Apache Dubbo 2.7 新特性

- 小马哥（mercyblitz）

议程

- 全面拥抱 Apache 社区
- 编程模型的新特性
- Cloud Native 的新特性

小马哥

浙江 杭州

扫一扫上面的二维码图案，加我微信

小马哥（mercyblitz）

一线互联网技术专家

Apache Dubbo PPMC 成员

Alibaba Nacos 架构成员

微服务架构师

擅长领域：

微服务架构设计与实践

API 设计/实现

Java 生态体系

全面拥抱 Apache 社区

- Maven GAV 调整
- Java package 调整
- ASF 信息/ LICENSE 更新

Maven GAV 调整

```
...
<parent>
    <groupId>org.apache</groupId>
    <artifactId>apache</artifactId>
    <version>19</version>
</parent>

<groupId>org.apache.dubbo</groupId>
<artifactId>dubbo-parent</artifactId>
<version>2.7.0-SNAPSHOT</version>
...
```

Java package 调整

```java
package org.apache.dubbo.demo.provider;

import org.apache.dubbo.demo.DemoService;
import org.apache.dubbo.rpc.RpcContext;

import java.text.SimpleDateFormat;
import java.util.Date;

public class DemoServiceImpl implements DemoService {

    @Override
    public String sayHello(String name) {
        System.out.println("[" + new SimpleDateFormat("HH:mm:ss").format(new Date()) + "] Hello " + name +
", request from consumer: " + RpcContext.getContext().getRemoteAddress());
        return "Hello " + name + ", response from provider: " + RpcContext.getContext().getLocalAddress();
    }

}
```

编程模型的新特性

- 基础设施
- 全面异步编程
- 全面注解驱动编程

基础设施

- Java 8+（必须）
- Netty 4（默认）
- Spring 4+（可选）

全面异步编程

- 服务端和消费端全面异步编程
- CompletableFuture 服务返回
- 消费端 CompletableFuture 异步兼容

服务端和消费端全面异步编程

```java
public class AsyncServiceImpl implements AsyncService {

    public CompletableFuture<String> sayHello(String name) {
        RpcContext savedContext = RpcContext.getContext();
        RpcContext savedServerContext = RpcContext.getServerContext();
        return CompletableFuture.supplyAsync(() -> {
            System.out.println(savedContext.getAttachment("consumer-key1"));
            savedServerContext.setAttachment("server-key1", "server-value1");
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            return "async response from provider.";
        });
    }

}
```

```java
RpcContext.getContext().setAttachment("consumer-key1", "consumer-value1");
CompletableFuture<String> future = asyncService.sayHello("async call request");
RpcContext savedServerContext = RpcContext.getServerContext();
future.whenComplete((v, t) -> {
    System.out.println(savedServerContext.getAttachment("server-key1"));
    if (t != null) {
        t.printStackTrace();
    } else {
        System.out.println("Response: " + v);
    }
});
System.out.println("Executed before response return.");
```

消费端 CompletableFuture 异步兼容

```java
public class AsyncServiceImpl implements AsyncService {

    public String sayHello(String name) {
        System.out.println("async provider received: " + name);
        return "hello, " + name;
    }

}
```

```java
final AsyncService asyncService = (AsyncService) context.getBean("asyncService");

CompletableFuture<String> f = RpcContext.getContext().asyncCall(() -> asyncService.sayHello("async call request"));

System.out.println("async call ret :" + f.get());


RpcContext.getContext().asyncCall(() -> {
        asyncService.sayHello("oneway call request1");
        asyncService.sayHello("oneway call request2");
});
```

全面注解驱动编程

```java
@Controller("annotationAction")
public class AnnotationAction {

    @Reference(version = "${dubbo.version}", methods = {
            @Method(name = "sayName",
                    retry = true,
                    executes = 3,
                    arguments = {
                            @Argument(index = 0)
                    })
    })
    private DemoService demoService;

    public String doSayName(String name) {
        return demoService.sayName(name);
    }

}
```

```java
@Service(version = "${dubbo.version}")
public class AnnotationServiceImpl implements DemoService {

    @Method(reliable = true)
    public String sayName(String name) {
        return "annotation:" + name;
    }

    public Box getBox() {
        return null;
    }

}
```

