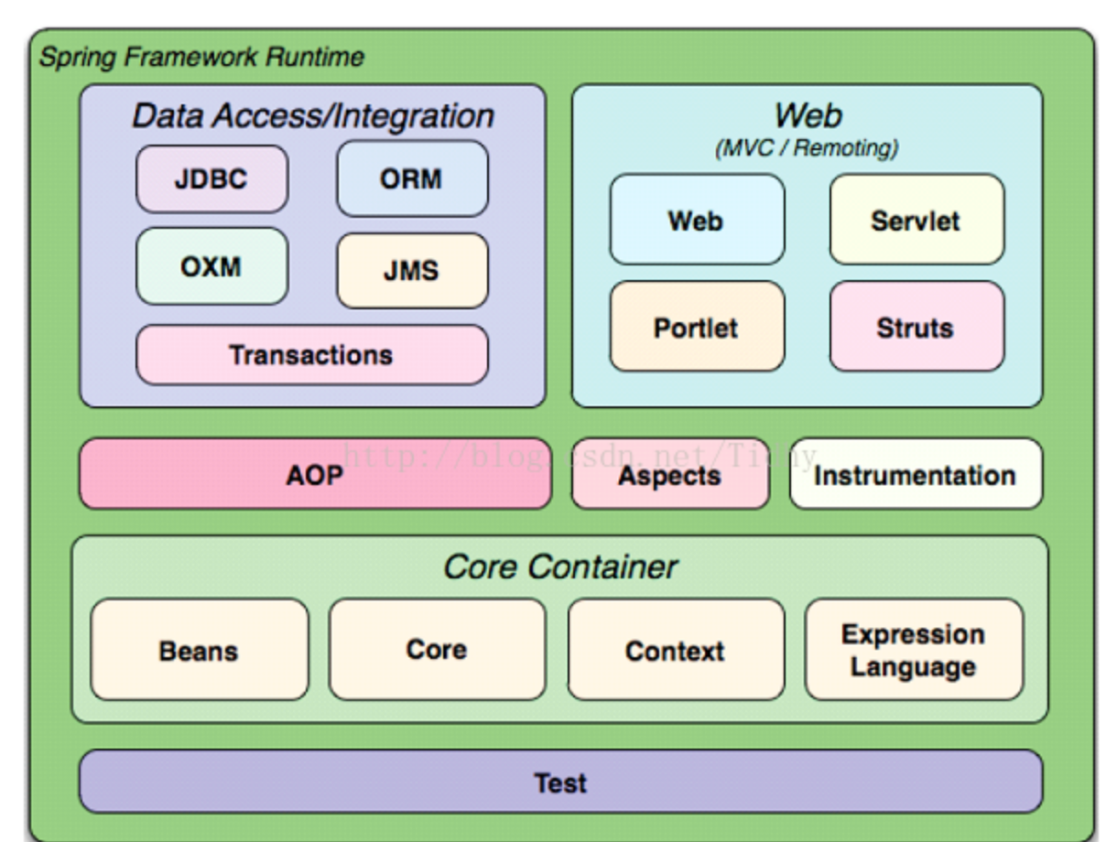
# Spring源码深度解读

# Spring整体架构与环境搭建

Spring框架架构图：



1.Core Container：核心容器包含Core、Beans、Context与Expression Language模块，Core与Beans框架的基础模块，提供IOC功能；核心概念就是Factory；Core包含核心的工具类；Beans用来生成Bean及产生依赖关系；Context提供了国际化、事件传播、资源加载、ApplicationContext是Context模块的关键；EL用于在运行时查询与操作对象；

2.Data Access：JDBC抽象层；ORM对象关系映射API；OXM提供了一个对Object/XML映射实现的抽象层；

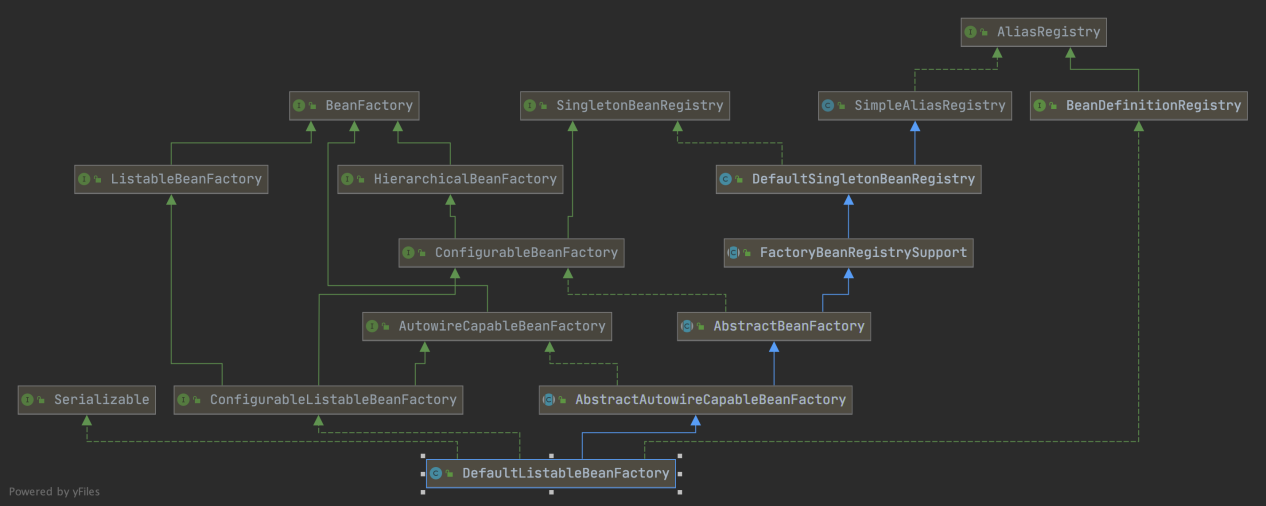
3.Web：Web、Web-Servlet、Web-Structs、Web-Portlet；

