

CXF讲义

讲师：陈伟俊

WebService概述

- **Web Services**是由企业发布的完成其特定商务需求的在线应用服务,其他公司或应用软件能够通过Internet来访问并使用这项在线服务。
- **Web services** 使用 **XML** 来编解码数据, 并使用 **SOAP** 来传输数据。
- 用简单点的话说, 就是系统对外的接口!

CXF概述

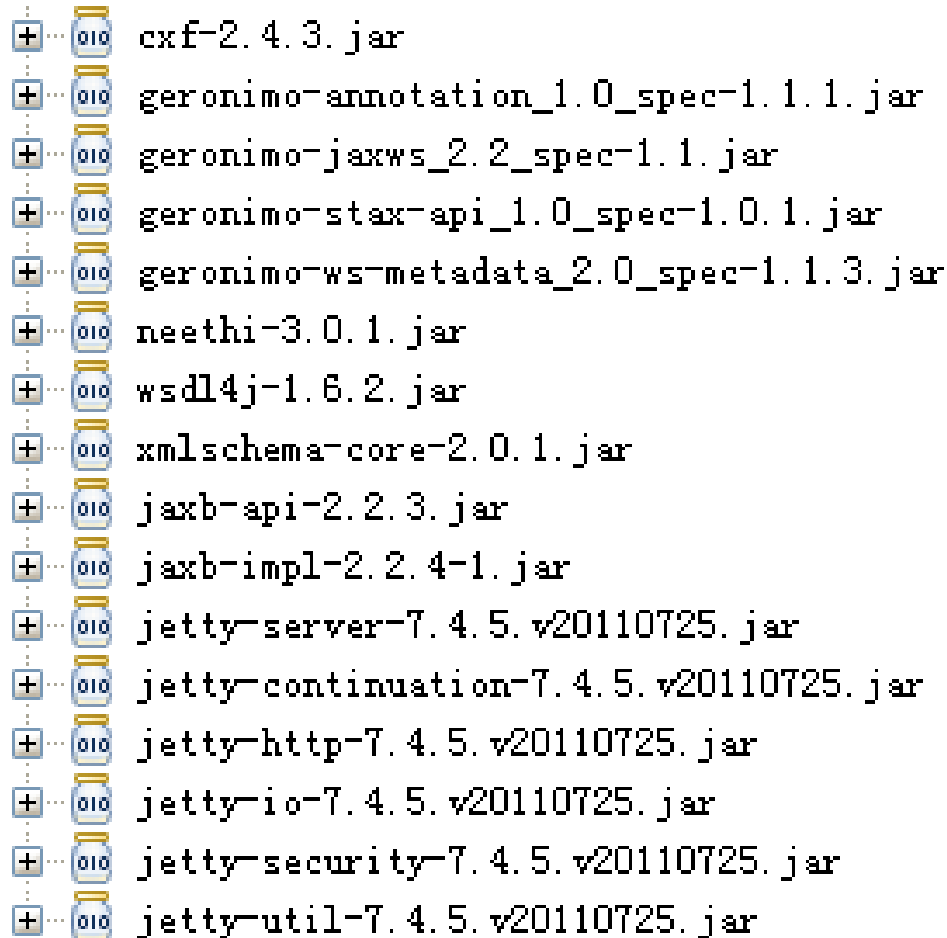
- Apache CXF = Celtix + XFire, Apache CXF 的前身叫 Apache CeltiXfire, 现在已经正式更名为 Apache CXF 了, 以下简称为 CXF。
- CXF 继承了 Celtix 和 XFire 两大开源项目的精华, 提供了对 [JAX-WS](#) 全面的支持, 并且提供了多种 Binding、DataBinding、Transport 以及各种 Format 的支持, 并且可以根据实际项目的需要, 采用代码优先 (Code First) 或者 WSDL 优先 (WSDL First) 来轻松地实现 Web Services 的发布和使用。
- Apache CXF 已经是一个正式的 Apache 顶级项目。

CXF入门例子

- 服务器端
- 客户端
- 如果运行发生如下异常：输入jar包冲突
 - JAXB 2.1 API is being loaded from the bootstrap classloader, but this RI
 - 用打印语句输出如下环境变量的值，找到对应位置拷贝jaxb-api-2.1.jar和jaxb-impl-2.1.12.jar到endorsed目录，没有的话自己创建一个
 - `System.out.println(System.getProperty("java.endorsed.dirs"));`

