Dubbo

凯盛软件

相关链接

凯盛软件

http://dubbo.io/

https://github.com/alibaba/dubbo

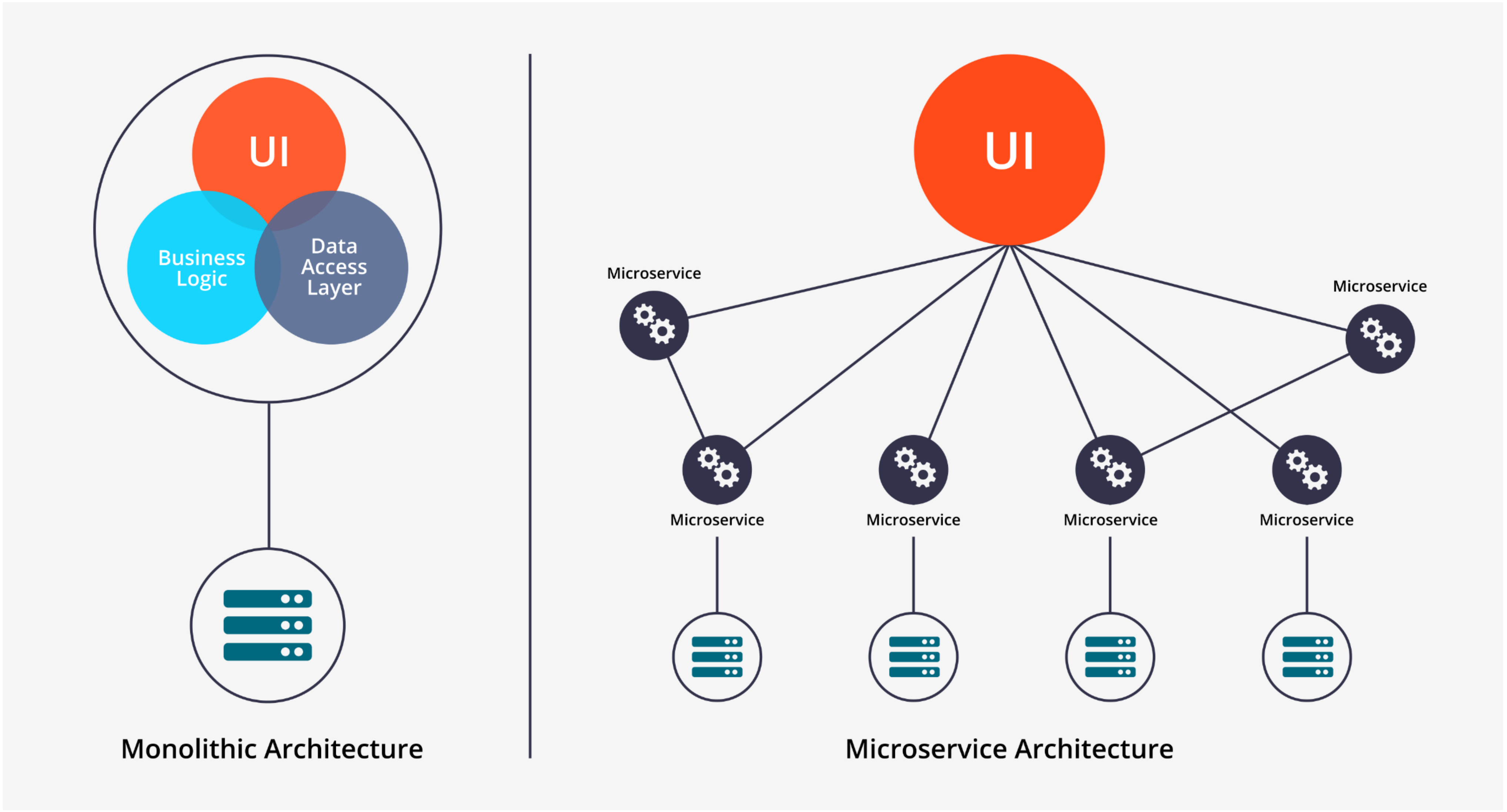
微服务

凯盛软件

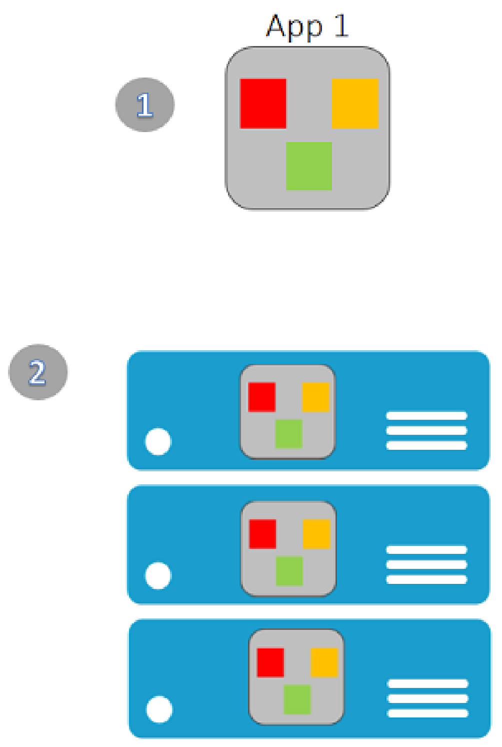
- 微服务是一种架构风格，一个大型复杂软件应用由一个或多个微服务组成。系统中的各个微服务可被独立部署，各个微服务之间是松耦合的。每个微服务仅关注于完成一件任务并很好地完成该任务。在所有情况下，每个任务代表着一个小的业务能力。
- 微服务的概念源于2014年3月Martin Fowler所写的一篇文章“Microservices”(http://martinfowler.com/articles/microservices.html)。
- 尽管“微服务”这种架构风格没有精确的定义，但其具有一些共同的特性，如围绕业务能力组织服务、自动化部署、智能端点、对语言及数据的“去集中化”控制等等。