4.AOP：

# 容器的基本实现

## 2.4 Spring的结构组成

核心类：DefaultListableBeanFactory，Spring注册与加载bean的核心实现。类图结构：



* AliasRegistry：对Alias的简单的增删改查等操作；



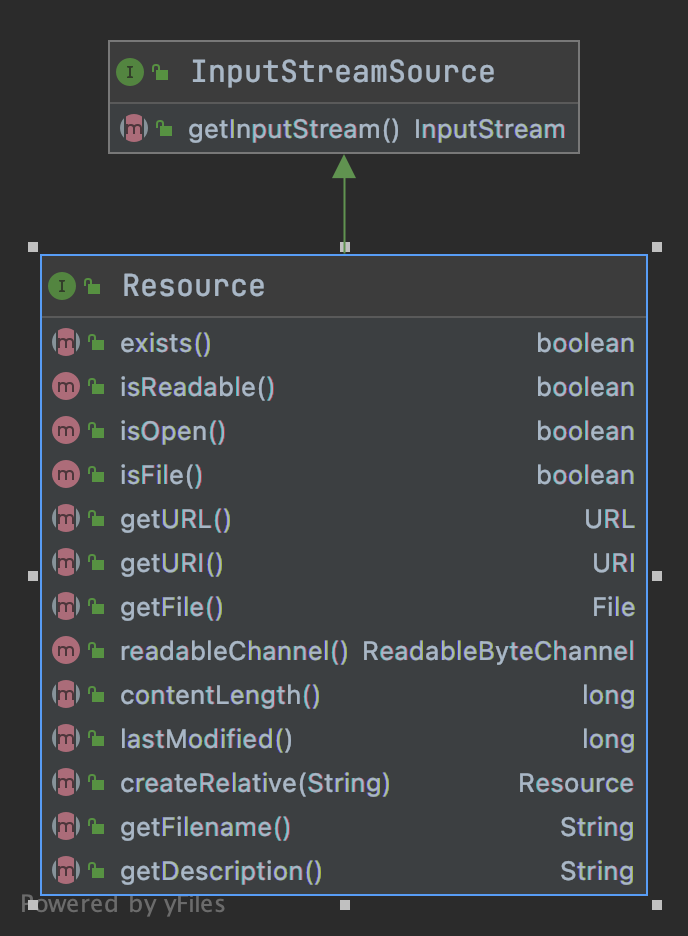
* + SimpleAliasRegistry：使用map作为存储对AliasRegistry实现；
  + SingletonBeanRegistry：定义对单例的注册与获取；
  + BeanFactory：定义获取Bean与Bean的各种属性;
  + DefaultSingletonBeanRegistry：SingletonBeanRegistry接口的实现；
  + HierarchicalBeanFactory：继承了BeanFactory，增加了parentFactory的支持；
  + BeanDefinitionRegistry：定义对BeanDefinition的各种增删改操作；
  + FactoryBeanRegistrySupport：在DefaultSingletonBeanRegistry的基础上增加了对FactoryBean的处理；
  + ConfigurableBeanFactory：提供配置Factory的各种方法；
  + ListableBeanFactory：根据各种条件获取bean的配置清单；
  + AbstractBeanFactory：综合FactoryBeanRegistrySupport与ConfigurableBeanFactory的功能；
  + AutowireCapableBeanFactory：提供创建bean、自动注入、初始化以及应用bean的后处理器；
  + AbstractAutowireCapableBeanFactory：综合AbstractBeanFactory与AutowireCapableBeanFactory的功能；
  + DefaultListableBeanFactory：综合上面的所有的功能；

XmlBeanFactory对DefaultListableBeanFactory类进行了扩展，主要用于从XML文件中读取BeanDefinition。

* XMLbeanDefinitionReader:
* ResourceLoader: 定义资源加载器根据给定的资源文件地址返回对应的Resource；
* BeanDefinitionReader：定义资源文件读取并转换为BeanDefinition的功能；
* EnvironmentCapable: 定义获取Environment的方法；
* DoucmentLoader：资源文件加载到转换为Document的功能；
* AbstractBeanDefinitionReader：对BeanDefinitionReader、EnvironmentCapable接口功能的实现；

