



## ♥Spring框架知识体系详解♥

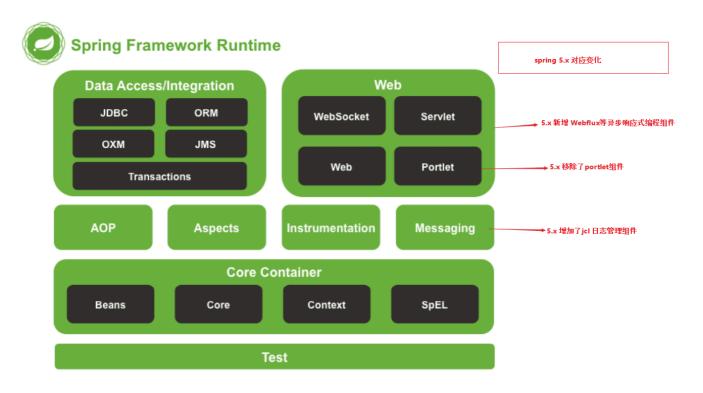
## 提示

本系列主要介绍Spring框架整体架构,Spring的核心IOC,AOP的案例和具体实现机制;以及SpringMVC框架的案例和实现机制。 @pdai

## 相关文章

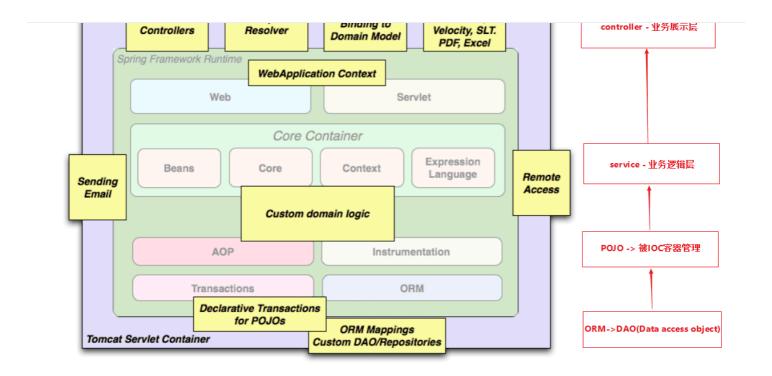
首先,从Spring框架的整体架构和组成对整体框架有个认知。

- Spring基础 Spring和Spring框架组成
  - 。 Spring是什么? 它是怎么诞生的? 有哪些主要的组件和核心功能呢? 本文通过这几个问题帮助你构筑Spring和Spring和Spring Framework的整体认知。

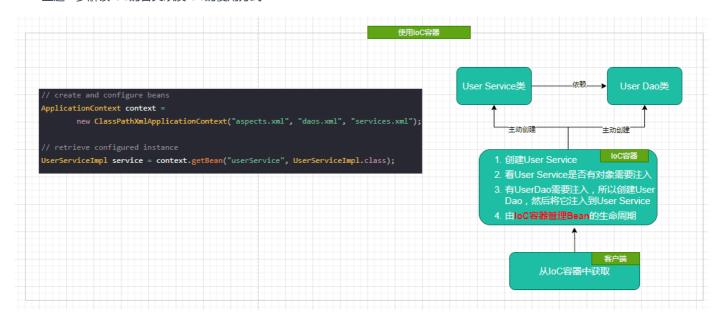


其次,通过案例引出Spring的核心(IoC和AOP),同时对IoC和AOP进行案例使用分析。

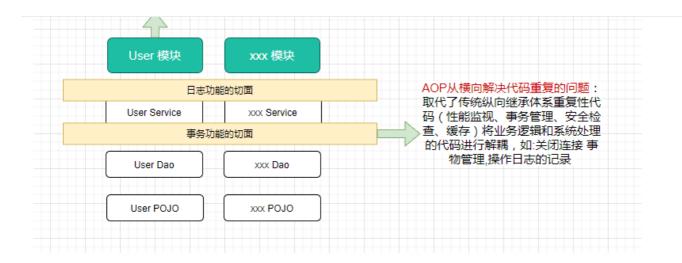
- Spring基础 Spring简单例子引入Spring的核心
  - 。 上文中我们简单介绍了Spring和Spring Framework的组件,那么这些Spring Framework组件是如何配合工作的呢? 本文主要承接上文,向你展示Spring Framework组件的典型应用场景和基于这个场景设计出的简单案例,并以此引出Spring的核心要点,比如IOC和AOP等;在此基础上还引入了不同的配置方式,如XML,Java配置和注解方式的差异。



