GLOBAL 2007

Обучение Java

Тема **Web Services**

Часть **Знакомство с веб-сервисами**

Автор **OMENDBA (omendba@gmail.com)**

20.12.2007

Web Services – это услуги, предоставляемые по WWW с использованием той или иной реализации языка XML, протокола HTTP или других Web-протоколов. Есть множество определений этого понятия.

Каждая фирма, разрабатывающая Web Services, дает свое определение Web-услуг.

Web Service – это приложение, идентифицируемое строкой URI, интерфейсы и связи которого могут определяться, описываться и отыскиваться документами XML. Оно взаимодействует напрямую с другими приложениями по межсетевым протоколам с помощью сообщений, записанных на языке XML.

По этому определению Web Service – это не всякая услуга, оказываемая посредством Web-технологии. Пересылка файлов, даже XML-документов, к ней не относится. Обычными Web-услугами пользуются независимые приложения: браузеры, связанные с Web-сервером только на время оказания услуги. В отличие от них услуги, названные “Web Services”, предоставляются, как правило, в рамках распределенного приложения. Можно сказать, что обычные Web-услуги предоставляются клиенту-человеку, а Web Services – это услуги, оказываемые клиенту-программе. Как правило, Web Service применяется как компонент распределенной информационной системы, разбросанной по компьютерам с разной архитектурой и разными средствами сетевой связи.

Процедуры и объекты, которые сервер предоставляет всем нуждающимся в них клиентам, называются распределенными процедурами и объектами. С точки зрения клиента это удаленные процедуры и объекты. Технологии обращения к удаленным процедурам и объектам существуют давно. Более того, есть несколько различных технологий. В голову сразу приходит множество аббревиатур: RPC, RMI, DCOM, COM+, CORBA, .NET. Однако их реализация ограничивается выбранной технологией: сокетами BSD, технологией Java или Microsoft Windows. Только технология CORBA претендует на всеобщность, но это делает ее чрезвычайно громоздкой и запутанной, потому что CORBA стремится использовать возможности всех или хотя бы наиболее распространенных платформ.

Использование распределенных процедур началось на рубеже 80-х годов с создания в фирме Xerox механизма вызова удаленных процедур RPC (Remote Procedure Call). Суть RPC заключается в том, что на машину клиента вместо вызываемой процедуры пересылается небольшой программный код, называемый заглушкой (stub). Заглушка внешне выглядит как вызываемая процедура, но ее код не выполняет процедуру, а преобразует (marshall) ее аргументы к виду, удобному для пересылки. Такое преобразование называется сборкой (marshalling). После сборки заглушка устанавливает связь с сервером по понятному для него протоколу и пересылает собранные аргументы на сервер. Клиент, вызвавший удаленную процедуру, взаимодействует не с ней, а с заглушкой, как с обычной локальной процедурой, выполняющейся на его машине. Сервер, получив собранные аргументы процедуры, разбирает (unmarshall) их, вызывает процедуру, передает ей параметры, дожидается ре-

зультата, собирает (marshall) его и пересылает заглушке. Заглушка снова разбирает (unmarshall) результат и передает его клиенту как обычная локальна процедура.

Протокол SOAP возник в 1998 году в фирме UserLand и корпорации Microsoft, но затем его разработка была передана в консорциум W3C, который и готовит сейчас рекомендации по его применению. Их можно посмотреть на странице проекта http://www.w3.org/TR/SOAP/.

Протокол SOAP не различает вызов процедуры и ответ на него, а просто определяет формат послания в виде документа XML. Послание может содержать вызов процедуры, ответ на него, запрос на выполнение каких-то других действий или просто текст. Спецификацию SOAP не интересует содержимое послания, она задает только его оформление.

Корневой элемент посылаемого документа XML <Envelope> содержит необязательный заголовок <Header> и обязательное тело <Body>. Схема SOAP-послания такова:

```
<?xml version='1.0'?>
<env:Envelope
     xmlns:env="http://www.w3.org/2002/06/soap-envelope">
   <env:Header>
      <!— Блоки заголовка —>
   </env:Header>
   <env:Body>
      <!— Содержимое послания —>
  </env:Body>
</env:Envelope>
```

В заголовке содержится один или несколько блоков, оформление и содержание которых никак не регламентируются. Точно так же ничего не говорится о содержании тела послания. Тем не менее, различают процедурный стиль послания SOAP, предназначенный для вызова удаленных процедур, и документный стиль, предназначенный для обмена документами XML. Процедурный стиль часто называют RPC-стилем, а документный стиль –XML-стилем.

