МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

ТЕХНОЛОГИИ ИНТЕРНЕТ

УЧЕБНОЕ ПОСОБИЕ

Б.Ю. ХОДИЕВ Т.И. САРСАТСКАЯ

Ташкент

МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

Б.Ю. ХОДИЕВ, Т.И.САРСАТСКАЯ

Технологии Интернет

Рекомендовано в качестве учебного пособия для студентов высших учебных заведений Министерством высшего и среднего специального образования Республики Узбекистан

Technologies the Internet

Editor in chief: academician S.S. Gulamov

Ташкент - 2003

Ходиев Б.Ю., Сарсатская Т.И. Технологии Интернет. – Уч. пособие. – Ташкент, изд.ТГЭУ, 2003. – с.108.

Данное учебное пособие вторая книга в серии учебных пособий по современным информационным технологиям.

Настоящее пособие посвящено специальным технологиям Интернет. Основное внимание уделяется Всемирной Сети «изнутри», приводится обзор современных технологий глобальной Сети, рассматриваются возможности, сфера использования, достоинства и недостатки таких технологий, как Java, CGI, SSI,CSS, Macromedia Flash и др. Даны основы языка разметки гипертекста HTML.

Пособие рассчитано на преподавателей, аспирантов, студентов, специалистов и предпринимателей, работающих в области информационных технологий. Оно может представлять интерес для всех желающих ознакомиться с состоянием и перспективами развития сети Интернет.

Научный редактор **академик С.С. Гулямов**

Рецензенты: **д.э.н., проф. Абдувахидов А.М.,**
д.э.н., проф. Шодиев Т.Ш.

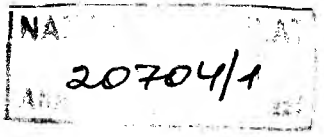
Khodiev B. Yu., Sarsatskaya T. I. Internet technology – Textbook – Tashkent, published in TSUE, 2003. - 108 pg.

The present textbook is a second book in Modern Information Technology serial. The book is devoted to special Internet technology. Main attention is paid to the "inner" side WorldNet, survey of existing technologies is given; the book considers the possibilities, sphere of usage, advantages and disadvantages of search technologies as Java, CGI, SSI, Macromedia Flash and others. It gives the language basis of hypertexts marking of HTML.

The textbook is useful for teachers, postgraduates, students, specialists and businessmen working in the field of information technology. It can also be useful for those who are interested in studying the conditions and the prospects of Internet development.

Editor in chief **academician S.S. Gulamov.**

Reviewers: **prof. A.M. Abduvahidov,**
prof. T.Sh. Shodiev.



1. Современный Интернет

Интернет, Всемирная паутина, информационная магистраль, киберпространство, "седьмой континент", цифровая нервная система и, наконец, Сеть. У нее много имен, но еще больше лиц. Невозможно найти область человеческой деятельности, которая не соприкасалась бы с ней, и не могла бы извлечь из нее какие-либо выгоды. Сеть сетей, связывающая миллионы компьютеров во всем мире, уже давно превратилась из игрушки в ежедневно необходимый инструмент. Какое самое сильное впечатление от первого контакта с Интернет? Исчезновение времени. Человеку, работающему в Интернет, уже сейчас время привычнее измерять в миллисекундах. И если раньше в ожидании писем от близких людей мы спокойно проводили дни, месяцы, а иногда и годы, то теперь нервничаем, когда пакеты, летящие до ближайшего DNS сервера, задерживаются на 1-2 секунды. Интернет сегодня популярен из-за своих возможностей в поиске и доступе к необходимой информации, ее обмене, а также как способ оперативного представления на весь мир своей информации. Замечательным свойством Интернета является его способность передавать по своим каналам текст, изображение и голос, что позволяет заменить им почту, телефон, радио и даже ТВ. Через Интернет доступны многие архивы, газеты, журналы, большая часть из которых существует только в Интернете. Удобная и простая система для пользователя система Интернет на самом деле имеет сложное внутренне строение. В настоящей работе рассматриваются основные технологии, позволяющее Интернет функционировать эффективно.

2. Основы Web-технологий

Основа WWW — файлы в формате HTML (Hyper Text Markup Language — язык разметки гипертекста), или гипертекстовые страницы. Гипертекст — это легкая в использовании и чрезвычайно мощная система связанных слов и фраз, позволяющая легко перемещаться по специальным образом организованным страницам. Она связывает фразу или слово одной страницы с любой другой страницей, абзацем, фразой или словом. Например, если на Web-странице в предложении «Начальный этап правления Амира Темура - периода расцвета узбекской государственности» слова «Амир Темур» выделены как гиперссылка, тогда, щелкнув по ней, вы попадаете на страничку с историческим комментарием о роли Амира Темура в истории. Если развить идею гипертекста и включить в него графику, видео и звук, мы получим гипермедиа. Гипермедиа — среда, основанная, как и гипертекст, на взаимосвязях, в которой в качестве гиперссылок могут выступать визуальные и аудиокомпоненты. Гипертекст и гипермедиа являются

фундаментальными для WWW технологиями, а HTML — средство для работы с этими технологиями.