服务器端必须jar包

- 服务器端必须jar包



- + cxf-2.4.3.jar
- + geronimo-annotation_1.0_spec-1.1.1.jar
- + geronimo-jaxws_2.2_spec-1.1.jar
- + geronimo-stax-api_1.0_spec-1.0.1.jar
- + geronimo-ws-metadata_2.0_spec-1.1.3.jar
- + neethi-3.0.1.jar
- + wsdl4j-1.6.2.jar
- + xmlschema-core-2.0.1.jar
- + jaxb-api-2.2.3.jar
- + jaxb-impl-2.2.4-1.jar
- + jetty-server-7.4.5.v20110725.jar
- + jetty-continuation-7.4.5.v20110725.jar
- + jetty-http-7.4.5.v20110725.jar
- + jetty-io-7.4.5.v20110725.jar
- + jetty-security-7.4.5.v20110725.jar
- + jetty-util-7.4.5.v20110725.jar

服务器端接口

@WebService

public interface HelloWorld {

String sayHello(@WebParam(name="name") String name);

}

服务器端实现类

```
public class HelloWorldImpl implements HelloWorld {
```

```
    @Override
```

```
public String sayHello(String name) {
```

```
    System.out.println("sayHello方法被调用...");
```

```
return "hello," + name;
```

```
}
```

