МИНИСТЕРСТВО ОБРАЗОВАНИЯ ПРИМОРСКОГО КРАЯ

краевое государственное автономное профессиональное образовательное учреждение

«Лесозаводский индустриальный колледж»

МАТЕРИАЛЛЫ ДЛЯ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

по МДК 05.01 группа ИС-2, специальность 09.02.07 «Информационные системы и программирование» Контакты преподавателя: maya_tok@mail.ru

Задание: составить конспект лекции, подготовиться к тестированию по данной теме

Лекция_ Веб-ориентированные ИС

С каждым годом web-приложения – вспомогательные программные средства, предназначенные для автоматизированного выполнения действий на web-серверах - приобретают все большую популярность из-за их универсальности, удобства использования и гибкости.

За годы существования Интернета состав web-приложений, выполняемые ими функции, принципы и архитектура их построения претерпели значительные изменения – от простейших средств хранения HTML-страниц до решений, ориентированных на поддержку работы корпоративных информационных систем и их партнеров. Web-системы имеют много преимуществ перед обычными системами, которые работают по технологии клиент-сервер. Достаточно разместить web-приложение на хостинге и можно работать с ним с любого компьютера, который имеет доступ к Интернету. С одной стороны, это удобно, а с другой предъявляет дополнительные требования к надежности создаваемого программного обеспечения. Главное преимущество web-приложений – это удобство в поддержке и администрировании: отсутствие необходимости установки приложения на каждое рабочее место, удобство при обновлении версий, возможность настройки интерфейса для каждого пользователя, а многоуровневая и проверенная система защиты web-приложений ограничит возможность получения данных сторонними лицами. Для современных инновационных учреждений web-системы будут оптимальным выбором при автоматизации рабочих процессов.

Рассмотрим основные технологии создания web-приложений:

1. AJAX (Asynchronous JavaScript and XML) – подход к построению пользовательских интерфейсов web-приложений, при котором в ответ на каждое действие пользователя web-страница на его браузере, не перезагружается полностью – с web-сервера только догружаются нужные ему данные. Этим обеспечивается оперативная работа как одного, так и групп пользователей с приложениями. AJAX представляет собой не одну, а группу технологий и базируется на принципах использования DHTML для динамичного изменения содержания страницы и использования XMLHttpRequest для обращения к серверу. С учетом этих принципов можно создавать удобные web-интерфейсы на тех страницах сайтов, где необходимо активное взаимодействие с пользователями. Популярность AJAX приобрела после того, как компания Google начала применять его при создании Gmail, Google maps, Google suggest.
2. ASP (Active Server Pages) – технология создания web-приложений, использующая объектную модель интерфейса, созданного на основе ISAPIфильтра. ASP упростила задачи генерации HTMLстраниц и позволила производить обращение к компонентам баз данных. Принцип, заложенный в основу интерфейса приложения, заключается в том, что на web-странице присутствуют фрагменты кода, который интерпретируется web-сервером и предоставляет пользователю готовый результат выполнения выбранных фрагментов кода. Новейшей версией технологии Active Server Pages является ASP.NET, ключевая в архитектуре Microsoft .NET Framework. Основное отличие этой тех-

нологии от ASP с точки зрения архитектуры приложений заключается в том, что код, присутствующий на Web-странице, не интерпретируется, а компилируется и кэшируется, что, естественно, способствует повышению производительности приложений. С помощью ASP.NET можно создавать webприложения и web-сервисы, которые не только позволяют реализовать динамическую генерацию HTML-страниц, но и интегрируются с серверными компонентами и могут использоваться для решения широкого круга бизнес-задач, возникающих перед разработчиками современных webприложений.

3. JSP (Java Server Pages) – технология создания web-приложений, основанная на однократной компиляции Java-кода (сервлета) при первом обращении к нему с последующим выполнением методов этого сервлета и помещением полученных результатов в набор данных, которые отправляются в браузер.

Вопросы построения пользовательского интерфейса являются одними из самых важных в процессе разработки приложения. В случае разработки web-приложения они являются особенно актуальными. Выбор средств программной реализации представляет собой сложную задачу и является одним из важных этапов при разработке приложения.

Выбранные программные продукты должны удовлетворять как текущим, так и будущим потребностям предприятия, при этом следует учитывать финансовые затраты на приобретение необходимого оборудования, самой системы, разработку необходимого программного обеспечения на её основе, а также обучение персонала Сегодня рынок предоставляет достаточно широкий выбор для разработчика.

Программная среда Microsoft Visual Studio 2010 – это мощная среда разработки, обеспечивающая высокое качество кода на протяжении всего цикла разработки программного обеспечения, от проектирования до разработки. Продукт позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также web-сайты, webприложения, web-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight [2].

Сайт – основной продукт при создании веб-ориентированных ИС.

