УДК 004.4

Н.Н. Глибовец, С.С. Гороховский, И.В. Коваль, А.Н. Корень

Эволюция принципов и средств веб-программирования

Раскрыто понятие развитых интернет-программ, описаны принципы, положенные в основу нового подхода к созданию веб-систем. Проведено сравнение развитых интернет-программ с классическими веб-программами и настольными программами. Выделены преимущества и недостатки развитых интернет-программ.

The work reveals the concept of the advanced internet-programs, describes the principles of a new approach for creation of web-systems. An analysis of the advanced internet-programs with classic web-programs, application programs is performed. As result the advantages and disadvantages of the advanced internet-programs are singled out.

Розкрито поняття розвинених інтернет-програм, описано принципи, покладені в основу нового підходу до створення веб-систем. Проведено порівняння розвинених інтернет-програм з класичними веб-програмами та з настільними прикладними програмами. Виокремлено переваги та недоліки розвинених інтернет-програм.

Введение. В середине 1990-х годов с развитием сети Интернет всеобщее признание получила модель «тонкого клиента». Она уменьшила затраты на разработку и доставку интернет-программ конечному пользователю и расширила диапазон типов программ, которые можно предоставлять через Интернет. Идея модели основана на использовании *HTML* для представления данных и мощных серверов, динамично формирующих и отправляющих страницы веб-браузерам [1].

Сегодня эта модель доказала свою успешность, несмотря на характерные недостатки и ограничения. Особенно это касается пользовательского интерфейса программ, мультимедийного наполнения и общего совершенствования решений. Рост требований пользователей и стремление организаций предоставлять через Интернет наиболее удобные и качественные программы привел к появлению развитых интернет-программ (РИП), ставших новым шагом в их эволюции. Они отражают постепенный, однако неизбежный переход веб-программ от простой модели тонкого клиента к модели распределенных функций. Здесь РИП напоминают настольные прикладные программы в клиент-серверной архитектуре.

Далее проведено сравнение РИП с традиционными веб-программами и настольными прикладными программами с целью выявления их преимуществ и недостатков, приведен обзор средств и технологий реализации развитых интернет-программ, а именно: *Flex*, *Flash*, *Apollo*, *Ajax*, *JavaFX*, *OpenLaszlo*, *Windows Presentation Foundation*.

Развитие принципов и средств построения веб-систем

Прошло уже 20 лет со времени создания первого браузера [2]. Веб-системы – распределенные и реализуют клиент-серверную архитектуру. Эволюция принципов построения веб-систем заключается в развитии клиент-серверной архитектуры, моделях клиента и сервера.

Модель тонкого клиента. В середине 1990-х годов превалировала модель «тонкого клиента», когда вся обработка данных происходит на сервере, а клиентскую часть используют только для отображения статического контента. Модель взаимодействия унаследована от первоначального назначения веб как среды гипертекста. Модель «тонкого клиента» удачна с технической точки зрения, однако она неудобна для пользователя многих приложений. Пока сервер выполняет свою работу, пользователь ожидает. Взаимодействие с программой каждый раз происходит через сервер, а это требует отправки данных на сервер, ответа от сервера и перегрузки страницы на клиентской стороне.

Модель толстого клиента. Указанные недостатки и стремление использовать веб для предоставления пользователям мощных систем

привели к модели «толстого клиента». Такой клиент, кроме взаимодействия с сервером, способен самостоятельно выполнять многие функции. Значительная (но не вся) часть функциональности переносится с сервера на сторону клиента.

Модель предполагает инсталляцию на клиенте специального программного обеспечения. В контексте веб-систем клиентское программное обеспечение «толстого клиента» часто действует как расширение браузера. Оно берет на себя ответственность за визуализацию клиентского интерфейса системы и взаимодействие с сервером, а также служит средой выполнения клиентской части веб-системы.

Перенос части функциональности веб-системы на сторону клиента обеспечивает пользователю взаимодействие с системой удобным для него способом. Система не обращается к серверу всякий раз, когда пользователь выполняет действие. Когда клиентская часть способна выполнить операцию самостоятельно, реакция системы на действие пользователя будет почти мгновенной, как у настольных прикладных программ, без перегрузки интерфейса.