Cloud Native 的新特性

- 服务治理
- 分布式元信息管理
- Spring Cloud 整合

服务治理

- 注册中心精简
- Metrics
- 服务短路（熔断）

注册中心精简

```
dubbo://10.122.111.22:20880/com.xxx.compose.ic.service.vas.ValueAddServiceCompose?
anyhost=true&application=goods-
compose&default.actives=400&default.delay=-1&default.dispatcher=all&default.group=online&default.loadbalan
ce=leastactive&default.service.filter=-
monitor&default.threads=400&default.timeout=2000&default.version=1.0&delay=1&dubbo=2.6.2&environment=produ
ct&interface=com.xxx.compose.ic.service.vas.ValueAddServiceCompose&logger=slf4j&methods=queryValueAddServi
ceEditVOs,auditPassByValueAddServiceIdList,getValueAddServiceVOByServiceIds,getValueAddServiceInfoByGoodsI
dList,getRuleMatchVOByGoodsIdList,getValueAddServiceEditVOById,getValueAddServiceEditVOByIds,queryValueAdd
ServiceVOs,disableValueAddService,auditRejectByValueAddServiceIdList,saveValueAddServiceEditVO,enableValue
AddService&organization=someorg&owner=somebody&pid=54812&revision=1.18.0628.3&side=provider&timestamp=1530
671441040
```

```
dubbo://10.122.111.22:20880/com.xxx.compose.ic.service.vas.ValueAddServiceCompose?
timeout=1000&group=online&version=1.0&weight=100&timestamp=1530671441040
```

Metrics

incubator-dubbo / dubbo-metrics / dubbo-metrics-api / src / main / java / org / apache / dubbo / metrics /

ralf0131 and chickenlj Merge pull request #1966, introduces dubbo metrics API module. ...

..

BucketCounter.java	Merge pull request #1966, introduces dubbo metrics API module.
Compass.java	Merge pull request #1966, introduces dubbo metrics API module.
Counter.java	Merge pull request #1966, introduces dubbo metrics API module.
Counting.java	Merge pull request #1966, introduces dubbo metrics API module.
Gauge.java	Merge pull request #1966, introduces dubbo metrics API module.
IMetricManager.java	Merge pull request #1966, introduces dubbo metrics API module.
Metric.java	Merge pull request #1966, introduces dubbo metrics API module.
MetricFilter.java	Merge pull request #1966, introduces dubbo metrics API module.
MetricLevel.java	Merge pull request #1966, introduces dubbo metrics API module.
MetricManager.java	Merge pull request #1966, introduces dubbo metrics API module.
MetricName.java	Merge pull request #1966, introduces dubbo metrics API module.
MetricRegistry.java	Merge pull request #1966, introduces dubbo metrics API module.
MetricSet.java	Merge pull request #1966, introduces dubbo metrics API module.
NOPMetricManager.java	Merge pull request #1966, introduces dubbo metrics API module.

服务短路（熔断）

- 基于 Metrics 的 Cluster 扩展，自动熔断
- 基于 Metrics 的 Loadbalance 扩展，自动识别异常节点，动态踢出或加入

分布式元信息管理

- 注册元信息
- 配置元信息
- Alibaba Nacos 整合

Alibaba Nacos 整合

- 注册中心
- 配置中心
- Dubbo 外部化配置（分布式）

Alibaba Nacos 编程模型（SDK）

```
try {
    // Initialize the configuration service, and the console automatically obtains the following
        String serverAddr = "{serverAddr}";
        String dataId = "{dataId}";
        String group = "{group}";
        Properties properties = new Properties();
        properties.put("serverAddr", serverAddr);
        ConfigService configService = NacosFactory.createConfigService(properties);
    // Actively get the configuration.
        String content = configService.getConfig(dataId, group, 5000);
        System.out.println(content);
} catch (NacosException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Alibaba Nacos 编程模型（SDK）