- Spring基础 Spring核心之控制反转(IOC)
  - 。 在Spring基础 Spring简单例子引入Spring的核心中向你展示了IoC的基础含义,同时以此发散了一些IoC相关知识点; 本节将在此基础上进一步解读IOC的含义以及IOC的使用方式

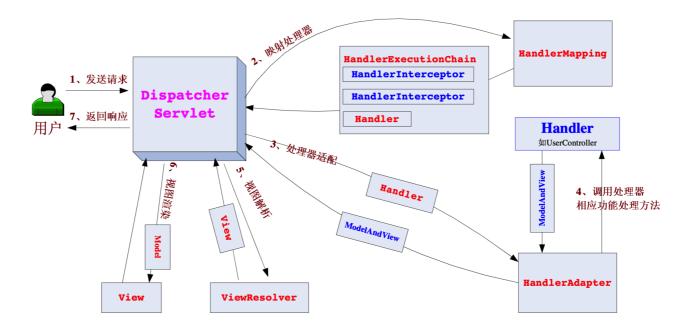


- Spring基础 Spring核心之面向切面编程(AOP)
  - 。 在Spring基础 Spring简单例子引入Spring的核心中向你展示了AOP的基础含义,同时以此发散了一些AOP相关知识点; 本节将在此基础上进一步解读AOP的含义以及AOP的使用方式。



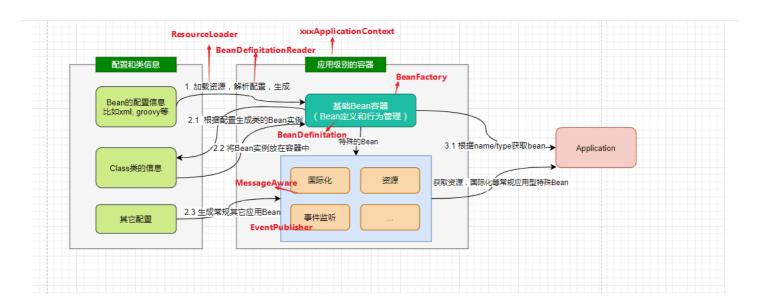
基于Spring框架和IOC, AOP的基础, 为构建上层web应用, 需要进一步学习SpringMVC。

- Spring基础 SpringMVC请求流程和案例
  - 。 前文我们介绍了Spring框架和Spring框架中最为重要的两个技术点(IOC和AOP),那我们如何更好的构建上层的应用呢(比如web 应用),这便是SpringMVC;Spring MVC是Spring在Spring Container Core和AOP等技术基础上,遵循上述Web MVC的规范推出的web 开发框架,目的是为了简化Java栈的web开发。 本文主要介绍SpringMVC的请求流程和基础案例的编写和运行。

