

Введение в технологию Web-служб.

Часть 1

Константин Разумовский

Около двух лет назад в обиход разработчиков программного обеспечения по всему миру постепенно начало входить понятие **Web-службы** (Web service). Поначалу эту технологию робко называли перспективной, затем, уже громче, — модной. На сегодняшний день у аналитиков не вызывает сомнений следующий факт: в ближайшие годы Web-службы станут ключевой технологией для создания распределенных приложений, предлагая независимый от платформы, протокола связи и языка программирования способ интеграции компонентов распределенных программных систем.

1. Введение во введение

Эта статья открывает серию материалов, посвященных технологии Web-служб. Ввиду широкого спектра технологий, спецификаций и программных продуктов, окружающих Web-службы, материал разбит на несколько частей. Цель сегодняшней статьи — кратко обосновать важность и актуальность технологии Web-служб, а также дать интуитивное определение понятию Web-службы. В последующих материалах мы определим Web-службы формально, детальнее познакомимся с технологиями, используемыми при создании Web-служб, затронем некоторые более глубокие вопросы, связанные с этой технологией, а также обсудим, как создать Web-службу своими руками.

2. От e-business — к dynamic e-business

Итак, в нашей серии статей речь пойдет об использовании Web-служб для построения распределенных систем. Здесь и далее под **распределенной системой** мы понимаем

программную систему, компоненты которой функционируют на различных физических устройствах в сети, таких как персональные компьютеры, мэйнфреймы, мобильные телефоны и даже стиральные машины и холодильники. Таким образом, к распределенным системам относится огромное количество существующих программных систем, а с учетом тенденций всеобщей глобализации и интеграции, таких систем будет становиться с каждым годом все больше и больше. Практически любая система уровня предприятия, любое Web-приложение являются распределенными системами. Пример распределенной системы представлен на рис. 1.

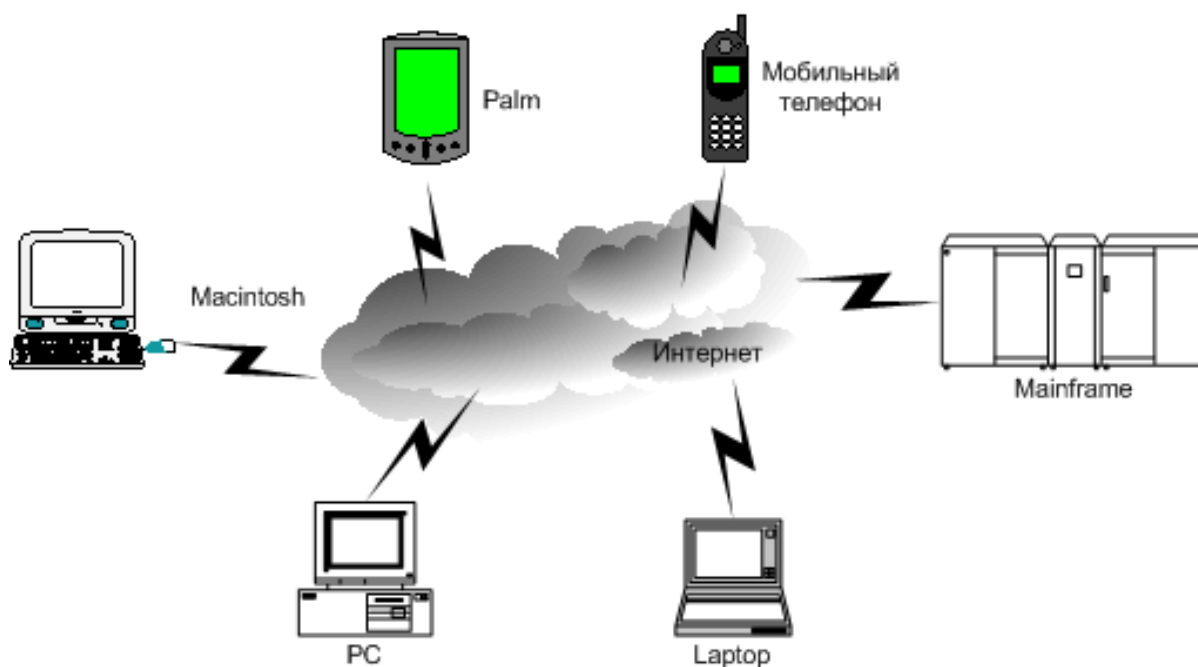


Рисунок 1. Пример распределенной системы.

