

12_扩展思路

仅供 编程导航 <<https://www.code-nav.cn/post/1816420035119853569>> 内部成员观看，请勿对外分享！

鱼皮带大家开发的 RPC 框架中，已经具备了大多数 RPC 框架应具有的核心功能，但毕竟是教程项目，没办法面面俱到，跟市面上主流的 RPC 框架比如 Dubbo、gRPC 相比肯定还有很大的差距。

之前每节教程中，几乎都给大家提供了一些基于当节教程的扩展点。这里鱼皮再提供更多 RPC 框架的扩展点，大家可以自行实现。

其实最好的扩展思路，来源于去学习主流成熟的 RPC 框架，比如阅读 Dubbo 的官方文档、了解它们的特性和设计。

H2 ▾ 可选扩展点

1) RPC 请求类中支持携带参数列表，可用于安全校验等。

参考思路：比如服务提供者参数列表、服务消费者参数列表，服务端收到请求后可以根据参数列表中的值，判断如何进一步处理，比如在参数列表中携带 token 可以实现安全校验。

2) 开发服务管理界面。

参考思路：类似 Nacos 注册中心面板，需要一定的前端基础。

3) 项目支持读取 yml / yaml 等更多类型的配置文件，作为全局配置。

参考思路：仿照现有的 ConfigUtils 工具类，支持更多读取配置的方式，甚至可以使用 SPI 机制允许开发者二次扩展配置解析器。

4) 实现拦截器机制，服务调用前和服务调用后可以执行额外的操作。

参考思路：可以参考 Spring MVC 或 Servlet 的 Filter 机制，使用责任链模式实现；可以在服务提供者处理前后、服务消费者调用前后增加拦截器，用于进行日志校验、安全校验等。还可以使用 SPI 机制，支持用户自定义拦截器。

5) 自定义异常。

参考思路：自定义异常类 `RpcException`，根据业务区分错误码（`ErrorCode`），比如消费者异常、注册中心异常、提供者异常等，让报错更清晰明确。

6) 服务支持指定版本号。

参考思路：虽然目前框架已经预留了版本字段，但都是默认值 1.0，还需要从服务提供者、再到消费者代理调用的一条完整路径上去应用版本号。

7) 支持消费方指定某个服务级别的负载均衡器、重试策略、容错机制。

参考思路：目前只能通过全局配置改变对所有服务的负载均衡调用规则。实现的话可能需要修改 `ServiceProxy` 类，让它支持传参，根据消费端的配置来动态创建 `ServiceProxy`。

8) 支持指定服务分组：服务提供者能够选择服务的分组，服务消费者能够使用指定分组的服务。 (Dubbo 支持)

参考思路：目前框架仅仅是预留了服务分组字段，用默认值填充。还需要从服务提供者、再到消费者代理调用的一条完整路径上去应用服务分组。可以进一步实现多环境功能。

9) 服务消费方支持设定超时时间。

参考思路：可以通过修改 TCP 客户端请求相关的代码实现。

10) 处理 Bean 注入问题：目前本地服务注册表存储的是 class，然后通过反射创建实例，但是如果 Bean 包含有参构造函数，或者给属性注入了其他示例，这种方式就行不通了。

参考思路：本地注册时要放入实现类的对象实例，而不是 class 类型。

11) 压力测试 RPC 框架的性能。

参考思路：通过 JMeter 等压力测试工具验证请求、响应的传输耗时和 QPS。

12) 服务注册信息缓存优化，支持区分服务 key，而不是公用同一个缓存对象。

参考思路：目前的缓存实现是最简单的，但如果有多多个服务，可能会冲突，可以使用 Map 等结构维护多个缓存信息。

13) 服务注册信息缓存增加过期时间，定期刷新缓存。

参考思路：使用 Caffeine 构建缓存。

url=https%3A%2F%2Fwww.yuque.com%2Fu37765561%2Fak85bt%2F3799aa1db9c1435a8aa61928