# IOC的简单流程

## IOC简介以及环境搭建

* 1. 简介

1. IOC简称控制反转，把bean对象的创建/初始化/销毁等工作交给spring容器，由容器控制bean对象的生命周期。
   1. Jar包导入
2. Spring-context包含spring-core、spring-beans、spring-aop、spring-expression等基础的jar包。

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

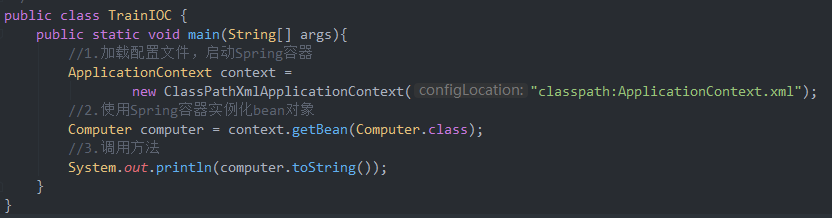
<version>4.3.11.RELEASE</version>

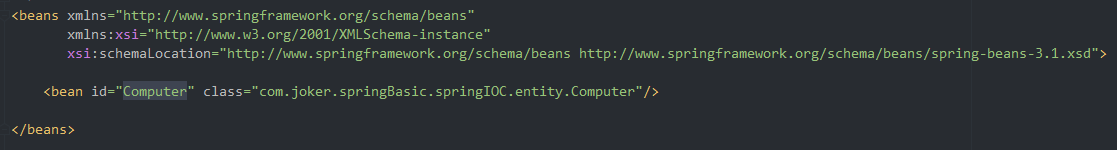
</dependency>

## IOC加载流程

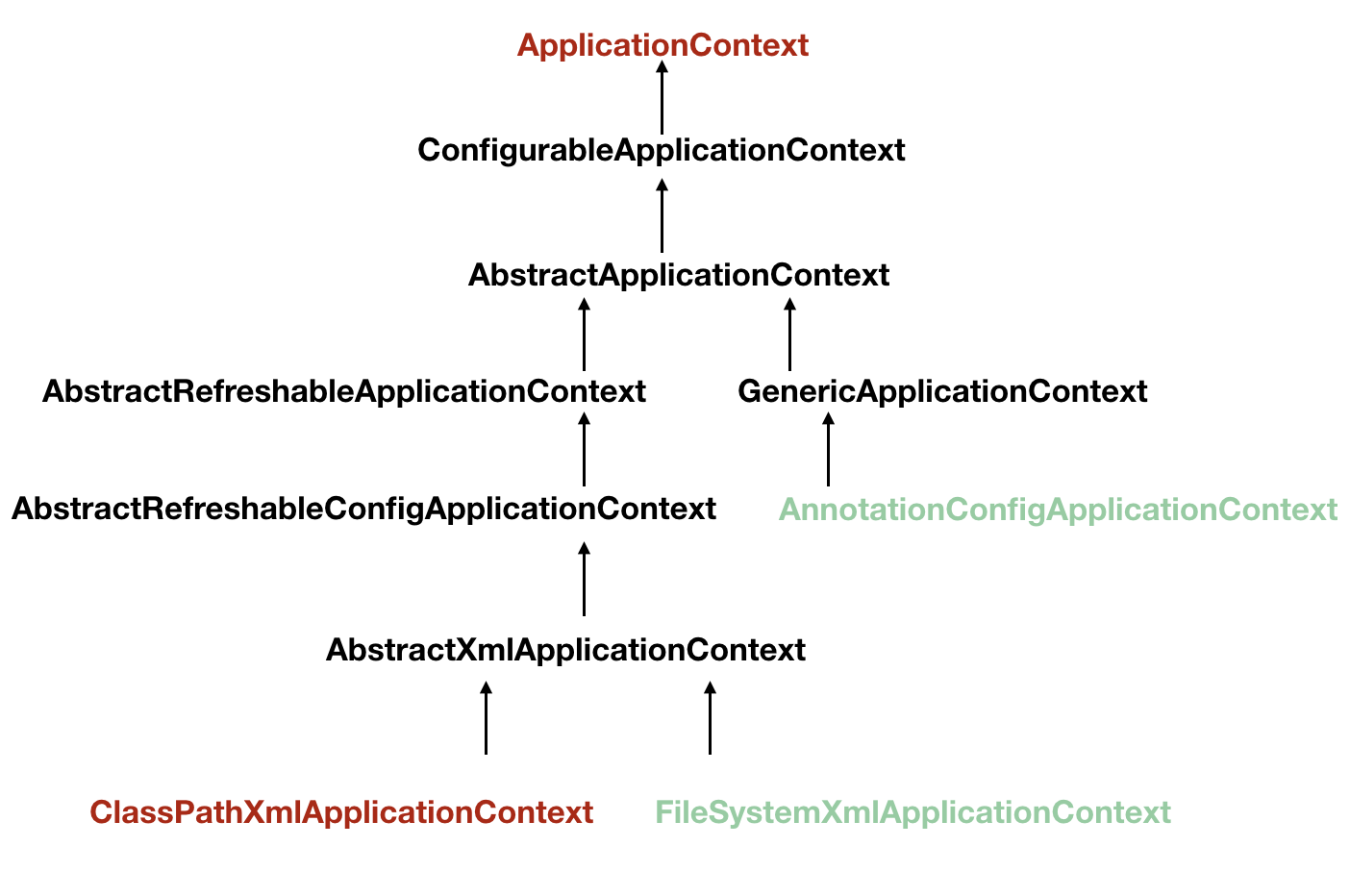
* 1. 基本步骤

1. 加载配置文件，初始化spring容器。
2. Spring容器实例化bean对象。
3. 实例化对象调用方法。





* 1. 加载配置文件



1. 核心是根据配置文件或者注解，来构建spring的ApplicationContext。
2. 构建方式总共有三种：

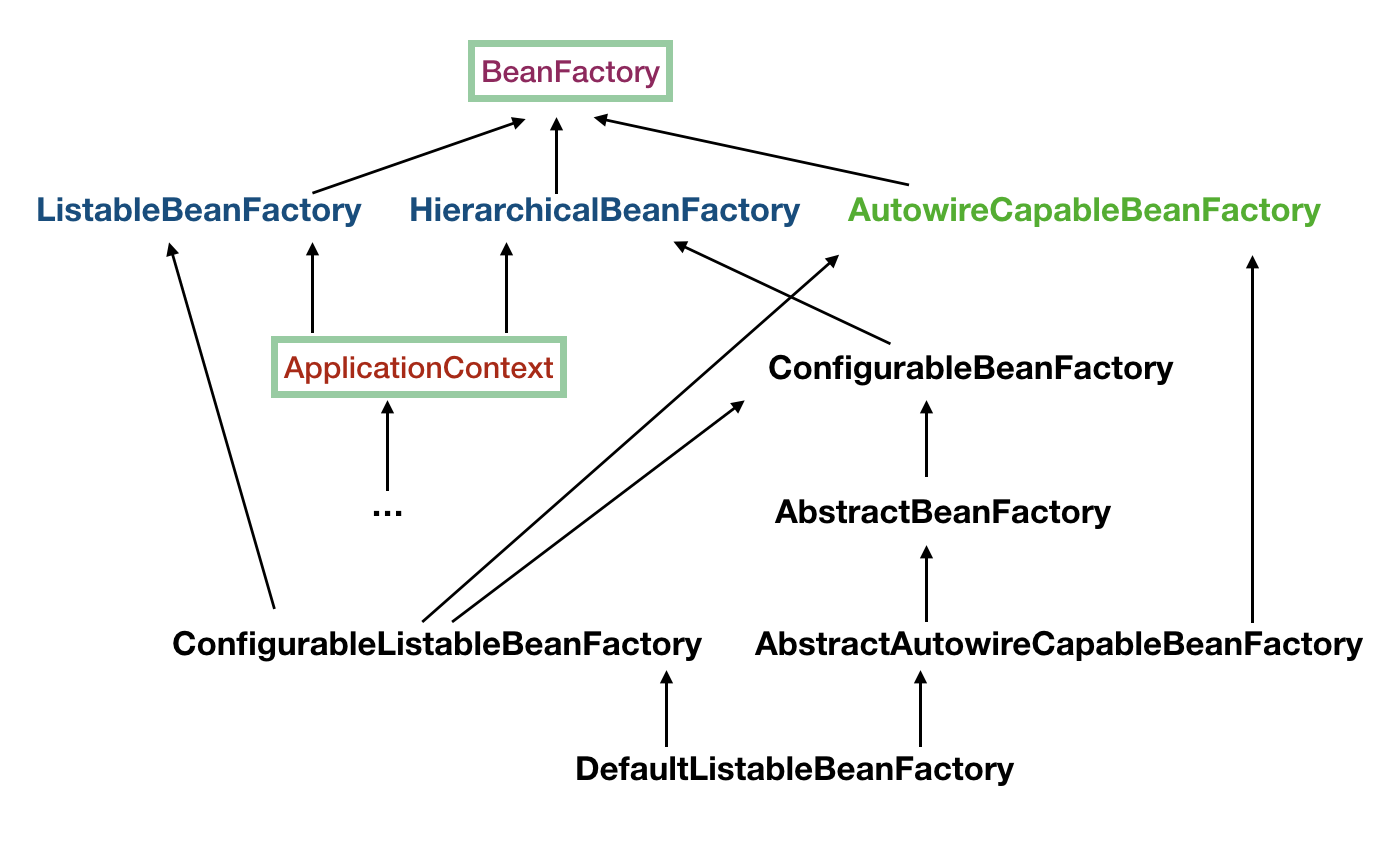
ClassPathXmlApplicationContext：需要xml配置文件的默认路径，maven中是resource.