Идея гипертекстовой информационной системы в том, что пользователь имеет возможность просматривать документы (страницы текста) в том порядке, в котором ему это больше нравится, а не последовательно, как это принято при прочтении книг. Поэтому Т.Нельсон и определил гипертекст как нелинейный текст. Достигается это путем создания специального механизма связи различных страниц текста при помощи гипертекстовых ссылок, т.е. у обычного текста есть ссылки типа "следующий – предыдущий ", а у гипертекста можно построить еще сколь угодно много других ссылок. Примерами гипертекста являются энциклопедии, системы типа «Help».

Простой, на первый взгляд, механизм построения ссылок оказывается довольно сложной задачей, так как можно построить статистические ссылки, динамичные ссылки, ассоциированные с документом в целом или только с отдельными его частями, т.е. контекстные ссылки. Дальнейшее развитие этого подхода приводит к расширению понятия гипертекста за счет других информационных ресурсов, включая графику, аудио-видеоинформацию до понятия гипермедиа.

2.1.Основные компоненты технологии World Wide Web

К 1989 году гипертекст представлял новую, многообещающую технологию, которая, с одной стороны, имела относительно большое число реализаций, а, с другой стороны, делались попытки построить формальные модели гипертекстовых систем, которые носили скорее описательный характер и были навеяны успехом реляционного подхода описания данных. Идея Тима Бернерс Ли заключалось в том, чтобы применить гипертекстовую модель к информационным ресурсам, распределенным в сети, и сделать это максимально простым способом. Он заложил три краеугольных камня системы из четырех существующих ныне, разработав:

- Язык гипертекстовой разметки документов HTML (Hyper Text Markup Lanquage).
- Универсальный способ адресации ресурсов в сети URL (Universal Resource Locator).
- Протокол обмена гипертекстовой информацией HTTP (Hyper Text Transfer Protocol).
- Универсальный интерфейс шлюзов CGI (Common Gateway Interface).

Идея HTML - пример чрезвычайно удачного решения проблемы построения гипертекстовой системы при помощи специального средства управления отображением. На разработку языка гипертекстовой разметки существенное влияние оказали два фактора: исследование в области интерфейсов гипертекстовых систем и желание обеспечить простой и

быстрый способ создания гипертекстовой базы данных, распределенной на сети. В 1989 году активно обсуждалась проблема интерфейса гипертекстовых систем, т.е. способов отображения гипертекстовой информации и навигации в гипертекстовой сети. Значение гипертекстовой технологии сравнивали со значением книгопечатания. Утверждалось, что лист бумаги и компьютерные средства отображения-воспроизведения серьезно отличаются друг от друга, и поэтому форма представления информации тоже должна отличаться. Наиболее эффективной формой организации гипертекста были признаны контекстные гипертекстовые ссылки, а кроме того было признано деление на ссылки, ассоциированные со всеми документами в целом и отдельными его частями.

Самым простым способом создания любого документа является его печатание в текстовом редакторе.

Опыт создания хорошо размеченных для последующего отображения документов в CERNe был; трудно найти физика, который не пользовался бы системой TeX или LaTeX. Кроме того, к тому времени существовал стандарт языка разметки – Standard Generalised Markup Lanquage (SGML). Следует также принять во внимание, что, согласно своим предложениям, Бернес Ли предполагал объединить в единую систему имеющиеся информационные ресурсы CERN. Обычно гипертекстовые системы имеют специальные программные средства построения гипертекстовых связей. Собственно гипертекстовые ссылки хранятся в специальных форматах или даже составляют специальные файлы. Такой подход хорош для локальной системы, но не для распределенной на множестве различных компьютерных платформ. В HTML гипертекстовые ссылки встроены в тело документа и хранятся как его часть. Часто в системах применяют специальные форматы хранения данных для повышения эффективности доступа. В WWW документы - это обычные ASCII-файлы, которые можно подготовить в любом текстовом редакторе. Таким образом, проблема создания гипертекстовой базы данных была решена чрезвычайно просто. В качестве базы для разработки языка гипертекстовой разметки был выбран SGML.

Следуя академическим традициям, Бернес Ли описал HTML в терминах SGML (как описывают язык программирования в терминах формы Бекуса – Наура). Естественно, что HTML были реализованы все разметки, связанные с выделением параграфов, шрифтов, стилей и тому подобное, так как реализация для NEXT подразумевала графический интерфейс. Важным компонентом языка стало описание встроенных и ассоциированных гипертекстовых ссылок, встроенной графики и обеспечение возможности поиска по ключевым словам. С момента разработки первой версии языка (HTML 1.0) прошло уже несколько лет. За это время произошло довольно серьезное развитие языка. Почти вдвое увеличилось число элементов разметки, оформление документов все

больше приближается к оформлению качественных печатных изданий, развиваются средства описания нетекстовых информационных ресурсов и способы взаимодействия с прикладным программным обеспечением. Совершенствуется механизм разработки типовых стилей. Фактически, в настоящее время HTML развивается в сторону создания стандартного языка разработки интерфейсов как локальных, так и распределенных систем.

