# EOSPlatform+Dubbo+Zookeeper

# 最佳实践

随着分布式服务框架的流行，许多公司都引入了业界流行的分布式服务框架，本文介绍如何使用EOSPlatform+Dubbo+Zookeeper实现分布式服务的开发与运行。

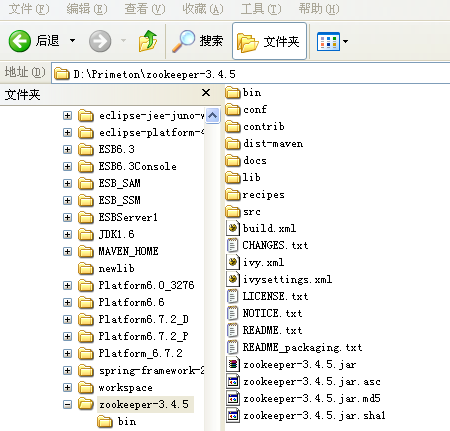
## 环境准备

## 环境选择

|  |  |
| --- | --- |
| EOSPlatform | 6.7.1 |
| Zookeeper | 3.4.5 |
| Dubbo | 2.5.3 |
| Tomcat | 5.5.27 |

## 搭建Zookeeper环境

下载zookeeper-3.4.5.tar.gz解压到自定义目录中，如下图：



打开conf目录，将文件zoo\_sample.cfg重命名为zoo.cfg。

打开bin目录，用UE打开zkServer.cmd修改如下代码。

|  |
| --- |
| 必须为JDK1.6 否则可能会报classNotfound异常  setlocal  call "%~dp0zkEnv.cmd"  set JAVA\_HOME=D:\Primeton\JDK1.6  set ZOOMAIN=org.apache.zookeeper.server.quorum.QuorumPeerMain  echo on  %JAVA\_HOME%\bin\java "-Dzookeeper.log.dir=%ZOO\_LOG\_DIR%" "-Dzookeeper.root.logger=%ZOO\_LOG4J\_PROP%" -cp "%CLASSPATH%" %ZOOMAIN% "%ZOOCFG%" %\*  使用指定的JAVAHOME，防止JDK版本问题  endlocal |

## 搭建服务监控平台环境

下载dubbo-admin-2.5.3.war，部署到Tomcat5.5.27的应用目录中，路径如下：\*\*\*\*\apache-tomcat-5.5.27\webapps\;

## 控制台启动运行

先启动zookeeper，再启动tomcat服务器。

通过浏览器访问dubbo-admin应用，用户名/密码（root/root）

如下所示：



内存与注册中心都运行正常说明环境准备完成。

## EOSPlatform扩展开发

## 修复EOS产品中Spring容器的BUG

com.primeton.spring.context.ExtendParentApplicationContext.java中有如下两个方法：

|  |
| --- |
| /\* (non-Javadoc)  \* @see org.springframework.beans.factory.ListableBeanFactory#getBeansOfType(java.lang.Class)  \*/  **public** Map getBeansOfType(Class type) **throws** BeansException {  **return** **new** HashMap();//防止空指针异常  }  /\* (non-Javadoc)  \* @see org.springframework.beans.factory.ListableBeanFactory#getBeansOfType(java.lang.Class, boolean, boolean)  \*/  **public** Map getBeansOfType(Class type, **boolean** includePrototypes,  **boolean** allowEagerInit) **throws** BeansException {  // **TODO** Auto-generated method stub  **return** **new** HashMap();//防止空指针异常  } |

黄色部分为修改内容，原先返回null值，会导致空指针异常。

## 增加启动时开启Spring容器

增加一个启动监听类：SpringContextStartupListener，内容如下：

|  |
| --- |
| /\*\*  \*  \* 由于分布式服务框架需要初始化Spring容器用于发布订阅服务，因此加入启动过程  \*  \* **@author** yujl (mailto:yujl@primeton.com)  \*/  **public** **class** SpringContextStartupListener **implements** IRuntimeListener {  /\* （非 Javadoc）  \* @see com.eos.runtime.core.IRuntimeListener#start(com.eos.runtime.core.RuntimeEvent)  \*/  **public** **void** start(RuntimeEvent arg0) {  ApplicationContextFactory.*getContext*();//预先初始化Spring容器  }  /\* （非 Javadoc）  \* @see com.eos.runtime.core.IRuntimeListener#stop(com.eos.runtime.core.RuntimeEvent)  \*/  **public** **void** stop(RuntimeEvent arg0) {  }  } |

将SpringContextStartupListener添加到handler-startup.xml文件的结尾部分，如下图所示：

|  |
| --- |
| <!-- 初始化Spring 容器-->  <handler handle-class=*"com.primeton.spring.context.runtime.SpringContextStartupListener"* />  </handlers> |

