Dubbo Cloud Native

实践与思考

马昕曦（小马哥）

@mercyblitz

自我介绍

- 马昕曦（小马哥）
- 一线互联网公司技术专家，十余年Java EE 从业经验，Apache Dubbo 维护者、架构师以及微服务布道师。目前主要负责阿里巴巴集团微服务技术实施、架构衍进、基础设施构建等。重点关注云计算、微服务以及软件架构等领域。通过SUN Java（SCJP、SCWCD、SCBCD）以及Oracle OCA等的认证。

主要议程

- Cloud Native 基础设施
- Cloud Native 架构选型
- Dubbo Cloud Native 准备

Cloud Native 基础设施

- CNCF Cloud Native Definition v1.0

  Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

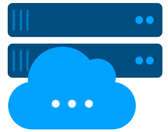
Cloud Native 基础设施

服务发现

Service Discovery

负载均衡

Load Balancing



服务网关

Service Gateway

分布式配置

Distributed Configuration

服务熔断

Circuit Breakers

跟踪监控

Tracing & Monitoring

Cloud Native 基础设施

- 服务发现

Cloud Native 基础设施

- 负载均衡

Cloud Native 基础设施

- 服务网关

Kong

Cloud Native 基础设施

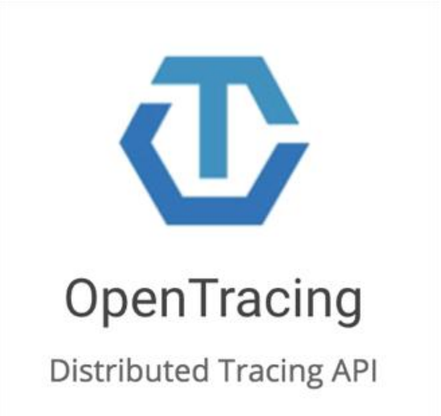
