众安保险

HSF的那些事儿

千 橙

金融技术部

简单远程调用

客户端

```
代码
HelloService helloService=getProxy(HelloService.class,”127.0.0.1”,”2580);
System.out.println(helloService.sayHello(“hi, charles”));
Pu blic  <T>  T getProxy(Class<T> interfaceClass, String host, int port){
return (T) Proxy.newProxyInstance(interfaceClass.getClassLoader(), new
        Class<?>[] {interfaceClass},
        new InvocationHandler() {
        public Object invoke(Object proxy, Method method, Object[]
        arguments) throws Throwable {
        Socket socket = new Socket(host, port);
        ObjectOutputStream output = new
        ObjectOutputStream(socket.getOutputStream());
        output.writeUTF(method.getName());
        output.writeObject(method.getParameterTypes());
        output.writeObject(arguments);
        ObjectInputStream input = new
        ObjectInputStream(socket.getInputStream());
        return input.readObject();}
    });
}
输出：
hi,Charles ,wellcome.
```

服务端

```
代码：
HelloServiceImple:
Public String sysHello(String input){return input+” ,wellcome.”}
Public void static Main(String[] args){
ServerSocket server = new ServerSocket(port);
for(;;) {
        final Socket socket = server.accept();
        new Thread(new Runnable() {
        ObjectInputStream input = new
        ObjectInputStream(socket.getInputStream());
        String methodName = input.readUTF();
        Class<?>[] parameterTypes = (Class<?>[])input.readObject();
        Object[] arguments = (Object[])input.readObject();
        ObjectOutputStream output = new
        ObjectOutputStream(socket.getOutputStream());
        Method method = service.getClass().getMethod(methodName,
                parameterTypes);
        Object result = method.invoke(service, arguments);
        output.writeObject(result);
        } ).start();
}}
```