Клиент с сервером асинхронно взаимодействуют в фоновом режиме, т.е. клиентская часть отсылает запросы к серверу и, не прекращая работу, ожидает ответ. Таким образом, пользователь может продолжать взаимодействие с программой, а когда результат запроса к серверу поступит в клиентскую часть, она выполнит нужные операции.

Сервисно-ориентированная архитектура. Роль сервера в веб-системах также изменилась. Для традиционных веб-программ сервер выполнял роль ответственного за бизнес-логику программы и управление данными. Он контролировал выполнение каждой мельчайшей операции. С возникновением таких понятий, как сервисно-ориентированная архитектура (*SOA*) и веб-сервисы (*Web Services*), сервер можно рассматривать как поставщика сервисов. Важно, что поставщик сервиса может быть и потребителем.

Следует различать термины *сервисы* и *веб-сервисы*. Первый обозначает технологии, используемые для установления соединений, и то, что соединяется с использованием веб-сервисов. Сервисы – это конечные точки соединений. Сочетание сервисов формирует сервисно-ориентированную архитектуру.

Технологии, положенные в основу веб-сервисов. Рассмотрим такие технологии, а именно: *WSDL*, *UDDI* и *SOAP*.

Язык описания веб-сервисов *WSDL* (*Web Services Description Language*) составляет основу веб-сервисов. Рис. 1 иллюстрирует использование *WSDL*. Поставка сервиса и его использования описаны следующими шагами:

1. Поставщик описывает сервис на языке *WSDL* и размещает его в директории сервисов, которая может использовать *UDDI* (*Universal Description*, *Discovery*, *Integration*). Другие формы директорий также можно использовать.

2. Потребитель сервиса посылает один или более запросов к директории, чтобы получить адрес сервиса и определить, как использовать этот сервис.

3. Часть *WSDL*-описания, сформированного поставщиком сервиса, передается потребителю. Это дает информацию потребителю о том, какого формата должны быть запросы к поставщику сервиса и ответы от него.

4. Потребитель сервиса использует *WSDL* для отправки запроса к поставщику сервиса.

5. Поставщик сервиса предоставляет ответ определенного формата потребителю.

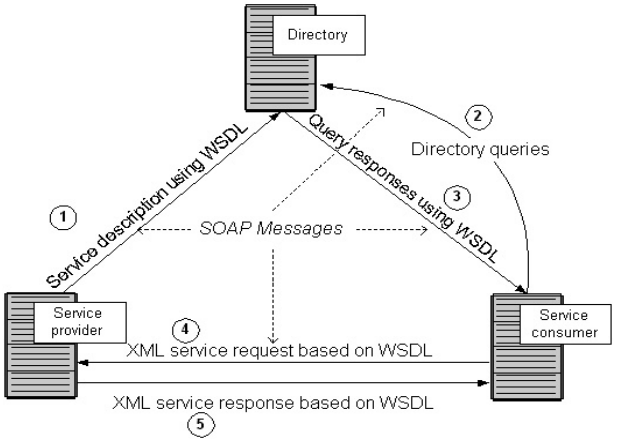

Рис. 1. Использование *WSDL* для механизма работы веб-сервисов

UDDI – протокол для публикации и нахождения метаданных о сервисах. Директорию мож-

но реализовать как *UDDI*-реестр, предназначенный для обеспечения возможности находить сервисы, описанные на языке *WSDL* как на этапе разработки программ, так и во время их исполнения. Альтернатива *UDDI – ebXML Registry*.

Все сообщения, которыми обмениваются поставщик с потребителем, направляются посредством *SOAP* как формата конвертов для сообщений (может расширяться, базируется на *XML*). Акроним *SOAP* происходит от *Simple Object Access Protocol*, однако со временем связь с таким конкретным значением букв утратилась. *SOAP* обычно использует протоколы *HTTP* и *HTTPS*, однако можно использовать и другие, например *SMTP* и *XMPP*. Альтернативы *SOAP – JSON-RPC*, *XINS*, *Burlap*, *GXA*, *Hessian Web Service Protocol*, *REST*, *XML-RPC*, *BEEP*.

С развитием сервисно-ориентированной архитектуры возникло понятие сервисно-ориентированного клиента (*Service-Oriented Client*, *SOC*), сфокусированного на сервисах, а не на структуре классов объектов, реализующих эти сервисы.