- 分布式配置

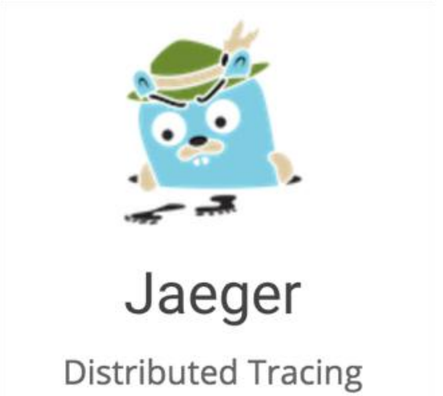
Cloud Native 基础设施

- 服务熔断

Cloud Native 基础设施

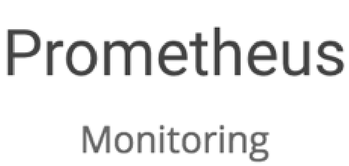
- 链路跟踪





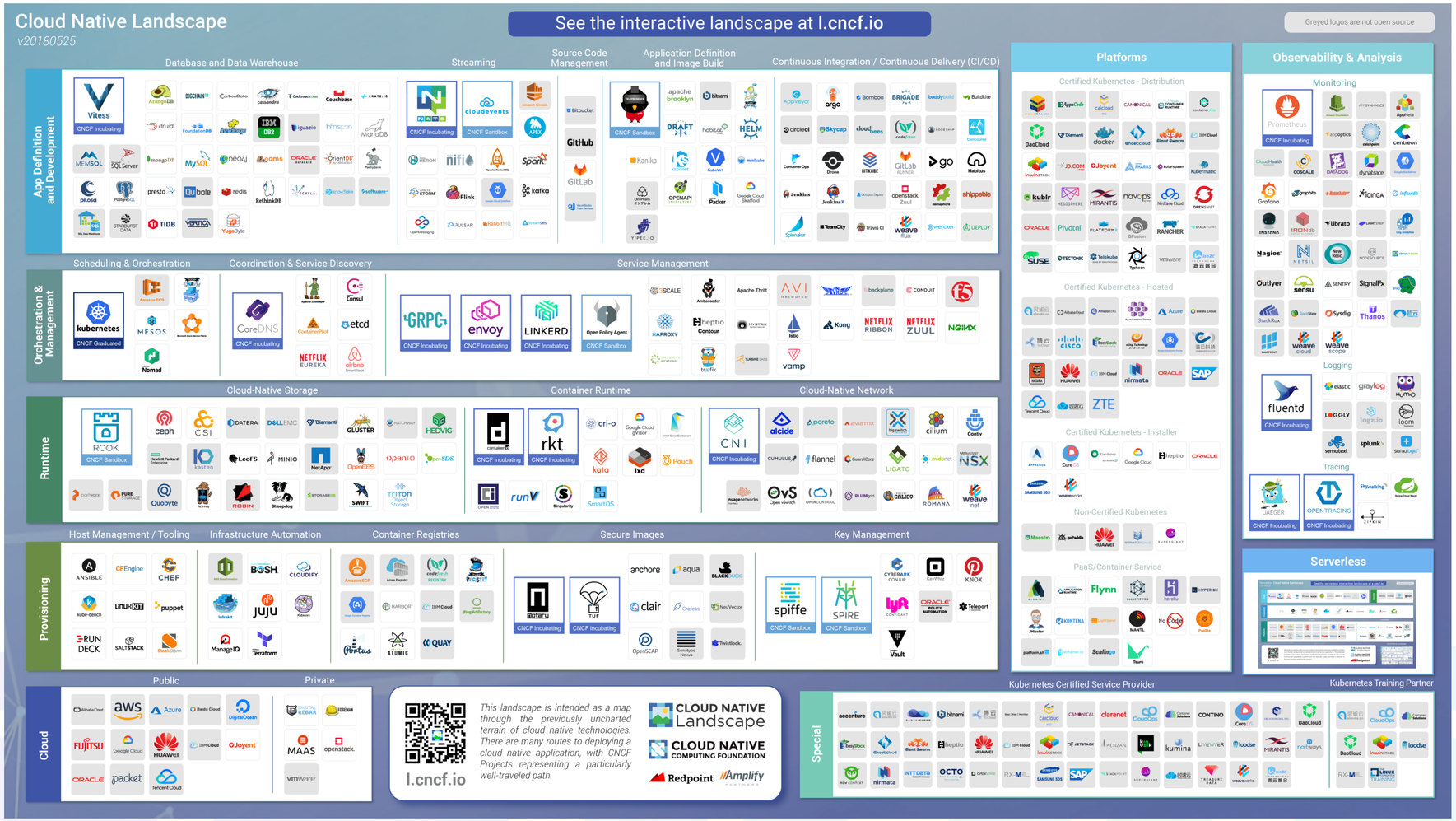
Cloud Native 基础设施

- 服务监控



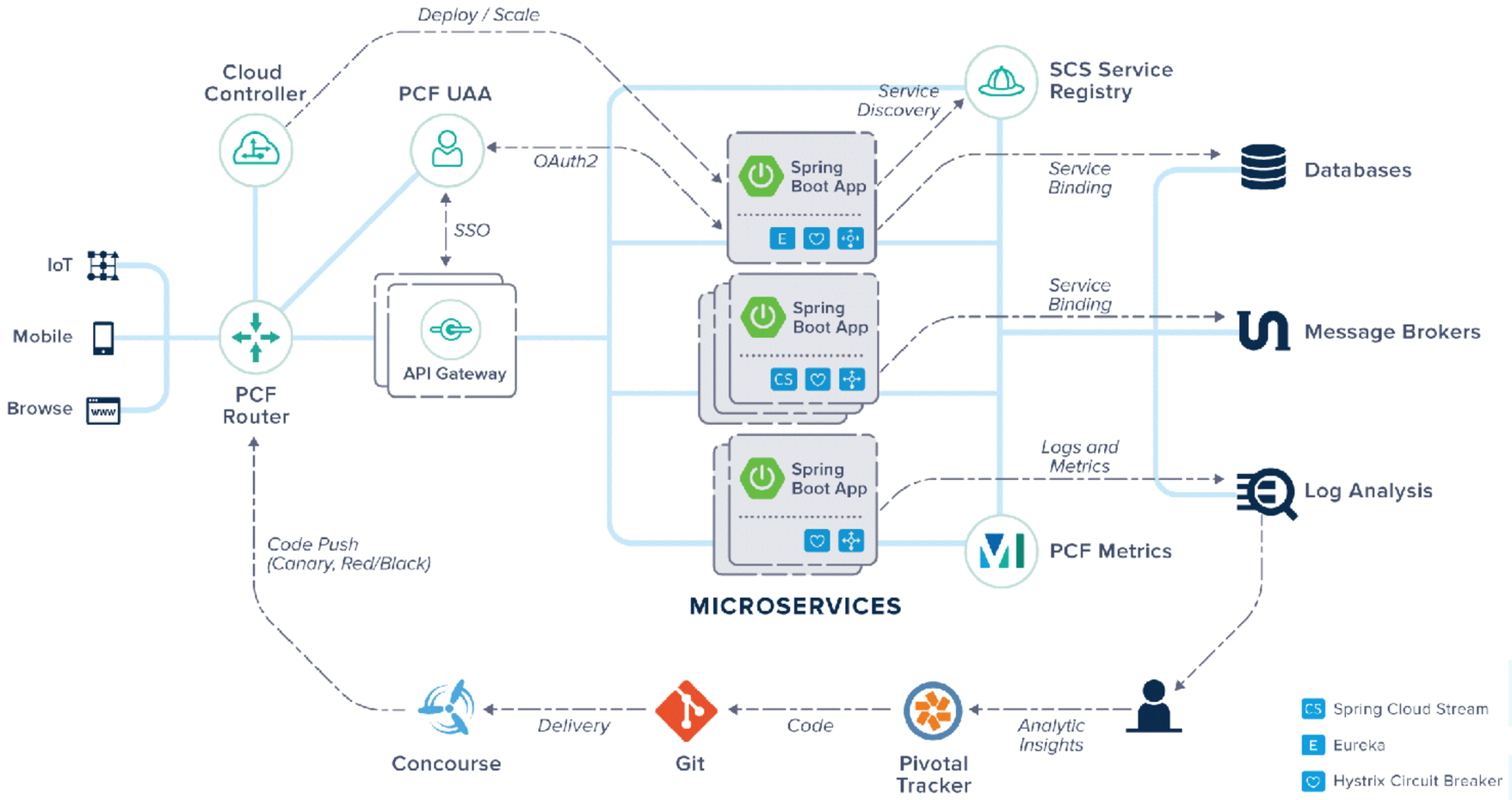
Cloud Native 架构选型

- CNCF 架构体系



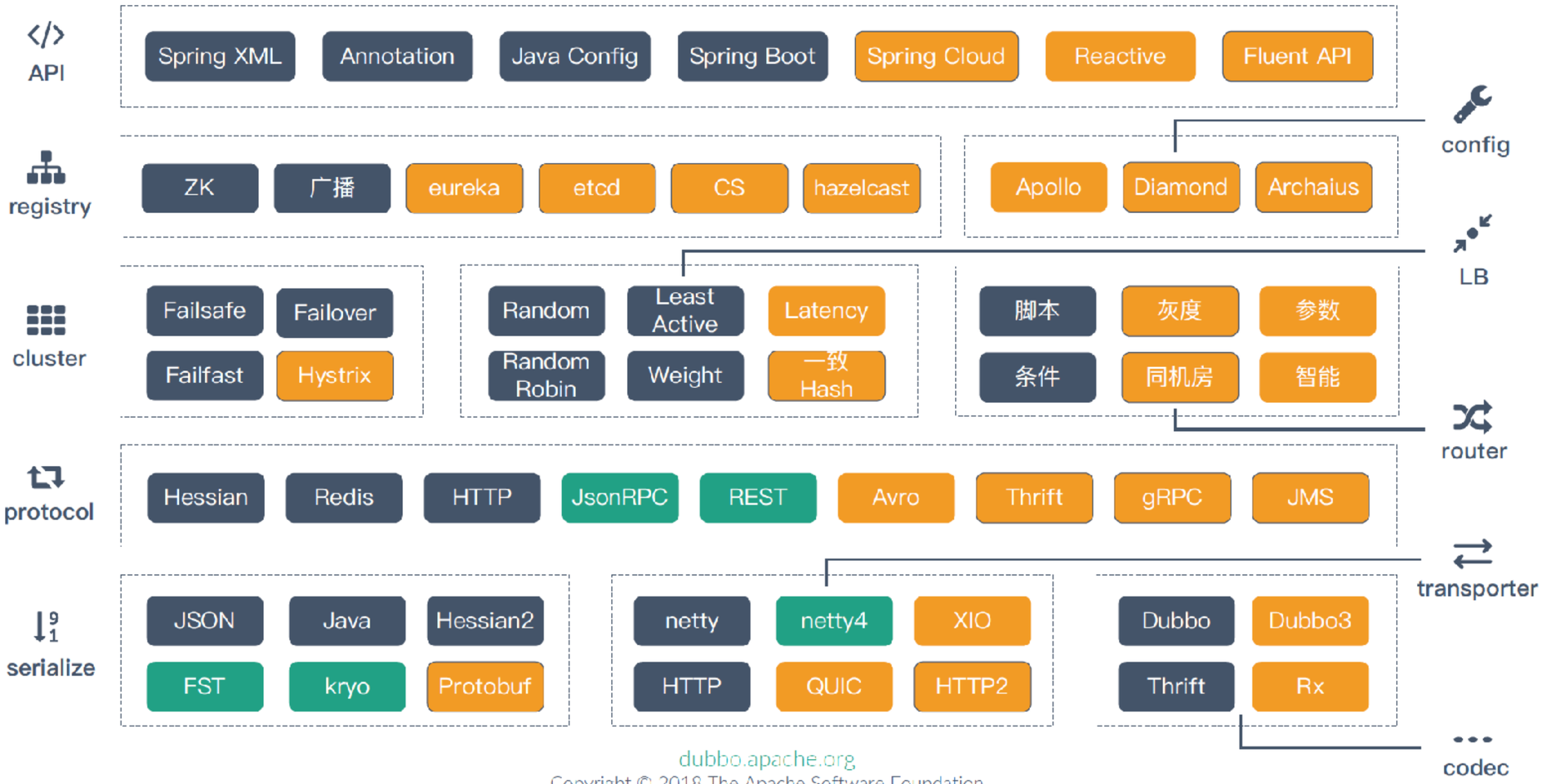
Cloud Native 架构选型

- Spring Cloud 架构体系



Cloud Native 架构选型

- Dubbo 架构体系



dubbo.apache.org
Copyright © 2018 The Apache Software Foundation

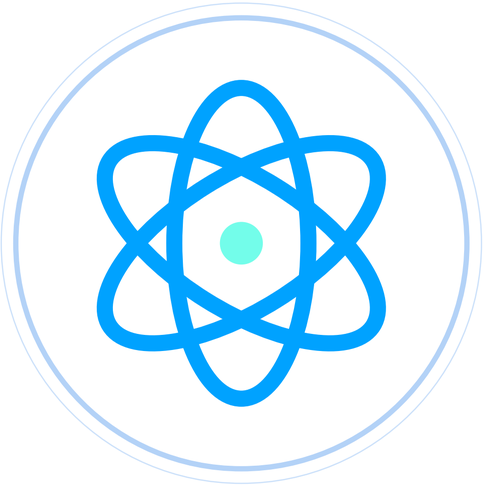
Dubbo Cloud Native 准备

注解驱动

注解驱动提升 Spring 、Spring Boot 编程友好性

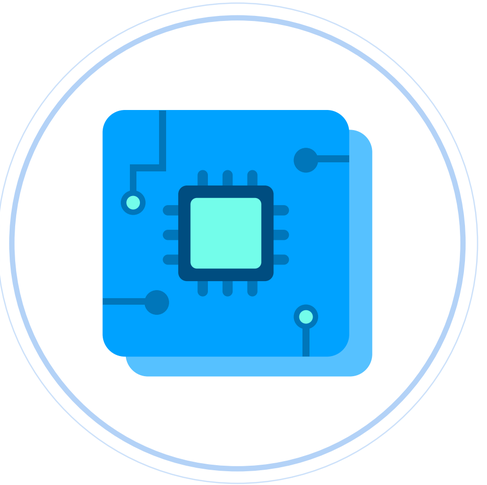
外部化配置

统一外部化配置模型，适配 Spring Cloud Config



Reactive

异步非阻塞响应式编程提升应用伸缩性



REST

REST化 Dubbo 服务，融入微服务架构体系

Dubbo 注解驱动

- 传统 XML 配置驱动

