[一个简单RPC框架是如何炼成的(III)——实现带参数的RPC调用](http://blog.csdn.net/crylearner/article/details/46899687)

上一篇，我们制定了一个很简单的RPC消息 的格式，但是还遗留了两个问题

* 我们并没有实现相应的encode和decode方法，没有基于可以跨设备的字符串传输，而是直接的内存变量传递。
* **现在的RPC request不支持带参数的请求命令。如add(a, b), 如何在RPC消息中描述参数a,b 。**

我先来实现第二个问题，即带参数的RPC调用。

其实，也没什么太大不同。既然是要带参数，那只能扩展原来的Request消息了，加个parameter成员，用于表示参数，具体的格式采用字典方式，{ ’arg1‘， arg1, 'arg2', arg2 ,....}。 这样就可以解决多参数的表示问题。

**[python]** [view plain](http://blog.csdn.net/crylearner/article/details/46899687) [copy](http://blog.csdn.net/crylearner/article/details/46899687)

1. **class** Request(object):
2. '''''
3. @RPC请求，包含命令id和请求内容两部分。这个实现，与具体的RPC协议相关。
4. @这里是简化起见，采用python自身的字典作为其协议的数据结构
5. '''

8. **def** \_\_init\_\_(self):
9. '''''
10. Constructor
11. '''
12. self.id = 0 #id的作用在于将Request和Response建立绑定关系.在异步调用时就有用
13. self.command = None #sayHello
14. self.parameter = {}
16. **def** \_\_str\_\_(self):
17. **return** ''.join(('id: ', str(self.id),  '    command: ', str(self.command), '    parameter: ', str(self.parameter)))

add(a=1, b=2)的RPC 请求就是这个样子了

**[python]** [view plain](http://blog.csdn.net/crylearner/article/details/46899687) [copy](http://blog.csdn.net/crylearner/article/details/46899687)

1. Request : id = 3, command = 'add', parameter = {'a':1, 'b':2}

对应的，客户端的add方法，我们可以这么写

**[python]** [view plain](http://blog.csdn.net/crylearner/article/details/46899687) [copy](http://blog.csdn.net/crylearner/article/details/46899687)

1. **def** add(self, a, b):
2. req = Request()
3. req.id = 3
4. req.command = 'add'
5. req.parameter = {'a':a, 'b':b}
6. **return** self.request(req)

那么服务端收到这个RPC请求后，怎么处理得到参数呢？一个传统而稍显笨拙的方式是：

**[python]** [view plain](http://blog.csdn.net/crylearner/article/details/46899687) [copy](http://blog.csdn.net/crylearner/article/details/46899687)

1. **def** add(self, a, b):
2. **return** a + b
4. **def** procReqeust\_\_add(self, req):
5. parameter = req.parameter
6. a = parameter.get('a')
7. b = parameter.get('b')
8. **return** self.add(a, b)

这种方式的缺点就是没办法偷懒。（懒懒懒大笑），每一个RPC调用，都要怎么处理一下，烦死了，没有任何技术含量的纯苦力活，但还考验细心，一不小心搞错a或者b的名字了，呵呵，等着被请喝茶吧。

这时候，大杀器上场，脚本语言就是找个好。总有你想不到的方便。直接上代码

**[python]** [view plain](http://blog.csdn.net/crylearner/article/details/46899687) [copy](http://blog.csdn.net/crylearner/article/details/46899687)

1. **def** procReqeust\_\_add(self, req):
2. parameter = req.parameter
3. **return** self.add(\*\*parameter)

对上面\*\*parameter不懂的同学自行度娘。这里只是简单解释一下：\*\*parmater的作用同那块笨拙的代码一样，但有一个前提条件，即使add声明时，参数变量名a，b不能变。（c++同学可能晕了，竟然有语音有这样的说法疑问）

至此，使用这种新的方式，我们server的代码就是这个样子的，对没有参数的方法，上面\*\*也是可以放心使用的

**[python]** [view plain](http://blog.csdn.net/crylearner/article/details/46899687) [copy](http://blog.csdn.net/crylearner/article/details/46899687)

1. **def** procRequest(self,req):
2. rsp = Response()
3. rsp.id = req.id
4. **if** req.command == 'sayHello':
5. rsp.result = self.sayHello(\*\*req.parameter)
6. **elif** req.command == 'add':
7. rsp.result = self.add(\*\*req.parameter)
8. **else**:
9. **raise** Exception("unknown command")

over，搞定，还挺满意的