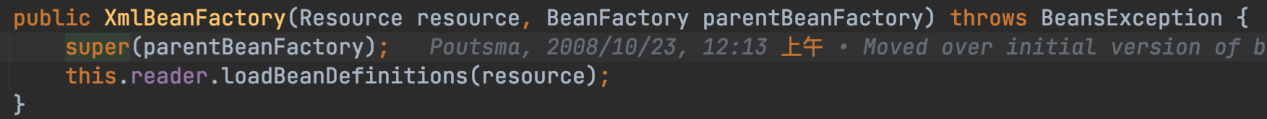
## 2.5 容器的基础：XMLBeanFactory

XmlBeanFactory首先通过ClasspathResource获取资源，得到资源定义；Java中不同来源资源抽象为URL，由注册的不同的handler处理，使用URL的前缀（协议）识别：比如file: http: jar:等；URL没有相对classpath的handler，Spring内部做了自己的抽象结构。



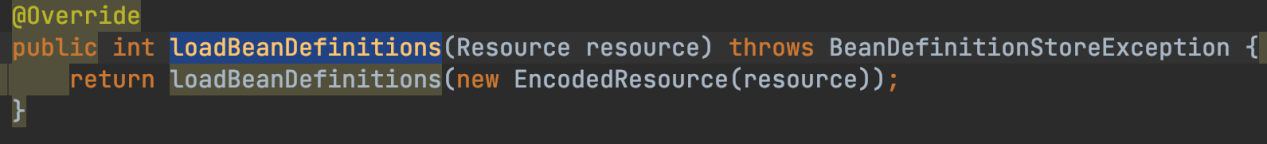
InputStreamSource封装任何能够返回InputStream的类，Resource接口抽象了底层资源的所有的操作，有状态判断的方法、获取URL\URI等方法，获取资源的元数据的一些方法；不同来源的资源文件都有对应的Resource的实现。

