Dubbo-Example

> *作者：Eagle*
>
> *此教程参考阿里巴巴Dubbo官方网站（http://dubbo.io）的例子进行编写*
>
> *代码在提交Github时已通过测试，运行无误*
>
>

1. 开发环境

> *正所谓工欲善其事，必先利其器。*

笔者的开发环境

1. 操作系统：OSX 10.10.5
2. 开发语言：Java
3. 构建工具：Maven
4. 注册中心：ZooKeeper

验证Java

在命令行运行 `java -version`:

```
eagledeMacBook-Pro:~ eagle$ java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
```

验证Maven

在命令行运行 `mvn -v`:

```
eagledeMacBook-Pro:~ eagle$ mvn -v
Apache Maven 3.2.5 (12a6b3acb947671f09b81f49094c53f426d8cea1; 2014-12-
15T01:29:23+08:00)
Maven home: /Users/eagle/ProgramTool/apache-maven-3.2.5
Java version: 1.7.0_80, vendor: Oracle Corporation
Java home:
/Library/Java/JavaVirtualMachines/jdk1.7.0_80.jdk/Contents/Home/jre
Default locale: zh_CN, platform encoding: UTF-8
OS name: "mac os x", version: "10.10.5", arch: "x86_64", family: "mac"
```

简单介绍ZooKeeper的安装及配置

`获取`

先到官网(http://zookeeper.apache.org)下载`stable`版本的`Release`包，发布此教程时，笔者使用的是3.4.6版本。

`安装`

将下载下来的包`zookeeper-3.4.6.tar.gz`解压，得到`zookeeper-3.4.6`文件夹之后，将其文件夹复制到个人喜好的位置。

`配置`（*此处在展示单机配置，关于集群配置可自行查阅文档*）

在与文件夹`zookeeper-3.4.6`同个目录下，创建三个文件夹，分别是`data`，`datalog`，`logs`：

```
eagledeMacBook-Pro:standalone eagle$ ls
data        datalog     logs        zookeeper-3.4.6
```

进入文件夹`zookeeper-3.4.6`下的conf目录，复制`zoo_sample.cfg`文件，并将其命名为`zoo.cfg`，然后打开`zoo.cfg`文件进行编辑:

```
默认使用的是`clientPort=2181`，此处暂不要对其修改，以免无法运行笔者提供的代码。
```

在`zoo.cfg`文件里添加两个配置，分别是dataDir和dataLogDir，并指向刚刚创建的data目录和datalog目录对应的路径，如:

```
dataDir=/Users/eagle/ProgramTool/zookeeper/standalone/data
dataLogDir=/Users/eagle/ProgramTool/zookeeper/standalone/datalog
```

`运行`

通过命令行cd到`zookeeper-3.4.6`目录下，运行`bin/zkServer.sh start`：

```
eagledeMacBook-Pro:zookeeper-3.4.6 eagle$ bin/zkServer.sh start
JMX enabled by default
Using config: /Users/eagle/ProgramTool/zookeeper/standalone/zookeeper-
3.4.6/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

测试

运行`bin/zkCli.sh -server 127.0.0.1:2181`进行登录，登录成功之后则显示内容大致如下:

```
WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: 127.0.0.1:2181(CONNECTED) 0]
```

运行一些简单的指令进行测试:

```
[zk: 127.0.0.1:2181(CONNECTED) 7] ls /
[zookeeper]
[zk: 127.0.0.1:2181(CONNECTED) 8] create /dubbo dubbo
Created /dubbo
[zk: 127.0.0.1:2181(CONNECTED) 9] ls /
[dubbo, zookeeper]
[zk: 127.0.0.1:2181(CONNECTED) 10] get /dubbo
dubbo
cZxid = 0x5a
ctime = Fri Oct 09 10:27:07 CST 2015
mZxid = 0x5a
mtime = Fri Oct 09 10:27:07 CST 2015
pZxid = 0x5a
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 5
numChildren = 0
```

当删除一个有子节点的节点时，会提示`Node not empty`:

```
[zk: 127.0.0.1:2181(CONNECTED) 2] get /dubbo
dubbo
cZxid = 0x2
ctime = Wed Sep 30 16:56:42 CST 2015
mZxid = 0x2
mtime = Wed Sep 30 16:56:42 CST 2015
pZxid = 0x6
cversion = 1
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 5
numChildren = 1
[zk: 127.0.0.1:2181(CONNECTED) 3] delete /dubbo
Node not empty: /dubbo
```

此时可以使用`rmr`指令:

```
[zk: 127.0.0.1:2181(CONNECTED) 5] rmr /dubbo
```

`关闭`

运行`bin/zkServer.sh stop`即可关闭:

```
eagledeMacBook-Pro:zookeeper-3.4.6 eagle$ bin/zkServer.sh stop
JMX enabled by default
Using config: /Users/eagle/ProgramTool/zookeeper/standalone/zookeeper-
3.4.6/bin/../conf/zoo.cfg
Stopping zookeeper ... STOPPED
```

2. 编码

> *说到底，还是要撸代码的.....*

这个例子需要两个项目，一个为Provider，一个为Consumer

Provider项目

使用Maven构建一个简单的J2SE项目

新建一个服务接口`DemoService.java`:

```
public interface DemoService {
    /**
    * 测试方法
    * @param name
    * @return
    */
    String sayHello(String name);
}
```

新建一个服务接口实现类`DemoServiceImpl`:

```
public class DemoServiceImpl implements DemoService {
    /**
    * 测试方法 - 实现
    * @param name
    * @return
    */
    public String sayHello(String name) {
        return "Hello " + name;
    }
}
```

通过Spring进行注入，并将其服务注册到ZooKeeper:

```
<!-- 和本地bean一样实现服务 -->
<bean id="demoService" class="cn.eaglefire.app.service.impl.DemoServiceImpl"
/>

<!-- 提供方应用信息，用于计算依赖关系 -->
<dubbo:application name="hello-world-app-provider"  />

<!-- 使用zookeeper广播注册中心暴露服务地址 -->
<dubbo:registry address="zookeeper://127.0.0.1:2181" />

<!-- 用dubbo协议在20880端口暴露服务 -->
<dubbo:protocol name="dubbo" port="20880" />

<!-- 声明需要暴露的服务接口 -->
<dubbo:service interface="cn.eaglefire.app.service.DemoService"
ref="demoService" />
```

Consumer项目

使用Maven构建一个简单的J2SE项目

新建一个服务接口`DemoService.java`:

```
public interface DemoService {
    /**
    * 测试方法
    * @param name
    * @return
    */
    String sayHello(String name);
}
```

通过Spring把注册到ZooKeeper的服务注入进来:

```
<!-- 使用zookeeper注册中心暴露服务地址 -->
<dubbo:registry address="zookeeper://127.0.0.1:2181" />

<!-- 生成远程服务代理，可以像使用本地bean一样使用demoService -->
<dubbo:reference id="demoService"
interface="cn.eaglefire.app.service.DemoService" />
```

项目之间的连接点

可以很明显的看到，在两个项目中都必须有服务接口`DemoService.java`，在实际开发应用中，这个接口将打包在一个jar包，并提供给Provider和Consumer两个项目使用，Provider对接口进行实现并注册到ZooKeeper，而Consumer在到ZooKeeper中寻找对应的接口实现。

3. 测试

> *不测试一下，怎么知道做得对不对*

Provider项目

编写个主方法并直接运行，运行之后将在ZooKeeper中`注册`接口实现:

```
public class Provider {
    /**
    * 主方法
    * @param args
    * @throws Exception
    */
    public static void main(String[] args) throws Exception {
        //
        System.out.println("Begin to load");
        // 加载Spring配置文件
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(new String[]
{"spring/ApplicationContext.xml"});
        context.start();
        //
        System.out.println("End to load");
        // 为保证服务一直开着，利用输入流的阻塞来模拟
        System.in.read();
    }
}
```

运行结果:

```
Connected to the target VM, address: '127.0.0.1:50305', transport: 'socket'
Begin to load
log4j:WARN No appenders could be found for logger
(org.springframework.core.env.StandardEnvironment).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for
more info.
End to load
```

Consumer项目

编写个主方法并直接运行，运行之后将在ZooKeeper中`寻找`接口实现:

```
public class Consumer {
    /**
    * 主方法
    * @param args
    * @throws Exception
    */
    public static void main(String[] args) throws Exception {
        //
        System.out.println("Begin to load");
        // 加载Spring配置文件
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext(new String[]
{"spring/ApplicationContext.xml"});
        context.start();
        //
        System.out.println("End to load");

        // 调用远程方法
        DemoService demoService =
(DemoService)context.getBean("demoService");
        String result = demoService.sayHello("Eagle");
        System.out.println("The result is: "+result);

        // 为保证服务一直开着，利用输入流的阻塞来模拟
        System.in.read();
    }
}
```

运行结果:

```
Connected to the target VM, address: '127.0.0.1:50309', transport: 'socket'
Begin to load
log4j:WARN No appenders could be found for logger
(org.springframework.core.env.StandardEnvironment).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for
more info.
End to load
The result is: Hello Eagle
```

4. 结语

> *本教程到此结束，欢迎指正，互相交流。*