Сайты бывают разные и делятся они, как минимум, на два типа – одностраничные и многостраничные. Однако, существует более широкая квалификация:

1. Сайт-визитка.

Простой, одностраничный сайт, содержащий в себе информацию о компании. Сайт-визитка не требует к себе особого внимания, создается «один раз и навсегда». Обычно включает в себя общую информацию о владельце и контакты для связи, не вдаваясь в подробности.

2. Сайт компании.

Более серьезный подход для создания сайта. Обычно, это многостраничный сайт, рассказывающий подробнее об услугах, истории и контактах компании. Информирует посетителей о стоимости услуг и по факту выполняет функции офиса компании в режиме онлайн.

3. Промо-сайт.

Промо-сайты используются для быстрой раскрутки какой-либо рекламной акции компании, а также для быстрого продвижения услуг, товаров и брендов.

4. Интернет-магазин.

Подходит для онлайн торговли. Может быть создан с возможностью заказа и оплаты товара через интернет, бронирования услуги и т.п. Также может быть создан в качестве онлайн витрины, без возможности оформления заказа.

5. Порталы.

Ресурсы, на которых можно размещать информацию различного характера. Более продвинутый функционал, чем у блога. Чаще всего – это новостные сайты.

6. Доски объявлений.

Сайт, где можно размещать информацию о покупке или продаже товаров, услуг, информационные объявления и прочее.

7. Блог.

Любите писать интересный контент? Тогда без блога не обойтись. Делитесь текстами, медиа и ссылками. Наполняйте сайт самостоятельно, рассказывайте людям о том, что интересно вам и вашей аудитории.

Этапы создания сайта (http://www.connectdesign.ru/web-design/index.shtml**)**

Рабочий процесс предполагает несколько основных этапов

Предварительный этап

Цель предварительного этапа разработки — выявить все требования к дизайну и технологической основе проекта, а также определить цели и задачи ресурса и составить подробный план разработки проекта. В ходе одной или серии предварительных встреч будет составлен предварительное техническое задание и «бриф о дизайне», содержащий требования к графическому оформлению проекта. Координировать работу специалистов и общаться с сотрудниками компании будет менеджер проекта. Менеджером проекта является технический специалист или дизайнер в зависимости от сложности проекта. В рамках этого этапа определяется стоимость проекта, составляется договор и вносится предоплата.

Резервирование доменного имени и размещение краткой информации

Многие доменные имена уже заняты и часто приходится проявлять изобретательность - мы готовы помочь с поиском оптимального имени. Чтобы сайт начал работать как можно быстрее, на период разработки сайта, на первой странице размещается логотип, контакты и краткая справка о сфере деятельности компании. Не следует создавать страницы с названием "Under Construction". Это отпугивает посетителей от повторного посещения.

Разработка архитектуры проекта

Сначала нужно ответить на вопрос: зачем нужен сайт? Какие цели и задачи он будет выполнять? В некоторых проектах необходимо провести маркетинговые исследования, выявить ключевые потребности пользователей сайта. На основании потребностей компании составляется техническое задание и утверждается план работ.

Разработка концепции дизайна

Исходя из фирменного стиля компании и технического задания, разрабатываются макеты ключевых страниц. Работа над дизайном сайта предполагает следующие стадии: Анализ требований пользовательского интерфейса. Интуитивно понятная организация материала и продуманная навигация являются необходимыми условиями эффективного дизайна. Мы стремимся к тому, чтобы наши разработки были удобны и понятны любому пользователю.

Одобрение дизайна

Стадия предполагает согласование и внесение необходимых корректив в первоначальную концепцию дизайна проекта. Результатом работы на этапе разработки дизайна и архитектуры проекта станут эскизы всех типовых страниц веб-сайта.

Создание технологической основы

На основании технического задания и утвержденной концепции дизайна будет разработана оптимальная структура базы данных, созданы действующие шаблоны всех разделов сайта, а также разработаны и настроены интерактивные сервисы. Параллельно будет вестись разработка механизмов и интерфейсов системы администрирования сайта, которая позволит сотрудникам компании оперативно обновлять материалы сайта и настраивать сервисные функции. На данном этапе будет создана рабочая версия сайта, готовая к наполнению текстовыми и графическими материалами.

Текстовый редактор, чтобы писать код. Это может быть текстовый редактор (например, Visual Studio Code, Notepad++, Sublime Text, Atom, GNU Emacs, или VIM) или гибридный редактор (например, Dreamweaver or WebStorm)

Наполнение контентом

На этом этапе происходит наполнение контентом баз данных сайта. При необходимости наши дизайнеры и копирайтеры создадут и обработают дополнительные графические материалы для наполнения разделов сайта. Мы также готовы провести консультации и принять участие в написании, стилистической обработке и переводе текстовых материалов, которые будут опубликованы на сайте. После создания соответствующих шаблонов и механизмов все тексты, файлы для скачивания

и необходимые иллюстрации будут сверстаны либо перенесены через административный интерфейс в базу данных и опубликованы на сайте.