ClasspathResource的底层使用的class或者ClassLoader提供的底层方法得到的InputStream；

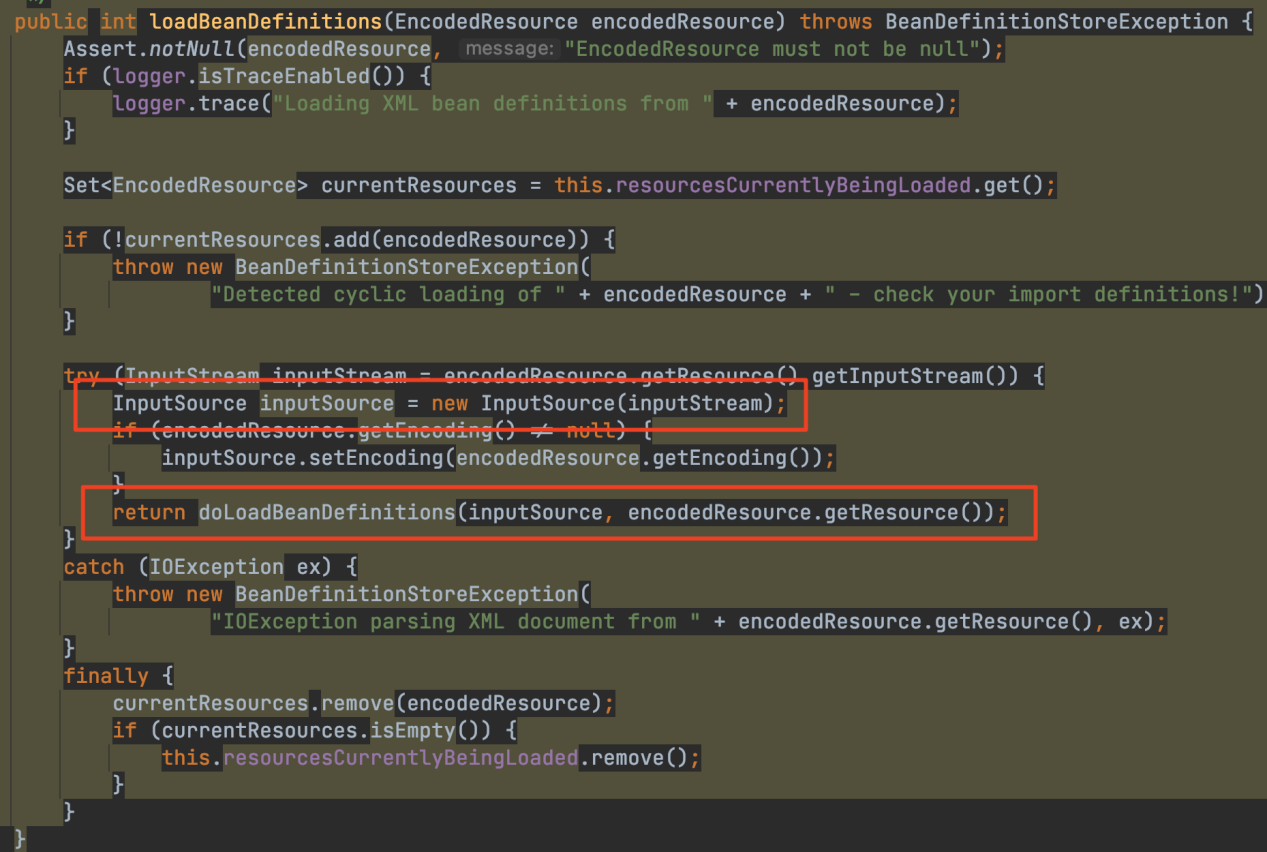


This.reader.loadBeanDefinitions(Resource)是真正的资源加载的逻辑点，

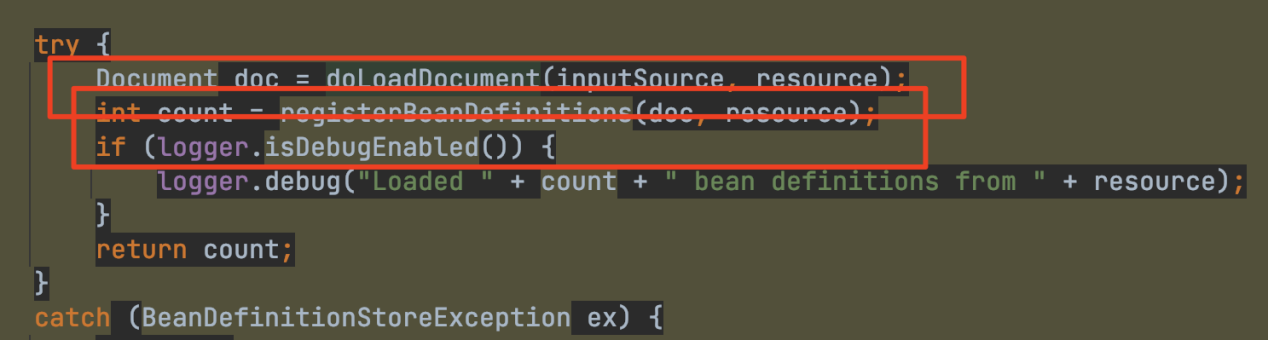
* + 首先转换为有编码的Resource



* 获取InputSource



* + 生成Document并注册bean



## 2.6 获取XML的验证模式

DTD与XSD验证模式。DTD=Document Type Definition=文档类型定义；XML的约束模式语言，用于验证XML文档是否符合规范，包含：

* 元素的定义规则；
* 元素间关系的定义规则；
* 元素可使用的属性；
* 可使用的实体或符号规则。

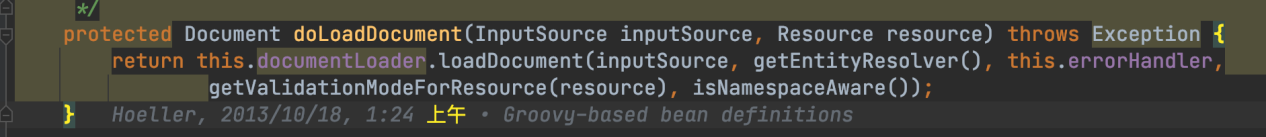
DTD的例子：

!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

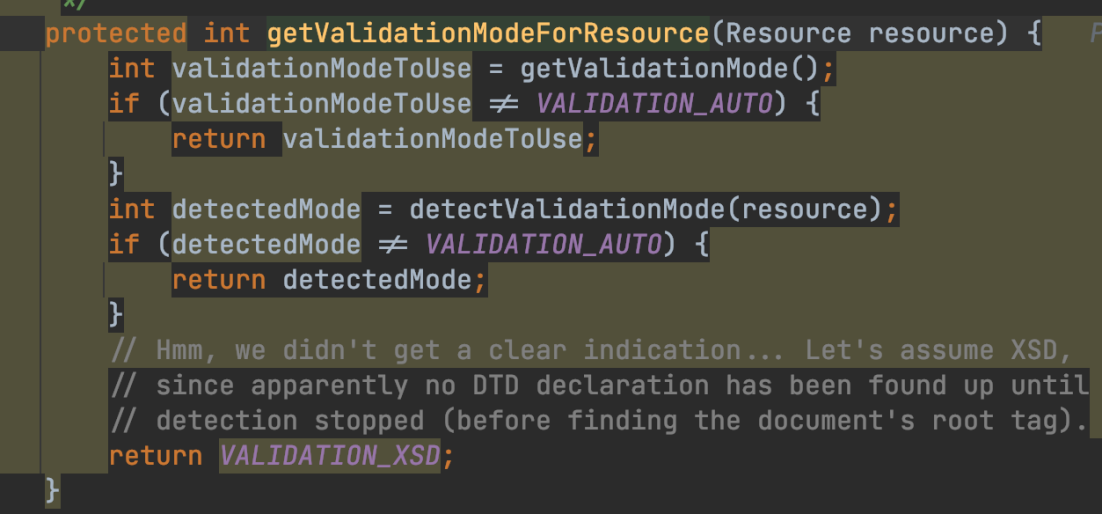
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

