webservice

# 客户/服务器模式

## 便民网站案例分析

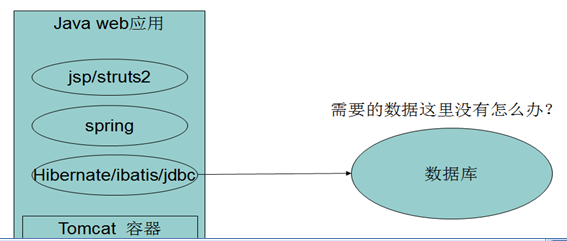
如何实现便民查询网站：



便民查询网站关键要解决问题是要查询的数据从哪里查询？

方案1：

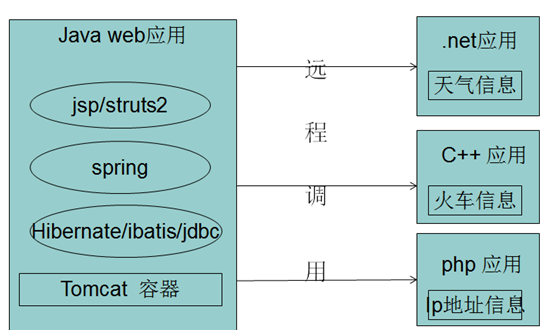
要查询的数据从数据库中查询。



天气实时信息、火车时刻实时信息是本系统的数据库中是不存在的，此方案不可行。

方案2：

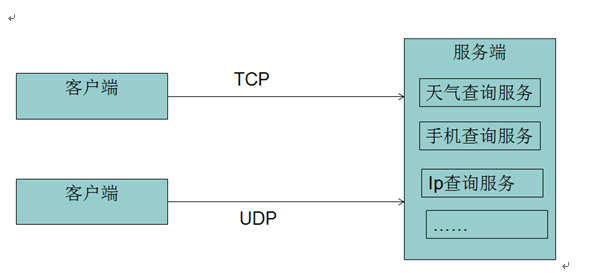
天气实时信息在气象局系统中、火车时刻实时信息在12306系统中，本系统通过网络去调用气象局系统、12306系统去获取要的数据。



本系统需要跨平台远程调用其它系统（气象局系统、12306）.

## 客户/服务器模式

将上图抽象成如下图：



客户端：用于调用远程的服务端，是调用方。

服务端：用于对外提供服务供远程的计算机去调用，是被调用方。

如果客户端还对外提供服务，客户端同时还是一个服务端。

如果服务端去调用其它的服务，服务端同时还是一个客户端。

客户端程序和服务端程序可以在一台计算机上部署运行，一般情况下客户端和服务端程序是部署在不同的计算机上。

客户端程序和服务端程序通过网络协议（tcp/ip）进行通信。

包括 tcp和udp：

tcp：是一种面向连接的协议，客户端和服务端经过三次握手建立连接，传输数据稳定。

udp：是一种数据报协议，传输数据不稳定，速度要比tcp要快。

服务端要对外监听一个服务端口，客户端通过服务端口请求服务端。

如果服务端使用java编程，最终要通过java类方法、java接口对外提供服务。

客户端需要知道服务端的ip+端口方可调用服务端。

客户/ 服务器模式是一种网络架构，早期应用于c/s系统，web兴起之后应用于b/s系统，b/s方式更方便进行系统开发和维护。

webservice就是客户/服务端模式一个应用技术。

# 客户服务器Socket实现

socket是网络通信基础，socket可以基于tcp、udp实现网络通信。所有网络通信的技术都是基于socket来实现的。

java程序主要实现的服务端编程，使用socket可以实现服务端。

## 分析socket实现步骤

使用socket实现天气查询。

客户端

socket类

服务端

serverSocket类

outputstream

inputstream

outputstream

inputstream

1、服务端启动一个天气查询服务

2、客户端创建一个sokcet连接。

3、客户端通过outputstream向服务端发送数据（发送城市名称）

4、服务端通过inputstream接收客户端发送的数据

5、服务端根据城市名称查询天气

6、服务端通过outputstream将查询天气的结果响应给客户端

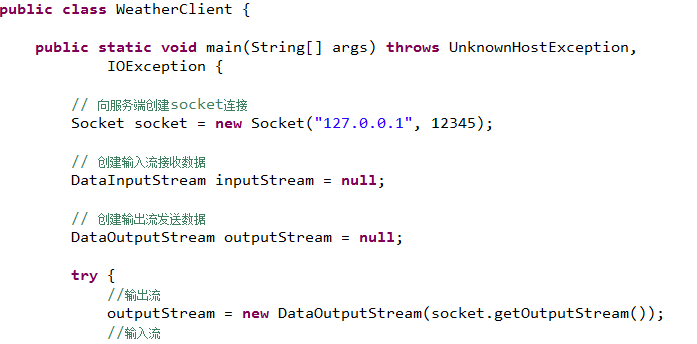
7、客户端通过inputstream接收服务端响应的数据

## 实现

### 服务端



### 客户端



### 调试

启动服务端

客户端向服务端发送数据

## 服务端多线程处理（企业开发掌握）

服务端必须要具备以下能力：

1、服务端要有可持续服务能力，对多个客户端持续提供服务。

2、服务端要有并发处理能力，通常使用多线程来处理客户端的请求。

### 服务端程序修改

解决持续服务的问题：

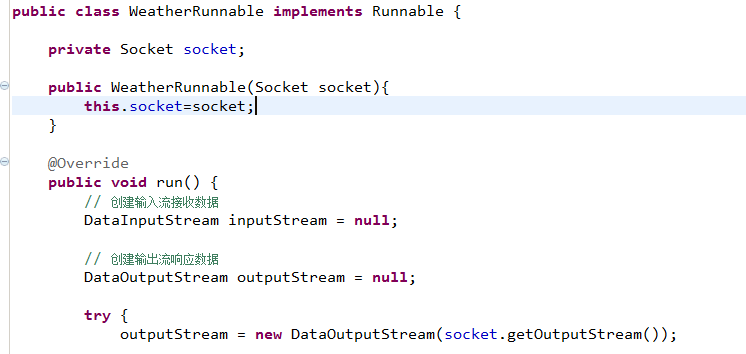
使用while(true)



解决并发处理问题：

使用多线程处理客户端的请求，客户端请求到服务端，服务端开启一个线程来处理。

java实现多线程，建议使用Runnable接口。



服务端多线程处理逻辑：



## socket编程总结

socket可以实现客户/服务器模式应用，基于tcp/ip协议进行通信。

### 优点

1、跨平台特性

socket是具有跨平台特性的，将socket理解为计算机与计算机之间进行网络通信的接口或规范。

不同的编程语言都提供了socket 实现方法。

跨平台：操作系统不同、编程语言不同。

2、socket传输数据的速度快

socket是基于tcp、udp进行传输。

### 缺点

使用socket编程，在接收和发送数据时通过手动解析流的方式去完成，编程实现困难。

设想：可以使用面向对象方法解决。

# 什么是webservice

## 什么是webservice

webservice即web服务，是一种跨平台的远程调用技术，实现客户/ 服务器模式的应用。

webservice基于soap协议传输数据。

SOAP(Simple Object Access Protocol)协议是基于http的应用层协议，http基于tcp的应用层协议。

soap属于w3c标准。

soap实现面向对象开发。

采用wsdl作为描述语言即webservice使用说明书，来描述服务端提供webservice接口的内容，wsdl属w3c标准。

xml是webservice的跨平台的基础，XML主要的优点在于它既与平台无关，又与厂商无关。

XSD，W3C为webservice制定了一套传输数据类型，使用xml进行描述，即XSD(XML Schema Datatypes)，任何编程语言写的webservice接口在发送数据时都要转换成webservice标准的XSD发送。

当前非SOAP协议的webService以轻量为首要目标，比如http rest方式也是webservice的一种方式，或者直接使用http自定义数据协议，比如http传输json数据，http传输xml数据等。

## webservice三要素

### soap(掌握)

SOAP即简单对象访问协议(Simple Object Access Protocal) 是一种简单的基于 XML 的协议，它使应用程序通过 HTTP 来交换信息，简单理解为soap=http+xml。

Soap协议版本主要使用soap1.1、soap1.2。

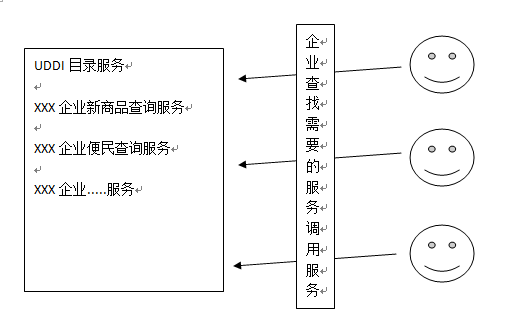
SOAP不是webservice的专有协议，其他应用协议也使用soap传输数据。例如，SMTP、tr069等。

### wsdl(掌握)

WSDL 是基于 XML 的用于描述Web Service及其函数、参数和返回值。通俗理解Wsdl是webservice的使用说明书。

### UDDI（了解）

UDDI旨在将全球的webservcie资源进行共享，促进全球经济合作。



UDDI现状：

目前大部分企业使用webservice并不是必须使用UDDI，因为用户通过WSDL知道了web service的地址，可以直接通过WSDL调用webservice。

