1 课程回顾

- 什么是 webservice
  - 远程调用技术：系统和系统之间的调用，获取远程系统里的业务数据
  - Webservice 使用 http 传输 SOAP 协议的数据的一种远程调用技术
- Webservice 入门程序
  - 服务端
    - 第一步：创建 SEI 接口
    - 第二步：创建 SEI 实现类，加入@WebService
    - 第三步：发布服务，用 Endpoint 的 publish 方法，两个参数，1.服务地址；2.实现类
    - 第四步：测试服务，通过阅读使用说明书，确定接口、方法、参数和返回值存在，说明书服务发布成功
      - WSDL 地址：服务地址+?wsdl
      - WSDL 阅读方式：从下往上，service>binding>portType>方法、参数和返回值
  - 客户端
    - 第一步：使用 wsimport 命令生成客户端代码
    - 第二步：创建服务视图，从 service 的 name 属性获取
    - 第三步：获取服务类，从 portType 的 name 属性获取
    - 第四步：调用查询方法，从 operation 标签的 name 属性获取
  - Webservice 优缺点：
    - 优点：使用 http 发送数据，跨防火墙；使用 XML 封装数据，跨平台；支持面向对象
    - 缺点：使用 XML 封装，需要额外传输标签，性能较差
- Webservice 应用场景
  - 宏观
    - 软件集成和复用
  - 微观
    - 适用场景：
      - 不考虑性能，不考虑客户端类型，建议使用 webservice
      - 服务端已经确定 webservice，客户端只能使用 webservice
    - 不适用：
      - 考虑性能时，不建议使用 webservice
      - 同构程序不建议使用 webservice。
- Webservice 三要素
  - WSDL:
    - 定义：web 服务描述语言，他是 webservice 服务端的使用说明书，说明服务、接口、方法、参数和返回值，他是伴随服务发布成功，自动生成，无需编写
    - 文档结构：
      - Service
      - Binding

        - portType
        - message
        - types
    - 阅读方式：从下往上
- SOAP:
    - 定义：SOAP 即简单对象访问协议，他是使用 http 传输 XML 格式的数据，跨平台，跨防火墙，他不是 webservice 专有协议
    - Soap=http+xml
    - 协议格式：
        - 必有：envelope 和 body
        - 非必有：header 和 fault
    - SOAP1.1 和 1.2 区别
        - 相同点：
          都是用 POST 发送请求
          协议格式都相同：都有 envelope 和 body 标签
        - 不同点
          Content-type 不同：
          SOAP1.1，text/xml;charset=utf-8;SOAP1.2,application/soap+xml;charset=utf-8
          命名空间不同：

- Webservice 的四种客户端调用方式
    - 生成客户端的调用方式
    - Service 编程调用方式
    - HttpURLConnecton 调用方式
    - Ajax 调用方式
- 深入开发：用注解修改 WSDL 内容
    - @Webservice
    - @WebMethod
    - @WebParam
    - @WebResult
    - 修改完 WSDL 之后，需要重新生成客户端

CXF

2 课程安排：

- CXF 的介绍、安装和配置
- CXF 发布 SOAP 协议的服务
- CXF+Spring 整合发布 SOAP 的服务
- CXF 发布 REST 服务

- 什么是 REST
- CXF+Spring 整合发布 REST 服务
- 综合案例

3 CXF 介绍、安装和配置

3.1 CXF 介绍

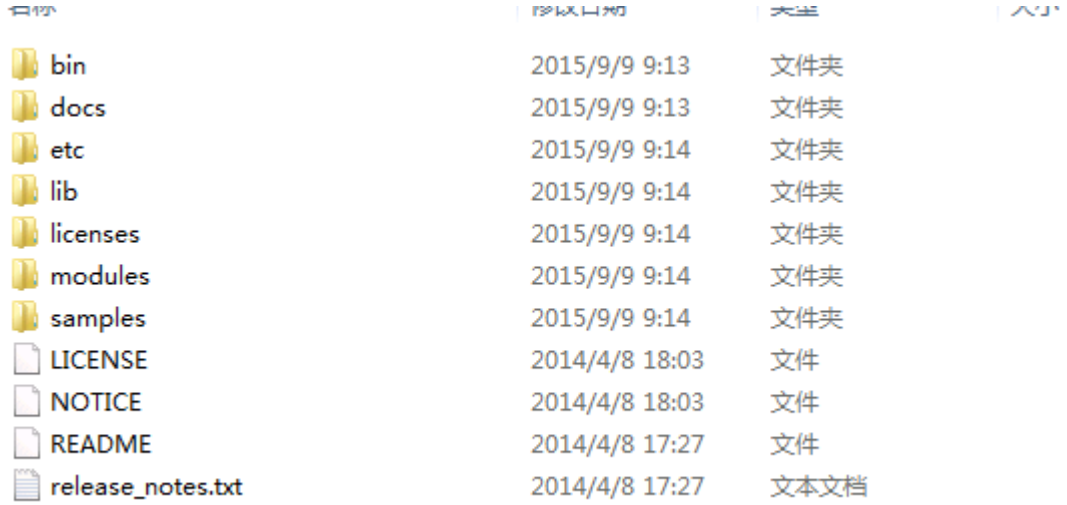
- CXF 是一个开源的 webservice 框架，提供很多完善功能，可以实现快速开发
- CXF 支持的协议：SOAP1.1/1.2，REST
- CXF 支持数据格式：XML，JSON（仅在 REST 方式下支持）

