Кондратьев Павел Сергеевич

УлГТУ ФИСТ, ИВТАСмд-11

WEB-ТЕХНОЛОГИИ КАК ПЛАТФОРМА ДЛЯ РЕАЛИЗАЦИИ РАСПРЕДЕЛЁННЫХ СИСТЕМ

Мощность интернет-технологий позволяет создавать на их основе полноценные информационные системы, обладающие рядом очень выгодных характеристик. Одна из причин использования вебсистем как технологии состоит в возможности использования распределенных сред как с точки зрения серверной составляющей, так и с точки зрения пользователя.

При этом распределенность приложений можно понимать, как два феномена:

* распределенные пользователи системы;
* распределенная система (серверная часть системы).

В веб-приложениях возможны оба сценария работы. Первым свойством обладают все веб-приложения априори, а вот второй вид архитектуры используется не всегда и накладывает дополнительные требования на программно-технические средства системы.

Изучим два смысла распределенности систем для понимания условий их применения и основных возможностях.

Во-первых, нужно сказать, когда в использовании системы возникает потребность в поддержке распределенной работы пользователей. Для ответа на этот вопрос нужно проанализировать современные тенденции бизнеса.

В случае с публичной системой пользователи системы могут быть распределены географически в силу естественных причин. Основные потребители таких услуг уже привыкли, что они в том или ином виде доступны через Интернет (например, различные справочные сервисы).

В случае с системой для внутреннего применения компании ее пользователи также могут быть распределены, но по другим причинам. Сегодня особую актуальность приобретают концепции «мобильных работников» и «домашнего офиса», которые предполагают удаленную работу с основными корпоративными системами, т.е. приложения класса MRP/ERP должны поддерживать веб-интерфейс. Очевидно, что имеет смысл подумать о бизнес-приложениях, изначально поддерживающих веб-интерфейс. В данное время идет массированная миграция традиционных приложений и систем на вебинтерфейсы. Это может быть в виде дополнительных компонентов или в виде полной переработки приложения.

Поскольку сценарий работы с распределенными пользователями поддерживается любой системой с веб-интерфейсом, изучим архитектуру таких приложений на абстрактном примере с использованием распространенных программных средств и посмотрим, какую роль исполняет каждый компонент.

Итак, в работе веб-приложения участвуют две стороны: сервер и клиент. Основная нагрузка ложится на сервер. С клиентской стороны работает только веб-браузер, отображающий интерфейс, сформированный серверной частью. На рис.1 представлена полная схема взаимодействия компонентов рассматриваемой информационной системы.

На правой части рисунка расположены компоненты клиентской части приложения. Как видно, никаких специальных компонентов не используется: наличие веб-браузера и офисного пакета для современного ПК давно стало нормой. Можно заявлять, что клиентский ПК может быть любой системой с графическим интерфейсом и относительно современными версиями ПО. На самом деле приложение может быть не совсем совместимо со всеми платформами (операционными системами и браузерами, об этом подробнее дальше).

В элементе «веб-браузер» в зависимости от особенностей приложения могут требоваться дополнительные компоненты. В частности, это расширения функциональности веб-браузера в виде дополнений {plugins): виртуальная машина Java, воспроизведение Flash, AIR, Sil- verlight и другие компоненты. Эти дополнения могут быть как широкораспространенными, так и специфическими для отдельного вебприложения (например, специальные элементы управления ActiveX). Требования к наличию дополнений к браузеру определяются средой разработки приложения и типом веб-интерфейса.

