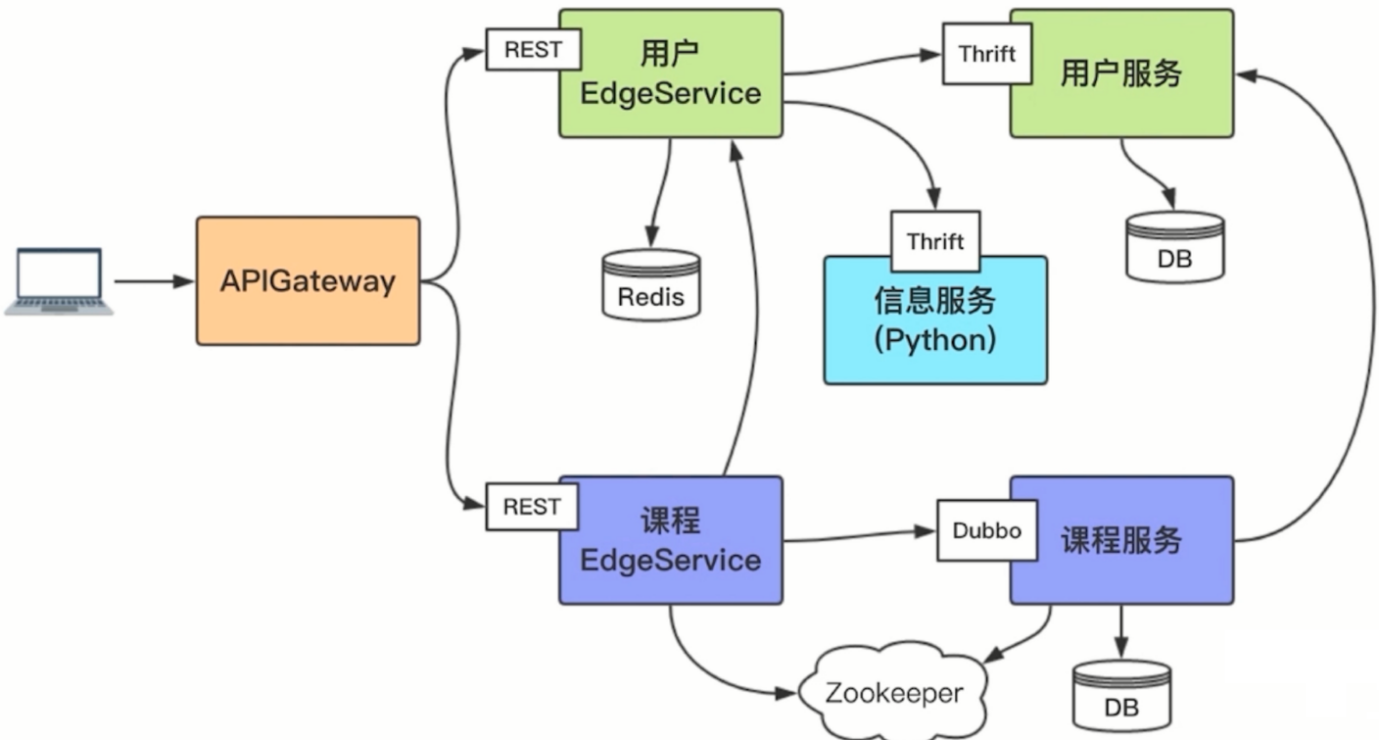
# Docker+Kubernetes(k8s)微服务容器化实践

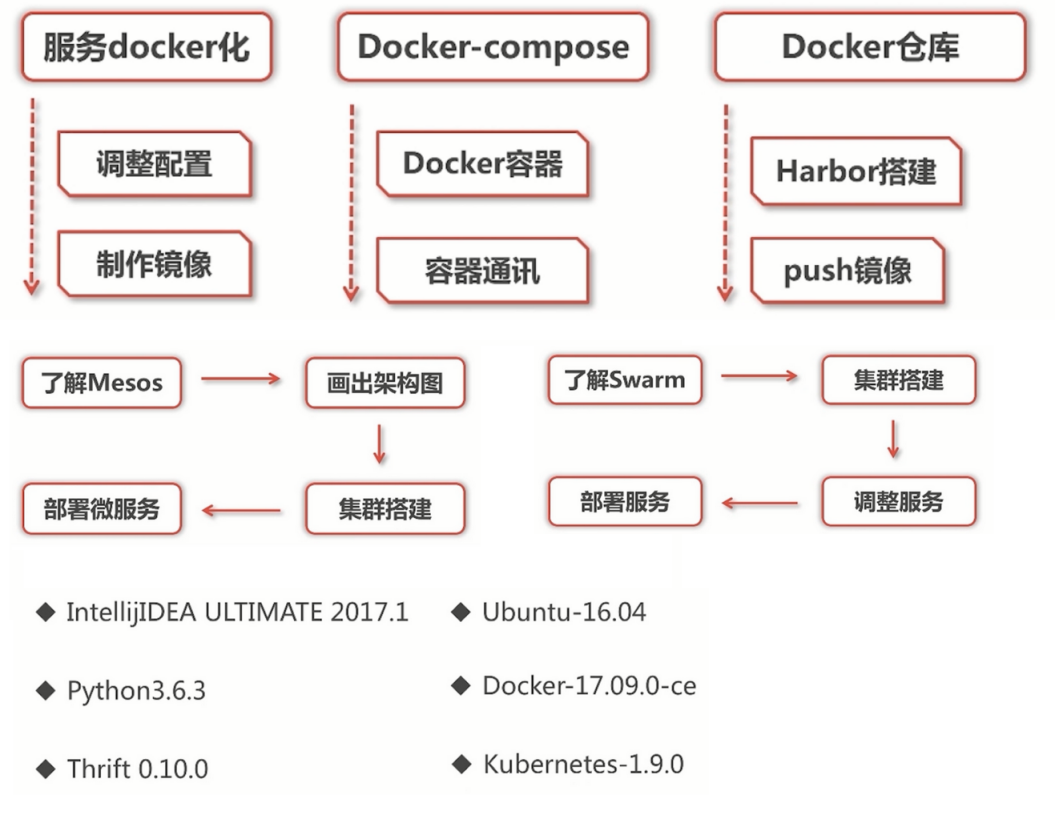
## 第一章 初识微服务

### 1.1概述

（1）Web应用整体框架

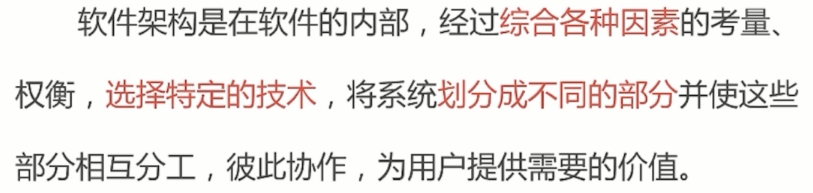


（2）容器化与DevOps工具

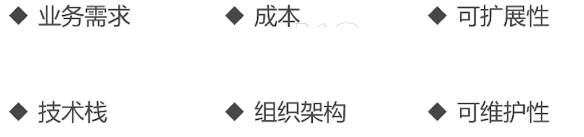


### 1.2软件架构的进化

·软件架构定义



·影响因素



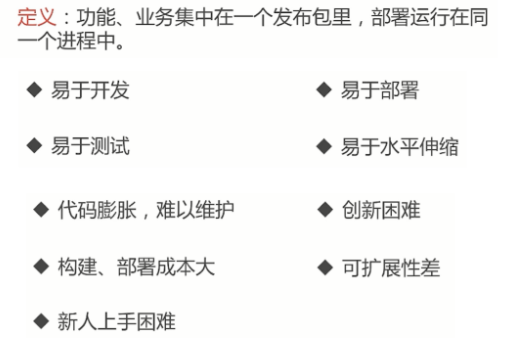
·软件架构进化

·一层架构

·MVC

·dubbo

·单体架构



### 1.3微服务

使用一套小服务来开发单个应用的方法，每个服务运行在独立的进程里，一般采用轻量级的通讯机制互联，并且他们可以通过自动化的方式部署。

·多微才算微

·代码量？

·开发时间

·不可度量

·微服务特征

·单一职责

·轻量级通讯

·隔离性

·有自己的数据

·技术多样性

（1）Grpc 使用HTTP2,序列化效率最高

（2）Dubbo使用的是python序列化

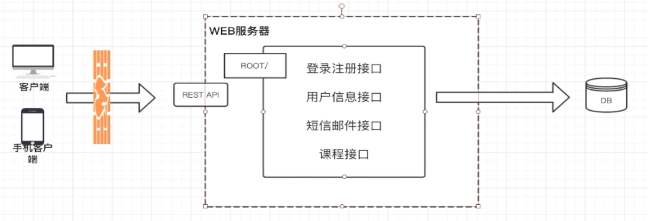
·微服务诞生背景

