WebService

# 1. 复习准备

## 1.1. Schema约束

几个重要知识：

|  |
| --- |
| 1. **namespace**   **相当于schema文件的id**   1. **targetNamespace属性**   **用来指定schema文件的namespace的值**   1. **xmlns属性**   **引入一个约束, 它的值是一个schema文件的namespace值**   1. **schemaLocation属性**   **用来指定引入的schema文件的位置**  schema规范中:  1. 所有标签和属性都需要有schema文件来定义  2. 所有的schema文件都需要有一个id, 但在这里它叫namespace,  3. namespace的值由什么来指定?  由targetNamespace属性来指定, 它的值是一个url(很有可能不存在)  4. 如何引入一个Schema约束?  属性? : 用xmlns属性  属性值: 对应的schema文件的id(namespace值)  5. 如果引入的schema不是w3c组织定义, 必须指定schema文件的位置  6. schema文件的位置由什么属性来指定?  属性? : schemaLocation  属性值: namespace path  7. 如果引入了N个约束, 需要给n-1个取别名 |

## 1.2. HTTP协议

几个重要知识：

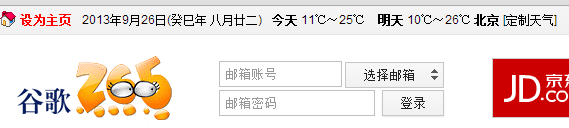
|  |
| --- |
| **1. 请求的组成：  请求行(请求方式 path http1.1)**  **请求头**  **请求体: 只有post请求有**  **2. 响应的组成  响应状态行:**  **响应头**  **响应体 : 浏览器解析显示的数据**  **3. 请求的过程** |
|  |

# 提出2个问题

## 问题一:







1. 它们公司服务器的数据库中都保存了天气预报数据吗?
2. 如果没有, 那数据都存在哪了呢?
3. 这些网站是如何得到这些数据的呢?

## 问题二:

**各个门户网站显示的股票行情信息数据又是怎么来的呢?**

**面试问题:**

1. **webservice相当于HTTP+?+?**

**? : xml**

**? : schema**

1. **wsdl是什么?**

**webservice定义语言, 对应.wsdl文档, 一个webservice会对应一个唯一的wsdl文档, 定义了客户端与服务端发送请求和响应的数据格式和过程**

1. **如何发布一个webservice?**

**定义SEI @webservice @webMethod**

**定义SEI的实现**

**发布: Endpoint.publish(url, SEIImplObject)**

1. **如何请求一个webservice?**
2. **根据wsdl文档生成客户端代码 jdk/cxf**
3. **根据生成的代码调用webService**

# 3. 关于Web Service的几个问题

## 3.1. Web service是什么？

|  |
| --- |
| 1. 基于Web的服务：服务器端整出一些资源让客户端应用访问（获取数据）  2. 一个跨语言、跨平台的规范（抽象）  3. 多个跨平台、跨语言的应用间通信整合的方案（实际）  以各个网站显示天气预报功能为例:  气象中心的管理系统将收集的天气信息并将数据暴露出来(通过WebService Server), 而各大站点的应用就去调用它们得到天气信息并以不同的样式去展示(WebService Client).  网站提供了天气预报的服务，但其实它们什么也没有做，**只是简单了调用了一下气象中心服务器上的一段代码而已**。 |

浏览器

Java

JSP

http协议

webService

Client

webService

Server

Servlet

.net应用

Service

php应用

webService

Client

移动客户端应用

Dao

webService

Client

## 为什么要用Web service？

web service能解决：

跨平台调用

跨语言调用

远程调用

## 3.3. 什么时候使用web Service?

|  |
| --- |
| 1. 同一家公司的新旧应用之间  2. 不同公司的应用之间  分析业务需求：天猫网与中通物流系统如何交互？  3. 一些提供数据的内容聚合应用：天气预报、股票行情 |



中通物流管理应用(.net)

天猫网应用(Java)

访问物流数据表数据的程序

显示物流信息的程序

HTTP

写一段代码调用WebService Server暴露的接口, 得到物流数据

WebService Client

写一段代码调用上面的程序返回物流数据

WebService Server

HTTP + XML

SOAP

# 4. Web Service中的几个重要术语

## 4.1. WSDL：web service definition language

|  |
| --- |
| 直译 : WebService定义语言  1. 对应一种类型的文件**.wsdl**  2. **定义了web service的服务器端与客户端应用交互传递请求和响应数据的格式和方式**  3. 一个web service对应一个**唯一的wsdl文档** |