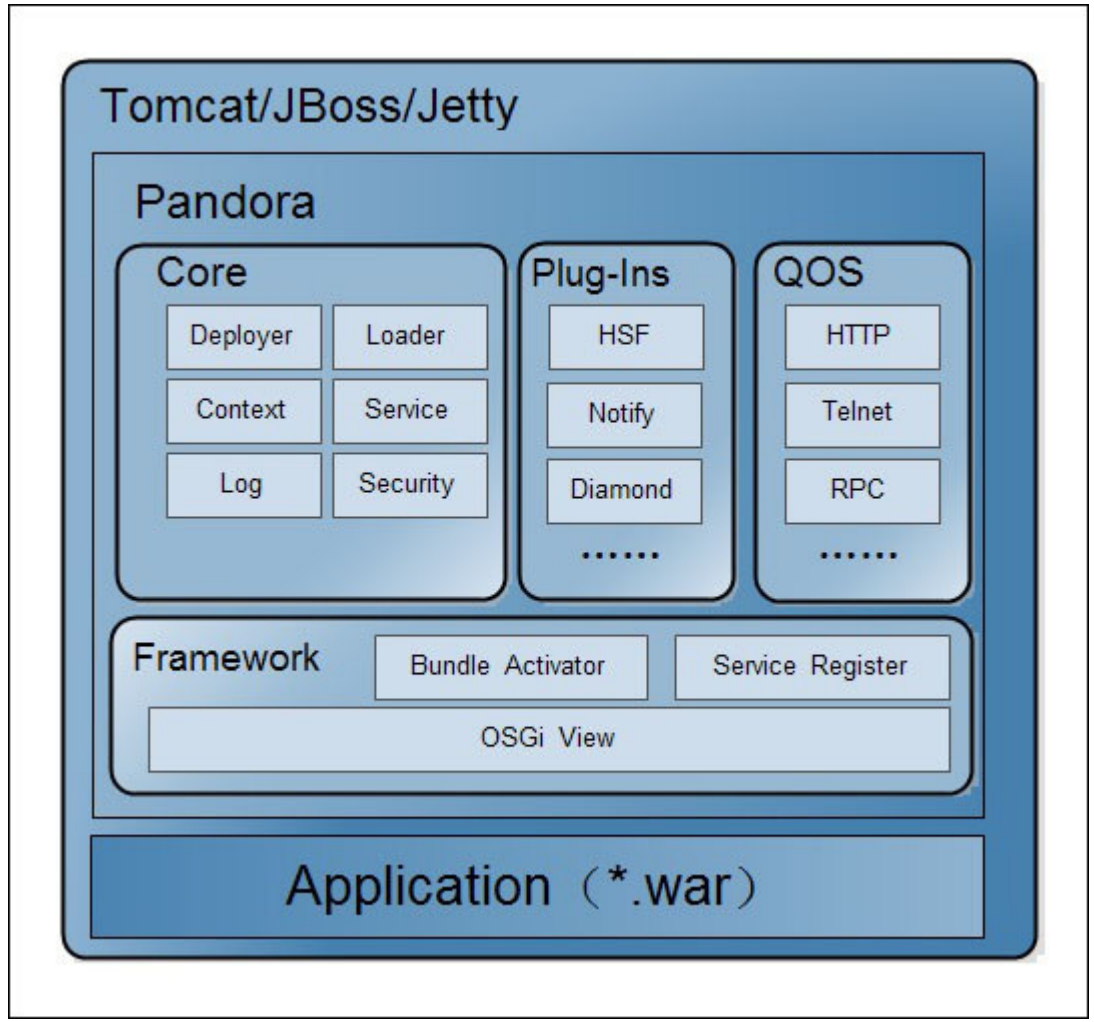
Demo： simple-rpc

https://github.com/iqiancheng/simple-rpc

好像不够优雅

- 网络传输方式:BIO
- 序列化方式:Java
- 线程模型:每连接每线程
- JDK代理

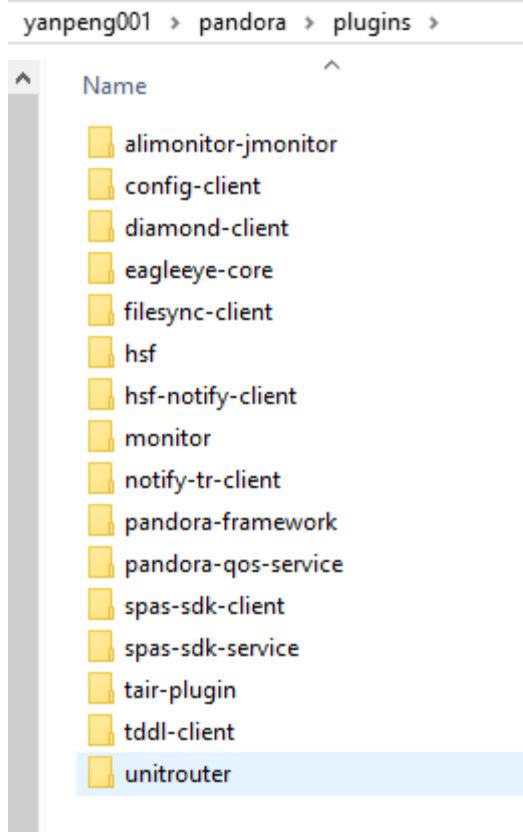
C:\Users\yanpeng001\logs\tomcat-monitor

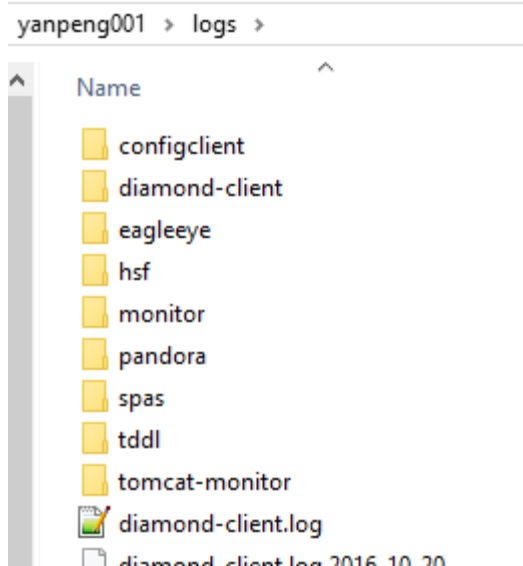


http://www.tuicool.com/articles/jIJ36r

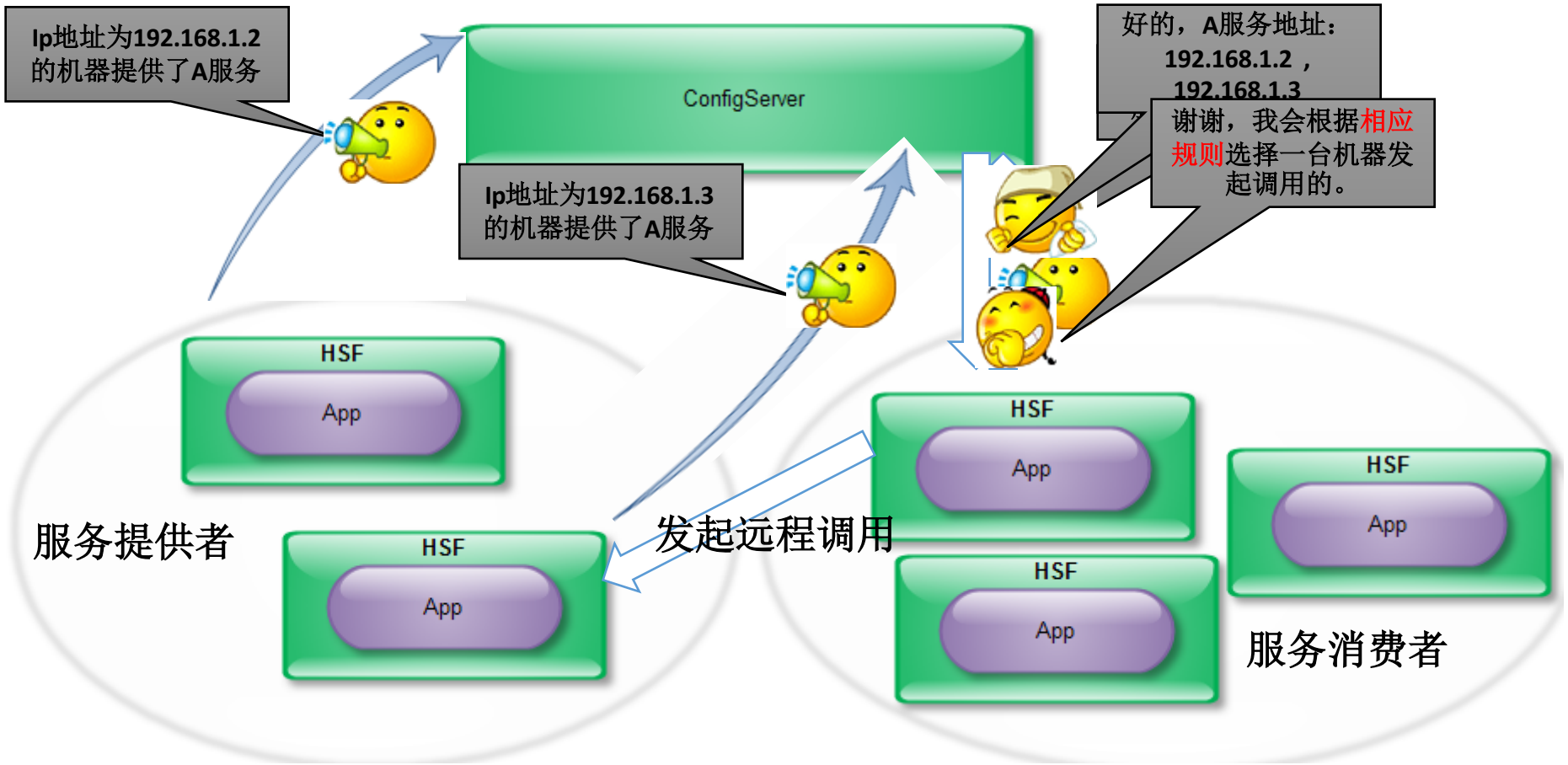
Pandora

```
**********************************************************************************************
**                                                                                          **
**                            Pandora Container                                             **
**                                                                                          **
**      Pandora Host:     192.168.200.227                                                   **
**      Pandora Version:  2.0.5.5.9                                                         **
**      SAR Version:      2_5_151013                                                        **
**      Package Time:     2015-10-13 22:24:54                                               **
**                                                                                          **
**      Plug-in Modules: 16                                                                 **
**                                                                                          **
**         spas-sdk-client ................................ 1.1.5                          **
**         eagleeye-core .................................. 1.4.4                          **
**         diamond-client ................................. 3.7.1.6                        **
**         spas-sdk-service ............................... 1.2.0                          **
**         config-client .................................. 1.6.6.5                        **
**         unitrouter ..................................... 1.0.14-SNAPSHOT                **
**         notify-tr-client ............................... 1.8.19.26                      **
**         monitor ........................................ 1.0.4                          **
**         hsf-notify-client .............................. 1.8.19.9                       **
**         hsf ............................................ 2.1.1.3                        **
**         filesync-client ................................ 1.0.3                          **
**         tair-plugin .................................... 1.0.4                          **
**         pandora-qos-service ............................ 2.0.9                          **
**         pandora-framework .............................. 2.0.7                          **
**         alimonitor-jmonitor ............................ 1.1.2                          **
**                                                                                          **
**      [WARNING] All these plug-in modules will override maven pom.xml dependencies.       **
**                See More: http://t.cn/RhtjBZp                                             **
**                                                                                          **
**********************************************************************************************
```