3.2 CXF 的安装和配置

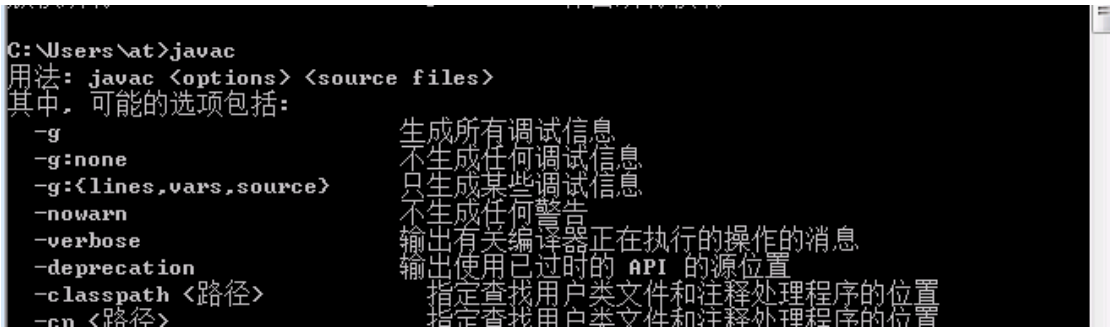
- 下载地址

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

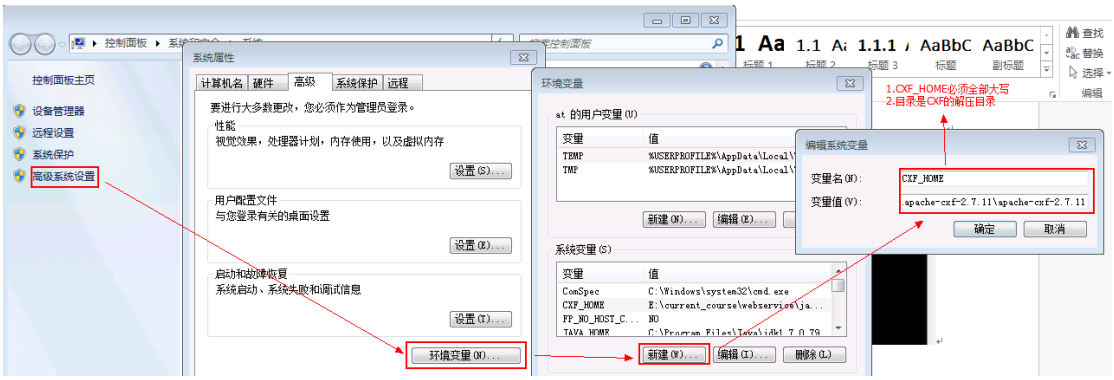
- 包结构介绍



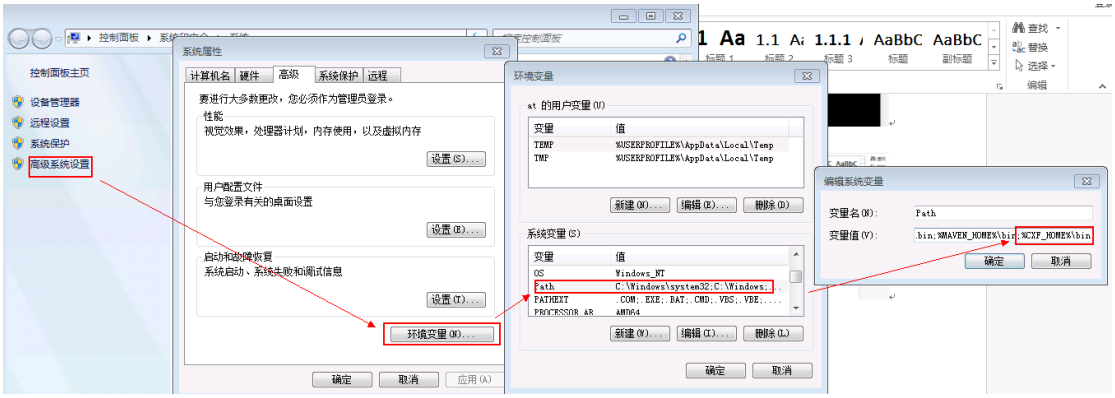
- 安装和配置
  - 第一步：安装 JDK，建议 1.7



➢ 第二步：解压 apache-cxf-2.7.11.zip 到指定目录，创建 CXF_HOME



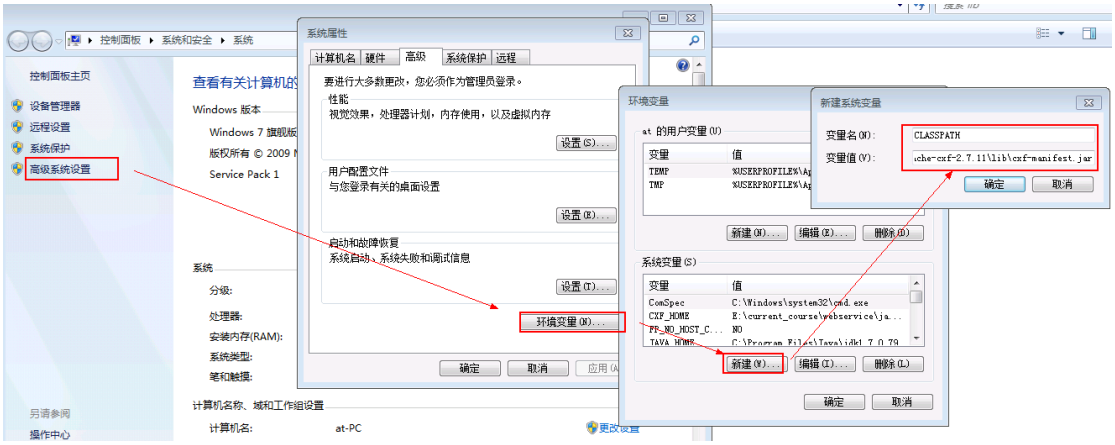
➢ 第三步：把 CXF_HOME 加入到 Path 路径下



➢ 第四步：测试，在 cmd 下加入 wsdl2java –h

```
C:\Users\at>wsdl2java -h
wsdl2java -fe|-frontend <front-end-name> -db|-databinding <data-binding-name> -w
v <wsdl-version> -p <[wsdl-namespace =]package-name>* -sn <service-name> -b <bin
ding-file-name>* -reserveClass <class-name>* -catalog <catalog-file-name> -d <ou
tput-directory> -compile -classdir <compile-classes-directory> -impl -server -cl
ient -clientjar <jar-file-name> -all -autoNameResolution -allowElementReferences
|-aer(=true) -defaultValues(=class-name-for-DefaultValueProvider) -ant -nexclude
 <schema-namespace [= java-package-name]>* -exsh ((true, false)) -noTypes -dns (
Default value is true) -dex ((true, false)) -validate([=all|basic|none]) -keep -
wsdlLocation <wsdlLocation> -xjc(<xjc-arguments>* -asyncMethods([=method1,method2
,...]>* -bareMethods([=method1,method2,...]>* -mimeMethods([=method1,method2,...
]>* -noAddressBinding -faultSerialVersionUID <fault-serialVersionUID> -encoding
<encoding> -exceptionSuper <exceptionSuper> -mark-generated -h|-?|-help -version
|-v -verbose|-V -quiet|-q|-Q -wsdlList <wsdlurl>

Options:

   -fe|-frontend <front-end-name>
                 Specifies the front end  (defaults to JAXWS)
```

- 如果不想使用 IDE（比如 Eclipse），需要在环境变量下配置如下信息



4 CXF 发布 SOAP 协议的服务

4.1 需求

服务端：发布服务，接收客户端的城市名，返回天气数据给客户端
客户端：发送城市名给服务端，接收服务端的响应信息，打印

4.2 实现

4.2.1 服务端

开发步骤：
第一步：导入 Jar 包
第二步：创建 SEI 接口，要加入@WebService

```
package cn.itcast.ws.cxf.server;

import javax.jws.WebService;

/**
 *
 * <p>Title: WeatherInterface.java</p>
 * <p>Description:SEI接口</p>
 * <p>Company: www.itcast.com</p>
```

```
 * @author  传智.at
 * @date    2015年11月27日上午9:47:43
 * @version 1.0
 */
@WebService
public interface WeatherInterface {