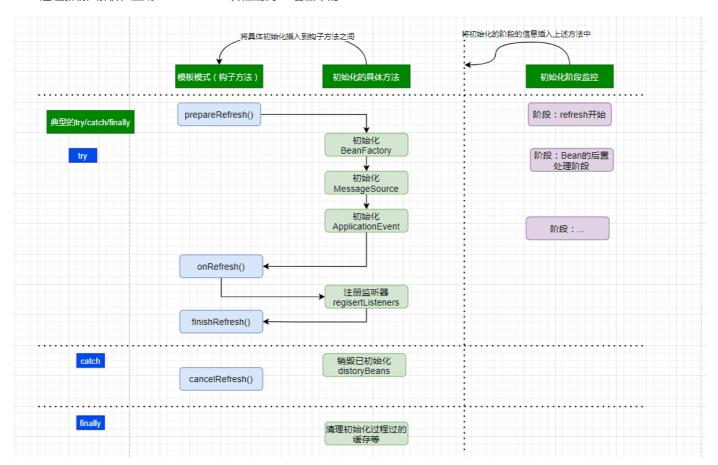


Spring进阶 - IoC, AOP以及SpringMVC的源码分析

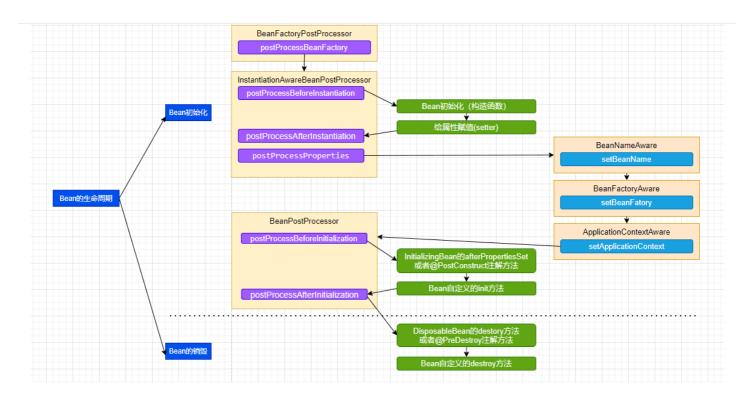
- Spring进阶 Spring IOC实现原理详解之IOC体系结构设计
  - 。 在对IoC有了初步的认知后,我们开始对IOC的实现原理进行深入理解。本文将帮助你站在设计者的角度去看IOC最顶层的结构设计



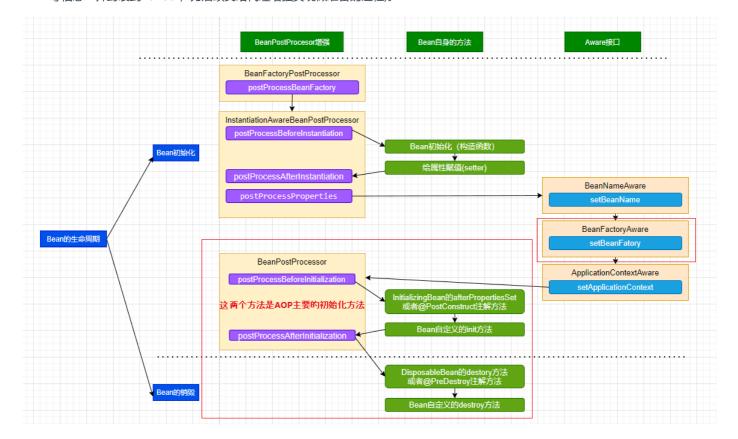
- Spring进阶 Spring IOC实现原理详解之IOC初始化流程
  - 。上文,我们看了IOC设计要点和设计结构;紧接着这篇,我们可以看下源码的实现了:Spring如何实现将资源配置(以xml配置为例) 通过加载,解析,生成BeanDefination并注册到IoC容器中的



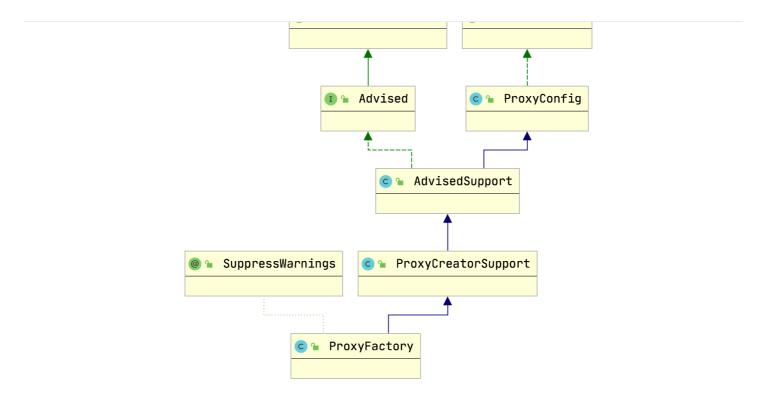
- Spring进阶 Spring IOC实现原理详解之Bean实例化(生命周期,循环依赖等)
  - 。上文,我们看了IOC设计要点和设计结构;以及Spring如何实现将资源配置(以xml配置为例)通过加载,解析,生成BeanDefination 并注册到IoC容器中的;容器中存放的是Bean的定义即BeanDefinition放到beanDefinitionMap中,本质上是一个 ConcurrentHashMap<S tring,Object>;并且BeanDefinition接口中包含了这个类的Class信息以及是否是单例等。那么如何从BeanDefinition中实例化Bean 对象呢,这是本文主要研究的内容?