С серверной частью все сложнее. Серверная часть в целом может называться «сервером приложений», точнее «сервером веб-прило- жений». Как правило, в таком сервере объединяются все необходимые компоненты для обеспечения работы веб-приложения. Кроме того, все компоненты могут быть размещены в рамках одного физического или виртуального сервера. Для решения задачи масштабирования (крупное приложение с высокой нагрузкой) размещение компонентов можно организовать раздельно, включая сложные архитектуры высокой доступности и производительности (кластеры). Кластерные архитектуры востребованы в сфере высоконагруженных проектов (highload), к которым относятся популярные порталы, поисковые системы, публичные почтовые сервисы, другие сервис-приложения, предоставляемые большому количеству пользователей. За счет кластеров одновременно решаются две задачи: высокая способность к масштабированию нагрузки за счет возможности расширения (в том числе без останова системы) и такая же надежность работы (доступность), которая реализуется массовым дублированием оборудования. Вопросы высокой доступности в сфере веб-техноло- гий стоят достаточно жестко, так как в среде Интернета пользователи привыкли к круглосуточному обслуживанию своих запросов. Следовательно, приложение должно функционировать постоянно, с минимальными простоями. Для глобальных систем поддержка высокого уровня доступности становится проблемой, так как нет «ночи» из-за разницы часовых поясов у пользователей.

Например, обновления ПО и ремонт частей аппаратного обеспечения не должны вызывать простоев в системе (так как «ночи» нет, то это нельзя сделать «пока никто не заметит»). Чтобы реализовать системы с высоким уровнем доступности, разработано множество технических и организационных решений, обсуждение которых выходит за рамки данной книги.

Возвращаясь к программной части сервера, отметим, что она состоит из трех крупных компонентов:

1. веб-сервера;
2. сервера БД;
3. программного интерпретатора и непосредственно программного кода веб-приложения.

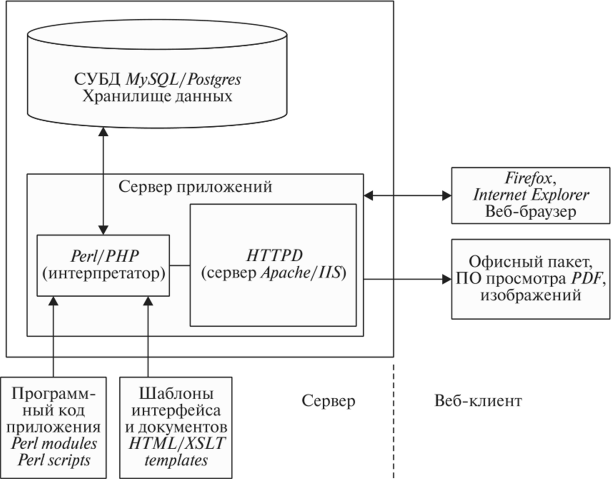


Рисунок 1. Схема взаимодействия компонентов ИС с веб-интерфейсом

Веб-сервер является связующим звеном между веб-приложением и клиентом. Он принимает запросы от клиента в виде набора данных по протоколу HTTP и возвращает в ответ документы на языке HTML и файлы, которые требуются дополнительно (картинки, листы стилей, другие объекты). При этом документы и другие файлы могут быть как статическими (находиться в файловой системе сервера), так и динамическими, т.е. сформированными программной частью. Наиболее распространенным веб-сервером является Apache. Кроме того, существуют разработки специальных веб-серверов для различных приложений, входящих в большие комплексы, например, при использовании веб-приложений на технологиях компании Microsoft логично использовать Internet Information Server (IIS). Некоторые производители информационных систем используют Apache в неявном виде под другими названиями (например, Oracle web-server). От выбора этого компонента зависит набор возможностей по языкам программирования, применяемым для разработки информационной системы.

В рамках реализации сложных проектов с высокой нагрузкой вебсервер делится на две части: «легкий» (front-end) и «тяжелый» (backend). Разделение основано на функциональности. Как было указано выше, веб-сервер может выполнять как простейшие запросы по получению обычного файла с диска и передачи его клиенту, так и сложные запросы к веб-приложению с использованием дополнительных модулей-интерпретаторов или передавать их внешним приложениям для выполнения. Идея состоит в том, что для выполнения простых запросов будет использоваться простой веб-сервер («легкий»), для сложных соответственно «тяжелый». Преимущество такого подхода – в экономном расходовании аппаратных ресурсов сервера (оперативная память, процессорное время, иногда операции ввода-вывода).