单一应用架构 VS 分布式服务

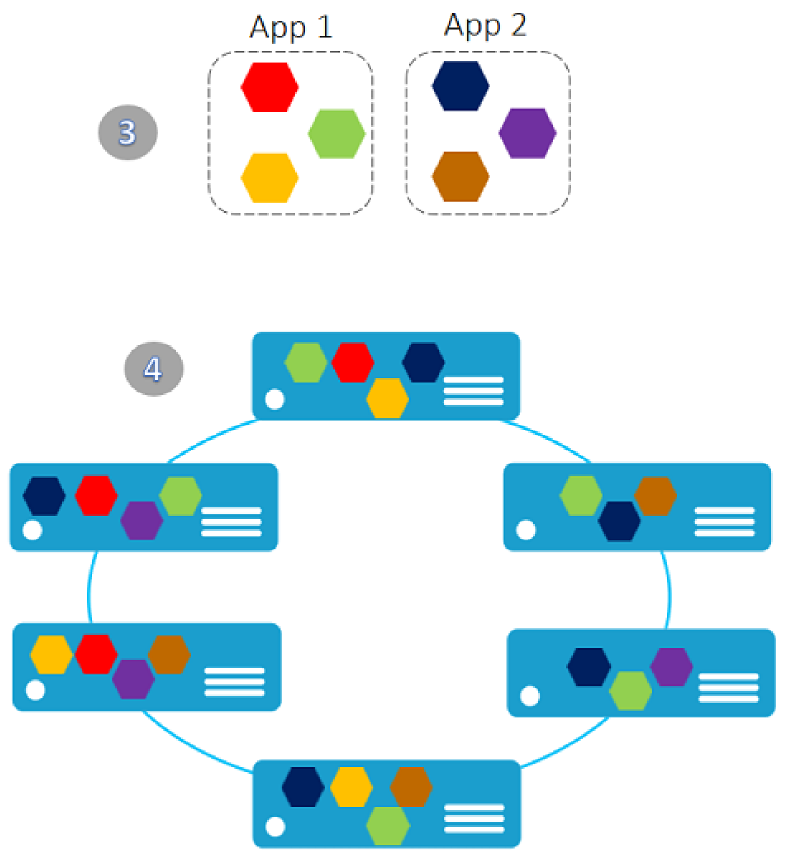
凯盛软件



Monolithic application approach

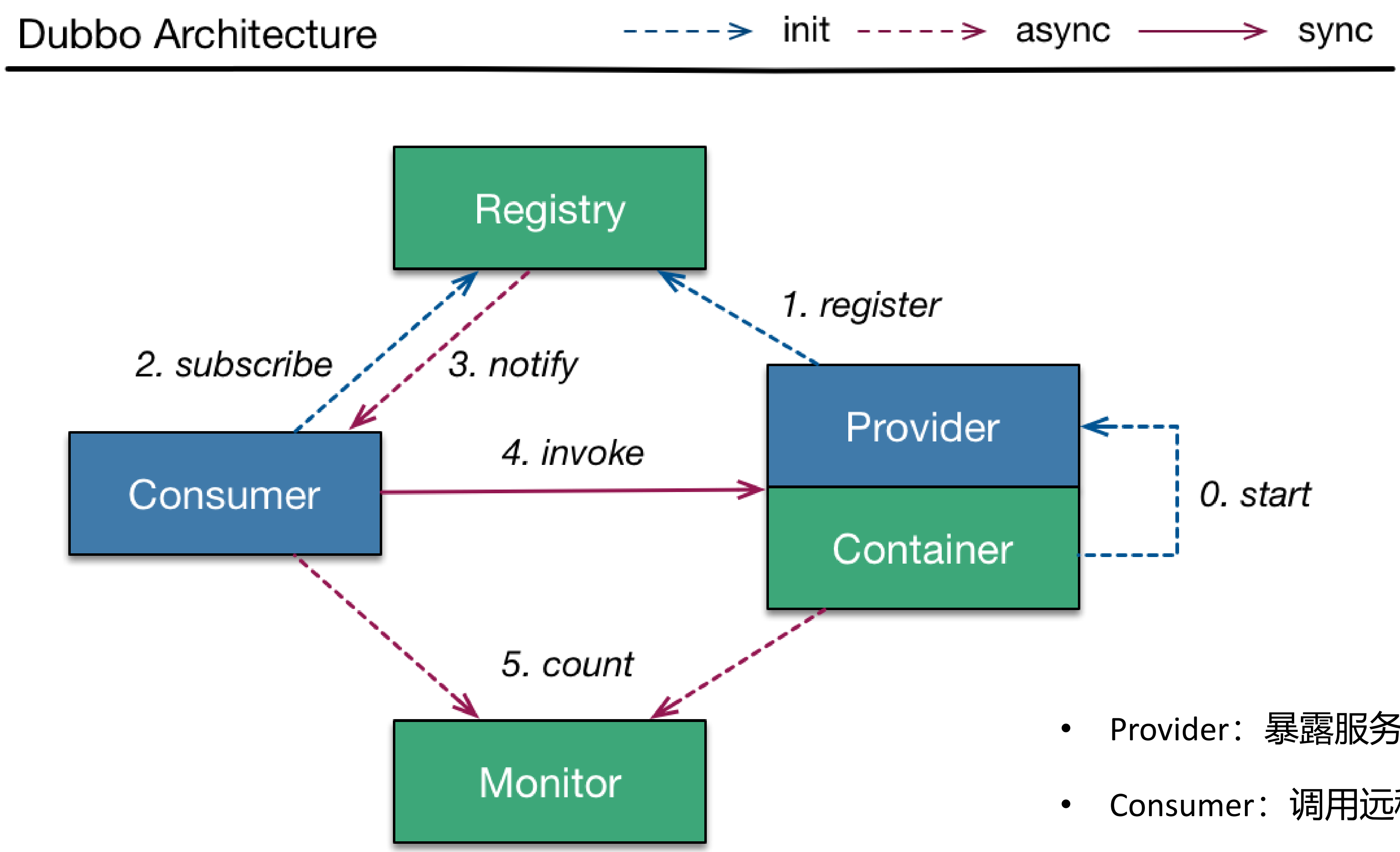


Microservices application approach



DUBBO是一个分布式服务框架，致力于提供高性能和透明化的RPC（Remote Procedure Call）远程服务调用方案，是阿里巴巴SOA服务化治理方案的核心框架，每天为2,000+个服务提供3,000,000,000+次访问量支持，并被广泛应用于阿里巴巴集团的各成员站点。

RPC（Remote Procedure Call）指远程过程调用，是一种通过网络调用远程过程（或方法）的协议。RPC是基于Client/Server模式，Client端携带必要参数调用Server端的方法，并获取Server端返回的方法执行结果。



- Provider：暴露服务的服务提供方
- Consumer：调用远程访问的服务消费方
- Registry：服务注册与发现的注册中心
- Monitor：统计服务的调用次数和调用时间的监控中心
- Container：服务运行容器

调用关系说明

1. 服务容器负责启动，加载，运行服务提供者。
2. 服务提供者在启动时，向注册中心注册自己提供的服务。
3. 服务消费者在启动时，向注册中心订阅自己所需的服务。
4. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。
5. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。
6. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

