

## **СОЗДАНИЕ ВЕБ-СЛУЖБ\***

Практическое руководство

Д.А. Семячкин

\* \* \*

[dms@keldysh.ru](mailto:dms@keldysh.ru)

---

*\* Работа выполнена при поддержке Российского фонда фундаментальных исследований (проекты 05-01-00626-а, 04-07-90299-в).*

---

*Как превратить Java-класс в веб-службу? Какие программные средства для этого можно использовать и какова технологическая цепочка изготовления веб-службы и клиента к ней? Эти и некоторые другие вопросы будут рассмотрены в этом руководстве.*

На сегодняшний день существует большое количество некоммерческих и коммерческих средств разработки веб-служб. В списке [1], приводимом на сайте UBS, содержится 46 существующих подобных средств на январь 2005 года как крупных (IBM, Microsoft, Sun Microsystems, Apache Software и др.), так и небольших производителей программного обеспечения.

Все они различаются по степени полноты, возможностям, условиям лицензирования, требованиям к программно-аппаратному обеспечению и т.п. Выбор средства разработки может зависеть от степени подготовленности разработчика и решаемой задачи. В большинстве случаев можно рекомендовать широко используемый во многих программных разработках инструментальный пакет Apache Axis [2] от Apache Software Foundation, интегрируемый в веб-сервер Apache's Jakarta Tomcat [3]. Отличительной особенностью Axis является наличие в нем утилит (WSDL2java, java2WSDL), позволяющих автоматизировать некоторые стадии создания веб-службы и клиента к ней, а также обширной сопроводительной документации и примеров.

Перед началом работы необходимо выполнить ряд подготовительных действий:

1. Скачать дистрибутив веб-сервера Apache's Jakarta Tomcat и установить его, пользуясь руководством по установке, которое можно найти на сайте [3].
  2. Скачать дистрибутив Axis и установить его, распаковав дистрибутив в директорию `AXIS_HOME` и переписав директорию `AXIS_HOME/webapps/axis` в директорию `TOMCAT_HOME/webapps`:
3. Настроить среду разработки, добавив в переменную окружения `CLASSPATH` пути к библиотекам `AXIS`, XML-парсеру (например, `Xerces`, поставляемый вместе с `Axis`) и пользовательским классам:

```
> export CLASSPATH=\
$AXIS_HOME/src/axis/lib/axis.jar:\
$AXIS_HOME/src/axis/lib/jaxrpc.jar:\
$AXIS_HOME/src/axis/lib/commons-logging.jar:\
$AXIS_HOME/src/axis/lib/commons-discovery.jar:\
$AXIS_HOME/src/axis/lib/tt-bytecode.jar:\
$AXIS_HOME/src/axis/lib/wsdl4j.jar:\
$TOMCAT_HOME/common/lib/xerces.jar:\
$AXIS_HOME/src/axis:\
$CLASSPATH
```

Axis предоставляет два способа превращения Java-класса в веб-службу: Instant Deployment (“быстрое” создание веб-службы) и Custom Deployment (способ, основанный на WSDD – Web Service Deployment Descriptor).

Кроме самого Java-класса (или его интерфейса) разработчику необходимо знать адрес (URI), где будет располагаться веб-служба.

Для “быстрого” создания веб-службы (JWS, Java Web Service) из Java-класса требуется сделать следующее:

1. Переименовать файл, содержащий исходный код Java-класса, в файл с расширением `jws`.
2. Переместить этот файл в директорию `ТОМКАТ_НОМЕ/webapps/axis`.

После этого все публичные (`public`) методы Java-класса становятся доступными через веб-службу.

Несмотря на простоту этого способа, он обладает рядом недостатков:

- служба не содержит Java-классы, инкапсулирующие в себе сетевые операции, которые потом придется реализовывать при создании клиента;
- JWS требует наличие исходного кода всех используемых в программе Java-классов, которого может не быть;
- можно использовать только стандартные типы данных – `Integer`, `Double` и др., т.к. средствами JWS отсутствует возможность преобразовывать (`serialize`) и возвращать (`return`) через веб-службу нестандартные типы;
- средствами JWS нет возможности организации аутентификации и контроля доступа к службе.

По этим причинам предпочтительнее использовать второй способ, основанный на механизме дескрипторов веб-служб (WSDD).