## webservice开发规范

JAVA 中共有三种WebService 规范，分别是JAX-WS（JAX-RPC）、JAXM&SAAJ、JAX-RS。

# jaxws入门程序

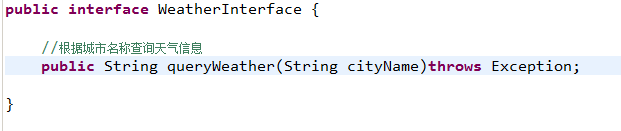
## 需求

使用jaxws开发方法开发天气查询服务（根据城市名称查询天气信息）。

## 服务端

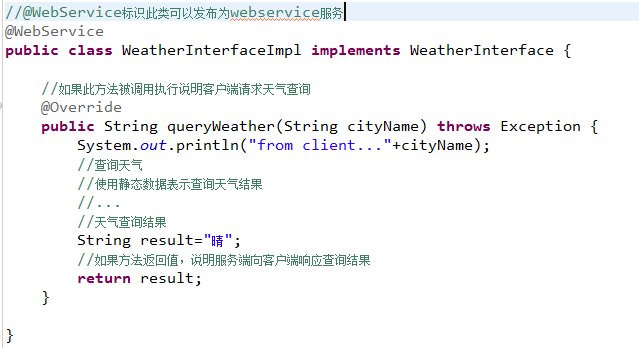
### 定义接口

定义服务接口SEI(Service Endpoint Interface)，SEI在webservice中称为portType。



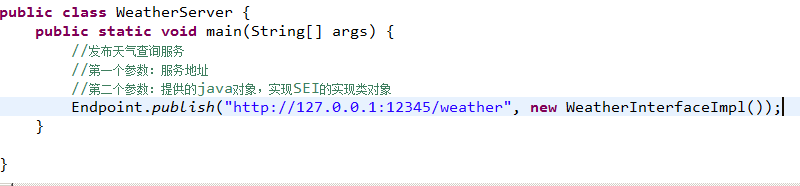
### 定义接口实现类

接口实现类实现SEI，最终实现类接收客户端的请求查询天气。



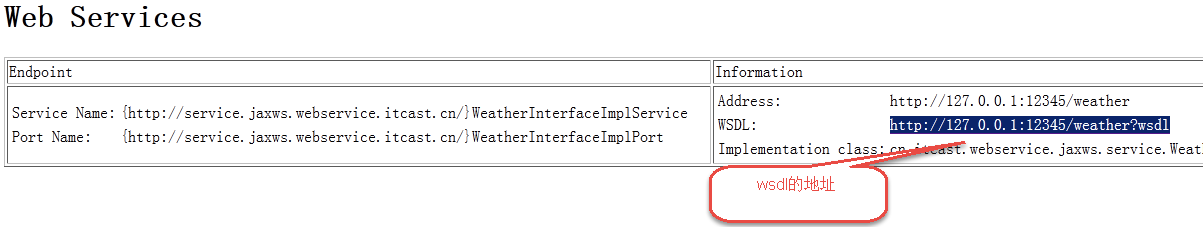
### 启动服务

启动服务让SEI的实现类接收客户端的请求。



## wsdl

### wsdl地址



wsdl地址就是：webservice的服务地址+?wsdl

### 什么是wsdl

WSDL 指网络服务描述语言(Web Services Description Language)，描述的webservice服务接口的内容（方法的输入参数、输出参数以方法功能描述）

WSDL是一种使用 XML 编写的文档。这种文档可描述某个 Web service。它可规定服务的位置，以及此服务提供的操作（或方法）。

WSDL 是一种 XML 文档

WSDL 用于描述网络服务

WSDL 也可用于定位网络服务

### 如何阅读wsdl（掌握）

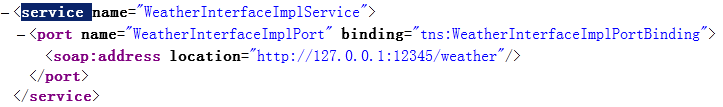
wsdl文档namespace的名称默认就是SEI的包名的倒叙。



wsdl文档从下向上阅读 。

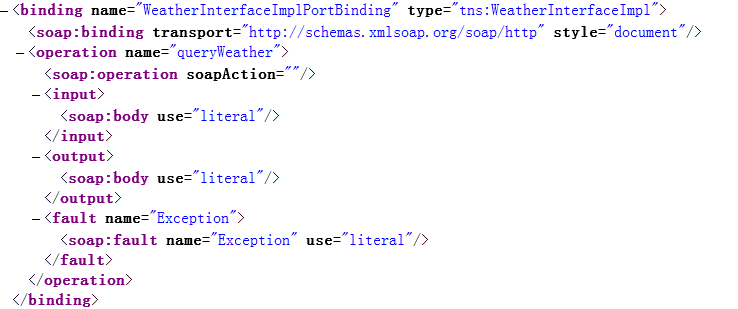
* 服务视图

在service标签中可以包括多个webservice的port（服务端点），通过服务端点访问服务。

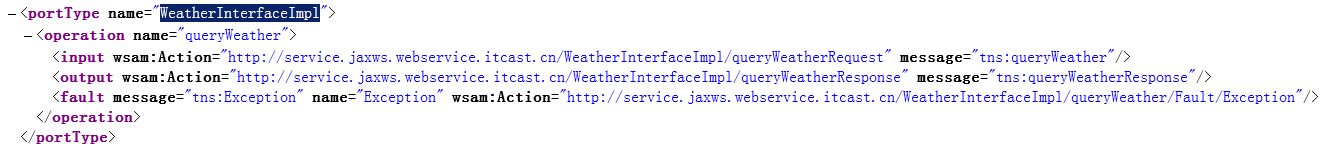


* bingding

通过binding找到portType



* portType(服务端点类型，即SEI)



queryWeather：方法名

porttype的名称和服务端点类型即SEI的类名默认一致。

* 方法输入和输出



## 客户端

### 客户端实现思路

编写客户端程序，向webservice服务端发起soap请求，其实底层还是socket请求。

客户端使用jaxws要面向对象开发，面向对象方式向服务端发起请求。

第一步：

使用jdk提供的wsimport工具生成webservice服务端的调用代码

第二步：

使用调用代码，创建一个代理对象（portType的代理对象）

### wsimport工具

jdk的bin目录 中提供了wsimport，根据wsdl地址生成webservice服务的调用代码。

常用参数为:

-d<目录> - 将生成.class文件。默认参数。

-s<目录> - 将生成.java文件。

-p<生成的新包名> -将生成的类，放于指定的包下

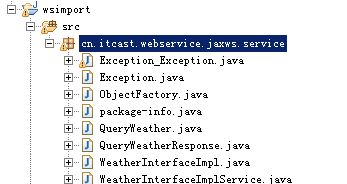
wsimport必须后边跟wsdl的地址。

命令格式：

wsimport wsdl地址 -s 源文件存放的目录



生成的调用代码如下：



生成的调用代码默认包名是wsdl的namspace的倒叙。

### 将生成的调用代码拷贝到客户端工程



# webservice与socket区别

## socket优点

sockct是跨平台，基于tcp、udp通信，速度快。

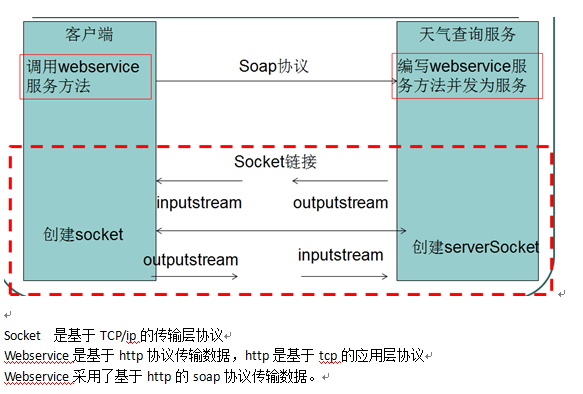
## socket缺点

socket需要程序手动通过流的方式去解析发送和接收的数据。

## webservice原理

webservice基于soap协议(基于http的应用层协议)进行客户端和服务端通信。

soap基于http，http基于tcp，webservice基于soap协议通信，底层还是socket.



结论：

Webservice采用soap协议进行通信，底层基于socket通信，webservice不需专门针对数据流的发送和接收进行处理，是一种跨平台的面向对象远程调用技术。

## webservice优点

1 、采用xml支持跨平台远程调用。

2、基于http的soap协议，可跨越防火墙。

3、支持面向对象开发。

4、有利于软件和数据重用，实现松耦合。

## webservice缺点

由于soap是基于xml传输，本身使用xml传输会传输一些无关的东西从而效率不高，随着soap协议的完善，soap协议增加了许多内容，这样就导致了使用soap协议进行数据传输的效率不高。

# webService应用场景

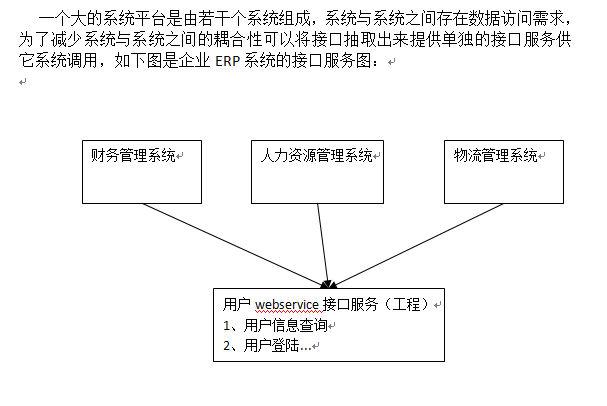
webservice用于开发远程调用接口，接口提供服务。

## 用于公开服务接口

由于soap协议是标准协议，webservice可以用于开发一些面向互联网公开的服务接口，比如：公司的产品信息查询、天气查询、火车时刻查询。

注意：上边用于开发公开服务接口的前提对性能要求不高。

## 用户内部服务接口



# webservice技术

## webservice技术扩展

由于soap协议本身的缺点：面向对象开发，传输数据的速度慢。

扩展1：

如果考虑速度不用soap，直接使用http方式+自定义数据格式，比如：http+自定义xml、http+自定义json、http+自定义字符串（用户id#用户名称#用户年龄）