XSD是XML Schema Definition的简写，XSD描述了XML文档的结构，

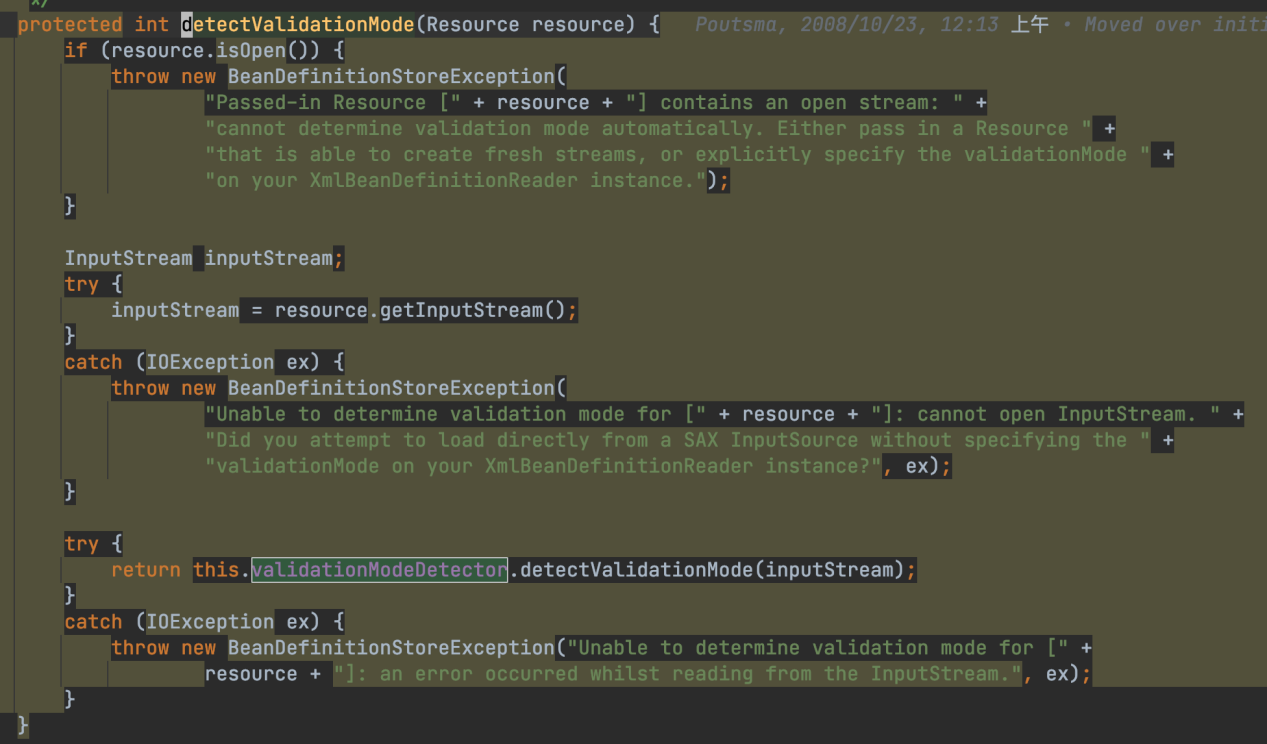
加载Bean资源，形成Document的代码如下：



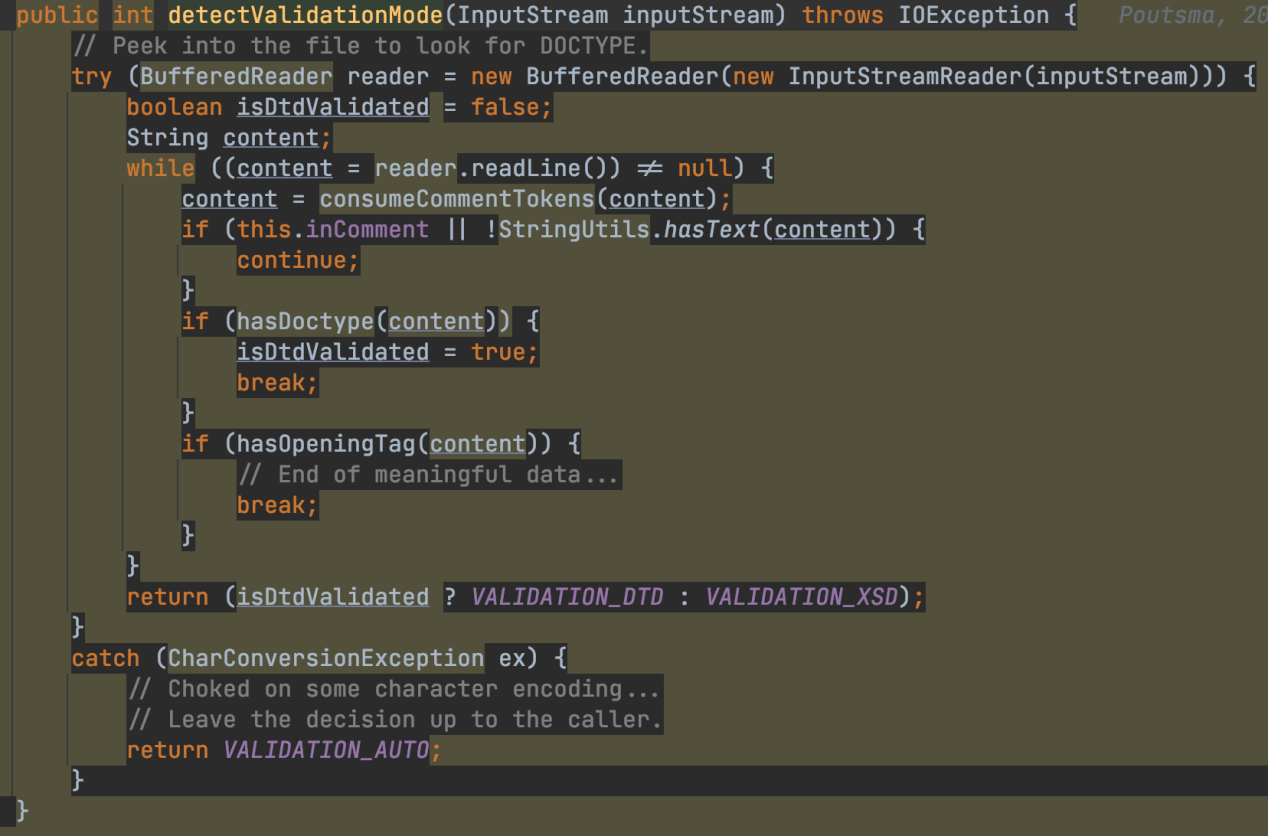
里面调用的获得XML文档验证的代码如下：



调用内部方法，判断验证模式：

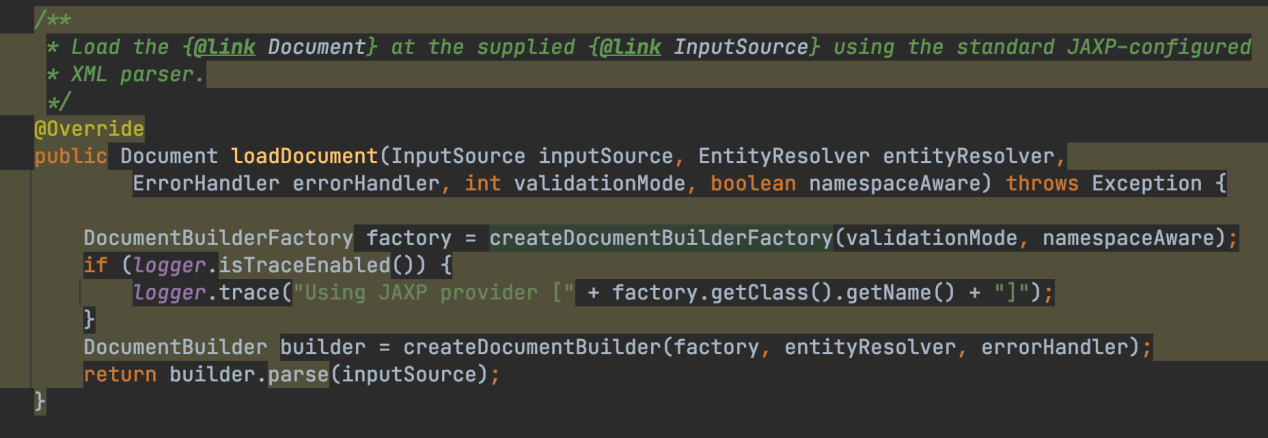


内部使用的方法：



## 2.7 获取Document

加载为Document的过程如下：