Пакет SAAJ обеспечивает создание SOAP-посланий и обмен ими в синхронном режиме P2P (point-to-point). Часто такой способ обмена сообщениями оказывается невозможным из-за того, что участники обмена не всегда одновременно находятся на связи. В таких случаях приходится обмениваться сообщениями асинхронно, не дожидаясь ответа на посланное сообщение. Так работает электронная почта. В технологии Java асинхронный обмен SOAP-посланиями обеспечивается интерфейсами и классами пакета JAXM.

Веб-сервис под Sun Java Application Server Platform Edition 9.1.1

Код клиента, отсылающего SOAP-сообщение на сервер. Одновременно с этим клиент также получает SOAP-сообщение от сервера.

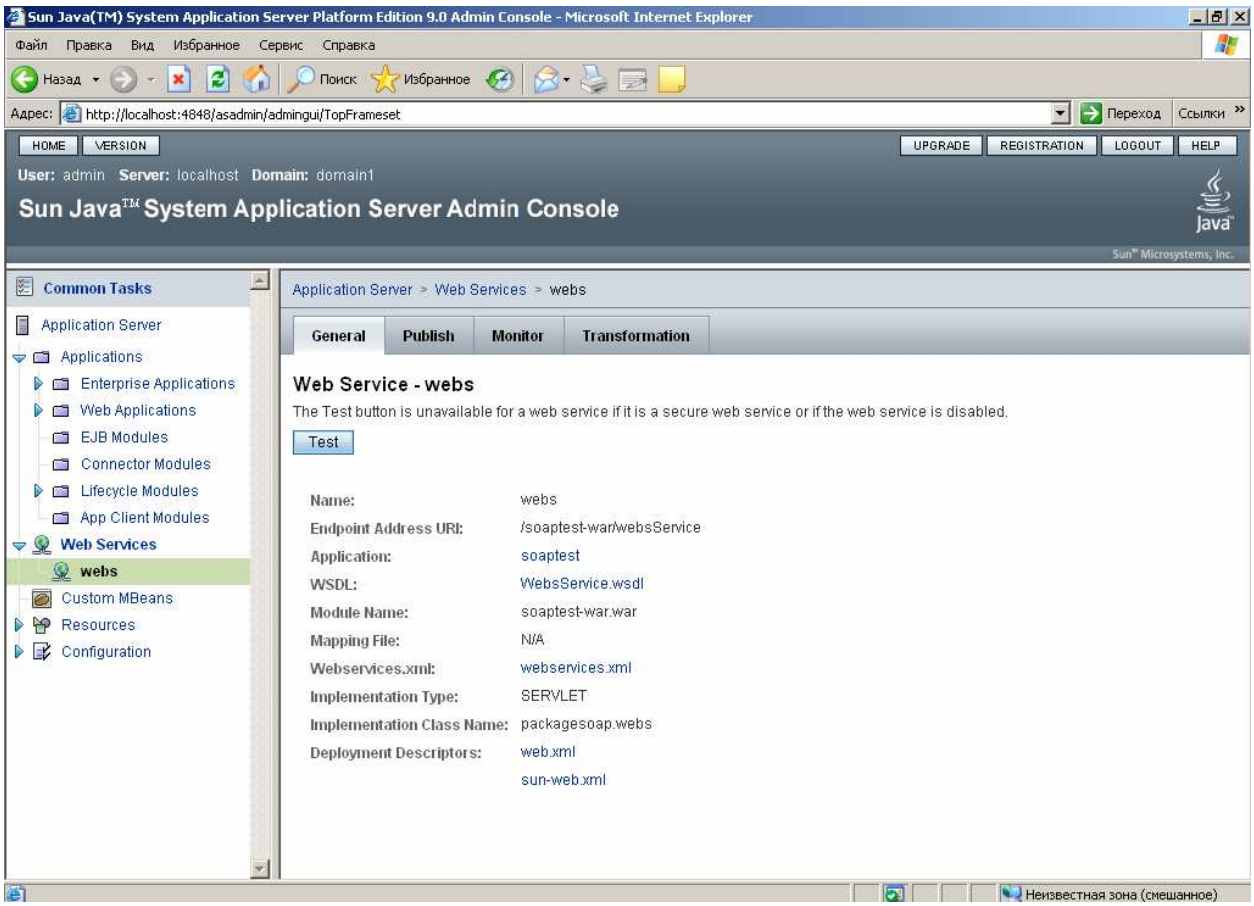
```
package packagesoap;
import java.net.*;
import java.io.*;
public class testSoap {
    public static void main(String[] args) throws Exception{
        String message ="<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://" +
                "schemas.xmlsoap.org/soap/envelope/'><SOAP-ENV:Header/>" +
                "<SOAP-ENV:Body><m:getTest xmlns:m='http://packagesoap/'/>" +
                "</SOAP-ENV:Body></SOAP-ENV:Envelope>";
        byte[] data = message.getBytes("UTF-8");
        URL url = new URL("http://localhost:8080/soaptest-war/websService");
        URLConnection uc = url.openConnection();
        uc.setDoOutput(true);
        uc.setDoInput(true);
        uc.setUseCaches(false) ;
        uc.setRequestProperty("Content-Type","text/xml;charset=\"utf-8\"") ;
        uc.setRequestProperty("SOAPAction","");
        uc.setRequestProperty("Content-Length",String.valueOf(message.length()));
        uc.connect();
        DataOutputStream dos =new DataOutputStream(uc.getOutputStream());
        dos.write(data, 0, data.length);
        dos.close();
        BufferedReader br = new BufferedReader(
                new InputStreamReader(uc.getInputStream(), "UTF-8"));
        String res = null;
        while ((res = br.readLine()) != null)
            System.out.println(res);
        br.close();
    }
}
```