Тестирование готового решения

После завершения всех работ по интеграции систем и наладке оборудования интернет-представительство будет предоставлено на тестирование сотрудникам Вашей компании. Настройка оборудования и перенос на хостинговую платформу Мы готовы взять на себя консультационные функции по выбору и настройке оборудования, необходимого для размещения и корректного функционирования сайта. Хорошо зная все параметры и особенности работы проекта, мы поможем вам подобрать оптимальную конфигурацию оборудования и программного обеспечения.

Запуск проекта

После завершения всех процедур и работ по тестированию сайта и обучению специалистов группы поддержки проект будет готов к открытию для посетителей. На данном этапе подписываются необходимые документы о приемке-сдаче.

Поддержка

Чтобы сайт начал работать и ваши пользователи смогли вас найти, он регистрируется в различных поисковых системах и каталогах. Регистрация — лишь первый шаг в рекламной компании. Мы готовы помось в проведении рекламной компании и раскрутке сайта, а также наполнять его материалами в ходе дельнейшего использования.

Рабочая группа

Для выполнения всех работ по созданию сайта из числа сотрудников студии будет сформирована постоянная рабочая группа. Ее составят специалисты, которые хорошо знакомы со всеми особенностями процесса создания сайта.

Координировать работу этих специалистов и общаться с сотрудниками компании будет менеджер проекта. В круг его задач будет входить организация встреч, ведение переговоров и сбор необходимых для работы данных. В любое рабочее время сотрудники компании смогут связаться с ним по телефону или электронной почте, чтобы оперативно решить все рабочие вопросы. Он также будет составлять план работ студии, контролировать процесс выполнения задач и представлять результаты работы.

Функции специалистов рабочей группы:

Дизайнер: разработка эскизов типовых страниц и элементов сайта; создание графических форм и элементов навигации; актуализация элементов дизайна.

Веб-технолог: проектирование баз данных; скриптинг; разработка, установка и настройка интерактивных сервисов; разработка интерфейса и механизмов системы администрирования сайта.

HTML-кодер: верстка и адаптация текстового наполнения; оптимизация HTML-кода; размещение иллюстраций и графических элементов.

Редактор-копирайтер: редактирование и корректура текстов; написание дополнительных текстовых блоков и анонсов; контент-менеджмент.

https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web - проектирование веб на HTML с JavaScript

https://docs.microsoft.com/ru-ru/visualstudio/ide/?view=vs-2019 -руководство Visualstudio для написания кода.

Рассмотрим основные технические компоненты и строение сайта.

1. **Дизайн сайта** отвечает за визуальное представление и организацию информации, способствует взаимодействию ресурса с его посетителями. Отвечающим за создание дизайна специалистом обычно является веб-дизайнер. Именно он должен спроектировать логическую структуру всех страниц сайта, разработать способы подачи материала и проработать внешнее оформление ресурса.

Для каждой страницы дизайн разрабатывается отдельно в зависимости от ее функций.

2. **Верстка** представляет собой процесс написания особого кода для браузеров. У такого файла расширение html, и отображается он только в браузерах. Нужно следить, чтобы сверстанный макет одинаково выглядел во всех браузерах, будь то Opera, Mozilla Firefox, Google Chrome или Internet Explorer.

3. **Программирование** составляет примерно 50-70% работы над сайтом. Программная часть ресурса – довольно обширное понятие. Программирование отвечает за то, чтобы многочисленные страницы сайта отображались с одинаковым дизайном, но разным содержимым. Также программный код ответственен за визуальные эффекты и многое другое. Благодаря ему сайт становится «живым» и динамичным.

4. **Веб-сервер** – это компьютер с установленным на нем программным обеспечением, которое предназначено для того, чтобы отвечать на запросы веб-клиента круглосуточно в режиме реального времени. Под веб-клиентом подразумевается браузер, который отправляет серверу запрос на определенную страницу, и если она доступна, то пользователь может увидеть ее. Чтобы пользователь увидел нужную ему страницу, его браузер должен получить от сервера соответствующий html-код. После чего код и все визуальные элементы распознаются и предстают перед пользователем в виде готовой понятной любому страницы.

5. **Клиентская часть** представляет собой код, загружаемый вместе с кодом html. Это может быть CSS, JavaScript, ActionScript.

6. **Контент сайта**, то есть все содержимое сайта. Сюда входят тексты, картинки, видео, flash и прочие файлы. Контент часто путают с элементами дизайна, но это совершенно разные вещи. Оформление сайта отвечает на вопрос «как разместить», а контент отвечает на вопрос «что разместить». Что касается текстового наполнения, то именно оно наполняет любой веб-ресурс смыслом, благодаря которому пользователи могут найти сам сайт в поисковых системах. Интересный контент – это ключ к увеличению посещаемости, читаемости и коммерческих показателей, таких как звонки, продажи или подписки на e-mail рассылки.

