Pylon 2.0接入文档

1. 概述

- pylon是实现了Eleme-rpc协议(Pylon Json Rpc Protocol & Zeus Thrift Binary Rpc Protocol)的SOA框架。
- pylon支持应用作为客户端接入，作为服务器端接入，或同时作为客户端服务端接入。
- pylon客户端与服务端基于接口契约编程，服务端提供接口服务，客户端。
- pylon基于JDK8开发。
- pylon使用maven进行依赖管理，可以从公司中央仓库(http://maven.dev.elenet.me/nexus/)获取。
- pylon最新版本参考Pylon 2.0 RELEASE NOTE
- 示例代码

2. 基础配置

2.1. pom.xml

pylon的rpc协议会传递调用方法的参数列表时需要用到参数名称，所以编译接口契约jar包时需要添加参数–parameters，可以配置pom中build的plugin：

Maven Plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.2</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <compilerArgs>
            <arg>-parameters</arg>
        </compilerArgs>
    </configuration>
</plugin>
```

2.2. Eclipse

使用Eclipse进行本地调试时，需进行如下配置：

- Preferences -> Java -> Compiler
  - JDK Compiliance -> "Compiler compliance level"设置为1.8
  - Classfile Generation -> "Store information about method parameters (usable via reflection)"设置为勾选

2.3. IDEA

使用IDEA进行本地调试时，需进行如下配置：

- Preferences -> "Build, Execution, Deployment" -> Compiler -> "Java Compiler"
  - "Project bytecode version"设置为1.8
  - "Additional command line parameters"添加"-parameters"

3. 配置文件

3.1. 介绍

- 配置文件格式为json，由三个部分组成：

commonConf

- 通用配置
- 必须声明
- 字段说明

key	含义	类型	必选	备注
huskarUrl	Huskar API Url	String	否	不配置表示不连接Huskar API(本地调试模式)
huskarToken	Huskar API Token	String	否	本地调试模式下可以不配置；生产环境找 郭渝 申请 (需提供appid，邮件方式)；测试环境使用eyzZ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.sRxFS5OivgNh80IiH-bBB7n6-ITMt6QeP1sIRAiV2wc
traceUrl	Etrace Collector Url	String	否	不配置表示不发送Etrace信息
metricUrl	Statsd Server Url	String	否	不配置表示不发送Statsd信息

serverConf

- 服务端配置
- 服务端必须声明
- 字段说明

key	含义	类型	必选	备注
name	注册服务名	String	是	现有服务：暂时保持不变；新服务：必须和appId一致
protocol	协议	String	是	目前支持json
group	注册集群	String	是	\
port	业务端口	int	是	\
workerGroupSize	I/O处理线程池大小	int	否	默认值为二倍CPU核数
threadPoolSize	业务处理线程池大小	int	是	\
bufferQueueSize	缓冲队列大小	int	是	不建议小于10
exitDelayInMillis	退出等待毫秒数	long	否	默认值为2000，作用于初始化失败或收到SIGINT时的进程退出
validatable	是否开启参数校验	boolean	否	默认值为false
initializer	IServiceInitializer实现类类名	String	是	\
interfaces	提供服务的接口名列表	List<String>	是	\

clientConfs

- 客户端配置
- 有依赖服务则需要声明
- 为clientConf数组
- 字段说明

key	含义	类型	必选	备注
name	依赖服务名	String	是	由服务方提供
protocol	协议	String	是	目前支持json / thrift
group	依赖集群	String	是	由服务方提供
threadPoolSize	线程池大小	int	是	\

timeoutInMillis	超时毫秒数	int	是	\
lbStrategy	负载均衡策略	String	否	目前支持round-robin(简单轮询)；默认值为round-robin
tpStrategy	线程池策略	String	否	目前支持semaphore(线程池满时抛出异常) / queue(线程池满时排队等待执行
interfaces	调用的接口名列表	List<String>	是	\
providerList	调用的Url列表	List<String>	否	仅在本地调试模式下生效；Url格式为hostname:port / ip:port

3.2. 示例

3.2.1. 本地调试模式

Configure.json