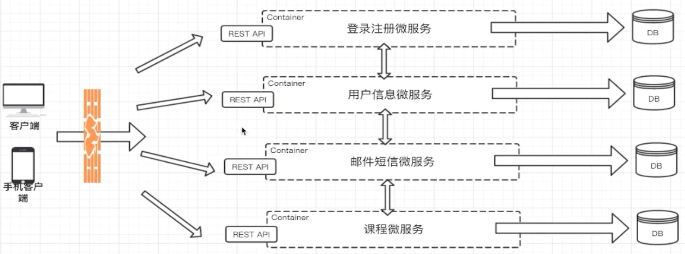
·互联网行业的快速发展

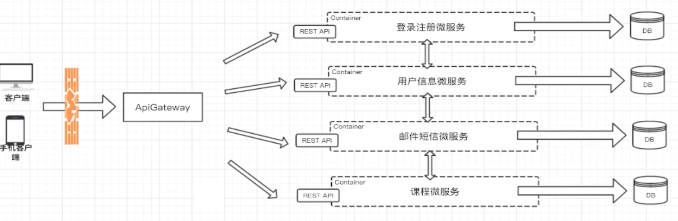
·敏捷开发，精益方法深入人心

·容器技术的成熟

### 1.4画出微服务架构图







### 1.5微服务架构的优势

·独立性

·敏捷性

·技术栈灵活

·高效团队

不足：

·额外的工作

·数据一致性

·沟通成本（接口变更）

## 微服务带来的问题及解决方案分析

### 2.1微服务结构带来的问题

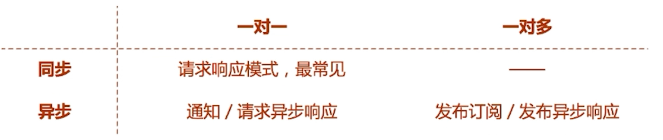
·微服务间如何通讯

·微服务如何发现彼此

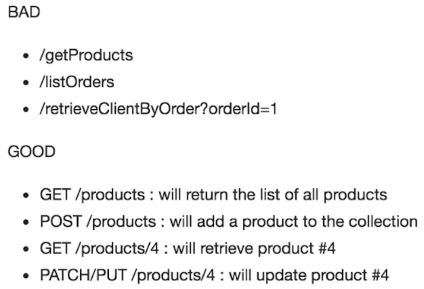
·微服务怎样部署？更新？扩容？

### 2.2微服务间如何通讯

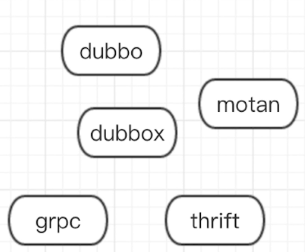
·从通讯模式角度考虑



·REST API



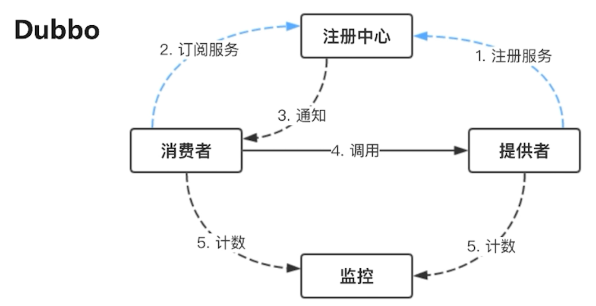
·RPC框架

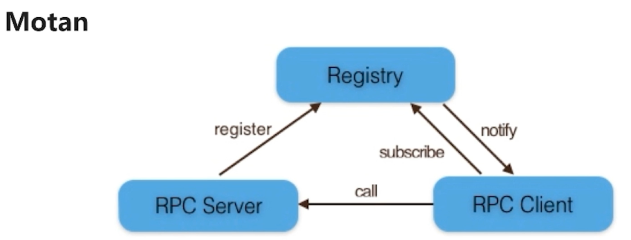


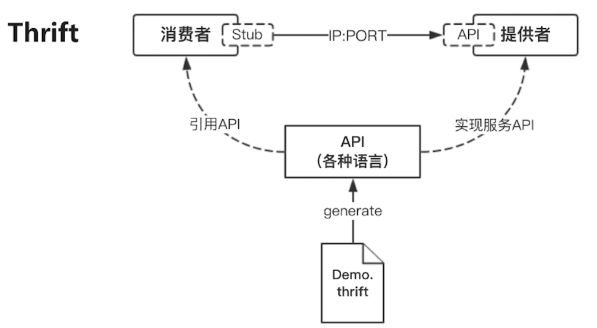
·MQ

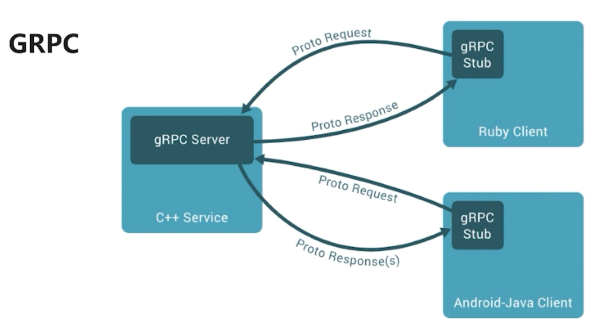
·如何选择RPC框架

