## Spring项目启动后的加载流程

ContextLoaderListener通过调用继承自ContextLoader的initWebApplicationContext方法实例化SpringIoc容器。

在实例化Spring IoC容器的过程中，最主要的两个方法是createWebApplicationContext和configureAndRefreshWebApplicationContext方法。

createWebApplicationContext方法用于返回XmlWebApplicationContext实例，即Web环境下的SpringIoC容器。

configureAndRefreshWebApplicationContext用于配置XmlWebApplicationContext，读取web.xml中通过contextConfigLocation标签指定的XML文件，通过调用refresh来调用AbstractApplicationContext中的refresh初始化：

1. BeanFactory并实例化XML文件中配置的bean，Spring将配置文件的bean的信息解析成为一个个的BeanDefinition对象并装入到容器的Bean定义注册表，但此时Bean还未初始化；obtainFreshBeanFactory()会调用自身的refreshBeanFactory(),而refreshBeanFactory()方法由子类AbstractRefreshableApplicationContext实现，该方法返回了一个创建的DefaultListableBeanFactory对象，这个对象就是由ApplicationContext管理的BeanFactory容器对象。

2. 调用工厂后处理器：根据反射机制从BeanDefinitionRegistry中找出所有BeanFactoryPostProcessor类型的Bean，并调用其postProcessBeanFactory()接口方法

经过第一步加载配置文件，已经把配置文件中定义的所有bean装载到BeanDefinitionRegistry这个Beanfactory中，对于ApplicationContext应用来说这个BeanDefinitionRegistry类型的BeanFactory就是Spring默认的DefaultListableBeanFactory。容器扫描BeanDefinitionRegistry中的BeanDefinition，使用java反射自动识别出Bean工厂后处理器的bean，然后调用这些bean工厂后处理器对BeanDefinitionRegistry中的BeanDefinition进行加工处理，具体步骤包括：Resource定位、BeanDefinition的载入、向IOC容器注册这些BeanDefinition：

1. 对使用到占位符的<bean>元素标签进行解析，得到最终的配置值，这意味着对一些半成品式的BeanDefinition对象进行加工处理并取得成品的BeanDefinition对象。、

2. 对BeanDefinitionRegistry中的BeanDefinition进行扫描，通过Java反射机制找出所有属性编辑器的Bean，并自动将它们注册到Spring容器的属性编辑器注册表中

3. 注册Bean后处理器：根据反射机制从BeanDefinitionRegistry中找出所有BeanPostProcessor类型的Bean，并将它们注册到容器Bean后处理器的注册表中；

4. 初始化消息源：初始化容器的国际化信息资源；

5. 初始化应用上下文事件广播器

6. 初始化其他特殊的Bean

7. 注册事件监听器

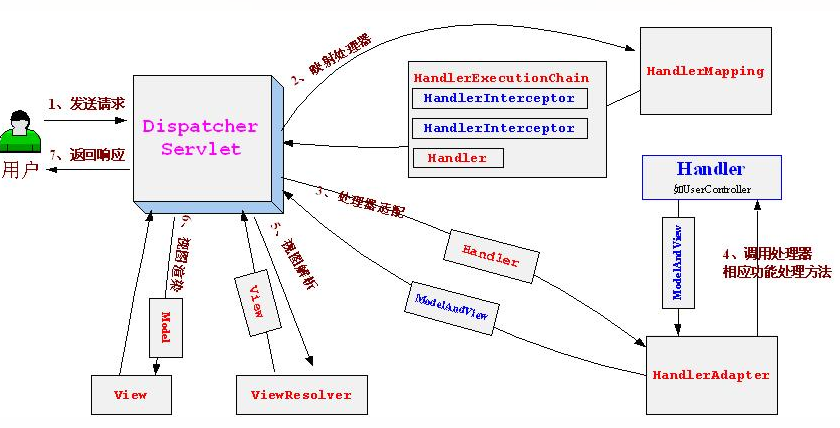
8. 初始化singleton的Bean：实例化所有singleton的Bean，并将它们放入Spring容器的缓存中