直接使用http方式比soap协议简单。

比如：第三方支付的对接接口使用http方式。

扩展2：

直接使用socket作为webservice也是可以的，使用此方法主要是出于对性能的考虑。socket是基于tcp、udp，性能高。

扩展3：

基于tcp的通信，支持面向对象，使用hessian框架。

总结：

webservice就是发布一个服务接口，供其它计算机调用，可以实现的技术有好多，socket、http、soap方式、hessian等，在实际应用时根据具体的需求和场景去选择应用技术。

## 企业中对webservice技术选型

### 协议约定

一般情况下服务端的协议决定了客户端的协议。

比如：企业用户查询服务接口对接，首先了解服务端接口的协议，客户端和服务端保持一致使用相同的协议去调用服务端。

### 通用性

对于一个webservice主要考虑接口的通用性时，在不要求性能的前提下可以使用soap协议。

如果要考虑接口的通用性还要考虑接口的性能可以采用http+自定义协议，比如http+json串，因为http本身就是web应用中的基础协议，json格式的数据也成为当前web开发中流行数据格式。

### 高性能

对webservice接口性能要求较高，优先使用性能高的接口协议。

性能从高到低是：socket>hessian>http>soap

### 开发规范

当然要注意，有些正规的公司会制定自己公司的开发规范，比如内部系统接口统一采用hessian，统一采用http，并不总是使用性能最高的方式，因为公司会综合考虑自己的人力资源情况在有限的成本下使用最合适的技术才是公司在进行技术选型时所遵循的原则。

# 调用公网天气查询webservice客户端

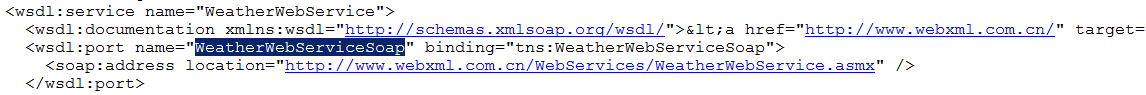
## 需求

编写webservice客户端调用公网天气查询的webservice接口。

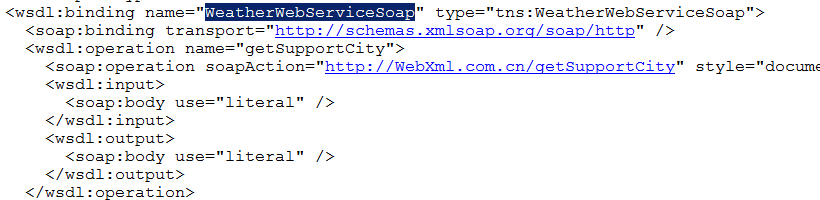
根据城市名称查询天气。

## 阅读公网天气查询的wsdl

服务视图：



binding：



portType：



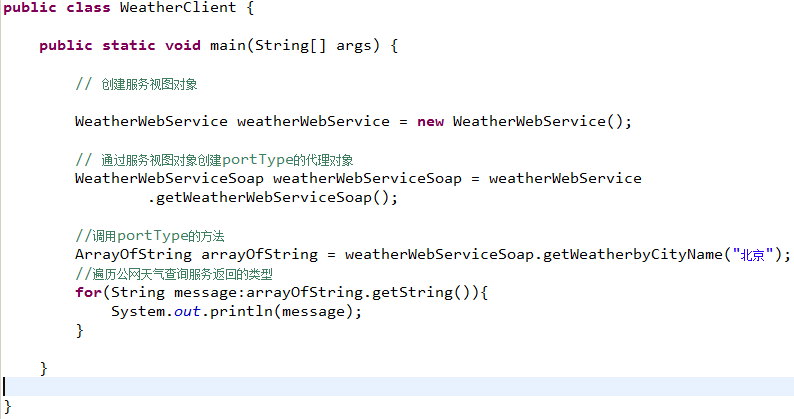
## 公网天气查询客户端实现

### wsimport生成调用代码



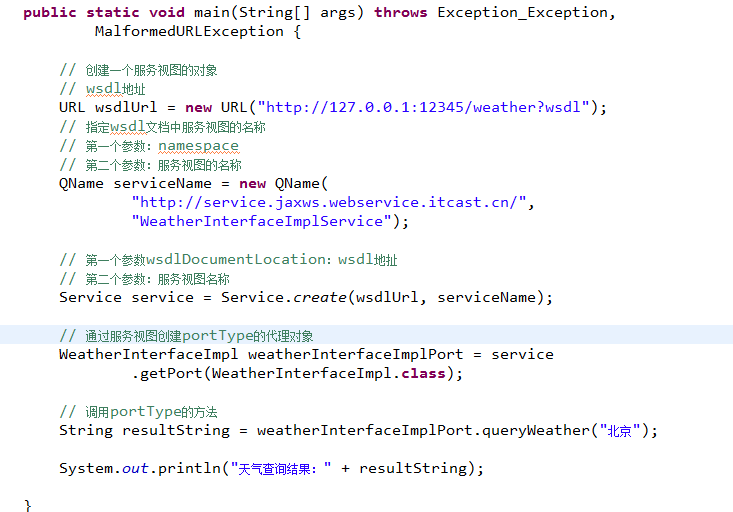


### 创建客户端工程将生成代码拷贝到工程中



# 使用service类编写webservice客户端

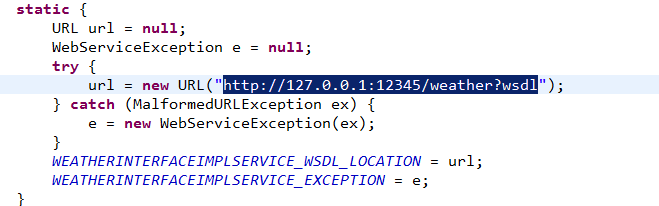
## 使用javax.xml.ws.Service创建客户端



## 使用service类和使用wsimport方式区别

wsimport：使用生成代码创建portType的代理对象。

使用生成代码创建服务视图对象，在生成代码中将webservice的地址硬编码在生成代码中，如果将来webservice地址变更了，应该重新生成调用代码，不方便进行系统维护。



service：使用service类创建portType的代理对象。

使用service类创建服务视图对象时，需要程序员手动指定webservice地址，程序员可以将webservice地址在配置文件中配置，如果将来webservice地址变更了只需要修改配置文件中地址即可。

结论：

建议使用service类方法编写webservice客户端。

## 使用service类调用公网天气webservice



# soap

## 什么是soap

soap是一种网络通信协议，soap是基于http的应用层协议。

soap协议就是通过http传输xml数据，soap数据格式是xml格式。

soap是简单对象访问协议，支持面向开发。

soap协议常用的版本是soap1.1和soap1.2版本。

## soap 协议格式内容

### 使用Tcp/ip monitor工具监控tcp请求及响应的数据

Tcp/ip monitor工具原理基于代理方式进行监控：

客户端

请求服务端ip和端口 改为请求Tcp/ip monitor工具的端口

服务端

启动一个webservice服务

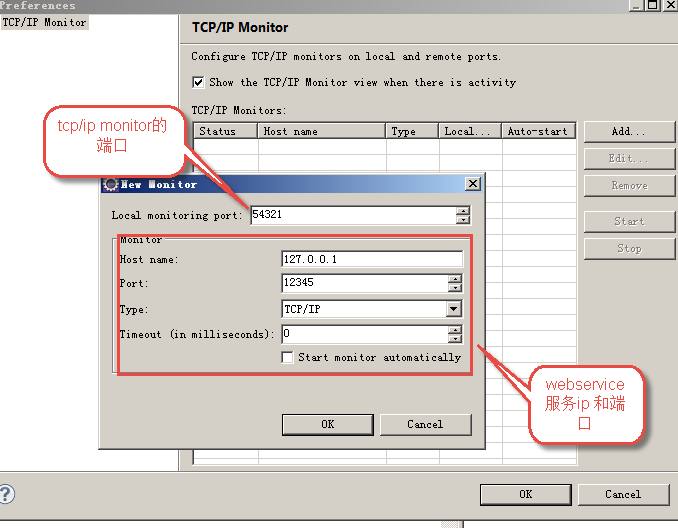
监听一个服务端口

soap协议

Tcp/ip monitor工具

soap协议

eclipse中自带了Tcp/ip monitor工具：



### soap1.1协议

客户端 程序向Tcp/ip monitor工具的端口发送webservice请求。

例子中，客户端要向http://127.0.0.1:54321/weather?wsdl发送数据，要使用service类客户端开发方法。

* soap1.1请求：

POST /weather HTTP/1.1

Accept: text/xml, multipart/related

Content-Type: text/xml; charset=utf-8

SOAPAction: "http://service.jaxws.webservice.itcast.cn/WeatherInterfaceImpl/queryWeatherRequest"

User-Agent: JAX-WS RI 2.2.4-b01

Host: 127.0.0.1:54321

Connection: keep-alive

Content-Length: 227

soap协议体内容

<?xml version="1.0" ?>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

<S:Body>

<ns2:queryWeather xmlns:ns2="http://service.jaxws.webservice.itcast.cn/">

<arg0>北京</arg0>

</ns2:queryWeather>

</S:Body>

</S:Envelope>

* soap1.1.响应：

HTTP/1.1 200 OK

Transfer-encoding: chunked

Content-type: text/xml; charset=utf-8

Date: Sun, 13 Dec 2020 12:15:27 GMT

<?xml version="1.0" ?>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">

<S:Body>

<ns2:queryWeatherResponse xmlns:ns2="http://service.jaxws.webservice.itcast.cn/">

<return>晴</return>

</ns2:queryWeatherResponse>

</S:Body>

</S:Envelope>

* soap1.1协议：

post请求方式

Content-Type: text/xml;