应用层：HSFSpringProviderBean/ HSFSpringConsumerBean
协议层：RPC协议（TCP/IP协议，Webservice协议，Google protocol buffers协议）/序列化协议（java，Hessian）
核心服务层：RPC服务/路由规则服务/地址服务/配置服务/notify消息服务/OSGI容器及jar依赖管理。
QOS（Quality of Service，服务质量）：监控 日志（哈勃，logstat）
容器接入层：Tomcat，Jboss


Ip地址为192.168.1.2
的机器提供了A服务
ConfigServer
好的，A服务地址：
192.168.1.2 ，
192.168.1.3
谢谢，我会根据相应规则选择一台机器发起调用的。
Ip地址为192.168.1.3
的机器提供了A服务
HSF
App
HSF
App
HSF
App
HSF
App
HSF
App
服务提供者
发起远程调用
服务消费者

HSF-Container

HSF配置-服务端

- HSF服务端入口HSFSpringProviderBean
  - HSFSpringProviderBean 提供了HSF 服务发布的功能，可以将Spring Bean 发布为HSF服务。它有一系列的属性，用于控制服务的各种配置信息，比如服务接口，服务实现，版本等等

```
<bean id="smsSenderProvider" class="com.taobao.hsf.app.spring.util.HSFSpringProviderBean">
    <property name="serviceInterface"
        value="com.zhongan.sms.SmsSender" />
    <property name="target" ref="smsSender" />
    <property name="serviceVersion" value="${za.kawaii.hsf.providerversion.common}" />
    <property name="clientTimeout" value="${za.kawaii.hsf.clienttimeout.common}" />
    <property name="clientIdleTimeout" value="60" />
    <property name="serializeType" value="java" />
</bean>
```

HSF配置-服务端参数

属性名字	作用	默认值
serviceInterface	服务接口	
target	服务实现	
serviceVersion	版本	1.0.0
clientTimeout	超时时间	3s
corePoolSize	核心线程数	0
maxPoolSize	最大线程数	0
delayedPublish	延迟发布	False
serializeType	序列化方式	Hessian
serviceGroup	组别	HSF
supportAsynCall	支持可靠异步调用	False
methodToInjectConsumerIp	注入客户端ip方法	
methodSpecial	针对个别方法设置超时 methodName clientTimeout	

HSF配置-客户端

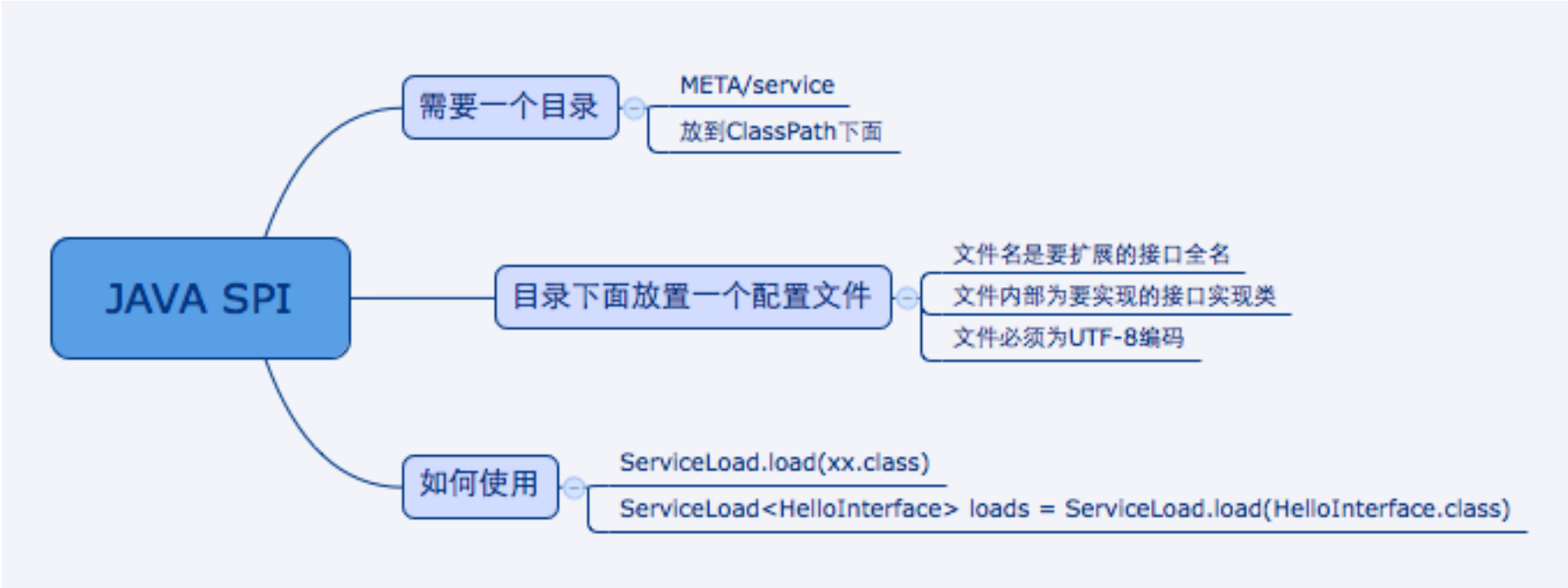
- HSF客户端入口HSFSpringConsumerBean
  - HSFSpringProviderBean 为Spring 提供了HSF 服务调用的功能，为单元测试、开发环境部署以及正式部署提供支持。

```
<bean id="policyService" class="com.taobao.hsf.app.spring.util.HSFSpringConsumerBean" init-method="init">
    <property name="interfaceName" value="com.zhongan.core.policy.service.PolicyService" />
    <property name="version" value="${za.kawaii.hsf.consumerversion.common}" />
</bean>
```

HSF配置-客户端参数

属性名字	作用	默认值
interfaceName	服务接口	
version	版本	1.0.0
group	组别	HSF
asyncallMethods	异步调用方法	
callBackHander	ReliableCallback时的回调函数	
callbackMethodSuffix	回调方法名后缀	
interfaceMethodToAttachContext	设置调用上下文的方法	
invokeContextThreadLocal	调用上下文ThreadLocal的bean	
Target	Test时服务端url	
methodSpecials	方法的超时，覆盖服务端的值	
callbackInvoker	回调方式调用时的统一回调函数覆盖asyncallMethod里的配置	

