ExtensionFactory根据接口名和配置文件中的key name获取实现类本质还是通过ExtensionLoader获取指定接口的实现类,其实功能和ExtensionLoader类似,但只能根据key和接口名获取实现类实例,相当于ExtensionLoader的接口缩小版本



ExtensionLoader

ExtensionFactory

SpringExtensionFactory

AdaptiveExtensionFactory

List<ExtensionFactory> factories

new ExtensionLoader(InterfaceType)

<-----

使用ExtensionLoader创建
ExtensionFactory,默认是
一)创建ExtensionLoader,一个接口创
建一个ExtensionLoader

赋值给this.objectFactory
用于属性注入

List<T> getActivateExtension(URL url, String[] values, String group)

调用getExtensionClasses, 去指定文件下载 this.type该接口的子类型,加载到内存

二)获取带有Activate注解的子类集合

一)遍历实现类的Activate注解与入参对比满足以下两个条件加入返回list:

2 "url参数key集合中的任何一个元素"以"Activate注解的value数组的任何一个元素"结尾

二)满足以下条件的实现类也会被加入返回数组:

一)满足以下条件的实现实也会被加入返回致组: 入参values含有的配置文件实现类的key对应的实现类

getDefaultExtension

<-----

三)获取默认实现类即接口SPI注解中指定的实现 类对应的key

<-----

getExtension(String name)

四)获取指定name的实现类

createExtension(String name)

五)创建指定name的实现类,并 注入属性,如果存在其他同类型 的wrapper类则在外面层层包裹 wrapper类,并为实现类注入属性 注意1-5创建的实例都是非动态代理的真实实例,第六个创建的实例可能是动态代理出来的实例,该动态代理会根据运行时url参数做决定做不同的事情

getAdaptiveExtension

六)1在类上加上@Adaptive注解的类,则直接返回该类 2接口方法的定义处是否加了@Adaptive注解,如果加则直接抛异常: No adaptive method on extension 3接口方法上如果有@Adaptive注解,注解中的value与url中的参数的key一致,url中的key对应的value就是spi中的name,获取相应的实现类。

4@SPI注解中的value是默认值,如果通过URL获取不到关于取哪个类作为Adaptive类的话,就使用这个默认值,当然如果URL中可以获取到,就用URL中的,会以为接口名作为key(接口名中的大写字母变小写,并在前面加"."),取出url的对应的value

<-----

ProtocolFilterWrapper的buildInvokerChain 方法或根据Activate的group(provider or cponsumer)和value选择filter加入到 provider还是consumer端