命名空间：http://schemas.xmlsoap.org/soap/envelope/

soap协议内容：

必需有 Envelope 元素，此元素将整个 XML 文档标识为一条 SOAP 消息

可选的 Header 元素，包含头部信息

必需有Body 元素，包含所有的调用和响应信息

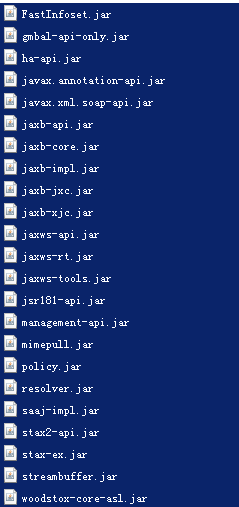
可选的 Fault 元素，提供有关在处理此消息所发生错误的信息

### soap1.2协议

更改服务端程序，支持soap1.2协议。

* 下载 jaxws-ri-2.2.8

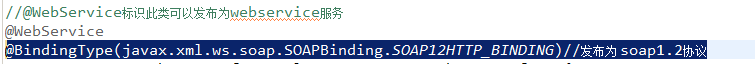
Jaxws实现soap1.2需要加入jaxws扩展包，从sun下载jaxws-ri-2.2.8，解压jaxws-ri-2.2.8并将lib下的jar包加载到服务端的工程中。



* 添加BindingType

在SEI实现类上添加如下注解

@BindingType(javax.xml.ws.soap.SOAPBinding.SOAP12HTTP\_BINDING)



* soap1.2请求内容：

POST /weather HTTP/1.1

Accept: application/soap+xml, multipart/related

Content-Type: application/soap+xml; charset=utf-8;action="http://service.jaxws.webservice.itcast.cn/WeatherInterfaceImpl/queryWeatherRequest"

User-Agent: JAX-WS RI 2.2.4-b01

Host: 127.0.0.1:54321

Connection: keep-alive

Content-Length: 225

<?xml version="1.0" ?>

<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">

<S:Body>

<ns2:queryWeather xmlns:ns2="http://service.jaxws.webservice.itcast.cn/">

<arg0>北京</arg0>

</ns2:queryWeather>

</S:Body>

</S:Envelope>

* soap1.2响应：

HTTP/1.1 200 OK

Transfer-encoding: chunked

Content-type: application/soap+xml; charset=utf-8

Date: Sat, 12 Dec 2020 12:30:19 GMT

<?xml version='1.0' encoding='UTF-8'?>

<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">

<S:Body>

<ns2:queryWeatherResponse xmlns:ns2="http://service.jaxws.webservice.itcast.cn/">

<return>晴</return>

</ns2:queryWeatherResponse>

</S:Body>

</S:Envelope>

### soap1.1和soap1.2的对比

相同之处：

soap1.1和soap1.2都是使用post方法

都包括Envelope和body

内容类型context-type不同：

soap1.1使用text/xml

soap1.2使用application/soap+xml

命名空间Envelope xmlns不同：

soap1.1使用<http://schemas.xmlsoap.org/soap/envelope/>

soap1.2使用<http://www.w3.org/2003/05/soap-envelope>

## 以原始http方式发送soap内容

**public** **class** WeatherClient3 {

**public** **static** **void** main(String[] args) **throws** Exception\_Exception,

IOException {

//定义url，指定webservice地址

URL url = **new** URL("http://127.0.0.1:12345/weather");

//创建一个http的连接

HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();

//设置要请求及响应

httpURLConnection.setDoInput(**true**);

httpURLConnection.setDoOutput(**true**);

//设置http请求的方法为POST

httpURLConnection.setRequestMethod("POST");

//设置content-type

// httpURLConnection.setRequestProperty("Content-Type", "text/xml; charset=utf-8");

httpURLConnection.setRequestProperty("Content-Type", "application/soap+xml; charset=utf-8;");

//请求的soap协议内容

String soapString = *sent\_soap12xmlString*("北京");

//通过输出流发送数据

OutputStream outputStream = httpURLConnection.getOutputStream();

//通过http连接向webservice的服务端发送soap数据（xml格式）

outputStream.write(soapString.getBytes());

outputStream.close();

//通过输入流接收响应数据

InputStream inputStream = httpURLConnection.getInputStream();

//通过读取inputStream的内容获取服务端响应信息

**byte**[] b = **new** **byte**[1024];

//通过一个输出流接收buffer中的数据

ByteArrayOutputStream byteArrayOutputStream = **new** ByteArrayOutputStream();

**while**(**true**){

**int** len = inputStream.read(b);

**if**(len == -1){

**break**;

}

byteArrayOutputStream.write(b, 0, len);

}

//读取数据完成

inputStream.close();

//从byteArrayOutputStream获取数据，获取到响应的数据

String responseString = byteArrayOutputStream.toString();

byteArrayOutputStream.close();

System.*out*.println(responseString);

}

//发送的soap1.1协议内容

// public static String sent\_soap11xmlString(String cityName){

// String soapString = "<?xml version=\"1.0\" ?>" +

// "<S:Envelope xmlns:S=\"http://schemas.xmlsoap.org/soap/envelope/\">" +

// "<S:Body>" +

// "<ns2:queryWeather xmlns:ns2=\"http://service.jaxws.webservice.itcast.cn/\">" +

// "<arg0>"+cityName+"</arg0>" +

// "</ns2:queryWeather>" +

// "</S:Body>" +

// "</S:Envelope>";

// return soapString;

// }

**public** **static** String sent\_soap12xmlString(String cityName){

String soapString = "<?xml version=\"1.0\" ?>" +

"<S:Envelope xmlns:S=\"http://www.w3.org/2003/05/soap-envelope\">" +

"<S:Body>" +

"<ns2:queryWeather xmlns:ns2=\"http://service.jaxws.webservice.itcast.cn/\">" +

"<arg0>"+cityName+"</arg0>" +

"</ns2:queryWeather>" +

"</S:Body>" +

"</S:Envelope>";

**return** soapString;

}

}

# 区域查询接口案例

企业开发常将接口的数据格式定为xml数据格式，如果采用soap方式的webservice实现接口，常用通过soap传输xml业务数据。

通过此案例的开发，了解企业中开发接口的过程。

## 案例分析

* 区域查询的意义：

一个平台中有很多的系统，这些系统都需要查询区域信息，为了简化开发，减少系统间的耦合性，提高系统的可扩展性将一些通常接口提取出来组成一个服务接口，供其它的系统调用。

企业的ERP系统

CRM客户关系管理系统

人力资源管理系统

.....

将区域 查询的功能抽取出来组成一个系统，对外提供区域查询服务

mysql 区域信息

* 接口分析

企业开发有很多接口需要开发，一般情况下由服务端先确定接口格式，接口格式包括请求的内容和响应的内容。

服务端开发程序员确定好接口后，叫上客户端程序员一起来讨论接口的合理性。

由于xml数据跨平台特性，规定请求的数据格式和响应的数据格式都采用xml格式描述。

客户端发送数据格式：

<?xml version="1.1" encoding="utf-8"?>

<queryarea>

<parentid> </parentid>//父级区域id

<start></start>//起始记录，从1开始

<end></end>//结束记录

</queryarea>

服务端响应数据格式：

<?xml version="1.0" encoding="UTF-8"?>

<areas>

<area>

<areaid> </areaid>//区域id

<areaname></areaname>//区域名称

<arealevel></arealevel>//区域等级

< parentid ></ parentid >//父级区域id

</area>

//…..

</areas>

* 程序实现步骤

客户端

2、将请求的数据格式化为xml数据

3、请求服务端的区域查询接口

9、客户端接收服务端响应数据

10、客户端将接收到服务端的xml数据解析，转成java对象方便处理

服务端

1、启动区域查询服务

4、service接收客户端的请求

5、解析客户端请求的xml数据，把xml数据转成java对象

6、service调用dao查询mysql数据库

7、将查询到区域列表转成xml数据

8、将区域列表的xml数据响应给客户端

## 准备环境

创建mysql数据库，导入area.sql脚本。

java：jdk1.7

eclipse indigo

需要加入mysql驱动包

mysql-connector-java-5.1.7-bin.jar

## 服务端

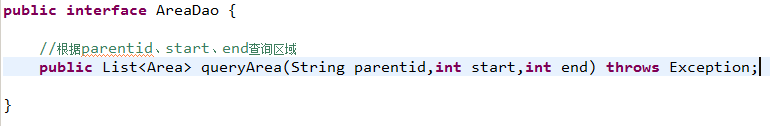
### Dao 数据访问接口

分析：

连接mysql数据库查询webservice数据库中的area表的数据。

功能需求是：根据parentid、start、end查询数据

sql：SELECT \* FROM AREA WHERE parentid='1.' LIMIT 0,2



**public** **class** AreaDaoImpl **implements** AreaDao {

@Override

**public** List<Area> queryArea(String parentid, **int** start, **int** end)

**throws** Exception {

// 通过jdbc查询数据库

// 数据库连接

Connection connection = **null**;

// 预编译的statement，向数据库发起请求

PreparedStatement preparedStatement = **null**;

// 结果集

ResultSet resultSet = **null**;

//区域列表

List<Area> list = **new** ArrayList<Area>();

**try** {

// 加载数据库驱动

Class.*forName*("com.mysql.jdbc.Driver");

// 获取数据库连接

connection = DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/webservice", "root", "mysql");

//构造 sql

String sql = "SELECT areaid,areaname,arealevel,parentid FROM AREA WHERE parentid=? LIMIT ?,?";

//创建statement对象

preparedStatement = connection.prepareStatement(sql);

//向statement中设置参数

preparedStatement.setString(1, parentid);

start = start -1;

**int** len = end - start;

preparedStatement.setInt(2, start);

preparedStatement.setInt(3, len);

//向数据库发起请求

resultSet = preparedStatement.executeQuery();

//解析resultSet

**while**(resultSet.next()){

Area area = **new** Area();