```
{
  "commonConf": {},
  "serverConf": {
    "name": "me.ele.pylon.serviceA",
    "protocol": "json",
    "group": "stable",
    "port": 8888,
    "threadPoolSize": 10,
    "bufferQueueSize": 10,
    "initializer": "me.ele.pylon.AServiceInitializer",
    "interfaces": [
      "me.ele.pylon.service.AService"
    ]
  },
  "clientConfs": [{
    "name": "me.ele.pylon.serviceB",
    "protocol": "json",
    "group": "stable",
    "threadPoolSize": 10,
    "timeoutInMillis": 1000,
    "interfaces": [
      "me.ele.pylon.service.BService"
    ],
    "providerList": ["vpca.B.service-1.elene.me:5555"]
  }, {
    "name": "me.ele.pylon.serviceC",
    "protocol": "thrift",
    "group": "stable",
    "threadPoolSize": 15,
    "timeoutInMillis": 1500,
    "interfaces": [
      "me.ele.pylon.service.CService"
    ],
    "providerList": ["vpca.C.service-1.elene.me:6666"]
  }]
}
```

3.2.2. 发布系统模板

Configure.json.etpl

```
{
  "commonConf": {
    "huskarUrl": "{{_ .ELE_HUSKAR_URL}}",
    "huskarToken": "{{_ .APP_HUSKAR_TOKEN}}",
    "traceUrl": "{{_ .ELE_TRACE_URL}}",
    "metricUrl": "{{_ .ELE_STATSD_URL ""}}"
  },
  "serverConf": {
    "name": "me.ele.pylon.serviceA",
    "protocol": "json",
    "group": "{{_ .SERVER_GROUP}}",
    "port": {{_ .SERVER_PORT}},
    "threadPoolSize": {{_ .SERVER_THREAD_POOL_SIZE}},
    "bufferQueueSize": {{_ .SERVER_BUFFER_QUEUE_SIZE}},
    "initializer": "me.ele.pylon.AServiceInitializer",
    "interfaces": [
      "me.ele.pylon.service.AService"
    ]
  },
  "clientConfs": [{
    "name": "me.ele.pylon.serviceB",
    "protocol": "json",
    "group": "{{_ .B_GROUP}}",
    "threadPoolSize": {{_ .B_THREAD_POOL_SIZE}},
    "timeoutInMillis": {{_ .B_TIMEOUT_IN_MILLIS}},
    "interfaces": [
      "me.ele.pylon.service.BService"
    ]
  }, {
    "name": "me.ele.pylon.serviceC",
    "protocol": "thrift",
    "group": "{{_ .C_GROUP}}",
    "threadPoolSize": {{_ .C_THREAD_POOL_SIZE}},
    "timeoutInMillis": {{_ .C_TIMEOUT_IN_MILLIS}},
    "interfaces": [
      "me.ele.pylon.service.CService"
    ]
  }]
}
```

3.3. 指定配置文件

- main方法参数传入文件路径
- 默认值为"conf/Configure.json"

4. 服务端

4.1. 配置

- 配置中应含有commonConf和serverConf。

4.2. Maven依赖

接口定义依赖

```
<dependency>
 <groupId>eleme-jarch</groupId>
 <artifactId>pylon-contract</artifactId>
 <version>${pylon.version}</version>
</dependency>
```

服务依赖

```
<dependency>
 <groupId>eleme-jarch</groupId>
 <artifactId>pylon-core</artifactId>
 <version>${pylon.version}</version>
</dependency>
```

4.3. 定义服务接口

Interface