Эволюция средств построения веб-систем

Средства для построения веб-систем можно разделить на три группы технологий: клиентской стороны, серверной стороны и обмена данными[1]. Далее приведен краткий обзор наиболее популярных представителей каждой из групп.

Технологии клиентской стороны. Для создания первых веб-страниц использовали *HTML* (*HyperText Markup Language*). Первое общедоступное описание *HTML* включало в себя информацию о 22 тегах. Тринадцать из них до сих пор существуют в спецификации *HTML* 4 [3].

На смену *HTML* пришел новый язык разметки веб-страниц *XHTML* (*eXtensible HyperText Markup Language*), главное отличие которого от *HTML* состоит в строгих правилах описания разметки страницы. С 2001 года *XHTML* 1.1 стал рекомендацией *W3C* (*World Wide Web Consortium*) [4], однако не смог полностью вытеснить *HTML*.

Язык описания таблиц стилей *CSS* (*Cascading Style Sheets*) используют для оформления документа, написанного на языке разметки. Чаще каскадные таблицы стилей применяют для оформления веб-страниц, написанных на *HTML* или *XHTML*, хотя их можно применять к любому типу *XML*-документов, включая *SVG* или *MXML*.

CSS классифицируется по уровням и профилям. Каждый уровень *CSS* строится на основе предыдущего и, как правило, добавляет новые особенности в таблицы стилей. Уровни в *CSS* обозначаются *CSS*1, *CSS*2 и *CSS*3. Под профилями чаще всего понимают наборы из одного или нескольких уровней *CSS*, создаваемых специально для конкретного устройства или интерфейса. Так, профили *CSS* существуют для мобильных устройств или принтеров.

*CSS*1 и *CSS*2 утверждены как стандарты *W3C*, а работа над *CSS*3 продолжается [5].

Как скриптовый язык *JavaScript* используется в браузерах и довольно слабо связан с языком программирования *Java*, а его синтаксис напоминает язык программирования Си. Одна из важнейших и самых полезных особенностей *JavaScript* – работа с объектной моделью документа (*Document Object Model*, *DOM*) некоторой веб-страницы и выполнение операций, недоступных для *HTML* [6].

Ajax (*Asynchronous JavaScript and XML*) – это технология создания интерактивных интернет-программ. Основная идея технологии заключается в передаче ответственности за обмен небольшими объемами данных с сервером на клиентскую часть (на собственно веб-страницу) так, чтобы не было необходимости перегружать страницу, когда пользователю нужна такая смена. Это повышает интерактивность, скорость и удобство использования страницы пользователем. *Ajax* использует: *XHTML* (или *HTML*) и *CSS* для форматирования и разметки информации; *DOM*, к которой получают доступ с помощью *JavaScript* для динамической работы и изменения информации на странице; объекты *XMLHttpRequest* или *IFrame* для асинхронного обмена данными с веб-сервером; *XML* или любой другой формат как формат передачи данных между клиентом и сервером [7].

[1] Некоторые технологии объединяют средства реализации как клиентской, так и серверной сторон, например *Java*, *Flex*. Поэтому можно сказать, что классификация довольно условна.

Flash стал первым представителем нового подхода к созданию интерактивных интернет-программ. Технология *Flash* появилась в 1997 году в компании *Macromedia*, а с 2005 года продолжает развитие как продукт компании *Adobe Systems*. Интернет-программа, созданная с помощью *Flash*, – это файл скомпилированного байткода, выполняемого в собственной среде *Flash Player*. На основе технологии *Flash* и языка программирования *ActionScript* развилась кардинально новая технология – *Adobe Flex* [8].

Технологии серверной стороны. К технологиям сервера обычно относят серверные языки и каркасы программирования. Существует множество языков, каркасов и технологий программирования серверных сценариев: *PHP*, *Perl*, *Python*, *Java*, *JSP*, *ASP*, *ASP.NET*, *Ruby* и др.

Язык программирования *PHP* (*Hypertext Preprocessor*) разработан для создания динамических веб-страниц. *PHP* обычно выполняется в среде веб-сервера, получая на входе *PHP*-код и выдавая веб-страницы на выходе. Интерпретатор языка можно установить на большинстве веб-серверов, операционных системах или платформах бесплатно, что обусловило его популярность среди разработчиков веб-страниц.