http://dubbo.io/books/dubbo-user-book/preface/architecture.html

使用步骤

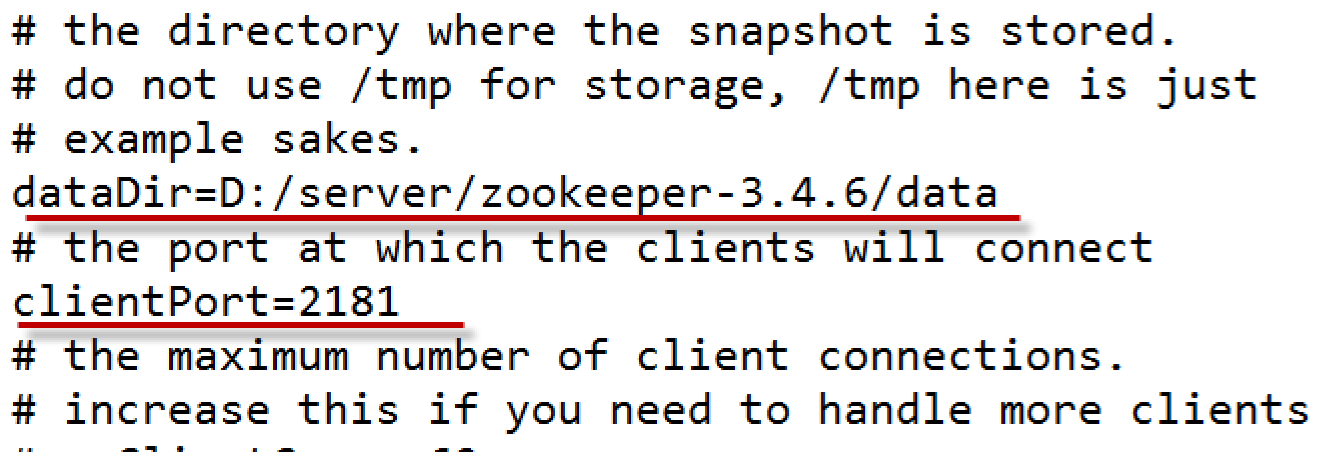
凯盛软件

1. 安装zookeeper（服务的注册中心）
2. 安装监控中心（可选）
3. 注册服务
4. 使用服务

安装zookeeper

凯盛软件

- 下载并解压 https://zookeeper.apache.org/
- 修改配置文件 conf/zoo.cfg



```
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=D:/server/zookeeper-3.4.6/data
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
```

- 双击bin/zkServer.cmd启动服务

项目中添加dubbo的依赖

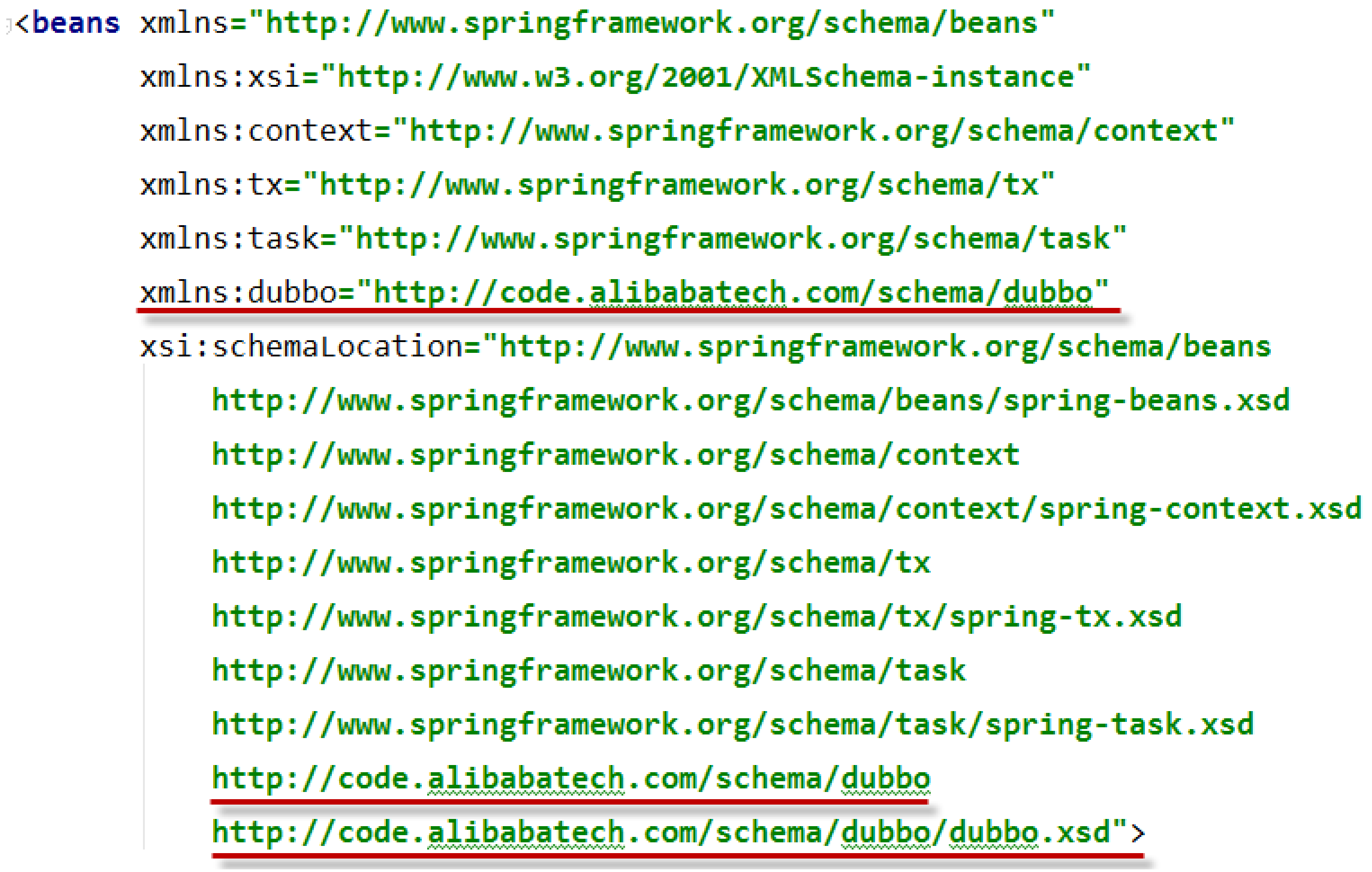
凯盛软件

```xml
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.3.13.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.3.13.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>dubbo</artifactId>
    <version>2.5.7</version>
    <exclusions>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

