

Введение в технологию Web-служб. Часть 2

Константин Разумовский

Эта статья продолжает серию материалов, посвященных технологии Web-служб. В первой части статьи мы ввели понятие распределенной системы и отметили, что одной из главных проблем таких систем является сложность интеграции неоднородных компонентов системы. Как мы выяснили, Web-службы с успехом используются для решения этой проблемы. Обсуждая возможный процесс интеграции на основе использования независимых от платформы, открытых протоколов и стандартов, мы естественным образом пришли к понятию Web-службы. Мы отметили, что одно из необходимых условий, которым должно удовлетворять некоторое приложение, чтобы считаться Web-службой, состоит в том, чтобы уметь принимать запросы и отправлять ответы на них в формате XML. В сегодняшней статье мы попробуем дать более формальное определение Web-службы, а также поговорим о связанных с Web-службами концепциях и технологиях.

1. Определение Web-службы

Прежде всего, необходимо заложить терминологический фундамент для нашего обсуждения. Надо сразу оговориться, что технология Web-служб весьма молода и находится в той стадии развития, когда еще не выработаны общепринятые термины и определения, и имеет место определенная терминологическая путаница.

Итак, существует множество определений Web-службы. Поскольку процесс стандартизации Web-служб начался не так давно, каждая из компаний, разрабатывающих решения для Web-служб, в своих документах вынуждена определять Web-службу по-своему. Существуют определения IBM, Sun, Microsoft, но чтобы не быть обвиненным в пристрастиях к определенным производителям, я приведу определение независимого консорциума W3C (World Wide Web Consortium,

<http://www.w3.org>), который уже около 8 лет успешно занимается разработкой спецификаций для Web-технологий и содержит в качестве своих членов все упомянутые компании. Нижеследующее определение предлагается в рабочем проекте документа "Web Services Architecture" (<http://www.w3.org/TR/ws-arch>) и звучит следующим образом:

Web-служба (Web service) — это программная система, идентифицируемая с помощью URI (Unified Resource Identifier), внешние интерфейсы и связи которой определены и описаны с помощью XML. Другие программные системы должны иметь возможность находить эти описания. Далее эти программные системы могут взаимодействовать с Web-службой в стиле, определенном в ее описании, используя XML-сообщения, передаваемые по некоторому Интернет-протоколу.

Приведенное определение является предельно общим и позволяет называть Web-службами широкий класс различных приложений, даже не имеющих никакого отношения к Web-технологиям (что, казалось бы, противоречит самому названию Web-служб). Как видно из определения, такие приложения должны удовлетворять двум основным условиям:

1. Предоставлять описание своего интерфейса в виде XML.
2. Поддерживать обмен сообщениями в формате XML.

Таким образом, легко видеть, что ключевой технологией для Web-служб является XML. Воистину, XML — основа Web-служб, и подробнее об этом — в следующем разделе.

2. Магия XML

Данный раздел ни в коей мере не претендует ни на обзор, ни на введение в XML. XML — это весьма обширная технология, различные области которой определяются несколькими спецификациями концерна W3C. Описать ее с помощью нескольких тысяч символов (а больше формат статьи мне использовать не позволяет) решительно невозможно. Поэтому мы попробуем просто остановиться на некоторых принципиально важных для Web-служб особенностях этой технологии.

Итак, первый стандарт языка XML (eXtensible Markup Language, расширяемый язык

разметки) был представлен консорциумом W3C в 1998 году. XML — это метаязык, то есть язык для описания других языков разметки, скажем, таких как HTML (Hypertext Markup Language). XML позволяет определить набор правил (элементы, атрибуты, сущности и т.д., а также их порядок), по которым будут создаваться документы, соответствующие новому языку. Для определения правил обычно используются формат DTD (Document Type Definition) или XSD (XML Schema Definition Language). XML-документы имеют иерархическую структуру и предназначены для хранения данных. Данные хранятся в текстовом формате (используется универсальный стандарт кодировки Unicode, включающий символы практически всех алфавитов мира). В древовидную структуру XML можно уложить информацию из хранилища данных с практически любой структурой.