Каркас *ASP* (*Active Server Pages*) для написания серверных скриптов от *Microsoft* используют для создания динамических веб-страниц и позиционируют как дополнение к *IIS* (*Internet Information Services*). Создание веб-страниц на *ASP* достаточно удобно благодаря большому количеству встроенных объектов. Каждый объект соответствует определенной группе функциональности, полезной для создания динамических веб-страниц. Большинство *ASP*-страниц написаны на *VBScript*, но для их написания можно использовать любой язык активного (*ActiveX*) создания скриптов: *JScript* или *ActivePerl*, например. Последняя версия *ASP* 3.0 появилась в 2000 году (с *IIS* 5.0) и развилась в технологию *ASP.NET*.

Каркас *ASP.NET* для создания интернет-программ активно продвигает компания *Microsoft*. Его можно использовать для построения динамических веб-страниц, интернет-программ и веб-сервисов. *ASP.NET* представляет собой неотъемлемую часть платформы *.NET*, есть потомком технологии *ASP*. *ASP.NET* и базируется на *CLR* (*Common Language Runtime*) от *Microsoft*. В 2007 году *Microsoft* выпустила *ASP.NET AJAX* 1.0 как расширение *ASP.NET* с поддержкой технологии *Ajax* [9].

Объектно-ориентированный язык *Java* разработала *Sun Microsystems* в начале 1990-х годов. *Java*-программы обычно компилируются в байткод, а затем интерпретируются виртуальной машиной *Java* (*Java Virtual Machine*, *JVM*), что и обусловливает платформенную независимость этого языка. *Java* довольно активно используется для написания таких серверных программ, как веб-сервисы, сервлеты и *EJB* (*Enterprise Java Beans*).

Java-технологию *JSP* используют для динамической генерации *HTML*, *XML* или документов других типов в ответ на запрос клиента. Эта технология позволяет *Java*-коду и некоторым предопределенным действиям добавляться к статическому контенту. *JSP*-код компилируется в *Java*-сервлеты.

Технологии обмена данными

Стоит выделить две технологии обмена данными между клиентом и сервером: *CSV* и *XML*.

CSV (*Comma-Separated Values*) – формат для хранения и передачи набора данных. Вид данных в этом формате – это значение определенных переменных в определенной последовательности с некоторым разделителем между ними (вначале для этого использовали знак запятой, теперь можно применить любой другой разделитель). Формат достаточно прост и его использование можно считать устаревшим в пользу *XML*, несмотря на то, что он дает определенный выигрыш в скорости пересылки и распаковки данных.

XML – язык разметки общего назначения. Основная цель разметки – улучшение обмена данными между различными информационными системами, особенно в Интернете. *XML* представляет собой рекомендацию и открытый стандарт *W3C*. *XML* базируется на теговом описании объектов, их свойств и составных частей. Для обработки *XML*-формата разработаны как отдельные языки обработки (*XSL*, *XSLT*),

так и стандартные средства (классы–парсеры *XML*) в большинстве языков программирования. Можно сказать, что *XML* – это не только де-юре, но и де-факто стандарт в современном программировании.

Развитые интернет-программы

Термин *развитая интернет-программа* (*Rich Internet Application*) был предложен компанией *Macromedia* (теперь часть компании *Adobe Systems*) в 2002 году. Однако, как чаще всего бывает в ИТ, термин возник для обозначения концепции, начавшей формироваться как название нового вида Интернет-приложений, которые разрабатываются авангардом сообщества веб-разработчиков. Кроме термина *развитая интернет-программа* возникали и другие термины для обозначения таких программ, в частности: отдаленные сценарии (*Remote Scripting*), *X Internet*, развитые клиенты (*Rich clients*), развитые веб-приложения (*Rich web applications*).

Однако термин *развитая интернет-программа* (РИП) наиболее распространен. РИП – это интернет-программы, использующие преимущества толстого (интерактивного) клиента для обеспечения более интуитивно понятного, быстрого и эффективного взаимодействия с пользователем [10].