7. **CMS - системы управления содержимым,** тоже можно отнести к одному из элементов сайта. Система управления содержимым позволяет управлять всеми элементами ресурса через административную часть. Использование CMS не обязательно, однако оно необходимо всем, кто хочет самостоятельно контролировать и редактировать содержимое сайта. Существует целый ряд разнообразных CMS, отличающихся друг от друга удобством пользования, набором свойств и принципами работы.

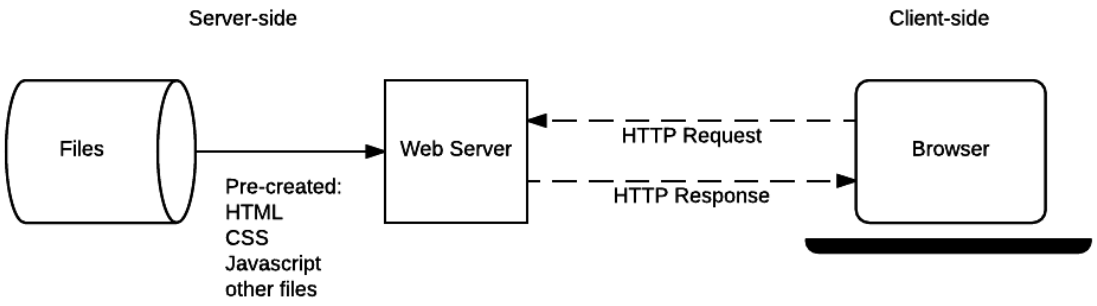
8. **Доменное имя и хостинг**. **Доменное имя** – это уникальный адрес сайта в Сети (например, www.site.by). А **хостинг** – это услуга, которая предоставляется специальными компаниями, хранящими на своих серверах все необходимые для работы сайта данные. Без этих компонентов строение было бы неполным, а сам сайт - недоступен для пользователей и посетителей. Хостинг предполагает выделение свободного места на сервере для размещения и хранения какого-либо сайта. Чтобы владелец был уверен в том, что его сайт всегда виден пользователям Интернета, он обязательно должен обзавестись уникальным доменным именем, а также надежным хостингом.

Какие бывают веб-страницы.

1. Статические

Схема ниже показывает базовую архитектуру веб-сервера для статического сайта (статический сайт - это тот, который возвращает одно и то же жестко закодированное содержимое с сервера всякий раз, когда запрашивается конкретный ресурс). Когда пользователь хочет перейти на страницу, браузер отправляет HTTP-запрос «GET» с указанием его URL.

Сервер извлекает запрошенный документ из своей файловой системы и возвращает HTTP-ответ, содержащий документ и успешный статус (обычно 200 OK). Если файл не может быть извлечен по каким-либо причинам, возвращается статус ошибки (смотри ошибки клиента и ошибки сервера).



2. Динамические

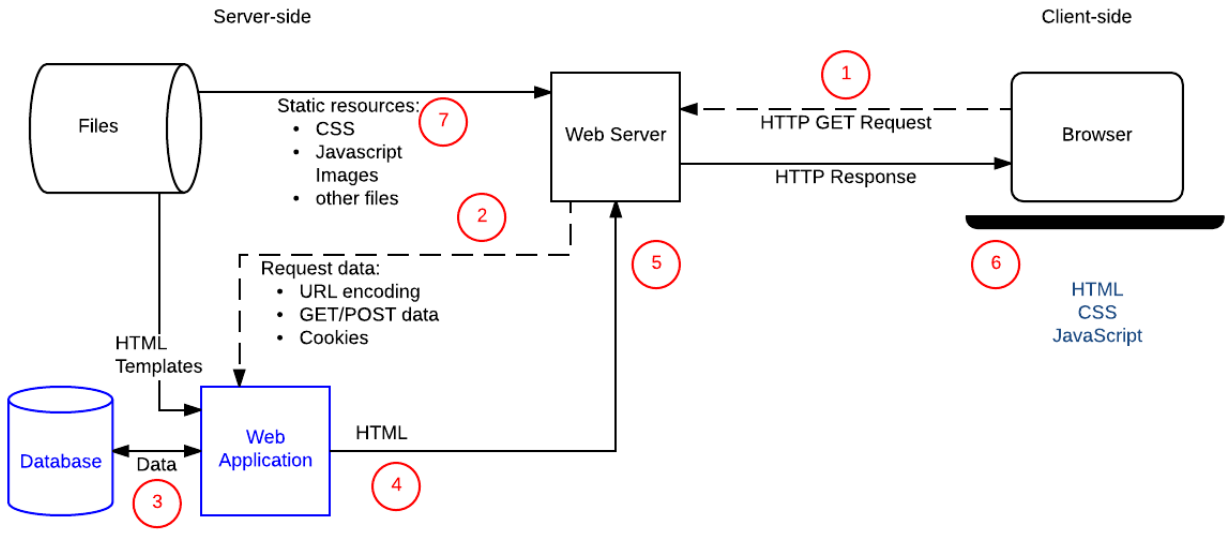
Динамический веб-сайт - это тот, где часть содержимого ответа генерируется динамически только при необходимости. На динамическом веб-сайте HTML-страницы обычно создаются путем вставки данных из базы данных в заполнители в HTML-шаблонах (это гораздо более эффективный способ хранения большого количества контента, чем использование статических сайтов).