Java SPI 服务发现机制

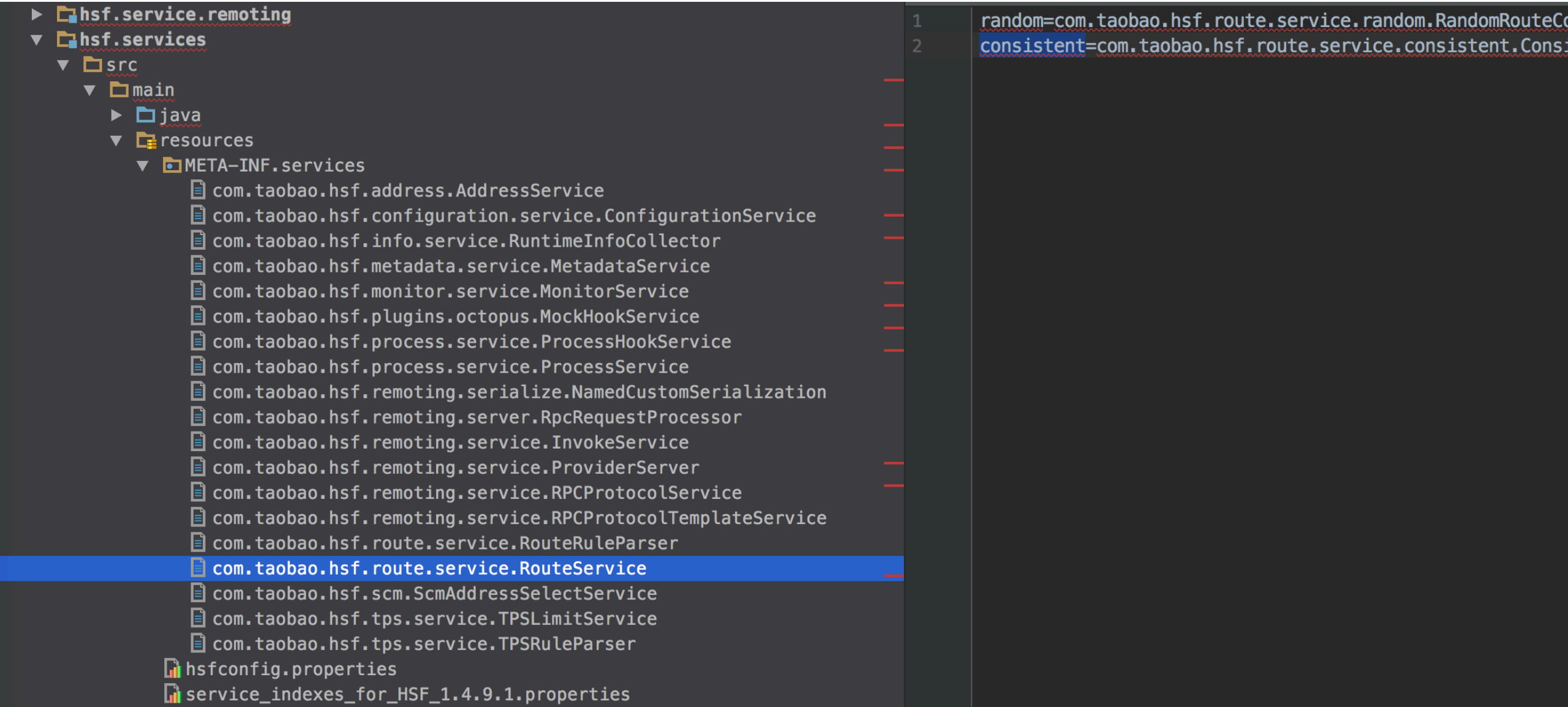


Service Provider Interface

SPI现实应用场景: java.sql.Driver的spi实现，有mysql驱动、oracle驱动等。以mysql为例，实现类是com.mysql.jdbc.Driver，在mysql-connector-java-5.1.6.jar中，我们可以看到有一个META-INF/services目录，目录下有一个文件名为java.sql.Driver的文件，其中的内容是com.mysql.jdbc.Driver

HSFServiceContainerHelp

HSF中的SPI实现应用类



HSFServiceContainerHelp

HSF中的SPI实现应用类

```
56      /unchecked/
57      public  <T> List<T> getInstances(Class<T> classType) {
58          List<T> list = (List<T>) INSTANCE_CACHE.get(classType);
59          if (list == null) {
60              try {
61                  list = new ArrayList<T>();
62                  for (T instance : ServiceLoader.load(classType, HSFServiceContainer.class.getClassLoader())) {
63                      list.add(instance);
64                  }
65                  INSTANCE_CACHE.putIfAbsent(classType, list);
66                  return (List<T>) INSTANCE_CACHE.get(classType);
67              } catch (RuntimeException e) {
68                  if (HSFPluggable.class.isAssignableFrom(classType)) {
69                      LOGGER.warn("plugin: " + classType.getName() + " is not activated.");
70                      return null;
71                  } else {
72                      throw e;
73                  }
74              }
75          } else {
76              if (List.class.isAssignableFrom(list.getClass())) {
77                  return list;
78              } else {
79                  throw new RuntimeException("[Init HSFService Container Error(List)]" + classType);
```

HSF-Registry

ConfigServer：服务地址的注册；消费端服务地址的推送；感应服务提供者的状态，当服务者断开时重新推送服务地址

注意：Client端根据路由规则计算调用时的可用地址

Routing & FailOver的逻辑在config-client包中

ConfigServer是什么？

ConfigServer是一款分布式软负载中间件，具有以下特点：

- 非持久数据和生命周期
- 自动聚合
- 生命周期关联
- 及时通知
- 使用内存保存数据

ConfigServer的功能

- 通过维护和dataserver心跳来获知集群中存活节点的信息
- 根据存活节点的信息来构建数据在集群中的分布表。
- 提供数据分布表的查询服务。
- 调度dataserver之间的数据迁移、复制。