Таким образом, с помощью XML легко можно определять новые форматы данных, являющиеся простыми, переносимыми, независимыми от конкретной операционной системы, а также приложения, эти данные создавшего. В то же время данные XML-документа являются строго структурированными, поддерживают возможности поиска и преобразования для последующего визуального отображения, то есть подходят как для автоматической обработки, так и для восприятия человеком. Ниже представлен пример простейшего XML-документа, упрощенно моделирующего телефонный справочник.

```
<?xml version="1.0"?>
<PHONE_BOOK>
  <PERSON>
    <NAME> K. Razumovsky </NAME>
    <PHONE> 1234567 </PHONE>
  </PERSON>
  <PERSON>
    <NAME> M. Simkin </NAME>
```

```

        <PHONE> 7654321 </PHONE>

    </PERSON>

</PHONE_BOOK>
    
```

Если теперь вернуться к проблеме обмена сообщениями между компонентами распределенной системы, которую мы затронули в первой части статьи, то легко видеть, что перечисленные выше достоинства делают XML-сообщения исключительно удобными для такого обмена. Это может быть как обмен совершенно произвольными данными, так и отправка инструкций для удаленного вызова методов с последующим возвратом результатов. На рис.1 представлена возможная схема такого обмена.

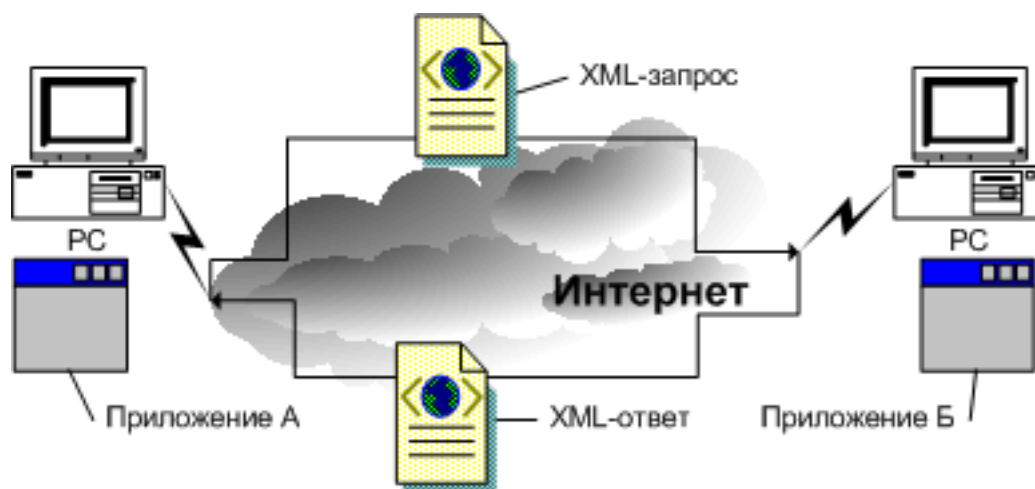


Рисунок 1. Обмен данными в формате XML между приложениями по сети.

3. В мире Web-служб

До сих пор мы уклончиво говорили о Web-службах, как о неких «приложениях, удовлетворяющих определенным условиям». Настала пора поговорить об этих приложениях более конкретно и обсудить принципы их работы.

Вообще говоря, существует много различных моделей для реализации Web-служб. Некоторые из них представлены в табл. 1.

Web-службы	RPC-ориентированные	Ориентированные на документ
Модель взаимодействия	RPC	обмен сообщениями

Модель обработки	направленная на бизнес объекты	направленная на обработку документа
Модель взаимодействия	синхронная	асинхронная

Таблица 1. Модели Web-служб.

Как видно из таблицы, Web-службы могут реализовывать вызов удаленных методов (RPC, remote procedure call), а могут просто принимать произвольные XML-сообщения для некоторой обработки. В первом случае они могут взаимодействовать с некоторым бизнес объектом системы (определяемым полученным запросом), вызывая в свою очередь его методы для выполнения необходимой операции. В то же время они могут просто выполнять заданные функции в зависимости от параметров пришедшего XML-документа, что более характерно для модели обмена сообщениями. При этом клиент Web-службы, отправив запрос, может остановить свою работу и ждать ответа (синхронная модель), а может продолжать работу и принять ответ через некоторое время, когда тот придет от Web-службы (асинхронная модель).