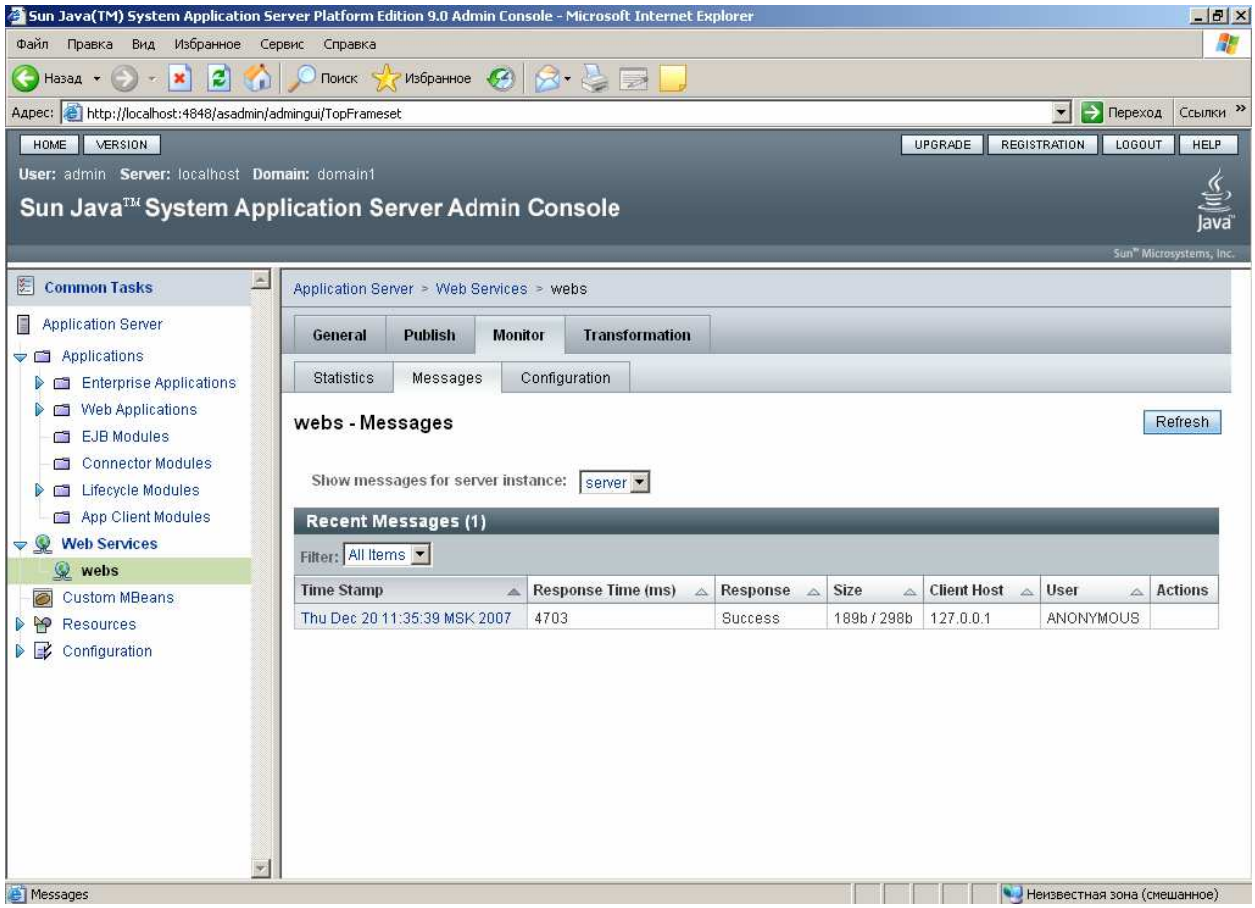
При выполнении данной программы получаем следующее сообщение:

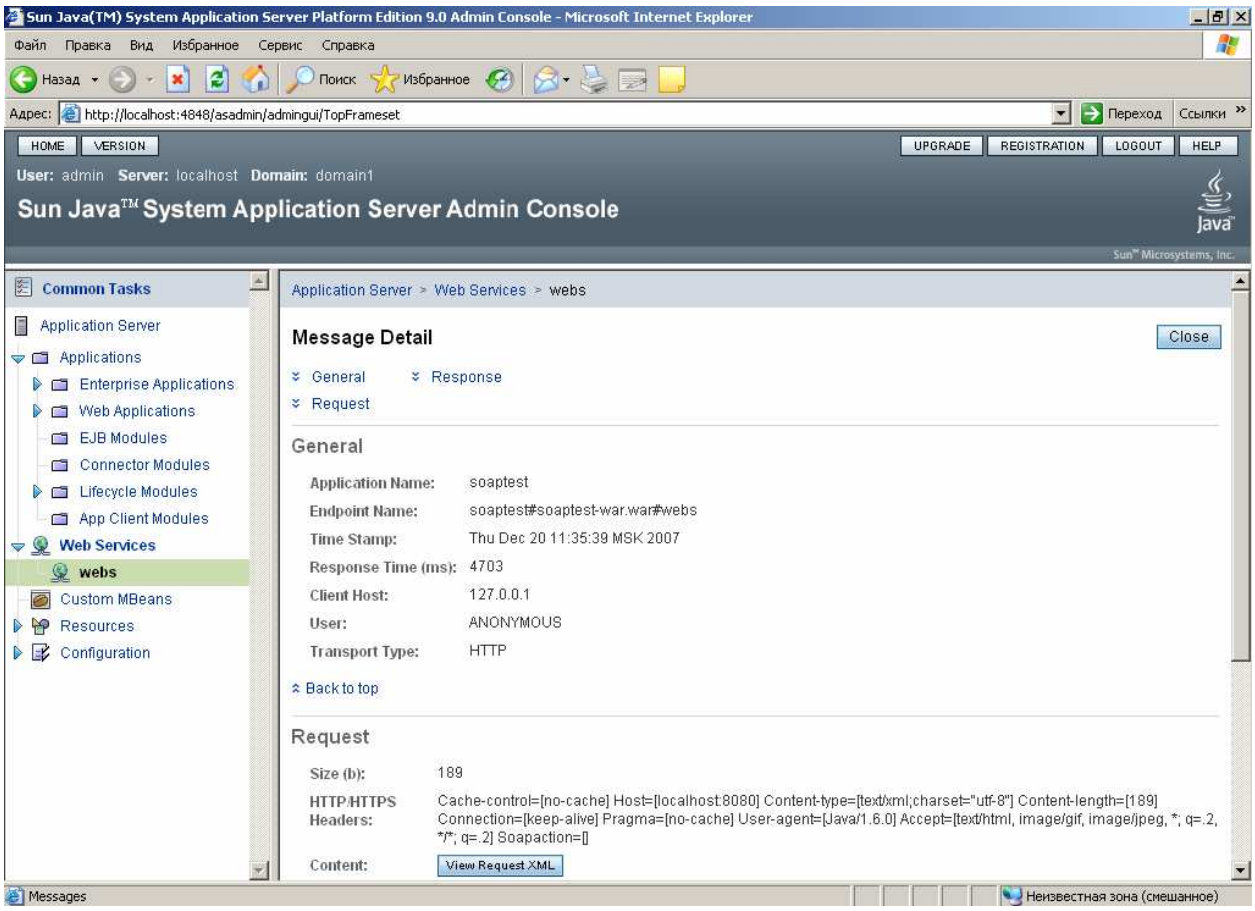
```
<?xml version="1.0" ?><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://packagesoap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soapenv:Body><ns1:getTestResponse><return>OK
WER_SERVICES</return></ns1:getTestResponse></soapenv:Body></soapenv:Envelope>
```

Код веб-сервиса, внедренного на J2EE-сервер с помощью NetBeans IDE 5.5:

```
package packagesoap;
import javax.jws.WebService;
@WebService()
public class webs {
      public String getTest() {
        return "OK WER_SERVICES";
    }
}
```