所以关于TR心跳机制的结论是：

1、**只有客户端会主动向服务端发起心跳包**

2、客户端和服务端在(current - lastRead >= idleTime + maxReadIdle)成立的时候都会主动关闭客户端连接

3、**idleTime是客户端维护，报送到服务端**；maxReadIdle对应客户端和服务端都是一样的，默认是30s

4、所以，目前configserver的心跳超时时间正常情况下，最长是35s（在机器的load较高导致线程的调度不及时时，该时间会大于35s）

http://gitlab.alibaba-inc.com/middleware/configserver/wikis/HeartBeat

HSF-Routing

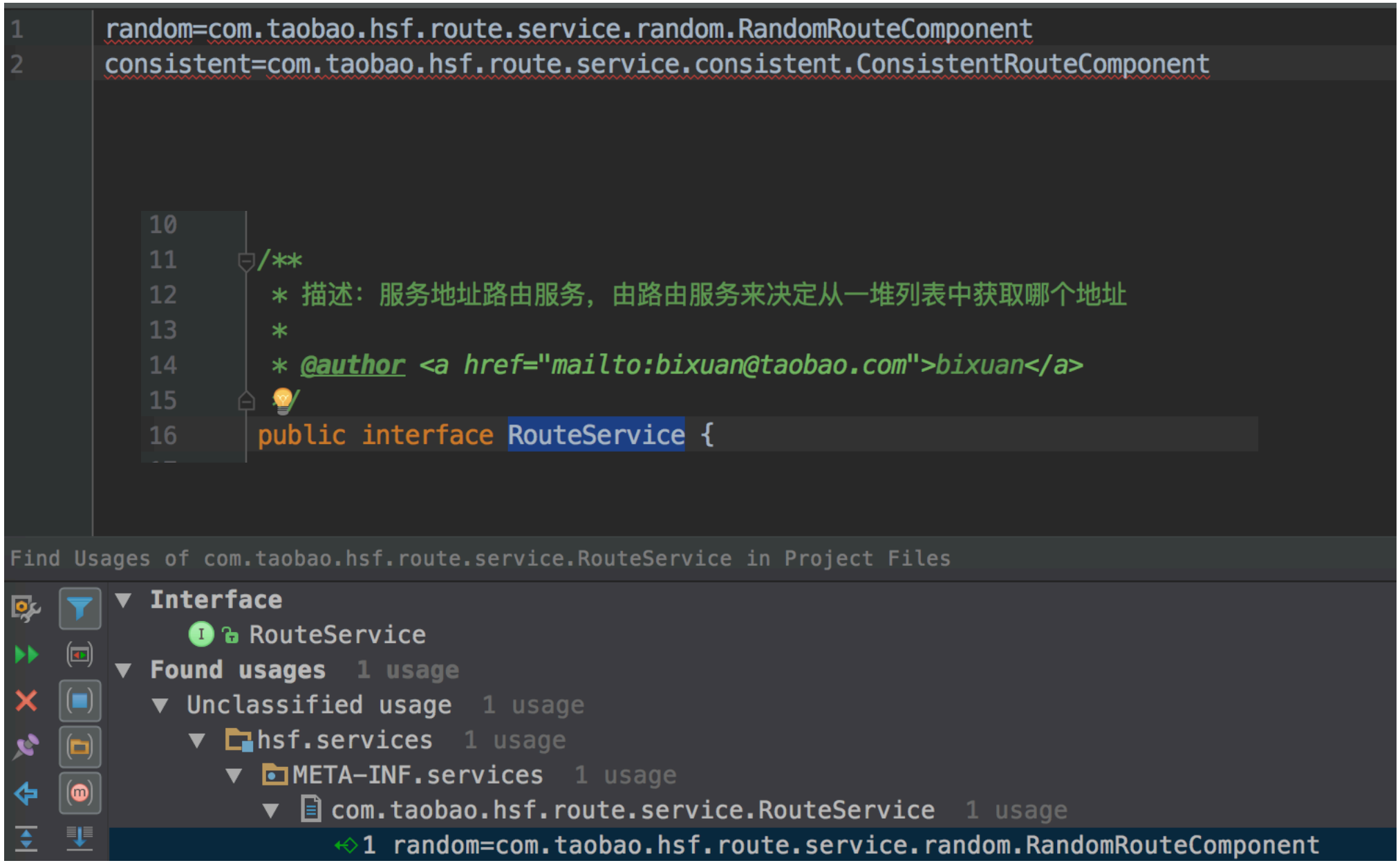
ThreadPoolManager

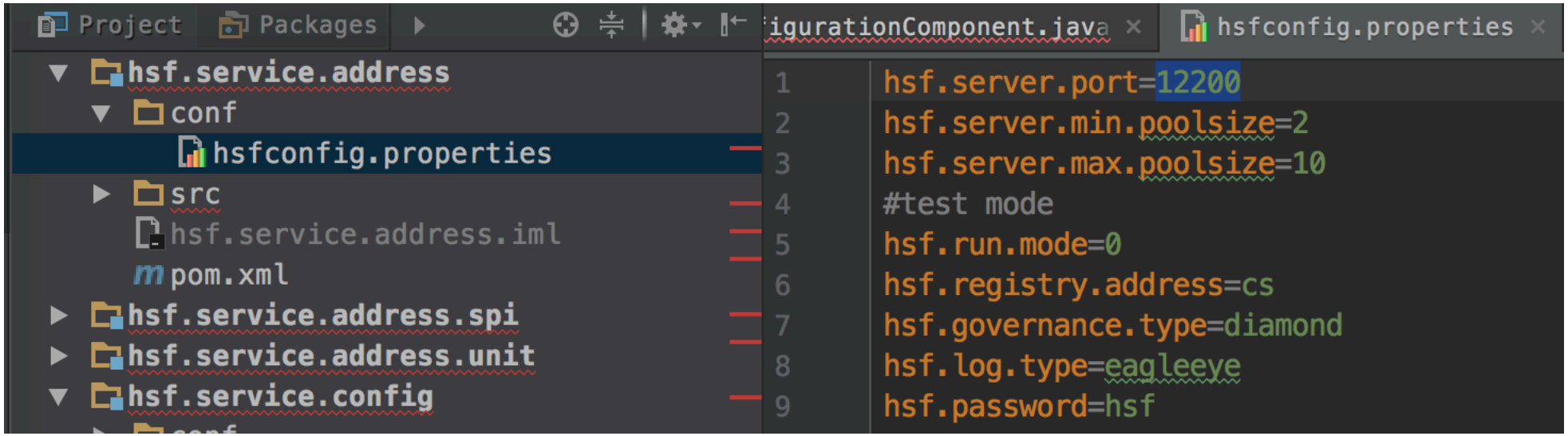
```java
public class ThreadPoolManager {
    private static final long keepAliveTime = 300L;

    // no reject and use IO thread to let it compatible with biz thread
    private final RejectedExecutionHandler handler = new IgnoreRunsPolicy();
    private final ThreadPoolExecutor defaultPoolExecutor;
    private final Map<String, ThreadPoolExecutor> poolCache = new HashMap<~>();

    public ThreadPoolManager(int corePoolSize, int maximumPoolSize, int queueSize) {
        final BlockingQueue<Runnable> workQueue = new SynchronousQueue<Runnable>();
        final ThreadFactory threadFactory = new NamedThreadFactory(HSFThreadNameSpace.HSF_NETTY_PROCESSOR_DEFAULT);
        defaultPoolExecutor = new ThreadPoolExecutor(corePoolSize, maximumPoolSize, keepAliveTime, TimeUnit.SECONDS,
                workQueue, threadFactory, handler);
    }

    public void allocThreadpool(final String serviceUniqueName, int corePoolSize, int maximumPoolSize)
            throws HSFException {
        if (poolCache.containsKey(serviceUniqueName)) { // 对同一个服务重复分配线程池时，抛出异常
            throw new HSFException(MessageFormat.format(
                    "[ThreadPool Manager] Duplicated thread pool allocation request for service [{0}].",
                    new Object[] { serviceUniqueName }));
        }

        if (defaultPoolExecutor == null || defaultPoolExecutor.isShutdown()) { // 线程池已关闭
            throw new HSFException(MessageFormat.format(
                    "[ThreadPool Manager] Can not allocate thread pool for service [{0}].",
                    new Object[] { serviceUniqueName }));
        }

        int balance = defaultPoolExecutor.getMaximumPoolSize(); // 剩余线程数量
        // 剩余线程数量小于申请的线程数量
        if (balance < maximumPoolSize) {
            throw new HSFException(
```

```
/**
 * 描述: 服务地址服务实现类
 *
 * <pre>
 * 路由规则分为接口级-->方法级-->参数级
 * 若某一级的路由规则不存在，即其返回的key为null，或在routingRuleMap中找不到相应值，
 *      则认为这一级对路由地址不做限制，地址取上一级的全部地址。
 *
 * 方法级路由规则（正则式）最终地址列表会和接口级地址列表作交集
 * 参数级路由规则（正则式）最终地址列表会和方法级地址列表作交集
 *
 * 地址（ip）正则式列表的语义为: 若一个地址满足任何一个列表中的正则式，则地址会被加入到最终调用列表中
 *
 *
 * 路由规则是一段string表示的groovy代码，代码中包含一个groovy类，groovy类包含以下几个方法中的全部或者部分
 *      Map<String, List<String>> routingRuleMap(); (key: 用户定义的key; value: 地址过滤的正则式列表)
 *      String interfaceRoutingRule(); 返回routingRuleMap()中的key，对应接口级规则
 *      String mathodRoutingRule(String methodName, String[] paramTypeStrs); 返回routingRuleMap()中的key，方法级规则
 *      Object argsRoutingRule(String methodName, String[] paramTypeStrs); --参数级规则
 *         返回一个groovy闭包，闭包接受一个Object[]参数，返回规则映射表routingRuleMap中的一个key
 *
 * HSF每次接到推送的路由规则，或新的地址列表后:
 *    1. 调用groovy.routingRuleMap获得routingRuleMap；若方法不存在则认为不做任何限制，忽略以下各步骤。
 *    2. 调用interfaceRoutingRule获得接口级key，以返回key取得routingRuleMap对应的正则式列表，计算接口级地址列表并缓存
 *       如果interfaceRoutingRule方法不存在、返回null或routingRuleMap中不存在的key，
 *       则取全部可用服务地址作为接口级地址列表
 *    3. 将routingRuleMap中除接口级之外的其他正则式列表，分别作用于接口级地址列表，存储计算结果
 *    4. 对每个接口方法，以其签名为参数，调用groovy.mathodRoutingRule，以返回key取得3中缓存的地址列表，