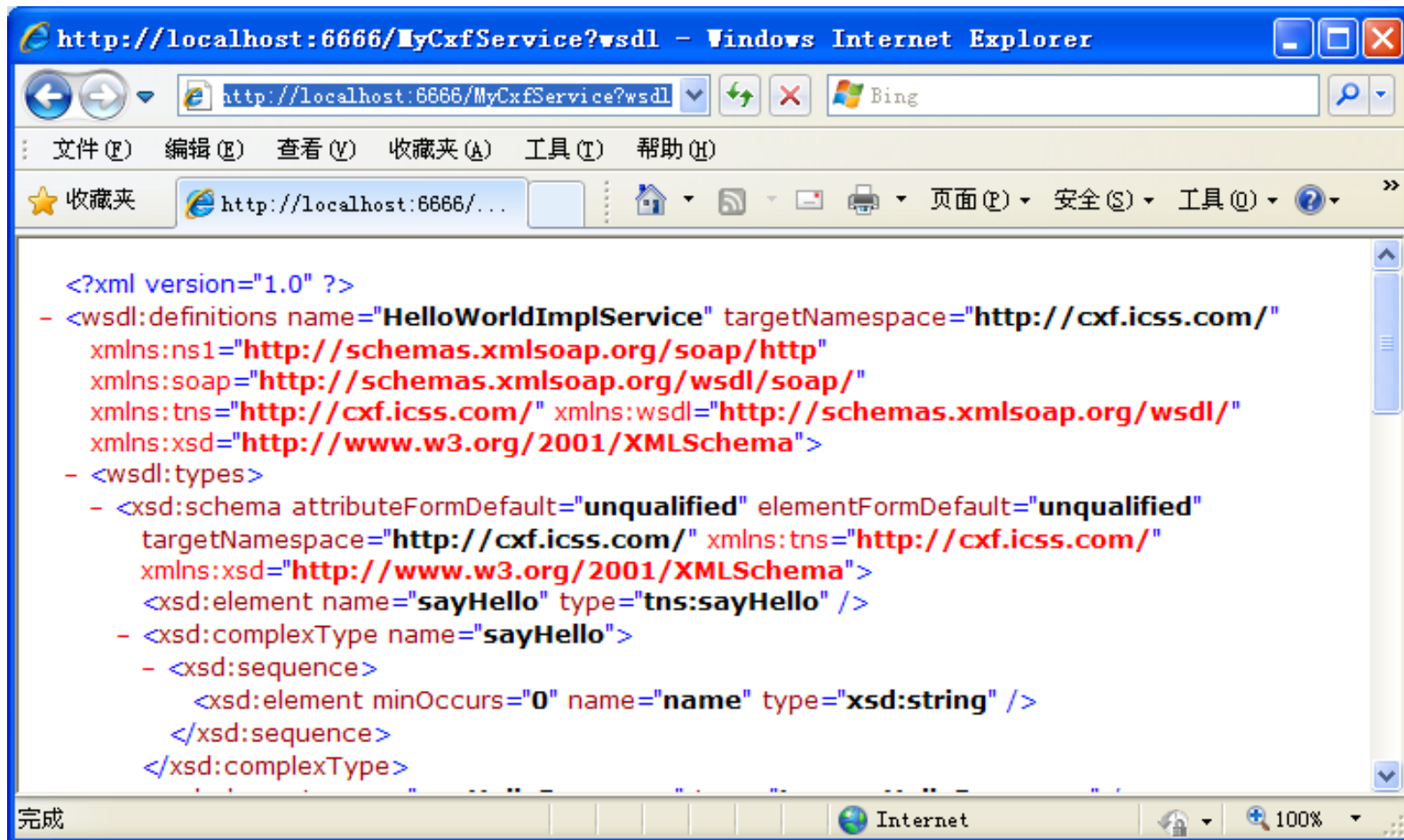
```
}
```

服务器端发布程序：











```
public class MainServer {  
    public static void main(String[] args) {  
        //创建一个JaxWs服务器工厂  
        JaxWsServerFactoryBean factory = new JaxWsServerFactoryBean();  
        //设置发布类  
        factory.setServiceClass>HelloWorldImpl.class);  
        //设置服务发布的地址  
        factory.setAddress("http://localhost:6666/MyCxfService");  
        //根据设置创建一个服务器  
        Server server = factory.create();  
        //启动服务器  
        server.start();  
    }  