//获取的数据内容根据接口内容去获取

String areaid\_l = resultSet.getString("areaid");

String areaname\_l = resultSet.getString("areaname");

String arealevel\_l = resultSet.getString("arealevel");

String parentid\_l = resultSet.getString("parentid");

area.setAreaid(areaid\_l);

area.setArealevel(arealevel\_l);

area.setParentid(parentid\_l);

area.setAreaname(areaname\_l);

list.add(area);

}

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

//释放资源

**if**(resultSet!=**null**){

resultSet.close();

}

**if**(preparedStatement!=**null**){

preparedStatement.close();

}

**if**(connection!=**null**){

connection.close();

}

}

**return** list;

}

}

### Service服务接口

分析：

接收客户端请求的xml数据

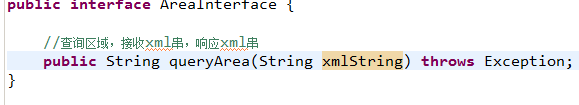
解析xml数据，得到请求的参数parentid、start、end

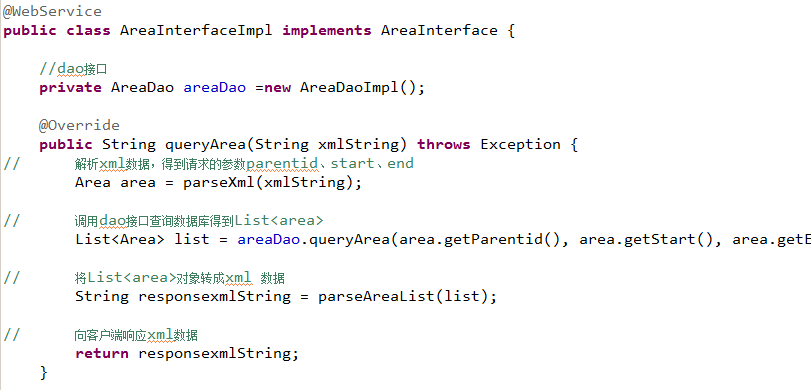
调用dao接口查询数据库得到List<area>

将List<area>对象转成xml 数据

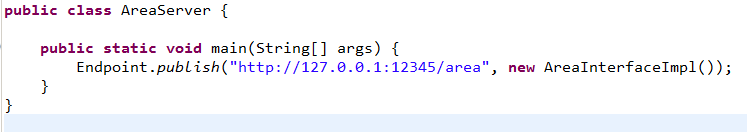
向客户端响应xml数据

最后将服务接口发布为webservice服务。





### 发布服务



## 客户端

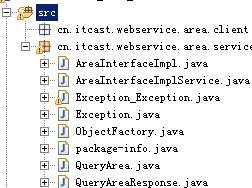
分析：

将请求的数据构造 成xml格式数据

将xml数据发送到webservice服务

将服务端响应的xml数据解析成java对象

### 使用wsimport生成调用代码



### 使用service类的方法编写客户端



## 总结

企业开发中常用将xml、json作为接口传输的数据格式，目前 由于互联网项目较多json比较流行。

开发接口过程：

1、由服务端开发人员定义服务接口（包括服务端响应内容及客户端请求内容）

2、服务端开发人员和客户端开发人员坐下来一块儿讨论确定接口协议（数据格式）

3、服务端开发人员和客户端开发人员按照接口文档的描述进行接口开发

4、服务端开发人员和客户端开发人员开发完成后进行接口调试。

# 区域查询案例深入

## xml串改为面向对象

### soap支持面向对象

soap作为简单对象访问协议它是支持面向对象开发的，在webservice接口传输过程中是可以传输java对象的，其实面对程序员相当于传输的java对象，但是最后还是将对象序列化为soap协议进行传输，如下图：

客户端

2、请求区域查询

传输area对象

6、到客户端，将soap协议反序列化java对象

服务端

1、发布区域查询服务

3、到服务端，将soap协议反序列化java对象

4、查询区域，得到List<Area>

5、向客户端响应

List<Area>

请求area对象, 将area对象序列化soap协议

响应List<area>对象, 将List<area>对象序列化soap协议

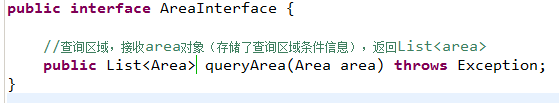
### 服务端

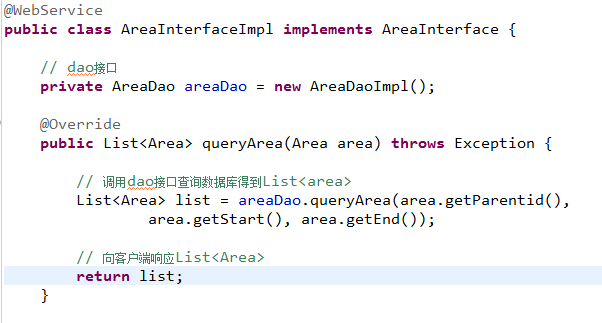
#### dao

同上一例子。

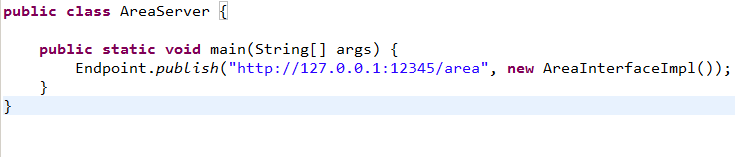
#### service

服务接口方法接收java对象，方法返回java对象 。





#### 发布服务



### 客户端

#### 使用wsimport生成调用代码

#### 客户端代码

**public** **class** AreaClient {

**public** **static** **void** main(String[] args) **throws** MalformedURLException, Exception\_Exception {

//使用service类方法编写客户端

//wsdl地址

URL wsdlDocumentLocation = **new** URL("http://127.0.0.1:12345/area?wsdl");

//serviceName

QName serviceName = **new** QName("http://service.area.webservice.itcast.cn/", "AreaInterfaceImplService");

//创建服务视图对象

Service service = Service.*create*(wsdlDocumentLocation, serviceName);

//生成 portType的代理对象

AreaInterfaceImpl areaInterfaceImplPort = service.getPort(AreaInterfaceImpl.**class**);

//调用 areaInterfaceImplPort的方法相当于请求服务端

Area area = **new** Area();

area.setParentid("1.1.");

area.setStart(1);

area.setEnd(2);

List<Area> list = areaInterfaceImplPort.queryArea(area);

//服务端响应的 List<Area>

System.*out*.println(list);

}

}

### 总结

soap协议支持面向对象开发，传输对象比传输xml方便，但是之所以企业中开发webservice采用xml描述请求和响应的数据是因为xml数据格式是一种跨平台的标准数据格式，企业在开发中会选择一些标准的数据格式作为接口协议，比如：xml、json。

实际应用时根据企业中的开发需求确定是传输java对象还是xml串以及json串等。

## jaxws注解

### 注解说明

通过jaxws提供的注解主要用于规范webservice接口。

### 注解例子

将区域查询的服务接口使用注解进行规范。

//targetNamespace指定命名空间

@WebService(targetNamespace = "http://area.itcast.cn/", serviceName = "AreaWebService", name = "AreaServiceSoap", portName = "AreaServiceSoap")

**public** **class** AreaInterfaceImpl **implements** AreaInterface {

// dao接口

**private** AreaDao areaDao = **new** AreaDaoImpl();

//如果不想让某个方法发布为服务，可以将exclude设置成true

@WebMethod(operationName="queryAreaByParentid",exclude=**false**)

@Override

**public** List<Area> queryArea(@WebParam(name="area")Area area) **throws** Exception {

// 调用dao接口查询数据库得到List<area>

List<Area> list = areaDao.queryArea(area.getParentid(),

area.getStart(), area.getEnd());

// 向客户端响应List<Area>

**return** list;

}

}

如果通过注解修改了serviceName、portType等信息，需要重新使用wsimport生成调用代码。

**public** **class** AreaClient {

**public** **static** **void** main(String[] args) **throws** MalformedURLException, Exception\_Exception {

//使用service类方法编写客户端

//wsdl地址

URL wsdlDocumentLocation = **new** URL("http://127.0.0.1:12345/area?wsdl");

//serviceName

QName serviceName = **new** QName("http://area.itcast.cn/", "AreaWebService");

//创建服务视图对象

Service service = Service.*create*(wsdlDocumentLocation, serviceName);

//生成 portType的代理对象

AreaServiceSoap areaServiceSoap = service.getPort(AreaServiceSoap.**class**);

//调用 areaInterfaceImplPort的方法相当于请求服务端

Area area = **new** Area();

area.setParentid("1.1.");

area.setStart(1);

area.setEnd(2);

List<Area> list = areaServiceSoap.queryAreaByParentid(area);

//服务端响应的 List<Area>

System.*out*.println(list);

}

}

使用注解前要确定好webservice接口信息（portType名称,serviceName名称等）。

### 使用注解注意

@WebMethod对所有非静态的公共方法对外暴露为服务.

对于静态方法或非public方法是不可以使用@WebMethod注解的.

