WebService

回顾：

1. WebService的基本概念
2. WebService基础应用
3. WebService的基本要素
4. WebService的客户端调用方式
5. 如何使用注解修改WSDL内容

今天任务

1. CXF的基本介绍及安装、配置
2. 使用CXF发布SOAP协议的服务
3. Spring中整合CXF发布服务
4. 使用Spring整合CXF发布REST服务
5. 综合案例介绍

教学目标

1．掌握CXF的基本介绍及安装、配置
2．掌握使用CXF发布SOAP协议的服务
3．掌握Spring中整合CXF发布服务
4．掌握使用Spring整合CXF发布REST服务

第一章 CXF的介绍、安装和配置

1.1 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）

1.2 安装和配置

1.2.1 下载地址 http://cxf.apache.org/download.html

1.2.2 安装和配置

注意： 安装配置之前需要先安装jdk。

1.将下载的文件进行解压,配置环境变量，新建CXF_HOME,输入下载文件库的路径，
示例：C:\software\apache-cxf-3.2.1\apache-cxf-3.2.1
2.在系统环境变量的Path追加：第三步C:\software\apache-cxf-3.2.1\apache-cxf-3.2.1\bin目录
3.测试，在cmd下加入wsdl2java -h

第二章 CXF发布SOAP协议的服务

2.1 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

2.2 实现

2.2.1 服务端

1.导入jar包
//只需要引入lib下的cxf-manifest.jar即可 ：
//具体步骤如下：
// 右键项目-properties-Java Build Path--Add External JARs即可
2.创建SEI接口,要加入@WebService接口
备注：@BindingType(SOAPBinding.SOAP12HTTP_BINDING)表示发布SOAP1.2的服务端

```
@WebService
```

```
    @BindingType(SOAPBinding.SOAP12HTTP_BINDING)
    public interface WeatherInterface {
        public String queryWeather(String cityName);
    }
3.创建SEI接口实现类。
public class WeatherInterfaceImpl implements WeatherInterface {

    @Override
    public String queryWeather(String cityName) {
        // TODO Auto-generated method stub
        if ("北京".equals(cityName)) {
            return "冷且霾";
        } else {
            return "暖且晴";
        }
    }
}
4.发布服务(使用JaxWsServerFactoryBean发布)
   备注：endpoint仅支持发布实现类，JaxWsServerFactoryBean支持发布接口。
   public class WeatherServer {
    //用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；
    public static void main(String[] args) {
        JaxWsServerFactoryBean factoryBean =new JaxWsServerFactoryBean();
        //设置服务接口
        factoryBean.setServiceClass(WeatherInterface.class);
        //设置服务实现类
        factoryBean.setServiceBean(new WeatherInterfaceImpl());
        //设置服务地址
        factoryBean.setAddress("http://127.0.0.1:8888/weather");
        //发布
        factoryBean.create();
    }
}
5.测试服务是否发布成功，阅读使用说明书.直接访问即可。
```

2.2.2 客户端

```
1.生成客户端代码
      1.1 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
      1.2 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
      1.3 wsdl2java支持SOAP1.1和SOAP1.2



  完整示例：wsdl2java -p com.sky.cxf.weather -d . http://127.0.0.1:8888/weather?
```

```
wsdl
          //根据指定说明书地址生成客户端代码 －p指定生成的包名  －d .表示输出到当前目录
2.使用说明书，使用生成代码调用服务端
         //使用JaxWsProxyFactoryBean调用服务端，设置2个参数，1.设置服务接口；
    public class WeatherClient {
    public static void main(String[] args) {
        JaxWsProxyFactoryBean factoryBean =new JaxWsProxyFactoryBean();
        //设置服务接口
        factoryBean.setServiceClass(WeatherInterface.class);
        //设置服务地址
        factoryBean.setAddress("http://127.0.0.1:8888/weather");
        //获取服务接口实例
        WeatherInterface weatherInterface
=factoryBean.create(WeatherInterface.class);
        //调用查询方法
        String string = weatherInterface.queryWeather("宝鸡");
        System.out.println(string);
    }
}
```