Повсеместная необходимость в распределенных системах как раз и определяет важность данного направления, а также то огромное внимание, которое уделяют ему крупнейшие производители программного обеспечения. Огромная важность технологии Web-служб — не пустой звук и не рекламный тезис отдела маркетинга отдельно взятой компании. На сегодняшний день все ведущие игроки на рынке разработки программного обеспечения отводят Web-службам важнейшую роль в своих стратегических планах. IBM, Microsoft, Sun, Borland, Oracle, BEA и иже с ними изо всех сил пытаются активно участвовать в процессе продвижения технологии Web-служб, выпуская соответствующие программные продукты для разработки и

внедрения Web-служб, а также проводя мощнейшую рекламную кампанию среди действующих и потенциальных клиентов. Чтобы быть на гребне волны и делать деньги на этой технологии, Microsoft решается на корректировку годами проверенного брэнда, выпуская VisualStudio .Net на смену уже ставшей классической VisualStudio. Sun Microsystems в спешном порядке меняет свою священную корову — платформу J2EE, добавляя в нее спецификации для поддержки Web-служб. Корпорация IBM в конце 90-х годов вместе со многими другими компаниями активно продвигавшая на рынок концепцию **e-business**, подразумевая под этим использование Интернет-технологий для усовершенствования и изменения бизнес-процессов предприятия, с началом нового века провозгласила своей новой стратегией поддержку **dynamic e-business**. Dynamic e-business, согласно IBM, — это качественно новый уровень технологий для e-business, характерной чертой которых является преодоление задач интеграции в распределенных системах с помощью технологии Web-служб.

Одно из самых непредсказуемых и громких противостояний в нынешней программной индустрии — схватка платформ J2EE и .Net, также спровоцировано борьбой сторон за право называться лучшей технологией для разработки и поддержки Web-служб. В этой смертельной схватке отделов маркетинга Sun и Microsoft, последняя заведомо обладает несоизмеримо большими ресурсами, но на стороне Java — более 400 компаний, организаций и частных лиц, совместно развивающих эту открытую платформу, участвуя в процессе JCP (Java Community Process). А значит, невозможно однозначно предсказать результат этого противостояния. Впрочем, этот спор — тема для отдельного разговора. Тем более, что у него есть и другая, неожиданная сторона — как мы покажем далее, именно Web-службы позволяют организовывать удаленные вызовы, совершенно не заботясь о том, на какой платформе работает вызываемый сервис, и позволяя, таким образом, обеим платформам прекрасно сосуществовать даже в пределах одной распределенной системы.

Как вы уже поняли, технология Web-служб весьма молодая, поэтому несмотря на стремительное продвижение на рынок, ее массовое использование — дело недалекого будущего. Приведенная ниже таблица содержит прогноз внедрения Web-служб, сделанный авторитетной компанией Gartner Group в июне 2001-го года.

Год	Состояние технологии
2001	Будут разработаны инструменты для поддержки

	Web-служб. Разработчики начнут приобретать новые средства разработки, ориентированные на Web-службы. Начнется разработка реальных Web-служб.
2002	Начнут появляться Web-службы, потребителями которых являются предприятия и компании. Web-службы, ориентированные на конечных пользователей будут существовать в большом количестве.
2003	Значительно возрастут масштабы использования Web-служб государственными и правительственными организациями.

Таблица 1. Прогноз внедрения технологии Web-служб.

Можно сказать, что пока этот прогноз вполне оправдывается — к настоящему моменту компании начинают все активнее и активнее использовать Web-службы в своих бизнес-процессах. У всех на слуху пример популярной поисковой службы Google, анонсировавшей недавно механизм Web API, который позволяет разработчикам обращаться с запросами более чем к 2 млн. документов в Web, опираясь на концепцию Web-служб. В то же время технология Web-служб еще не является достаточно зрелой (прежде всего, в части безопасности), чтобы быть активно используемой в правительственных и государственных организациях. Отметим, что здесь мы приводим только первую часть прогноза Gartner. Вторую часть таблицы, содержащую прогноз на 2004-й и 2005-й года мы обсудим во второй части нашей статьи, ибо эта часть прогноза затрагивает понятия и технологии, о которых мы будем говорить позже.

3. Сильные стороны слабого связывания

Прежде чем дать определение Web-службы и погрузиться в пучину технических подробностей, давайте с высоты птичьего полета окинем взглядом некоторые важные тенденции отрасли программного обеспечения в последние годы и попробуем понять, чем же была продиктована необходимость появления Web-служб.