对public方法可以使用**@WebMethod(exclude=true)**定义为非对外暴露的服务。

如果修改webservice的接口内容，比如namespace、portType，必须要重新生成客户端调用代码。

# CXF框架

## cxf介绍

cxf的apache的开源项目，是一个webservice开发框架。它支持多种协议，比如：SOAP1.1,1,2、[XML](http://baike.baidu.com/view/63.htm)/[HTTP](http://baike.baidu.com/view/9472.htm)、[REST](http://baike.baidu.com/view/1077487.htm)ful HTTP 或者 [CORBA](http://baike.baidu.com/view/153815.htm)。

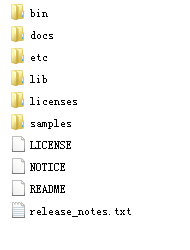
Cxf是基于SOA总线结构，依靠spring完成模块的集成，实现SOA方式。

灵活的部署：可以运行在Tomcat,Jboss,Jetty(内置),weblogic等web容器上面。

## 下载cxf

由于apache-cxf-3.0.5和spring3.2版本整合，从官网下载apache-cxf-3.0.5

http://cxf.apache.org/download.html



## 安装配置cxf

解压apache-cxf-3.0.5目录 。

CXF\_HOME=cxf的目录

Path = %JAVA\_HOME%\bin;%CXF\_HOME%\bin;

CLASSPATH=.;%CXF\_HOME%\lib\cxf-manifest.jar

## cxf入门程序

### 需求

服务端：使用cxf框架发布区域查询服务。

客户端：使用service类方式编写客户端。

客户端：使用cxf框架编写客户端。

### 服务端

#### 创建java工程

将cxf下的所有jar加入 工程。

将数据驱动包加入工程。

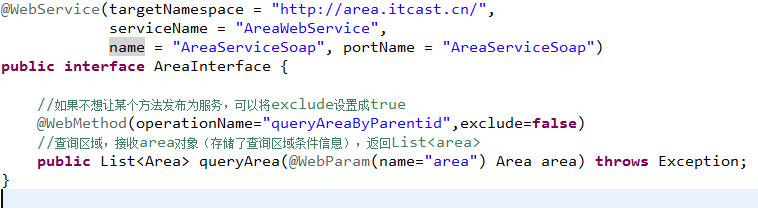
#### Dao

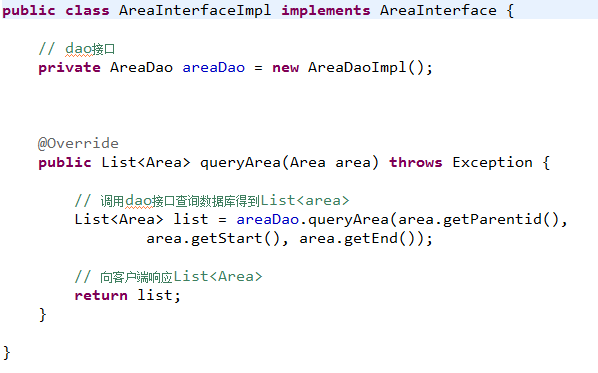
使用上边案例的代码。

#### Service

创建服务接口和服务类的方法同上边章节描述，编写SEI及SEI的实现。

**注意：与jaxws编程不同的是将@WebService注解加在SEI接口上边。**





#### 发布服务



### 客户端

#### 使用wsdl2java工具生成客户端调用代码

先让我们了解一下cxf的wsdl2java工具，它的功能就如同wsimport一样，可以生成客户端调用的代码。

在命令行执行：



#### 使用service类方式编写客户端

**public** **static** **void** main(String[] args) **throws** MalformedURLException, Exception\_Exception {

//使用service类方法编写客户端

//wsdl地址

URL wsdlDocumentLocation = **new** URL("http://127.0.0.1:12345/area?wsdl");

//serviceName

QName serviceName = **new** QName("http://area.itcast.cn/", "AreaWebService");

//创建服务视图对象

Service service = Service.*create*(wsdlDocumentLocation, serviceName);

//生成 portType的代理对象

AreaServiceSoap areaServiceSoap = service.getPort(AreaServiceSoap.**class**);

//调用 areaInterfaceImplPort的方法相当于请求服务端

Area area = **new** Area();

area.setParentid("1.1.");

area.setStart(1);

area.setEnd(2);

List<Area> list = areaServiceSoap.queryAreaByParentid(area);

//服务端响应的 List<Area>

System.*out*.println(list);

}

#### JaxwsProxyFactoryBean调用服务端

cxf提供 JaxwsProxyFactoryBean调用cxf发布的服务端。

在客户端工程加入cxf的jar包 。

**public** **class** AreaClientByJaxwsProxyFactoryBean {

**public** **static** **void** main(String[] args) **throws** MalformedURLException, Exception\_Exception {

//使用JaxwsProxyFactoryBean调用服务端

JaxWsProxyFactoryBean jaxWsProxyFactoryBean = **new** JaxWsProxyFactoryBean();

//指定调用服务的地址

jaxWsProxyFactoryBean.setAddress("http://127.0.0.1:12345/area");

//指定portType即SEI

jaxWsProxyFactoryBean.setServiceClass(AreaServiceSoap.**class**);

//生成 portType的代理对象

AreaServiceSoap areaServiceSoap = (AreaServiceSoap) jaxWsProxyFactoryBean.create();

//调用 areaInterfaceImplPort的方法相当于请求服务端

Area area = **new** Area();

area.setParentid("1.1.");

area.setStart(1);

area.setEnd(2);

List<Area> list = areaServiceSoap.queryAreaByParentid(area);

//服务端响应的 List<Area>

System.*out*.println(list);

}

}

# CXF与Spring整合

cxf依赖spring框架，一般企业开发需要将cxf和spring整合。

通过spring容器管理cxf发布的服务bean。

## 需求

cxf框架依赖spring框架，一般使用cxf框架需要和spring集成使用。

服务端使用cxf和spring整合发布区域查询服务。

客户端使用cxf和spring整合实现。

## 服务端

### 创建javaweb工程

cxf框架程序将运行在web容器中，比如tomcat，这里创建java web工程。

向工程中加入cxf的jar包、数据库驱动包等。

### Dao

dao接口使用上边的例子。

创建applicationContext-dao.xml文件配置dao接口，由spring容器统一管理。

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.2.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.2.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-3.2.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-3.2.xsd "*>

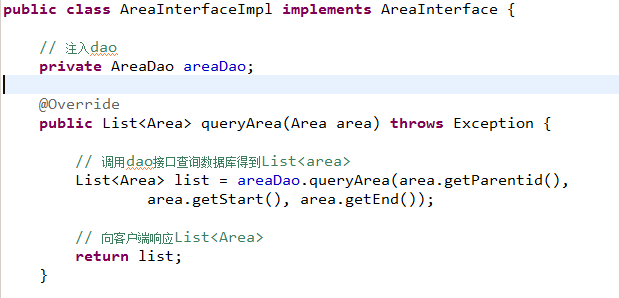
<!-- 配置 dao -->

<bean id=*"areaDao"* class=*"cn.itcast.webservice.area.dao.AreaDaoImpl"*/>

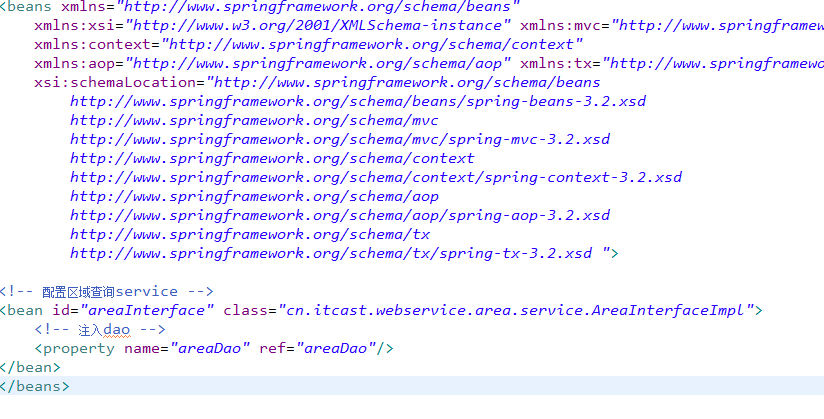
</beans>

### Service

在service接口实现类中注入dao：



创建applicationContext-service.xml，配置service。



### 通过cxf和spring整合发布服务

创建applicationContext-cxf.xml配置发布webservice服务。



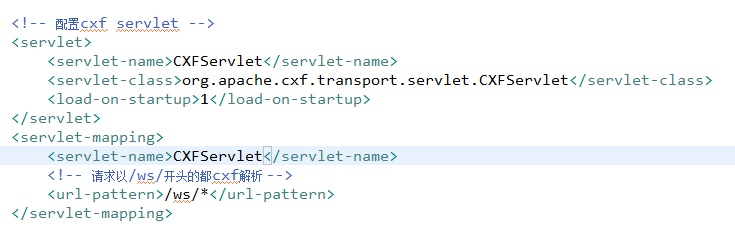
### web容器中加载spring容器



### 配置cxf的servlet

配置cxf的servlet，所有cxf的服务请求由servlet进行解析。

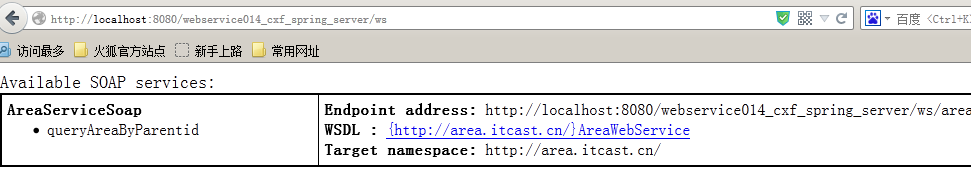
在web.xml中配置：



### 部署tomcat启动

访问tomcat的地址：

http://ip：port/工程名/ws/



## 客户端

客户端使用cxf和spring整合实现

### 创建java工程

向工程中加入cxf的jar包、spring的包。

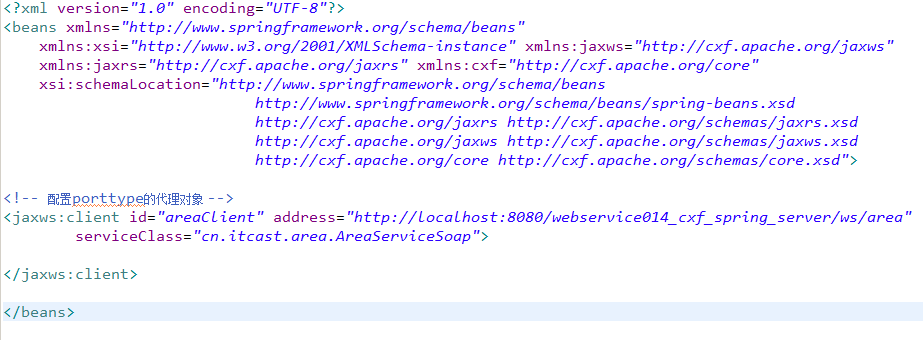
### 使用wsdl2java生成客户端调用代码

略

### 开发客户端

使用cxf和spring整合的标签<jaxws:client 代替jaxWsProxyFactoryBean

创建applicationContext-cxf.xml配置portType的代理对象：



客户端代码如下：



# CXF 发布RESTful服务

## 什么是RESTful

计算机连接互联网要获取互联网上的资源，一个资源对应一个url，计算机通过url获取资源。

REST，即Representational State Transfer的缩写。我对这个词组的翻译是"表现层状态转化"