Динамический сайт может возвращать разные данные для URL-адреса на основе информации, предоставленной пользователем или сохраненными настройками, и может выполнять другие операции как часть возврата ответа (например, отправку уведомлений).

Большая часть кода для поддержки динамического веб-сайта должна выполняться на сервере. Создание этого кода известно, как «**программирование серверной части**» (или иногда «**программирование бэкенда**»).

Схема ниже показывает простую архитектуру *динамического сайта*. Как и на предыдущей схеме, браузеры отправляют HTTP-запросы на сервер, затем сервер обрабатывает запросы и возвращает соответствующие HTTP-ответы.

Запросы статических ресурсов обрабатываются так же, как и для статических сайтов (статические ресурсы – это любые файлы, которые не меняются, обычно это: CSS, JavaScript, изображения, предварительно созданные PDF-файлы и прочее).



Запросы динамических данных отправляются (2) в код серверной части (показано на диаграмме как *Веб-приложение*). Для «динамических запросов» сервер интерпретирует запрос, читает необходимую информацию из базы данных (3), комбинирует извлеченные данные с шаблонами HTML и возвращает ответ, содержащий сгенерированный HTML (5, 6).

Одинаково ли программирование серверной части и клиентской?

Теперь обратим внимание на код, задействованный в серверной части и клиентской части. В каждом случае код существенно различается:

- Они имеют различные цели и назначение.
- Как правило, они не используют одни и те же языки программирования (исключение составляет JavaScript, который можно использовать на стороне сервера и клиента).
- Они выполняются в разных средах операционной системы.

Код, который выполняется в браузере, известный как **код клиентской части**, прежде всего связан с улучшением внешнего вида и поведения отображаемой веб-страницы. Это включает в себя выбор и стилизацию компонентов пользовательского интерфейса, создание макетов, навигацию, проверку форм и т. д. Напротив, программирование веб-сайта на стороне сервера в основном включает выбор содержимого, которое возвращается браузеру в ответ на запросы. Код на стороне сервера обрабатывает такие задачи, как проверка отправленных данных и запросов, использование баз данных для хранения и извлечения данных и отправка правильных данных клиенту по мере необходимости.

Код клиентской части написан с использованием HTML, CSS и JavaScript - он запускается в веб-браузере и практически не имеет доступа к базовой операционной системе (включая ограниченный доступ к файловой системе).

Веб-разработчики не могут контролировать, какой браузер может использовать каждый пользователь для просмотра веб-сайта - браузеры обеспечивают противоречивые уровни совместимости с функциями кода на стороне клиента, и одной из задач программирования на стороне клиента является изящная обработка различий в поддержке браузера.

Код серверной части может быть написан на любом количестве языков программирования – примеры популярных языков серверной части включают в себя PHP, Python, Ruby,C# и NodeJS(JavaScript). Код серверной части имеет полный доступ к операционной системе сервера, и разработчик может выбрать какой язык программирования (и какую версию) он хотел бы использовать.

Разработчики обычно пишут свой код, используя **веб-фреймворки.**

Веб-фреймворки - это наборы функций, объектов, правил и других конструкций кода, предназначенных для решения общих проблем, ускорения разработки и упрощения различных типов задач, стоящих в конкретной области.

И снова, поскольку и клиентская и серверная части используют фреймворки, области очень разные и, следовательно, фреймворки тоже разные. Фреймворки клиентской части упрощают вёрстку и представление данных, тогда как фреймворки серверной части обеспечивают много «обычного» функционала веб-сервера, который вы возможно в противном случае должны были осуществлять самостоятельно (например, поддержка сессий, поддержка пользователей и аутентификация, простой доступ к базе данных, шаблонам библиотек, и т.д.).

Внимание: Фреймворки клиентской части часто используются для ускорения написания кода клиентской части, но вы также можете решить писать весь код руками; на самом деле, написание кода руками может быть более быстрым и эффективным, если вам нужен небольшой простой веб-сайт UI.

И наоборот, вы практически никогда не посмотрите в сторону написания кода серверной части веб-приложения без фреймворка: осуществление жизненно важной функции, такой как HTTP сервер действительно сложно сделать с нуля скажем на Python, но веб-фреймворки для Python, такие как Django обеспечивают это из коробки вместе с другими полезными инструментами.

