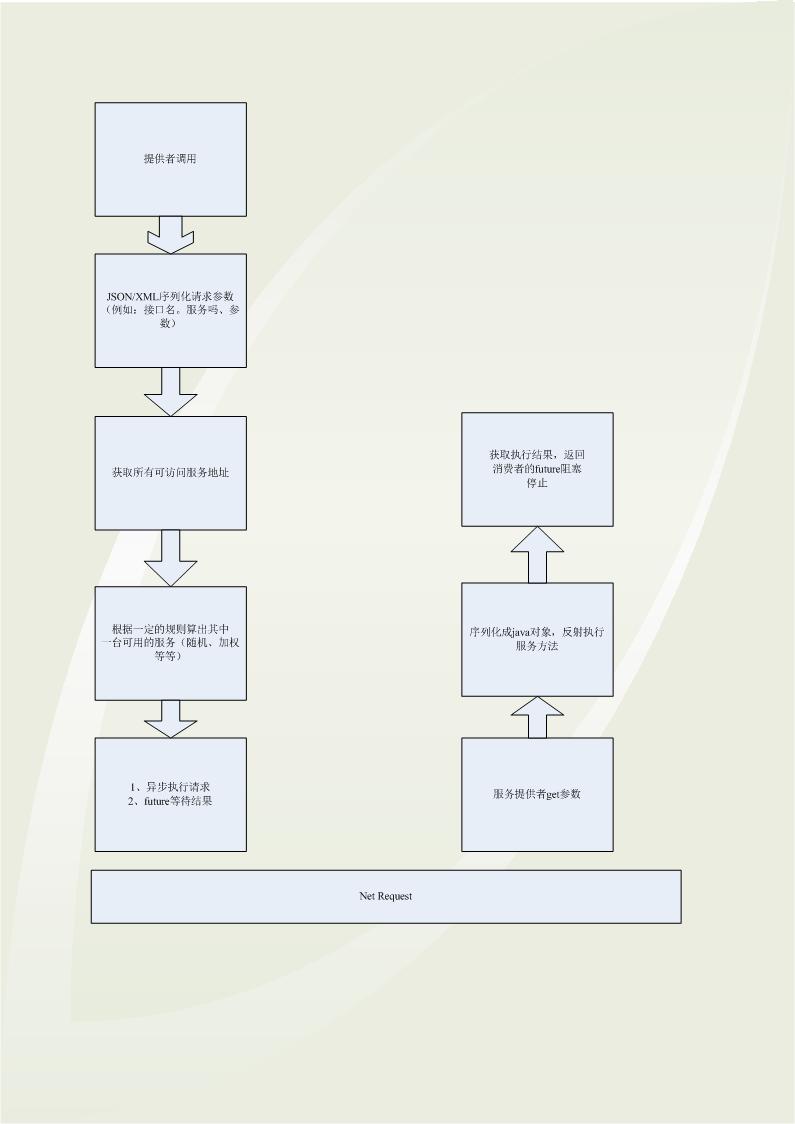
**HTTP\_RPC**

一、应用场景：

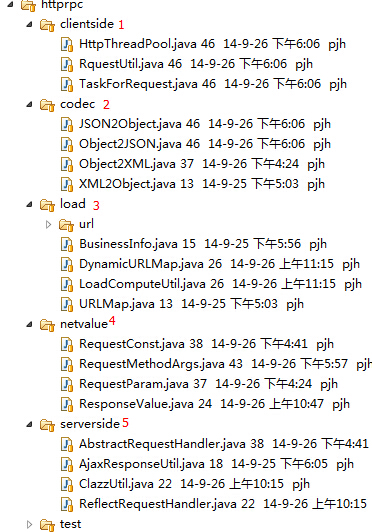
某些服务需要独立的被多个模块或者应用调用；

同一个服务多份部署，需要提供无差别服务，同时需要负载均衡避免因负载而出现的问题

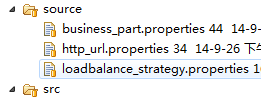
二、图解逻辑：



三、图解包结构



1. 消费者请求代码，默认线程池thread数目是服务器CPU number；
2. 负责消费者请求、提供者执行时xml、json到JavaBean之间的转换；
3. 对负载中的服务器提供消费者选择服务策略，默认随机；
4. 请求参数或者网络传输参数的JavaBean表示
5. 服务端处理请求handler，默认实现有基于反射构造对象执行，真是的情况应该是提供者自己实现AbstractRequestHandler抽象类，因为不同的应用对“服务”的定义是不同的。



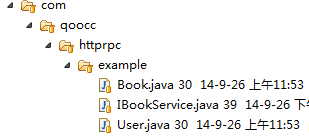
http\_url.properties:配置着所有可选服务器

loadbalance\_strategy.properties:配置默认的选举策略

四、使用方式（待补充）

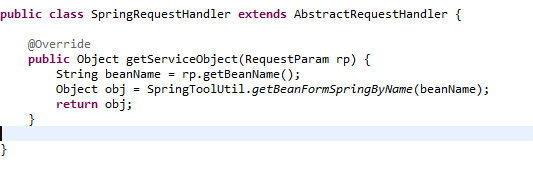
1、消费者端需要对JavaBean进行xml/json的转换，因此原则上要求服务端是基于接口的开发，即定义一套接口，消费者端和提供端都需要该接口，但是服务端提供对接口的实现。

例如：



这就是一套标准的接口。

2、提供这端需要自定义处理业务的handler，例如：



因为spring容器的广大应用，我们很多服务组件都是由spring来实现管理的，当然线程安全也是它实现的，因此我将spring托管的beanName作为服务的名字。

3、服务这段的action或者servlet要接受三个字符参数

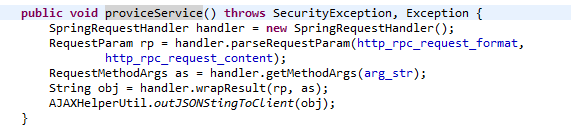


第一个是请求参数内容（http\_rpc\_request\_content）的encode方式，XML或者JSON

第二个参数是请求内容，定义着请求的服务名字，方法名，接口或者类名

第三个参数是参数的XML，以供服务端decode

4、服务端执行如下



其实每个服务端只需要有一个servlet/action一个方法即可，因为所有的服务调用都是通过该方法，所有不同业务的执行都隐藏在了服务端自定义的handler中，例如我的例子是springRequestHandler。