```
public interface SampleService{
 String sayHello(String name) throws ServiceException, ServerException;

 default String sayHi(String name) throws ServiceException,
ServerException {
     return "hi," + name;
 }
}
```

约束

- 接口不允许继承其它接口
- 接口不允许为范型接口
- 接口方法不允许重载
- 接口方法不允许为范型方法
- 默认实现(default method)会作为fallback被调用
- Java Bean的编写规范请参考Pylon Java Bean编码规则
- 关于异常
  - 支持抛出以下异常(或其子类):
    - me.ele.contract.exception.ServiceException
      - 受检异常
      - 表示业务相关的异常(如用户不存在，红包已过期等)

        - 打印WARN级别log
        - 不计入熔断统计
    - me.ele.contract.exception.ServerException
        - 受检异常
        - 表示服务内部的异常(如数据库连接超时，redis服务不可用等)
        - 打印ERROR级别log
        - 计入熔断统计
    - java.lang.RuntimeException
        - 非受检异常
        - 打印ERROR级别log
        - 计入熔断统计。
- 允许继承异常，同时可以扩展字段(提供public getter/setter)
- 异常类必须提供参数列表为(String message)的构造器
- 框架内部的异常(Pylon 内置Rpc异常)均为非受检异常，无需显示声明抛出

4.4. 实现服务接口

Implementor

```
public class SampleServiceImpl implements SampleService {
  ...
}
```

4.5. 实现Pylon接口

ServiceInitializer

```
public class SampleServiceInitializer implements IServiceInitializer {
    @Override
    public void init() {
        ...
    }
    @Override
    public Object getImpl(Class<?> iface) {
        ...
    }

    @Override
    public IServiceChecker getChecker() {
        ...
    }

    @Override
    public IServiceDumper getDumper() {
        ...
    }

    @Override
    public IAuthStrategy getAuthStrategy() {
        ...
    }
}
```

实现me.ele.contract.iface.IServiceInitializer接口

- init
  - 初始化方法
  - 在其它几个方法调用前先被调用
  - 可不重写
- getImpl
  - 返回指定接口对应的实现实例
  - 每个声明的接口只会获取一个实例
  - 必须重写
- getChecker 方法
  - 返回me.ele.contract.iface.IServiceChecker的一个实现实例
  - 参考Pylon 内置Server使用说明#1.业务可用性检查(心跳)
  - 可不重写
- getDumper 方法
  - 返回me.ele.contract.iface.IServiceDumper的一个实现实例
  - 参考Pylon 内置Server使用说明#2.业务内部状态查看
  - 可不重写
- getAuthStrategy方法
  - 返回me.ele.contract.iface.IAuthStrategy的一个实现实例。
  - 参考Pylon 服务授权自定义策略
  - 可不重写

4.6. 启动/初始化

- 本质是通过me.ele.core.container.Container作为MainClass启动。
- 若main方法传入参数，则使用此参数作为配置文件路径；否则，使用默认路径conf/Configure.json
- 具体方案可以参考示例代码中的启动脚本。
- 可通过配置JVM参数-Dsoa.register.failure.ignore=true使得忽略注册失败

5. 客户端

5.1. 配置

- 配置中应含有commonConf和clientConfs。

5.2. Maven依赖

客户端依赖

```
<dependency>
 <groupId>eleme-jarch</groupId>
 <artifactId>pylon-core</artifactId>
 <version>${pylon.version}</version>
</dependency>
```

5.3. 获取接口定义

5.3.1. json协议

- 由服务方提供Maven dependency
- 将Maven dependency声明在pom中

5.3.2. thrift协议

- 由服务方提供thrift文件
- 在thrift文件中声明java的namespace(eg : namespace java me.ele.xxx)
- 在code-gen UI上用thrift文件和API列表来生成Java源码(Pylon Thrift2Java规则)
- 将Java源码放入项目中

5.4. 初始化

客户端初始化

```
//
ClientUtil.getContext().initClients("conf/Client.json");
// "conf/Configure.json"
ClientUtil.getContext().initClients();
```

5.5. 调用

ClientExample