Краткий обзор технологий для Интернет-приложений (Наталия Елманова)

Технологии, применяемые в Web-клиентах

- Скриптовые языки
- Java-апплеты
- Элементы управления ActiveX
- Приложения Macromedia Flash

Технологии создания серверных частей Web-приложений

- CGI
- ISAPI и Apache DSO
- ASP, JSP, PHP
- ASP .NET
- Несколько слов о серверах приложений
- Web-сервисы

10-15 лет назад большинство Web-сайтов представляло собой набор статических HTML-страниц. Сегодня подобные сайты по-прежнему встречаются — нередко именно так выполнены небольшие персональные Web-сайты, а также сайты небольших компаний, не претендующие ни на что, кроме размещения относительно небольшого объема редко меняющейся информации.

Отметим, однако, что в процессе превращения Интернета из набора информационных ресурсов в инструмент ведения бизнеса технологии создания сайтов существенно изменились — большинство Web-сайтов крупных компаний представляет собой набор приложений, обладающих интерактивностью, средствами персонализации, средствами взаимодействия с клиентами (вплоть до приема заказов и платежей) и партнерами, а нередко — и средствами интеграции с «внутренними» корпоративными приложениями компании. О средствах создания подобных Web-сайтов чуть более подробно будет рассказано в статье «Продукты для создания корпоративных Интернет-решений» в настоящем номере журнала.

Технологии, применяемые в Web-клиентах

Одним из направлений развития Web-приложений стало размещение некоторой части логики приложения (такой как проверка корректности вводимых данных) в самом Web-клиенте, например в Web-браузере. В частности, современные Web-браузеры способны интерпретировать код на скриптовых языках, выполнять Java-апплеты и элементы управления ActiveX, использовать другие дополнения, такие как Macromedia Flash Player. Рассмотрим все эти возможности браузеров подробнее.

Скриптовые языки

Большинство современных Web-браузеров способно интерпретировать код на скриптовых языках, таких как VBScript и JavaScript. Код на этих языках внедряется в Web-страницу и интерпретируется браузером. Типичный пример применения скриптовых языков — проверка корректности

данных, вводимых пользователем в соответствующие поля HTML-формы, непосредственно в процессе ввода или после него, без обращения к Web-серверу.

Отметим, что код, созданный с помощью скриптовых языков, не может работать самостоятельно — он выполняется в адресном пространстве браузера. Кроме того, скриптовые языки содержат ограниченный набор средств (например, они не обладают средствами доступа к файловой системе).

Java-апплеты

Практически все современные браузеры способны отображать и выполнять Java-апплеты — специальные Java-приложения, которые пользователь получает в составе Web-страницы. Эти приложения нередко включаются в состав Web-страниц с целью добавления функциональности, которую сложно или невозможно реализовать с помощью скриптовых языков. Апплеты могут выполняться на всех платформах, для которых доступна виртуальная Java-машина.

Отметим, что, поскольку апплеты реализуют выполнение кода на компьютере клиента, они в определенной степени являются потенциально опасным содержимым. Именно поэтому все современные браузеры обладают доступными пользователю средствами ограничения возможностей выполнения апплетов.

Элементы управления ActiveX

Некоторые из современных браузеров (в частности, Microsoft Internet Explorer) могут служить контейнерами для элементов управления ActiveX — специальных COM-серверов, выполняющихся в адресном пространстве браузера и также получаемых в составе Web-страницы.

С помощью элементов управления ActiveX, как и посредством Java-апплетов, можно реализовать любую функциональность, в том числе и неблагоприятную для компьютера пользователя, при этом, в отличие от Java-апплетов, при выполнении элементов управления ActiveX в общем случае нет никаких ограничений на доступ к файлам и иным ресурсам операционной системы и сети, а код, содержащийся в них, выполняется от имени загрузившего их пользователя.

Для контроля безопасности их выполнения имеется еще одно средство, называемое электронной цифровой подписью. Цифровая подпись помещается внутрь элемента управления ActiveX, для чего требуется наличие соответствующего электронного сертификата. Электронная подпись, помимо сведений о фирме-производителе, содержит и другую полезную информацию. Так, например, если файл с элементом управления ActiveX после добавления электронной подписи был изменен, то об этом будет немедленно сообщено перед запуском такого элемента управления — при добавлении подписи к элементу управления ActiveX происходит вычисление контрольной суммы соответствующего файла. Отметим, однако, что в России в настоящее время нет авторизованных компаний, которые могли бы выдать электронный сертификат международного образца. Естественно, наличие электронного сертификата не гарантирует отсутствия потенциально опасного содержимого, но, по крайней мере, позволяет клиенту установить его источник.

