

# Введение в технологию Web-служб.

## Часть 3

Константин Разумовский

Этой статьей мы завершаем серию материалов, посвященных введению в технологию Web-служб. В предыдущих частях статьи мы ввели понятие *распределенной системы*, понимая под этим программную систему, компоненты которой функционируют на различных физических устройствах в сети, и отметили, что огромное количество существующих программных систем являются распределенными. Мы указали на то, что одной из главных проблем таких систем является сложность интеграции неоднородных компонентов системы. Мы выяснили, что эту проблему можно успешно решить с помощью технологии Web-служб. Мы обсудили определение Web-службы, отметив, что некоторая программная система может считаться Web-службой, если удовлетворяет двум основным условиям. Во-первых, она должна предоставлять свое описание на некотором основанном на XML языке, а во-вторых, она должна уметь общаться с другими программными системами с помощью XML-сообщений, передаваемых по какому-либо Интернет-протоколу. Мы также обсудили очень популярную в настоящее время концепцию *архитектуры, ориентированной на службы*. В рамках данной концепции любой программный компонент рассматривается как *служба*, доступ к которой может быть динамически получен некоторой заинтересованной в этом стороной при посредничестве так называемого *реестра служб*. Данная модель заставляет рассматривать технологию Web-служб не только и не столько как очередное техническое нововведение, но и как новое решение для мира бизнеса.

В сегодняшней статье мы постараемся перейти от теории к практике: мы поговорим о методах и инструментах для разработки Web-служб, обсудим конкретные шаги, необходимые для создания Web-служб и проиллюстрируем все это на простом примере.

### 1. Что делать?

Итак, мы захотели решить задачу интеграции компонентов некоторой распределенной системы с помощью технологии Web-служб. Прежде всего, перечислим шаги, которые необходимо пройти в процессе создания Web-службы, а затем поговорим о том, какова должна быть их последовательность, и от чего она зависит. Итак, обычно при создании Web-службы приходится решать следующие основные задачи:

**1. Создание приложения, реализующего функциональность Web-службы.** Как мы отмечали в предыдущей части статьи, в качестве такого приложения могут выступать класс Java или .Net, компонент EJB или COM, унаследованное приложение (legacy application), написанное, скажем, на COBOL или PL, хранимая процедура базы данных и т.д. Поскольку Web-службы часто используются для интеграции уже существующих систем, где приложения создавались еще до принятия решения обернуть их в Web-службы, этот шаг может не требовать каких-либо дополнительных усилий.

**2. Создание описания интерфейса Web-службы.** Когда мы определяли Web-службу, мы отметили, что ее интерфейс должен быть описан на некотором, основанном на XML языке, и в настоящее время чаще всего для описания интерфейса используется язык **WSDL** (Web Services Description Language). Описание интерфейса определяет набор операций, которые поддерживает данная Web-служба, а также количество и тип параметров этих операций. Ссылка на интерфейс Web-службы может публиковаться в реестрах служб, таких, как **UDDI** (см. предыдущую часть статьи). В некоторых случаях, однако, нет необходимости создавать в явном виде файл с описанием интерфейса, поэтому и этот шаг приходится выполнять не всегда.

**3. Опубликование описания интерфейса в реестре служб.** Этот шаг опять-таки является необязательным, он, как правило, необходим в том случае, если мы хотим, чтобы нашу службу могли найти во внешнем (public) реестре служб и, возможно, динамически вызвать.

**4. Развертывание Web-службы (Web service deployment).** Это важнейший, и часто, самый трудоемкий шаг при создании Web-службы. Можно говорить о двух логических компонентах, участвующих в вызове Web-службы - клиентском приложении, инициирующем вызов, и серверной части, принимающей запрос клиента и непосредственно осуществляющей вызов. На шаге развертывания мы должны предпринять действия, которые позволят осуществить вызов Web-службы на стороне сервера. Необходимые для развертывания действия зависят от реализации

Web-службы, платформы, на которой она работает, протокола, по которому ее вызывают, а также конкретных продуктов, обеспечивающих транспорт и среду работы Web-службы. В данной статье мы будем рассматривать классический вариант вызова Web-службы с помощью протокола SOAP, а в качестве его реализации использовать популярный open-source продукт Apache Axis. Сама Web-служба, которую мы будем вызывать, реализована в виде Java-класса. Для такой Web-службы развертывание включает создание J2EE enterprise-приложения и установка его на сервере приложений, таком как Apache Tomcat, IBM WebSphere Application Server, BEA WebLogic Server или каком-либо другом. При этом наше приложение должно содержать код реализации Web-службы, а также так называемый *дескриптор развертывания SOAP*, который позволяет связать адрес Web-службы с реализующим ее программным компонентом (в нашем случае – Java-классом).