Как уже было отмечено выше, компоненты распределенной системы могут работать на совершенно различных физических устройствах. Естественно, различными могут

быть и операционные системы, под управлением которых функционируют эти компоненты, а также языки программирования, на которых они созданы. Здесь и возникает главная проблема распределенной системы: как связать ее разнородные компоненты воедино? Как из .Net приложения на персоналке «достучаться» до большой ЭВМ и вызвать там программу на Коболе, написанную несколько десятилетий назад? Как из EJB-компонента, «проживающего» на Unix-системе, вызвать метод COM-объекта, созданного на удаленном компьютере где-то в глубинах Internet?

Самым очевидным, но и самым недалководидным решением задачи интеграции компонентов является подход, при котором для связи взаимодействующих компонентов вырабатывается некий специальный формат данных, а возможно даже, и протокол связи, и на обеих сторонах создаются программы, умеющие этот формат обрабатывать. При таком способе взаимодействия компонентов системы говорят о **жестком (сильном) связывании** (tight coupling). Подключение новых компонентов к такой системе требует больших финансовых и временных затрат, что, естественно, неприемлемо. Тем не менее, именно таким образом связаны подсистемы большинства современных распределенных программных комплексов.

В противовес такому негибкому дизайну Web-службы предлагают другую архитектуру, между компонентами которой существует так называемая **слабая связность** (loose coupling). Слабая связность предполагает, в частности, что компонентам системы вовсе не обязательно знать, как устроены взаимодействующие с ними подсистемы, а для взаимодействия нет необходимости в определении новых форматов данных и создании специального программного обеспечения.

Принцип слабой связности далеко не нов. Считается, что именно слабая связность позволила Web-технологиям в рекордно короткие сроки стать чрезвычайно популярными. Когда из броузера мы заходим на Web-сайт и инициируем выполнение некой логики на сервере мы совершенно не заботимся, о том, что именно мы вызываем — cgi-скрипты, сервлеты, JSP, ASP или что-то еще. Наш броузер всегда в состоянии корректно обработать ответ сервера — что бы он ни вернул. В свою очередь, серверу абсолютно безразлично, кто, откуда и с помощью какого Web-клиента вызвал его приложения. Web-службы также используют слабо связанную модель взаимодействия, но идут дальше — в качестве ответа от сервера приходит не HTML-документ, а данные

в формате XML, содержащие в себе сериализованные (сохраненные в текстовый формат) объекты.

Очень точно о роли Web-служб высказался один из создателей языка Java, вице-президент компании Sun Microsystems Джеймс Гослинг: «Web-службы обеспечивают однородное представление неоднородной среды». Действительно, как мы еще не раз увидим далее, Web-службы обеспечивают такой способ интеграции распределенных систем, при котором неоднородности системы (различные платформы, операционные системы, языки программирования) незаметны для взаимодействующих компонентов.

4. Ничто не забыто

До сих пор мы с вами занимались тем, что всячески пытались обосновать важность решения задачи интеграции в распределенных системах. У читателя естественным образом может возникнуть вопрос: если эта проблема настолько важна, неужели ее не пытались решить до появления Web-служб, ведь распределенные системы активно обсуждают уже несколько десятков лет? Пытались и неоднократно! Наиболее известными из этих попыток были CORBA и DCOM. Формат данной публикации не позволяет дать хоть сколько-нибудь серьезного представления об этих технологиях, но и не упомянуть о них мы не можем.

CORBA (Common Object Request Broker Architecture) — это технология создания распределенных систем в соответствии со спецификациями, предлагаемыми консорциумом OMG (Object Management Group). Спецификация CORBA 1.1 была представлена в 1991 году и стала первой серьезной технологией построения распределенных систем, предлагающей реальную независимость от инфраструктуры, сетевых протоколов, операционных систем и языков программирования. В основе CORBA лежат три понятия:

- язык описания интерфейсов OMGIDL (Interface Definition Language), с помощью которого описываются интерфейсы всех объектов, участвующих во взаимодействии;
- брокер объектных запросов ORB (Object Request Broker), являющийся посредником, промежуточным звеном системы и скрывающий в себе всю сложную логику взаимодействия клиентского приложения с серверными объектами;

- протокол ИОР (Internet Inter-ORB Protocol), обеспечивающий транспортную поддержку для запросов клиента и ответов сервера, а также взаимодействия между ORB.

К сожалению, на практике вышло так, что ORB от различных производителей оказались не совместимыми (или не полностью совместимыми), что не позволило строить на основе CORBA распределенные системы, действительно независимые от платформы и поставщика программного обеспечения для промежуточного слоя.

