**利用JAX-WS开发Web服务**

本文提供了一个使用[**Java**](http://lib.csdn.net/base/17)如何开发基于SOAP的Web Services，其客户端可以是Perl、Ruby、[**Python**](http://lib.csdn.net/base/11)或Java等。

[**Java SE**](http://lib.csdn.net/base/12) 6封装了JAX-WS(Java API for XML-Web Services)，而JAX-WS同时支持基于SOAP的Web服务和REST风格的Web服务。JAX-WS通常可简写为JWS，当前，JWS的版本为2.x。

基于SOAP的Web服务可用单个Java类的实现，但是最好是用“接口+实现”的方式来实现最佳。

Web服务的接口称为SEI，即Service Endpoint Interface；  
而Web服务的实现称为SIB，即Service Implementation Bean。

SIB可以是一个POJO，也可以是无状态的会话EJB。本文的SIB是普通Java类，通过JDK 6的类库即可实现Web服务的发布。

代码1：服务接口类SEI

**[java]** [view plain](http://blog.csdn.net/chszs/article/details/5050756) [copy](http://blog.csdn.net/chszs/article/details/5050756)

[print?](http://blog.csdn.net/chszs/article/details/5050756)

1. **package** myweb.service;
2. **import** javax.jws.WebService;
3. **import** javax.jws.WebMethod;
4. **import** javax.jws.soap.SOAPBinding;
5. **import** javax.jws.soap.SOAPBinding.Style;
6. @WebService
7. @SOAPBinding(style=Style.RPC)
8. **public** **interface** TimeServer {
9. @WebMethod
10. String getTimeAsString();
11. @WebMethod
12. **long** getTimeAsElapsed();
13. }

代码2：服务实现类SIB

**[java]** [view plain](http://blog.csdn.net/chszs/article/details/5050756) [copy](http://blog.csdn.net/chszs/article/details/5050756)

[print?](http://blog.csdn.net/chszs/article/details/5050756)

1. **package** myweb.service;
2. **import** java.text.DateFormat;
3. **import** java.util.Date;
4. **import** javax.jws.WebService;
5. @WebService(endpointInterface = "myweb.service.TimeServer")
6. **public** **class** TimeServerImpl **implements** TimeServer {
7. /\*\*
8. \* 返回从1970年1月1日0点0时0分起的毫秒数
9. \*/
10. **public** **long** getTimeAsElapsed() {
11. **return** **new** Date().getTime();
12. }
13. /\*\*
14. \* 返回如“2009-12-21”格式的日期
15. \*/
16. **public** String getTimeAsString() {
17. Date date = **new** Date();
18. DateFormat df = DateFormat.getDateInstance();
19. **return** df.format(date);
20. }
21. }

代码3：服务发布类Publisher

**[java]** [view plain](http://blog.csdn.net/chszs/article/details/5050756) [copy](http://blog.csdn.net/chszs/article/details/5050756)

[print?](http://blog.csdn.net/chszs/article/details/5050756)

1. **package** myweb.service;
2. **import** javax.xml.ws.Endpoint;
3. **public** **class** TimeServerPublisher {
4. **public** **static** **void** main(String[] args){
5. // 第一个参数是发布的URL
6. // 第二个参数是SIB实现
7. Endpoint.publish("http://127.0.0.1:10100/myweb", **new** TimeServerImpl());
8. }
9. }