```
//
SampleService client =
ClientUtil.getContext().getClient(SampleService.class);
//
System.out.println(client.sayHello("Tom"));
System.out.println(client.sayHi("Jack"));
```

6. 服务端 + 客户端

6.1. 配置

- 配置中应含有commonConf，serverConf和clientConfs。

6.2. Maven依赖

- 同4.2.Maven依赖

6.3. 定义服务接口

- 同4.3.定义服务接口

6.4. 实现服务接口

- 同4.4.实现服务接口

6.5. 实现Pylon接口

- 同4.5.实现Pylon接口

6.6. 获取接口定义

- 同5.3.获取接口定义

6.7. 启动/初始化

- 同4.6.启动/初始化

6.8. 调用

- 同5.5.调用

7. 集成功能

7.1. 内置Etrace

- 参考Pylon 内置Etrace使用说明

7.2. 内置Huskar-client

- 参考Pylon 内置Huskar Client使用说明

7.3. 内置Statsd

- 参考Pylon 内置Statsd使用说明

7.4. Statsd降级

- 参考Pylon Statsd降级配置方法

7.5. 内置Elog

- 参考Pylon 内置Elog

7.6. Elog降级

- 参考Pylon Elog降级配置方法

7.7. 运行时状态

7.7.1. 服务端

- 参考Pylon 内置Server使用说明#3.框架内部状态查看

7.7.2. 客户端

- 参考Pylon 客户端框架内部状态

7.8. 运行时配置

- 参考Pylon 内置Server使用说明#4.运行时配置

7.9. 客户端健康检查

- 参考Pylon 客户端健康检查算法

7.10. 客户端熔断

- 参考Pylon 客户端熔断算法

7.11. 服务降级

- 参考Pylon 服务降级配置方法

7.12. 服务授权

- 参考Pylon 服务授权配置方法

7.13. 服务端参数校验

- 参考Pylon 服务端参数校验

7.14. 客户端异步调用

- 参考Pylon 客户端异步调用说明

7.15. 客户端调用超时

- 参考Pylon 客户端调用超时功能

7.16. 控制台功能

- 参考Pylon Console控制台

7.17. 服务授权自定义策略

- 参考Pylon 服务授权自定义策略

8. 测试

- Pylon支持两种测试方式：单元测试和本地调试。

8.1. 单元测试

- 用于服务端。如果依赖了其他服务，需要手动配置各依赖服务的providerList。
- 可以通过@ConfigurePath注解指定配置文件路径，默认为conf/Configure.json。
- 可以通过@TimeoutInMillis注解指定超时毫秒数，默认为30000ms。

8.1.1. Maven依赖

单元测试依赖

```
<dependency>
 <groupId>eleme-jarch</groupId>
 <artifactId>pylon-test</artifactId>
 <version>${pylon.version}</version>
</dependency>
```

8.1.2. 使用TestNG测试

- 需要显式依赖TestNG；继承AbstractTestNGServerTests。

Test with TestNG

```
@ConfigurePath("conf/Configure.json")
@TimeoutInMillis(1000)
public class SampleTestWithTestNG extends AbstractTestNGServerTests {
    @Test
    public void testImplMode() {
        ImplModeIface client =
ClientUtil.getContext().getClient(ImplModeIface.class);
        Assert.assertEquals(client.testString("sample"), "sample");
        Assert.assertEquals(client.testLong(3), Long.valueOf(3));
    }
}
```

8.1.3. 使用JUnit4测试

- 需要显式依赖Junit4；继承AbstractJunit4ServerTests。

Test with Junit4

```
@ConfigurePath("conf/Configure.json")
@TimeoutInMillis(1000)
public class SampleTestWithJunit4 extends AbstractJunit4ServerTests {
    @Test
    public void testImplMode() {
        ImplModeIface client =
ClientUtil.getContext().getClient(ImplModeIface.class);
        Assert.assertEquals(client.testString("sample"), "sample");
        Assert.assertEquals(client.testLong(3), Long.valueOf(3));
    }
}
```

8.2. 本地调试

- 开启本地调试模式需要做如下配置：
    - huskarUrl置为空

- clientConf需要配置providerList

9. 打包

- 强烈建议不要将应用打成with-dependencies的包(这样非常容易造成class冲突)。
- 推荐使用maven-assembly-plugin插件，配合dist.xml，conf目录存放配置文件，bin目录存放可执行脚本，lib目录存放所有依赖jar包。

Maven Plugin