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schem
        http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

    <dubbo:application name="annotation-provider"/>
    <dubbo:registry address="127.0.0.1:4548"/>
    <dubbo:annotation package="com.alibaba.dubbo.config.spring.annotation.provider"/>

</beans>
```

Dubbo 注解驱动

- @DubboComponentScan - 服务端示例

```java
@Configuration
@DubboComponentScan("com.alibaba.dubbo.demo.provider") // 扫描 Dubbo 组件
public class ProviderConfiguration {

    /**
     * 当前应用配置
     */
    @Bean("dubbo-annotation-provider")
    public ApplicationConfig applicationConfig() {
        ApplicationConfig applicationConfig = new ApplicationConfig();
        applicationConfig.setName("dubbo-annotation-provider");
        return applicationConfig;
    }

    /**
     * 当前连接注册中心配置
     */
    @Bean("my-registry")
    public RegistryConfig registryConfig() {
        RegistryConfig registryConfig = new RegistryConfig();
        registryConfig.setAddress("N/A");
        return registryConfig;
    }

    /**
     * 当前连接注册中心配置
     */
    @Bean("dubbo")
    public ProtocolConfig protocolConfig() {
        ProtocolConfig protocolConfig = new ProtocolConfig();
        protocolConfig.setName("dubbo");
        protocolConfig.setPort(12345);
        return protocolConfig;
    }
]
```

```java
@Service(
        version = "1.0.0",
        application = "${dubbo.application.id}",
        protocol = "${dubbo.protocol.id}",
        registry = "${dubbo.registry.id}"
)
public class DefaultDemoService implements DemoService {

    public String sayHello(String name) {
        return "Hello, " + name + " (from Spring Boot)";
    }

}
```

Dubbo 注解驱动

- @DubboComponentScan – 客户端示例

```java
@Configuration
@DubboComponentScan
public class ConsumerConfiguration {

    /**
     * 当前应用配置
     */
    @Bean
    public ApplicationConfig applicationConfig() {
        ApplicationConfig applicationConfig = new ApplicationConfig();
        applicationConfig.setName("dubbo-annotation-consumer");
        return applicationConfig;
    }

    /**
     * 当前连接注册中心配置
     */
    @Bean
    public RegistryConfig registryConfig() {
        RegistryConfig registryConfig = new RegistryConfig();
        registryConfig.setAddress("N/A");
        return registryConfig;
    }