    public String queryWeather(String cityName);
}
```

第三步：创建 SEI 实现类

```
package cn.itcast.ws.cxf.server;

/**
 *
 * <p>Title: WeatherInterfaceImpl.java</p>
 * <p>Description:SEI实现类</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日上午9:50:59
 * @version 1.0
 */
public class WeatherInterfaceImpl implements WeatherInterface
{

    @Override
    public String queryWeather(String cityName) {
        System.out.println("from client..."+cityName);
        if("北京".equals(cityName)){
            return "冷且霾";
        } else {
            return "暖且晴";
        }
    }

}
```

第四步：发布服务，JaxWsServerFactoryBean 发布，设置 3 个参数，1.服务接口；2.服务实现类；3.服务地址；

endpoint 仅支持发布实现类，JaxWsServerFactoryBean 支持发布接口。

```java
package cn.itcast.ws.cxf.server;

import org.apache.cxf.jaxws.JaxWsServerFactoryBean;
import
org.apache.cxf.tools.java2wsdl.processor.internal.jaxws.generator.JaxwsServerGenerator;

/**
 *
 * <p>Title: WeatherServer.java</p>
 * <p>Description:服务端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日上午9:51:36
 * @version 1.0
 */
public class WeatherServer {

    public static void main(String[] args) {
        //JaxWsServerFactoryBean发布服务
        JaxWsServerFactoryBean jaxWsServerFactoryBean = new
JaxWsServerFactoryBean();
        //设置服务接口

    jaxWsServerFactoryBean.setServiceClass(WeatherInterface.class);
        //设置服务实现类
        jaxWsServerFactoryBean.setServiceBean(new
WeatherInterfaceImpl());
        //设置服务地址

    jaxWsServerFactoryBean.setAddress("http://127.0.0.1:12345/weather");
        //发布
        jaxWsServerFactoryBean.create();
    }
}
```

第五步：测试服务是否发布成功，阅读使用说明书，确定关键点
如果在 CXF 发布的服务下，直接访问服务地址，会如下异常

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<soap:Fault>
       <faultcode>soap:Server</faultcode>
      ▼<faultstring>
          No binding operation info while invoking unknown method with params unknown.
       </faultstring>
     </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

此时直接访问使用说明书地址即可

4.2.1.1 发布 SOAP1.2 的服务端

- 在接口上加入如下注解：

@BindingType(SOAPBinding.SOAP12HTTP_BINDING)