Sun Java(TM) System Application Server Platform Edition 9.0 Admin Console - Microsoft Internet Explorer
Файл Правка Вид Избранное Сервис Справка
Назад Поиск Избранное
Адрес: http://localhost:4848/asadmin/admingui/TopFrameset
Переход Ссылки
HOME VERSION
UPGRADE REGISTRATION LOGOUT HELP
User: admin Server: localhost Domain: domain1
Sun Java™ System Application Server Admin Console
Common Tasks
Application Server
Applications
Enterprise Applications
Web Applications
EJB Modules
Connector Modules
Lifecycle Modules
App Client Modules
Web Services
webs
Custom MBeans
Resources
Configuration
Application Server > Web Services > webs
General Publish Monitor Transformation
Web Service - webs
The Test button is unavailable for a web service if it is a secure web service or if the web service is disabled.
Test
Name: webs
Endpoint Address URI: /soaptest-war/websService
Application: soaptest
WSDL: WebsService.wsdl
Module Name: soaptest-war.war
Mapping File: N/A
Webservices.xml: webservices.xml
Implementation Type: SERVLET
Implementation Class Name: packagesoap.webs
Deployment Descriptors: web.xml
sun-web.xml
Неизвестная зона (смешанное)


Sun Java(TM) System Application Server Platform Edition 9.0 Admin Console - Microsoft Internet Explorer
Файл Правка Вид Избранное Сервис Справка
Назад Поиск Избранное
Адрес: http://localhost:4848/asadmin/admingui/TopFrameset
Переход Ссылки
HOME VERSION
UPGRADE REGISTRATION LOGOUT HELP
User: admin Server: localhost Domain: domain1
Sun Java™ System Application Server Admin Console
Common Tasks
Application Server
Applications
Enterprise Applications
Web Applications
EJB Modules
Connector Modules
Lifecycle Modules
App Client Modules
Web Services
webs
Custom MBeans
Resources
Configuration
Application Server > Web Services > webs
General Publish Monitor Transformation
Statistics Messages Configuration
webs - Messages
Refresh
Show messages for server instance: server
Recent Messages (1)
Filter: All Items
Time Stamp
Response Time (ms)
Response
Size
Client Host
User
Actions
Thu Dec 20 11:35:39 MSK 2007
4703
Success
189b / 298b
127.0.0.1
ANONYMOUS
Messages
Неизвестная зона (смешанное)



Request

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV='http://schemas.xmlsoap.org/soap/envelope/'><SOAP-ENV:Header/><SOAP-
ENV:Body><m:getTest xmlns:m='http://packagesoap/'/></SOAP-ENV:Body></SOAP-
ENV:Envelope>
```

Response

```
<?xml version="1.0" ?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://packagesoap/"><soapenv:Body><ns1:getTestResponse><return>OK
WER_SERVICES</return></ns1:getTestResponse></soapenv:Body></soapenv:Envelope>
```

getTest Method invocation

Method parameter(s)

Type	Value

Method returned

`java.lang.String : "OK WER_SERVICES"`

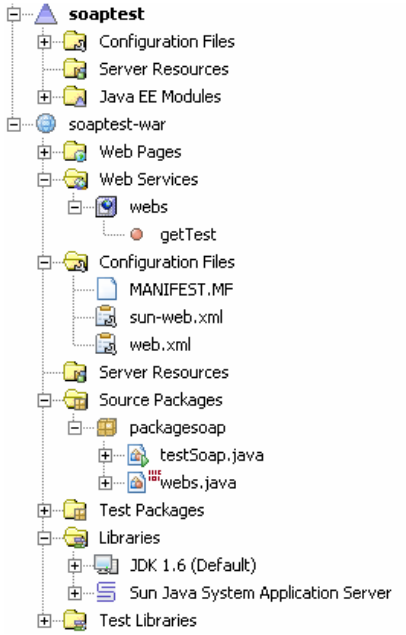
SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns1="http://packagesoap/">
    <soapenv:Body>
        <ns1:getTest/>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://packagesoap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <soapenv:Body>
        <ns1:getTestResponse>
            <return>OK WER_SERVICES</return>
        </ns1:getTestResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Структура приложения:



WebsService.wsdl:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://packagesoap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             targetNamespace="http://packagesoap/" name="websService">
    <types>
        <xsd:schema>
            <xsd:import namespace="http://packagesoap/"
                        schemaLocation="http://localhost:8080/soaptest-war/websService
                        /__container$publishing$subctx/WEB-INF/wsdl/WebsService_schema1.xsd"
                        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                        xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"/>
        </xsd:schema>
    </types>
    <message name="getTest">
        <part name="parameters" element="tns:getTest"/>
    </message>
    <message name="getTestResponse">
        <part name="parameters" element="tns:getTestResponse"/>
    </message>
    <portType name="webs">
        <operation name="getTest">
            <input message="tns:getTest"/>
            <output message="tns:getTestResponse"/>
        </operation>
    </portType>
    <binding name="websPortBinding" type="tns:webs">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                      style="document"/>
        <operation name="getTest">
            <soap:operation soapAction=""/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <service name="websService">
        <port name="websPort" binding="tns:websPortBinding">
            <soap:address location="http://localhost:8080/soaptest-war/websService"
                          xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                          xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"/>
        </port>
    </service>
</definitions>
```