Далее нам следует напомнить банальную истину, которая, как показывает практика, очевидна не для всех наших читателей. При работе с элементами управления ActiveX и Java-апплетами абсолютно бесполезно полагаться на антивирусное программное обеспечение (неважно, клиентское оно или серверное): признаков, характерных для вирусов (таких как способность внедряться внутрь исполняемых файлов и документов), подобные приложения, как правило, не содержат. Можно лишь запретить загрузку или выполнение соответствующего кода либо на уровне настроек браузера, либо на уровне корпоративных или персональных брандмауэров.

Приложения Macromedia Flash

Приложения Macromedia Flash являются сегодня наиболее популярным расширением функциональности Web-браузеров — с их помощью многие Web-дизайнеры придают своим сайтам интерактивность и оригинальность.

Модель безопасности приложений Flash основана на том, что Macromedia Flash Player, как и виртуальная Java-машина, выполняет приложения в ограниченном адресном пространстве, при этом выполняемые приложения не имеют доступа к файловой системе (кроме одного конкретного каталога, используемого Macromedia Flash Player для служебных целей) и другим ресурсам компьютера пользователя; исключение делается для микрофонов и видеокамер, однако пользователь должен дать разрешение на передачу данных, полученных с этих устройств. Доступ к сетевым ресурсам ограничивается доменом, с которого было получено приложение. Отметим, что приложения Flash также могут управляться с помощью кода JavaScript, присутствующего на той же странице.

Технологии создания серверных частей Web-приложений

Как мы уже убедились, возможности, связанные с выполнением кода в Web-клиентах, могут быть существенно ограничены как технологически, так и с помощью администрирования и пользовательских настроек. Это в целом соответствует вполне разумным требованиям безопасности. Именно поэтому, наряду с развитием средств расширения функциональности браузеров, развивались и технологии, связанные с выполнением кода приложений не в браузерах, а на самих Web-серверах. Ниже мы очень кратко рассмотрим наиболее распространенные из них.

Common Gateway Interface (CGI) — это стандартный интерфейс, позволяющий выполнять серверные приложения, вызываемые через URL. Входной информацией для таких приложений служит содержимое HTTP-заголовка либо тело запроса, в зависимости от применяемого протокола. CGI-приложения генерируют HTML-код, который возвращается браузеру. Основная проблема всех CGI-приложений заключается в том, что при каждом клиентском запросе сервер загружает это приложение в отдельное адресное пространство, а затем инициирует его выполнение и выгрузку. Эта особенность ограничивает производительность приложений и возможность одновременной обработки большого количества клиентских запросов.

ASP, JSP, PHP

Очередной шаг в развитии технологий создания Интернет-приложений — появление средств, позволяющих отделить задачи Web-дизайна от задач, связанных с реализацией функциональности

приложений. Первой из таких технологий стала Active Server Pages (ASP). Основная идея ASP заключается в создании Web-страниц с внедренными в них фрагментами кода на скриптовых языках. Однако, в отличие от рассмотренных выше средств применения скриптовых языков для расширения функциональности браузеров, указанные фрагменты кода интерпретируются не браузером, а сервером (точнее, предназначенной для этого ISAPI-библиотекой), и результат выполнения этих фрагментов кода замещает сам фрагмент кода в той версии страницы, которая передается в пользовательский браузер.

Вскоре после ASP появились и другие технологии, реализующие идею размещения внутри Web-страницы кода, выполняемого Web-сервером. Наиболее известной из них сегодня является технология JSP (Java Server Pages), основная идея которой — однократная компиляция Java-кода (сервлета) при первом обращении к нему, выполнение методов этого сервлета и помещение результатов выполнения этих методов в набор данных, отправляемых в браузер. Еще одной популярной технологией подобного типа является PHP (Personal Home Pages), которая использует CGI-приложения, интерпретирующие внедренный в HTML-страницу код на скриптовом языке.

ASP .NET

Новейшей версией технологии Active Server Pages является ASP .NET, ключевая в архитектуре Microsoft .NET Framework. Основное отличие этой технологии от ASP с точки зрения архитектуры приложений заключается в том, что код, присутствующий на Web-странице, не интерпретируется, а компилируется и кэшируется, что, естественно, способствует повышению производительности приложений.

С помощью ASP .NET можно создавать Web-приложения и Web-сервисы, которые не только позволяют реализовать динамическую генерацию HTML-страниц, но и интегрируются с серверными компонентами и могут использоваться для решения широкого круга бизнес-задач, возникающих перед разработчиками современных Web-приложений.

Что такое CMS и как ее использовать

Современные сайты ушли далеко вперед по сравнению с более простыми собратьями из прошлого десятилетия. Помимо модного дизайна и продуманных методов раскрутки, они включают в себя солидный набор функций, который сложно повторить простой самописной странице. Полноценный сайт с серьезным набором возможностей проблематично написать «с нуля», поэтому программистам во многом помогает CMS, более известная как движок.