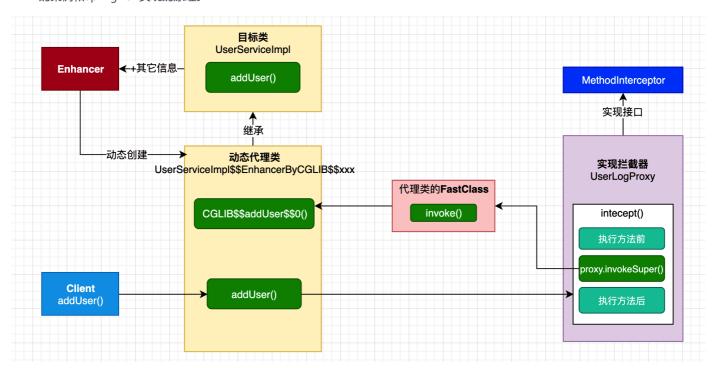
- Spring进阶 Spring AOP实现原理详解之切面实现
  - 。 前文,我们分析了Spring IOC的初始化过程和Bean的生命周期等,而Spring AOP也是基于IOC的Bean加载来实现的。本文主要介绍Spring AOP原理解析的切面实现过程(将切面类的所有切面方法根据使用的注解生成对应Advice,并将Advice连同切入点匹配器和切面类等信息一并封装到Advisor,为后续交给代理增强实现做准备的过程)。



- Spring进阶 Spring AOP实现原理详解之AOP代理
  - 。 上文我们介绍了Spring AOP原理解析的切面实现过程(将切面类的所有切面方法根据使用的注解生成对应Advice,并将Advice连同切入点匹配器和切面类等信息一并封装到Advisor)。本文在此基础上继续介绍,代理(cglib代理和JDK代理)的实现过程。



- Spring进阶 Spring AOP实现原理详解之Cglib代理实现
  - 。 我们在前文中已经介绍了SpringAOP的切面实现和创建动态代理的过程,那么动态代理是如何工作的呢?本文主要介绍Cglib动态代理 的案例和SpringAOP实现的原理。