**5. Создание описания реализации Web-службы.** На втором шаге мы обсуждали создание описания интерфейса Web-службы на языке WSDL. Теперь, когда мы развернули Web-службу на сервере, мы можем создать WSDL-документ, содержащий информацию о реализации службы. Этот документ должен содержать ссылку на интерфейс Web-службы, который он реализует, и также может публиковаться в реестрах служб, таких, как UDDI. Надо отметить, что одному и тому же интерфейсу могут соответствовать несколько различных реализаций (и соответственно, их описаний).

**6. Опубликование описания реализации в реестре служб.** Этот шаг является необязательным. Как уже было сказано, он необходим в том случае, если мы хотим, чтобы нашу службу могли найти во внешнем реестре служб и, возможно, динамически вызвать.

**7. Создание клиентского приложения для доступа к Web-службе.** Как правило, на основе описания реализации Web-службы создается так называемый прокси-класс, который используется клиентским приложением и скрывает в себе все сложные подробности вызова Web-службы (например, использование API реализации протокола SOAP).

## 2. Четыре пути

Итак, мы выяснили, **что** необходимо сделать для создания Web-службы. Но с чего

следует начать разработку, и какова должна быть последовательность действий? В зависимости от того, с какими начальными данными мы приступаем к разработке Web-службы, различают четыре основных сценария создания Web-служб [1]:

	Интерфейс Web-службы не определен	Интерфейс Web-службы определен
Реализации не существует	разработка «с нуля»	разработка «сверху вниз»
Реализация существует	разработка «снизу вверх»	«встреча посередине»

**Таблица 1. Сценарии создания Web-службы.**

Сценарий разработки **«снизу вверх»** соответствует самому, пожалуй, распространенному случаю, когда приложение, реализующее функциональность Web-службы уже существует. Как мы рассказывали в предыдущих части статьи, в Web-службы часто «оборачивают» унаследованные приложения, предоставляя, таким образом современный, универсальный интерфейс к коду, написанному, быть может, десятки лет назад на каком-нибудь мало используемом сегодня языке программирования. Используя готовую реализацию и современные средства разработки Web-служб (о которых мы расскажем ниже) легко автоматически сгенерировать описание Web-службы. Далее, опять-таки используя автоматизированные средства разработки, необходимо развернуть службу на сервере приложений, а затем, при необходимости, опубликовать в реестре служб.

Сценарий разработки **«с нуля»** (не существует ни реализации, ни интерфейса службы) отличается от вышеописанного лишь тем, что, прежде всего, необходимо реализовать программный компонент, собственно несущий в себе функциональность будущей Web-службы. Как мы уже сказали, реализовывать его можно практически на любом удобном для разработчика языке программирования. Далее необходимо выполнить те же действия, что и в предыдущем сценарии.

О разработке **«сверху вниз»** говорят в том случае, когда имеется описание интерфейса Web-службы (обычно на языке WSDL), однако реализация еще не создана. Понятно, что в этом случае, прежде всего, необходимо реализовывать службу на некотором языке программирования. При этом важно, чтобы реализация в точности соответствовала заявленному интерфейсу. Тут нам на помощь снова могут прийти современные средства разработки, которые позволяют по описанию Web-службы создать так называемый «скелет» (skeleton) – основу будущей реализации, на которую

программист должен нарастить «мышцы», добавив в методы классов необходимую функциональность.

Последний сценарий «**встреча посередине**» предполагает, что в наличии имеется как реализация службы, так и описание ее интерфейса. В этом случае может возникнуть проблема несоответствия имеющейся реализации заявленному интерфейсу. Решить эту проблему можно, например, «обернув» имеющуюся реализацию в новую, соответствующую интерфейсу службы.

### **3. Инструменты**

Из сказанного выше понятно, что процесс создания Web-служб вовсе не так «элементарен», как это часто пытаются представить в статьях, посвященных продвижению этой популярной технологии на рынок. Этот процесс требует создания достаточно большого количества различного рода программных артефактов (реализация службы, WSDL-файлы с описанием службы, дескрипторы развертывания и т.д.), что может быть весьма трудоемко и чревато ошибками, если делать все это вручную. К счастью, современные средства разработки (например, такие, как IBM WebSphere Studio Application Developer или Microsoft Visual Studio .NET), позволяют в значительной мере автоматизировать процесс создания и развертывания Web-служб.

