# Spring源码解析

# 准备：

## 源码获取的两种方式

1. 通过maven获取源码包
2. Githhub获取 <https://github.com/spring-projects/spring-framework.git>

Spring使用gradle进行版本管理，idea可以默认导入。

执行过程中可能出现找不到cglib和objenesis两个jar包，可以通过执行Spring-core>>Tasks>>other>>cglibRepackJar和objnesisRepackJar两个指令来解决。

2.1 改变依赖包命名空间

Spring源码使用jarjar的方式，将获取cglib和objenesis包反编译，通过修改常量池方式，改变原始的包路径以及所有引用关系。这种方式，可以避免工程与依赖之间的类型冲突。

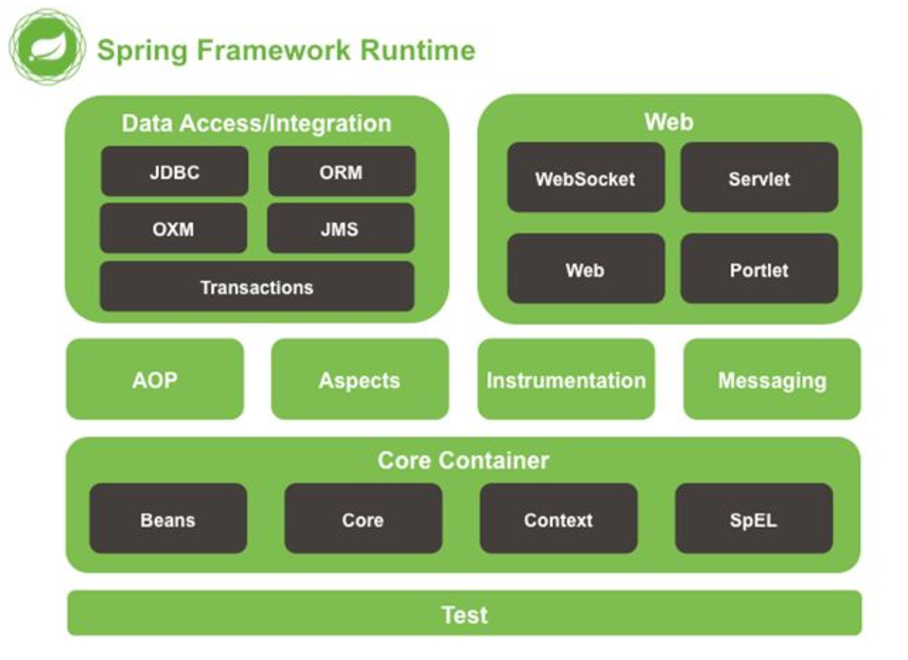
**路径映射：**

**pattern**: **"net.sf.cglib.\*\*"**, **result**: **"org.springframework.cglib.@1"**

**pattern**: **"org.objectweb.asm.\*\*"**, **result**: **"org.springframework.asm.@1"**

**pattern**: **"org.objenesis.