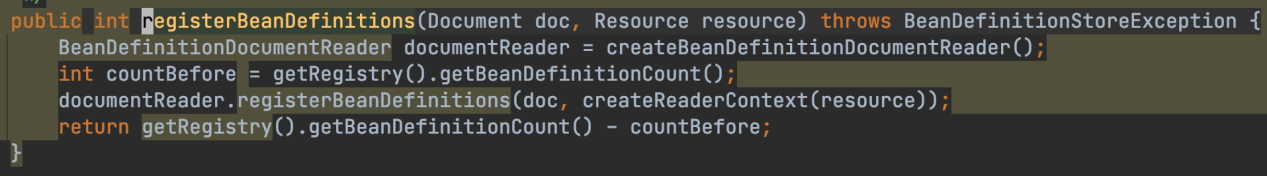


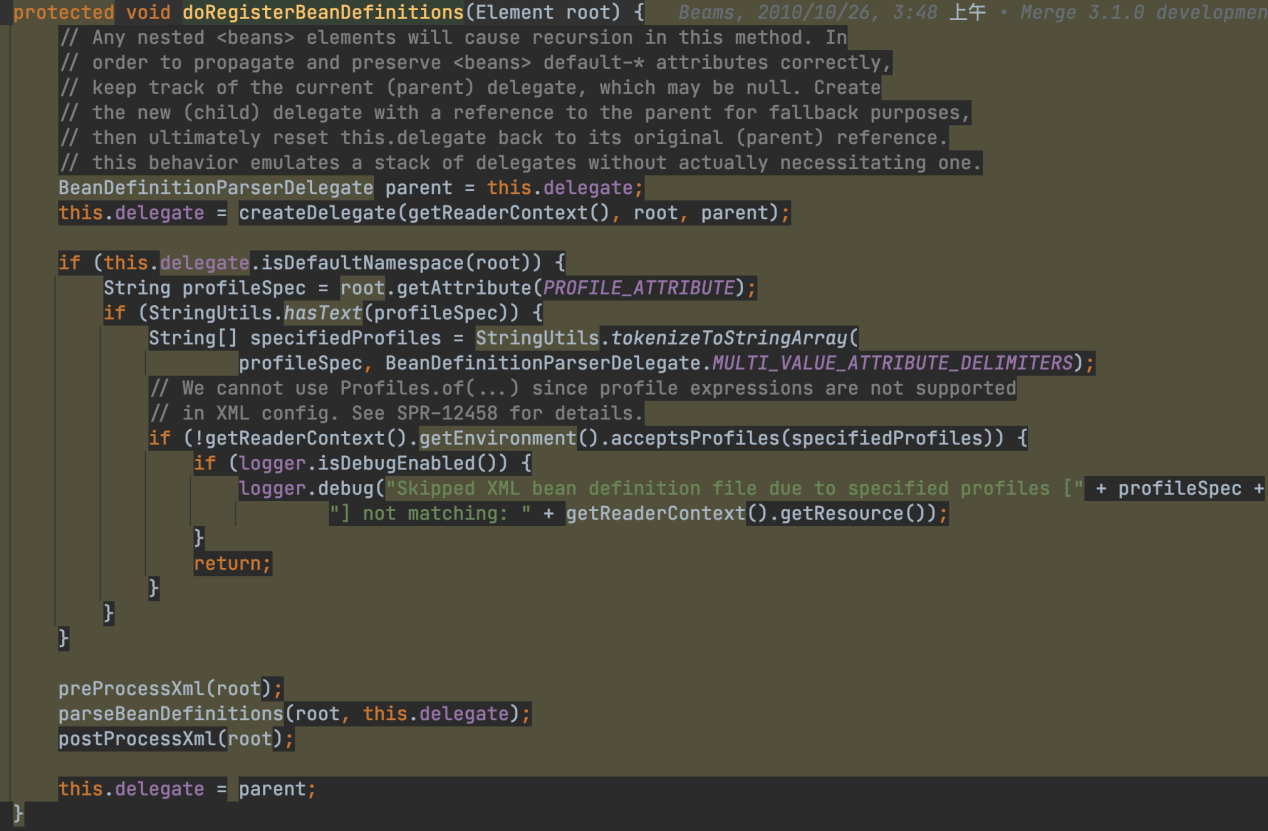


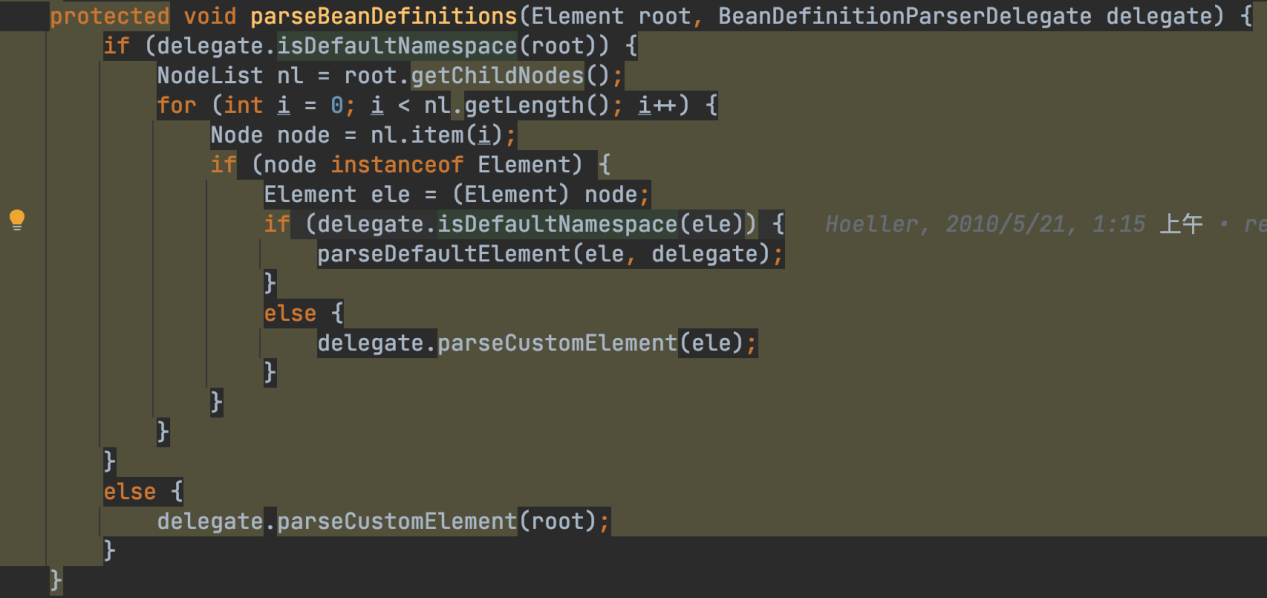
EntityResolver是用来解析自定义DTD、XSD文件位置的。

## 2.8 解析与注册BeanDefinitions

注册Bean定义的方法：



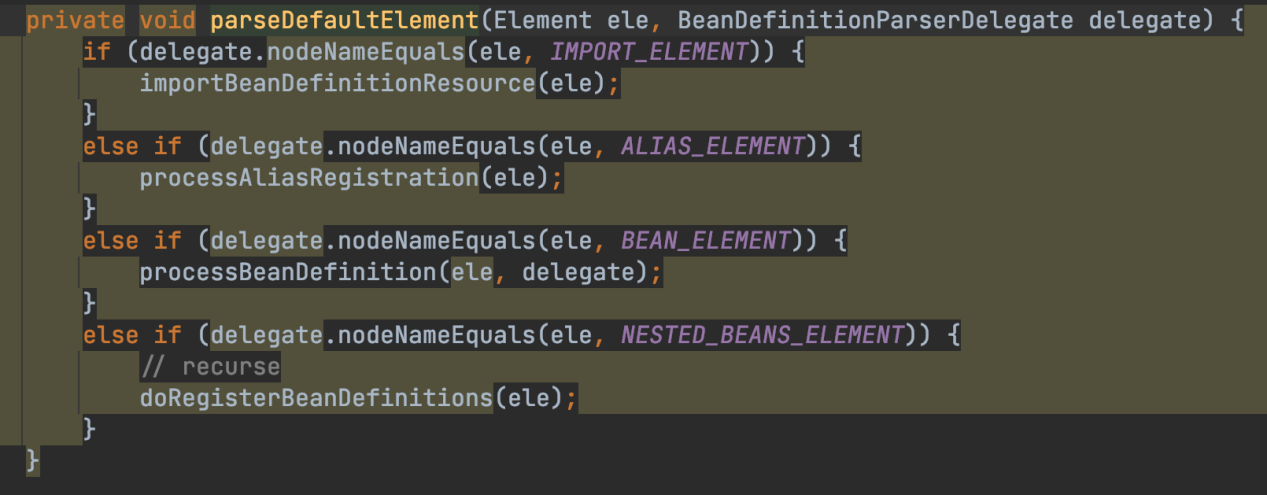