2.2.3 拦截器

拦截器原理：

1.拦截器可以拦截请求和响应
2.拦截器可以有多个
3.拦截器可以根据需要自定义

拦截器使用

拦截器可以加到服务端也可加到客户端(加入到服务器需要重新发布)
获取拦截器列表，将自己的拦截器加入列表中

```
factoryBean.getInInterceptors().add(new LoggingInInterceptor());
factoryBean.getOutInterceptors().add(new LoggingOutInterceptor());
```

第三章 CXF+Spring整合发布SOAP的服务

3.1 服务端

3.1.1 第一步：创建web工程，引入cxf的jar包

3.1.2 第二步：创建SEI接口

3.1.3 第三步：创建SEI接口实现类

3.1.4 第四步：配置Spring配置文件 applicationContext.xml

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

```
xmlns:cxf="http://cxf.apache.org/cxf"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
xmlns:jaxws="http://cxf.apache.org/jaxws"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
  http://cxf.apache.org/cxf
                        http://cxf.apache.org/schemas/core.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 ">
    <!--jaxws:endpoint发布soap协议的服务，对EndPoint类封装，该类不需要实现任何接口-->
    <jaxws:endpoint address="/hello" implementor="com.ws.cxf.server1.Hello">
</jaxws:endpoint>
    <!--jaxws:server发布SOAP协议的服务 ， 对JaxWsServerFactoryBean类封装 -->
    <jaxws:server serviceClass="com.ws.cxf.server.WeatherInterface"
        address="/weather">
        <jaxws:serviceBean>
            <ref bean="weatherInterface" />
        </jaxws:serviceBean>
         <!--配置拦截器-->
        <jaxws:inInterceptors>
            <ref bean="inInterceptor" />
        </jaxws:inInterceptors>
        <jaxws:outInterceptors>
            <ref bean="outInterceptor" />
        </jaxws:outInterceptors>
    </jaxws:server>
    <!-- 配置服务实现类 -->
    <bean name="weatherInterface" class="com.ws.cxf.server.WeatherInterfaceImpl">
</bean>
    <!--配置拦截器bean -->
    <bean name="inInterceptor"
class="org.apache.cxf.interceptor.LoggingInInterceptor"></bean>
    <bean name="outInterceptor"
class="org.apache.cxf.interceptor.LoggingOutInterceptor"></bean>
</beans>
```

3.1.5 第五步：配置web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">
  <display-name>CXFSpring</display-name>
  <!-- 配置Spring环境 -->

    <context-param>
```

```
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>
    <!-- Bootstraps the root web application context before servlet initialization
-->
    <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>
</web-app>
```

3.1.6 第六步：启动tomcat,部署web工程到tomcat

3.1.7 测试服务是否发布成功

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

  示例：http://localhost/CXFSpring/ws/weather?wsdl

  http://localhost/CXFSpring/ws/hello?wsdl

3.2 客户端

3.2.1 第一步：创建web工程作为客户端，引入cxf的jar包

3.2.2 第二步：生成客户端代码

```
执行命令：wsdl2java -p 包名 －d . 服务说明书地址
示例：
    wsdl2java -p com.ws.cxf.client  －d . http://localhost/CXFSpring/ws/weather?
wsdl
```

3.2.3 第三步：配置spring的配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.springframework.org/schema/beans"
xmlns:cxf="http://cxf.apache.org/cxf"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
xmlns:jaxws="http://cxf.apache.org/jaxws"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
  http://cxf.apache.org/cxf
                        http://cxf.apache.org/schemas/core.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 ">
    <!--jaxws:client实现客户端配置,对JaxWsProxyFactoryBean类封装-->
    <jaxws:client id="weatherClient"
address="http://localhost/CXFSpring/ws/weather"
        serviceClass="com.ws.cxf.client.WeatherInterface"></jaxws:client>
</beans>
```

3.2.4 第四步：初始化spring上下文，获取接口实现类，调用查询方法

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

import com.ws.cxf.client.WeatherInterface;

public class WeatherClient {
    public static void main(String[] args) {
        ApplicationContext context = new
ClassPathXmlApplicationContext("classpath:applicationContext.xml");
        WeatherInterface weatherInterface = (WeatherInterface)
 context.getBean("weatherClient");
        System.out.println(weatherInterface.queryWeather("北京"));
    }
}
```

第四章 CXF发布REST服务

4.1 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

4.2 需求

- 第一个：查询单个学生

- 第二个：查询多个学生

4.3 实现

4.3.1 服务端

开发步骤：

1.导入jar

2.创建学生类，需要加入@ XmlRootElement