- 重新发布服务端

4.2.1.2 拦截器

- 原理：
  - 拦截器可以拦截请求和响应
  - 拦截器可以有多个
  - 拦截器可以根据需要自定义

- 使用
  - 拦截器必须加到服务端，在服务端发布之前

  - 获取拦截器列表，将自己的拦截器加入列表中

```
//加入拦截器
jaxWsServerFactoryBean.getInInterceptors().add(new LoggingInInterceptor());
jaxWsServerFactoryBean.getOutInterceptors().add(new LoggingOutInterceptor())

//发布
jaxWsServerFactoryBean.create();
```

4.2.2 客户端

第一步：生成客户端代码

- Wsdl2java 命令是 CXF 提供的生成客户端的工具，他和 wsimport 类似，可以根据 WSDL 生成客户端代码
- Wsdl2java 常用参数：
  -d，指定输出目录
  -p，指定包名，如果不指定该参数，默认包名是 WSDL 的命名空间的倒序
- Wsdl2java 支持 SOAP1.1 和 SOAP1.2

第二步：使用说明书，使用生成代码调用服务端

JaxWsProxyFactoryBean 调用服务端，设置 2 个参数，1.设置服务接口；2.设置服务地址

```
package cn.itcast.cxf.client;

import org.apache.cxf.jaxws.JaxWsProxyFactoryBean;

import cn.itcast.cxf.weather.WeatherInterface;

/**
 *
 * <p>Title: WeatherClient.java</p>
 * <p>Description:客户端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日上午10:12:24
 * @version 1.0
 */
public class WeatherClient {

    public static void main(String[] args) {
        //JaxWsProxyFactoryBean调用服务端
        JaxWsProxyFactoryBean jaxWsProxyFactoryBean = new
JaxWsProxyFactoryBean();
        //设置服务接口

    jaxWsProxyFactoryBean.setServiceClass(WeatherInterface.cla
ss);
        //设置服务地址

    jaxWsProxyFactoryBean.setAddress("http://127.0.0.1:12345/w
eather");
        //获取服务接口实例
        WeatherInterface weatherInterface =
jaxWsProxyFactoryBean.create(WeatherInterface.class);
        //调用查询方法
```

```
        String weather = weatherInterface.queryWeather("保定");
        System.out.println(weather);
    }
}
```

5 CXF+Spring 整合发布 SOAP 协议的服务

5.1 服务端

开发步骤：
第一步：创建 web 项目（引入 jar 包）

第二步：创建 SEI 接口

第三步：创建 SEI 实现类

第四步：配置 spring 配置文件，applicationContext.xml，用<jaxws:server 标签发布服务，设置 1.服务地址；2.设置服务接口；3 设置服务实现类

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jaxws="http://cxf.apache.org/jaxws"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
xmlns:cxf="http://cxf.apache.org/core"
    xsi:schemaLocation="http://www.springframework.org/schema/
beans

http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://cxf.apache.org/jaxrs
http://cxf.apache.org/schemas/jaxrs.xsd
                        http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd
                        http://cxf.apache.org/core
http://cxf.apache.org/schemas/core.xsd">
    <!-- <jaxws:server发布SOAP协议的服务 ， 对
JaxWsServerFactoryBean类封装-->
    <jaxws:server address="/weather"
serviceClass="cn.itcast.ws.cxf.server.WeatherInterface">
        <jaxws:serviceBean>
```

```
            <ref bean="weatherInterface"/>
        </jaxws:serviceBean>
    </jaxws:server>
    <!-- 配置服务实现类 -->
    <bean name="weatherInterface"
class="cn.itcast.ws.cxf.server.WeatherInterfaceImpl"/>
</beans>
```

第五步：配置 web.xml，配置 spring 配置文件地址和加载的 listener，配置 CXF 的 servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>ws_2_cxf_spring_server</display-name>

  <!-- 设置spring的环境 -->
  <context-param>
    <!--contextConfigLocation是不能修改的  -->
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-
value>
  </context-param>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</
listener-class>
  </listener>

  <!-- 配置CXF的Servlet -->
  <servlet>
    <servlet-name>CXF</servlet-name>
    <servlet-
class>org.apache.cxf.transport.servlet.CXFServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CXF</servlet-name>
    <url-pattern>/ws/*</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
```

```
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

第六步：部署到 tomcat 下，启动 tomcat

第七步：测试服务，阅读使用说明书

WSDL 地址规则：http://ip:端口号/项目名称/servlet 拦截路径/服务名称?wsdl

5.1.1 拦截器配置

配置 applicationContext.xml 中。

```
<jaxws:server address="/weather" serviceClass="cn.itcast.ws.cxf.server.WeatherInterface">
    <jaxws:serviceBean>
        <ref bean="weatherInterface"/>
    </jaxws:serviceBean>

    <!-- 配置拦截器 -->
    <jaxws:inInterceptors>
        <ref bean="inIntercepter"/>
    </jaxws:inInterceptors>
    <jaxws:outInterceptors>
        <ref bean="outIntercepter"/>
    </jaxws:outInterceptors>
</jaxws:server>
<!-- 配置拦截器的bean -->
<bean name="inIntercepter" class="org.apache.cxf.interceptor.LoggingInInterceptor"/>
<bean name="outIntercepter" class="org.apache.cxf.interceptor.LoggingOutInterceptor"/>
```

5.1.2 Endpoint 标签发布服务

<jaxws:endpoint>标签

```
package cn.itcast.ws.cxf.server;

import javax.jws.WebService;

/**
 *
```

```
 * <p>Title: HelloWorld.java</p>
 * <p>Description:简单类</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日上午11:11:10
 * @version 1.0
 */
@WebService
public class HelloWorld {
    public String sayHello(String name){
        return "hello,"+name;
    }

