Spring笔记

# IoC

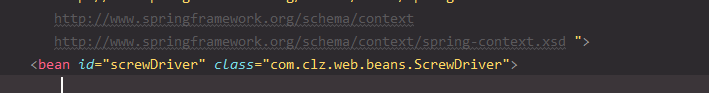
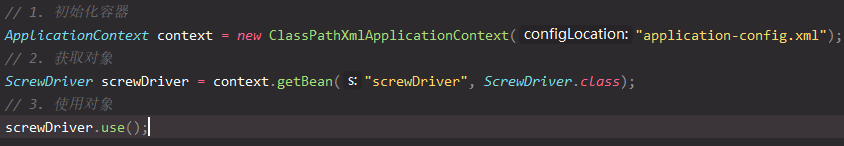
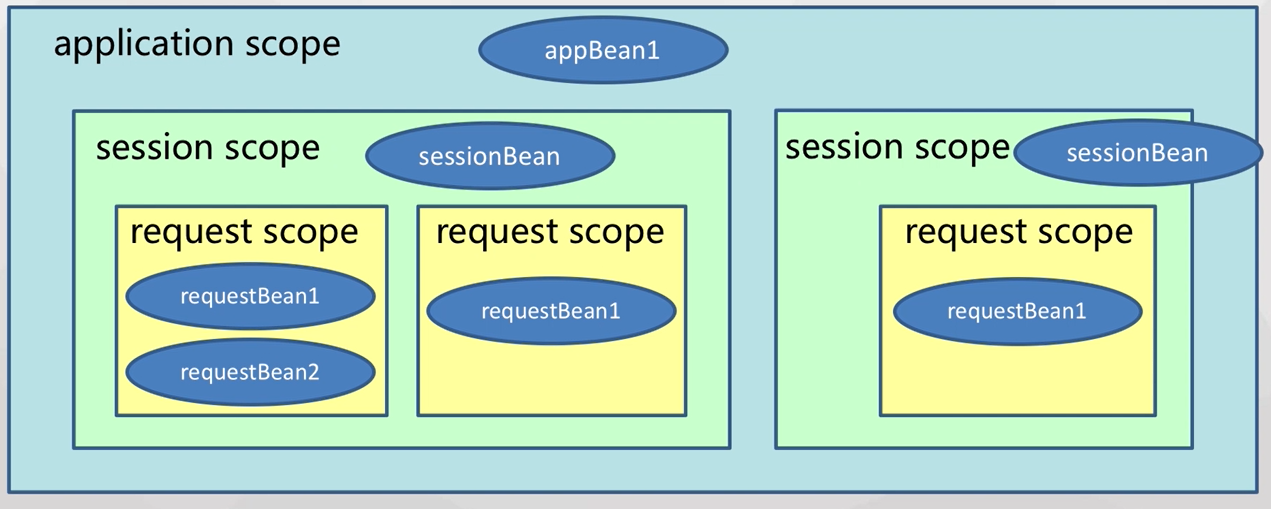
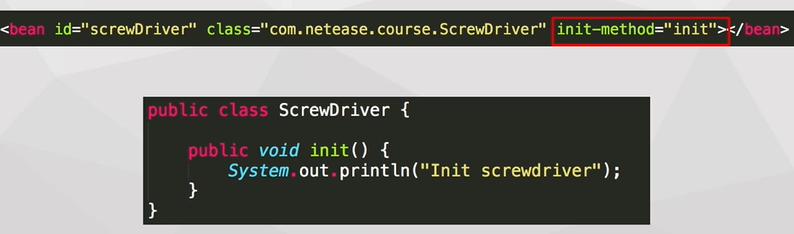
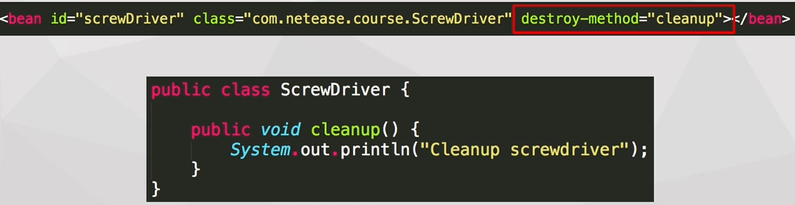
1. Inversion of Control（IoC，控制反转）
   1. 所谓控制反转就是将bean的创建的任务交给容器，而不是手动创建
   2. 即对bean的控制由用户反转给容器
2. Dependency Injection（DI，依赖注入）

