

# 第二十章 运维架构升级

---

## Nacos

---

### 历史

Nacos 阿里内部

- 配置中心 - Diamond
- 注册中心 - Config Server

API 易用性 - 设计水平三流

稳定性 - 一流和二流之间

### 基本功能

Nacos 对标中间件 - Consul

- 注册中心
  - Eureka 仅仅注册中心 (Netflix)
  - Consul
  - Zookeeper (不适合做注册中心)
- 配置中心

- Zookeeper (勉强配置中心)
- Consul
- Apollo

<https://nacos.io/zh-cn/>

# Nacos Spring

---

## 介绍

将 Nacos 与 Spring 编程模型整合

## 核心特性

一手微信study322 爱学习都加我

- [Annotation-Driven](#) (注解驱动)
- [Dependency Injection](#) (依赖注入)
- [Externalized Configuration](#) (外部化配置)
- [Event-Driven](#) (事件驱动)

## 工程结构

**nacos-spring-context**

# nacos-spring-samples

## 注解驱动

### @EnableNacos

### @EnableNacosDiscovery

### @EnableNacosConfig

一手微信study322 爱学习都加我

## 依赖注入

### Spring 依赖注入设计技巧

#### 获取上下文

- 注解元信息
  - 自定义注解 - XXX
    - 属性方法
      - String a()
      - int b() default 0;
      - Class c() default void.class

- Spring AnnotationAttributes( LinkedHashMap )
  - "a" -> "\${注解实际标注的值}"
  - "b" -> "\${注解实际标注的值:0}"
  - "c" -> "\${注解实际标注的值:void.class}"

AnnotationAttributes 的来源:

- ImportBeanDefinitionRegistrar 中的方法参数-  
AnnotationMetadata#getAnnotationAttributes(Annotation) 方法
- org.springframework.core.annotation.AnnotationUtils#getAnnotationAttributes(java.lang.reflect.AnnotatedElement, java.lang.annotation.Annotation)
- 被注入类型
  - @Autowired UserService -> UserService.class
- 被注入对象 (名称) 信study322 爱学习都加我
  - 字段 -> 字段名 (依赖查找中的名称)
  - 方法 -> 方法参数 (依赖查找中的名称)
  - 构造器 -> 构造器参数 (依赖查找中的名称)
- Aware 接口注入对象
  - BeanClassLoaderAware -> ClassLoader
  - BeanNameAware -> 当前处理 Bean 名称
  - ApplicationContextAware -> 当前  
ApplicationContext
  - EnvironmentAware -> 当前 Environment

## 定位注入对象

- org.springframework.beans.factory.annotation.InjectionMetadata InjectedElement
- 成员 (Member)
  - 字段 (Field)
  - 方法 (Method)
- 属性描述器 (PropertyDescriptor)

## 管理生命周期

- Bean 生命周期
  - InitializingBean
  - DisposableBean
  - SmartInitializingSingleton
- IoC 生命周期
  - BeanPostProcessor
  - InstantiationAwareBeanPostProcessorAdapter
  - MergedBeanDefinitionPostProcessor
  - ...

## 获取被注入对象

- IoC 依赖查找
  - 手段
    - 名称
    - 类型

- 名称 + 类型
- 来源
  - BeanDefinition 创建 Bean (配置 或 API 注册)
  - 外部注册 Singleton Bean (API 注册) -  
SingletonBeanRegistry#registerSingleton
  - 可解析依赖 (API 注册) -  
ConfigurableListableBeanFactory#registerResolvableDependency
- 自定义实现

## 事件驱动

一手微信study322 爱学习都加我

## 理解 Spring 事件架构

### 核心接口

- 事件发布
  - org.springframework.context.ApplicationEventPublisher (应用)
  - org.springframework.context.event.ApplicationEventMulticaster (内核)
- 事件对象
  - org.springframework.context.ApplicationEvent
- 事件监听器
  - org.springframework.context.ApplicationListener
- 事件发布者注入

- `org.springframework.context.ApplicationEventPublisherAware`

## 通用 Listener 设计技巧

### 设计模式

观察模式 (Observer) 的扩展，一个事件会触发一个或多个 Listener 执行

### 事件发布

- 静态代理
- 动态代理
  - JDK 动态代理 (接口)
  - 字节码提升框架 (类)

### 执行模型

- 同步
- 异步

### 方法签名

- `public void + 动作描述 (方法名) + 事件 (EventObject 实例) 作为参数 + (可选地) 增加异常 (Exception)`
  - `public void configReceived(ConfigReceivedEvent event);`
- `public void + 动作描述 (方法名) + 事件源作为方法 + + (可选地) 增加异常 (Exception)`

- `public void configReceived(String content);`

## 超时设定

避免因为某个 Listener 的执行堵塞后续 Listener 执行

参考：

- `com.alibaba.nacos.spring.context.annotation.config.NacosConfigListenerMethodProcessor`
- `com.alibaba.nacos.spring.context.event.AnnotationListenerMethodProcessor`

## 候选监听方法

- 通过 Listener 方法候选规则（约束）：
  - `public` 一手微信study322 爱学习都加我
  - 非 `static`
  - 非 `native`
  - 返回类型：`void`
- 处理 Listener 方法
  - Spring 场景
    - Bean 信息
    - 方法信息
    - 依赖服务 (`ConfigService`)



# 元数据驱动

## 设计原则

- 全局元数据
  - 公用
- 局部元数据
  - 私有

## @Value 和 @NacosValue 的区别

@Value 获取的配置来源 Environment PropertySources

@NacosValue 配置来源 Nacos Server

@NacosValue 能取到的配置 (Nacos Server) , @Value 不一定能取到 (PropertySources)

@Value 能取到的配置 (PropertySources) , @NacosValue 不一定能取到 (Nacos Server)

如何将 Nacos Server 与 PropertySources 配置打通呢?

答: 通过 @NacosPropertySource, 类似于 @PropertySource

# 依赖工程

## Alibaba Spring Context Support

代码实现依赖注入 -

com.alibaba.spring.beans.factory.annotation.AbstractAnnotationBeanPostProcessor

## 其他

---

## 消息传输设计

微信study322 爱学习都加我

多设计元信息

- Body (Payload)
  - 配置内容
- Header (Metadata)
  - 创建时间
  - 最后修改时间
  - ETag

# 设计水平

一流制定规范

二流制定 API 或者框架

三流基本运用或者熟练运用

# 兼容性测试

## 技巧

利用 Profile 等类似的技术来测试不同依赖版本

一手微信study322 爱学习都加我

# 开发人员专业度

- 技术视野
- 实施场景
  - 在职公司运用
  - 分享其他公司运用

# 作业

---

完善

@org.geektimes.projects.user.mybatis.annotation.Enable  
MyBatis 实现，尽可能多地注入  
org.mybatis.spring.SqlSessionFactoryBean 中依赖的组件

一手微信study322 爱学习都加我