rest就是一个理念，软件构建的模式、或说风格，遵循rest的理念设置一个架构称为RESTful架构。

* 表现层：

"表现层"其实指的是"资源"（Resources）的"表现层"，资源就是用一个url表示。

比如：

http://www.jd.com/item/001 对应一个商品资源

一个资源的表现层，能表现哪些样式即内容格式：

http://www.jd.com/item/001 该 url展示给用户可以是一个html、还可以是json串、还可以xml、还可以是一个pdf报表。

rest要求url规范化：

原来url：http://www.jd.com/item.jsp?type=101&id=100

rest要求规范后的url：http://www.jd.com/item/101/100 特点：将参数通过url传给jsp。

* **状态转化**

rest是对http协议的一个很好诠释。

http方法：get、post、put、delete

客户端要想操作服务端的资源，客户端要向服务端发送url，比如：http://www.jd.com/item/101/100

操作方法：查询、更新、插入、删除

原始对操作方法是定义很多不同的url：

比如：查询商品：http://www.jd.com/item.jsp？id=001

更新商品：http://www.jd.com/editItem.jsp?id=001&name=???

rest规范：

对服务端的操作通过设置http的方法来实现查询、更新、插入、删除

**GET用来获取资源，POST用来新建资源（也可以用于更新资源），PUT用来更新资源，DELETE用来删除资源。**

比如：

更新商品：

客户端发送url：http://www.jd.com/item/101/100

http的方法设置为：**PUT**

删除商品：

客户端发送url：http://www.jd.com/item/101/100

http的方法设置为：**DELETE**

**服务端判断http请求方法，如果是put进行更新，如果是delete执行删除操作。**

综合上面的解释，我们总结一下什么是RESTful架构：

　　（1）每一个URI代表一种资源；

实现：url使用rest风格

实现开发中常用。

　　（2）客户端和服务器之间，传递这种资源的某种表现层；

实现：客户端请求url，设置http头中的Accept（客户端可以解析的内容类型，application/json、application/xml）

服务端接收到请求，判断http头中的Accept，如果是application/json服务端将响应的数据以json格式输出。

实际开发中，要固定一种或几种常用的格式，比如：json、xml。

　　（3）客户端通过四个HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"。

实现：

客户端请求url，设置http的方法

服务端接收请求，根据http的方法判断操作类型：查询、更新、删除、插入。

实现开发中：用的不多。

## CXF发布rest服务

### 需求

使用CXF框架发布区域查询服务

区域查询的url实现rest风格

将区域查询结果输出成json及xml格式。

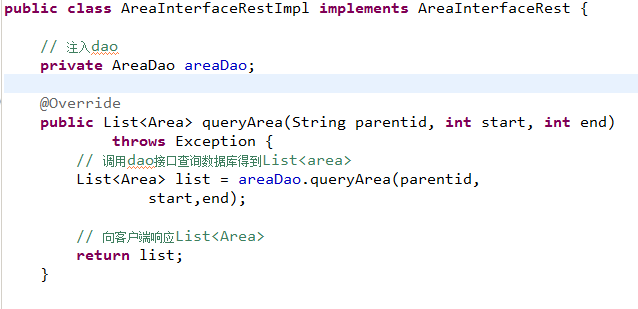
### Dao

使用前边章节的代码。

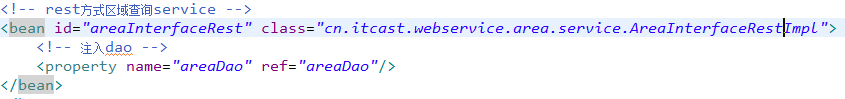
### service

**创建一个rest的service，实现区域查询。**

****

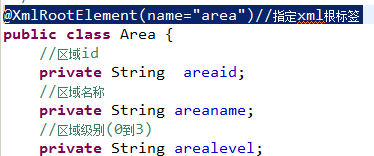
****

**在applicationContext-service.xml中配置：**

****

### 修改po类添加xml根标签

由于rest方法输出xml、json数据，rest方法返回值为List<Area>，在Area类中指定xml根标签



### 配置cxf 的rest服务

在applicationContext-cxf.xml配置文件中配置rest服务：



### 发布rest服务

使用cxf发布rest服务，所有的rest请求由cxf的servlet解析：

在web.xml中配置cxf的servlet：

<!-- cxf的servlet -->

<servlet>

<servlet-name>CXFServlet</servlet-name>

<servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

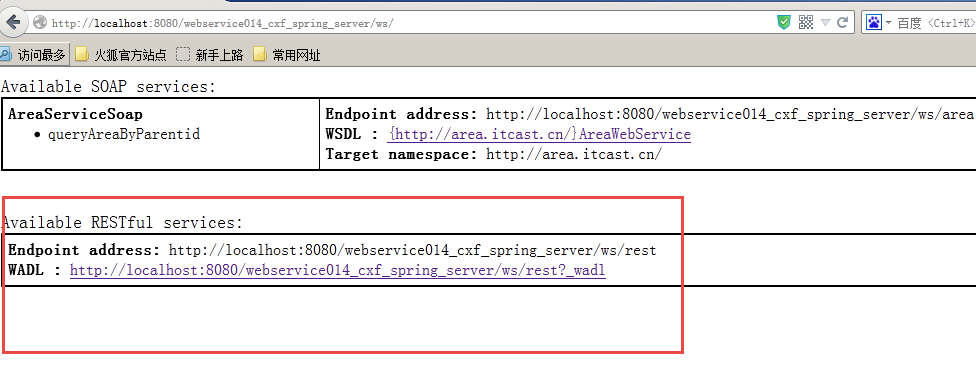
<servlet-mapping>

<servlet-name>CXFServlet</servlet-name>

<url-pattern>/ws/\*</url-pattern>

</servlet-mapping>

### 部署调试



rest服务地址：

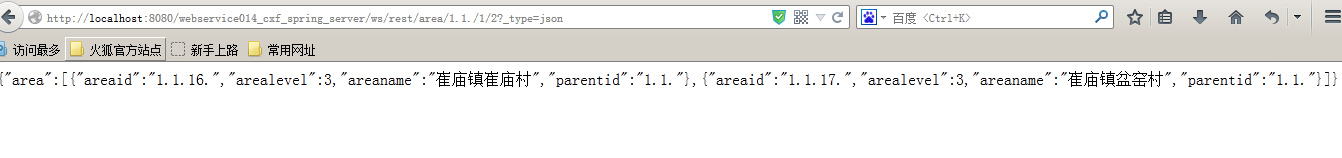
http://ip：port/工程路径/ws(cxf的servlet配置的

url-pattern)/rest(applicationContext-cxf.xml中配置rest地址)/rest接口中@Path指定的url

rest服务返回xml：



rest服务返回json：



# webservice扩展技术--httpclient

## httpclient介绍

HttpClient 是 Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包。

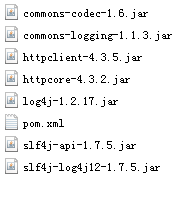
**webservice一般都基于http协议的，需要编写http客户端调用http服务端，企业开发中好多接口协议采用http协议，客户端模拟http请求实现http协议通信。**

## 下载httpclient

官方下载：http://hc.apache.org/httpcomponents-client-4.3.x/index.html

下载4.3版本。

httpclient及依赖包：



## httpclient测试

使用java程序实现http客户端（相当 于浏览器）向http 服务端发起http请求。

### 需求

使用httpclient向CXF发布rest服务发起请求，查询区域信息、添加区域

1、使用get方式发起请求查询区域信息

2、使用post方式发起请求添加区域信息

### get请求区域 查询

**public** **class** DoGet {

**public** **static** **void** main(String[] args) **throws** Exception {

//httpclient发送get请求查询区域信息

//创建httpclient客户端对象

CloseableHttpClient httpClient = HttpClients.*createDefault*();

//指定get请求的地址，向get请求中添加请求参数

URI uri =**new** URIBuilder("http://localhost:8080/webservice014\_cxf\_spring\_server/ws/rest/area/1.1./1/2")

.setParameter("\_type", "xml")//设置一个请求参数

// .setParameter(arg0, arg1)可以设置多个参数

.build();

//创建http的get请求

HttpGet httpGet = **new** HttpGet(uri);

//使用httpclient客户端对象发起http的get请求

CloseableHttpResponse response = **null**;

**try** {

//请求后获取响应结果

response = httpClient.execute(httpGet);

//如果response响应成功，代码是200

**if**(response.getStatusLine().getStatusCode() == 200){//响应成功

//从response中获取响应数据

String content = EntityUtils.*toString*(response.getEntity(), "utf-8");

System.*out*.println(content);

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally**{

//释放资源

**if**(response!=**null**){

response.close();