РИП сочетают лучшие свойства настольных прикладных программ, традиционных веб-программ и интерактивной мультимедийной связи. От настольных прикладных программ они заимствовали мощную функциональность интерфейса, интерактивный интерфейс пользователя, быструю реакцию интерфейса в сравнении с перегрузкой страницы, привычное поведение клиентского интерфейса (например, *drag-and-drop*), возможность работать *online* и *offline*. В РИП сохранились такие качества традиционных веб-программ, как доступность и низкие затраты на установку, скорость установки, кросс-платформенность, использование прогрессивной загрузки для получения контента и данных, существенную поддержку принятых стандартов. И вдобавок, РИП свойственно использование интерактивного аудио и видео.

РИП как новый шаг в эволюции веб-программ. Для всех развитых интернет-программ свойственно наличие промежуточного уровня кода между пользователем и сервером – *клиентское ядро* (*client engine*). Оно действует как расширение браузера и обычно принимает на себя ответственность за визуализацию клиентского интерфейса программы и взаимодействие с сервером.

Вторая отличительная черта РИП – асинхронное взаимодействие с сервером, т.е. отсылка запросов к серверу и без прекращения работы ожидание ответа.

Пример асинхронного взаимодействия – *предварительная выборка* (*prefetching*), в которой программа предусматривает дальнейшую потребность в определенных данных и загружает их до того, как пользователь сделает запрос на них. Таким образом ускоряется ответ на запрос пользователя. Технология предварительной выборки используется в *Google Maps* для загрузки соседних фрагментов карты до того, как пользователь пожелает просмотреть их.

Использование клиентского ядра в развитых интернет-программах обеспечивает ряд преимуществ РИП а сравнении с традиционными веб-программами.

РИП имеют более развитую функциональность, которая может быть настолько богатой, насколько это позволяет технология создания клиентской части. Можно реализовать *drag and drop*; использовать ползунки для фильтрации, изменять и сортировать данные, проводить расчеты на клиентской стороне, что не требует пересылки данных на сервер и ожидания ответа от него.

В РИП интерфейс более интерактивен в сравнении с традиционными веб-программами. Время ответа программы пользователю в среднем меньше, поскольку программе для выполнения определенной операции не всегда следует обращаться к серверу.

РИП обеспечивают более сбалансированную нагрузку между клиентом и сервером. Потребность в вычислительных ресурсах клиента и сервера более сбалансирована. Это обеспечивает освобождение ресурсов сервера, что дает возможность серверному аппаратному обеспечению одновременно поддерживать большее количество клиентских соединений.

РИП имеют возможность асинхронного взаимодействия с сервером. Клиентское ядро может взаимодействовать с сервером асинхронно, т.е. отправлять запросы к серверу и продолжать работу, а когда ответ получен, выполнять необходимые действия.

В РИП сетевой трафик значительно уменьшается благодаря тому, что клиентское ядро более интеллектуально в сравнении со стандартным веб-браузером при решении, какими именно данными следует обмениваться с сервером. Это ускоряет отдельные запросы или ответы благодаря меньшим объемам данных пересылки для каждого взаимодействия, и в целом уменьшить нагрузку на сеть. Однако использование технологии асинхронной предварительной выборки может и нейтрализовать это преимущество или даже привести к росту сетевого трафика и обратить это в недостаток РИП. Технология предварительной выборки не может предопределить каждое последующее действие пользователя, поэтому всегда происходит загрузка избыточных данных.

Сравнение РИП с настольными программами. РИП не требуют инсталляции. Распространение и доступ к таким программам – это мгновенный и автоматизированный процесс. Изменения и обновления до новых версий на клиенте автоматические. Пользователь может использовать программу с любого компьютера, подключенного к Интернет; как правило, не существенно, какая операционная система установлена на компьютере. РИП обеспечивают упрощенный доступ географически распределенных пользователей к общим данным и более устойчивы к инфицированию вирусами в сравнении с обычными настольными программами.

С популяризацией использования веб-пользователи все реже принимают решение в пользу установки программ, если существует веб-аналог и необходимость в доступе к программе с разных компьютеров. Это наблюдается даже тогда, когда интернет-программа медленнее и менее функциональна, чем настольная. Хорошим примером такой программы может быть интернет-почта (*Web-based email*).