}
```

5.2 客户端

开发步骤：
第一步：引入 jar 包

第二步：生成客户端代码

第三步：配置 spring 配置文件，applicationContent.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jaxws="http://cxf.apache.org/jaxws"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
xmlns:cxf="http://cxf.apache.org/core"
    xsi:schemaLocation="http://www.springframework.org/schema/
beans

http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://cxf.apache.org/jaxrs
http://cxf.apache.org/schemas/jaxrs.xsd
                        http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd
                        http://cxf.apache.org/core
```

```
http://cxf.apache.org/schemas/core.xsd">
    <!-- <jaxws:client实现客户端 ， 对JaxWsProxyFactoryBean类封装-->
    <jaxws:client id="weatherClient" address="http://127.0.0.1:8080/ws_2_cxf_spring_server/ws/weather" serviceClass="cn.itcast.cxf.weather.WeatherInterface"/>
</beans>
```

第四步：从 spring 上下文件获取服务实现类

第五步：调用查询方法，打印

```
package cn.itcast.cxf.client;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import cn.itcast.cxf.weather.WeatherInterface;

public class WeatherClient {

    public static void main(String[] args) {
        //初始化spring的上下文
        ApplicationContext context = new ClassPathXmlApplicationContext("classpath:applicationContext.xml");
        WeatherInterface  weatherInterface = (WeatherInterface) context.getBean("weatherClient");
        String weather = weatherInterface.queryWeather("保定");
        System.out.println(weather);
    }
}
```

6 上午课程回顾

- CXF 的介绍、安装和配置
  - CXF 是一个开源的 webservice 的框架，提供很多成熟的功能，实现快速开发

- CXF 支持的协议：SOAP1.1/1.2，REST
- CXF 支持的数据格式：XML，JSON

- 安装和配置
  - 安装 JDK，建议 1.7
  - 解压 cxf 压缩包到指定目录，配置 CXF_HOME
  - CXF_HOME 加入 Path 中
  - 测试成功，在 cmd 中输入 wsdl2java –h

- CXF 发布 SOAP 协议的服务
  - 服务端
    - 第一步：引入 jar 包
    - 第二步：创建 SEI 接口，加入@WebService 注解
    - 第三步：创建 SEI 实现类
    - 第四步：发布服务，JaxWsServerFactoryBean 发布服务，设置 3 个参数，1.服务接口；2.服务实现类；3.服务地址
    - 第五步：测试服务
  - 客户端
    - 第一步：引入 jar 包
    - 第二步：生成客户端代码
    - 第三步：JaxWSProxyFactoryBean 调用服务端，设置 2 个参数，1.服务接口；2.服务地址
    - 第四步：获取实现类的实例，调用查询方法

- CXF+Spring 整合发布 SOAP 协议的服务
  - 服务端
    - 第一步：创建 web 项目（引入 jar 包）
    - 第二步：创建 SEI 接口
    - 第三步：创建 SEI 实现类
    - 第四步：配置 Spring 配置文件，applicationContext.xml，<jaxws:server,
    - 第五步：配置 web.xml，spring 配置文件，listener，cxf 的 servlet
    - 第六步：部署 tomcat 下，启动 tomcat
    - 第七步：测试服务是否发布成功
      - WSDL 地址规则：http://ip:端口号/项目名称/servlet 拦截路径/服务名称?wsdl
  - 客户端
    - 第一步：引入 jar 包
    - 第二步：生成客户端
    - 第三步：配置 spring 的配置文件，applicationContext.xml，<jaxws:client>
    - 第四步：初始化 spring 上下文，获取接口实现类，调用查询方法

7 CXF 发布 REST 的服务

7.1 什么是 REST

- 定义：REST 就是一种编程风格，它可以精确定位网上资源（服务接口、方法、参数）
- REST 支持数据格式：XML、JSON
- REST 支持发送方式：GET，POST

7.2 需求

- 第一个：查询单个学生
- 第二个：查询多个学生

7.3 实现

7.3.1 服务端

开发步骤：
第一步：导入 jar 包

第二步：创建学生 pojo 类，要加入@ XmlRootElement

```
package cn.itcast.ws.rest.pojo;

import java.util.Date;

import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * <p>Title: Student.java</p>
 * <p>Description:学生实体类</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日下午3:00:17
 * @version 1.0
 */
```

```java
@XmlRootElement(name="student")//@XmlRootElement可以实现对象和XML数据之间的转换
public class Student {

    private long id;

    private String name;

    private Date birthday;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Date getBirthday() {
        return birthday;
    }

    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

}
```

第三步：创建 SEI 接口

```java
package cn.itcast.ws.rest.server;

import java.util.List;

import javax.jws.WebService;
import javax.ws.rs.GET;
```

```java
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import cn.itcast.ws.rest.pojo.Student;

/**
 *
 * <p>Title: StudentInterface.java</p>
 * <p>Description:学生接口</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日下午3:03:08
 * @version 1.0
 */
@WebService
@Path("/student")//@Path("/student")就是将请求路径中的
“/student”映射到接口上
public interface StudentInterface {

    //查询单个学生
    @GET//指定请求方式，如果服务端发布的时候指定的是GET（POST），
那么客户端访问时必须使用GET（POST）
    @Produces(MediaType.APPLICATION_XML)//指定服务数据类型
    @Path("/query/{id}")//@Path("/query/{id}")就是将“/query”映
射到方法上，“{id}”映射到参数上，多个参数，以“/”隔开，放到“{}”中
    public Student query(@PathParam("id")long id);

    //查询多个学生
    @GET//指定请求方式，如果服务端发布的时候指定的是GET（POST），
那么客户端访问时必须使用GET（POST）
    @Produces(MediaType.APPLICATION_XML)//指定服务数据类型
    @Path("/queryList/{name}")//@Path("/queryList/{name}")就是
将“/queryList”映射到方法上，“{name}”映射到参数上，多个参数，以“/”
隔开，放到“{}”中
    public List<Student> queryList(@PathParam("name")String
name);

}
```