\*\*"**, **result**: **"org.springframework.objenesis.@1"**

# 整体架构



Core:各种工具类 StringUtils ClassUtils ReflectUtils

Beans：资源读取、解析、初始化、获取，IOC的基石

Context：依托于Beans组件，是事件传播、国际化、后置处理器等运作的舞台

## 一．spring-ioc

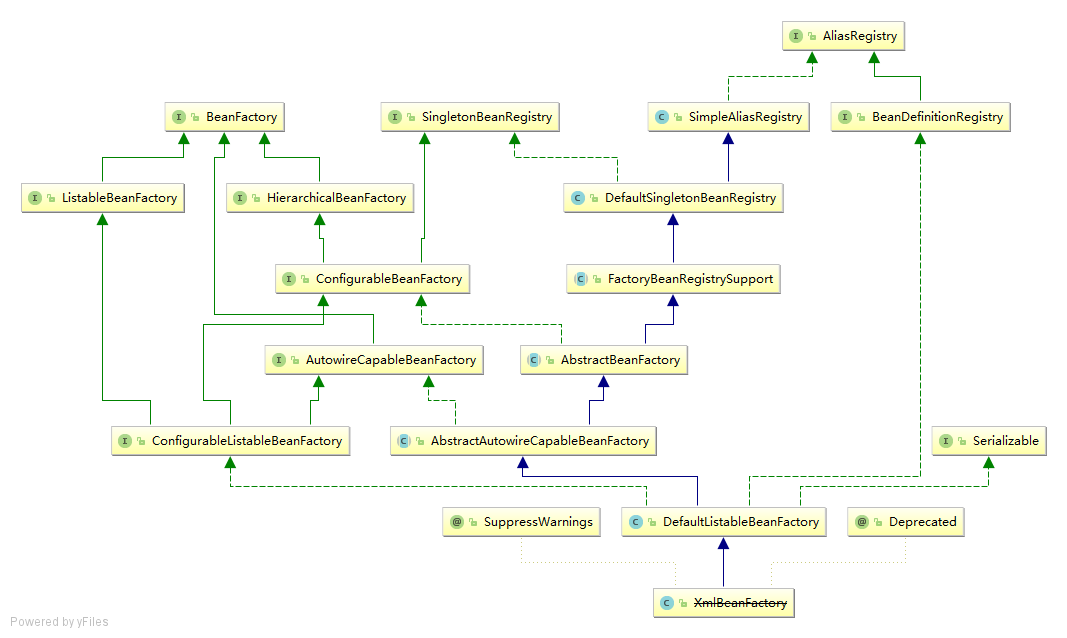
### 1.1 一个简单的使用

**public static void** main(String[] args) {  
 XmlBeanFactory factory = **new** XmlBeanFactory(**new** ClassPathResource(**"beans.xml"**));  
 factory.getBean(**"bean"**);  
}

实现的功能：

1. 配置文件读取
2. 配置解析（包括但不限于xml中定义的id、name、class、property、scope、init-method等标签），生成代表bean身份的数据结构beanDefinition
3. 根据beanDefinition，实例化bean，并进行属性填充（实现自动注入、单例下的循环依赖）。
4. 注册实例
5. 查询实例并调用