## 补充Zookeeper和Dubbo相关jar文件

|  |
| --- |
| netty-3.2.5.Final.jar |
| dubbo-2.5.3.jar |
| slf4j-api-1.6.1.jar |
| slf4j-log4j12-1.6.1.jar |
| zkclient-0.1.jar |
| zookeeper-3.4.5.jar |

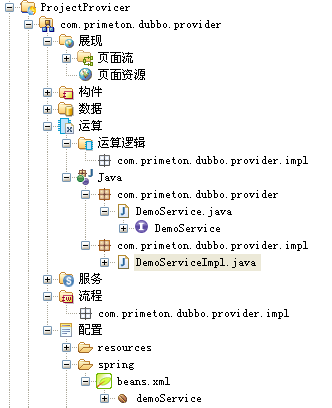
将上述jar包放入EOS应用的\WEB-INF\lib目录下。

PS：后面提到的服务提供者应用和服务消费者应用都需要添加Jar包和配置启动文件。

此时EOSPlatform扩展开发完毕。

## 服务提供者示例开发

新建一个EOS项目（项目名称为ProjectProvicer）对应应用为（default\_provider）；新建一个构件包（名称为：com.primeton.dubbo.provider）：



在构件包中新建一个接口示例和一个实现的示例如下：

接口类DemoService：

|  |
| --- |
| **package** com.primeton.dubbo.provider;  /\*\*  \*  \* 服务提供方接口  \*  \* **@author** yujl (mailto:yujl@primeton.com)  \*/  **public** **interface** DemoService {  /\*\*  \* 这是一个测试  \* **@param** input  \* **@return**  \*/  String testHelloWorld(String input);    } |

实现类DemoServiceImpl：

|  |
| --- |
| **package** com.primeton.dubbo.provider.impl;  **import** org.apache.commons.lang.StringUtils;  **import** com.primeton.dubbo.provider.DemoService;  /\*\*  \*  \* 服务提供方实现  \*  \* **@author** yujl (mailto:yujl@primeton.com)  \*/  **public** **class** DemoServiceImpl **implements** DemoService {  /\* （非 Javadoc）  \* @see com.primeton.dubbo.provider.DemoService#testHelloWorld(java.lang.String)  \*/  **public** String testHelloWorld(String input) {  **if**(StringUtils.*isBlank*(input)){  **return** "input must not null";  }  **return** "Hello " + input;  }  } |

SpringBean配置编码如下：

|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:sca=*"http://www.springframework.org/schema/sca"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  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/tx http://www.springframework.org/schema/tx/spring-tx.xsd*  *http://www.springframework.org/schema/sca http://www.osoa.org/xmlns/sca/1.0/spring-sca.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>  <!-- 提供方应用信息，用于计算依赖关系 -->  <dubbo:application name=*"hello-world-app"* />  <!-- 使用multicast广播注册中心暴露服务地址 先不使用注册中心-->  <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=*"com.primeton.dubbo.provider.DemoService"* ref=*"demoService"* />  <!-- 和本地bean一样实现服务 -->  <bean id=*"demoService"* class=*"com.primeton.dubbo.provider.impl.DemoServiceImpl"* />  </beans> |

到此，服务提供者就开发完成了，此时通过导出构件包并部署到Tomcat的default\_provider应用当中。

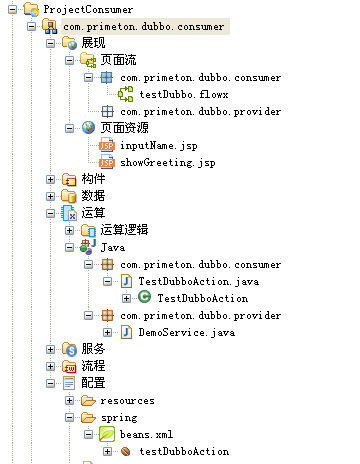
启动服务提供者所在的Tomcat，访问服务监控平台会看到如下界面：



如此， 服务提供者就将服务发布到Zookeeper注册中心当中管理起来了。

## 服务消费者示例开发

新建一个EOS项目（项目名称为ProjectConsumer）对应的应用名称为（default\_consumer）；新建一个构件包（名称为com.primeton.dubbo.consumer），如下图所示：



首先需要引入服务提供方接口DemoService；

然后增加一个SpringBean用于封装DemoService接口，代码如下：

|  |
| --- |
| **public** **class** TestDubboAction {  **private** DemoService demoServiceDL;    **public** String sayHello(String input){  **if**(demoServiceDL != **null**){  **return** demoServiceDL.testHelloWorld(input);  }  **return** "";  }  **public** DemoService getDemoServiceDL() {  **return** demoServiceDL;  }  **public** **void** setDemoServiceDL(DemoService demoServiceDL) {  **this**.demoServiceDL = demoServiceDL;  }  } |