```
@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;
    .......
    set/get.....
    }
```

3.创建SEI接口

```
package com.sky.ws.rest.server;
import java.util.List;
import javax.jws.WebService;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import com.sky.pojo.Student;
@WebService
@Path("/student")//@Path("/student")就是将请求路径中的“/student”映射到接口上
public interface StudentInterface {
   /**
    * 根据id查询单个学生
    * @param id
    * @return
    */
    @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 queryStuById(@PathParam("id")long id);
    /**
     * 根据查询多个学生

     * @param name
```

```
     * @return
     */
    @GET
    @Produces({MediaType.APPLICATION_XML,"application/json;charset=utf-8"})
    @Path("/querylist/{name}")
    public List<Student> queryStudentList(@PathParam("name") String name);
}
```

4.创建SEI接口实现类

```
package com.sky.ws.rest.server;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.jws.WebService;
import javax.ws.rs.Path;
import com.sky.pojo.Student;
public class StudentInterfaceImpl implements StudentInterface {
    @Override
    public Student queryStuById(long id) {
        Student st = new Student();
        st.setId(id);
        st.setName("张三");
        st.setBirthday(new Date());
        return st;
    }
    @Override
    public List<Student> queryStudentList(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;
    }
}
```

5.发布服务

```
package com.sky.ws.rest.server;
```

```
import org.apache.cxf.jaxrs.JAXRSServerFactoryBean;
public class StudentServer {

    public static void main(String[] args) {
        //JAXRSServerFactoryBean发布REST的服务
        JAXRSServerFactoryBean factoryBean =new JAXRSServerFactoryBean();
        //设置服务实现类
         factoryBean.setServiceBean(new StudentInterfaceImpl());
        //设置资源类,如果有多个资源类，可以以“,”隔开。
         factoryBean.setResourceClasses(StudentInterfaceImpl.class);
        //设置服务地址
        factoryBean.setAddress("http://localhost:8801/user");
        //发布服务
        factoryBean.create();
    }
}
```

6.测试服务

```
6.1 测试查询单个学生：http://localhost:8801/user/student/query/1002
 结果如下：
    <student>
          <birthday>2017-11-19T21:39:18.449+08:00</birthday>
          <id>1002</id>
          <name>张三</name>
    </student>
6.2 测试查询多个学生：http://localhost:8801/user/student/querylist/1002
GET请求默认是xml类型：
结果如下：
    <students>
          <student>
                <birthday>2017-11-19T21:40:16.609+08:00</birthday>
                <id>110</id>
                <name>张三</name>
          </student>
          <student>
                <birthday>2017-11-19T21:40:16.609+08:00</birthday>
                <id>120</id>
                <name>李四</name>
          </student>
    </students>
6.3 多个类型选择：http://localhost:8801/user/student/querylist/1002?_type=json
    结果如下：
        {"student":[


                    {"birthday":"2017-11-
```

```
19T21:41:28.204+08:00","id":110,"name":"张三"},
{"birthday":"2017-11-19T21:41:28.204+08:00","id":120,"name":"李四"}
                    ]
        }
```

6.4 http://localhost:8801/user/student/querylist/1002?_type=xml

备注:

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，否则会报异常

如果在同一方法上同时指定XML和JSON媒体类型，默认返回XML

4.3.2 客户端