第四步：创建 SEI 实现类

```
package cn.itcast.ws.rest.server;
```

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import cn.itcast.ws.rest.pojo.Student;

/**
 *
 * <p>Title: StudentInterfaceImpl.java</p>
 * <p>Description:学生的实现类</p>
 * <p>Company: www.itcast.com</p>
 * @author    传智.at
 * @date       2015 年 11 月 27 日下午 3:12:54
 * @version 1.0
 */
public class StudentInterfaceImpl implements StudentInterface {

	@Override
	public Student query(long id) {
		Student st = new Student();
		st.setId(110);
		st.setName("张三");
		st.setBirthday(new Date());
		return st;
	}

	@Override
	public List<Student> queryList(String name) {

		Student st = new Student();
		st.setId(110);
		st.setName("张三");
		st.setBirthday(new Date());

		Student st2 = new Student();
		st2.setId(120);
		st2.setName("李四");
		st2.setBirthday(new Date());

		List<Student> list = new ArrayList<Student>();
		list.add(st);
		list.add(st2);
		return list;
```

```
    }

}
```

第五步：发布服务，JAXRSServerFactoryBean 发布服务，3 个参数，1：服务实现类；2.设置资源类；3.设置服务地址

```
package cn.itcast.ws.rest.server;

import org.apache.cxf.jaxrs.JAXRSServerFactoryBean;

import cn.itcast.ws.rest.pojo.Student;

/**
 *
 * <p>Title: StudentServer.java</p>
 * <p>Description:服务端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日下午3:16:06
 * @version 1.0
 */
public class StudentServer {

    public static void main(String[] args) {
        //JAXRSServerFactoryBean发布REST的服务
        JAXRSServerFactoryBean jaxRSServerFactoryBean = new
JAXRSServerFactoryBean();

        //设置服务实现类
        jaxRSServerFactoryBean.setServiceBean(new
StudentInterfaceImpl());
        //设置资源类，如果有多个资源类，可以以“,”隔开。

    jaxRSServerFactoryBean.setResourceClasses(StudentInterface
Impl.class);
        //设置服务地址

    jaxRSServerFactoryBean.setAddress("http://127.0.0.1:12345/
user");
        //发布服务
        jaxRSServerFactoryBean.create();
    }
}
```

第六步：测试服务

http://127.0.0.1:12345/user/student/query/110 查询单个学生，返回 XML 数据

```
<student>
<birthday>2015-11-27T15:22:14.240+08:00</birthday>
<id>110</id>
<name>张三</name>
</student>
```

http://127.0.0.1:12345/user/student/queryList/110?_type=json 查询多个学生，返回 JSON

```
{"student":[{"birthday":"2015-11-27T15:24:21.565+08:00","id":110,"name":"张三"},{"birthday":"2015-11-27T15:24:21.565+08:00","id":120,"name":"李四"}]}
```

http://127.0.0.1:12345/user/student/queryList/110?_type=xml 查询多个学生，返回 XML

```
<students>
<student>
<birthday>2015-11-27T15:30:33.754+08:00</birthday>
<id>110</id>
<name>张三</name>
</student>
<student>
<birthday>2015-11-27T15:30:33.754+08:00</birthday>
<id>120</id>
<name>李四</name>
</student>
</students>
```

如果服务端发布时指定请求方式是 GET（POST），客户端必须使用 GET（POST）访问服务端，否则会报如下异常

```
十一月 27, 2015 3:26:37 下午 org.apache.cxf.jaxrs.utils.JAXRSUtils findTargetMethod
警告: No operation matching request path "/user/student/query/110" is found, Relative Path: /query/110, HTTP Method: GET,
十一月 27, 2015 3:26:37 下午 org.apache.cxf.jaxrs.impl.WebApplicationExceptionMapper toResponse
警告: javax.ws.rs.ClientErrorException
        at org.apache.cxf.jaxrs.utils.SpecExceptions.toHttpException(SpecExceptions.java:110)
        at org.apache.cxf.jaxrs.utils.ExceptionUtils.toHttpException(ExceptionUtils.java:149)
        at org.apache.cxf.jaxrs.utils.JAXRSUtils.findTargetMethod(JAXRSUtils.java:477)
```

如果在同一方法上同时指定 XML 和 JSON 媒体类型，在 GET 请求下，默认返回 XML，在 POST 请求下，默认返回 JSON

7.3.2 客户端

可以自学一下 httpclient

http://hc.apache.org/httpclient-3.x/

```
package cn.itcast.cxf.client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

/**
 *
 * <p>Title: HttpClient.java</p>
 * <p>Description:HttpURLConnection 调用方式</p>
 * <p>Company: www.itcast.com</p>
 * @author    传智.at
 * @date      2015 年 11 月 26 日下午 3:58:57
 * @version 1.0
 */
public class HttpClient {

    public static void main(String[] args) throws IOException {
        //第一步：创建服务地址，不是 WSDL 地址
        URL url = new URL("http://127.0.0.1:12345/user/student/query/110");
        //第二步：打开一个通向服务地址的连接
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        //第三步：设置参数
        //3.1 发送方式设置：POST 必须大写
        connection.setRequestMethod("POST");
        //3.2 设置数据格式：content-type
        //3.3 设置输入输出，因为默认新创建的 connection 没有读写权限，
        connection.setDoInput(true);