编译以上代码：

**javac myweb/service/\*.java**

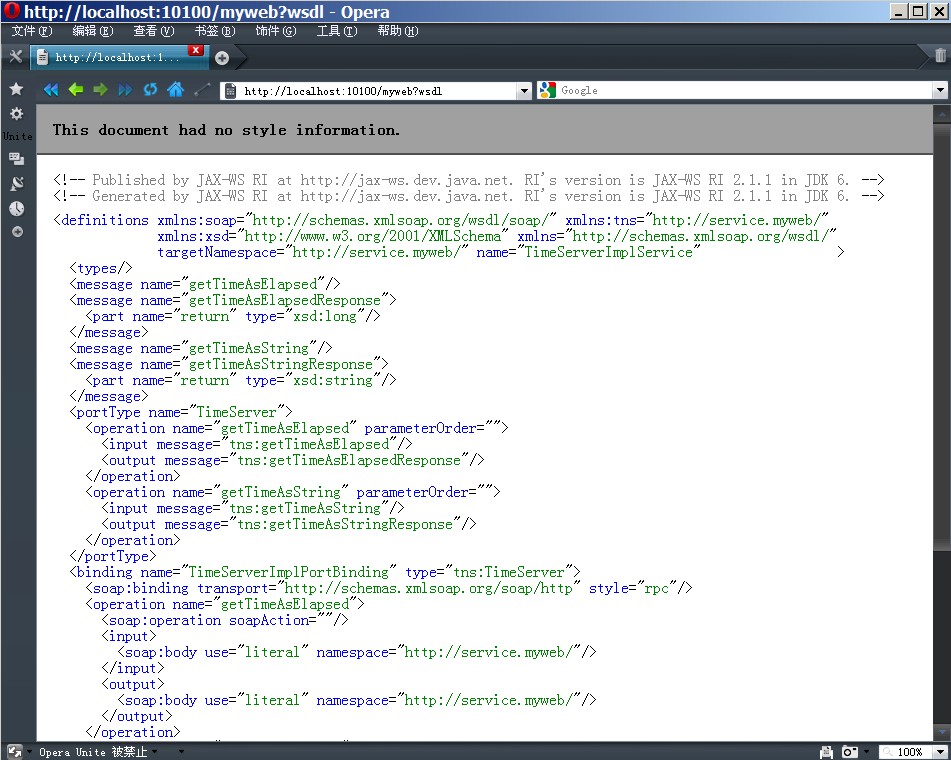
运行服务：

**java myweb/service/TimeServerPublisher**

在浏览器地址栏输入：

[**http://localhost:10100/myweb?wsdl**](http://localhost:10100/myweb?wsdl)

显示如下图所示：



也可编写客户端代码测试服务。

Java客户端：

**[java]** [view plain](http://blog.csdn.net/chszs/article/details/5050756) [copy](http://blog.csdn.net/chszs/article/details/5050756)

[print?](http://blog.csdn.net/chszs/article/details/5050756)

1. **package** myweb.client;
2. **import** javax.xml.namespace.QName;
3. **import** javax.xml.ws.Service;
4. **import** java.net.URL;
5. **import** myweb.service.\*;
6. **public** **class** TimeClient {
7. **public** **static** **void** main(String[] args) **throws** Exception{
8. URL url = **new** URL("http://localhost:10100/myweb?wsdl");
9. // 第一个参数是服务的URI
10. // 第二个参数是在WSDL发布的服务名
11. QName qname = **new** QName("http://service.myweb/","TimeServerImplService");
12. // 创建服务
13. Service service = Service.create(url, qname);
14. // 提取端点接口，服务“端口”。
15. TimeServer eif = service.getPort(TimeServer.**class**);
16. System.out.println(eif.getTimeAsString());
17. System.out.println(eif.getTimeAsElapsed());
18. }
19. }

运行客户端，显示结果如下：

**2009-12-21  
1261402511859**

也可用Ruby编写客户端，如下：

**[ruby]** [view plain](http://blog.csdn.net/chszs/article/details/5050756) [copy](http://blog.csdn.net/chszs/article/details/5050756)

[print?](http://blog.csdn.net/chszs/article/details/5050756)

1. #!/usr/bin/ruby
2. # one Ruby package for SOAP-based services
3. require 'soap/wsdlDriver'
4. wsdl\_url = 'http://127.0.0.1:10100/myweb?wsdl'
6. service = SOAP::WSDLDriverFactory.**new**(wsdl\_url).create\_rpc\_driver
7. # Save request/response messages in files named '...soapmsgs...'
8. service.wiredump\_file\_base = 'soapmsgs'
9. # Invoke service operations.
10. result1 = service.getTimeAsString
11. result2 = service.getTimeAsElapsed
12. # Output results.
13. puts "Current time is: #{result1}"
14. puts "Elapsed milliseconds from the epoch: #{result2}"

运行结果相同！