## SOAP：simple object access protocal

|  |
| --- |
| 直译: 简单对象访问协议   1. 是一种简单的、基于**HTTP和**[**XML**](http://baike.baidu.com/view/63.htm)**的协议**, 用于在WEB上交换**结构化的数据** 2. soap消息：**请求消息**和**响应消息** 3. **http+xml片断** |

## SEI：WebService EndPoint Interface(终端)

|  |
| --- |
| 直译: **web service的终端接口，**  1. 就是WebService服务器端用来处理请求的接口 |

## CXF：Celtix + XFire

|  |
| --- |
| 一个apache的用于开发webservice服务器端和客户端的框架 |

# 开发webservice

## 5.1. 概述

* **开发手段：**
  + 使用JDK开发(1.6及以上版本)
  + 使用CXF框架开发(工作中)
* **组成：**
  + 服务器端
  + 客户端

## 使用JDK开发WebService

### 1).开发服务器端

* **Web Service编码：**
  + @WebService( SEI和SEI的实现类)
  + @WebMethod(SEI中的所有方法)
* **发布Web Service：**
  + Endpoint(终端, 发布webservice)

### 2). 开发客户端

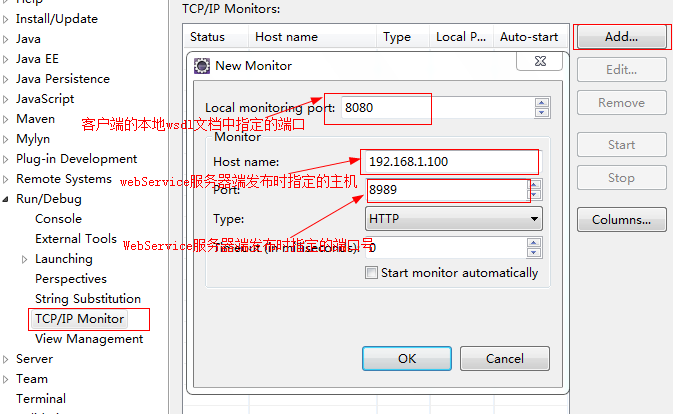
* **使用eclipse提供的web service浏览器访问**
  + 查看对应的wsdl文档：…..?wsdl (一般浏览器)
  + 请求webService并查看请求和响应消息(**webservice浏览器**)
* **创建客户端应用编码方式访问**
  + 借助jdk的wsimort.exe工具生成客户端代码：

**wsimport -keep url**  //url为wsdl文件的路径

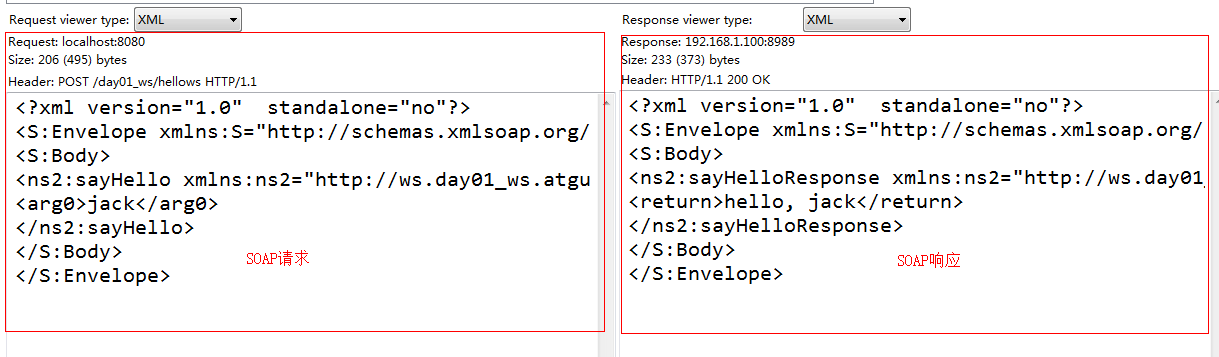
* + 借助生成的代码编写请求代码

## 监听请求: 使用Eclipse的TCP/IP工具(端口转发)

1. 将服务器端的WSDL文档保存到客户端本地
2. 修改文档: 将端口号从8989改为8080
3. 根据本地的wsdl文档生成客户端代码, 并编写客户端的调用代码
4. 配置eclipse的TCP/IP,启动监听