}

httpClient.close();

}

}

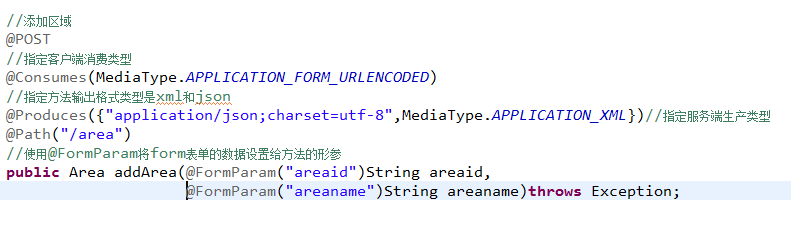
}

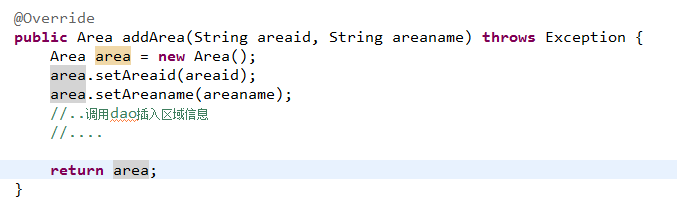
### post请求 添加区域

post是通过表单请求rest接口。

#### 开发post方式的添加区域接口

在cxf的rest服务工程中开发添加区域接口：





#### httpclient的post客户端

**public** **class** DoPost {

**public** **static** **void** main(String[] args) **throws** Exception {

// 创建httpclient客户端对象

CloseableHttpClient httpClient = HttpClients.*createDefault*();

// 指定post请求的地址

URI uri = **new** URIBuilder(

"http://localhost:8080/webservice014\_cxf\_spring\_server/ws/rest/area?\_type=xml")

.build();

// 创建http的post请求

HttpPost httpPost = **new** HttpPost(uri);

//设置post提交的表单=================

//定义表单提交的参数

List<NameValuePair> parameters = **new** ArrayList<NameValuePair>();

//设置表单参数,参数名为areaid,值是1010

parameters.add(**new** BasicNameValuePair("areaid", "1010"));

parameters.add(**new** BasicNameValuePair("areaname", "北京"));

//定义一个post表单

UrlEncodedFormEntity urlEncodedFormEntity = **new** UrlEncodedFormEntity(parameters, "utf-8");

//设置post提交的表单=================

//将表单设置到httpPost中

httpPost.setEntity(urlEncodedFormEntity);

// 使用httpclient客户端对象发起http的post请求

CloseableHttpResponse response = **null**;

**try** {

// 请求后获取响应结果

response = httpClient.execute(httpPost);

// 如果response响应成功，代码是200

**if** (response.getStatusLine().getStatusCode() == 200) {// 响应成功

// 从response中获取响应数据

String content = EntityUtils.*toString*(response.getEntity(),

"utf-8");

System.*out*.println(content);

}

} **catch** (Exception e) {

e.printStackTrace();

} **finally** {

// 释放资源

**if** (response != **null**) {

response.close();

}

httpClient.close();

}

}

}

# webservice扩展技术--hessian框架

## hessian介绍

hessian是基于http的远程调用的框架，使用这个框架开发远程调用应用（客户/服务器模式应用）。

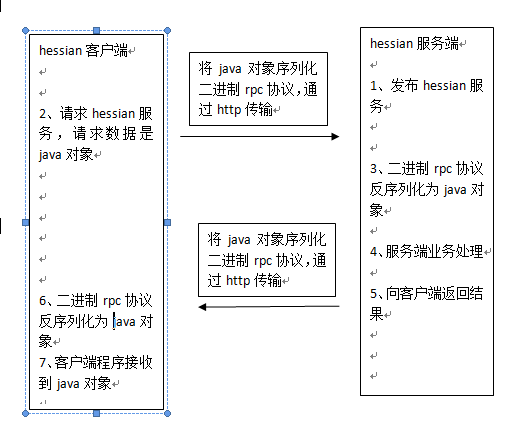
hessian，使用hessian自己的二进制rpc协议。

应用场景：

适用于传输数据量小的接口。是一种高效简洁的远程调用框架，它采用的是二进制RPC协议（Binary），具有轻量、传输量小、平台无关的特点，特别适合于目前网络带宽比较小的手机网络应用项目。

缺点：如果service层中返回的对象是复杂对象,使用它就会削弱Hessian的传输量小的优点，而且也会增加Hessian客户端的代码量。既然它是把对象序列化为二进制流的形式在http信道中传输，那么对于安全性高的应用不应该采用hessian。

## hessian传输原理



注意：

1、客户端调用服务端的服务接口需要将服务端的接口拷贝到客户端工程中，从而生成代理对象调用服务端。

2、服务接口所使用到的pojo对象需要实现序列化接口，并且也拷贝到客户端。

3、服务端的接口及pojo类拷贝到客户端要保证包名不变。

## 下载hessian

官方地址：http://hessian.caucho.com/

下载4.0.37版本。

hessian是跨平台，因为hessian框架为各种编写语言提供的hessian的类库。

## 需求

使用hessian发布区域查询服务

客户端使用hessian完成调用。

## 准备环境

使用hessian+spring整合开发。

加入hessian的jar、spring的jar、数据库驱动包、log日志包。

使用tomcat为web容器运行hessian程序。

## 服务端

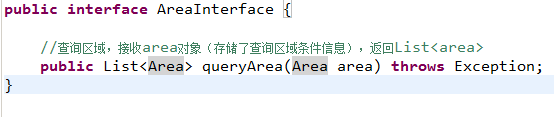
### Dao

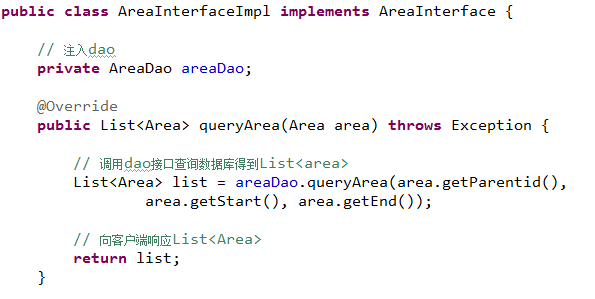
使用前边章节的代码。



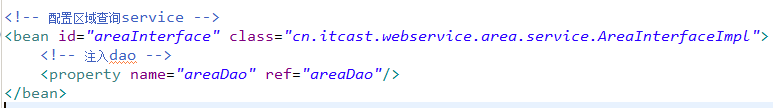


### service

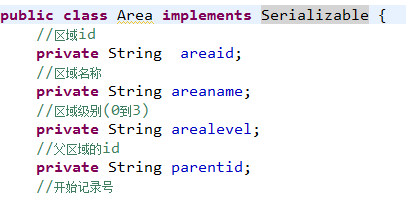




在applicationContext-service.xml中配置service接口：



### po实现Serializable接口



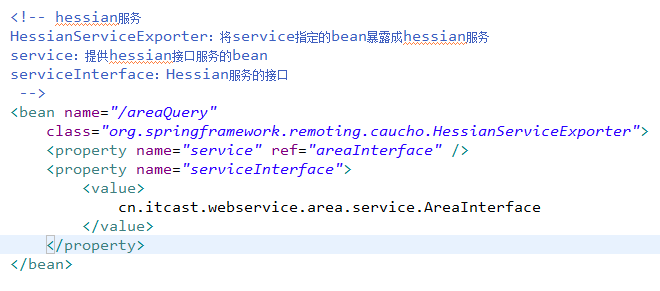
### spring容器在web容器中加载

在web.xml中加spring容器：



### 配置hessian服务

创建hessian-service.xml（spring的配置文件）配置hessian 服务：



### 配置hessian的servlet

通过http请求hessian服务接口由hessian的serlvet解析，hessian的servlet使用的是springmvc的servlet。



### 发布服务

部署tomcat ，hessian 服务地址：http://ip:port/工程名/hessian/areaQuery

hessian支持post请求。

## 客户端

实现方式：

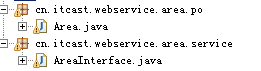
1、单独使用hessian

2、hessian 和spring整合实现

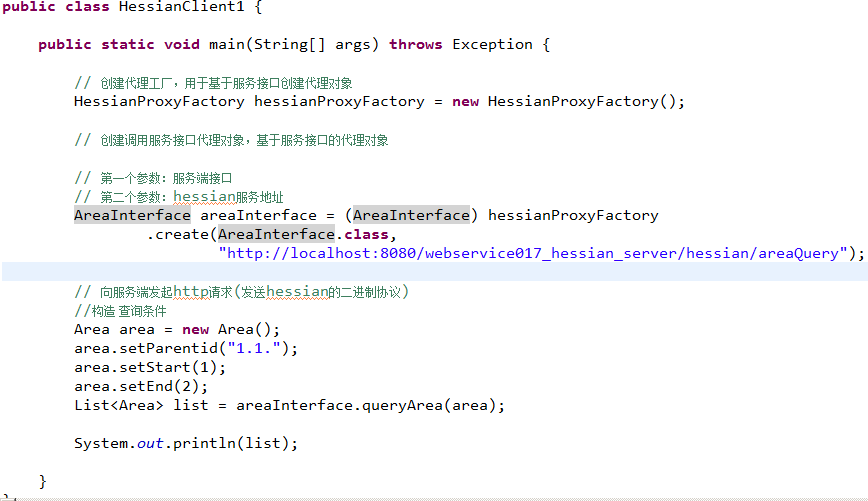
### 创建java工程

加入hessian和spring的包。

### 拷贝服务端的接口和po类



### 单独使用hessian开发客户端



### hessian和spring整合开发客户端

创建applicationContext-hessianclient.xml，配置调用服务接口代理对象：



#### 测试客户端