Клиент на SAAJ:

```
package packagesoap;
import java.net.*;
import java.util.*;
import javax.xml.soap.*;
public class testSoap {
    public static void main(String[] args) throws Exception{
        SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
        SOAPConnection con = scf.createConnection();
        MessageFactory mf = MessageFactory.newInstance();
        SOAPMessage msg = mf.createMessage();
        msg.getMimeHeaders().setHeader("SOAPAction", "ServiceInfo");
        SOAPPart part = msg.getSOAPPart();
        SOAPEnvelope envelope = part.getEnvelope();
        SOAPBody body = envelope.getBody();
        Name bodyName = envelope.createName("getTest","m","http://packagesoap/");
        body.addBodyElement(bodyName);
        URL endpoint = new URL("http://localhost:8080/soaptest-war/websService");
        //Отправляем сообщение
        SOAPMessage response = con.call(msg, endpoint);
        //Анализируем ответ
        SOAPPart sp = response.getSOAPPart();
        SOAPEnvelope respEnv = sp.getEnvelope();
        SOAPBody respBody = respEnv.getBody();
        Iterator it=respBody.getChildElements();
        while (it.hasNext()) {
            SOAPBodyElement  bodyElement = (SOAPBodyElement)it.next();
            System.out.println(bodyElement.getTextContent());
        }
    }
}
```

В результате выполнения кода, получаем следующее:

```
OK WER_SERVICES
```

Клиент, использующий Axis:

```
package packagesoap;
import java.net.*;
import javax.xml.namespace.QName;
import org.apache.axis.*;
import org.apache.axis.client.*;
public class testSoap {
    public static void main(String[] args) throws Exception{
        Service service = new Service();
        Call call = (Call)service.createCall();
        String endpoint = "http://localhost:8080/soaptest-war/websService";
        call.setTargetEndpointAddress(new URL(endpoint));
        call.setOperationName(new QName("http://packagesoap/","getTest"));
        String response = (String)call.invoke(new Object[]{});
        System.out.println(response);
    }
}
```

В результате работы программы, получим результат:

```
OK WER_SERVICES
```

Request

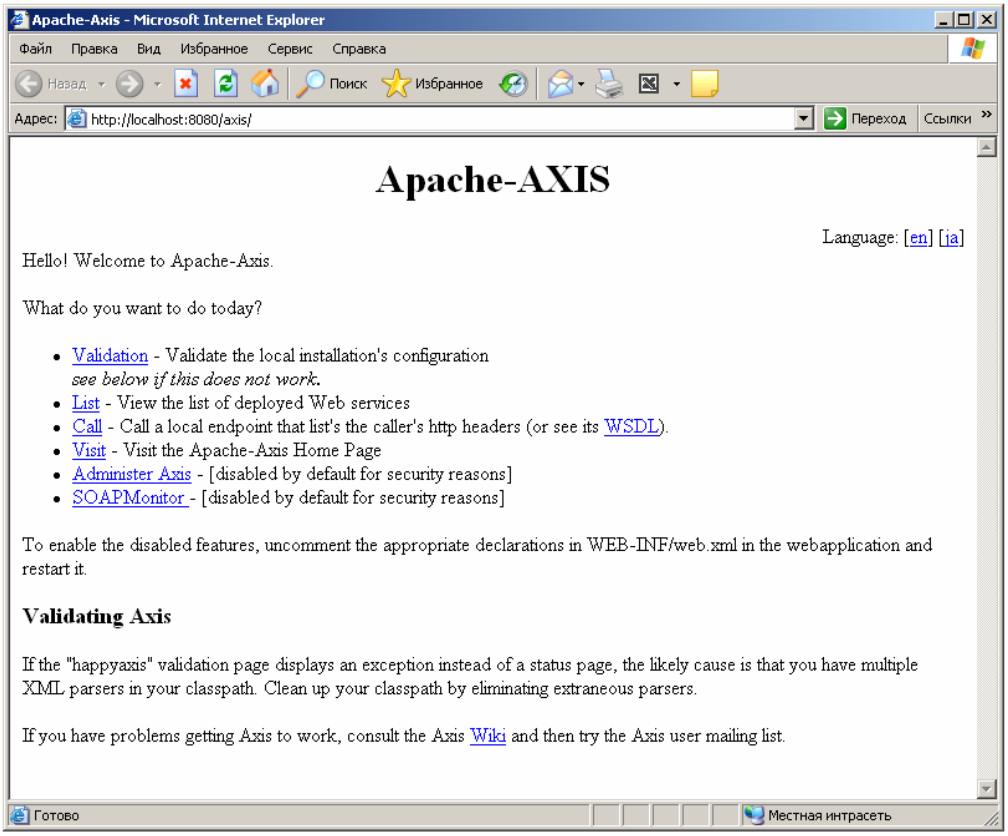
```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><soapenv:Body><ns1:getTest soap-
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://packagesoap/"/></soapenv:Body></soapenv:Envelope>
```