Вторым краеугольным камнем WWW стала универсальная форма адресации информационных ресурсов. Universal Resource Identification (URI) представляет собой довольно стройную систему, учитывающую опыт адресации и идентификации e-mail,Gopher WAIS, telnet, ftp и.т.п. Но реально из всего, что описано в URI, для организации баз данных в WWW требуется только Universal Resource Locator (URL). Без наличия этой спецификации вся мощь HTML оказалась бы бесполезной. URL используется в гипертекстовых ссылках и обеспечивает доступ к распределенным ресурсам сети. В URL можно адресовать как другие гипертекстовые документы формата HTML, так и ресурсы e-mail, telnet, ftp, Gopher, WAIS, например. Различные интерфейсные программы по разному осуществляют доступ к этим ресурсам. Одни, как, например, Netscape, сами способны поддерживать взаимодействие по протоколам, отличным от протокола HTTP, базового для WWW, другие, как, например, Chimera, вызывают для этой цели внешние программы. Однако, даже в первом случае, базовой формой предоставления отображаемой информации является HTML, а ссылки на другие ресурсы имеют форму URL.

Следует отметить, что программы обработки электронной почты в формате MIME также имеют возможность отображать документы, представленные в формате HTML. Для этой цели в MIME зарезервирован тип "text/html". Третьим в списке стоит протокол обмена данными в World Wide Web - Hyper Text Transfer Protocol. Данный протокол предназначен для обмена гипертекстовыми документами и учитывает специфику такого обмена. Так, в процессе взаимодействия клиент может получить новый адрес ресурса на сети (relocation), запросить встроенную графику, принять и передать параметры и т.п. Управление в HTTP реализовано в виде ASCLL-команд. Реально разработчик гипертекстовой базы данных сталкивается с элементами протокола только при использовании внешних расчетных программ или при доступе к внешним относительно WWW информационным ресурсам, например, базам данных.

Последняя составляющая технологии WWW - это уже плод работы группы NCSA – спецификации Common Gateway Interface. CGI была специально разработана для расширения возможностей WWW за счет подключения всевозможного внешнего программного обеспечения. Такой подход логично продолжал принцип публичности и простоты разработки и

наращивания возможностей WWW. Если команда CERN предложила простой и быстрый способ разработки баз данных, то NCSA развила этот принцип в разработке программных средств. Надо заметить, что в общедоступной библиотеке CERN были модули, позволяющие программистам подключать свои программы к северу HTTP, но это требовало использования этой библиотеки. Предложенный и описанный в CGI способ подключения не требовал дополнительных библиотек и буквально ошеломлял своей простотой. Сервер взаимодействовал с программами через стандартные потоки ввода/вывода, что упрощает программирование до предела. При реализации CGL чрезвычайно важное место заняли методы доступа, описанные в HTTR. И хотя реально используются только два из них (GET и POST), опыт развития HTML показывает, что сообщество WWW ждет развития и CGL по мере усложнения задач, в которых будет использоваться WWW- технология.

2.2. Как обратиться к файлу в Интернет

Документы, из которых состоит Web, находятся на Интернет-серверах. Чтобы добраться до какого-либо WWW-документа, необходимо набрать в браузере, в строке «адрес», путь к этому документу. Этот путь называется URL (Uniform Resourse Locator — унифицированный указатель ресурса). URL состоит из названия сервера и, если требуется, названия каталога, в котором находится документ, и названия самого документа (т.е. имени файла HTML), разделенных знаком обратной косой черты (обратным слэшем) «/».

Регистр символов, используемых в URL, не имеет никакого значения, т. е. www.gov.uz указывает на абсолютно тот же сервер, что и Www.gov.uz. В то же время, иногда регистр символов, используемых в названиях директорий и файлов на сервере, может иметь значение (это зависит от программного обеспечения, установленного на сервере).

Для передачи разных типов данных были созданы протоколы, базирующиеся на TCP/IP. При работе с гипертекстовыми файлами используется HTTP (Hypertext Transfer Protocol — протокол передачи гипертекста). Пересылка по сети файлов любых типов осуществляется с помощью протокола FTP (File Transfer Protocol — протокол передачи файлов). Используя этот протокол, можно загружать файлы web-странички на сервер. Кроме того, существуют протоколы передачи файлов новостей (news), службы Gopher и т. д.

В принципе, перед адресом документа, который нужно загрузить из Сети, необходимо проставлять тип протокола, по которому этот документ будет передаваться на ваш компьютер, добавляя к нему двоеточие и два обратных слэша (http://, ftp:// и т. д.), но делать это на практике нет необходимости, так как современные браузеры делают это автоматически.

Итак, после того, как набран URL интересующего документа, программа-браузер, используя соответствующий протокол, обращается к

требуемому серверу и запрашивает у него указанный файл. Далее файл передается на ваш компьютер и браузер и, используя команды форматирования и ссылки на объекты, содержащиеся в тексте, генерирует готовый для просмотра документ.

2.3. Основы работы сервера Web

Особая роль в глобальной сети принадлежит Web серверам. Известно, что Web-сервер хранит информацию в виде тестовых файлов, называемых web-страницами. Кроме текста, такие страницы содержат ссылки на другие страницы, ссылки на графические изображения, аудио- и видео информацию, различные объекты ввода данных (поля, кнопки, формы и т.п.). Страницы Web представляют собой как бы связующее звено между объектами различных типов. Эти объекты проектируются с помощью языка HTML (см. рис.1).

