Web Service笔记

笔记本： Web Service笔记
创建时间： 2018/12/23 10:50 **更新时间：** 2019/1/12 21:06
作者： 584614151@qq.com
标签： Web Service笔记

★ 什么是Web Service？

不同平台（OS）、不同语言的应用之间远程调用的规范！！！

★ Web Service的应用场景：

▲ 不同公司的业务整合：

不同公司之间进行业务整合时，此时必然带来信息化系统的整合，

此时两个公司之间信息化平台往往存在运行平台不同、开发语言不同的问题，

——这就是Web Service的应用场景。

★ Web Service与SOA：

SOA(Service Orient Architect：面向服务的架构）：新型的软件构建方式；

SOA的理念：整个软件由一个一个的服务组件来完全解耦的方式组合在一起，当某个服务组件不合适（性能低下）

直接替换该服务组件即可，对其他服务组件没有任何影响。

电脑： 主版（通信总线）、内存（临时存储）、CPU（计算）、显卡（显示计算）、硬盘（持久存储）

SOA = ESB(Enterprise Service Bus、企业服务总线） + Web Service（各组件的规范）

SOA = 淘宝（ESB） 天天快递(内存) 顺丰快递（CPU） 邮政快递(显卡)

	程序单元	粒度	开发难度
面向过程	函数	小	大
面向对象	类	↓	↓
面向服务	组件	大	小

★ Web Service的开发：

Web Service是Java EE的技术规范。

JavaEE的技术规范还有很多，比如JPA、Servle、JDBC等等

所以商业的Java EE服务器比如WebLogic、Websphere就内置支持Web Service。

▲ 开源的Web Service框架：

只要开发的系统使用了此框架就可以在tomcat中也能够实现系统整合。

java领域中最流行的一个Web Service实现框架是CXF！！！

▲ CXF的下载和安装

1. 到官网http://cxf.apache.org下载zip压缩包。

2. 解压apache-cxf-3.1.1.zip，把bin目录添加到PATH环境变量。

3. 解压后文件结构：

- bin：保存该框架的一些工具命令。
- docs：API文档。
- lib： 包含CXF的核心JAR包和第三方依赖JAR包。
- samples： 丰富的示例。

CXF2.x提供该CXF框架的总JAR包，从CXF3.x开始，CXF只提供分模块的JAR包，不再提供总的JAR包。

4. 拷贝CXF必须的8个JAR包

asm.jar、cxf.jar（65个）、mina.jar、neethi.jar、stax2-api.jar、woodstox-core-asl.jar、wsdl4j.jar、xmlschema-core.jar

说明：如果在Java SE项目中使用，还需要添加Jetty的jar包和Servlet规范包,使用slf4j的JAR包。

如果在Java Web项目中使用slf4j的JAR包。

Web项目最终需要部署在Web服务器（代替了Jetty），Web服务器一定自带Servlet规范包。

/**

Tomcat与Jetty：开源、免费的Web服务器。功能是完全相似的，关系是竞争， 但是都不属于Java EE服务器。

Jetty可以作为嵌入式服务器,性能较好，Jetty在很多领域比tomcat其实都要好。

Apache组织最有名的产品是Apache。

Tomcat 富二代

Eclipse组件最有名的产品是Eclipse，AspectJ、Jetty。

**/

★ 使用CXF开发Web Service的服务端

（1）定义一个Web Service接口。

并使用@WebService注解修饰。

（2）定义一个Web Service实现类

也使用@WebService注解修饰，并指定serviceName和endpointInterface两个属性。

Web Service组件 = 接口 + 实现类。

（3）调用Endpoint类的publish方法暴露Web Service接口即可。

一旦Web Service暴露成功， 一定会将自身的WSDL文档暴露出来。

其他任何语言，任何平台上的应用都可根据WSDL文档来调用此Web Service组件。

★ 使用CXF开发Web Service的客户端(京东)