此类不做任何工作，只是调用DemoService接口。

Spring配置文件修改如下：

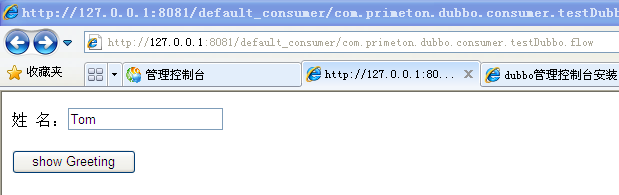
|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:sca=*"http://www.springframework.org/schema/sca"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd*  *http://www.springframework.org/schema/sca http://www.osoa.org/xmlns/sca/1.0/spring-sca.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd"*>  <!-- 消费方应用名，用于计算依赖关系，不是匹配条件，不要与提供方一样 -->  <dubbo:application name=*"consumer-of-helloworld-app"* />  <!-- 使用multicast广播注册中心暴露发现服务地址 先屏蔽注册中心，通过直连实现  <dubbo:registry address="multicast://224.5.6.7:1234" />-->  <dubbo:registry address=*"zookeeper://127.0.0.1:2181"* />  <!-- 生成远程服务代理，可以和本地bean一样使用demoService -->  <dubbo:reference id=*"demoService"* interface=*"com.primeton.dubbo.provider.DemoService"*/>    <bean id=*"testDubboAction"* class=*"com.primeton.dubbo.consumer.TestDubboAction"*> <!-- 和本地服务一样使用远程服务 -->  <property name=*"demoServiceDL"* ref=*"demoService"* />  </bean>  </beans> |

通过Spring配置，将远程服务demoService注入到TestDubboAction中。

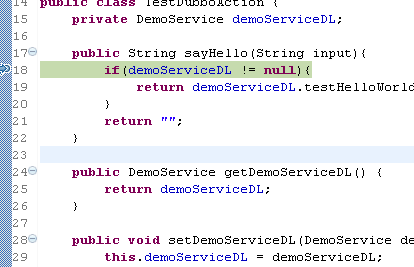
然后需要开发两个页面和一个页面流用于调用上述TestDubboAction，具体相关代码参考EOS帮助文档中Spring的配置与开发。

至此，服务消费者也开发完成；通过导出构件包并部署到Tomcat的default\_consumer应用当中。

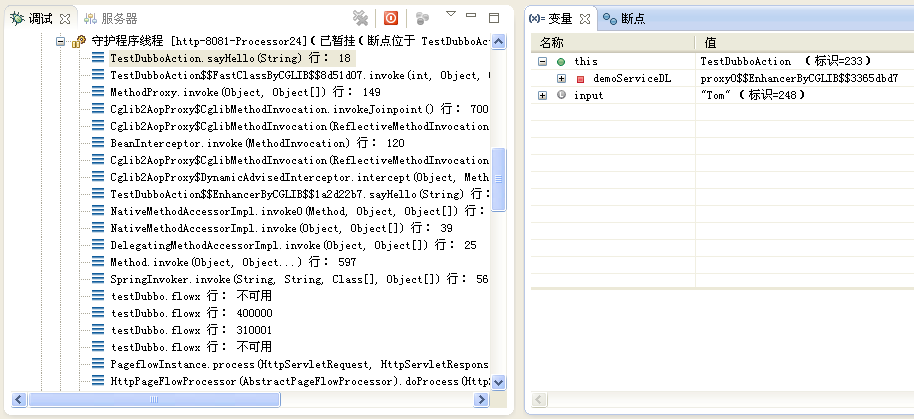
运行服务消费者对应的Tomcat服务器，访问如下页面：



点击 show Greeting按钮，后台通过页面流首先进入TestDubboAction中，如下图所示：

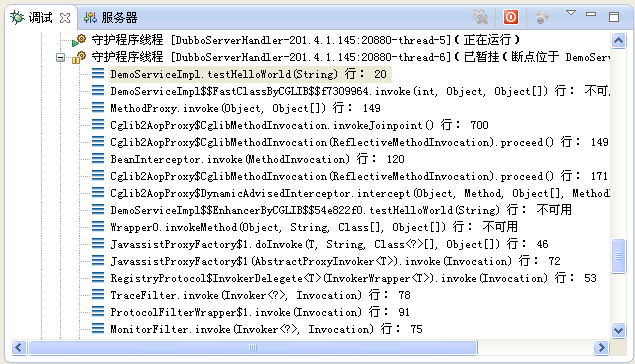


此时我们可以看到代码堆栈为:



Dubbo框架通过Spring配置封装了一个DemoService实例，其实这是一个远程调用的代理客户端。

然后通过Dubbo的协议调用了远端服务，服务方的调用栈为：



服务方也屏蔽了协议的实现，通过代理间接调用了DemoServiceImple实例的testHelloWorld方法。

如此，Dubbo与EOSPlatform无缝结合完成。

查看控制台可以看到服务消费者信息如下：