```
<!--zookeeper 及客户端-->
<dependency>
    <groupId>org.apache.zookeeper</groupId>
    <artifactId>zookeeper</artifactId>
    <version>3.3.3</version>
</dependency>
<dependency>
    <groupId>com.101tec</groupId>
    <artifactId>zkclient</artifactId>
    <version>0.10</version>
</dependency>
```

Spring的配置文件中添加XSD

凯盛软件



```xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:task="http://www.springframework.org/schema/task"
       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context.xsd
           http://www.springframework.org/schema/tx
           http://www.springframework.org/schema/tx/spring-tx.xsd
           http://www.springframework.org/schema/task
           http://www.springframework.org/schema/task/spring-task.xsd
           http://code.alibabatech.com/schema/dubbo
           http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
```

Provider 注册服务

凯盛软件

```xml
<!--服务名称，自定义-->
<dubbo:application name="ProductService"/>
<!--注册中心地址-->
<dubbo:registry address="zookeeper://192.168.1.112:2181"/>
<!--dubbo的协议和端口 添加host可以指定使用的网卡-->
<dubbo:protocol host="192.168.1.112" name="dubbo" port="20880"/>
<!--暴露服务-->
<bean id="productService" class="com.kaishengit.service.impl.ProductServcieImpl"/>
<dubbo:service interface="com.kaishengit.service.ProductServcie" ref="productService"/>
```

启动Spring容器

凯盛软件

```java
public static void main(String[] args) throws IOException {
    ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext("spring-dubbo-provider.xml");
    context.start();
    System.out.println("ProductService Provider start....");
    //防止退出
    System.in.read();
}
```

Consumer 消费服务

凯盛软件

```
<!--服务名称，自定义-->
<dubbo:application name="ProductServiceConsumer"/>
<!--dubbo的协议和端口-->
<dubbo:registry address="zookeeper://192.168.1.112:2181"/>
<!--接收消费服务-->
<dubbo:reference interface="com.kaishengit.service.ProductServcie" id="rpcProductService"/>
```

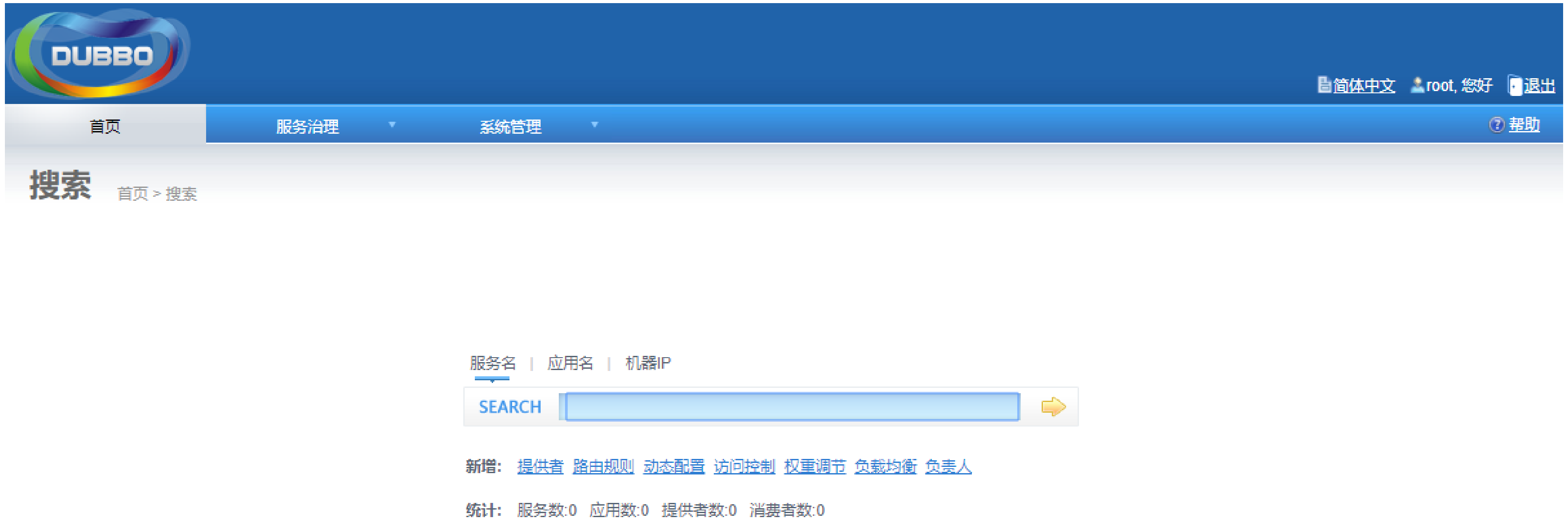
调用远程服务

凯盛软件