9. 发布上下文刷新事件：在此处时容器已经启动完成，发布容器refresh事件创建上下文刷新事件，事件广播器负责将些事件广播到每个注册的事件监听器中。

依赖注入的原理

依赖注入的过程是用户第一次向IoC容器索要Bean时触发的。当用户向IoC容器索要Bean时会触发getBean方法，之后使用CGLIB通过反射生成Bean对象，之后将这些对象的依赖关系通过根据BeanDefinition的信息完成两个递归从而完成依赖注入。一个递归是在上下文体系中查找需要的Bean和创建Bean的递归调用；另一个递归是在依赖注入时通过递归调用容器的getBean方法得到当前Bean的依赖Bean，同时也触发对依赖Bean的创建和注入

## SpringMVC请求响应流程



1、DispatcherServlet前端控制器接收发过来的请求，交给HandlerMapping处理器映射器

2、HandlerMapping处理器映射器，根据请求路径找到相应的HandlerAdapter处理器适配器（处理器适配器就是那些拦截器或Controller）

3、HandlerAdapter处理器适配器，处理一些功能请求，返回一个ModelAndView对象（包括模型数据、逻辑视图名）

4、ViewResolver视图解析器，先根据ModelAndView中设置的View解析具体视图

5、然后再将Model模型中的数据渲染到View上

Spring事务失效的几大原因

如使用mysql且引擎是MyISAM，则事务会不起作用，原因是MyISAM不支持事务，可以改成InnoDB

如果使用了spring+mvc，则context:component-scan重复扫描问题可能会引起事务失败

@Transactional 注解开启配置，必须放到listener里加载，如果放到DispatcherServlet的配置里，事务也是不起作用的。

@Transactional 注解只能应用到 public 可见度的方法上。 如果你在 protected、private 或者final、static的方法上使用 @Transactional 注解，它也不会报错，事务也会失效。

spring内部catch的就是 RuntimeException, service抛出RuntimeException可以回滚

## 说一下Spring的核心模块

Spring Core 核心容器：提供Spring的基本功能IOC

Spring AOP 面向切面：提供面向切面编程支持，基于代理实现

Spring ORM 对象关系映射：第三方持久层框架整合

Spring Context 应用上下文：提供Bean的访问方法

## Spring Bean的作用域

Singleton:ioc容器中存放一个共享的bean示例，默认

Prototype:每次通过容器获取bean时，容器都会创建一个新的实例

Request:一次请求中容器会返回该bean的同一实例

Session:一次session中容器会返回该bean的同一实例

Global Session : 略

## Spring Bean 的生命周期

1. 实例化一个bean
2. 按照Spring上下文的配置进行ioc注入
3. BeanNameAware接口 setBeanName()
4. BeanFactoryAware 接口 setBeanFactory()
5. ApplicationContextAware 接口 setApplicationContext()
6. BeanPostProcessor接口 postProcessBeforeInitialization() 缓存技术？
7. Init-method
8. BeanPostProcessor接口 postProcessAfterInitialization()
9. DisposableBean 接口 destory()方法
10. Destory-method

## 谈谈Spring框架中的设计模式

代理模式 AOP

单利模式 Bean

模板方法

前段控制器模式

工厂模式

## BeanFactory和ApplicationContext的区别

ApplicationContext 是 BeanFactory的子接口

BeanFactory主要提供了getBean方法

ApplicationContext额外提供了：

国际化

消息发送

响应机制

统一加载资源

事件机制

## 数据库的三大范式

1.第一范式 数据表中的每一列都是不可拆分的最小单元

2.第二范式 一张表只描述一件事，如果不只有一件，需要拆分

3.第三范式 所有列必须依赖主键，和主键相关

## 什么是TCP UDP

TCP(Transmission Control Protocol 传输控制协议)

是一种面向连接的可靠的，基于字节流的传输层协议

UDP(User Datagram Protocol 用户数据报协议)

是一种无连接的面向数据报的传输层协议

## TCP和UDP的区别

TCP面向连接 UDP无连接

TCP 可靠 UDP 不可靠

TCP 基于字节流 UDP 基于数据报

TCP 点对点一对一 UDP 一对多