# Server端

## 安装插件

安装CXF插件：

|  |
| --- |
| compile ":cxf:1.1.1" |

## 使用命令发布web service

注意，该方式已经不推荐使用了！

### Endpoint和service

CXF插件引入了一个新的artefact类型endpoint，endpoint类型对象都位于grails-app/endpoints目录下。

插件会自动将grails-app/endpoints和grails-app/services目录下有webservice注解的对象发布成CXF web service。

可以使用create-endpoint和create-endpoint-simple命令分别创建endpoint类型是JAX-WS和SIMPLE类型的endpoint。例如：

|  |
| --- |
| grails create-endpoint com.tapi.grails.demo.hello |

运行上面的命令后，就会在grails-app/endpoints下新建一个HelloEndpoint.groovy类:

|  |
| --- |
| package com.tapi.grails.demo  import org.grails.cxf.utils.EndpointType  class HelloEndpoint {  static *expose* = EndpointType.*JAX\_WS*  static *excludes* = []  String serviceMethod(String s) {  return s  }  } |

也可以使用create-cxf-service 和create-cxf-service-simple命令在grails-app/services目录新建可以发布成web service的service类。例如：

|  |
| --- |
| grails create-cxf-service com.tapi.grails.demo.greet |

上面的命令运行后，就会在grails-app/services目录下增加一个GreetService.groovy类:

|  |
| --- |
| package com.tapi.grails.demo  import org.grails.cxf.utils.EndpointType  class GreetService {  static *expose* = EndpointType.*JAX\_WS*  static *excludes* = []  String serviceMethod(String s) {  return s  }  } |

通过命令创建的service和endpoint会在应用运行时被插件自动发布成web service。

建议使用endpoint类型发布web service，这样便于和普通service区分。把普通service发布成web service的目的是为了重用代码。

## 使用annotation发布web service

在cxf插件的新版本推荐直接使用注解发布endpoint或者service为web service。

### 发布简单服务

Simple front end server endpoint就是将一个简单的service对象发布为web service。例如：有如下一个service：

|  |
| --- |
| package com.tpai.grails.demo  class HelloSimpleService {  public String hello(String name) {  // TODO Auto-generated method stub  return "hello ${name}!";  }  public int add(int a, int b) {  // TODO Auto-generated method stub  return a + b;  }  } |

使用@GrailsCxfEndpoint将其发布成一个简单web service，代码如下：

|  |
| --- |
| package com.tpai.grails.demo  import org.grails.cxf.utils.EndpointType  import org.grails.cxf.utils.GrailsCxfEndpoint  @GrailsCxfEndpoint(expose = EndpointType.*SIMPLE*)  class HelloSimpleService {  public String hello(String name) {  // TODO Auto-generated method stub  return "hello ${name}!";  }  public int add(int a, int b) {  // TODO Auto-generated method stub  return a + b;  }  } |

启动应用，该web service的wsdl访问url为：

http://localhost:8080/HelloWebService/services/helloSimple?wsdl

发布简单web service有时在使用JAXB绑定时会出错，所以这并不是一种常用的方式。

### 发布JAX-WS服务

发布JAX-WS类型的web service除了@GrailsCxfEndpoint外还需要使用到@WebMethod, @WebResult and @WebParam。例如：

|  |
| --- |
| package com.tapi.grails.demo  import javax.jws.WebMethod  import javax.jws.WebParam  import javax.jws.WebResult  import org.grails.cxf.utils.EndpointType  import org.grails.cxf.utils.GrailsCxfEndpoint  @GrailsCxfEndpoint(expose = EndpointType.*JAX\_WS*)  class HelloJAXEndpoint {  @WebResult(name = 'helloResponse')  @WebMethod(operationName = 'hello')  public String hello(@WebParam(name="name") String name) {  // TODO Auto-generated method stub  return "hello ${name}!";  }  @WebResult(name = 'addResponse')  @WebMethod(operationName = 'add')  public int add(@WebParam(name="a") int a, @WebParam(name="b") int b) {  // TODO Auto-generated method stub  return a + b;  }  } |