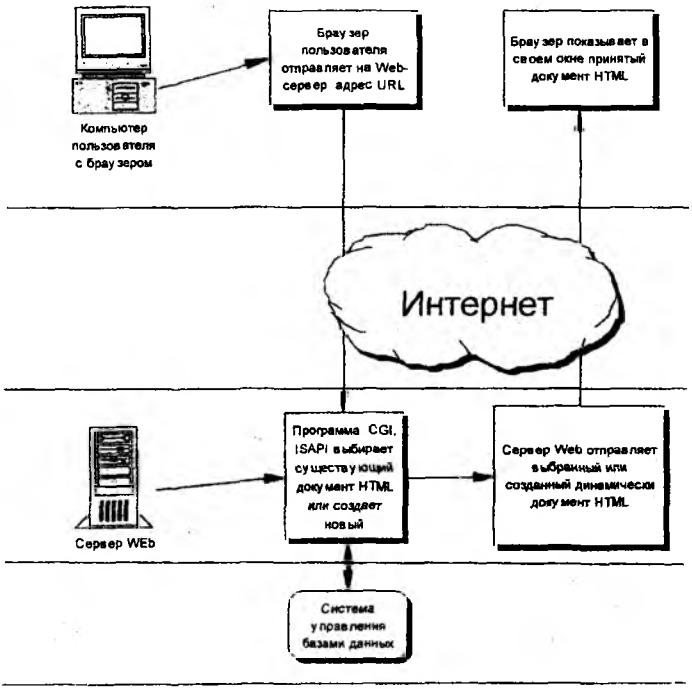


Рис.1.Схема формирования динамических документов HTML с помощью CGI и ISAPI

Различают пассивные и активные серверы Web. Если страницы сервера содержат только статическую информацию – текстовую, мультимедийную, гиперссылки, то такой сервер называется пассивным. Если же страница сервера имеет возможность вступать в диалог с пользователем, то такой сервер будет активным. Естественно, активные

серверы обладают значительными преимуществами по сравнению с пассивными серверами. Они служат основой для создания интерактивных приложений в сети Интернет с базами данных, имеют средства ввода и обработки запросов. Например, Интернет-магазины, поисковые и справочные сайты, сайты с опросами пользователей требуют возможностей диалога с пользователями, а, значит, их серверы должны быть активными. Работа активных серверов предполагает применение специальных программных расширений сервера Web, таких как CGI и ISAPI.

Другой возможный вариант – использование серверных сценариев и технологии активных страниц Active Server Pages (ASP). Однако Web сервер выполняет только часть работы. Он отвечает за получение данных от пользователя и подготовку страниц, отправляемых обратно. Запросы к базам данных и другим активным объектам осуществляются через сервер баз данных или другим серверным приложениям. При этом используются упомянутые средства ASP, CGI, ISAPI.

2.4. Активность компьютера-клиент

Целесообразно разделять работу между клиентом и сервером, чтобы добиться оптимальной производительности в условиях каналов Интернет и ограниченных ресурсов Web. Например, предварительную обработку введенных пользователем данных, которые будут отправлены серверу, имеет смысл выполнять на стороне клиента. Это позволит исключить повторные передачи неправильно заполненных форм. Графическое представление результатов запроса также лучше выполнять на клиентской машине, что позволит сократить объем передаваемых по сети данных. Для реализации клиентской активности используются технологии JavaScript, VBScript, аплеты Java и элементы управления ActiveX.

2.5. Архитектура построения системы

От описания основных компонентов перейдем к архитектуре взаимодействия программного обеспечения в системе World Wide Web. WWW построена по хорошо известной схеме "клиент-сервер" (рис.2).

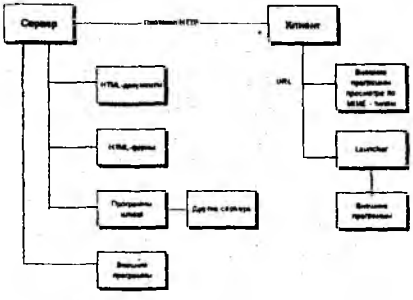


Рис. 2. Схема взаимодействия программного обеспечения в системе WWW

Программа-клиент выполняет функции интерфейса пользователя и обеспечивает доступ практически ко всем информационным ресурсам Интернет. В этом смысле она выходит за обычные рамки работы клиента только с сервером определенного протокола, как это происходит в telnet, например. Довольно широко распространенное мнение, что Mosaic или Netscape, которые, безусловно, являются WWW-клиентами, это просто графический интерфейс в Интернет, является отчасти верным. Однако, как уже было отмечено, базовые компоненты WWW-технологии (HTML и URL) играют при доступе к другим ресурсам Mosaic не последнюю роль, и поэтому мультипротокольные клиенты должны быть отнесены именно к World Wide Web, а не к другим информационным технологиям Интернет. Фактически, клиент - это инспектор HTML. И как типичный интерпретатор клиент в зависимости от команд (разметки) выполняет различные функции. В круг этих функций входит не только размещение текста на экране, но и обмен информацией с сервером по мере анализа полученного HTML текста, что наиболее наглядно происходит при отображении встроенных в текст графических образов. При анализе URL спецификации или по команде сервера клиент запускает дополнительные внешние программы для работы с документами в форматах, отличных от HTML, например, GIF, JPEG, MPEG, Postscript и.т.п. Для запуска клиентом программ независимо от типа документа была разработана программа Luncher, но в последнее время гораздо большее распространение получил механизм согласования запускаемых программ через MIME-типы.