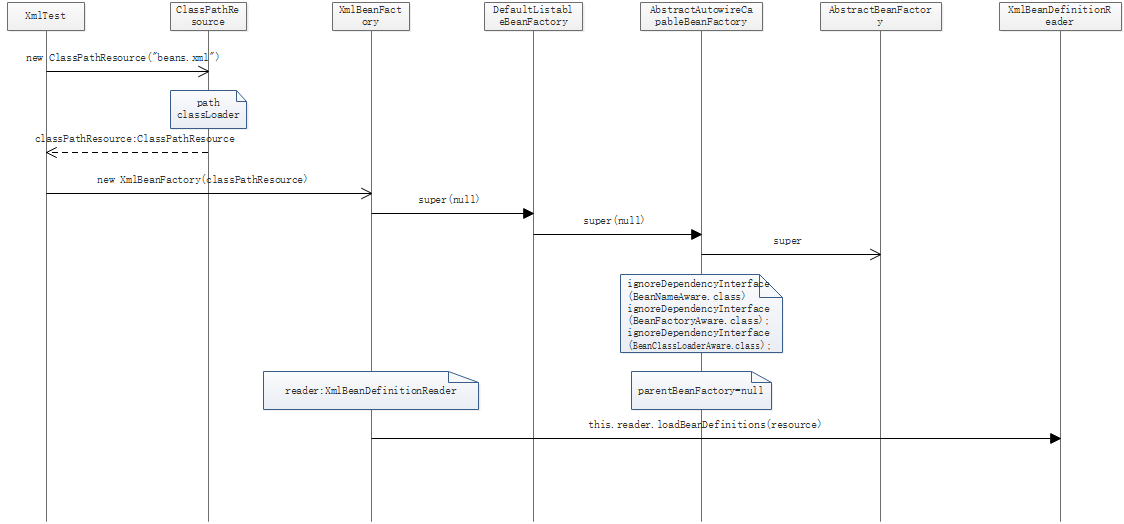
## 1.2 xmlBeanFactory结构



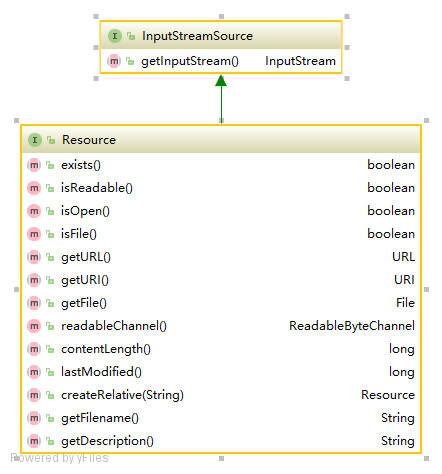
结构说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 描述 | 主要数据结构和方法 |
| AliasRegistry | 接口 | 别名的CRUD规范 | registerAlias  removeAlias  getAliases |
| BeanDefinitionRegistry | 接口 | BeanDefinition CRUD规范  实现AliasRegistry接口 | registerBeanDefinition  removeBeanDefinition  getBeanDefinition |
| SimpleAliasRegistry | 类 | 别名的CRUD具体实现 | Map<String, String> aliasMap |
| SingletonBeanRegistry | 接口 | 单例的注册和获取规范 | registerSingleton  getSingleton |
| BeanFactory | 接口 | Bean的获取规范 | getBean |
| DefaultSingletonBeanRegistry | 类 | 单例的获取 | Map<String, Object> singletonObjects  Map<String, ObjectFactory<?>> singletonFactories  Map<String, Object> earlySingletonObjects  Set<String> registeredSingletons  Map<String, Set<String>> containedBeanMap  Map<String, Set<String>> dependentBeanMap  Map<String, Set<String>> dependenciesForBeanMap |
| HierarchicalBeanFactory | 接口 | Bean的继承，类似java类装载机制，优先父工厂加载 | getParentBeanFactory  containsLocalBean |
| ListableBeanFactory | 接口 | Bean的批量获取规范 | getBeanNamesForType  getBeansOfType |
| FactoryBeanRegistrySupport | 抽象类 | 支持对factoryBean的处理 | Map<String, Object> factoryBeanObjectCache  getObjectFromFactoryBean  getFactoryBean |
| ConfigurableBeanFactory | 接口 | Bean转换、解析规范 | setBeanExpressionResolver  setConversionService  addPropertyEditorRegistrar  registerCustomEditor  setTypeConverter  addBeanPostProcessor  registerScope  getMergedBeanDefinition |
| AbstractBeanFactory | 抽象类 | Bean的获取和属性设置 | BeanExpressionResolver beanExpressionResolver  ConversionService conversionService  Set<PropertyEditorRegistrar> propertyEditorRegistrars  Map<Class<?>, Class<? extends PropertyEditor>> customEditors  TypeConverter typeConverter  List<BeanPostProcessor> beanPostProcessors  Map<String, RootBeanDefinition> mergedBeanDefinitions |
| AutowireCapableBeanFactory | 接口 | 提供自动创建bean、注入、初始化以及后置处理规范 | autowireBean  createBean  applyBeanPropertyValues  initializeBean  applyBeanPostProcessorsBeforeInitialization  applyBeanPostProcessorsAfterInitialization  resolveDependency |
| AbstractAutowireCapableBeanFactory | 抽象类 | 提供自动创建bean、注入、初始化以及后置处理以及类的获取和属性设置 | InstantiationStrategy instantiationStrategy  ParameterNameDiscoverer parameterNameDiscoverer  Set<Class<?>> ignoredDependencyTypes  Set<Class<?>> ignoredDependencyInterfaces |
| ConfigurableListableBeanFactory | 接口 | 忽略指定类型 | ignoreDependencyType |
| DefaultListableBeanFactory | 类 | 注册、获取、解析、忽略等 | Map<Class<?>, Object> resolvableDependencies  Map<String, BeanDefinition> beanDefinitionMap  Map<Class<?>, String[]> allBeanNamesByType  Map<Class<?>, String[]> singletonBeanNamesByType  List<String> beanDefinitionNames  Set<String> manualSingletonNames |