```
public static void main(String[] args) throws IOException {

    ApplicationContext context = new ClassPathXmlApplicationContext("spring-dubbo-consumer.xml");

    ProductServcie productServcie = (ProductServcie) context.getBean("rpcProductService");
    List<String> productNames = productServcie.findAllProductNames();
    for(String name : productNames) {
        System.out.println(name);
    }
    System.in.read();

}
```

Dubbo监控中心

凯盛软件



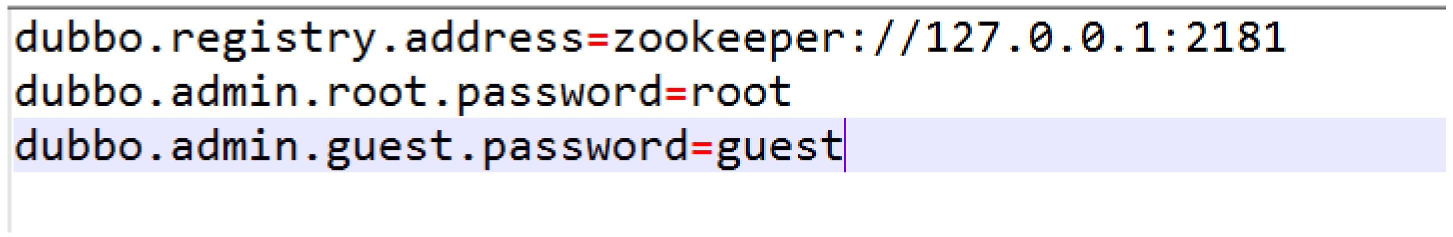
新增: 提供者 路由规则 动态配置 访问控制 权重调节 负载均衡 负责人

统计: 服务数:0 应用数:0 提供者数:0 消费者数:0

从github获取dubbo源代码，并编译

凯盛软件

- git clone https://github.com/alibaba/dubbo.git
- 修改duboo/dubbo-admin/src/main/webapp/WEB-INF/dubbo.properties文件

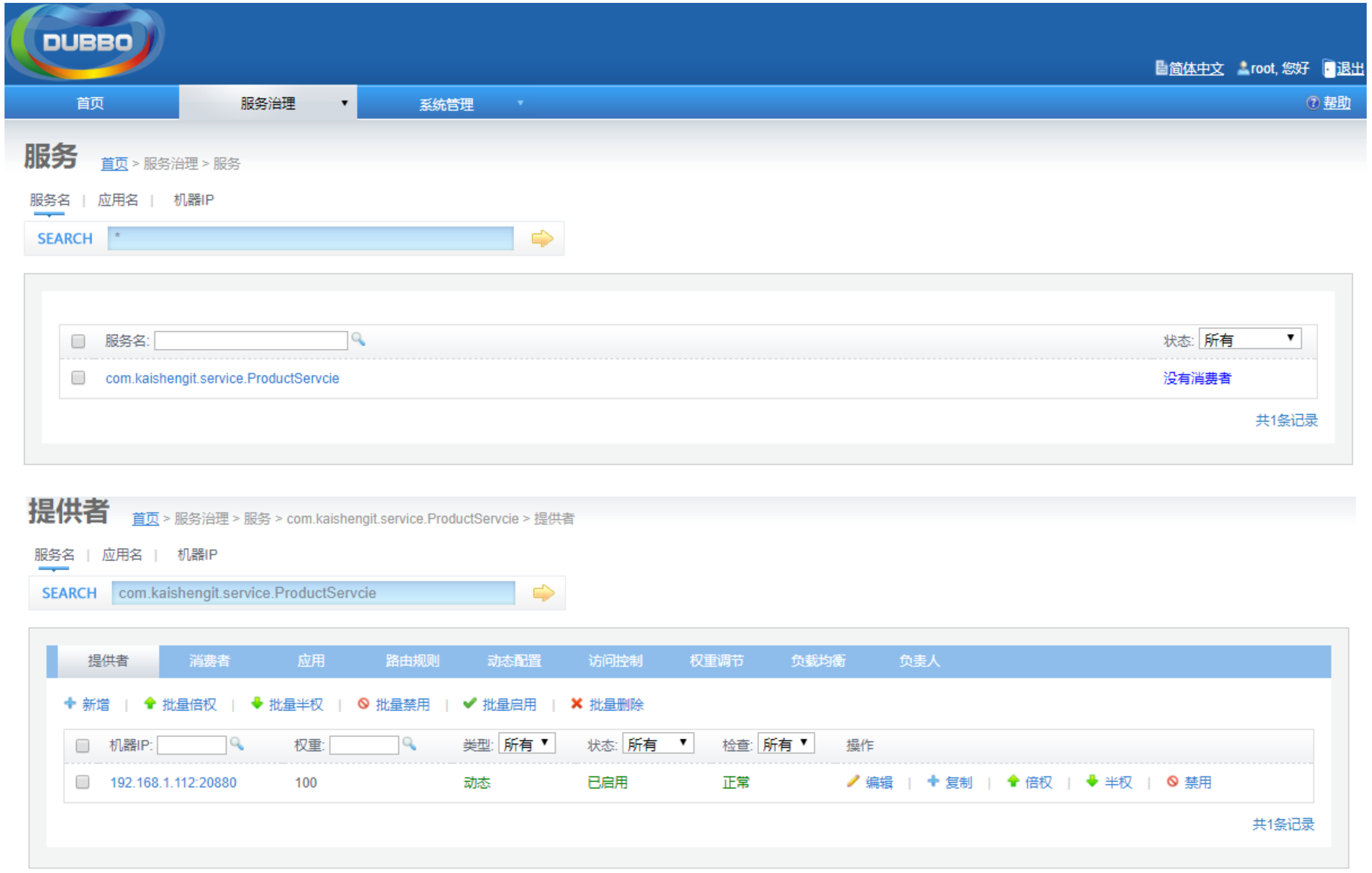