Программный интерпретатор – это сердце системы. С помощью программного интерпретатора выполняется код, содержащий логику работы системы (алгоритмы). Программы, работающие в этом интерпретаторе, формируют те динамические элементы, о которых шла речь выше. Поясним, почему этот элемент называется именно так. В веб-среде могут выполняться приложения, написанные на различных языках программирования. Их можно разделить на две группы: компилируемые (исполняемые как обычные программы на компьютере) и интерпретируемые (требующие интерпретатора для выполнения).

При разработке веб-приложений используются оба этих типа, но особую популярность заслужили именно интерпретируемые языки: Perl, РНР, Ruby, Python. Интерпретируемые программы медленнее выполняются, но процесс их разработки значительно проще и быстрее. Учитывая постоянно повышающиеся требования бизнеса к скорости внедрения изменений в программных продуктах (информационных системах), быстрый цикл разработки программ становится решающим фактором. Кроме того, мощность компьютеров постоянно возрастает, позволяя до определенного предела забывать о производительности приложений в угоду удобству и скорости разработки.

В приведенном примере используется интерпретируемый язык Perl как наиболее мощный и подходящий для большинства проектов по созданию информационных систем с веб-интерфейсом. Программный код системы состоит из модулей и скриптов. Эти два компонента реализуют логику выполнения и основную функциональность системы. Для формирования динамических документов (отчетов, таблиц, представлений, текстовых документов) используются различные шаблоны. По сути, шаблоны определяют внешний вид системы и выходных документов.

Сервер баз данных выполняет понятную функцию: накопление структурированной информации и ее выдача по запросам веб-при- ложения. При этом между программной частью системы и сервером баз данных могут использоваться различные интерфейсы (как стандартные, так и специальные). Не все языки программирования вебприложений имеют надежные и функциональные интерфейсы к серверам баз данных различных разработчиков. От сервера баз данных зависит производительность приложения и функциональность с точки зрения данных. Например, развитые СУБД позволяют использовать распределенные хранилища данных и имеют мощные возможности по защите от сбоев и восстановлению после них. На рис. 4.1 обозначена СУБД MySQL, имеющая серьезную репутацию в мире веб-разработок. Кроме MySQL могут применяться многие известные продукты на рынке таких производителей, как Oracle, Microsoft, Sybase, и др.

Кратко про Статический/Динамический сайт.

**Статический сайт –** сайт, состоящий из статичных html (htm, dhtml, xhtml) страниц, составляющих единое целое. Содержит в себе (в виде HTML-размеченных) текст, изображения, мультимедиа содержимое (аудио, видео) и HTML-теги. Теги бывают как служебные, предназначенные для обозревателя, так и предназначенные для размещения, формирования внешнего вида и отображения информации. Все изменения на сайт вносятся в исходный код документов (страниц) сайта, для чего необходимо иметь доступ к файлам на веб сервере.

**Достоинства**

* Для создания страницы не обязательно знать языки веб-программирования.
* Хорошая кэшируемость страницы.
* Быстрая скорость загрузки страниц.
* Минимальные требования к веб-серверу и минимальная нагрузка на него.
* Простота переноса на другой сервер или локальный компьютер.

**Недостатки**

* Невозможность динамической генерации содержимого.
* Невозможность полноценной поддержки посетителей (выбор внешнего вида, cookie).
* Для наполнения сайта информацией необходимо получать доступ к файлам-страницам посредством FTP, или сторонним веб-скриптам, позволяющим редактировать страницы.
* При большом количестве страниц (файлов), если возникает необходимость внести однотипные изменения (дизайн, оформление, добавление новых разделов) необходимо использовать стороннее ПО (утилиты).