На сегодняшний день существуют две основных платформы для создания Web-служб. Платформа **J2EE** (Java 2 Enterprise Edition) позволяет создавать Web-службы на основе Java-приложений (Java класс, EJB компонент) для любых архитектур и операционных систем. Практически любой класс Java можно объявить как Web-службу. Например, он может быть таким:

```
public class MyWebService {  
  
    public int getSum (int arg1, int arg2) {  
  
        return (arg1+arg2); }  
}
```

Этот простейший класс позволит клиентам Web-службы получить сумму двух целых чисел, переданных в качестве аргументов.

Платформа **Microsoft .Net** позволяет создавать Web-службы на основе таких языков программирования как Visual Basic.Net, C++, C# . Microsoft также заявляет, что ее Web-службы могут работать на любой платформе, и в теории это на самом деле так. На сегодняшний момент, однако, J2EE остается единственной платформой, действительно позволяющей создавать Web-службы для любых операционных систем, так как Microsoft еще не успела создать интерпретатор CLR (common runtime

environment) для выполнения своего промежуточного кода на операционных системах, отличных от Windows. Ниже мы приводим код для простейшей ASP.Net Web-службы на языке C#.

```
<%@ WebService Language="C#" Class="MyWebService" %>

using System;

using System.Web.Services;

[WebService()]

class MyWebService : WebService {

    [WebMethod()]

    public int getSum (int arg1, int arg2) {

        return (arg1+arg2); }}
```

Самое приятное состоит в том, что приложения, написанные для одной платформы, могут вызывать Web-службы, функционирующие на другой. И здесь мы опять можем вспомнить слова вице-президента компании Sun Microsystems Джеймса Гослинга, которые я уже приводил в первой части статьи: «Web-службы обеспечивают однородное представление неоднородной системы».

Подобным же образом Web-службы оказывают неоценимую помощь, когда речь идет об интеграции так называемых унаследованных систем (legacy systems), то есть интеграции старых, созданных до начала эпохи интернет-технологий систем с новыми современными компонентами. Интерфейсы, через которые осуществляется взаимодействие с унаследованными системами, часто весьма сложны и решение задачи интеграции становится чрезвычайно трудоемким, но и отказаться от использования системы и написать все заново слишком дорого. Теперь у проблемы появляется простое решение: необходимо просто «завернуть» старую систему в Web-службу и тогда ее вызов из вновь подключаемых компонентов станет тривиальной задачей!

4. Service-Oriented Architecture

Итак, как мы выяснили, Web-службы позволяют повторно использовать ранее написанный код, что весьма существенно для экономии ресурсов предприятий. Теперь повторно используемые компоненты надо всего лишь обернуть в Web-службы, и их можно будет впоследствии легко вызвать из любой точки сети. Более того, функциональность таких компонент можно будет продавать другим компаниям, благо Web-службы обеспечат стандартный интерфейс для взаимодействия с ними.

Здесь мы затронули серьезный аспект, связанный с применением Web-служб в будущем. Поскольку эта технология легко позволяет сделать свои приложения доступными через Интернет, открываются фантастические возможности для интеграции готового функционала в свои системы. В связи с этим некоторые эксперты говорят о возможности концептуального изменения процесса разработки программного обеспечения в недалеком будущем. «Хватит писать свой собственный код, покупайте уже готовые программы и подключайте их при помощи Web-служб», - заявляют руководители Sun, Microsoft, IBM.

Ввиду легкости интеграции с новыми партнерами бизнес становится более гибким и динамичным. Именно поэтому, как мы рассказывали в первой части статьи, IBM говорит о появившейся возможности динамического ведения электронного бизнеса (dynamic e-business). Архитектура, поддерживающая такую модель, получила название **SOA** (Service-Oriented Architecture - архитектура, ориентированная на службы).