Недостатки и ограничения РИП. Поскольку эти программы выполняются внутри «песочницы (*Sandbox*), они имеют ограниченный доступ к системным ресурсам. Если предположение о доступе к ресурсам оказывается ложным, РИП могут перестать работать корректно.

Запрет выполнения скриптов распространяется на конкретные РИП, использующие *JavaScript* или другой язык скриптов. Если пользователь заблокирует выполнение скриптов в браузере, РИП будет работать некорректно или вообще не будет работать.

Скорость *обработки клиентом* касается тех РИП, которые используют для программирования клиентской части интерпретированный язык программирования, например *JavaScript*, что влечет потерю производительности. Данный недостаток отсутствует в РИП с скомпилированным клиентским языком программирования, например *Java* или в РИП с использованием *Flash*- или *Flex*-технологий, в которых операции выполняются родным для *Flash Player* кодом.

Хотя РИП и не требуется инсталлировать, функциональность клиентской части РИП обязан загружать клиент. Поскольку большей частью она автоматически кэшируется, функциональность клиентской части необходимо передать хотя бы один раз. В зависимости от размера и типа доставки время загрузки клиентской части может быть довольно долгим. Разработчики РИП имеют в своем арсенале средства для сокращения времени загрузки, например сжатие клиентской части или разбиение доставки на этапы через несколько страниц программы.

Текстовый контент развитых интернет-программ может быть недоступен для индексирования поисковыми машинами. Поэтому для РИП возникает проблема *потери прозрачности для поисковых машин.*

Заключение. Развитые интернет-программы – это базовый подход к созданию веб-систем, который реализует модель клиент-серверной архитектуры с толстым клиентом, использует преимущества асинхронного режима взаимодействия с сервером, поддерживает развитие сервисно-ориентированной архитектуры и сервисно-ориентированного клиента.

Окончание на стр. 76

Клиентская часть РИП дает больше возможностей, чем просто отображение страниц. Согласно критерию технологичности разработки, РИП, в сравнении с традиционными веб-программами, имеют следующие особенности: они более качественны и более сложны технологически; обеспечивают предоставление более совершенных решений (мощная функциональность, масштабируемость и пр.); времени на разработку РИП затрачивается больше, чем на разработку традиционных веб-приложений.

По критерию персонификации РИП более эффективны во взаимодействии с пользователем, более интерактивны, быстродействующи и интуитивно понятны. Конечно, РИП имеют недостатки. Однако не вызывает сомнений целесообразность перехода от традиционных Интернет-программ к РИП.

1. *Перевозчикова О.Л.* Основи системного аналізу об'єктів і процесів комп'ютеризації. – К.: Видав. дім «КМ Академія», 2003. – 432 с.
2. *The WorldWideWeb* browser. World Wide Web Consortium. – http:// www.w3.org/People/Berners-Lee/WorldWideWeb.html
3. *Index* of elements in HTML 4 // World Wide Web Consortium. – 24.12.1999. – http://www.w3.org/TR/1999/REC-html401-19991224/index/elements
4. *XHTML*. Wikipedia, the free encyclopedia // Wikimedia Foundation, Inc. – http://en.wikipedia.org/wiki/XHTML
5. *Cascading* Style Sheets // World Wide Web Consortium. – 25.04.2007. – http://www.w3.org/Style/CSS/
6. *JavaScript*. Wikipedia, the free encyclopedia // Wikimedia Foundation, Inc. – http://en.wikipedia.org/wiki/Java Script
7. *Garrett J.J.* Ajax: A New Approach to Web Applications. – 18.02.2005. – http://www.adaptivepath. com/publications/essays/archives /000385.php
8. Build engaging, cross-platform rich Internet applications. – http:// www.adobe.com/products/flex/
9. *ASP.NET*. Wikipedia, the free encyclopedia // Wikimedia Foundation, Inc. – http://en.wikipedia.org/wiki/ASP.NET
10. *Jeremy Allaire*. Macromedia Flash MX – A next-generation rich client // Macromedia white paper. – March 2002. – 14 p. – http://download.macromedia.com/pub/ flash/whitepapers/richclient.pdf

Поступила 05.04.2011
Тел. для справок: (044) 463-6985 (Киев)
E-mail: *glib@ukma.kiev.ua*, *gor@ukma.kiev.ua*, *ankoren@gmail.com*