 *       作为方法级地址列表。若mathodRoutingRule方法不存在、返回null或routingRuleMap中不存在的key，
 *       则取接口级地址列表作为方法级地址列表
 *    5. 对每个接口方法，以其签名为参数，调用groovy.argsRoutingRule，若不为空，则hsf编译返回的闭包，缓存编译后的闭包对象
 *       若argsRoutingRule方法不存在或返回空，说明接口方法不需要参数路由。hsf在接口方法->闭包的映射表中不保存映射
```

```
 *    5. 对每个接口方法，以其签名为参数，调用groovy.argsRoutingRule，若不为空，则hsf编译返回的闭包，缓存编译后的闭包对象
 *       若argsRoutingRule方法不存在或返回空，说明接口方法不需要参数路由。hsf在接口方法->闭包的映射表中不保存映射
 *
 * 方法正真调用时，HSF以方法签名在缓冲中查找groovy闭包对象，若找到，则将方法参数数组作为参数执行闭包，
 *    以闭包执行结果作为key取得3中缓存的地址列表，与该方法的方法级地址列表取交集，并缓存交集结果。
 *    闭包执行结果返回null，则认为不对地址列表做过滤，直接取方法签名本身对应的ip地址列表
 *
 *
 *   所有可用服务地址   -->   接口级地址      -->   方法级地址        -->   参数级地址
 *
 *                      |                    |    ┌ xxx.xxx.xxx.xxx |                    ┌ xxx.xxx.xxx.xxx
 *   xxx.xxx.xxx.xxx    |                    | M1-| ...             |                ┌ key1-| ...
 *   xxx.xxx.xxx.xxx    | xxx.xxx.xxx.xxx    |    └ xxx.xxx.xxx.xxx | M1-closure1    |      └ xxx.xxx.xxx.xxx
 *   ...                | xxx.xxx.xxx.xxx    | ...                  |                └ ...
 *                      | ...                |                      | ...
 *                      |                    | ...                  |
 *                      |                    |                      |                    ┌ xxx.xxx.xxx.xxx
 *                      |                    |    ┌ xxx.xxx.xxx.xxx |                ┌ key1-| ...
 *                      | xxx.xxx.xxx.xxx    | Mn-| ...             | Mn-closuren    |      └ xxx.xxx.xxx.xxx
 *   xxx.xxx.xxx.xxx    |                    |    └ xxx.xxx.xxx.xxx |                └ ...
 *                      |                    |                      |
 *                      ↓                    ↓                      ↓
 *                 接口级路由             方法级路由              参数级路由
 *            interfaceRoutingRule    mathodRoutingRule     argsRoutingRule
 *  \                   |                    |                      |                    /
 *   \                  |                    |                      |                   /
 *    \_________________Map<String, List<String>> routingRuleMap()_____________________/
 *
```

Route


random=com.taobao.hsf.route.service.random.RandomRouteComponent
consistent=com.taobao.hsf.route.service.consistent.ConsistentRouteComponent
/**
* 描述：服务地址路由服务，由路由服务来决定从一堆列表中获取哪个地址
*
* @author bixuan
*/
public interface RouteService {
Find Usages of com.taobao.hsf.route.service.RouteService in Project Files
Interface
RouteService
Found usages 1 usage
Unclassified usage 1 usage
hsf.services 1 usage
META-INF.services 1 usage
com.taobao.hsf.route.service.RouteService 1 usage
1 random=com.taobao.hsf.route.service.random.RandomRouteComponent


Project
Packages
igurationComponent.java
hsfconfig.properties
hsf.service.address
conf
hsfconfig.properties
src
hsf.service.address.iml
pom.xml
hsf.service.address.spi
hsf.service.address.unit
hsf.service.config
1 hsf.server.port=12200
2 hsf.server.min.poolsize=2
3 hsf.server.max.poolsize=10
4 #test mode
5 hsf.run.mode=0
6 hsf.registry.address=cs
7 hsf.governance.type=diamond
8 hsf.log.type=eagleeye
9 hsf.password=hsf

TCP粘包和拆包产生的原因

1.应用程序写入数据的字节大小大于套接字发送缓冲区的大小
2.进行MSS大小的TCP分段。MSS是最大报文段长度的缩写。MSS是TCP报文段中的数据字段的最大长度。数据字段加上TCP首部才等于整个的TCP报文段。所以MSS并不是TCP报文段的最大长度，而是：MSS=TCP报文段长度-TCP首部长度
3.以太网的payload大于MTU进行IP分片。MTU指：一种通信协议的某一层上面所能通过的最大数据包大小。如果IP层有一个数据包要传，而且数据的长度比链路层的MTU大，那么IP层就会进行分片，把数据包分成托干片，让每一片都不超过MTU。注意，IP分片可以发生在原始发送端主机上，也可以发生在中间路由器上。