Что представляет собой CMS

Знать, что такое CMS и какое у нее назначение, стоит каждому, кто имеет хоть небольшое отношение к теме разработки веб-ресурсов. Аббревиатура расшифровывается как Content Management System (система управления контентом). Название точно отражает суть. CMS – это ПО на базе скриптов, которое позволяет управлять содержимым ресурса, менять его, просматривать и контролировать. Сегодняшние системы обладают широкой функциональностью и состоят из огромного количества модулей, каждый из которых отвечает за свои элементы. Программное обеспечение помогает

составлять типовые сайты из блоков подобно конструктору. Для этого практически не требуется даже навыков программирования. Условно CMS можно разделить на несколько частей:

- хранилище баз данных, где находится информация о пользователях, наполнении сайта и других важных сущностях;
- хранилище элементов интерфейса, с которыми непосредственно взаимодействует пользователь при просмотре сайта;
- визуальный редактор, помогающий с легкостью создавать страницы.

Помимо того, множественные модули позволяют добавить к сайту те или иные дополнительные функции.

Для чего используется система

Современные CMS используются крайне широко: без них сложно обойтись любой компании, которая выходит на интернет-площадки и нуждается в собственном сайте. В отличие от специализированных IT-фирм, обладающих профессиональными командами специалистов, большинство непрофильных организаций не может обеспечить себе создание ресурса с нуля и потому применяет распространенные CMS для разработки типового сайта. Это отличное решение для тех, кто нуждается в ресурсе со стандартным набором функций, будь то визитка или интернет-магазин. CMS позволяет:

- наполнять сайт контентом, изменять и администрировать ресурс, при этом не являясь IT-специалистом и не имея серьезных навыков программирования;
- создавать новые страницы в короткие сроки без лишних затрат;
- оптимизировать внешний вид сайта и улучшать качество его наполнения.

От CMS во многом зависят функциональность ресурса, его возможности и удобство для пользователя. Правильно выбранная система позволит успешно создать и раскрутить сайт, сделав его привлекательным для клиента, надежным и работающим ровно так, как требуется.

Преимущества и недостатки

Из уже сказанного очевидно, что использование системы управления контентом при разработке сайта имеет сразу несколько важных преимуществ. Они особенно актуальны для владельцев непрофильных компаний, в штате которых нет большого количества веб-программистов:

- отсутствие необходимости в серьезных технических знаниях, продвинутых навыках программирования и верстки;
- высокая скорость создания сайта;
- удобство администрирования и добавления новых элементов;
- возможность без лишних сложностей создать красивый дизайн;
- простое наполнение, доступное даже людям без специальных знаний.

Эти плюсы делают использование CMS оптимальным решением для большинства типовых сайтов – намного более выгодным, чем создание ресурса «с нуля». Некоторые компании могут предлагать самописные системы, однако в большинстве случаев они несравнимы по качеству с популярными аналогами.

Единственным недостатком можно назвать сложность создания уникального сайта с нестандартными функциями, но такие ресурсы, как правило, требуются компаниям с соответствующими запросами (к примеру, работающим в IT-сфере). Подобные организации могут позволить себе штат программистов, которые самостоятельно разработают сайт.

Как выбрать CMS

Перед непосредственным созданием сайта для начала стоит изучить рынок: он предлагает множество вариантов с разными возможностями, условиями использования и ограничениями. Можно сказать, что все CMS делятся на две большие группы: открытые системы, которые распространяются бесплатно и позволяют пользователям редактировать исходный код, и проприетарные закрытые решения, которые не открывают код и, как правило, предоставляются на платной основе. Выделяют также автономные и динамические движки: первые используются для создания статичных сайтов, вторые – для интерактивных. На рынке существует несколько популярных систем:

- Drupal – бесплатная, но полнофункциональная и достаточно тяжелая CMS, имеющая в составе все необходимое для создания полноценного сайта;
- 1С Битрикс – объемная, многопрофильная платная система, чересчур тяжеловесная для простых задач, но хорошо справляющаяся со сложными;
- Joomla – крайне простой в использовании бесплатный движок, который применяют начинающие сайтостроители и компании, не требующие от ресурса мощных вычислений;
- MODx – удобная для разработчиков бесплатная CMS, обладающая высокой степенью защищенности и достаточной гибкостью для решения большинства задач;
- WordPress – известный по всему миру движок, который изначально предназначался для создания блогов, однако на данный момент имеет куда более широкую функциональность;
- DLE – отчасти аналог предыдущей системы, простой в использовании и интуитивно понятный;

- движки для создания форумов: phpBB, vBulletin и другие;
- системы для организации интернет-магазинов: как бесплатные (OpenCart, PrestaShop), так и платные (Umi.CMS, Shop-Script и другие);
- прочие конструкторы с разными функциями, но, как правило, в простых и малоизвестных CMS принцип работы и возможности довольно ограничены.