1. I/O、线程调度模型
2. 序列化方式，二进制、JSON：影响RPC通讯效率
3. 多语言支持
4. 服务治理





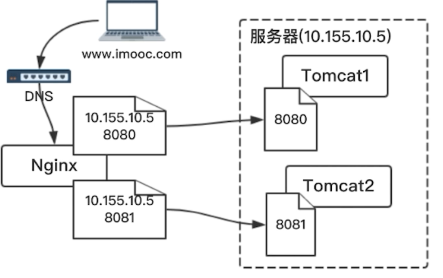




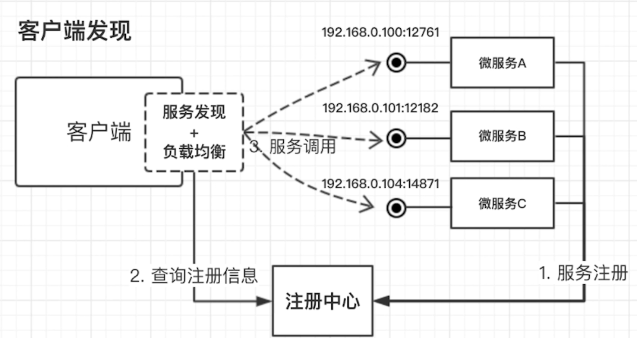


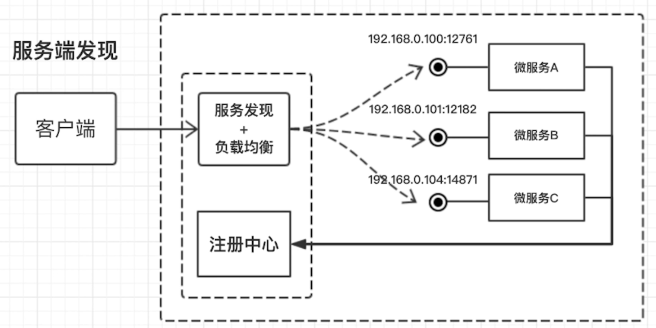
### 2.3服务发现

·传统服务VS微服务



·服务发现





·服务部署、更新和扩容

·服务编排

·流行的服务编排工具



### 2.4SpringBoot & SpringCloud

·SpringBoot与微服务

·SpringBoot核心功能

·独立运行 java -jar xxx.jar

·内嵌web服务器

·简化配置

·准生产的应用监控

·微服务SpringCloud

·统一配置管理

·服务注册与发现

·服务间调用

·负载均衡

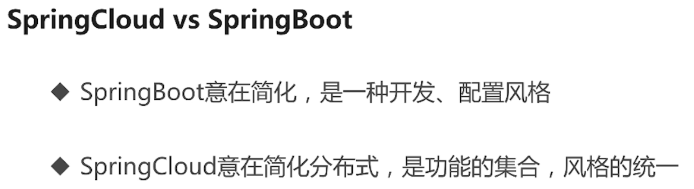
·分布式锁

·分布式Session

·深入理解

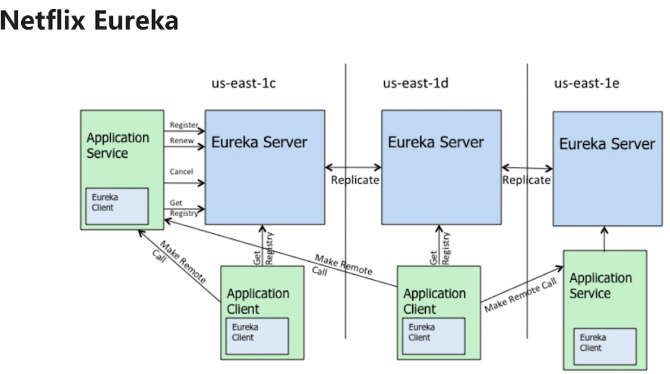
·一系列框架

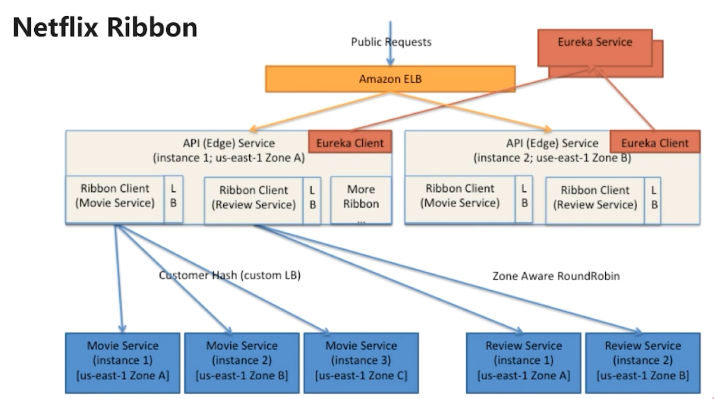
·简化java的分布式开发



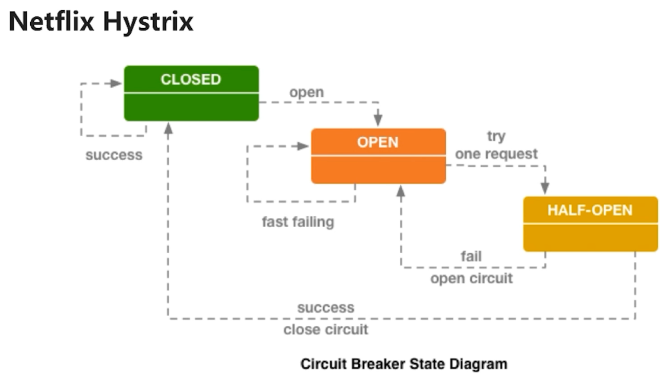