## 1.3 xmlBeanFactory初始化流程



## 1.4 Resource



### 1.4.1 FileUrlResource

继承UrlResource,通过URLStreamHandler获取流

### 1.4.2 classPathResouce

直接读取

**public** InputStream getInputStream() **throws** IOException {  
 InputStream is;  
 **if** (**this**.**clazz** != **null**) {  
 is = **this**.**clazz**.getResourceAsStream(**this**.**path**);  
 }  
 **else if** (**this**.**classLoader** != **null**) {  
 is = **this**.**classLoader**.getResourceAsStream(**this**.**path**);  
 }  
 **else** {  
 is = ClassLoader.*getSystemResourceAsStream*(**this**.**path**);  
 }  
 **if** (is == **null**) {  
 **throw new** FileNotFoundException(getDescription() + **" cannot be opened because it does not exist"**);  
 }  
 **return** is;  
}

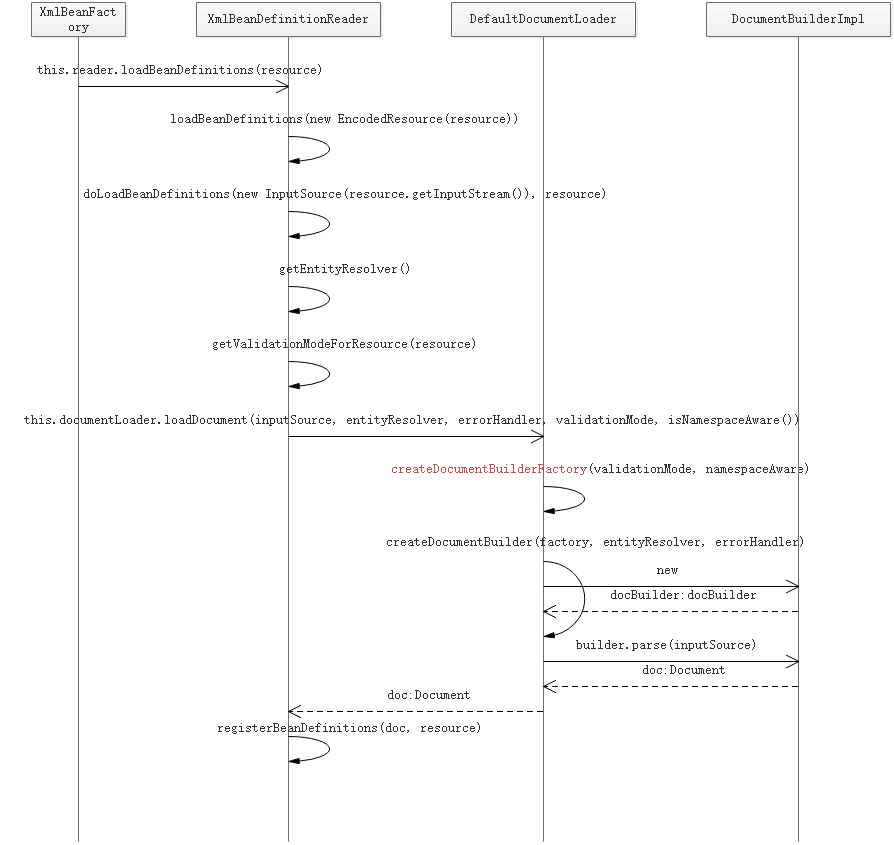
class.getResource(**""**)：当前类路径下

class.getResource(**"/"**)：跟路径下

classLoader.getResourceAsStream(**""**)：跟路径下

classLoader.getResourceAsStream(**"/"**)：不支持

## 1.5 读取配置



### 1.5.1 文档校验模式

Dtd：

*<?***xml version="1.0" encoding="UTF-8"***?>***<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd"*>***<**beans**>  
  
</**beans**>

Xsd:

*<?***xml version="1.0" encoding="UTF-8"***?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"  
  
 default-autowire="byName"**>  
  
</**beans**>

### 1.5.2 dom解析

DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();

factory.setValidating(**true**);

factory.setNamespaceAware(**true**);

factory.setAttribute(**"http://java.sun.com/xml/jaxp/properties/schemaLanguage"**

, **"http://www.w3.org/2001/XMLSchema"**);

DocumentBuilder docBuilder = factory.newDocumentBuilder();

docBuilder.setEntityResolver(entityResolver);

docBuilder.setErrorHandler(errorHandler);

docBuilder.parse(inputSource)