```
package com.sky.ws.rest.client;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
public class HttpClient {
    public static void main(String[] args) throws Exception {
        // 第一步：创建服务地址，不是WSDL地址
        URL url = new URL("http://localhost:8801/user/student/querylist/1001?
_type=json");
        // 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");
        // Post 请求不能使用缓存
        connection.setUseCaches(false);
        connection.setInstanceFollowRedirects(true);
        // 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);
        connection.setDoOutput(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();
        }
    }
}
```

第五章 CXF+Spring整合发布REST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POJO类(同上)

第三步：创建SEI接口(同上)

第四步：创建SEI实现类(同上)

第五步：配置Spring配置文件,applicationContext.xml

使用 <jaxrs:server>发布服务：设置1.服务地址；2.服务实现类

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:cxf="http://cxf.apache.org/cxf"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
    xmlns:jaxws="http://cxf.apache.org/jaxws"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
    http://cxf.apache.org/cxf
    http://cxf.apache.org/schemas/core.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 ">
      <!-- 发布REST的服务,对JAXRSServerFactoryBean类封装 -->
    <jaxrs:server address="/user">
        <jaxrs:serviceBeans>
            <ref bean="studentServiceImpl"/>
        </jaxrs:serviceBeans>
    </jaxrs:server>
    <!-- 配置服务类 -->

    <bean name="studentServiceImpl"
```

```
class="com.qf.service.impl.StudentServiceImpl"></bean>
</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://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
    <display-name>CXF_WS_REST_Spring</display-name>
<!-- 配置Spring环境，web服务器一启动就加载Spring的配置文件 applicationContext.xml-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>
    <!-- Bootstraps the root web application context before servlet initialization
-->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <!--配置CXF的Servlet  -->
   <servlet>
      <servlet-name>CXFServlet</servlet-name>
      <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
   </servlet>

   <servlet-mapping>
     <servlet-name>CXFServlet</servlet-name>
     <url-pattern>/ws/*</url-pattern>
   </servlet-mapping>
</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://localhost/CXF_WS_REST_Spring/ws/user?_wadl
```

5.2 客户端

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
```

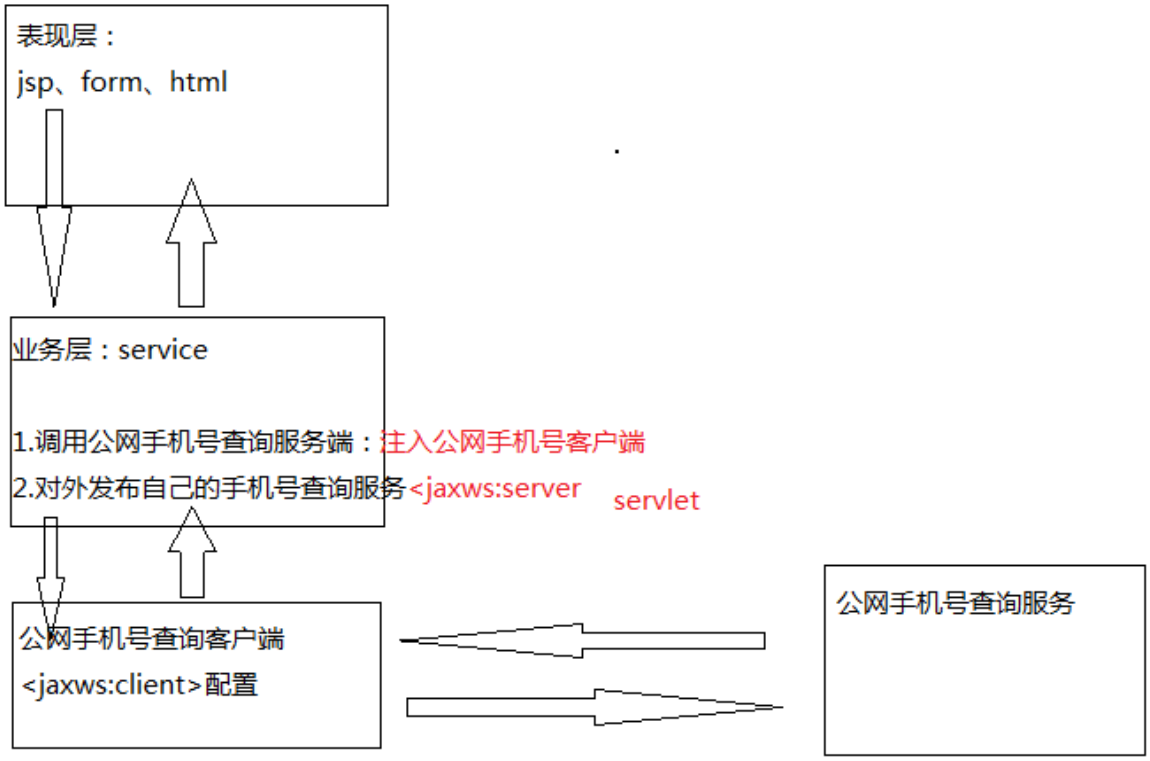
```
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript">
    window.onload = function() {
        document.getElementsByTagName("input")[0].onclick = function() {
            var xhr;
            if (window.XMLHttpRequest) {
                xhr = new XMLHttpRequest();
            } else {
                xhr = new ActiveXObject("Microsoft.XMLHTTP");
            }
    xhr.open("GET", "http://localhost/CXF_WS_REST_Spring/ws/user/stu/query/110?
_type=json");
    xhr.send();
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4) {
            if (xhr.status == 200) {
                var result = xhr.responseText;
                var stu = JSON.parse(result);