Response

```
<?xml version="1.0" ?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://packagesoap/"><soapenv:Body><ns1:getTestResponse><return>OK
WER_SERVICES</return></ns1:getTestResponse></soapenv:Body></soapenv:Envelope>
```

Веб-сервис под Tomcat 5.5.17

Для данного сервера приложений будем использовать Фреймворк Axis 1.4. Его установка заключается в копировании библиотек Axis в папку %TOMCAT %\server\lib. Затем скопируем папку axis из дистрибутива в папку %TOMCAT%\webapps. Осталось запустить браузер и проверить результаты работы:



Код веб-сервиса, находящегося в папке:

%TOMCAT_HOME%\webapps\axis\packageaxis

```
public class TestAxis {
    public String getTest() {
        return String.valueOf("J2ME + Axis 2007");
    }
}
```

Клиент, использующий библиотеки от Axis:

```
package packageaxis;
import org.apache.axis.client.*;
import java.net.*;
public class Main{
    public static void main(String[] args) throws Exception {
        Service service = new Service();
        Call call = (Call)service.createCall();
        String endpoint =
                "http://localhost:8080/axis/packageaxis/TestAxis.jws";
        call.setTargetEndpointAddress(new URL(endpoint));
        call.setOperationName("getTest");
        String response = (String)call.invoke(new Object[]{});
        System.out.println("Получено: " + response);
    }
}
```

```
Получено: OK Web Services from Axis!
```

Клиент, использующий SAAJ:

```
package packageaxis;
import java.net.*;
import java.util.*;
import javax.xml.soap.*;
public class Main{
    public static void main(String[] args) throws Exception {
        SOAPConnectionFactory scf = SOAPConnectionFactory.newInstance();
        SOAPConnection con = scf.createConnection();
        MessageFactory mf = MessageFactory.newInstance();
        SOAPMessage msg = mf.createMessage();
        msg.getMimeHeaders().setHeader("SOAPAction", "ServiceInfo");
        SOAPPart part = msg.getSOAPPart();
        SOAPEnvelope envelope = part.getEnvelope();
        SOAPBody body = envelope.getBody();
        Name bodyName = envelope.createName("getTest");
        body.addBodyElement(bodyName);
        URL endpoint = new URL("http://localhost:8080/axis/packageaxis/TestAxis.jws");
        //Отправляем сообщение
        SOAPMessage response = con.call(msg, endpoint);
        //Анализируем ответ
        SOAPPart sp = response.getSOAPPart();
        SOAPEnvelope respEnv = sp.getEnvelope();
        SOAPBody respBody = respEnv.getBody();
        Iterator it=respBody.getChildElements();
        while (it.hasNext()) {
            SOAPBodyElement  bodyElement = (SOAPBodyElement)it.next();
            System.out.println(bodyElement.getTextContent());
        }
    }
}
```

```
OK Web Services from Axis!
```

