简述dubbo 的 spi 思想

题目标签

学习时长：20分钟

题目难度：中等

知识点标签：dubbo 、spi

题目描述

简述dubbo 的 spi 思想

1. 面试官心理分析

继续深入问呗，前面一些基础性的东西问完了，确定你应该都 ok，了解 dubbo 的一些基本东西，那么问个稍微难一点点的问题，就是 spi，先问问你 spi 是啥？

然后问问你 dubbo 的 spi 是怎么实现的？其实就是看看你对 dubbo 的掌握如何。

2. 面试题剖析

spi，简单来说，就是 service provider interface ，说白了是什么意思呢，比如你有个接口，现在这个接口有 3 个实现类，那么在系统运行的时候对这个接口到底选择哪个实现类呢？这就需要 spi 了，需要根据指定的配置或者是默认的配置，去找到对应的实现类加载进来，然后用这个实现类的实例对象。

举个栗子：

你有一个接口 A。A1/A2/A3 分别是接口A的不同实现。你通过配置 接口 A = 实现 A2 ，那么在系统实际运行的时候，会加载你的配置，用实现 A2 实例化一个对象来提供服务。

spi 机制一般用在哪儿？插件扩展的场景，比如说你开发了一个给别人使用的开源框架，如果你想让别人自己写个插件，插到你的开源框架里面，从而扩展某个功能，这个时候 spi 思想就用上了。

3. Java spi 思想的体现

spi 经典的思想体现，大家平时都在用，比如说 jdbc。

Java 定义了一套 jdbc 的接口，但是 Java 并没有提供 jdbc 的实现类。

但是实际上项目跑的时候，要使用 jdbc 接口的哪些实现类呢？一般来说，我们要根据自己使用的数据库，比如 mysql，你就将 mysql-jdbc-connector.jar 引入进来；

oracle，你就将 oracle-jdbc-connector.jar 引入进来。

在系统跑的时候，碰到你使用 jdbc 的接口，他会在底层使用你引入的那个 jar 中提供的实现类。

4. dubbo 的 spi 思想

dubbo 也用了 spi 思想，不过没有用 jdk 的 spi 机制，是自己实现的一套 spi 机制。

Protocol protocol = ExtensionLoader.getExtensionLoader(Protocol.class).getAdaptiveExtension();

Protocol 接口，在系统运行的时候，，dubbo 会判断一下应该选用这个 Protocol 接口的哪个实现类来实例化对象来使用。

它会去找一个你配置的 Protocol，将你配置的 Protocol 实现类，加载到 jvm 中来，然后实例化对象，就用你的那个 Protocol 实现类就可以了。

上面那行代码就是 dubbo 里大量使用的，就是对很多组件，都是保留一个接口和多个实现，然后在系统运行的时候动态根据配置去找到对应的实现类。如果你没配置，那就走默认的实现好了，没问题。

> @SPI("dubbo")
> public interface Protocol {

```
int getDefaultPort();

@Adaptive
<T> Exporter<T> export(Invoker<T> invoker) throws RpcException;

@Adaptive
<T> Invoker<T> refer(Class<T> type, URL url) throws RpcException;

void destroy();
```

}

在 dubbo 自己的 jar 里，在 /META_INF/dubbo/internal/com.alibaba.dubbo.rpc.Protocol 文件中：

```
dubbo=com.alibaba.dubbo.rpc.protocol.dubbo.DubboProtocol
http=com.alibaba.dubbo.rpc.protocol.http.HttpProtocol
hessian=com.alibaba.dubbo.rpc.protocol.hessian.HessianProtocol
```

所以说，这就看到了 dubbo 的 spi 机制默认是怎么玩儿的了，其实就是 Protocol 接口，@SPI("dubbo") 说的是，通过 SPI 机制来提供实现类，实现类是通过 dubbo 作为默认 key 去配置文件里找到的，配置文件名称与接口全限定名一样的，通过 dubbo 作为 key 可以找到默认的实现类就是 com.alibaba.dubbo.rpc.protocol.dubbo.DubboProtocol 。

如果想要动态替换掉默认的实现类，需要使用 @Adaptive 接口，Protocol 接口中，有两个方法加了 @Adaptive 注解，就是说那俩接口会被代理实现。

啥意思呢?

比如这个 Protocol 接口搞了俩 @Adaptive 注解标注了方法，在运行的时候会针对 Protocol 生成代理类，这个代理类的那俩方法里面会有代理代码，代理代码会在运行的时候动态根据 url 中的 protocol 来获取那个 key，默认是 dubbo，你也可以自己指定，你如果指定了别的 key，那么就会获取别的实现类的实例了。

5. 自定义dubbo 的 spi 扩展

如何自己扩展 dubbo 中的组件

下面来说说怎么来自己扩展 dubbo 中的组件。

自己写个工程，要是那种可以打成 jar 包的，里面的 src/main/resources 目录下，搞一个 META-INF/services，里面放个文件叫： com.alibaba.dubbo.rpc.Protocol，文件里搞一个 my=com.bingo.MyProtocol 。

自己把 jar 弄到 nexus 私服里去。

然后自己搞一个 dubbo provider 工程，在这个工程里面依赖你自己搞的那个 jar，然后在 spring 配置文件里给个配置：

<dubbo:protocol name=”my” port=”20000” /> provider 启动的时候，就会加载到我们 jar 包里的 my=com.bingo.MyProtocol 这行配置里，接着会根据你的配置使用你定义好的 MyProtocol 了，这个就是简单说明一下，你通过上述方式，可以替换掉大量的 dubbo 内部的组件，就是扔个你自己的 jar 包，然后配置一下即可。

dubbo-spi

dubbo 里面提供了大量的类似上面的扩展点，就是说，你如果要扩展一个东西，只要自己写个 jar，让你的 consumer 或者是 provider 工程，依赖你的那个 jar，在你的 jar 里指定目录下配置好接口名称对应的文件，里面通过 key=实现类 。

然后对于对应的组件，类似 dubbo:protocol 用你的那个 key 对应的实现类来实现某个接口，你可以自己去扩展 dubbo 的各种功能，提供你自己的实现。