Поскольку в настоящее время существует уже достаточно много средств для создания и развертывания Web-служб, а также серверов приложений, поддерживающих Web-службы, мы упомянем лишь инструменты, которые пользуются наибольшей популярностью среди разработчиков. В сложном деле определения популярности мы доверимся мнению читателей известного журнала “**Web Services Journal**” (<http://www.sys-con.com/webservices/>), издаваемого компанией SYS-CON Media (она также издает такие журналы как “Java Developer’s Journal”, “NET Developer’s Journal” и др.). Награды SYS-CON Media имеют достаточно высокий авторитет, иногда их называют «Оскарами» в отрасли программного обеспечения. Итак, в 2002 году, выбор читателей журнала («Readers' Choice Award») в номинации «Лучший сервер приложений для Web-служб» пал на сервер компании BEA Systems - **BEA WebLogic Server 6.1**. Продукт компании IBM- **WebSphere Application Server** занял в этой номинации второе место. Другое решение IBM - **WebSphere Studio Application Developer**, было признано читателями лучшей интегрированной средой для разработки

Web-служб. Первенствовала IBM и в номинации «Лучший сайт, посвященный технологии Web-служб». Победителем тут признан ресурс **IBM developerWorks**, предлагающий бесплатные статьи, учебные курсы и пособия для разработчиков, относящиеся, в том числе, и к технологии Web-служб (<http://www-106.ibm.com/developerworks/webservices/>).

В целом же наибольшее количество читательских наград получили IBM, BEA ,Oracle, и Borland, что и неудивительно: именно эти компании (наряду с Microsoft) являются лидерами в области создания различного рода решений для Web-служб.

## 4. Ультрасовременная классика

Как мы отметили в предыдущем разделе, процесс создания Web-службы является достаточно трудоемким, а развертывание Web-службы обычно требует наличия сервера приложений. Поэтому далеко не каждый читатель сможет создать и запустить службу у себя дома. Кроме того, сама природа Web-служб предполагает обмен сообщениями с удаленными компьютерами. Учитывая все это, а также ограничения на объем данного материала, мы приводим в статье лишь пример **клиентского** приложения, которое будет вызывать Web-службу, развернутую в интернете на сервере <http://www.xmethods.net/> . Данный демонстрационный ресурс содержит большую коллекцию готовых к использованию развернутых Web-служб, выполняющих самые различные функции. С их помощью вы можете, например, мгновенно узнать погоду и температуру в любой точке земного шара, послать SMS сообщение на мобильный телефон, получить последние новости и курсы валют или даже определить, какому произведению Шекспира принадлежит любимая вами цитата, и каким персонажем она была произнесена. Мы, однако, обратимся к другому, быть может, менее эффективному, но зато гораздо более известному примеру.

Чтобы быть современным, необходимо знать классику. Именно поэтому в нашей статье мы обсудим ставший уже классическим для технологии Web-служб пример с получением котировок акций компаний (stockquotes). Текущие курсы акций компаний на бирже всегда представляют интерес для игроков на бирже, брокеров, владельцев акций, а ввиду небывалых скачков курсов и индексов на фондовом рынке в последние месяцы, этот пример только набирает актуальность.

Итак, предположим, что у нас есть некая распределенная система, и назрела

необходимость обрабатывать или отображать в этой системе информацию о текущих курсах акций различных компаний. Может быть, нам надо программно проанализировать эти курсы, чтобы принять некое решение (купить/продать те или иные акции), а может быть, мы просто содержим интернет-портал и хотим предоставить информацию о курсах пользователям портала через браузер, как это описано в [2]. В любом случае, при полном отсутствии специальных средств и технологий для решения этой задачи, нам придется попотеть. Выбрав компанию, которая владеет информацией о нужных курсах, нам нужно будет решить вопрос с физической линией, по которой будут передаваться данные. Далее, необходимо совместно с поставщиком данных оговорить протокол, а также формат данных, по которому они будут передаваться. Кроме того, надо будет написать специальные программы на обеих сторонах, которые бы позволяли обмениваться данными о курсах. Все это, очевидно, весьма и весьма трудоемко, дорого и неудобно.