·SpringCloud核心组件

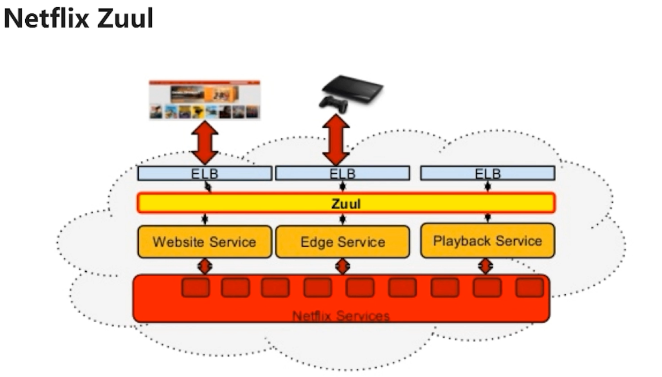


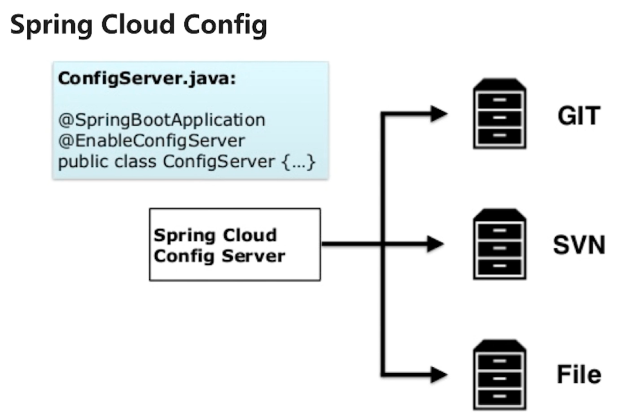




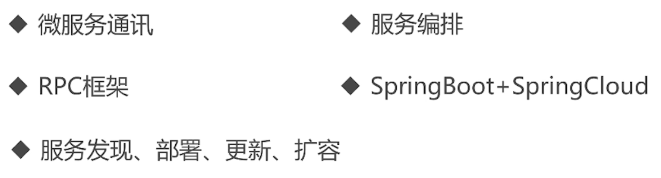
·Edge Service：Web RESTAPI







### 2.5小节



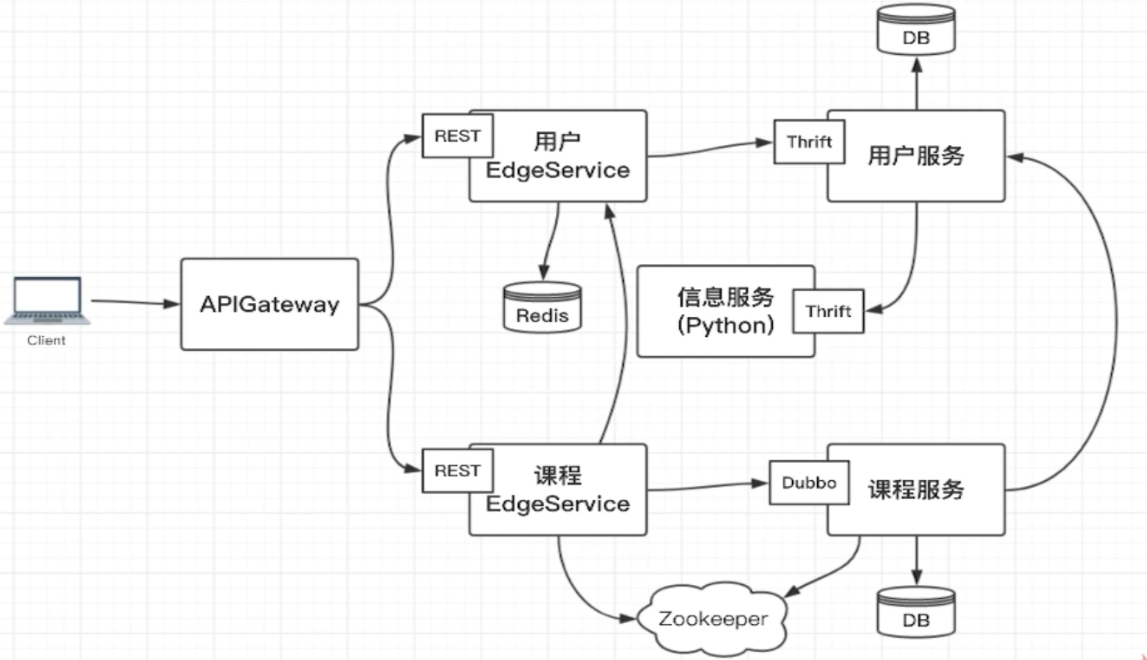
## 微服务开发

### 3.1微服务业务分析

·业务场景

·用户可以注册和登录

·登录用户可以对课程进行CRUD



### 3.2Thirft安装和验证

|  |
| --- |
| namespace java com.byf.thrift.demo  namespace py thrift.demo  service DemoService {  void sayHello(1:string name);  } |

|  |
| --- |
| C:\Users\BYF\Downloads\thrift-test>thrift-0.12.0.exe --gen java demo.thrift  C:\Users\BYF\Downloads\thrift-test>thrift-0.12.0.exe --gen py demo.thrift |

|  |