        //第五步：接收服务端响应，打印
        int responseCode = connection.getResponseCode();
```

```
            if(200 == responseCode){//表示服务端响应成功
                InputStream is = connection.getInputStream();
                InputStreamReader isr = new InputStreamReader(is);
                BufferedReader br = new BufferedReader(isr);

                StringBuilder sb = new StringBuilder();
                String temp = null;
                while(null != (temp = br.readLine())){
                    sb.append(temp);
                }
                System.out.println(sb.toString());
                //dom4j 解析返回数据，课下作业
                is.close();
                isr.close();
                br.close();
            }
        }

}
```

8 CXF+Spring 整合发布 REST 的服务

8.1 服务端

开发步骤：
第一步：创建 web 项目（引入 jar 包）

第二步：创建 POJO 类

第三步：创建 SEI 接口

第四步：创建 SEI 实现类

第五步：配置 Spring 配置文件,applicationContext.xml，<jaxrs:server，设置 1.服务地址；2.服务实现类

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jaxws="http://cxf.apache.org/jaxws"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
```

```
xmlns:cxf="http://cxf.apache.org/core"
   xsi:schemaLocation="http://www.springframework.org/schema/
beans

http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://cxf.apache.org/jaxrs
http://cxf.apache.org/schemas/jaxrs.xsd
                        http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd
                        http://cxf.apache.org/core
http://cxf.apache.org/schemas/core.xsd">
   <!-- <jaxrs:server发布REST的服务 ， 对JAXRSServerFactoryBean
类封装-->
   <jaxrs:server address="/user">
      <jaxrs:serviceBeans>
         <ref bean="studentInterface"/>
      </jaxrs:serviceBeans>
   </jaxrs:server>

   <!-- 配置服务实现类 -->
   <bean name="studentInterface"
class="cn.itcast.ws.rest.server.StudentInterfaceImpl"/>
</beans>
```

第六步：配置 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>ws_2_cxf_spring_server</display-name>

  <!-- 设置spring的环境 -->
  <context-param>
    <!--contextConfigLocation是不能修改的  -->
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-
value>
  </context-param>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</
listener-class>
```

```
  </listener>

  <!-- 配置CXF的Servlet -->
  <servlet>
    <servlet-name>CXF</servlet-name>
    <servlet-
class>org.apache.cxf.transport.servlet.CXFServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CXF</servlet-name>
    <url-pattern>/ws/*</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

第七步：部署到 tomcat 下，启动 tomcat

第八步：测试服务

REST 服务的使用说明书地址：

http://127.0.0.1:8080/ws_4_cxf_rest_spring_server/ws/user?_wadl

8.2 客户端

```
<!doctype html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script type="text/javascript">
    function queryStudent(){
        //创建 XMLHttpRequest 对象
        var xhr = new XMLHttpRequest();
```

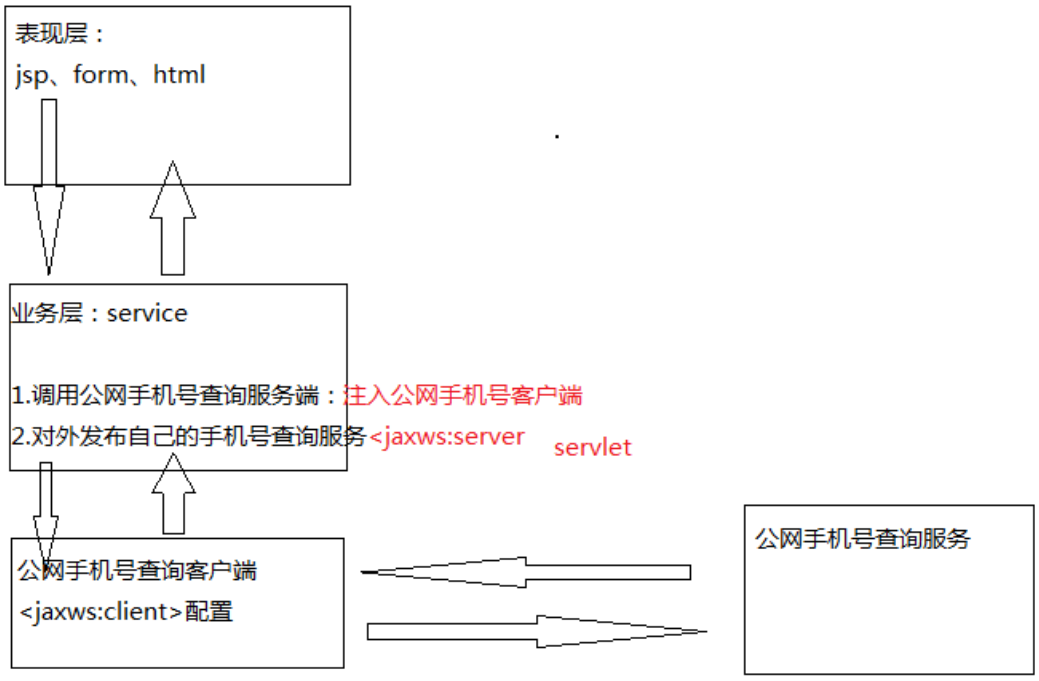
```
            //打开连接

        xhr.open("get","http://127.0.0.1:8080/ws_4_cxf_rest_spring_server/ws/user/student/queryList/110?_type=json",true);
            //设置回调函数
            xhr.onreadystatechange=function(){
                //判断是否发送成功和判断服务端是否响应成功
                if(4 == xhr.readyState && 200 == xhr.status){
                    alert(eval("("+xhr.responseText+")").student[0].name);
                }
            }
            //发送数据
            xhr.send(null);
        }
    </script>
  </head>
  <body>
    <input type="button" value="查询" onclick="javascript:queryStudent();"/>
  </body>
</html>
```