Изящное и универсальное решение этой задачи дает нам технология Web-служб. Она позволяет нам использовать стандартный протокол HTTP для обмена XML-сообщениями, в формате, определяемом протоколом SOAP. Поставщик данных о курсах акций один-единственный раз развертывает Web-службу, предоставляющую данные о курсах акций, а далее, любые заинтересованные стороны смогут стандартным образом осуществить вызов этой службы.

Поставщик службы может также создать описание своей службы на языке WSDL и опубликовать его в UDDI реестре. После этого все желающие смогут осуществить поиск в реестре, найти нужную им службу и вызвать ее. К сожалению, формат статьи не позволяет нам подробнее обсудить особенности языка WSDL. Отметим только, что с WSDL-описанием Web-службы для нашего примера ознакомиться можно по адресу <http://services.xmethods.net/soap/urn:xmethods-delayed-quotes.wsdl>. В этом файле содержится описание, как интерфейса, так и реализации Web-службы.

## 5. Java-клиент для Stock Quote Web-службы

В настоящем разделе мы приведем пример клиентского приложения, вызывающего Stock Quote Web-службу. Как вы, наверное, уже догадываетесь, служба будет возвращать нам текущее значение курса акций для компании, чье имя мы будем передавать ей в качестве параметра.

Как мы отмечали в предыдущей части статьи, на сегодняшний день существуют две основные платформы для создания распределенных приложений - J2EE (Java2 Enterprise Edition) и Microsoft .Net. Важным элементом каждой из этих платформ является поддержка технологии Web-служб. В данной статье мы проиллюстрируем создание клиента Web-службы для платформы J2EE. Эта платформа является открытой, для нее существуют свободно распространяемые библиотеки и продукты, позволяющие создавать и устанавливать Web-службы без необходимости приобретения дорогостоящих средств разработки, серверов приложений и т.д. На сегодняшний день только эта платформа позволяет создавать распределенные приложения, которые действительно могут надежно, безопасно и устойчиво работать на любых программных и аппаратных платформах.

Приведенный ниже фрагмент клиентского приложения для вызова Web-службы использует API популярной open-source реализации протокола SOAP, которая называется Axis и разработана в рамках проекта некоммерческой организации Apache (<http://xml.apache.org/axis>).

```
public float getQuote(String args[]){
    /*опускаем проверку корректности переданных аргументов */
    symbol = args[0];
    Float res = new Float(0.0F);
    /* создаем новый объект типа «служба» */
    Service service = new Service();
    /* создаем объект для удаленного вызова,
    инициализируем его адресом, на котором
    работает слушатель Axis, именами службы и
    операции, типами параметров,
    а также значением входного параметра */
    Call call = (Call) service.createCall();
    call.setTargetEndpointAddress(
        new URL("http://services.xmethods.com:80/soap"));
    call.setOperationName(new QName("urn:xmethods-delayed-quotes","getQuote"));
    call.addParameter("symbol", XMLType.XSD_STRING, ParameterMode.IN);
    call.setReturnType(XMLType.XSD_FLOAT);
    /* Вызываем удаленную службу */
    Object result = call.invoke(new Object[] { symbol });
    /* Пропускаем проверку возможной исключительной ситуации */
}
```

```
/* Возвращаем успешно полученное значение */  
res = (Float) result;  
return res.floatValue();}
```

Приведенное выше клиентское приложение использует API Axis для того, чтобы создать SOAP-сообщение и передать его по протоколу HTTP на сервер, где оно принимается сервлетом, входящим в серверную часть пакета Axis. Далее Axis разбирает сообщение и вызывает нужную Web-службу.

С полной версией описанного выше класса можно ознакомиться, скачав zip-файл с клиентским приложением для нашего примера по адресу <http://krazum.narod.ru>. В нем содержится все необходимое для того, чтобы запустить клиентское приложение с вашего компьютера без скачивания и установки дополнительных библиотек. В корневом каталоге архива содержится текстовый файл **Readme.txt**, содержащий пошаговые инструкции, необходимые для запуска примера. Для выполнения приложения на компьютере должна быть установлена среда выполнения Java (Java Runtime Environment, JRE).

## 6. О чем мы не рассказали

Хотелось бы отметить, что наша серия статей является всего лишь введением в технологию Web-служб. Мы рассмотрели Web-службы с высоты птичьего полета, не затронув многих связанных с ними серьезных проблем и вопросов. Ниже мы приводим три важных примера таких проблем. Каждая из них заслуживает отдельного разговора, мы упоминаем их здесь только для того, чтобы подчеркнуть насколько широким является спектр вопросов, связанных с Web-службами.