| --- |
| C:\Users\BYF\Downloads\thrift-test 的目录  2019/09/22 14:57 <DIR> .  2019/09/22 14:57 <DIR> ..  2019/09/22 14:56 120 demo.thrift  2019/09/22 14:56 <DIR> gen-java  2019/09/22 14:57 <DIR> gen-py  2019/09/22 14:52 3,887,104 thrift-0.12.0.exe |

### 3.3Python开发信息服务

|  |
| --- |
| namespace java com.byf.thrift.message namespace py message.api  service MessageService {  bool sendMobileMessage(1:string mobile, 2:string message);  bool sendEmailMessage(1:string email, 2:string message); } |

|  |
| --- |
| G:\MicroService\microservice\message-thrift-python-service\thrift>thrift-0.12.0.exe --gen py -out ../ message.t  hrift |

idea installed package failed目录缺少packaging\_tool.py

Intellij IDEA安装SDKs失败：

将目录F:\Program Files\JetBrains\IntelliJ IDEA 2019.1.1\plugins\python\helpers

拷贝至Intellij IDEA配置目录：C:\Users\BYF\.IntelliJIdea2019.1\config

|  |
| --- |
| **if** \_\_name\_\_ == **'\_\_main\_\_'**:  handler = MessageServiceHandler()  processor = MessageService.Processor(handler)  transport = TSocket.TServerSocket(**None**, **"9090"**)  tfactory = TTransport.TFramedTransportFactory()  pfactory = TBinaryProtocol.TBinaryProtocolFactory()   server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)  **print** (**"python thrift server start"**)  server.serve()  **print** (**"python thrift server exit"**) |
| org.apache.thrift.transport.TTransportException: java.net.ConnectException: Connection refused: connect  客户端连接时报错，修改None为具体IP地址 |
| **if** \_\_name\_\_ == **'\_\_main\_\_'**:  handler = MessageServiceHandler()  processor = MessageService.Processor(handler)  transport = TSocket.TServerSocket(**"127.0.0.1"**, **"9090"**)  tfactory = TTransport.TFramedTransportFactory()  pfactory = TBinaryProtocol.TBinaryProtocolFactory()   server = TServer.TSimpleServer(processor, transport, tfactory, pfactory)  **print** (**"python thrift server start"**)  server.serve()  **print** (**"python thrift server exit"**) |