Пользуясь вторым способом, необходимо сделать следующее:

1. Объявить в начале Java-класса имя пакета (`package`).
2. Скомпилировать исходный код Java-класса, используя компилятор `javac`, получить скомпилированный Java-класс.
3. Создать структуру директорий, соответствующую имени пакета, после чего скопировать туда скомпилированный Java-класс.
4. Переписать структуру директорий со всем содержимым в директорию `ТОМКАТ_НОМЕ/webapps/axis/WEB-INF/classes`.
5. Подготовить дескриптор веб-службы `<имя_службы>-deploy.wsdd`, например:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
    <service name="<имя_службы>" provider="java:RPC">
```

```
        <parameter name="className"
value="<имя_класса_службы>" />
        <parameter name="allowedMethods" value="*" />
    </service>
</deployment>
```

в котором нужно указать имя службы, имя класса службы, список методов класса, которые будут доступны через службу (“\*” означает, что будут доступны все публичные (public) методы класса).

6. Зарегистрировать веб-службу на сервере, используя утилиту Axis AdminClient:

```
> java org.apache.axis.client.AdminClient -p <порт> \
<имя_службы>-deploy.wsdd
<Admin>Done processing</Admin>
```

Эта утилита просто копирует содержимое файла <имя\_службы>-deploy.wsdd в конфигурационный файл сервера Axis `ТOMCAT_НОМЕ/webapps/axis/WEB-INF/server-config.wsdd`. Поэтому можно не пользоваться этой утилитой, а просто вставить описание службы в этот файл.

Приведенный способ устраняет часть минусов первого способа, однако не упрощает создания клиента к веб-службе, кроме того, появляется необходимость создавать вручную дескриптор веб-службы.

Имеющиеся в Axis средства позволяют упростить создание веб-службы и клиента к ней. Рассмотрим модификацию второго способа с использованием этих средств (рис. 1).

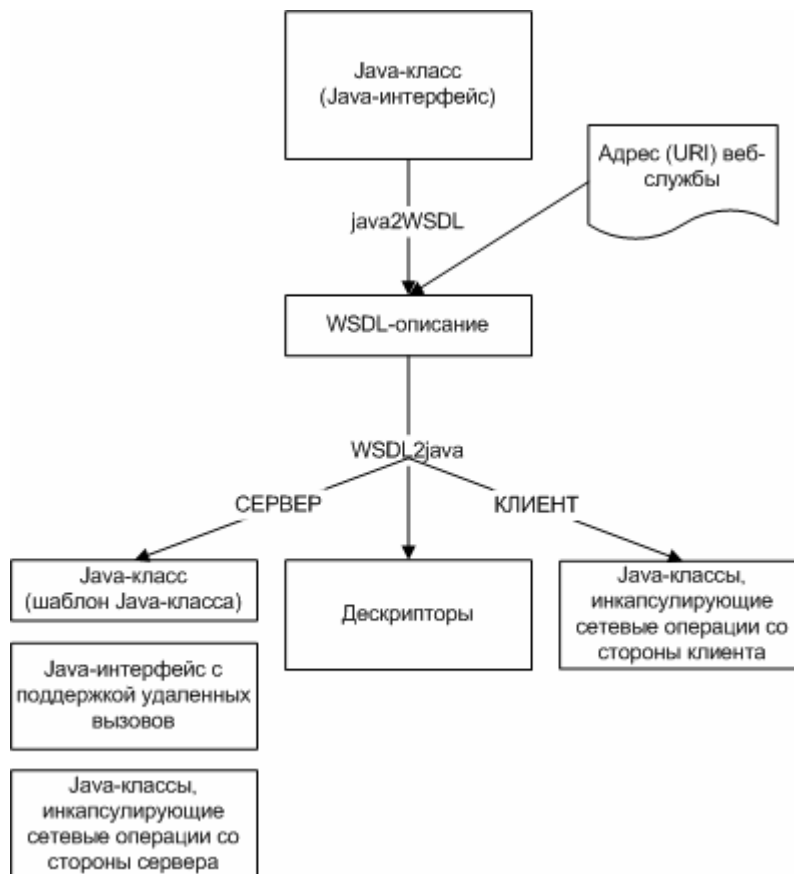


Рисунок 1

Пусть имеется исходный код Java-класса (или его интерфейса).

1. Создадим структуру директорий, соответствующую имени пакета, после чего скопируем туда исходный код этого Java-класса (или его интерфейса).
2. Перепишем структуру директорий со всем содержимым с директорию `AXIS_HOME/src/axis/`.
3. Используя утилиту `java2WSDL`, создадим WSDL-описание для этого интерфейса:

```

> java org.apache.axis.wsdl.Java2WSDL \
  -o <WSDL_описание> \
  -l http://localhost:<порт>/axis/services/<имя\_службы> \
  -n "urn:<пространство_имен>" \
  -p "<имя_пакета>" "urn: <пространство_имен>" \
  <имя_пакета>.<имя_службы>
  
```

где

“-o” – имя файла, в котором после выполнения команды будет содержаться WSDL-описание;

“-l” – расположение службы (endpoint);

“-n” – пространство имен;

“-p” – отображение пакета в пространство имен;  
последний параметр – Java-класс или Java-интерфейс, содержащий методы веб-службы.

Замечание. Утилита Java2WSDL имеет множество дополнительных параметров, описанных здесь:

<http://ws.apache.org/axis/java/reference.html#Java2WSDLReference>.

4. Запускаем утилиту WSDL2java, подавая ей на вход полученное, на предыдущем шаге WSDL-описание:

```
> java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s \  
-S true -Nurn: <пространство_имен> <имя_пакета> <WSDL_описание>
```

В результате выполнения этой утилиты будут автоматически сгенерированы файлы, часть из которых предназначены для службы, часть – для клиента (выделены жирным шрифтом):

- Шаблон Java-класса, который необходимо наполнить исходным кодом веб-службы (<имя\_службы>SoapBindingImpl.java);
- Интерфейс Java-класса с поддержкой удаленных вызовов java.rmi.Remote (<имя\_службы>.java);
- Java-файлы, инкапсулирующие в себе сетевые операции со стороны сервера и клиента (<имя\_службы>Service.java, <имя\_службы>**ServiceLocator.java**, <имя\_службы>SoapBindingSkeleton.java, <имя\_службы>**SoapBindingStub.java**);
- Дескриптор регистрации (deployment) веб-службы (deploy.wsdd);
- Дескриптор удаления (undeployment) веб-службы (undeploy.wsdd)

и другие файлы, связанные с реализацией службы.

Таким образом, мы получили все необходимые файлы для регистрации службы и создания клиента к ней без дополнительной ручной работы.

5. Компилируем все эти файлы:

```
> javac *.java
```

6. Заходим в систему под пользователем root и создаем директорию `$TOMCAT_HOME/webapps/axis/WEB-INF/classes/<имя_пакета>`, после чего копируем туда полученные на предыдущем шаге скомпилированные Java-классы веб-службы:

```
> cp <имя_службы>.class <имя_службы>SoapBindingImpl.class \  
<имя_службы>SoapBindingSkeleton.class \  
$TOMCAT_HOME/webapps/axis/WEB-INF/classes/<имя_пакета>
```

7. Регистрируем веб-службу на сервере, используя утилиту Axis AdminClient:

```
> java org.apache.axis.client.AdminClient -p <порт> \  
deploy.wsdd \  
<Admin>Done processing</Admin>
```

## 8. Создание клиента к веб-службе

При использовании сгенерированных автоматически Java-классов, инкапсулирующих в себе сетевые операции, создание клиента существенно упрощается и становится полностью прозрачным для разработчика. Вызов удаленного метода веб-службы напоминает вызов локального метода и выглядит следующим образом:

1. Создается объект locator, содержащий информацию о расположении службы:  
`<имя_службы>ServiceLocator locator = new <имя_службы>ServiceLocator();`
2. Создается объект stub, содержащий информацию о методах службы:  
`<имя_службы>SoapBindingStub stub = (<имя_службы>SoapBindingStub) locator.get<имя_службы>();`
3. Вызывается метод службы:  
`stub.<метод_службы>;`

Более подробно о клиентском приложении можно прочитать здесь: <http://ws.apache.org/axis/java/client-side-axis.html>

Замечание. Все эти операции можно автоматизировать с помощью программного средства Apache Ant [5]. Подробнее об этом здесь: <http://ws.apache.org/axis/java/ant/ant.html>. Пример файла build.xml можно найти в примерах, включенных в дистрибутив Apache Axis.

## ЛИТЕРАТУРА

[1] Некоммерческие средства разработки веб-сервисов и XML-приложений

[http://www.ubs.ru/ws/ws\\_noncomdevtools1.html](http://www.ubs.ru/ws/ws_noncomdevtools1.html)

[2] Apache Axis

<http://ws.apache.org/axis>

[3] Apache's Jakarta Tomcat

<http://jakarta.apache.org/tomcat/>

[4] Документация по Apache Axis

<http://ws.apache.org/axis/java/>

[5] Apache Ant

<http://ant.apache.org/>