9 综合案例

9.1 需求：

- 集成公网手机号归属地查询服务
- 对外发布自己的手机号归属地查询服务
- 提供查询界面

9.2 分析



9.3 实现

开发步骤：
第一步：创建 web 项目（引入 jar 包）

第二步：生成公网客户端代码

第三步：创建 SEI 接口

```
package cn.itcast.mobile.server;

import javax.jws.WebService;

/**
 *
 * <p>Title: MobileInterface.java</p>
 * <p>Description:SEI接口</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月27日下午4:21:24
```

```
 * @version 1.0
 */
@WebService
public interface MobileInterface {

    public String queryMobile(String phoneNum);
}
```

第四步：创建 SEI 实现类

```
package cn.itcast.mobile.server;

import cn.itcast.mobile.MobileCodeWSSoap;

public class MobileInterfaceImpl implements MobileInterface {

    private MobileCodeWSSoap mobileClient;

    @Override
    public String queryMobile(String phoneNum) {
        return mobileClient.getMobileCodeInfo(phoneNum, "");
    }

    public MobileCodeWSSoap getMobileClient() {
        return mobileClient;
    }

    public void setMobileClient(MobileCodeWSSoap mobileClient)
{
        this.mobileClient = mobileClient;
    }

}
```

第五步：创建 queryMobile.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-
8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
```

```
charset=utf-8">
<title>手机号归属查询网站</title>
</head>
<body>
    <form action="queryMobile.action" method="post">
        手机号归属地查询：<input type="text"
name="phoneNum"/><input type="submit" value="查询"/><br/>
        查询结果：${result}
    </form>
</body>
</html>
```

第六步：创建 MobileServlet.java

```
package cn.itcast.mobile.server.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.context.ApplicationContext;
import
org.springframework.web.context.support.WebApplicationContext
Utils;

import cn.itcast.mobile.server.MobileInterface;

/**
 *
 * <p>
 * Title: MobileServlet.java
 * </p>
 * <p>
 * Description:MobileServlet
 * </p>
 * <p>
 * Company: www.itcast.com
 * </p>
 *
 * @author 传智.at
 * @date 2015年11月27日下午4:42:23
```

```
 * @version 1.0
 */
public class MobileServlet extends HttpServlet {

    private MobileInterface mobileServer;

    public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        String phoneNum = request.getParameter("phoneNum");
        if(null != phoneNum && !"".equals(phoneNum)){
            ApplicationContext context =
WebApplicationContextUtils.getWebApplicationContext(this.getS
ervletContext());
            mobileServer = (MobileInterface)
context.getBean("mobileServer");
            String result = mobileServer.queryMobile(phoneNum);
            request.setAttribute("result", result);
        }
        request.getRequestDispatcher("/WEB-
INF/jsp/queryMobile.jsp").forward(request, response);
    }

    public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        this.doGet(request, response);
    }

}
```

第七步：配置 spring 配置文件，applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jaxws="http://cxf.apache.org/jaxws"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
xmlns:cxf="http://cxf.apache.org/core"
    xsi:schemaLocation="http://www.springframework.org/schema/
beans
```

```
http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://cxf.apache.org/jaxrs
http://cxf.apache.org/schemas/jaxrs.xsd
                        http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd
                        http://cxf.apache.org/core
http://cxf.apache.org/schemas/core.xsd">
    <!-- <jaxws:server发布服务-->
    <jaxws:server address="/mobile"
serviceClass="cn.itcast.mobile.server.MobileInterface">
        <jaxws:serviceBean>
            <ref bean="mobileServer"/>
        </jaxws:serviceBean>
    </jaxws:server>
    <!-- 配置服务实现类 -->
    <bean name="mobileServer"
class="cn.itcast.mobile.server.MobileInterfaceImpl">
        <property name="mobileClient" ref="mobileClient"/>
    </bean>

    <!-- 配置公网客户端 -->
    <jaxws:client id="mobileClient"
address="http://webservice.webxml.com.cn/WebServices/MobileCo
deWS.asmx"
        serviceClass="cn.itcast.mobile.MobileCodeWSSoap"/>

</beans>
```

第八步：配置 web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>ws_2_cxf_spring_server</display-name>

  <!-- 设置spring的环境 -->
  <context-param>
    <!--contextConfigLocation是不能修改的  -->
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-
```

```
value>
  </context-param>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</
listener-class>
  </listener>

  <!-- 配置CXF的Servlet -->
  <servlet>
    <servlet-name>CXF</servlet-name>
    <servlet-
class>org.apache.cxf.transport.servlet.CXFServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CXF</servlet-name>
    <url-pattern>/ws/*</url-pattern>
  </servlet-mapping>
  <!-- 配置mobileServlet -->
  <servlet>
    <servlet-name>mobileServlet</servlet-name>
    <servlet-
class>cn.itcast.mobile.server.servlet.MobileServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
    <servlet-name>mobileServlet</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

第九步：部署到 tomcat 下，启动 tomcat

第十步：测试

测试服务是否发布成功

测试查询界面