DCOM (Distributed Common Object Model)- технология создания распределенных систем, разработанная фирмой Microsoft на основе модели COM. Как известно, COM использует двоичную структуру объектов, что, в некотором смысле, является ее достоинством, позволяя использовать для работы с ней различные языки программирования (C++, VisualBasic, Visual J++ и др.). С другой стороны, такая архитектура определяет и главный недостаток COM/DCOM: каждый компонент распределенного DCOM-приложения должен обязательно выполняться под управлением операционной системы Windows. Надо сказать, что Microsoft неоднократно декларировала, что DCOM будет перенесена на другие платформы, но эти обещания так и не были выполнены.

Следует заметить, что кроме CORBA и DCOM существовали и существуют другие модели для удаленных вызовов в распределенных системах. Однако, все они, обладая теми или иными ограничениями, также не смогли завоевать популярность у разработчиков распределенных систем. Скажем, технология RMI (Remote Method Invocation) поддерживает вызов методов удаленных объектов, но только в том случае, когда и клиентская, и серверная часть являются Java- приложениями. В этом качестве она с успехом и используется, — например, для вызова методов EJB-объектов в распределенных J2EE-приложениях (наряду с протоколом ИОР и некоторыми сервисами CORBA).

5. Интуитивное определение Web-службы

В упомянутых выше технологиях легко можно выделить некоторые общие черты. В частности, каждая из них предлагала некий транспортный протокол для обмена данными между компонентами (ИОР в CORBA, RPC в DCOM), формат для передачи

значений по этому протоколу (CDR, NDR), а также способ адресации серверных объектов (IOR, OBJREF). Относительные неудачи технологий во многом были вызваны сложностью и несовместимостью указанных стандартов и форматов.

И вот, несколько лет назад появилась следующая идея. А что, если для взаимодействия распределенных объектов выбрать всем известные, популярные, и заведомо поддерживаемые всем поставщиками стандарты? Скажем, не выдумывать специального протокола обмена данными, а положиться на какой-нибудь проверенный существующий транспортный протокол (скажем, HTTP)? Далее, можно отказаться и от изобретения формата данных и опереться на популярный, платформенно-независимый формат хранения данных XML (eXtensible Markup Language), а в качестве ссылки на серверный объект в случае применения HTTP естественно использовать URI (Unified Resource Identifier). Полученная архитектура как раз и реализует идею Web-служб.

Подчеркнем, ключевым моментом здесь является использование универсального формата XML. Текстовый XML-документ имеет одинаковое представление на любой платформе, может хранить любые данные и легко передается по сети. Это и позволяет реализовать Web-службу на основе обмена XML сообщениями. Именно поэтому главное требование, которому должно отвечать приложение, чтобы называться Web-службой — это умение принять запрос на выполнение какого-либо функционала в формате XML и вернуть ответ в этом же формате. Приведенная ниже табл.2 содержит сравнение стандартов, используемых CORBA, DCOM и Web- службами.

	Web-службы	CORBA	DCOM
Транспортный протокол	HTTP, SMTP и др.	IIOP	RPC
Формат сообщения	XML	CDR	NDR
Способ адресации	URI	IOR	OBJREF

Таблица 2. Стандарты, используемые в основных технологиях построения распределенных систем.

Полагаю, далеко не все из читателей имеют достаточное представление об упомянутой выше технологии XML. Здесь я в очередной (но не в последний) раз должен сказать, что описывать серьезную технологию в одном-двух газетных абзацах — грандиозная профанация, но замалчивать тему и вовсе невозможно, так как понятие об XML

является принципиальным для понимания Web-служб. Выбирая между Сциллой полного неведения и Харибдой чрезмерно поверхностного обзора проблемы, я все же склоняюсь ко второй, поэтому следующая часть статьи будет начинаться с небольшого введения в XML.

6. Резюме

Итак, в сегодняшней статье мы затронули проблему интеграции компонентов распределенных систем. Как известно, распределенная архитектура наряду с массой достоинств (например, повышенной производительностью и надежностью системы) имеет и определенные недостатки, пожалуй, главным из которых является трудность интеграции программных компонентов неоднородной 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 некоторое приложение может считаться Web-службой, если оно умеет принимать запросы и отправлять ответы в формате XML. Это необходимое, но не достаточное условие принадлежности приложения к семье Web-служб. Остальные требования к Web-службам, а также многое другое мы обсудим в следующей части статьи.

Автор будет чрезвычайно благодарен за любые отзывы о материале.

7. Ссылки

1. <http://www.ibm.com/webservices> - страничка IBM, посвященная технологии Web-служб.
2. <http://www.java.sun.com/webservices> - страничка Sun Microsystems, посвященная технологии Web-служб.
3. <http://www.microsoft.com/webservices> - страничка Microsoft посвященная, технологии Web-служб.
4. Цимбал А. Технология CORBA для профессионалов. — СПб, «Питер», 2001.