Как известно, распределенная архитектура наряду с массой достоинств (повышенная мощность и надежность системы и т.д.) имеет и определенные недостатки, главными из которых являются трудность интеграции программных компонентов системы и проблема обеспечения безопасности. Web-службы с успехом используются для решения первой из этих проблем, о чем уже было достаточно сказано выше. Что же касается безопасности, то эта проблема вызвана необходимостью передачи данных по сети между различными физическими устройствами, входящими в систему. В процессе этой передачи данные могут быть похищены или подвергнуты искажению заинтересованными злоумышленниками. На сегодняшний день уже разработан

комплекс спецификаций и решений, направленных на предотвращение этих нежелательных явлений. Так, в апреле 2002-го года IBM, Microsoft и Veri Sign выпустили совместную спецификацию WS-Security, определяющую, как расширить протокол SOAP, чтобы обеспечить конфиденциальность, целостность, и аутентификацию при передаче сообщений.

Еще одним важным практическим вопросом является проблема хостинга Web-служб. На сегодняшний день в мире научились создавать Web-службы, и вопрос теперь состоит в том, как их оптимальным образом использовать и продавать. Как мы отмечали, Web-службы предлагают новый взгляд на программное обеспечение, в рамках которого отдельные программные компоненты рассматриваются как службы, доступ к которым может быть получен динамически. Однако доступ этот должен быть, вообще говоря, не бесплатным. Встает вопрос о том, что необходимо поместить Web-службы в некоторую среду, которая позволяла бы всем заинтересованным сторонам находить их, подписываться на их использование (заключать контракты) и осуществлять вызов служб. Среда должна обрабатывать обращения к Web-службам, осуществлять аутентификацию и авторизацию пользователей, и начислять плату за использование служб в соответствии с оговоренными в контракте правилами, а затем выставлять счет пользователям служб. Действия по созданию и настройке описанной выше среды и размещению Web-службы в ее пространстве называют **обеспечением** (provisioning) Web-службы. На сегодняшний момент не существует готовых решений, реализующих все аспекты обеспечения служб. В ближайшем будущем ожидается выход анонсированного IBM продукта Services Provisioning Manager, отчасти решающего эту задачу.

Наконец, нельзя не упомянуть проблемы составления композиций Web-служб. Выше мы описали, как создать отдельно взятую Web-службу, однако, в реальные системы могут содержать не одну, а десятки Web-служб, которые должны вызываться (или вызывать друг друга) в некотором порядке, определяемом бизнес-процессом, для автоматизации которого мы создаем программную систему. Возникает идея построения потока задач (workflow), элементарными компонентами которого будут выступать Web-службы. Эта идея воплощена, например, в предложенном IBM, Microsoft, BEA и SAP языке BPEL4WS (Business Process Execution Language for Web Services). Этот основанный на XML язык предлагает синтаксис для создания описаний бизнес-процессов, реализованных в виде набора взаимодействующих Web-служб.

Данное направление является весьма важным и перспективным, в настоящий момент консорциум W3C ведет работу по созданию единого стандарта, определяющего, как строить композиции из Web-служб.

## **7. Выведение из введения**

Перечень проблем, затронутых в нашем введении, является далеко не полным. Web-службы - это технология, все еще находящаяся на стадии становления. Без сомнения в ближайшие месяцы и годы в этой области грядут большие перемены, появятся новые концепции, спецификации и продукты. Главным в нашей серии статей было донести до читателя основные принципы технологии Web-служб, ибо, как известно, знание некоторых принципов легко возмещает незнание некоторых фактов.

## **8. Ссылки**

1. <http://www-3.ibm.com/software/solutions/webservices/pdf/WSDC.pdf> - Web Services Development Concepts – официальный документ корпорации IBM, предлагающий высокоуровневое описание процесса разработки Web-служб.
2. [http://www.javable.com/javaworld/03\\_02/01/](http://www.javable.com/javaworld/03_02/01/) - перевод статьи из журнала JavaWorld, рассказывающей о противостоянии двух платформ, поддерживающих Web-службы: J2EE и Net.
3. <http://www.javable.com/columns/webserv/> - русскоязычный ресурс, посвященный технологии Web-служб.