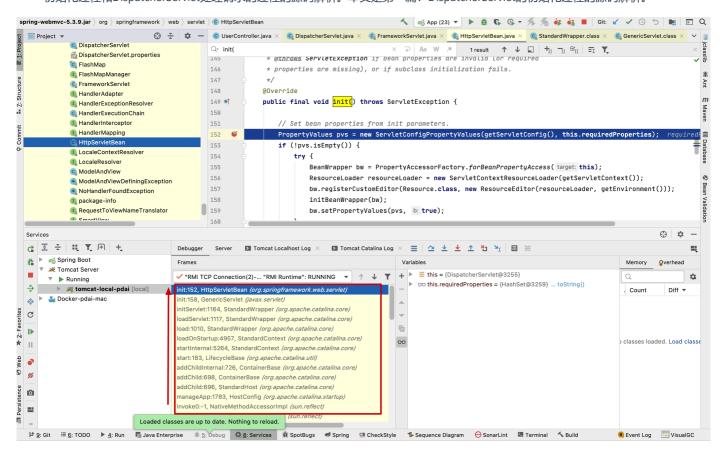
- Spring进阶 Spring AOP实现原理详解之JDK代理实现
  - 。 上文我们学习了SpringAOP Cglib动态代理的实现,本文主要是SpringAOP JDK动态代理的案例和实现部分。

```
420-springboot-demo-schedule-executor-servic .i* You are editing a file which is ignored
     420-springboot-demo-schedule-timer
     421-springboot-demo-schedule-springtasks
     422-springboot-demo-schedule-quartz
     423-springboot-demo-schedule-quartz-cluster
                                                             public final class $Proxy0 extends Proxy implements IUserService {
     427-springboot-demo-schedule-netty-hashedwl
     501-springboot-demo-opcua-milo-server
                                                                  private static Method m3;
     502-springboot-demo-opcua-milo-client
     511-springboot-demo-sms-ali
                                                                  private static Method m2;
                                                                  private static Method m0;
     = 611-springboot-demo-ratelimit-guava-rateli
     901-sprinboot-demo-starter-self-defined
                                                                  private static Method m4;
 ∨ ■ com

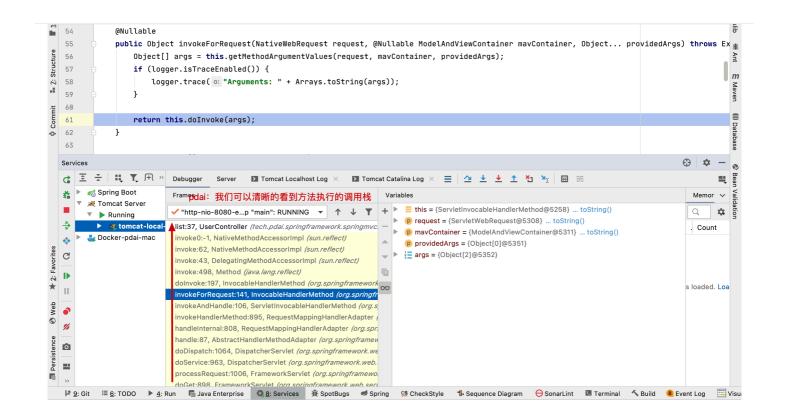
✓ Image: sun

                                                                  public $Proxv0(InvocationHandler var1) throws {
       ∨ □ prox
                                                                       super(var1);
            🔾 $Proxy0
     x001-springboot-test-oom
                                                   26 0
                                                                  public final boolean equals(Object var1) throws {
      👢 .gitignore
                                                                       try {
     ■ LICENSE
                                                                            return (Boolean)super.h.invoke( proxy: this, m1, new Object[]{var1});
     aplan.md
                                                                       } catch (RuntimeException | Error var3) {
     m pom.xml
                                                                            throw var3;
     README.md
                                                                       } catch (Throwable var4) {
> III External Libraries
  Scratches and Consoles
                                                                            throw new UndeclaredThrowableException(var4);
                                 ☼ − 34
\downarrow^a_{\mathbf{X}} \ \downarrow^a_{\mathbf{Z}} \ | \ \mathfrak{C}^{\bullet} \ | \ \mathfrak{P} \ | \ \mathfrak{F} \ | \ \mathbf{A} \ | \ \mathbf{Y} \ | \ \mathbf{O} \ | \ \mathbf{A} \ | \ \mathbf{T} \ | \ \mathbf{T}
                                                                  public final List findUserList() throws {
  © □ $Proxy0
```

- Spring进阶 SpringMVC实现原理之DispatcherServlet初始化的过程
  - 。 前文我们有了IOC的源码基础以及SpringMVC的基础,我们便可以进一步深入理解SpringMVC主要实现原理,包含DispatcherServlet的 初始化过程和DispatcherServlet处理请求的过程的源码解析。本文是第一篇: DispatcherServlet的初始化过程的源码解析。



- Spring进阶 SpringMVC实现原理之DispatcherServlet处理请求的过程
  - 。 前文我们有了IOC的源码基础以及SpringMVC的基础,我们便可以进一步深入理解SpringMVC主要实现原理,包含DispatcherServlet的 初始化过程和DispatcherServlet处理请求的过程的源码解析。本文是第二篇: DispatcherServlet处理请求的过程的源码解析。



## ♡ 我要纠错

Spring基础 - Spring和Spring框架组成 →

苏ICP备19053722号 | pdai | copyright © 2017-present