```
// Initialize the configuration service, and the console automatically obtains the following parc
String serverAddr = "{serverAddr}";
String dataId = "{dataId}";
String group = "{group}";
Properties properties = new Properties();
properties.put("serverAddr", serverAddr);
ConfigService configService = NacosFactory.createConfigService(properties);
String content = configService.getConfig(dataId, group, 5000);
System.out.println(content);
configService.addListener(dataId, group, new Listener() {
        @Override
        public void receiveConfigInfo(String configInfo) {
                System.out.println("receive1:" + configInfo);
        }
        @Override
        public Executor getExecutor() {
                return null;
        }
});
```

Alibaba Nacos 编程模型（Spring+）

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = {
        ListenersConfiguration.class,
        NacosBeanDefinitionRegistrarTest.class,
})
@EnableNacos(globalProperties = @NacosProperties(serverAddr = "11.163.128.36")) // Integration Test
public class NacosBeanDefinitionRegistrarTest {

    @Bean
    public Config config() {
        return new Config();
    }

    @NacosService
    private ConfigService configService;

    @Autowired
    private Config config;

    @Test
    public void testBind() throws Exception {
        configService.publishConfig(DATA_ID, DEFAULT_GROUP, "id=1\n name=mercyblitz\nvalue = 0.95");
         // Wait for async change of data
        Thread.sleep(1000);
        Assert.assertEquals(1, config.getId());
        Assert.assertEquals("mercyblitz", config.getName());
        Assert.assertTrue(0.95 == config.getValue());
    }
}
```

Alibaba Nacos 编程模型（Spring+）

```java
@Configuration
public class ListenersConfiguration {

    @NacosConfigListener(dataId = DATA_ID)
    public void onMessage(String value) {
        System.out.println("onMessage : " + value);
    }

    @NacosConfigListener(dataId = DATA_ID)
    public void onInteger(Integer value) {
        System.out.println("onInteger : " + value);
    }

    @NacosConfigListener(dataId = DATA_ID)
    public void onDouble(Double value) {
        System.out.println("onDouble : " + value);
    }

    @NacosConfigListener(dataId = "user", converter = UserNacosConfigConverter.class)
    public void onUser(User user) {
        System.out.println("onUser : " + user);
    }

}
```

Alibaba Nacos 资源

- https://nacos.io
- https://github.com/alibaba/nacos
- https://github.com/nacos-group

Spring Cloud 整合

- Spring MVC REST 整合
- 网关整合
- Spring RestTemplate 适配（实验性）

Spring MVC REST 整合

```
@Configuration
@EnableRestService
public class DubboProviderConfiguration {
    ...
}
```

```
public interface DemoService {

    @RequestMapping("/say-hello")
    String sayHello(String name);

}
```

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

    public String sayHello(String name) {
        hold();
        return "Say : Hello, " + name;
    }

    @RequestMapping("/say-hello/{name}")
    public String doSayHello(@PathVariable String name) {
        return "Do Say : Hello, " + name;
    }
}
```

网关整合

- 中心化 REST 网关
- REST 服务导出
- 分布式 REST 网关

Spring RestTemplate 适配（实验性）

- REST 服务路由
- 协议切换（dubbo://）

资源推荐

极部落

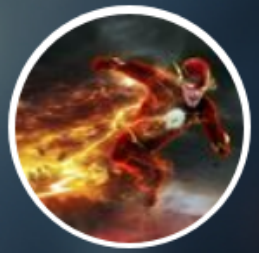
分享，彰显技术的价值！

极部落JAVA开发者大会

Netty 深入系列（视频）

Java读源码之Netty深入剖析

JavaCoder如果没有研究过Netty，那么你对Java语言的使用和理解仅仅停留在表面水平，如果你要进阶，想了解Java服务器的深层高阶知识，Netty绝对是一个必须要过的门槛。本课程带你从一个Socket例子入手，一步步深入探究Netty各个模块的源码，深入剖析Netty的工作流程和源码设计，让你不但“真懂”也要“会用”



闪电侠_

高级研发工程师

Spring Cloud 深入系列（书籍）

Spring Boot 深入系列（书籍）

小马哥新书《Spring Boot 编程思想》（即将上市）

场景分析
掌握技术选型

系统学习
拒绝浅尝辄止

重视规范
了解发展趋势

源码解读
理解设计思想

实战演练
巩固学习成果

极部落

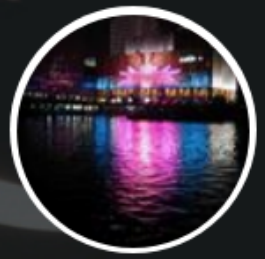
分享，彰显技术的价值！

极部落JAVA开发者大会

Spring Boot 深入系列（视频）

Spring Boot 2.0深度实践之核心技术篇

课程系统性地深度探讨 Spring Boot 核心特性，引导小伙伴对 Java 规范的重视，启发对技术原理性的思考，掌握排查问题的技能，以及学习阅读源码的方法和技巧，全面提升研发能力，进军架构师队伍。



小马哥

mercyblitz

微服务架构师

了解一下？

iTechPlus

极光推送

谢　谢