TCP粘包和拆包的解决策略

1.消息定长。例如100字节。

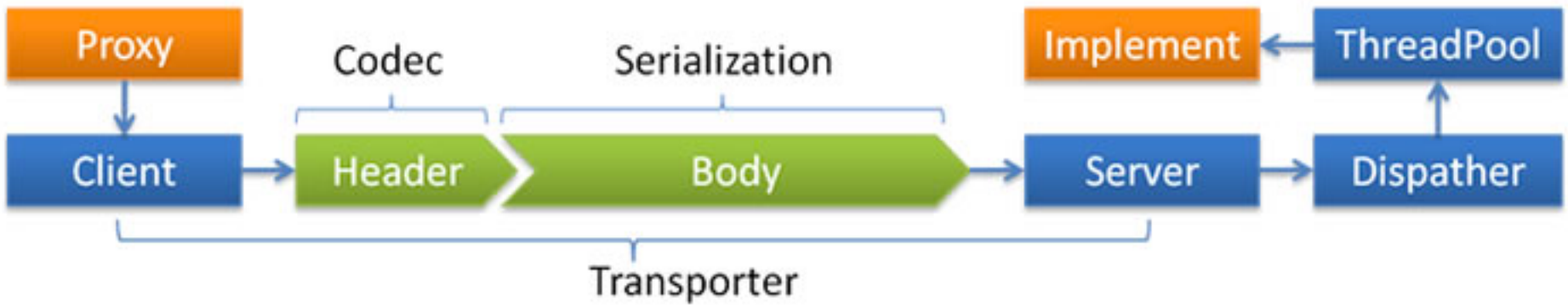
2.在包尾部增加回车或者空格符等特殊字符进行分割，典型的如FTP协议

3.将消息分为消息头和消息尾。

4.其它复杂的协议，如RTMP协议等。

```
 *
 * Common RPC Protocol
 * 请求:
 * 字节   描述
 * 1    标志HSF2协议
 * 2    版本
 * 3    请求
 * 4    序列化方式
 * 5~7 保留字节
 * 8~15     请求ID
 * 16~19    超时时间
 * 20~35    服务名，方法名，方法参数值的长度
 * 不等   服务名，方法名，方法参数值的值
 * +4   附加信息长度
 * 不等   附加信息值
 * 响应:
 * 字节   描述
 * 1    标志HSF2协议
 * 2    版本
 * 3    响应
 * 4    状态code
 * 5    序列化方式
 * 6~8 保留字节
 * 9~16     对应的请求ID
 * 17~20    返回值的长度大小
 * 不等   返回值的值
 *
 * @author kongming.lrq
 *
 */
public class RPCProtocol implements Protocol {
```

```
49  public Object decode(ByteBufferWrapper wrapper, int originPos) throws Exception {
50      if (wrapper.readableBytes() < 2) {
51          wrapper.setReaderIndex(originPos);
52          return null;
53      }
54      byte version = wrapper.readByte();
55      if (version == VERSION) {
56          byte type = wrapper.readByte();
57          if (type == REQUEST) {
58              return decodeRequest(wrapper, originPos);
59          } else if (type == RESPONSE) {
60              return decodeRpcResponse(wrapper, originPos);
61          } else {
62              LOGGER.error("", "protocol type : " + type + " is not supported!");
63              return null;
64          }
65      } else {
66          LOGGER.error("", "protocol version :" + version + " is not supported!");
67          return null;
68      }
69  }
70
71  private Object decodeRequest(ByteBufferWrapper wrapper, final int originPos) {
72      if (wrapper.readableBytes() < REQUEST_HEADER_LEN - 2) {
73          wrapper.setReaderIndex(originPos);
74          return null;
75      }
76      byte codecType = wrapper.readByte();
77      // ignore the extended three bytes
78      wrapper.setReaderIndex(wrapper.readerIndex() + 3);
79      long requestId = wrapper.readLong();
80      int timeout = wrapper.readInt();
81      int targetInstanceLen = wrapper.readInt();
82      int methodNameLen = wrapper.readInt();
83      int argsCount = wrapper.readInt();
84      int argInfosLen = argsCount * 4 * 2;
85      int expectedLenInfoLen = argInfosLen + targetInstanceLen + methodNameLen + 4;
86      int size = expectedLenInfoLen;
87      if (wrapper.readableBytes() < expectedLenInfoLen) {
88          wrapper.setReaderIndex(originPos);
89          return null;
90      }
91      int expectedLen = 0;
92      int[] argsTypeLen = new int[argsCount];
93      for (int i = 0; i < argsCount; i++) {
94          argsTypeLen[i] = wrapper.readInt();
95          expectedLen += argsTypeLen[i];
96      }
97      int[] argsLen = new int[argsCount];
```


Proxy
Codec
Serialization
Implement
ThreadPool
Client
Header
Body
Server
Dispather
Transporter

HSF问题排查

- 常见问题：服务无法订阅
- 排查内容：
  - 1，该服务是否发布
    - 通过config ops查看
  - 2，服务名，版本是否正确
    - 核对调用服务名和版本
  - 3，客户端机器环境设置
    - Hosts 文件中域名绑定是否正确
  - 4，客户端查看hsf.log，确认地址推送情况

HSF问题排查

- 常见问题：服务调用超时
- 排查内容：
  - 1，服务端处理是否流程过长
    - 检查服务端日志和设置的HSF超时时间
  - 2，是否是通讯/序列化异常
    - 查询客户端/服务端日志
  - 3，客户端,服务端是否full gc
    - 查看gc日志检查
  - 4，是否服务端处理异常
    - 特别注意share版本

通过telnet命令进入Pandora容器命令行模式

```
➜  ~ ssh 5080@10.139.107.1
5080@10.139.107.1's password:
Last login: Sun Jan 22 17:06:22 2017 from 10.139.0.6

Welcome to aliyun Elastic Compute Service!

[5080@iZ233e93pgiZ ~]$
[5080@iZ233e93pgiZ ~]$
[5080@iZ233e93pgiZ ~]$
[5080@iZ233e93pgiZ ~]$ jps -vl
812 sun.tools.jps.Jps -Denv.class.path=.:/opt/jdk1.7.0_76/lib/dt.jar:/opt/jdk1.7.0_76/lib/tools.jar -Dapplication.home=/opt/jdk1.7.0_76 -Xms8m
20018 org.apache.catalina.startup.Bootstrap -Djava.util.logging.config.file=/alidata1/5080/za-fcp-biz-card/.default/conf/logging.properties -Dpandora.qos.port=5082 -Dhsf.server.port=5081 -Dhsf.http.port=5083 -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5089 -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Dlog4j.defaultInitOverride=true -Dorg.apache.tomcat.util.http.ServerCookie.ALLOW_EQUALS_IN_VALUE=true -Dorg.apache.tomcat.util.http.ServerCookie.ALLOW_HTTP_SEPARATORS_IN_V0=true -Dproject.name=za-fcp-biz-card -Dhsf.http.enable=true -Dhsf.server.ip=10.139.107.1 -Xms1g -Xmx1g -XX:NewSize=512m -XX:MaxNewSize=512m -XX:PermSize=256m -XX:MaxPermSize=256m -XX:+UseConcMarkSweepGC -XX:CMSFullGCsBeforeCompaction=5 -XX:+UseCMSCompactAtFullCollection -XX:+CMSParallelRemarkEnabled -XX:+CMSPermGenSweepingEnabled -XX:+CMSClassUnloadingEnabled -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=70 -XX:+DisableExplicitGC -XX:+UseCompressedOops -XX:+DoEscapeAnalysis -XX:MaxTenuringTh
[5080@iZ233e93pgiZ ~]$ telnet localhost 5082
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Welcome to Pandora Console.
-------------------------------------------------------
|     Login Time     |     2017-01-22 18:01:54      |
-------------------------------------------------------
|      Host Ip       |        10.139.107.1          |
-------------------------------------------------------
|  Pandora Version   |          2.0.5.5.6           |
-------------------------------------------------------
|    Sar Version     |     hsf-2.1.1.2-restful      |
-------------------------------------------------------
|    Package Time    |     2017-01-22 14:01:56      |
-------------------------------------------------------
|    Project Name    |       za-fcp-biz-card        |
-------------------------------------------------------
```