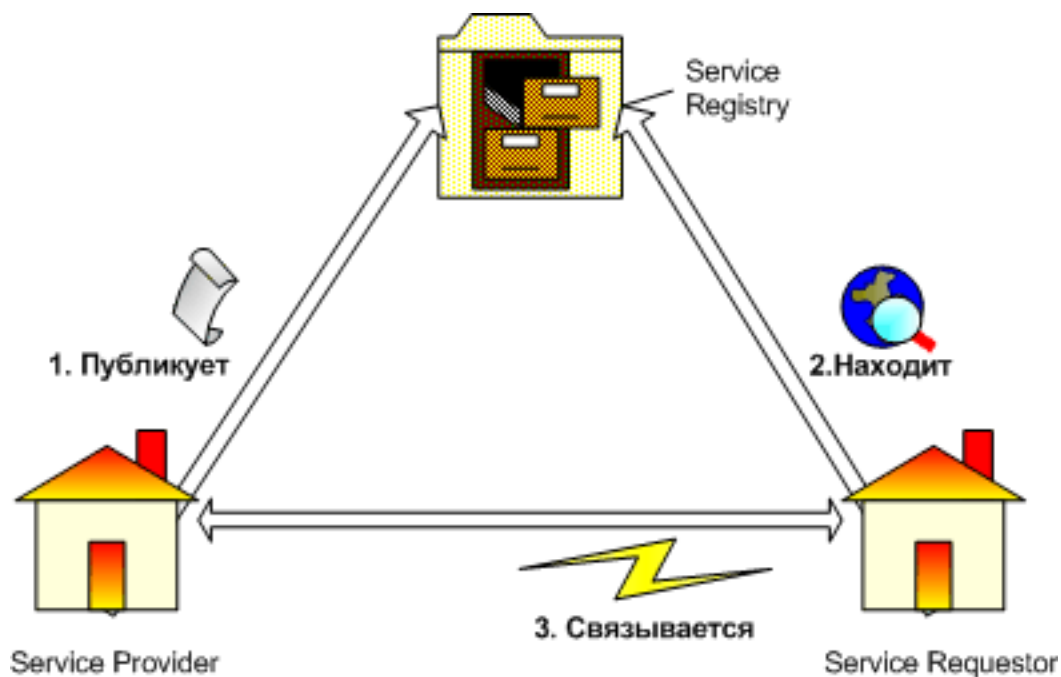


Рисунок 2. SOA: Архитектура, ориентированная на службы.

SOA предлагает представлять программные компоненты в виде служб, которые в любой момент могут быть получены по сети любым нуждающимся в них лицом или организацией.

Как видно на рис.2, SOA предусматривает три основные роли. Под поставщиком службы (**service provider**) можно понимать организацию, которая владеет службой или обеспечивает платформу, на которой работает служба. Потребитель службы (**service requestor**) нуждается в определенных функциях, предоставляемых службой. Реестр служб (**service registry**), выступает в качестве посредника предоставляя каталог с информацией о всевозможных службах, предлагаемых различными поставщиками служб.

На рис.2 представлены и три основные операции, предусмотренные архитектурой SOA. Поставщик службы **публикует** информацию о своих службах в реестре служб, где их **находит** потребитель служб. Используя найденную информацию, он **связывается** с Web-службой (вызывает ее, инициирует взаимодействие с ней).

5. HTTP, SMTP, XML, SOAP, WSDL, UDDI, WSFL, BPEL

Наверняка количество вынесенных в название данного подраздела аббревиатур выглядит несколько устрашающе. Что поделать - если сама концепция Web-служб предельно прозрачна, то ее реализация и, практическое использование требуют знакомства с целым рядом современных технологий и стандартов.

Как мы уже отметили выше, приведенное определение Web-служб не накладывает строгих ограничений на формат описания интерфейса приложения, используемый транспортный протокол и формат обмена сообщениями. Так, например, в качестве транспорта для передачи сообщений по сети мы можем использовать HTTP, SMTP, а также многие другие протоколы.