}
```


效果截屏

- 输入: `http://localhost:6666/MyCxfService?wsdl`



客户端必须jar包

- +  cxf-2.4.3.jar
- +  geronimo-annotation_1.0_spec-1.1.1.jar
- +  geronimo-jaxws_2.2_spec-1.1.jar
- +  geronimo-stax-api_1.0_spec-1.0.1.jar
- +  geronimo-ws-metadata_2.0_spec-1.1.3.jar
- +  jaxb-api-2.2.3.jar
- +  jaxb-impl-2.2.4-1.jar
- +  neethi-3.0.1.jar
- +  wsdl4j-1.6.2.jar
- +  xmlschema-core-2.0.1.jar

客户端接口

@WebService

public interface HelloWorld {

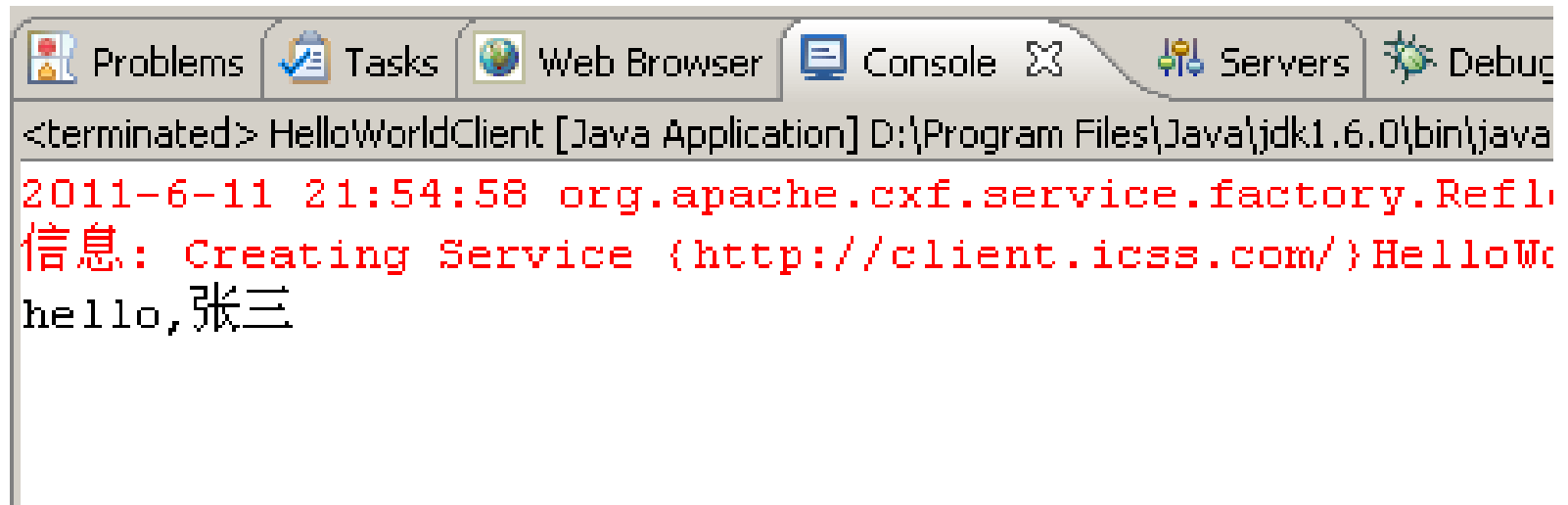
String sayHello(@WebParam(name="name") String name);