Другую часть программного комплекса WWW составляют сервер протокола HTTP базы данных документов в формате HTML, управляемые сервером, и программное обеспечение, разработанное в стандарте спецификации CGI. До самого последнего времени (до образования Netscape) реально использовалось два HTTP-сервера: сервер CERN и сервер NCSA. Но в настоящее время число базовых серверов расширилось. Появился очень неплохой сервер для MS-Windows и Apachie-сервер для Unix-платформ. Существуют и другие, но два последних можно выделить из соображений доступности использования.

Сервер для Windows - это shareware, но без встроенного самоликвидатора, как в Netscape. Учитывая распространенность персоналок в нашей стране, такое программное обеспечение дает возможность пробовать, что такое WWW. Второй сервер - это ответ на угрозу коммерциализации. Netscape уже не распространяет свой сервер свободно, и появилась вероятность, что NCSA - сервер также будет распространяться на коммерческой основе. В результате был разработан Apachie, который, по словам его авторов, будет свободно распространятся. Он реализует новые дополнения к протоколу HTTP, связанные с защитой от несанкционированного доступа, которые предложены группой по

разработке этого протокола и реализуются практически во всех коммерческих серверах.

База данных HTML - документов - это часть файловой системы, которая содержит текстовые файлы в формате HTML и связанные с ними графику и другие ресурсы. Особое внимание хотелось бы обратить на документы, содержащие элементы экранных форм. Эти документы реально обеспечивают доступ к внешнему программному обеспечению.

Прикладное программное обеспечение, работающее с сервером, можно разделить на программы-шлюзы и прочие. Шлюзы – это программы, обеспечивающее взаимодействие сервера с серверами других протоколов, например ftp, или с распределенными на сети серверами Oracle. Прочие программы – это программы, принимающие данные от сервера и выполняющие какие-либо действия: получение текущей даты, реализация графических ссылок, доступ к локальным базам данных или просто расчеты.

Завершая обсуждение архитектуры, хотелось бы еще раз подчеркнуть, что ее компоненты существуют практически для всех типов компьютерных платформ и свободно доступны в сети. Любой, кто имеет доступ в Интернет, может создать свой WWW-сервер, или, по крайне мере, посмотреть информацию с других серверов.

2.6. Обзор основных Web-технологий

2.6.1. HTML

С появлением Интернет проблема межплатформной несовместимости файловых форматов (UNIX, Windows, OS/2 и др.) стала ощущаться особенно остро: ведь тем, кто размещает информацию в Сети, хочется, чтобы она была доступна как можно более широкому кругу пользователей без особенных затрат. Очевидно, что самый простой путь решения — разработка общепринятого формата файлов, под который и будут «подстраиваться» вновь создаваемые приложения. В WWW таким стандартом стал HTML. Основа WWW — файлы в формате HTML (Hyper Text Markup Language — язык разметки гипертекста), или гипертекстовые страницы. Гипертекст — это легкая в использовании и чрезвычайно мощная система связанных слов и фраз, позволяющая легко перемещаться по специальным образом организованным страницам. Она связывает фразу или слово одной страницы с любой другой страницей, абзацем, фразой или словом.

Язык гипертекстовой разметки HTML был предложен Тимом Бернерсом-Ли в 1989 году в качестве одного из компонентов технологии разработки распределенной гипертекстовой системы World Wide Web.

Разработчики HTML пытались решить две задачи:

дать дизайнерам гипертекстовых баз данных простое средство создания документов;

сделать это средство достаточно мощным, чтобы отразить имевшееся на тот момент представление об интерфейсе пользователя гипертекстовых баз данных.

Первая задача была решена за счет выбора таговой модели описания документа. Такая модель широко применяется в системах подготовки документов для печати. Примером такой системы является хорошо известный язык разметки научных документов TeX, предложенный Американским математическим обществом, и программы его интерпретации.

К моменту создания HTML существовал стандарт языка разметки печатных документов – Standard Generalised Markup Lanquage, который и был взят в качестве основы HTML. Предполагалось, что такое решение поможет использовать существующее программное обеспечение для интерпретации нового языка. Однако, будучи доступным широкому кругу пользователей Интернет, HTML зажил своей собственной жизнью. Вероятно, многие администраторы баз данных WWW и разработчики программного обеспечения для этой системы имеют довольно смутное представление о стандартном языке разметки SGML.