- 查看java应用详细信息
- Pandora 端口
- telnet 进入pandora 容器

```
pandora>help
Welcome to Pandora Console!
```

通过help命令查看Pandora容器命令行用法

CMD	OPTIONS	ARGS	EXAMPLE	DESC
cd	[ ]	dir	cd hsf,cd .., cd ~	change current module
ls	[ ]	[ ]	ls	list all modules
v	[ ]	name	v hsf	get module version
find	[c]	classname	find c RPCProtocolService	query export class
use	[ ]	module	use hsf	set source classloader
current	[ ]	[ ]	current	current source classloader
source	[ ]	classname	source com.taobao.x.X	decompile class, use module first
view	[ ]	resource	view x.jar!/COPYRIGHT.txt	view resource, use module first

通过ls命令查看Pandora容器中有哪些插件启用

```
pandora>ls
Pandora Version: 2.0.5.5.6
Sar Version: hsf-2.1.1.2-restful
Total Modules: 14
```

PRIORITY	NAME	VERSION	STATE	DEPLOY_TIME
20	spas-sdk-client	1.1.4-SNAPSHOT	deployed	2017-01-22 14:03:41
50	monitor	1.0.1	deployed	2017-01-22 14:03:41
100	eagleeye-core	1.3.5.1	deployed	2017-01-22 14:03:41
150	diamond-client	3.7.1	deployed	2017-01-22 14:03:41
160	spas-sdk-service	1.1.9-SNAPSHOT	deployed	2017-01-22 14:03:41
300	config-client	1.6.6.5	deployed	2017-01-22 14:03:41
550	unitrouter	1.0.11	deployed	2017-01-22 14:03:41
600	notify-tr-client	1.8.19.26	deployed	2017-01-22 14:03:41
999	hsf-notify-client	1.8.19.8	deployed	2017-01-22 14:03:41
1000	hsf	2.1.1.2-restful	deployed	2017-01-22 14:03:41
1500	filesync-client	1.0.3	deployed	2017-01-22 14:03:41
2000	pandora.qos.service	2.0.2	deployed	2017-01-22 14:03:41
9999	alimonitor-jmonitor	1.0.7	deployed	2017-01-22 14:03:41
10000	tddl-client	5.1.18-1	undeploy	2017-01-22 14:03:41

发现HSF好舒服插件启用

查看Pandora容器中的
各个plugin的信息：
的服务状态和版本

查看HSF App中的
MetaData信息：
Provider&Subscriber
的服务状态
和版本 / 分组

```
pandora>cd hsf          进入hsf插件节点
hsf
hsf>help    通过help命令查看有哪些命令用法
Welcome to  HSF Pandora Console!
     cmd    [options]       [args]     example                          desc
     find   [serviceNamePattern]      find *ItemService*      查询指定某个服务的元素据信息
     chkremoting       查询HSF Remoting处理和接收到的请求数目

hsf>ls    尝试列出所有服务列表，包含服务提供方和调用方列表，以及详细Meta
As Provider side:
com.zhongan.fcp.biz.card.share.api.CreditBillApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditSignatureApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditRepayApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditUserAuthApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.WithdrawMsgNotifyApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditOrderApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditBiometricsApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditAccountApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.InnerCreditApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditPayApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditMessageApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditApplyApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.SynBankCardDataApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditLoanApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditBankAuthApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditUserInfoApi:1.0.0.xjk, status:true, group:HSF
com.zhongan.fcp.biz.card.share.api.CreditRefundApi:1.0.0.xjk, status:true, group:HSF

As Subscriber side:
com.zhongan.fcp.core.credit.api.QueryAcctApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.credit.api.SettlementFundLoanRetryApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.insurance.api.UnderwriteApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.bill.api.BillQueryApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.credit.api.WithdrawNotifyRecordApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.credit.api.AcctLoanRetryApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.credit.api.AcctBizApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.insurance.api.ClaimApi:1.0.0.xjk, group:HSF
com.zhongan.search.share.api.FcpAcctPayQueryApi:1.0.0, group:HSF
com.zhongan.account.api.CompanyAccountService:1.0.0, group:HSF
com.zhongan.fcp.core.config.api.ConfigAPI:1.0.1, group:HSF
com.zhongan.fcp.core.bill.api.BillPenaltyApi:1.0.0.xjk, group:HSF
com.zhongan.account.api.AccountAuthService:1.0.0, group:HSF
com.zhongan.pigeon.service.MessageSendService:1.0.0, group:HSF
com.zhongan.openapi.core.service.ExtapiService:1.0.0.tst, group:HSF
com.zhongan.fcp.core.credit.api.AcctApplyApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.config.api.MessageApi:1.0.1, group:HSF
com.zhongan.fcp.core.bill.api.PreTreatBillApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.bill.api.BillCreatedApi:1.0.0.xjk, group:HSF
com.zhongan.fcp.core.bill.api.RepayplanQueryApi:1.0.0.xjk, group:HSF
com.zhongan.contract.api.ContractSignBusinessService:1.0.0, group:HSF
com.zhongan.account.api.AccountQueryService:1.0.0, group:HSF
com.zhongan.account.api.PersonAccountService:1.0.0, group:HSF
com.zhongan.fcp.core.bill.api.BillOverdueApi:1.0.0.xjk, group:HSF
```

众安保险

Q&A

千 橙

金融技术部

众安保险

HSF

http://gitlab.alibaba-inc.com/middleware/hsf2-0

ConfigServer

http://gitlab.alibaba-inc.com/middleware/configserver