关键点：只要能访问远程Web Service的WSDL文档即可——无需知道远程Web Service的开发语言、运行平台。

（1）用wsdl2java（CXF）或 wsimport（JDK）访问远程WSDL文档可以用来生成Java代码。

CXF提供的一个命令: wsdl2java -encoding utf-8 WSDL的url

--- 建议使用此命令

--- 例如: wsdl2java -encoding utf-8 http://192.168.10.222:9898/lcl?wsdl

// javaEE提供的命令: wsimport -encoding utf-8 -keep WSDL的url -keep用于保留Java源代码。

只有当远程Web Service的接口发生改变（增加方法、减少方法、

方法形参、返回值发生改变）

时才需要重新执行这一步。

（2）找到第一步wsdl2java所生成系列类中， Service类的子类。

创建该Service子类的实例。

（3）以Service子类的实例为工厂，调用getXxxPort()方法，即可获取远程Web Service的代理。

通过代理调用远程Web Service的方法即可。

★ Web Service的形参、返回值类型的传输与转换。

A。 Web Service的形参、返回值类型是8个基本类型及其包装类、String等（标量类型）：

CXF完全可以自动处理。

B。 Web Service的形参、返回值类型是Java Bean(User.java,Book.java,Address.java)式的复合类、

或List集合、Set集合、数组。

CXF完全可以自动处理。

C。 Web Service的形参、返回值类型是非Java Bean式的复合类、Map集合等CXF无法自动处理的类型

可能CXF无法自动处理。

/**********

★ Web Service的相关技术基础：

- WSDL (Web Sevice Definition Language)：Web Service定义语言。

Web Service暴露之后，一定会对外暴露WSDL文档。

客户端要调用远程Web Service，也一定要访问到对方的WSDL文档。

- SOAP (Simple Object Access Protocol)：简单对象访问协议。

Web Service的底层数据通信协议。

- Xml:任何语言都支持XML的生成以及解析，Web Service底层数据流通就是以Xml片段进行的；

- 网络通信

/***

Java	XML	
package	targetNamespace：	用于给类、元素指定命名空间。
import	xmlns :	导入指定命名空间下的类、元素。

***/

★ WSDL规范

接口部分

<types.../>元素

该元素的内容就是一份标准的XML Schema。

XML Schema定义2个XML元素（元素能接受哪些子元素、属性等）

2N个<message.../>元素

为N个WS操作分别定义传入、传出消息。

每个消息都是一个与之同名的XML元素。

<portType.../>元素

N个<operation.../>子元素，每个<operation.../>子元素定义一个可供远程调用WS操作。

每个<operation.../>子元素包含一个<input.../>和<output.../>，

其中<input.../>定义了调用该WS操作的传入消息。

其中<output.../>定义了调用该WS操作的传出消息。

实现部分

<binding.../>

包含一个<soap:binding..../>指定了SOAP的传输风格，以及SOAP协议底层依赖网络协议。

再次包含N个<operation.../>元素，详细定义了N个可供远程调用WS操作。

<service.../>

name属性指定了Web Service的服务名。

包含嵌套的<soap:address .../>，该元素的location属性指定了Web Service的服务地址。

▲ WSDL文档定义了Web Service如下3方面：

- WHAT： 该Web Service包含了哪些可供调用的Web Service操作。

- HOW：　调用Web Service操作应该传入怎样的XML元素、服务器会返回怎样的XML元素。
- WHERE：Web Service的服务地址。

▲ 以sayHi操作为例：

传入消息：

```
<getUserInfo>
    <arg0>″李刚"</arg0>
</getUserInfo>
```

响应消息：

```
<getUserInfoResponse>
    <return>字符串</return>
</getUserInfoResponse>
```

▲ 以getPetsByOwner操作为例：

传入消息：

```
<checkUser>
    <arg0>
        <id>整型</id>
        <name>字符串</name>
        <pass>字符串</pass>
    </arg0>
</checkUser>
```

响应消息：

```
<checkUserResponse>
    <return>
        <color>字符串</color>
        <id>整型</id>
        <name>字符串</name>
    </return>
</checkUserResponse>
```