**Динамический сайт** – сайт, состоящий из динамичных страниц – шаблонов, контента, скриптов и прочего, в большинстве случаев в виде отдельных файлов (в Lotus Notes/Domino данные и все элементы дизайна, включая пользовательские скрипты, хранятся в одном файле). Страница сайта, показываемая в итоге браузеру пользователя, формируется на стороне сервера динамически, по запросу, из страницы-шаблона и отдельно хранимого содержимого (информации, скриптов и др.). Как правило, для отображения любого количества однотипных страниц используется одна страница-шаблон, в которую подгружается соответствующее содержимое, это позволяет единомоментно корректировать внешний вид сайта (множество всех его страниц), редактируя всего лишь один шаблон.

Редактирование собственно содержимого (как и страницы-шаблона) может производиться как средствами самого сайта, так и с применением стороннего ПО. Возможность править все страницы предоставляется только определенной категории пользователей (например, администраторам, или же зарегистрированным пользователям).

Процесс получения содержимого сайта обычно выглядит следующим образом:

* Генерация содержимого на стороне сервера;
* Передача сгенерированной странички клиенту;
* Генерация содержимого на стороне клиента.

**Проблема совместимости браузеров с веб-приложениями**

Как было отмечено ранее, веб-приложения неприхотливы в части клиентского компонента взаимодействия. В идеале, вы можете сесть за любой современный ПК, подключенный к Интернету, и начать работу в веб-приложении, открыв веб-браузер.

В реальности, существует проблема совместимости веб-приложений с браузерами и операционными системами. Разделим проблемы совместимости на две группы:

* принципиальная несовместимость приложения с браузером (некорректное отображение и работа элементов управления);
* требования к установке дополнительных модулей (расширений, плагинов), имеющих ограниченную совместимость.

Поясним причины первого вида проблемы. Странно, но при всей развитой системе стандартов и регулирующих организаций в Интернете совместимость браузеров с этими стандартами значительно отличается от полной, т.е. при создании HTML-кода страниц и приложений нужно учитывать особенности каждого из основных веб-браузеров, используемых на рынке (MS Internet Explorer, Opera, FireFox, Safari). При этом нет гарантии, что, протестировав приложение на основных видах браузеров, оно будет корректно работать на других, более редких продуктах.

Смысл второй проблемы довольно прост: в большинстве браузеров предусмотрена установка расширений (дополнительных модулей, называемых плагинами). Некоторые веб-приложения построены таким образом, что требуют наличия тех или иных расширений браузера для своей работы. В мире веб-сайтов характерный пример — Flash-объекты (анимированные мини-приложения с расширенными возможностями). До недавнего времени поддержка Flash в операционной системе Linux оставляла желать лучшего, поэтому сайты и приложения, построенные с использованием этой технологии, не могли там работать.

Наиболее часто встречающаяся разновидность несовместимости: веб-приложение ориентировано на работу в среде MS Windows + MS IE. Объясняется такая ситуация тем, что указанная связка обладает монопольным положением на рынке ПК и разработчики, не желающие тратить средства на доработку своих приложений на совместимость со всеми основными браузерами и платформами, выбирают самую распространенную. Такой подход сложно назвать приемлемым, так как тенденция состоит как раз в увеличении доли других веб-браузеров и операционных систем.

Попробуем подвести итог в этом вопросе с точки зрения заказчика бизнес-приложения, построенного на веб-технологиях. Во-первых, нужно точно определить целевую аудиторию приложения. Возможно, групп пользователей будет несколько. Например, в случае с веб-сайтом есть внешние пользователи (посетители) и внутренние пользователи (работники компании, пользующиеся системой управления сайтом). Если круг пользователей неконтролируем, т.е. имеет место публичный веб-сервис, то нужно ориентироваться на самый широкий диапазон совместимости. В противном случае можно потерять часть аудитории своего сервиса. Какова эта часть будет в абсолютных цифрах, можно определить, используя системы глобальной статистики.

Если круг пользователей ограничен и контролируется в плане программных средств, то нужно собрать данные об операционных системах и браузерах (включая версии), которые будут установлены у целевых пользователей. Исходя из этих положений, можно сформировать реальные требования к совместимости веб-приложения с пользовательскими операционными системами и браузерами.