    /**
     * 注册 AnnotationDemoServiceConsumer，@DubboComponentScan 将处理其中 @Reference 字段。
     * 如果 AnnotationDemoServiceConsumer 非 Spring Bean 的话，
     * 即使 @DubboComponentScan 指定 package 也不会进行处理，与 Spring @Autowired 同理
     */
    @Bean
    public AnnotationDemoServiceConsumer annotationDemoServiceConsumer() {
        return new AnnotationDemoServiceConsumer();
    }

}
```

```java
package com.alibaba.dubbo.demo.consumer;

import com.alibaba.dubbo.config.annotation.Reference;
import com.alibaba.dubbo.demo.DemoService;

/**
 * Annotation 驱动 {@link DemoService} 消费方
 *
 * @author <a href="mailto:mercyblitz@gmail.com">Mercy</a>
 */
public class AnnotationDemoServiceConsumer {

    @Reference(url = "dubbo://127.0.0.1:12345")
    private DemoService demoService;

    public String doSayHell(String name) {
        return demoService.sayHello(name);
    }
}
```

Dubbo 外部化配置

- 概念

外部化配置，又名配置外化。当同一的应用归档文件或者镜像部署在不同的环境时，通过外部的配置属性调整，从而影响应用运行时特性行为。

在Dubbo 注解驱动例子中，无论是服务提供方，还是服务消费方，均需要转配相关配置Bean：

```
@Bean
public ApplicationConfig applicationConfig() {
    ApplicationConfig applicationConfig = new ApplicationConfig();
    applicationConfig.setName("dubbo-annotation-consumer");
    return applicationConfig;
}
```

Dubbo 外部化配置

- Dubbo 配置类映射关系

配置类	标签	用途
`ProtocolConfig`	`<dubbo:protocol/>`	协议配置
`ApplicationConfig`	`<dubbo:application/>`	应用配置
`ModuleConfig`	`<dubbo:module/>`	模块配置
`RegistryConfig`	`<dubbo:registry/>`	注册中心配置
`MonitorConfig`	`<dubbo:monitor/>`	监控中心配置
`ProviderConfig`	`<dubbo:provider/>`	提供方配置
`ConsumerConfig`	`<dubbo:consumer/>`	消费方配置
`MethodConfig`	`<dubbo:method/>`	方法配置
`ArgumentConfig`	`<dubbo:argument/>`	参数配置

Dubbo 外部化配置

- Dubbo 配置类 Bean 绑定

Dubbo `*Config` Type	The prefix of property name for Single Bindings
`ProtocolConfig`	`dubbo.protocol`
`ApplicationConfig`	`dubbo.application`
`ModuleConfig`	`dubbo.module`
`RegistryConfig`	`dubbo.registry`
`MonitorConfig`	`dubbo.monitor`
`ProviderConfig`	`dubbo.provider`
`ConsumerConfig`	`dubbo.consumer`

Dubbo `*Config` Type	The prefix of property name for Multiple Bindings
`ProtocolConfig`	`dubbo.protocols`
`ApplicationConfig`	`dubbo.applications`
`ModuleConfig`	`dubbo.modules`
`RegistryConfig`	`dubbo.registries`
`MonitorConfig`	`dubbo.monitors`
`ProviderConfig`	`dubbo.providers`
`ConsumerConfig`	`dubbo.consumers`

Dubbo 外部化配置

- @EnableDubboConfig 配置示例

```
# 单 Dubbo 配置 Bean 绑定
## application
dubbo.application.id = applicationBean
dubbo.application.name = dubbo-demo-application