1. 执行客户端代码发送WebService请求



## 调用免费的web service(天气预报)

1. **Google”免费WebService”, 找到提供天气预报Webservice的网络地址**
   * <http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx>
2. **使用eclipse的web service浏览器访问**
3. **客户端编码方式访问**
   * 借助命令工具自动生成客户端代码
   * 借助生成的代码编写请求代码

|  |
| --- |
| 说明: 直接生成客户端代码会抛异常, 无法生成客户端代码, 解决办法:   1. 将对应的wsdl文档保存到本地 2. 修改wsdl文档的部分内容:   **将** <s:element ref="s:schema" /><s:any /> 替换成 <s:any minOccurs="2" maxOccurs="2"/>  备注: 这个是Java调用net的webservice都有的问题 |

## 使用CXF开发web service

加入cxf的jar包即可，其它不需要改动

## WebService请求深入分析

### 1). 分析WebService的WSDL文档结构

#### 1.1). 实例截图



<definitions>

<type>

<message>

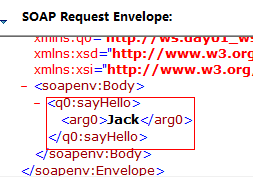
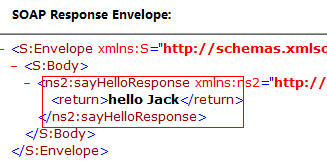
<portType>

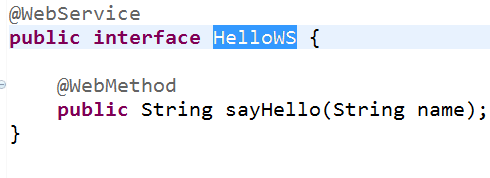
<binding>

<service>

#### 1.2). 文档结构

|  |
| --- |
| **<definitions>**  **<types>**  **<schema>**  **<element>**  **</types>**  **<message>**  **<part>**  **</message>**  **<portType>**  **<operation>**  **<input>**  **<output>**  **</portType>**  **<binding>**  **<operation>**  **<input>**  **<output>**  **</binding>**  **<service>**  **<port>**  **<address>**  **</service>**  **</definitions>** |



#### 1.3). 文档结构图

element

schema

types

message

part

SEI

input/output

operation

portType

SEI实现类

input/output

operation

binding

service

address

port

客户端生成的Service的实现类(工厂)

#### 1.4). 重要标签的说明

* **types** - 数据类型(标签)定义的容器，里面使用schema定义了一些标签结构供message引用
* **message** - 通信消息的数据结构的抽象类型化定义。引用types中定义的标签
* **operation** - 对服务中所支持的操作的抽象描述，一个operation描述了一个访问入口的请求消息与响应消息对。
* **portType** - 对于某个访问入口点类型所支持的操作的抽象集合，这些操作可以由一个或多个服务访问点来支持。
* **binding** - 特定端口类型的具体协议和数据格式规范的绑定。
* **service**- 相关服务访问点的集合
* **port** - 定义为协议/数据格式绑定与具体Web访问地址组合的单个服务访问点。

### 2). 测试CXF支持的数据类型

1. **基本类型**
   * **int,float,boolean等**
2. **引用类型**
   * String
   * 集合：数组，List, Set, Map
   * 自定义类型 Student

### 3). 一次Web service请求的流程

一次web service请求的本质:

1)客户端向服务器端发送了一个soap消息(http请求+xml片断)