# AOP

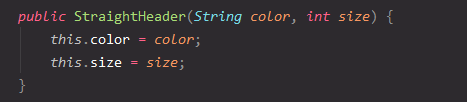
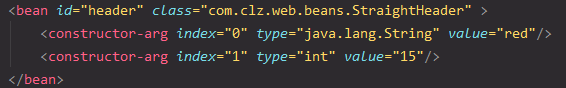
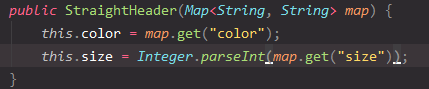
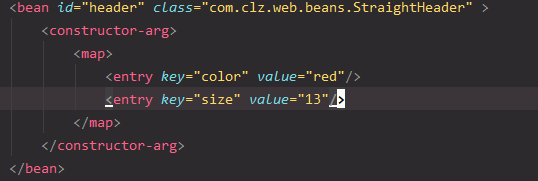
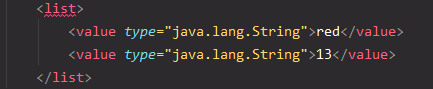
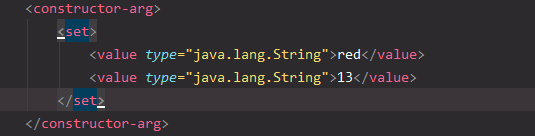
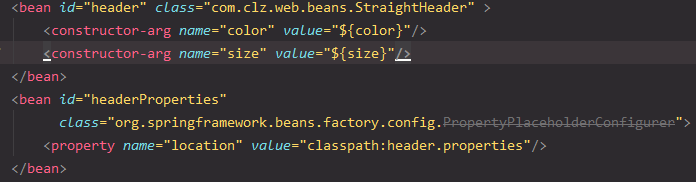
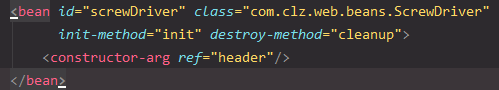
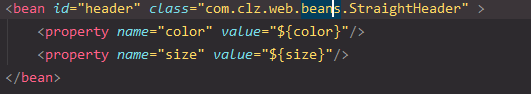
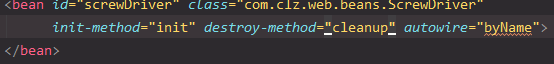
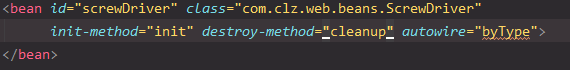
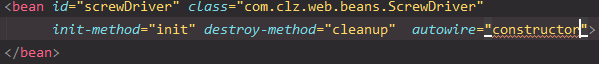
1. Aspect-Oriented Programming（AOP，面向切面编程）
2. 业务逻辑和非业务逻辑分离
3. Aspect
   1. 日志
   2. 安全

# IoC容器

## 概述

1. IoC容器 = ApplicationContext
   1. Org.springframework.context
   2. Spring-context
2. 容器初始化
   1. 
   2. 或者
3. Bean定义
4. Bean的使用
5. Bean的作用域
   1. 有效范围？创建？销毁？
   2. Singleton
      1. 默认就是
      2. 或者
   3. Prototype
      1. 每次引用都创建一个实例
   4. Request scope
   5. Session scope
   6. Application scope
6. Bean生命周期回调
   1. 创建
      1. 申请资源
   2. 销毁
      1. 释放资源