★ 一次Web Service调用的完整过程：

```
getUserInfo();
```

（1）客户端将调用参数封装成XML片段。

```
<getUserInfo>
 <arg0>罗老师</arg0
</getUserInfo>
```

（2）客户端通过网络将XML片段发送给服务端。

（3）服务端接收XML片段。

（4）服务端解析XML片段、提取String类型的数据；

服务端将string类型的数据转换成调用方法所需的参数。

```
getUserInfo("罗老师")
```

（5）执行方法，得到返回值。

（6）服务端将调用返回值封装成XML片段。

```
<getUserInfoResponse>
    <return>
        <arg0>物流信息</arg0
    </return>
</getUserInfoResponse>
```

（7）服务端通过网络将XML片段发送给客户端。

（8）客户端接收XML片段。

（9）客户端解析XML片段、提取String类型的数据；

客户端将string类型的数据转换成返回值。

从上面描述可以看出，一门编程语言要支持Web Service只要满足2个条件：

- 支持XML解析和生成。
- 支持网络通信。

几乎每个语言都满足，所以这就是Web Service能跨平台、跨语言的根本原因。

不是每个系统都会用到，但往往在一些大的公司，通常都会需要使用Web Service。

【备注】：Web Service可实现不同平台、不用语言之间的相互调用。

但Web Service并不是实现该功能的唯一技术。

★ 拦截器

使用CXF之后，整个Web Service交互过程中，网络上交互的XML片段（SOAP包）对开发者是完全透明。

所有SOAP包的封装、解析、提取数据、类型转换——这系列操作都是由CXF来负责完成。
好处：开发简单。
坏处：开发者无法访问、修改SOAP包。
CXF为了让开发者去访问、修改SOAP包，【CXF】添加拦截器机制。
拦截器：负责拦截Web Service交互过程中传输的SOAP包。

- ▲ CXF默认提供了两个拦截器：

  LoggingInInterceptor： 拦截所有传入的SOAP包，仅仅以日志方式记录该SOAP包。
  LoggingOutInterceptor： 拦截所有传出的SOAP包，仅仅以日志方式记录该SOAP包。

- **▲ 服务端添加拦截器**

（1）获取Endpoint.publish方法的返回值，并强转为EndpointImpl。
（2）调用EndpointImpl对象的getInInterceptors()或getOutInterceptors()
这两个方法即可获取In或Out拦截器List、向List集合中加In、Out拦截器即可。

- **▲ 客户端添加拦截器**

（1）以远程Web Service代理为参数，调用ClientProxy的getClient方法获取Client对象。
（2）调用Client对象的getInInterceptors()或getOutInterceptors()
这两个方法即可获取In或Out拦截器List、向List集合中加In、Out拦截器即可。

★ 自定义拦截器

1）需要继承 AbstractPhaseInterceptor并重写handleMessage方法；

2）在服务端的构造器中可以通过 super(Phase.PRE_INVOKE)指定在调用服务端方法之前进行拦截

3)在客户端的构造器中可以通过super(Phase.PREPARE_SEND)发送请求之前进行拦截，对Soap进行修改，在Soap包中加上 一个Header节点

★ CXF整合Spring

- 服务端开发步骤

（1）烤JAR包。
（2）在web.xml文件中配置CXF的核心Servlet：CXFServlet
（3）准备一个配置文件，并且该文件交给Spring管理在Spring配置文件导入jaxws:命令空间。
（4）在Spring文件中import一份CXF为Spring提供的配置文件。
（5）如果整合的是Struts 2这种“拦截全部请求”的MVC框架，
还需要配置MVC框架，让该框架放行Web Service请求。

- 客户端开发步骤

（1）烤JAR包。
（2）在Spring配置文件导入jaxws:命令空间。
（3）在Spring文件中import一份CXF为Spring提供的配置文件。
（4） 使用wsdl2java或wsimport根据远程WSDL文档生成Java代码。