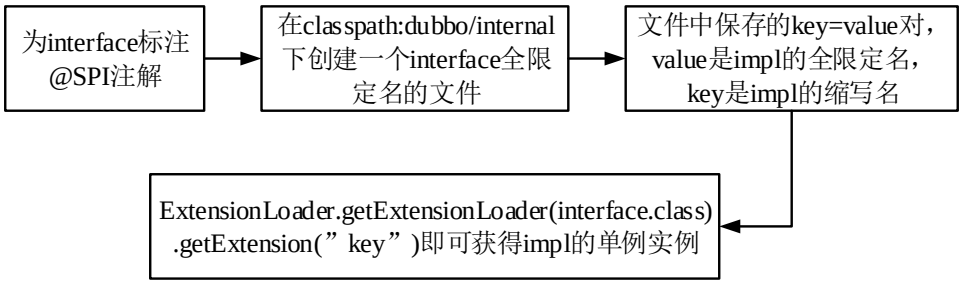
一、dubbo 的 SPI

dubbo 的 SPI 依赖于 ExtensionLoader 类，这个类可以通过加载指定文件夹下所配的实现类将应用相关类。

1.dubbo 的 SPI 扩展

(1)扩展流程



(2)相关注解

dubbo 通过三个注解@SPI、@Adaptive、@Activate 对接口的实现类进行动态扩展。

@SPI(String value):

标注在一个接口上，代表这个接口的实现类可以被扩展，value 是该扩展接口默认的实现类。

@Adaptive(String[] value):

①若标注在一个实现类上(不能标注在接口上)，表明这个类是一个自适应的装饰类，在调用 getAdaptiveExtension()时不会动态生成编译装饰类。

②若标注在接口的方法上，表明该方法需要被动态生成，在调用 getAdaptiveExtension()时会动态生成编译装饰类，没有标记@Adaptive 注解的方法默认抛出异常。

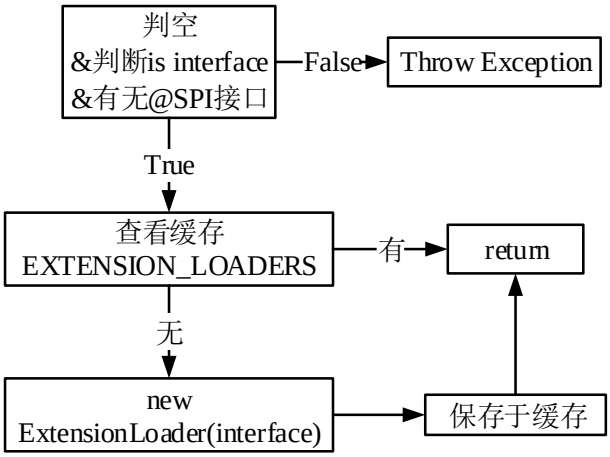
③其中的 value 为动态生成类需要过滤 url 的字段，会将@SPI 中 value 和@Adaptive 中的 value 先后进行遍历 url 名字的。

@Activate

@Adaptive 是获取一个适配类，而@Activate 则可以获取一批适配类，在 impl 上加上@Adaptive 注解在调用 Extension. getActivateExtension(URL url, String key)可以获得 url 满足条件的实现类。

2. ExtensionLoader 相关 api

① getExtensionLoader(Interface) 静态方法



②ExtensionLoader 的<init>

<init>中会保存 interface.class，并且初始化内部的 objectFactory 字段，调用的是 ExtensionLoader.getExtensionLoader(ExtensionFactory.class).getAdaptiveExtension()【1】方法来初始化 objectFactory

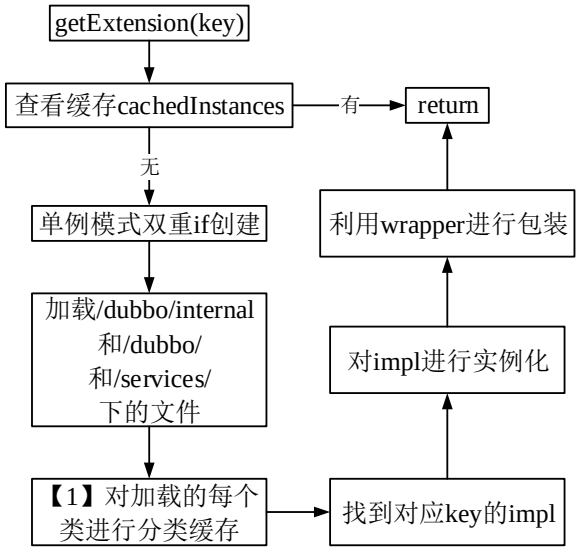
AdaptiveExtensionFactory 被@Adaptive 标记并且满足 SPI 扩展，所以【1】方法获得的其实是 AdaptiveExtensionFactory 实例。

③AdaptiveExtensionFactory 的作用

AdaptiveExtensionFactory 是装饰类，其内部组合了 SpringExtensionFactory 和 SpiExtensionFactory(分别提供了 Spring 的 IOC 和 SPI 的 IOC)

④getExtension(key) 实例方法

通过 getExtensionLoader(Interface)获得 ExtensionLoader 的实例之后，可以调用 getExtension(key)获取 SPI 配置的 key 对应 interface 的实现类



【1】中主要分成四类：

1）被@Adaptive 标记的类

2）含有一个参数为 interface 的构造函数的实现类（wrapper 类）

3）被@Activate 标记的类

4）其他类与 3）中的类

⑤ getAdaptiveExtension() 实例方法