## 依赖注入

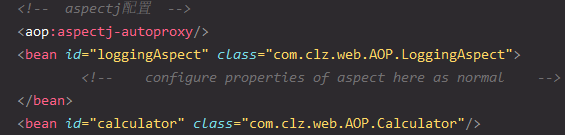
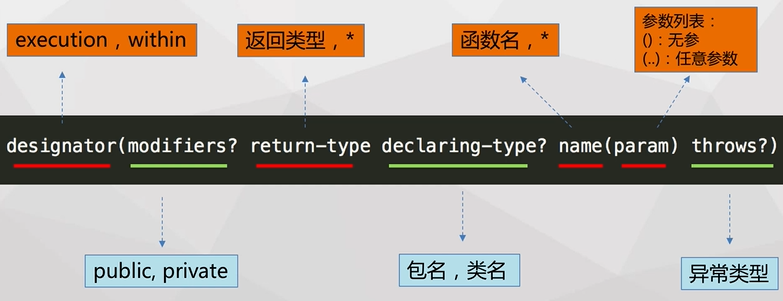
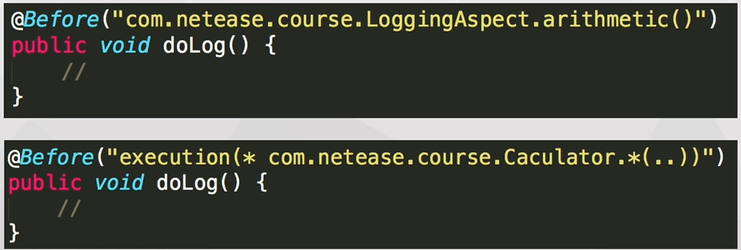
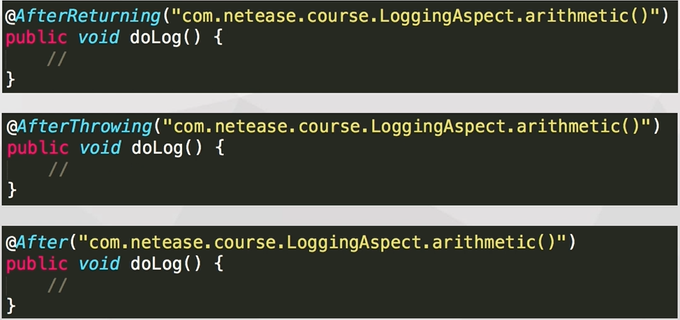
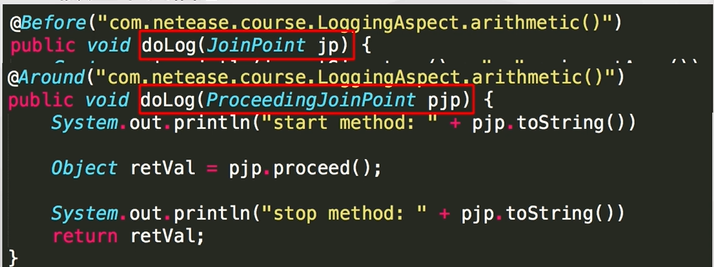
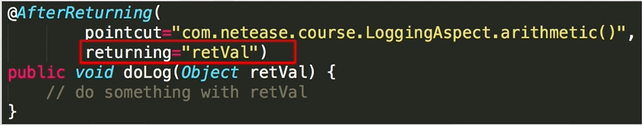
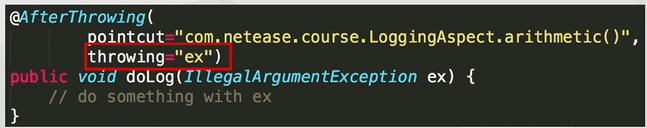
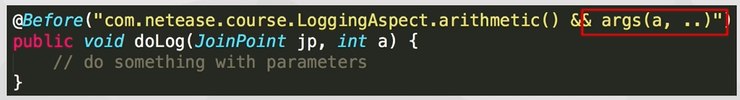
1. 定义接口
2. 实现接口
3. 以接口为基础注入
4. 依赖注入方式
   1. 基于构造函数——强依赖
      1. 基本类型（int， string）
         1. 
         2. 
      2. 集合
         1. Map
            1. 
            2. 
         2. List
            1. 
         3. Set
            1. 
      3. 配置文件
      4. Bean
   2. 基于setter方法——可选依赖
      1. 
      2. 必须要有默认构造函数
      3. 其他类型类似
   3. 自动装配——autowire——？？？
      1. byName：根据bean名称
         1. 必须要有setter方法
      2. byType：根据bean类型
      3. constructor：构造函数，根据类型
   4. 基于注解——annotation
      1. XML 繁琐，代码独立
      2. Annotation 简洁，代码耦合
      3. @Component：定义bean
      4. @Value：properties注入
      5. @Autowired & @ Resourse：自动装配依赖
      6. @PostConstruct & @PreDestory：生命周期回调