## module
dubbo.module.id = moduleBean
dubbo.module.name = dubbo-demo-module

## registry
dubbo.registry.address = zookeeper://192.168.99.100:32770

## protocol
dubbo.protocol.name = dubbo
dubbo.protocol.port = 20880

## monitor
dubbo.monitor.address = zookeeper://127.0.0.1:32770

## provider
dubbo.provider.host = 127.0.0.1

## consumer
dubbo.consumer.client = netty
```

```
/**
 * Dubbo 配置 Bean
 *
 * @author <a href="mailto:mercyblitz@gmail.com">Mercy</a>
 */
@EnableDubboConfig
@PropertySource("META-INF/config.properties")
@Configuration
public class DubboConfiguration {

}
```

Dubbo 注解驱动 +外部化配置

- 一站式注解 - @EnableDubbo

```
@Configuration
@EnableDubbo(scanBasePackages = "com.alibaba.dubbo.examples.annotation.action",
multipleConfig = true)
@PropertySource("classpath:/com/alibaba/dubbo/examples/annotation/dubbo-
consumer.properties")
@ComponentScan(value = {"com.alibaba.dubbo.examples.annotation.action"})
public class ConsumerConfiguration {

}
```

Dubbo Spring Boot 支持

- https://github.com/apache/incubator-dubbo-spring-boot-project

Dubbo Spring Boot Project

build passing | codecov 87% | chat on gitter | license Apache-2.0 | maven-central v0.2.0

Apache Dubbo(incubating) Spring Boot Project makes it easy to create Spring Boot application using Dubbo as RPC Framework. What's more, it aslo provides

- auto-configure features (e.g., annotation-driven, auto configuration, externalized configuration).
- production-ready features (e.g., security, health checks, externalized configuration).

> Apache Dubbo(incubating) is a high-performance, java based RPC framework open-sourced by Alibaba. As in many RPC systems, dubbo is based around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a dubbo server to handle client calls. On the client side, the client has a stub that provides the same methods as the server.

Dubbo Spring Boot 支持

- Spring Boot 1.x 和 2.0 并行支持

For now, `dubbo-spring-boot-starter` will separate two versions for Spring Boot 2.x and 1.x once release :

- `0.2.x` is a main stream release version for Spring Boot 2.x
- `0.1.x` is a legacy version for maintaining Spring Boot 1.x

Dependencies

versions	Java	Spring Boot	Dubbo
`0.2.0`	1.8+	`2.0.x`	`2.6.2` +
`0.1.1`	1.7+	`1.5.x`	`2.6.2` +

Dubbo Spring Boot 支持

- 工程模块

There are some modules in Dubbo Spring Boot Project, let's take a look at below overview:

dubbo-spring-boot-parent

The main usage of `dubbo-spring-boot-parent` is providing dependencies management for other modules.

dubbo-spring-boot-autoconfigure

`dubbo-spring-boot-autoconfigure` uses Spring Boot's `@EnableAutoConfiguration` which helps core Dubbo's components to be auto-configured by `DubboAutoConfiguration`. It reduces code, eliminates XML configuration.

dubbo-spring-boot-actuator

`dubbo-spring-boot-actuator` provides production-ready features (e.g., health checks, endpoints, and externalized configuration).

dubbo-spring-boot-starter

`dubbo-spring-boot-starter` is a standard Spring Boot Starter, which contains dubbo-spring-boot-autoconfigure and dubbo-spring-boot-actuator. It will be imported into your application directly.

dubbo-spring-boot-samples

The samples project of Dubbo Spring Boot that includes two parts:

Dubbo REST 支持

- 激活 REST - @EnableRestService （ @since 2.7.x ）

```
@SpringBootApplication
@EnableRestService
public class DubboProviderDemo {

    public static void main(String[] args) {
        new SpringApplicationBuilder(DubboProviderDemo.class)
                .run(args);
    }

}
```

Dubbo REST 支持

- Spring Web MVC 兼容（@since 2.7.x）

```java
/**
 * Demo Service interface
 */
public interface DemoService {

    // Spring MVC 注解支持
    @RequestMapping("/say-hello")
    String sayHello(String name);

}
```

Thank you !