Однако, при всей имеющейся свободе выбора на практике на сегодняшний день наиболее часто используют следующую модель вызова Web-службы: описанную с помощью языка **WSDL** Web-службу вызывают при помощи передаваемых по протоколу **HTTP** запросов, имеющих формат, определяемый стандартом **SOAP**. При этом сама Web-служба зарегистрирована в **UDDI** реестре. Ниже мы кратко обсудим эти важные технологии, сопутствующие Web-службам.

WSDL (Web Services Definition Language) — это основанный на XML язык, который позволяет описывать различные службы, в терминах допустимых операций, типов аргументов и возвращаемых значений, а также может предоставлять URI для доступа к службе. Ссылки на WSDL-описание службы могут помещаться в реестры, такие, как UDDI.

UDDI (Universal Description Discovery, and Integration) — это спецификация, определяющая, способ, с помощью которого поставщик служб может зарегистрировать себя и свои службы в UDDI реестре, а потребители служб — найти в этом реестре всю необходимую им информации для вызова требуемых служб. Именно UDDI-реестры часто выступают в качестве реестров служб в описанной выше архитектуре SOA.

SOAP (Simple Object Access Protocol) определяет формат XML-сообщений, используемых для обмена информацией между Web-службой и ее клиентами.

Кроме упомянутых выше основных стандартов, существует целый ряд других спецификаций, определяющих те или иные аспекты функционирования Web-служб. Так, например, языки **WSFL** (Web Services Flow Language) и **BPEL** (Business Process Execution Language) определяют правила, по которым можно строить композиции из Web-служб.

6. Жизнь.exe /?

Всякий раз, когда речь идет о новой «революционной» технологии, всегда очень хочется предугадать ее будущее, узнать, как распорядится жизнь ее судьбой: умрет ли она, подобно многим другим, или надолго останется у нас на вооружении.

Неблагодарным делом прогнозирования будущего Web-служб занимаются многие компании и аналитики. В табл. 2 мы приводим вторую часть таблицы, содержащей прогноз внедрения Web-служб, сделанный авторитетной компанией GartnerGroup в июне 2001-го года. Напомним, что первая часть прогноза обсуждалась в первой части статьи.

2004	Процесс адаптации компаниями моделей, основанных на Web-службах и вычислениях, ориентированных на службы вступит в пору юности. Все еще будут преобладать внутренние UDDI реестры. 40% финансовых транзакций будут использовать модели, основанные на Web-службах. 35% онлайн-услуг, предлагаемых правительством, будут предоставляться как Web-службы.
2005	Общие UDDI реестры привлекают все больше внимания по мере того, как активизируется обмен данными между различными компаниями и организациями. Больше внимание завоевывают динамические службы.

Таблица 2. Прогноз внедрения технологии Web-служб.

Как видно из таблицы, процесс внедрения Web-служб — дело не одного года. При этом большое внимание уделяется использованию UDDI реестров — как внутренних, расположенных внутри корпоративной сети, так и общих, доступных для публикации

и поиска через интернет.

Корпорация Microsoft в своих прогнозах неизменно отмечает блестящее будущее Web-служб, а также указывает три основных направления развития этой технологии:

1. Создание высокоуровневых служб для управления безопасностью Web-служб, гарантированной доставки сообщений, поддержки транзакций и т.д.
2. Совершенствование стандартов и спецификаций для Web-служб, таких как WSDL, SOAP и др.
3. Совершенствование средств для разработки решений, основанных на Web-службах.

Первые два направления — темы для большого отдельного разговора. А вот о некоторых решениях для разработки Web-служб мы поговорим в следующий раз подробнее.

7. Резюме

В сегодняшней части статьи мы дали определение Web-службы, поговорили о некоторых принципах работы приложений, которые являются Web-службами, а также кратко познакомились с несколькими важными технологиями поддержки Web-служб.

До сих пор мы уделяли основное внимание теоретическим аспектам технологии, почти не касаясь собственно процесса создания и вызова Web-служб. Устранением этого пробела мы и займемся в следующей, заключительной части нашего введения.