2) 服务器端处理完请求后, 向客户端返回一个soap消息

那么它的流程是怎样的呢？

wsdl

wsdl

<q0:sayHello> <arg0>BB</arg0>

</q0:sayHello>

class HelloWsImpl {

String sayHello(String anme) {}

}

String result = hellWS.sayHello("BOB");

http+xml

<ns2:sayHelloResponse">

<return>Hello BB</return>

</ns2:sayHelloResponse>

wsdl

wsdl

## CXF框架的深入使用

### 1).CXF的拦截器

#### 1.1) 理解

* **为什么设计拦截器？**
  1. 为了在webservice请求过程中,能动态操作请求和响应数据, CXF设计了拦截器.
* 拦截器分类：
  1. 按所处的位置分：服务器端拦截器，客户端拦截器
  2. 按消息的方向分：入拦截器，出拦截器
  3. 按定义者分：系统拦截器，自定义拦截器

server

Client

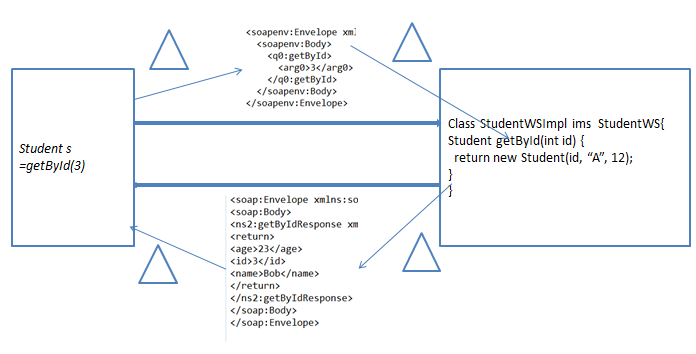
* **拦截器API**

Interceptor(拦截器接口)

AbstractPhaseInterceptor(自定义拦截器从此继承)

LoggingInInterceptor(系统日志入拦截器类)

LoggingOutInterceptor(系统日志出拦截器类)



#### 1.2) 编码实现拦截器

* **使用日志拦截器，实现日志记录**
  + LoggingInInterceptor
  + LoggingOutInterceptor
* **使用自定义拦截器，实现用户名与密码的检验**
  + 服务器端的in拦截器
  + 客户端的out拦截器
  + xfzhang/123456

### 2). 用CXF编写基于spring的web service

#### 2.1). 编码实现

1. **Server端**
   * 创建spring的配置文件beans.xml,在其中配置SEI
   * 在web.xml中，配置上CXF的一些核心组件
2. **Client端**
   * 生成客户端代码
   * 创建客户端的spring配置文件beans-client.xml,并配置
   * 编写测试类请求web service

#### 2.2). 添加自定义拦截器

1. **Server端**
   * 在beans.xml中，在endpoint中配置上入拦截器
2. **Client端**
   * 通过Client对象设置出拦截器