Вторым важным моментом, повлиявшим на судьбу HTML, стал выбор в качестве элемента гипертекстовой базы данных обычного текстового файла, который хранится средствами файловой системы операционной среды компьютера. Выбор был сделан под влиянием следующих факторов:

Такой файл можно создать в любом текстовом редакторе на любой аппаратурной платформе в среде любой операционной системе;

К моменту разработки сетевых информационных систем – Z39.50, в котором в качестве единицы хранения указывался простой текстовый файл в кодировке LATIN1, что соответствует US ASCII.

Таким образом, гипертекстовая база данных в концепции WWW - это набор текстовых файлов, написанных на языке HTML, который определяет форму представления информации (разметки) и структуры связей этих файлов (гипертекстовой ссылки).

Такой подход предполагает наличие еще одной компоненты технологии – интерпретатора языка. В World Wide Web функции интерпретатора разделены между сервером гипертекстовой базы данных и интерфейсом пользователя. Сервер, кроме доступа к документам и обработки гипертекстовых ссылок, осуществляет также препроцессорную обработку документов, в то время как интерфейс пользователя осуществляет интерпретацию конструкций языка, связанных с представлением информации.

К настоящему времени известна уже четвертая версия языка – HTML 4.0, которая находится в стадии развития. Если первая версия языка (HTML 1.0) была направлена на представление языка как такового, где

описание его возможностей носило скорее рекомендательный характер, вторая версия языка (HTML 2.0) фиксировала практику использования конструкции языка, версия ++ (HTML++) представляла новые возможности, расширяя набор элементов HTML в сторону отображения научной информации и таблиц, а также улучшения стиля компоновки изображений и текста, то версия 4.0 призвана упорядочить все нововведения и согласовать их с существующей практикой. Кроме этого в версии 4.0 снова делается попытка формализации интерфейса пользователя гипертекстовой распределенной системы.

Файлы HTML состоят из команд форматирования, текста и ссылок на другие файлы или объекты (графика, звуки, программы). Программа для просмотра HTML-документов (браузер) интерпретирует код HTML, содержащийся в файле, и согласно командам форматирования собирает готовую Web-страничку.

Метка языка HTML, или тег (англ. tag), или дескриптор, является основным средством разметки документа. При написании теги отделяются от остального текста угловыми скобками: «<» и «>». Регистр символов при этом также не имеет никакого значения, но хорошим тоном считается набор тегов в верхнем регистре, т. е. заглавными буквами. Это помогает визуально отделить теги от основного текста при последующем редактировании.

Существуют два вида тегов: требующие закрытия (т. е. ограничения действия) и не требующие закрытия. Например, дескриптор <I>, изменяющий начертание шрифта на наклонное, является тегом, требующим закрытия, т. к., если он не будет закрыт, то весь текст, следующий после него, будет наклонным.

Закрывающим элементом всегда служит тот же самый тег, но со знаком «/» после первой угловой скобки.

Рассмотрим использование нескольких основных тегов для создания web-страницы. Технология работы при этом следующая. Текст html-файла набирается в простом редакторе типа notepad (блокнот) и записывается с расширением htm. Просмотр осуществляется при помощи браузера, например, Интернет Explorer.

Описание тэгов

< html> ... </html> - метка < html> открывает HTML – документ, метка </html> завершает HTML – документ.

< head> ... </head> - эта пара меток указывает на начало и конец заголовка документа. Помимо наименования документа, в этот раздел может включаться множество служебной информации.

< title> ... </title> - все, что находится между этими метками, толкуется браузером как название документа. Netscape Navigator, например, показывает название текущего документа в заголовке окна и

печатает его в левом верхнем углу каждой страницы при выводе на принтер. Рекомендуется название не длиннее 64 символов.

< body> ... < /body> - эта пара меток указывает на начало и конец тела HTML-документа, каковое тело, собственно, и определяет содержание документа.

< H1> ... < /H1> - < H6> ... < /H6> - метки вида <Hi> (где i – цифра от 1 до 6) описывают заголовки шести различных уровней. Заголовок первого уровня – самый крупный, шестого уровня, естественно – самый мелкий.

<P> ... </P> - такая пара меток описывает абзац. Все, что заключено между ними, воспринимается как один абзац.

Метки <Hi> и <P> могут содержать дополнительный атрибут ALIGN, например:

<H1 ALIGN=CENTER> - выравнивание заголовка по центру</H1>.

*
* - эта метка используется, если необходимо перейти на новую строку, не прерывая абзаца. Очень удобно при публикации стихов.

<HR> - эта метка описывает горизонтальную линию. Она может дополнительно включать атрибуты SIZE (определяет толщину линии в пикселах) и/или WIDTH (определяет размах линии в процентах от ширины экрана).

<B> ... </B> - все, что находится между этими метками, будет написано жирным шрифтом.

<I> ... </I> - текст между этими метками будет написан наклонным шрифтом.

<TT> ...</TT> - текст между этими метками будет написан шрифтом, имитирующим пишущую машинку.

<EM> ... </EM> - от английского emphasis – акцент.

<STRONG> ... </STRONG> - от английского strong emphasis – сильный акцент.

<CODE> ... </CODE> - рекомендуется использовать для фрагментов исходных текстов.

<SAMP> ... </SAMP> - от английского sample – образец. Рекомендуется использовать для демонстрации образцов сообщений, выводимых на экран программами.