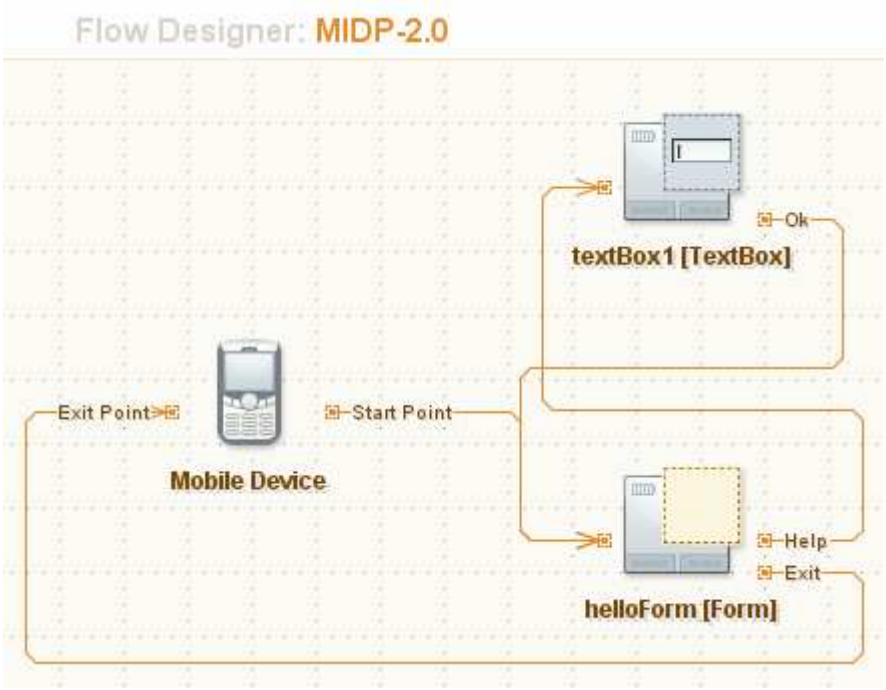
Структура приложения:

Обратившись по адресу
http://localhost:8080/axis/packageaxis/TestAxis.jws?WSDL
получим следующий код в браузере:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/axis/packageaxis/TestAxis.jws"
                  xmlns:apachesoap="http://xml.apache.org/xml-soap"
                  xmlns:impl="http://localhost:8080/axis/packageaxis/TestAxis.jws"
                  xmlns:intf="http://localhost:8080/axis/packageaxis/TestAxis.jws"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!--WSDL created by Apache Axis version: 1.4
        Built on Apr 22, 2006 (06:55:48 PDT)-->
    <wsdl:message name="getTestResponse">
        <wsdl:part name="getTestReturn" type="xsd:string"/>
    </wsdl:message>
    <wsdl:message name="getTestRequest">
    </wsdl:message>
    <wsdl:portType name="TestAxis">
```

```
        <wsdl:operation name="getTest">
            <wsdl:input message="impl:getTestRequest" name="getTestRequest"/>
            <wsdl:output message="impl:getTestResponse" name="getTestResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="TestAxisSoapBinding" type="impl:TestAxis">
        <wsdlsoap:binding style="rpc"
                          transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="getTest">
            <wsdlsoap:operation soapAction=""/>
            <wsdl:input name="getTestRequest">
                <wsdlsoap:body
                    encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://DefaultNamespace" use="encoded"/>
            </wsdl:input>
            <wsdl:output name="getTestResponse">
                <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                               namespace="http://localhost:8080/axis/packageaxis/TestAxis.jws"
                               use="encoded"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="TestAxisService">
        <wsdl:port binding="impl:TestAxisSoapBinding" name="TestAxis">
            <wsdlsoap:address
                location="http://localhost:8080/axis/packageaxis/TestAxis.jws"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Доступ в веб-сервису с помощью J2ME и kSOAP




```
package hello;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import org.ksoap2.*;
import org.ksoap2.transport.*;
import org.ksoap2.serialization.*;
class threadJ2ME extends Thread {
    public TextBox tb;
    public void run() {
        String endPointURL="http://localhost:8080/axis/packageaxis/TestAxis.jws";
        SoapObject method = new SoapObject(endPointURL,"getTest");
        SoapEnvelope envelope =
                new SoapSerializationEnvelope(SoapEnvelope.VER10);
        envelope.bodyOut = method;
        HttpTransport rpc = new HttpTransport(endPointURL);
        try {
```

```
            rpc.call(null, envelope);
        } catch (Exception e) {}
        SoapObject ret = (SoapObject) envelope.bodyIn;
        String echo = String.valueOf(ret.getProperty(0));
        tb.insert(echo,0);
    }
}
public class HelloMidlet extends MIDlet implements CommandListener {
    ...
    public void commandAction(Command command, Displayable displayable) {
        // Insert global pre-action code here
        if (displayable == helloForm) {
            if (command == exitCommand) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            } else if (command == helpCommand1) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_textBox1());
                threadJ2ME p = new threadJ2ME ();
                p.tb=this.textBox1;
                p.start();
            }
        } else if (displayable == textBox1) {
            if (command == okCommand3) {
                // Insert pre-action code here
                getDisplay().setCurrent(get_helloForm());
                // Insert post-action code here
            }
        }
        // Insert global post-action code here
    }
    ...
}
```