## 其它调用WebService的方式

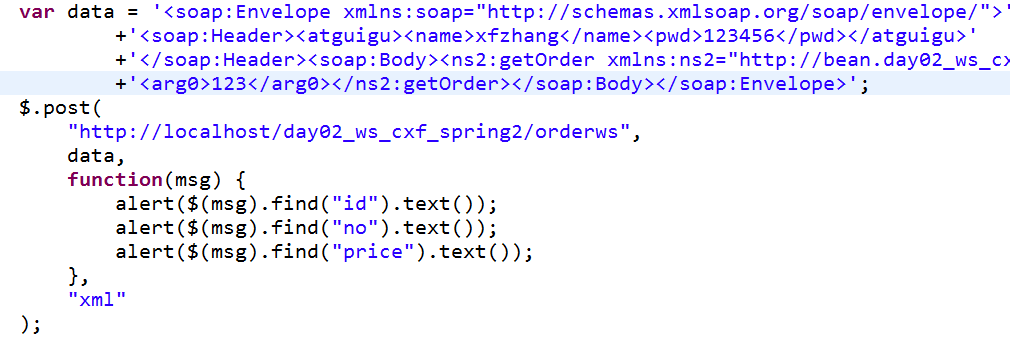
### 1). Ajax调用webService



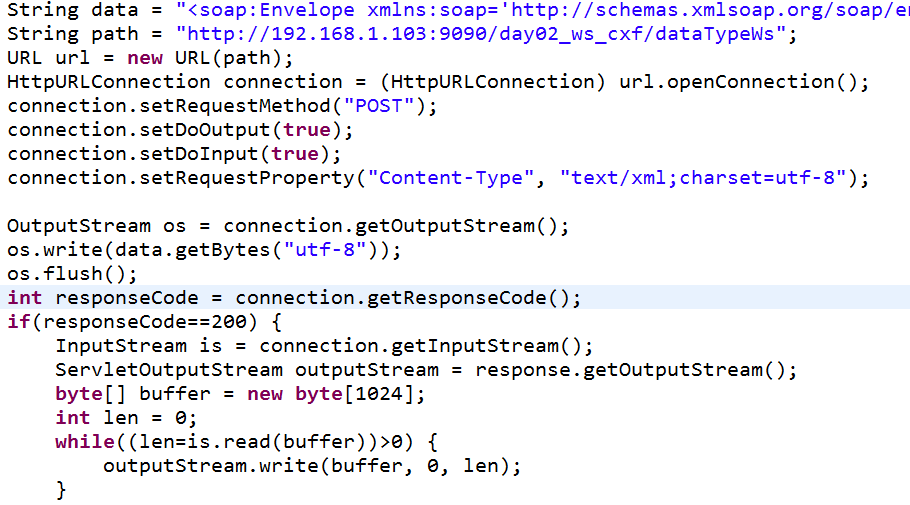
**跨域请求问题:**

|  |
| --- |
| **1. 什么是跨域请求? Host**  1. sina.com--=->baidu.com/xxx.jsp  2. localhost----🡪192.168.42.165  **2. 解决ajax跨域请求webservice的问题?**  在客户端应用中使用java编码去请求webservice, 在页面中去请求自己的后台 |

### 2). Jquery调用WebService



### 3). HttpURLConnection调用WebService



## 通过注解修改wsdl文档

### 1). JDK中的相关注解

#### 1.1). @WebService

|  |
| --- |
| * 作用在具体类上。而不是接口。 * 一个类只有添加了此注解才可以通过Endpoint发布为一个web服务。 * 一个添加了此注解的类，必须要至少包含一个实例方法。静态方法和final方法不能被发布为服务方法。 * WebService注解包含以下参数： |

#### 1.2). @WebMethod

|  |
| --- |
| * 此注解用在方法上，用于修改对外暴露的方法。 |

#### 1.3). @WebResult

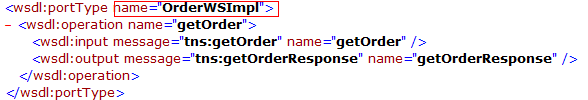
|  |
| --- |
| 用于定制返回值到WSDL的映射 |

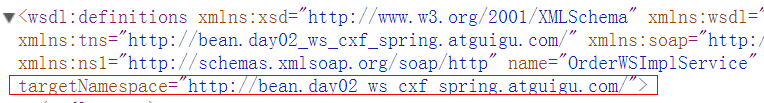
#### 1.4). @WebParam

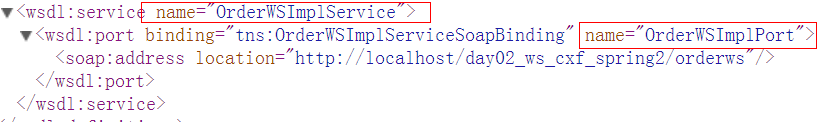
|  |
| --- |
| 用于定义WSDL中的参数映射 |

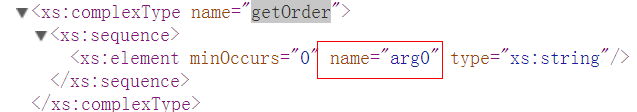
#### 1.5). @XmlElement

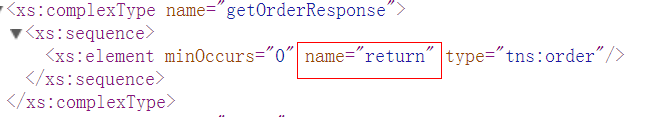
|  |
| --- |
| 用于定义实体类的属性到WSDL中的映射(get/set方法上) |

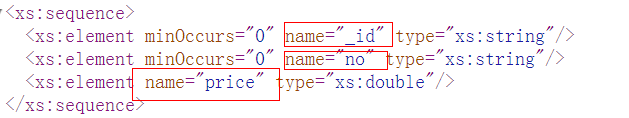












### 2). 说明

|  |
| --- |
| 即使是没有修改源代码，只修改了注解，客户端的代码也必须要重新生成, 否则调用将会失败。 |