FileSystemXmlApplicationContext：需要xml配置文件在系统中的路径

AnnotationConfigApplicationContext:采用配置类和注解来配置

# BeanFactory 工厂

## BeanFactory接口主要继承关系



1. ApplicationContext继承了ListableBeanFactory，这个 Listable 的意思就是，通过这个接口**可以获取多个 Bean**，最顶层 BeanFactory 接口的方法都是获取单个 Bean 的。
2. ApplicationContext 继承了 HierarchicalBeanFactory，**可以在应用中起多个 BeanFactory，然后可以将各个 BeanFactory 设置为父子关系**。
3. AutowireCapableBeanFactory用来自动装配Bean的, ApplicationContext 并没有继承它，ApplicationContext 接口定义中的方法 getAutowireCapableBeanFactory() 就可以使用。
4. ConfigurableListableBeanFactory 也是一个特殊的接口，看图，特殊之处在于它继承了第二层所有的三个接口，而 ApplicationContext 没有。

# Spring容器启动过程

1. 准备工作。
2. 创建BeanFactory，加载注册Bean。
3. 实例化Bean容器完成。
4. 准备Bean容器。
5. 初始化所有的sington bean

## ClassPathXmlApplicationContext工厂类

* 1. 主要构造方法

1. 如果已经有ApplicationContext，并且需要配置成父子关系

public ClassPathXmlApplicationContext(ApplicationContext parent) {

super(parent);

}

1. 主要构造方法

public ClassPathXmlApplicationContext(String[] configLocations, boolean refresh, ApplicationContext parent) throws BeansException {

super(parent);

//根据提供的路径，处理成配置文件数组(以分号、逗号、空格、tab、换行符分割)

setConfigLocations(configLocations);

if (refresh) {

//将原来的Application销毁，然后重新执行一次初始化操作

refresh();

}

}

## 核心方法refresh()

见文件refresh.java

1. 作用：销毁原来的ApplicationContext,再重新执行一次初始化操作。

## 创建Bean容器前的准备工作

文件prepareRefresh()，启动前准备方法

1. 记录启动时间。
2. 如果开启Info级别，打印更新日志；
3. 校验配置文件，类Environment

## 开始创建Bean容器

文件obtainFreshBeanFactory(),**创建Bean容器，加载Bean，定义注册Bean**

4.1 实例化Bean容器

1. 创建Bean容器，初始化BeanFactory，加载并注册bean，但未初始化bean实例。
2. ApplicationContext 继承自 BeanFactory，但是它不应该被理解为 BeanFactory 的实现类，而是说**其内部持有一个实例化的 BeanFactory（DefaultListableBeanFactory）**。以后所有的 BeanFactory 相关的操作其实是委托给这个实例来处理的。
3. DefaultListableBeanFactory实例实现了BeanFactory底下所有的三个接口，可以实现创建多个Bean，设置父子关系以及自动装配Bean三个功能。

4.2 加载Bean（spring bean）

(1) BeanDefinition接口是SpringBean的定义接口，在Java中定义的Bean会转换成一个个BeanDefinition，再存储在BeanFactory中。Bean在代码层面上，可以认为是

BeanDefinition的实例。

1. BeanDefinition 中保存了Bean 信息，比如这个 Bean 指向的是哪个类、是否是单例的、是否懒加载、这个 Bean 依赖了哪些 Bean 等等。

4.3

## 实例化Bean容器完成后

## 准备Bean容器prepareBeanFactory

## 初始化所有的sington beans