```
dubbo.registry.address=zookeeper://127.0.0.1:2181
dubbo.admin.root.password=root
dubbo.admin.guest.password=guest
```

- 在dubbo/dubbo-admin/中执行命令 mvn jetty:run
- 在地址栏中输入localhost:8080
- 账号为root密码也是root

服务治理

凯盛软件


DUBBO
简体中文 root, 您好 退出
首页 服务治理 系统管理 帮助

服务 首页 > 服务治理 > 服务

服务名 | 应用名 | 机器IP

SEARCH *

服务名: 状态: 所有

com.kaishengit.service.ProductServcie 没有消费者

共1条记录

提供者 首页 > 服务治理 > 服务 > com.kaishengit.service.ProductServcie > 提供者

服务名 | 应用名 | 机器IP

SEARCH com.kaishengit.service.ProductServcie

提供者 消费者 应用 路由规则 动态配置 访问控制 权重调节 负载均衡 负责人
新增 | 批量倍权 | 批量半权 | 批量禁用 | 批量启用 | 批量删除

机器IP:	权重:	类型: 所有	状态: 所有	检查: 所有	操作
192.168.1.112:20880	100	动态	已启用	正常	编辑 | 复制 | 倍权 | 半权 | 禁用

共1条记录

将服务打包成可运行的jar

凯盛软件

```
<build>
    <finalName>dubbo_privider</finalName>
    <resources>
        <!--打jar包时包含resources文件夹中的所有xml和properties文件-->
        <resource>
            <targetPath>${project.build.directory}/classes</targetPath>
            <directory>src/main/resources</directory>
            <includes>
                <include>**/*.xml</include>
                <include>**/*.properties</include>
            </includes>
        </resource>
        <!--
            http://dubbo.io/books/dubbo-user-book/demos/service-container.html
            Dubbo自带的Main方法会自动加载 META-INF/spring 目录下的所有 Spring 配置。
           当前是将src/main/resources中所有的xml文件拷贝到META-INF/spring文件夹中
       -->
        <resource>
            <targetPath>${project.build.directory}/classes/META-INF/spring</targetPath>
            <directory>src/main/resources/</directory>
            <filtering>true</filtering>
            <includes>
                <include>*.xml</include>
            </includes>
        </resource>
    </resources>
```

```xml
<plugins>
    <!-- 资源文件拷贝插件 -->
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-resources-plugin</artifactId>
        <version>2.7</version>
        <configuration>
            <encoding>UTF-8</encoding>
        </configuration>
    </plugin>
    <!-- java编译插件 -->
    <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>
            <encoding>UTF-8</encoding>
        </configuration>
    </plugin>
```

```xml
<!-- 打包jar文件时，配置manifest文件，加入lib包的jar依赖 -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>2.6</version>
    <configuration>
        <classesDirectory>target/classes/</classesDirectory>
        <archive>
            <manifest>
                <!-- 打包时 MANIFEST.MF文件不记录的时间戳版本 -->
                <useUniqueVersions>false</useUniqueVersions>
                <!-- 添加Class-Path -->
                <addClasspath>true</addClasspath>
                <!-- Class-Path添加前缀 -->
                <classpathPrefix>lib/</classpathPrefix>
                <!-- 指定Main-Class！！ -->
                <mainClass>com.alibaba.dubbo.container.Main</mainClass>
            </manifest>
            <manifestEntries>
                <Class-Path>.</Class-Path>
            </manifestEntries>
        </archive>
    </configuration>
</plugin>
```