```
<plugin>
   <artifactId>maven-assembly-plugin</artifactId>
   <version>2.5.4</version>
   <configuration>
      <appendAssemblyId>false</appendAssemblyId>
      <finalName>${appid}</finalName>
      <descriptors>
         <descriptor>${project.basedir}/dist.xml</descriptor>
      </descriptors>
   </configuration>
   <executions>
      <execution>
         <id>make-assembly</id>
         <phase>package</phase>
         <goals>
            <goal>single</goal>
         </goals>
      </execution>
   </executions>
</plugin>
```

dist.xml

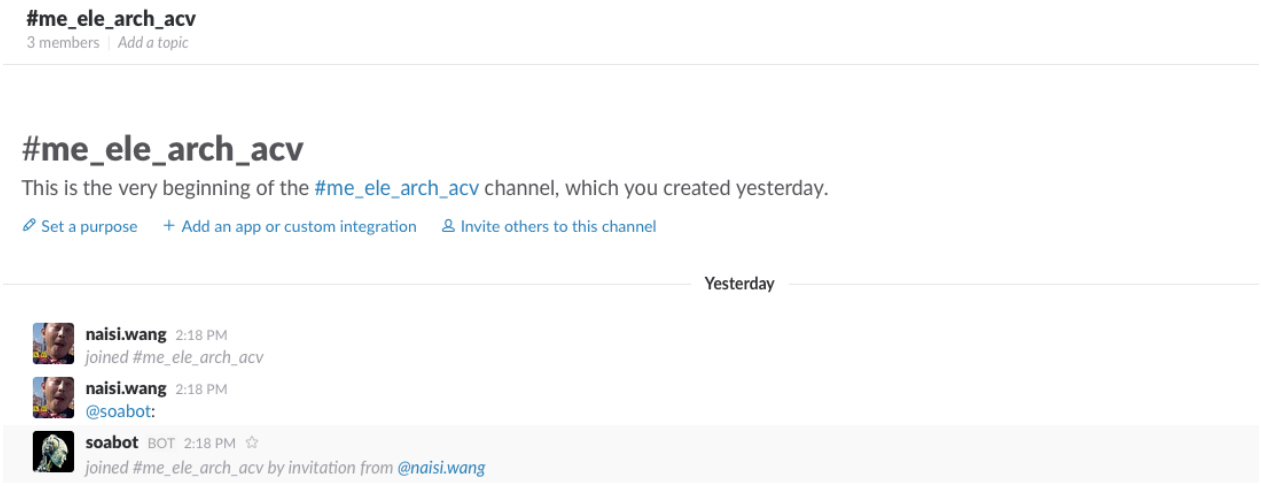
```
<assembly

xmlns="http://maven.apache.org/plugins/maven-assembly-plugin/assembly/1.
1.3"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-plugi
n/assembly/1.1.3 http://maven.apache.org/xsd/assembly-1.1.3.xsd">
   <id>package</id>
   <formats>
      <format>zip</format>
   </formats>
   <includeBaseDirectory>false</includeBaseDirectory>
   <fileSets>
      <fileSet>
         <directory>conf/</directory>
         <outputDirectory>conf/</outputDirectory>
      </fileSet>
      <fileSet>
         <directory>bin/</directory>
         <outputDirectory>bin/</outputDirectory>
         <fileMode>755</fileMode>
      </fileSet>
   </fileSets>
   <dependencySets>
      <dependencySet>
         <outputDirectory>/lib</outputDirectory>
         <fileMode>0444</fileMode>
      </dependencySet>
   </dependencySets>
</assembly>
```

10. 发布

- 示例代码包含打包发布的一切资源示例，建议clone到本地作为参考。
- 同时也请参考发布系统wiki。

11. 通知(日报/告警/监控)

- Pylon使用Slack的channel作为通知方式
- 应用对应的channel由其appId根据一定规则生成
  - .替换成_
  - 超过21个字符则截断至21个字符
  - 例如:
    - me.ele.arch.acv的channel是me_ele_arch_acv
    - napos.luna.gprinter.service的channel是napos_luna_gprinter_s
- 创建并加入channel后邀请soabot进入channel即可(如下图)



- 目前已经支持的功能：
  - 客户端/服务端性能日报
  - API兼容性告警(仅在发布到alpha环境时触发)
- 后续会开发其他通知功能