一、**CXF介绍**

        Apache CXF = Celtix + XFire，Apache CXF 的前身叫 Apache CeltiXfire。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支持多种协议，如：SOAP、XML/HTTP、RESTful HTTP 、 CORBA ，并且可以在多种传输协议上运行，比如：HTTP、JMS 、 JBI。CXF可以与[**spring**](http://lib.csdn.net/base/javaee)进行无缝集成。

CXF 框架是一种基于 Servlet 技术的 SOA 应用开发框架，要正常运行基于 CXF 应用框架开发的企业应用，除了 CXF 框架本身之外，还需要 JDK 和 Servlet 容器的支持。

CXF 下载地址：<http://cxf.apache.org/download.html>，选择“File”列中的zip格式下载。解压后可以看到一些文件夹：

└─apache-cxf-2.5.0

    ├─bin           各种工具  
    ├─docs        API文档  
    ├─etc          各种配置文件  
    ├─lib           开发所需jar包  
    ├─licenses  说明  
    ├─modules  jar包

    └─samples  使用例子

**samples文件夹中给提供了在各种环境下使用的例子，个人感觉只需参照样例即可。**

**二、使用CXF+Spring编写并发布WebService**

**首先**，新建一个Web项目，名字叫cxftest，将下载的压缩包中lib文件夹下的jar包全部拷贝到项目的WEB-INF/lib中，在web.xml中配置如下：

**[html]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **<servlet>**
2. **<servlet-name>**CXFServlet**</servlet-name>**
3. **<servlet-class>**org.apache.cxf.transport.servlet.CXFServlet**</servlet-class>**
4. **<load-on-startup>**1**</load-on-startup>**
5. **</servlet>**
6. **<servlet-mapping>**
7. **<servlet-name>**CXFServlet**</servlet-name>**
8. **<url-pattern>**/services/\***</url-pattern>**
9. **</servlet-mapping>**

**然后**，在项目src目录下新建包：com.test，在新建的包中创建TestService.java，代码如下：

**[java]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **package** com.test;
3. **import** javax.jws.WebService;
5. @WebService
6. **public** **interface** TestService {
8. **public** String sayHi(String name);
10. }

**接着**，在创建TestServiceImpl.[**Java**](http://lib.csdn.net/base/javase)，代码如下：

**[java]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **package** com.test;
3. **import** javax.jws.WebService;
5. @WebService(endpointInterface="com.test.TestService")
6. **public** **class** TestServiceImpl **implements** TestService {
8. **public** String sayHi(String name) {
10. **return** name+", welcome to here!";
11. }
12. }

CXF中采用注解的方式声明哪些类作为WebService进行发布，

@WebService：声明webservice接口；

@WebService(endpointInterface="com.test.TestService")：声明这个类是TestService接口的实现类。  
       **其次**，在WEB-INF下新建cxf-servlet.xml文件，代码如下：

**[html]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
3. **<beans** xmlns="http://www.springframework.org/schema/beans"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xmlns:jaxws="http://cxf.apache.org/jaxws"
6. xsi:schemaLocation="
7. http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd"**>**
10. **<import** resource="classpath:META-INF/cxf/cxf.xml" **/>**
11. **<import** resource="classpath:META-INF/cxf/cxf-extension-soap.xml" **/>**
12. **<import** resource="classpath:META-INF/cxf/cxf-servlet.xml" **/>**
14. **<jaxws:server** id="testService" serviceClass="com.test.TestService" address="/testService"**>**
15. **<jaxws:serviceBean>**
16. **<bean** class="com.test.TestServiceImpl" **/>**
17. **</jaxws:serviceBean>**
18. **</jaxws:server>**
19. **</beans>**

id：自定义，serviceClass：接口类的完整包名，address：此接口的访问路径，<jaxws:serviceBean/>标签用于配置接口的实现类

**然后**，在web.xml中配置启动时加载cxf-servlet.xml，并配置Spring的监听器，代码如下：

**[html]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **<context-param>**
2. **<param-name>**contextConfigLocation**</param-name>**
3. **<param-value>**/WEB-INF/cxf-servlet.xml**</param-value>**
4. **</context-param>**
6. **<listener>**
7. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
8. **</listener>**

web.xml文件的完整内容如下：

**[html]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5"
3. xmlns="http://java.sun.com/xml/ns/javaee"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**
8. **<context-param>**
9. **<param-name>**contextConfigLocation**</param-name>**
10. **<param-value>**/WEB-INF/cxf-servlet.xml**</param-value>**
11. **</context-param>**
13. **<listener>**
14. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
15. **</listener>**
17. **<servlet>**
18. **<servlet-name>**CXFServlet**</servlet-name>**
19. **<servlet-class>**org.apache.cxf.transport.servlet.CXFServlet**</servlet-class>**
20. **<load-on-startup>**1**</load-on-startup>**
21. **</servlet>**
22. **<servlet-mapping>**
23. **<servlet-name>**CXFServlet**</servlet-name>**
24. **<url-pattern>**/services/\***</url-pattern>**
25. **</servlet-mapping>**
27. **</web-app>**

**最后**，发布到tomcat中并启动，在浏览器中输入http://localhost:8080/cxftest/services/testService?wsdl，发布成功！

**三、**[**测试**](http://lib.csdn.net/base/softwaretest)**发布的WebService，生成客户端代码调用WebService**

        在浏览器中输入http://localhost:8080/cxftest/services/testService/sayHi?arg0=andy，返回内容正常，调用成功！这里要注意一点，在浏览器中测试接口时，参数名一定要和WSDL文档中的参数名一致，否则会出错。

        生成客户端代码，这里使用CXF自带的wsdl2java.bat工具生成客户端代码：wsdl2java.bat在bin文件夹中，在cmd窗口中进入到wsdl2java.bat所在目录下，

使用如下命令生成客户端代码：wsdl2java -d src -all http://localhost:8080/cxftest/services/testService?wsdl

-d：表示将转化后的代码放到哪里，如果不带此参数，则默认放到wsdl2java.bat所在目录下；

-all：以什么样的方式生成客户端代码，还有其他可选参数，这里不加阐述了，可以直接使用wsdl2java名来查看其他参数。

生成的客户端文件：ObjectFactory.java、package-info.java、SayHi.java、SayHiResponse.java、TestService.java、TestServiceImpl.java、

TestServiceService.java、TestService\_TestServicePort\_Client.java、TestService\_TestServicePort\_Server.java

TestService\_TestServicePort\_Client.java和TestService\_TestServicePort\_Server.java可以不要，前者是调用接口的例子，后者是发布接口的例子。

新建一个InvokeService.java，代码如下：

**[java]** [view plaincopy](http://blog.csdn.net/abc_key/article/details/6986028)

1. **import** com.test.TestService;
2. **import** com.test.TestServiceService;
4. **public** **class** InvokeService {
6. **public** **static** **void** main(String[] args) {
8. TestService s = **new** TestServiceService().getTestServicePort();
9. System.out.println(s.sayHi("andy"));
11. }
12. }

输出值：andy, welcome to here! ，调用成功！