alert(stu.student.id+"===="+stu.student.name+"===="+stu.student.birthday);
                    }
                }
            }
        };
    }
</script>
</head>
<body>
    <input type="button" value="点击查询" />
</body>
</html>
```

第六章 综合案例

6.1 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

6.2 分析：



6.3 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

```
wsimport -p com.sky.mobile -s .
http://ws.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl
```

第三步：创建SEI接口

```
package com.sky.mobile;
import javax.jws.WebService;
@WebService
public interface MobileInterface {
    public String queryMobile(String phoneNum);
}
```

第四步：创建SEI实现类

```
package com.sky.mobile;
public class MobileInterfaceImpl implements MobileInterface{
    private  MobileCodeWSSoap codeSoap;
    @Override
```

```
    public String queryMobile(String phoneNum) {
        // TODO Auto-generated method stub
        return codeSoap.getMobileCodeInfo(phoneNum, "");
    }
    public MobileCodeWSSoap getCodeSoap() {
        return codeSoap;
    }
    public void setCodeSoap(MobileCodeWSSoap codeSoap) {
        this.codeSoap = codeSoap;
    }
}
```

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="MobileServlet" 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 com.sky.servlet;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
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.WebApplicationContextUtils;

import com.sky.mobile.MobileInterface;

@WebServlet("/MobileServlet")
```

```
public class MobileServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String string= request.getParameter("phoneNum");
        ApplicationContext ac=WebApplicationContextUtils
                              .getWebApplicationContext(getServletContext());
        MobileInterface m =(MobileInterface) ac.getBean("mobileInterfaceImpl");
        request.setAttribute("result",m.queryMobile(string ));
        request.getRequestDispatcher("index.jsp").forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:cxf="http://cxf.apache.org/cxf"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
    xmlns:jaxws="http://cxf.apache.org/jaxws"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
    http://cxf.apache.org/cxf
    http://cxf.apache.org/schemas/core.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 ">
      <!-- 发布服务 -->
    <jaxws:server address="/mobile">
        <jaxws:serviceBean>
            <ref bean="mobileInterfaceImpl"/>
        </jaxws:serviceBean>
    </jaxws:server>
    <!-- 配置服务类 -->
    <bean name="mobileInterfaceImpl" class="com.sky.mobile.MobileInterfaceImpl">
      <property name="codeSoap" ref="codeSoap"></property>
    </bean>
    <!-- 配置公网客户端-->
    <jaxws:client id="codeSoap"


                address="http://ws.webxml.com.cn/WebServices/MobileCodeWS.asmx"
```

```
serviceClass="com.sky.mobile.MobileCodeWSSoap">
    </jaxws:client>
</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://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">
  <display-name>WS_QueryPhoneNumber</display-name>
<!-- 配置Spring环境，web服务器一启动就加载Spring的配置文件 applicationContext.xml-->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>
    <!-- Bootstraps the root web application context before servlet initialization
-->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

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

</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

课前默写

1. WebService的基本概念
2. WebService基础应用
3. WebService的基本要素
4. WebService的客户端调用方式
5. 如何使用注解修改WSDL内容

作业

1. 使用Spring整合CXF开发天气预报服务
2. 使用Spring整合CXF开发一个REST的学生信息查询服务

面试题

1. 常见的WebService框架比较
2. Spring如何整合CXF框架
3. 如何理解RESTful
4. 如何在项目中使用RESTful