}

客户端调用程序

```
public class MainClient {  
    public static void main(String[] args) {  
        //创建一个JaxWs的代理工厂  
        JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();  
        //设置访问地址  
        factory.setAddress("http://localhost:6666/MyCxfService");  
        //设置接口类型  
        factory.setServiceClass>HelloWorld.class);  
        //获得代理类实例  
        HelloWorld helloWorld = (HelloWorld) factory.create();  
        //调用方法  
        String str = helloWorld.sayHello("张三");  
        System.out.println(str);  
    }  
}
```

效果截屏



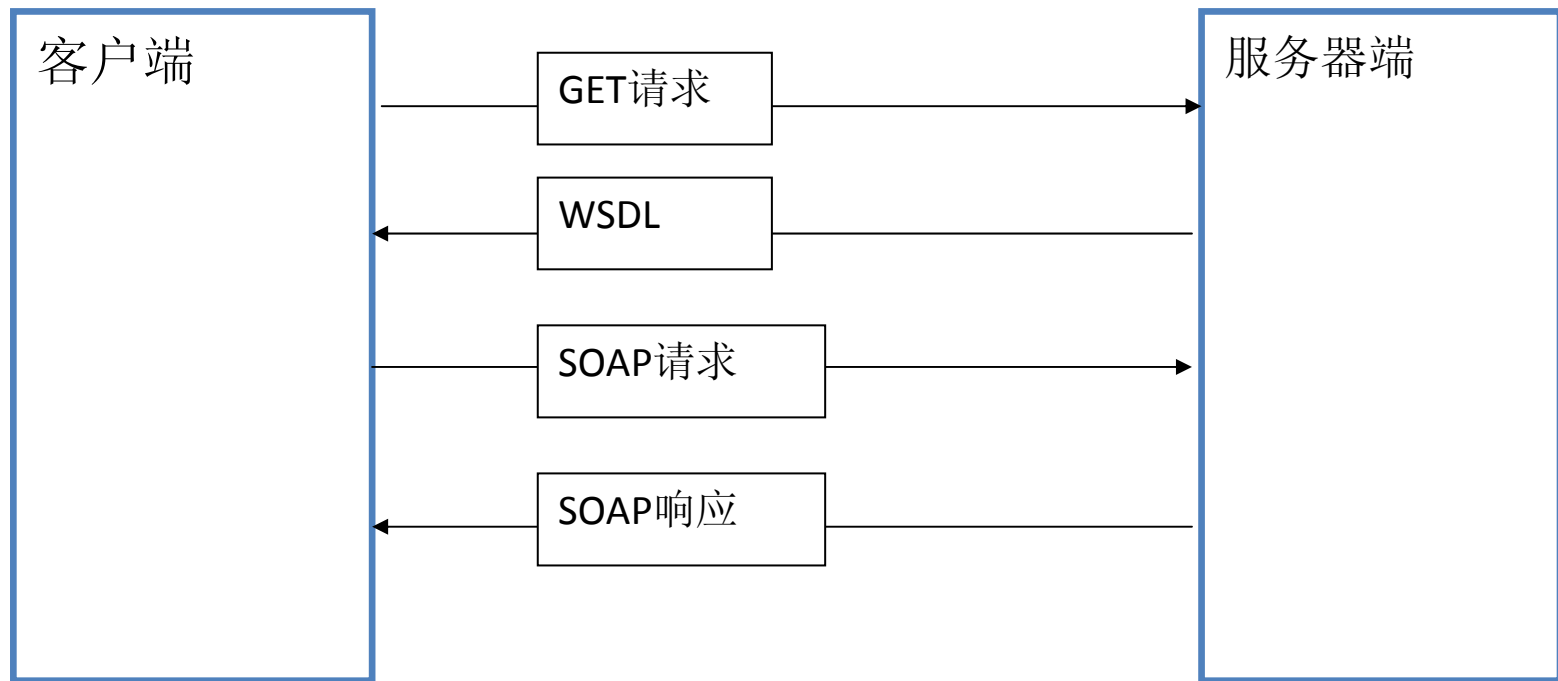
WebService工作原理以及SOAP简介

- **Web service**工作原理大致流程：客户端向服务器端发出一个**GET**请求，然后响应的就是**WSDL**，告诉客户端有哪些暴露的接口功能可以调用。
Web service提供一个工具类，叫做**WSDL2Java**，可以根据**WSDL**生成一系列的代理类。当客户端得到**WSDL**之后，就会向服务器端发出一个**SOAP**请求，这个**SOAP**请求就是一段**XML**，然后服务器端接收到**SOAP**请求，再对这个请求进行解析，然后再把请求的结果返回客户端。

WSDL

- **Web Services Description Language**的缩写，是一个用来描述**Web**服务和说明如何与**Web**服务通信的**XML**语言。为用户提供详细的接口说明书。






















WebService工作原理以及SOAP简介



Spring和CXF整合

- 可用通过Spring来发布CXF的web service，但是这样就需要用到tomcat服务器

服务器端必须jar包

- +  geronimo-stax-api_1.0_spec-1.0.1.jar
- +  geronimo-ws-metadata_2.0_spec-1.1.3.jar
- +  jaxb-api-2.2.3.jar
- +  jaxb-impl-2.2.4-1.jar
- +  neethi-3.0.1.jar
- +  wsdl4j-1.6.2.jar
- +  xmlschema-core-2.0.1.jar
- +  woodstox-core-asl-4.1.1.jar
- +  cxf-2.4.6.jar
- +  geronimo-annotation_1.0_spec-1.1.1.jar
- +  org.springframework.beans-3.0.1.RELEASE-A.jar
- +  org.springframework.context-3.0.1.RELEASE-A.jar
- +  org.springframework.context.support-3.0.1.RELEASE-A.jar
- +  org.springframework.asm-3.0.1.RELEASE-A.jar
- +  org.springframework.expression-3.0.1.RELEASE-A.jar
- +  org.springframework.core-3.0.1.RELEASE-A.jar
- +  org.springframework.web-3.0.1.RELEASE-A.jar
- +  commons-logging-1.0.4.jar
- +  org.springframework.aop-3.0.1.RELEASE-A.jar
- +  org.springframework.aspects-3.0.1.RELEASE-A.jar
- +  stax2-api-3.1.1.jar

服务器端接口

@WebService

public interface IHelloWorld {

```
/**  
 * 此处一定要加@WebParam(name="name"), 否则android客户端传递参数为空  
 */  
String sayHello(@WebParam(name="name") String name);  
  
}
```

服务器端实现类

```
public class HelloWorldImpl implements IHelloWorld {  
  
    @Override  
    public String sayHello(String name) {  
        System.out.println("sayHello方法被调用...name=" + name);  
        return "你好, " + name;  
    }  
  
}
```

服务器端web.xml核心配置

```
<!-- Spring容器初始化-->
<context-param>
    <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>
<!-- 核心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>/service/*</url-pattern>
</servlet-mapping>
```