```xml
<!-- 拷贝依赖的jar包到lib目录 -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <version>2.10</version>
            <executions>
                <execution>
                    <id>copy</id>
                    <phase>package</phase>
                    <goals>
                        <goal>copy-dependencies</goal>
                    </goals>
                    <configuration>
                        <outputDirectory>
                            <!-- 拷贝依赖到lib文件夹 -->
                            ${project.build.directory}/lib
                        </outputDirectory>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

其他配置

- 消费端直连提供者

```xml
<dubbo:reference interface="com.kaishengit.service.ProductServcie" id="rpcProductService"
    url="dubbo://192.168.1.112:20880"/>
```

- 启动容器不验证服务的存在性

```xml
<dubbo:reference interface="com.kaishengit.service.ProductServcie"
                 id="rpcProductService" check="false"/>
```

基于注解的服务注册

凯盛软件

```
@Configuration
@DubboComponentScan(basePackages = "com.kaishengit.service.impl") //服务实现类的扫描
public class Config {

    @Bean
    public ApplicationConfig applicationConfig() {
        ApplicationConfig applicationConfig = new ApplicationConfig();
        applicationConfig.setName("ProductService");
        return applicationConfig;
    }
    @Bean
    public ProtocolConfig protocolConfig() {
        ProtocolConfig protocolConfig = new ProtocolConfig();
        protocolConfig.setHost("192.168.1.112");
        protocolConfig.setPort(20880);
        protocolConfig.setName("dubbo");
        return protocolConfig;
    }
```

```java
@Bean
    public RegistryConfig registryConfig() {
        RegistryConfig registryConfig = new RegistryConfig();
        registryConfig.setAddress("zookeeper://127.0.0.1:2181");
        return registryConfig;
    }

}

@Service
@com.alibaba.dubbo.config.annotation.Service(timeout = 5000)
public class ProductServcieImpl implements ProductServcie {
          … …
}
```

```java
AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(Config.class);
context.start();
System.out.println("ProductService Provider start....");
//防止退出
System.in.read();
```

基于注解的服务消费

凯盛软件

```
@Configuration
@DubboComponentScan(basePackages = "com.kaishengit.service")
public class Config {

    @Bean
    public ApplicationConfig applicationConfig() {
        ApplicationConfig applicationConfig = new ApplicationConfig();
        applicationConfig.setName("ProductServiceConsumer");
        return applicationConfig;
    }

    @Bean
    public ConsumerConfig consumerConfig() {
        ConsumerConfig consumerConfig = new ConsumerConfig();
        consumerConfig.setTimeout(3000);
        return consumerConfig;
    }
    @Bean
    public RegistryConfig registryConfig() {
        RegistryConfig registryConfig = new RegistryConfig();
        registryConfig.setAddress("zookeeper://127.0.0.1:2181");
        return registryConfig;
    }

}
```

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = Config.class)
public class DubboTest {

    @com.alibaba.dubbo.config.annotation.Reference //注入所需的对象
    private ProductServcie productServcie;

    @Test
    public void findAll() {
        List<String> names = productServcie.findAllProductNames();
        for(String name : names) {
            System.out.println(name);
        }
    }
```

SpringBoot + Dubbo

凯盛软件

- 添加Maven依赖

```
<dependency>
    <groupId>io.dubbo.springboot</groupId>
    <artifactId>spring-boot-starter-dubbo</artifactId>
    <version>1.0.0</version>
</dependency>
```

https://github.com/dubbo/dubbo-spring-boot-project/tree/1.0.0

注册服务

凯盛软件

- 配置

```
spring.dubbo.application.name=UserProvider
spring.dubbo.registry.address=zookeeper://127.0.0.1:2181
spring.dubbo.protocol.host=192.168.1.112
spring.dubbo.protocol.name=dubbo
spring.dubbo.protocol.port=20880
#实现类所在的包
spring.dubbo.scan=com.kaishengit.dubboboot.service.impl
```

```java
@com.alibaba.dubbo.config.annotation.Service(timeout = 5000)
public class UserServiceImpl implements UserService {
    @Override
    public void sayHello(String name) {
        System.out.println(">>>>>>>>>>>>>>>>>>>>>>> hello, " + name);
    }
}

@SpringBootApplication
public class DubboBootApplication {

   public static void main(String[] args) {
      SpringApplication.run(DubboBootApplication.class, args);
        System.out.println("starting....");
   }
}
```

消费服务

凯盛软件

- 配置

```
spring.dubbo.application.name=UserConsumer
spring.dubbo.registry.address=zookeeper://127.0.0.1:2181
#使用服务的包
spring.dubbo.scan=com.kaishengit.dubboboot.controller
```

- 注入

```
package com.kaishengit.dubboboot.controller;

@Controller
public class UserController {

    @Reference
    private UserService userService;
```