启动应用，该web service的wsdl访问url为：

http://localhost:8080/HelloWebService/services/helloJAX?wsdl

## 使用WSDL First的方式开发web service。

### 使用endpoint继承接口

该方式需要先从wsdl生成java代码。

#### wsdl-to-java

将Wsdl2java.groovy脚本放到项目的grails-app/scripts目录下

如果wsdl文件为HellWorld.wsdl。

在cmd窗口运行插件的wsdl2java命令：

|  |
| --- |
| grails wsdl-to-java "--wsdl=HelloWorld.wsdl --mark" |

运行完毕后，将在项目的src/java目录下生成对应的java代码。

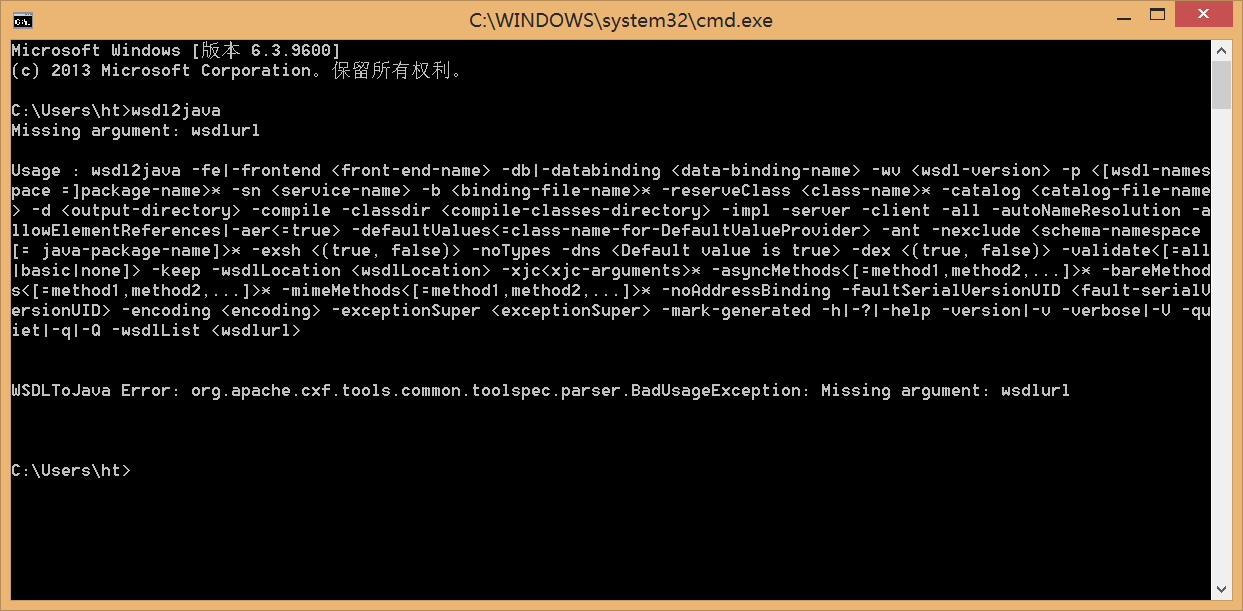
#### wsdl2java

也可以使用cxf自带的wsdl2java命令：

首先确保安装了CXF，目前插件对应的CXF版本是2.6.2。

将cxf目录/bin加到环境变量path中。

在命令窗口中运行wsdl2java如出现下面的信息，表示cxf安装正确：



cd到wsdl文件所在的目录，运行wsdl2java命令：

|  |
| --- |
| wsdl2java -p com.tapi.grails.demo -fe jaxws21 -encoding UTF-8 HelloWorld.wsdl |

需注意插件使用jazws2.1的版本，所以需要有参数-fe jaxws21.

正常情况下生成对应的java文件。将生成的代码放到项目的src/java目录下。

#### 创建web servie

有了server端的java代码之后，可以使用两种方式创建web service。

新建一个endpoint，在grails-app/endpoints目录下生成一个HelloWSDLEndpoint，代码如下：

|  |
| --- |
| package com.tapi.grails.demo  import org.grails.cxf.soap.HelloWorldService  class HelloWSDLEndpoint implements HelloWorldService{  @Override  public String hello(String name) {  // TODO Auto-generated method stub  return "hello ${name}!";  }  @Override  public int add(int a, int b) {  // TODO Auto-generated method stub  return a + b;  }  } |

该endpoint继承前面运行wsdl-to-java命令生成的server端接口HelloWorldService。运行应用run-app，则服务自动发布在：

http://localhost:8080/HelloWebService/services/helloWSDL?wsdl。

### 使用annotation

也可以直接使用wsld文件，只需要注解一个endpoint，EndpointType采用wsdl first的方式。

在grails-app/endpoints目录下生成HelloWSDLFirstEndpoint.groovy。对该endpoint加上注解，代码如下：

|  |
| --- |
| package com.tapi.grails.demo  import javax.jws.WebService  import org.grails.cxf.utils.EndpointType  import org.grails.cxf.utils.GrailsCxfEndpoint  @WebService(name = 'HelloWorldServiceService',  targetNamespace = 'http://Carbusiness/',  serviceName = 'HelloWorldServiceService',  portName = 'HelloWorldServicePort')  @GrailsCxfEndpoint(wsdl = 'org/grails/cxf/soap/HelloWorld.wsdl', expose = EndpointType.*JAX\_WS\_WSDL*)  class HelloWSDLFirstEndpoint {  public String hello(String name) {  // TODO Auto-generated method stub  return "hello ${name}!";  }  public int add(int a, int b) {  // TODO Auto-generated method stub  return a + b;  }  } |

运行应用run-app，则服务自动发布在：

http://localhost:8080/HelloWebService/services/helloWSDL?wsdl

# 客户端

假设有一个web service，其wsdl所在路径为：

http://10.100.133.101:8080/HelloWebService/services/helloJAX?wsdl

## 从wsdl生成java代码

得到web service的wsdl文件后，利用cxf的wsd2java命令生成客户端代码：

|  |
| --- |
| wsdl2java -p com.tapi.grails.demo.client -fe jaxws21 -encoding UTF-8 -client helloJAX.wsdl |

将生成的java代码复制到项目的src/java目录下。

## 配置cxf客户端

在Config.groovy中配置cxf客户端：

|  |
| --- |
| //web services client  cxf {  client {  helloServiceClient {  //used for invoking service  clientInterface = com.tapi.grails.demo.client.HelloJAXEndpoint  serviceEndpointAddress = "http://localhost:8080/HelloWebService/services/helloJAX"  }  }  } |

这样就可以使用名为：helloServiceClient的bean调用web service了。

## 注入service client bean

注意通过插件生成的客户端不能通过grails的依赖注入自动注入!

在需要调用web服务的controller或者service中通过如下方式注入服务客户端，即可调用web service：

|  |
| --- |
| package demo  import grails.plugin.springsecurity.annotation.Secured  /\*\*  \* TestHelloController  \* A controller class handles incoming web requests and performs actions such as redirects, rendering views and so on.  \*/  @Secured('permitAll')  class TestHelloController {    def grailsApplication    def index = {  def helloServiceClient = grailsApplication.mainContext.getBean("helloServiceClient")  def result = helloServiceClient.add(2, 8)  render helloServiceClient.hello("world${result}")  render "OK"  }  } |