# AOP技术

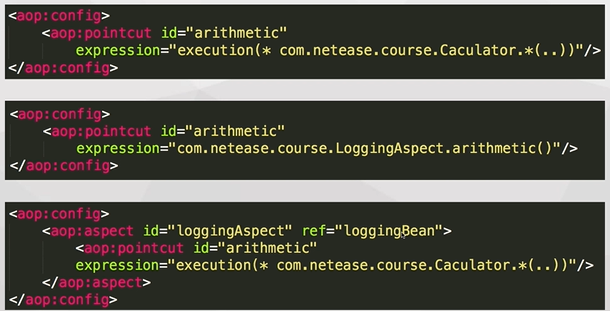
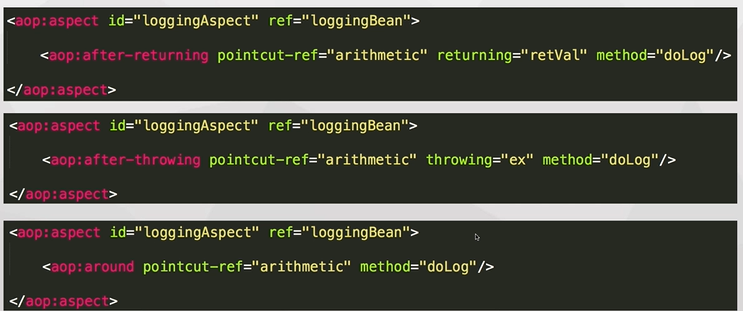
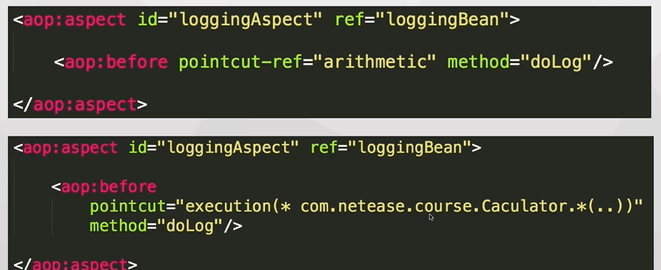
1. 日志
   1. 缺点
      1. 代码重复
      2. 耦合业务逻辑与非业务逻辑
      3. 耦合业务逻辑和非业务逻辑
2. AOP
   1. 优点
      1. 代码重用
      2. 解耦业务逻辑和非业务逻辑
3. AOP术语
   1. Aspect：日志，安全等功能
   2. Join point：函数执行或者属性访问
   3. Advice：在某个函数执行点上要执行的切面功能
   4. Pointcut：匹配横切目标函数的表达式
4. Advice类型
   1. Before：函数执行之前
   2. After returning：函数正常返回之后
   3. After throwing：函数抛出异常之后
   4. After finally：函数返回之后
   5. Around： 函数执行前后
5. SpringAoP
   1. 非完整AoP实现
   2. 整合AoP和IoC
   3. AspectJ：更完整的AoP实现

## 两种方式：

### @AspectJ annotation-based AOP

1. Aspectjwaever.jar
2. Aspectjrt.jar——runtime
3. Application-config.xml 配置
4. 定义Aspect
   1. 
   2. PointCut表达式
      1. 相当与给pointcut表达式定义一个别名叫arithmetic()
   3. 定义advice
      1. 
      2. 
      3. 函数上下文信息？
         1. 
      4. 返回值？
         1. 
      5. Advice参数？
         1. 异常
      6. 目标函数参数？
         1. 

### XML schema-based AOP

1. 定义Aspect
2. 定义pointcut
3. 定义advice

### 比较

1. Schema
   1. 配置集中
2. @AspectJ
   * 1. 配置分散
     2. 兼容AspectJ