服务器端applicationContext.xml

<!-- 引入cxf jar包中的配置文件 -->

<import resource="classpath:META-INF/cxf/cxf.xml" />

<import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />

<import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

<!-- 配置一个端点-->

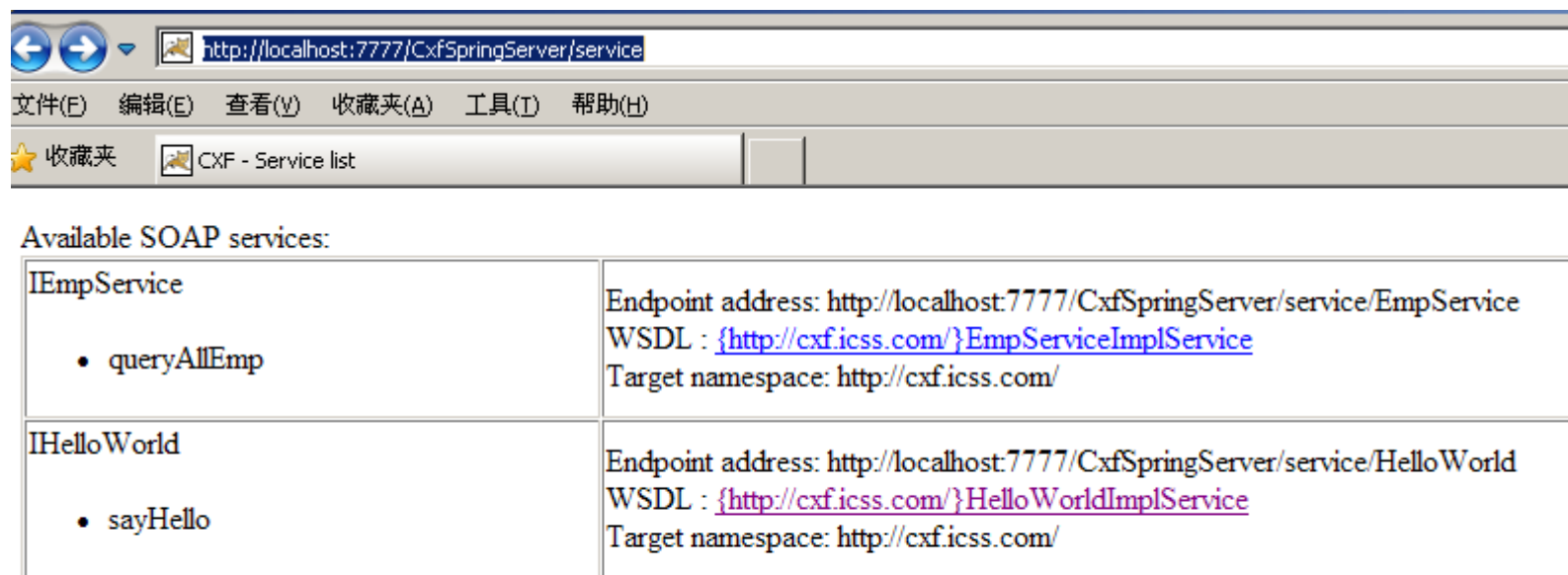
<jaxws:endpoint id="helloWorld" implementor="#helloWorldImpl"
address="/HelloWorld"></jaxws:endpoint>