<KBD> ... </KBD> - от английского keyboard – клавиатура. Рекомендуется использовать для указания того, что нужно ввести с клавиатуры.

<VAR> ... </VAR> - от английского variable – переменная. Рекомендуется использовать для написания имен переменных.

<UL> ... </UL> - текст, расположенный между этими метками, воспринимается как ненумерованный список. Каждый новый элемент списка следует начинать с метки *<LI>*.

<OL> ... </OL> - нумерованные списки устроены точно так же, как ненумерованные, только вместо символов, выделяющих новый элемент, используются цифры.

<DL> ... </DL> - список определений. Вместо меток <LI> в списках определений используются метки *<DT>* (от английского definition term – определяемый термин) и *<DD>* (от английского definition – определение определения). Если определяемые термины достаточно коротки, можно использовать модифицированную открывающую метку *<DL COMPACT>*.

В HTML переход от одного фрагмента текста к другому задается с помощью метки вида:

<A HREF="[адрес перехода]">выделенный фрагмент текста</A>

Если в адресе перехода не указан каталог, переход будет выполнен внутри текущего каталога. Если в адресе перехода не указан сервер, переход будет выполнен на текущем сервере.

Можно задать переход и к определенному месту внутри документа. Для этого необходимо создать в документе, к которому будет задан переход, некоторую опорную точку, или анкер.

Допустим, что необходимо осуществить переход из одного файла к определенному месту в другом файле. Сначала необходимо создать анкер в другом файле:

<A NAME="AAA">текст внутри документа</A>

Затем в исходном файле можно определить переход на этот анкер:

<A HREF="[имя другого файла]#AAA">переход к анкеру AAA</A>

Переход к этому анкеру можно определить и внутри другого документа, нужно включить в него фрагмент:

<A HREF="#AAA">переход к анкеру AAA</A>

Возможны ссылки и на другие виды ресурсов:

<A HREF=ftp://server/directory/file.ext>выгрузить файл</A>

Такая ссылка запустит протокол передачи файлов и начнет выгрузку файла file.ext, находящегося в каталоге directory на сервере server, на локальный диск пользователя.

<IMG SRC = "[имя файла]"> - для включения в документ изображения, записанного в каком-либо файле.

Эта метка может также включать атрибут ALT="[текст]":

<IMG SCR="[имя файла]" ALT="[текст]"

Браузер покажет на экране [текст] и начнет загружать на его место содержимое из файла [имя файла].

META-инструкция – это способ определить некоторую переменную путем указания ее имени (атрибут NAME) и значения (атрибут CONTENT). Заголовок HTML-документа может включать неограниченное количество META-инструкций.

<META NAME="[переменная]" CONTENT="[текст]">

META-инструкция определяет данную переменную, содержащую [текст].

Другая группа META-инструкций определяет эквиваленты команд протокола передачи гипертекстов.

<META HTTP-EQUIV="Content-type" CONTENT="text/html; charset=windows-1251">

Эта META-инструкция дает браузеру указание интерпретировать загружаемый документ как содержащий HTML-текст в кодировке Windows/1251.

<META HTTP-EQUIV="Refresh" CONTENT="[время]; URL=[документ]">

Такая META-инструкция дает браузеру указание: "Если через [время] секунд после завершения загрузки этого документа пользователь не перейдет к другому документу, начать загрузку ресурса [документ]".

<TABLE> ... </TABLE> - таблица начинается с метки <TABLE> и заканчивается меткой </TABLE>. Метка <TABLE> может включать несколько атрибутов:

ALIGN – устанавливает расположение таблицы по отношению к полям документа. Допустимые значения: ALIGN=LEFT (выравнивание влево), ALIGN=CENTER (выравнивание по центру), ALIGN=RIGHT (выравнивание вправо).

WIDTH – ширина таблицы. Ее можно записать в пикселах (например, WIDTH=400) или в процентах от ширины страницы (например, WIDTH=80%).

BORDER – устанавливает ширину внешней рамки таблицы и ячеек в пикселах (например, BORDER=4). Если атрибут не установлен, таблица показывается без рамки.

CELLSPACING – устанавливает расстояние между рамками ячеек таблицы в пикселах (например, CELLSPACING=2).

CELLPADDING – устанавливает расстояние между рамкой ячейки и текстом в пикселах (например, CELLPADDING=10).

Таблица может иметь заголовок (<CAPTION> …</CAPTION>), хотя он не является обязательным. Метка <CAPTION> может включать атрибут ALIGN. Допустимые значения: <CAPTION ALIGN=TOP> (заголовок помещается над таблицей) и <CAPTION ALIGN=BOTTOM> (заголовок помещается под таблицей).

Каждая строка начинается с метки <TR> и заканчивается меткой </TR>. Метка <TR> может включать следующие атрибуты:

ALIGN – устанавливает выравнивание текста в ячейках строки;

VALIGN – устанавливает вертикальное выравнивание текста в ячейках строки. Допустимые значения – TOP; MIDDLE; BOTTOM.