Spring的XML配置里面有2大声明，一个是在默认空间下的，也就是文档标题中声明的xmlns的内容，比如<bean id=xxx,class=xxxxx>，还有就是自定义空间的比如<tx:annotion-driven>等带冒号的。

# 默认标签的解析

上节说到Spring对配置文件的标签处理分为2种，默认标签解析与自定义标签解析，下面看下默认标签解析的逻辑：



可以看到主要分为了4种情况处理：import、alias、bean、beans。

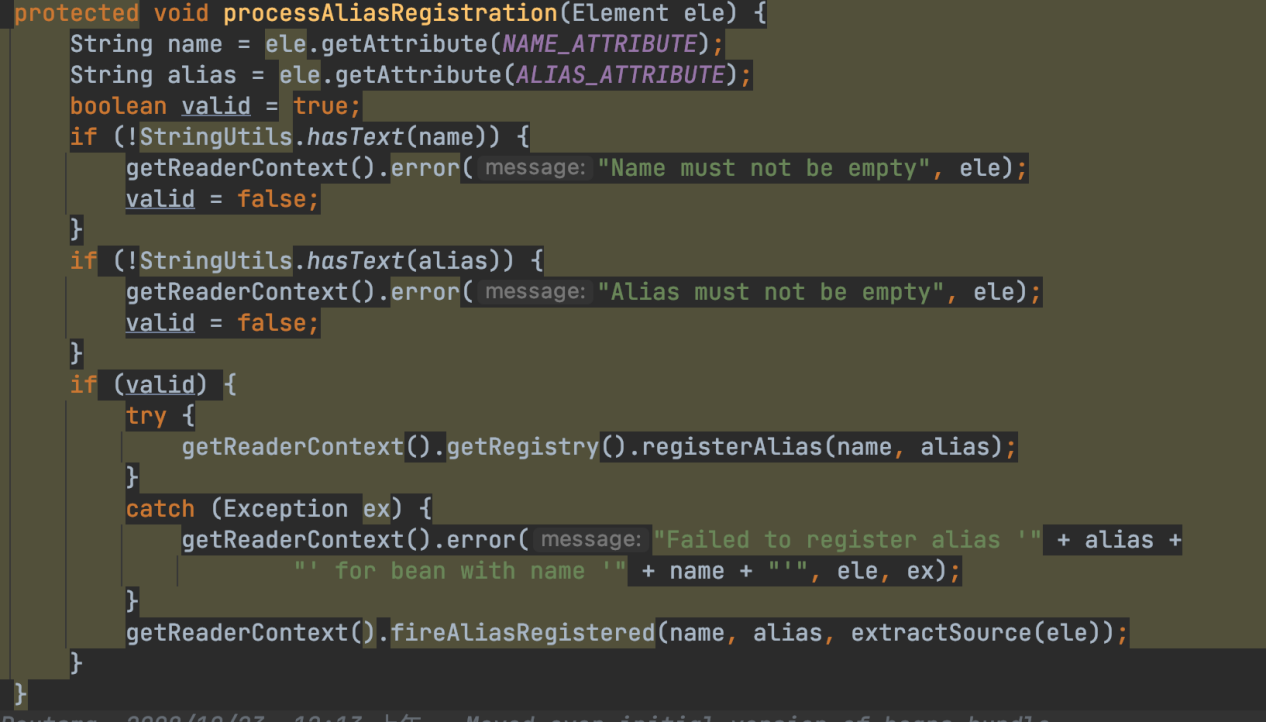
## 3.1 bean标签的解析与注册



首先，委托BeanDefinitionDelegate的parseBeanDefinitionElement进行元素解析，返回BeanDefinitionHolder，

首先看BeanDefinitionHolder bdHolder = delegate.parseBeanDefinitionElement(ele)这个方法，里面解析了<bean>标签的内容，并生成了BeanDefintion，BeanDefintion是<bean>在容器中的内部表示形式，有3种实现.

## 3.2 Alias标签的解析



## 3.3 import标签的解析

# 自定义标签的解析

## 4.1 自定义标签使用

# Bean的加载