<!-- 一个普通Bean-->

<bean id="helloWorldImpl" class="com.icss.cxf.HelloWorldImpl"></bean>

访问WSDL

- 访问: <http://localhost:7777/CxfSpringServer/service>
- 出现Service列表画面



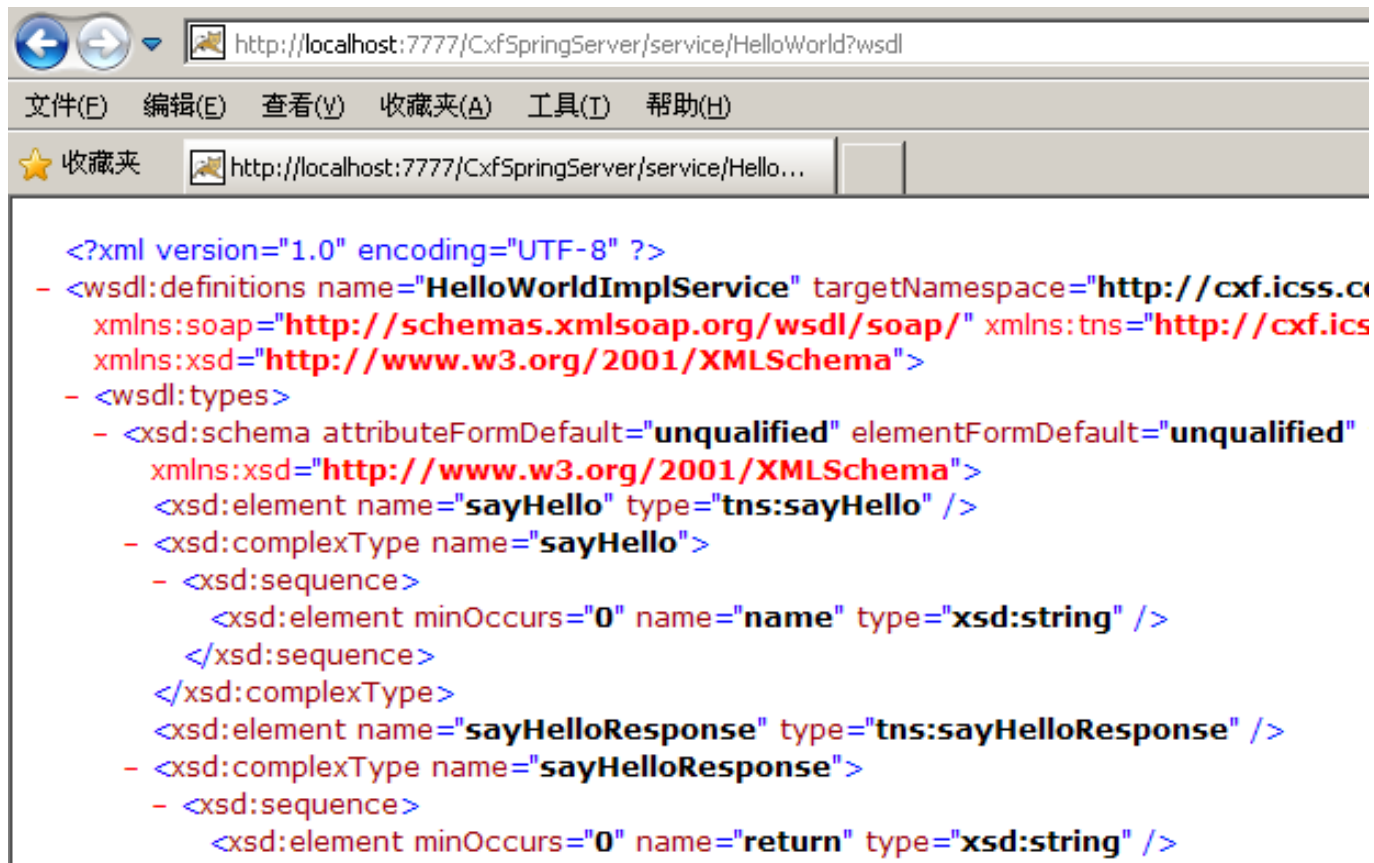
Available SOAP services:

IEmpService <ul style="list-style-type: none">• queryAllEmp	Endpoint address: http://localhost:7777/CxfSpringServer/service/EmpService WSDL : http://cxf.icss.com/HelloWorldImplService Target namespace: http://cxf.icss.com/
IHelloWorld <ul style="list-style-type: none">• sayHello	Endpoint address: http://localhost:7777/CxfSpringServer/service/HelloWorld WSDL : http://cxf.icss.com/HelloWorldImplService Target namespace: http://cxf.icss.com/

Available RESTful services:

访问WSDL






















- 点击某个链接进入



The screenshot shows a web browser window with the address bar containing the URL `http://localhost:7777/CxfSpringServer/service/HelloWorld?wsdl`. The browser's menu bar includes '文件(E)', '编辑(E)', '查看(V)', '收藏夹(A)', '工具(T)', and '帮助(H)'. The address bar also shows a '收藏夹' (Favorites) section with the same URL. The main content area displays the WSDL XML document, which is color-coded for readability. The XML defines a service named 'HelloWorldImplService' with a target namespace of 'http://cxf.icss.c...'. It includes several namespace declarations for 'soap', 'tns', and 'xsd'. The document structure is as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions name="HelloWorldImplService" targetNamespace="http://cxf.icss.c...
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://cxf.ics
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <wsdl:types>
  - <xsd:schema attributeFormDefault="unqualified" elementFormDefault="unqualified"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="sayHello" type="tns:sayHello" />
  - <xsd:complexType name="sayHello">
    - <xsd:sequence>
      <xsd:element minOccurs="0" name="name" type="xsd:string" />
    </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="sayHelloResponse" type="tns:sayHelloResponse" />
  - <xsd:complexType name="sayHelloResponse">
    - <xsd:sequence>
      <xsd:element minOccurs="0" name="return" type="xsd:string" />
```


客户端必须jar包

- +  geronimo-stax-api_1.0_spec-1.0.1.jar
- +  geronimo-ws-metadata_2.0_spec-1.1.3.jar
- +  jaxb-api-2.2.3.jar
- +  jaxb-impl-2.2.4-1.jar
- +  neethi-3.0.1.jar
- +  wsdl4j-1.6.2.jar
- +  xmlschema-core-2.0.1.jar
- +  woodstox-core-asl-4.1.1.jar
- +  cxf-2.4.6.jar
- +  geronimo-annotation_1.0_spec-1.1.1.jar
- +  org.springframework.beans-3.0.1.RELEASE-A.jar
- +  org.springframework.context-3.0.1.RELEASE-A.jar
- +  org.springframework.context.support-3.0.1.RELEASE-A.jar
- +  org.springframework.asm-3.0.1.RELEASE-A.jar
- +  org.springframework.expression-3.0.1.RELEASE-A.jar
- +  org.springframework.core-3.0.1.RELEASE-A.jar
- +  org.springframework.web-3.0.1.RELEASE-A.jar
- +  commons-logging-1.0.4.jar
- +  org.springframework.aop-3.0.1.RELEASE-A.jar
- +  org.springframework.aspects-3.0.1.RELEASE-A.jar
- +  stax2-api-3.1.1.jar

客户端接口

@WebService

public interface IHelloWorld {

String sayHello(@WebParam(name="name") String name);

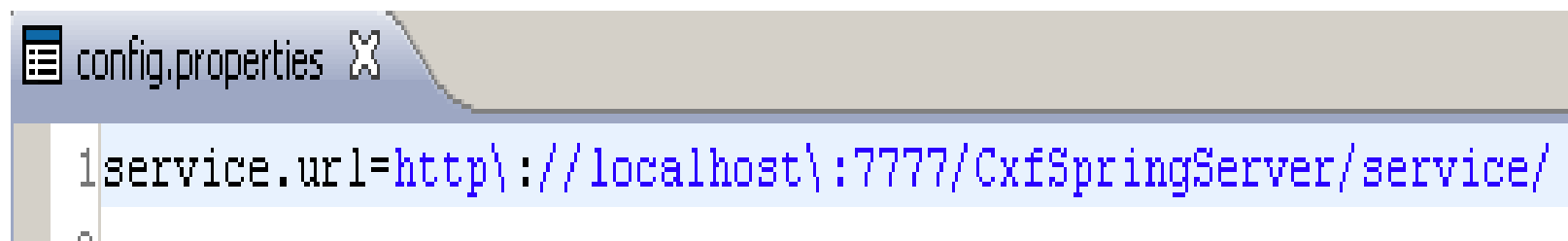
}

客户端Spring配置:applicationContext.xml

```
<!-- 引用外部配置文件config.properties，目的是重用服务器URL -->
<bean id="propertyConfigurer"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
    >
<property name="location" value="config.properties" />
</bean>

<!-- 配置一个Service客户端-->
<jaxws:client id="helloWorldClient"
address="${service.url>HelloWorld"
serviceClass="com.icss.cxf.IHelloWorld" />
```

客户端config.properties配置文件



```
config.properties X
1 service.url=http\://localhost\:7777/CxfSpringServer/service/
```

客户端调用程序

```
public class Client {  
  
    public static void main(String[] args) {  
  
        ApplicationContext context = new  
        ClassPathXmlApplicationContext("applicationContext.xml");  
  
        IHelloWorld helloWorld = (IHelloWorld) context.getBean("helloWorldClient");  
  
        String str = helloWorld.sayHello("tom");  
  
        System.out.println(str);  
    }  
}
```

注意

- 如果使用tomcat发布CXF出现JAR包冲突异常
- 解决办法就是通过执行
- `system.out.println(System.getProperty("java.endorsed.dirs"))`;找到相应位置，将`geronimo-jaxws_2.2_spec-1.1.jar`和`jaxb-api-2.2.3.jar`放到对应目录下即可。没有endorsed目录，自己建一个。例如WEB项目中，打印出来的路径是：`D:\Tomcat 6.0/common/endorsed`，直接把CXF包中的lib目录中的endorsed目录拷贝到`D:\Tomcat 6.0/common`中就好用了，没有common目录自己创建一个