**Технология java как универсальная платформа**

Технология Java – это более чем язык программирования, это платформа. Как правило, платформой называют сочетание аппаратной архитектуры («железа»), которая определяется типом используемого процессора, с операционной системой (Windows, Linux, Mac и др.).

Язык был назван Oak (Java) и представлял собой мощный инструмент для написания программ, работающих в сетевом окружении. Его объекты, распространяясь по сети, могли работать на любом устройстве, начиная с персонального компьютера и заканчивая видеомагнитофонами и тостерами. Кроме того, как язык, ориентированный на распределенную архитектуру, Oak имел механизмы безопасности, шифрования, процедуры аутентификации. Все эти механизмы были встроенными и поэтому незаметными, и удобными для пользователя.

В то время Web-сайты были больше похожи на обычные бумажные документы, чем на интерактивные приложения. Вся их работа заключалась в отправке запроса Web-сервер и получении ответа, который содержал обычный HTML-файл, отображаемый браузером на стороне клиента. В то время уже была освоена технология CGI, которая по запросу клиента запускала обычную программу на Webсервере, и ее результат отсылался обратно в качестве ответа.

Именно поэтому создатели Java задумали разработать не просто еще один язык программирования, а универсальную платформу для исполнения приложений, потому что «прародитель» этого языка OaK изначально создавался для разработки программ управления различными бытовыми приборами, от которых совместимости ждать не приходится.

Наличие виртуальной машины определяет многие свойства Java, однако сейчас остановимся на следующем вопросе – является Java языком компилируемым или интерпретируемым? В большинстве языков программирования программа должна быть либо скомпилирована, либо интерпретирована для того, чтобы компьютер мог ее выполнить. Язык Java необычен тем, что программа одновременно и компилируется, и интерпретируется.

В отличие от CGI-скриптов, код инициализации сервлета выполняется только один раз. Кроме того, обработка каждого клиентского запроса выполняется в отдельном потоке на сервере, что предотвращает создание лишних процессов, увеличивая, таким образом, производительность сервера. При этом сервлеты наследуют все свойства языка программирования Java.

В двух словах описать работу сервлета можно так: при приходе

запроса от клиента сервер с помощью специального конфигурационного файла может определить, какой сервлет выполнить, сервлет выполняется и создает HTML-страницу, которую сервер отправляет клиенту. При этом не разработчик, а сам сервлет или web-контейнер управляют жизненным циклом сервлета, то есть тем, как долго экземпляр сервлета существует в JVM и обрабатывает запросы.

По сути **сервлет** является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос-ответ.

Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP-специфичные сервлет классы.

Пакеты javax.servlet и javax.servlet.http обеспечивают интерфейсы и классы для создания сервлетов.

**Жизненный цикл сервлета состоит из следующих шагов:**

1. В случае отсутствия сервлета в контейнере.

* Класс сервлета загружается контейнером.
* Контейнер создает экземпляр класса сервлета.
* Контейнер вызывает метод init(). Этот метод инициализирует сервлет и вызывается в первую очередь, до того, как сервлет сможет обслуживать запросы. За весь жизненный цикл метод init() вызывается только один раз.

1. Обслуживание клиентского запроса. Каждый запрос обрабатывается в своем отдельном потоке. Контейнер вызывает метод service() для каждого запроса. Этот метод определяет тип пришедшего запроса и распределяет его в соответствующий этому типу метод для обработки запроса. Разработчик сервлета должен предоставить реализацию для этих методов. Если поступил запрос, метод для которого не реализован, вызывается метод родительского класса и обычно завершается возвращением ошибки инициатору запроса.
2. В случае если контейнеру необходимо удалить сервлет, он вызывает метод destroy(), который снимает сервлет из эксплуатации. Подобно методу init(), этот метод тоже вызывается единожды за весь цикл сервлета.