### 3.4dubbo上手示例

<http://dubbo.apache.org/en-us/docs/user/quick-start.html>

|  |
| --- |
| <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  xmlns="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd"**>   *<!-- provider's application name, used for tracing dependency relationship -->* <**dubbo:application name="demo-provider"**>  <**dubbo:parameter key="qos.enable" value="true"**/>  <**dubbo:parameter key="qos.accept.foreign.ip" value="false"**/>  <**dubbo:parameter key="qos.port" value="33333"**/>  </**dubbo:application**>  *<!-- use multicast registry center to export service -->* <**dubbo:registry address="multicast://224.5.6.7:1234?unicast=false"**/>  *<!-- use dubbo protocol to export service on port 20880 -->* <**dubbo:protocol name="dubbo" port="20880"**/>  *<!-- service implementation, as same as regular local bean -->* <**bean id="demoService" class="org.apache.dubbo.demo.provider.DemoServiceImpl"**/>  *<!-- declare the service interface to be exported -->* <**dubbo:service interface="org.apache.dubbo.demo.DemoService" ref="demoService"**/> </**beans**> |

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  xmlns="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd"**>   *<!-- consumer's application name, used for tracing dependency relationship (not a matching criterion),  don't set it same as provider -->* <**dubbo:application name="demo-consumer"**>  <**dubbo:parameter key="qos.enable" value="true"** />  <**dubbo:parameter key="qos.accept.foreign.ip" value="false"** />  <**dubbo:parameter key="qos.port" value="33333"** />  </**dubbo:application**>  *<!-- use multicast registry center to discover service -->  <!--<dubbo:registry address="multicast://224.5.6.7:1234?unicast=false" check="false" />-->  <!-- <dubbo:registry valid="false" check="false" zookeeperProtocol="false" id="org.apache.dubbo.config.RegistryConfig" prefix="dubbo.registries." />-->  <!-- generate proxy for the remote service, then demoService can be used in the same way as the  local regular interface -->* <**dubbo:reference id="demoService" check="false" interface="org.apache.dubbo.demo.DemoService" url="127.0.0.1:20880"**/> </**beans**> |

-Ddubbo.application.qos.enable=true -Ddubbo.application.qos.port=33333 -Ddubbo.application.qos.accept.foreign.ip=false

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>dubbo-demo</**artifactId**>  <**groupId**>com.byf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-consumer</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>5.0.8.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.dubbo</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.7.2</**version**>  </**dependency**>  </**dependencies**>  </**project**> |

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  <**parent**>  <**artifactId**>dubbo-demo</**artifactId**>  <**groupId**>com.byf</**groupId**>  <**version**>1.0-SNAPSHOT</**version**>  </**parent**>  <**modelVersion**>4.0.0</**modelVersion**>   <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-provider</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>   <**dependencies**>  <**dependency**>  <**groupId**>com.byf</**groupId**>  <**artifactId**>dubbo-demo-api</**artifactId**>  <**version**>1.0-SNAPSHOT</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context</**artifactId**>  <**version**>5.0.8.RELEASE</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.apache.dubbo</**groupId**>  <**artifactId**>dubbo</**artifactId**>  <**version**>2.7.2</**version**>  </**dependency**>  </**dependencies**>  </**project**> |

|  |
| --- |
| *###set log levels###* **log4j.rootLogger**=**info, stdout** *###output to the console###* **log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender log4j.appender.stdout.Target**=**System.out log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern**=**[%d{dd/MM/yy hh:mm:ss:sss z}] %t %5p %c{2}: %m%n** |