Каждая ячейка таблицы начинается с метки <TD> и заканчивается меткой </TD>. Метка <TD> может включать атрибуты:

NOWRAP– содержимое ячейки должно быть показано в одну строку;

COLSPAN – устанавливает "размах" ячейки по горизонтали. Например, COLSPAN=3 означает, что ячейка простирается на три колонки;

ROWSPAN – устанавливает "размах" ячейки по вертикали;

ALIGN – устанавливает выравнивание текста в ячейке;

VALIGN – устанавливает вертикальное выравнивание текста в ячейке;

WIDTH – устанавливает ширину ячейки в пикселах;

HEIGHT – устанавливает высоту ячейки в пикселах.

Если ячейка таблицы пуста, вокруг нее не рисуется рамка. Если ячейка пуста, а рамка нужна, в ячейку можно ввести символьный объект   (non-breaking space – неразрывающий пробел).

<FORM> ... </FORM> - форма открывается меткой <FORM> и заканчивается меткой </FORM>.

Метка <FORM> может содержать три атрибута, один из которых является обязательным:

ACTION – обязательный атрибут. Определяет, где находится обработчик формы.

METHOD – определяет, каким образом (с помощью какого метода протокола передачи гипертекстов) данные из формы будут переданы обработчику. Допустимые значения: METHOD=POST и METHOD=GET. Если значение атрибута не установлено, по умолчанию предполагается METHOD=GET.

ENCTYPE – определяет, каким образом данные из формы будут закодированы для передачи обработчику. Если значение атрибута не установлено, по умолчанию предполагается ENCTYPE=application/x-www-form-urlencoded.

<INPUT TYPE=submit> - орган управления для запуска процесса передачи данных из формы обработчику. Браузер нарисует на экране кнопку с надписью submit, при нажатии на которую все имеющиеся в форме данные будут переданы обработчику, определенному в метке <FORM>.

Надпись на кнопке можно задать такую, какая нравится, путем введения атрибута VALUE="[надпись]", например:

<INPUT TYPE=submit VALUE="Поехали!">

Надпись, нанесенную на кнопку, можно при необходимости передать обработчику путем введения в определение кнопки атрибута NAME=[имя], например:

<INPUT TYPE=submit NAME=button VALUE="Поехали!">

При нажатии на такую кнопку обработчик вместе со всеми остальными данными получит и переменную button со значением.

Каждый элемент <INPUT> должен включать атрибут NAME=[имя], определяющий имя элемента (и имя переменной, которая будет передана обработчику). Имя должно задаваться только латинскими буквами. Большинство элементов <INPUT> должно включать атрибут VALUE="[значение]", определяющий значение, которое будет передано обработчику под этим именем. Для элементов <INPUT TYPE=text> и <INPUT TYPE=password> этот атрибут не обязателен, т.к. значение соответствующей переменной может вводиться пользователем с клавиатуры.

<SELECT> ... </SELECT> - меню, которое могут содержать формы.

Меню <SELECT> из n элементов выглядит так:

<SELECT NAME="[имя]">
<OPTION VALUE="[значение 1]">[текст 1]
<OPTION VALUE="[значение n]">[текст n] </SELECT>

Метка <SELECT> может содержать атрибут MULTIPLE, присутствие которого показывает, что из меню можно выбрать несколько элементов (<SELECT MULTIPLE>).

Метка <OPTION> определяет элемент меню. Обязательный атрибут VALUE устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Метка <OPTION> может включать атрибут checked, показывающий, что данный элемент отмечен по умолчанию.

<TEXTAREA> ... </TEXTAREA> - поля для ввода текста.

<TEXTAREA NAME=address ROWS=5 COLS=50>
А здесь – Ваш адрес… </TEXTAREA>

Все атрибуты обязательны. Атрибут NAME определяет имя, под которым содержимое окна будет передано обработчику (в примере address). Атрибут ROWS устанавливает высоту окна в строках (в примере – 5). Атрибут COLS устанавливает ширину окна в символах (в примере– 50).

Текст, размещенный между метками <TEXTAREA> и </TEXTAREA>, представляет собой содержимое окна по умолчанию.

2.6.2. JAVA

Фирма Java Soft, оперативное подразделение фирмы Sun Microsystems, потратила много лет на разработку мощного языка программирования для 90-х годов и будущего столетия. Им стал JAVA – самый надежный, простой в использовании и гибкий из всех доступных сегодня языков. Он включает все лучшее из того, что свойственно его предшественникам, например, Си и Си ++, позволяет писать мощные приложения и обладает такими ценными характеристиками, как встроенные возможности мультимедиа, которые обеспечивают небывалую легкость создания мультимедийных продуктов. При этом он устраняет то, что неудобно было в Си и Си++, например, множественное наследование, перегрузку операций и указатели.

Самые лучшие черты JAVA – это его объектная ориентация и независимость от платформы. *Объектно-ориентированное программирование* (ООП) обещает сделать возможным повторное использование кода программ. Больше не нужно разрабатывать отдельные приложения для разных платформ. С помощью JAVA можно создать одно приложение, которое сразу будет работать на нескольких платформах Windows, Unix и Macintosh.

Учитывая, что независимость этого языка от платформы дает крупным корпорациям возможность экономить на одном проекте по нескольку миллионов долларов, не удивительно, что слово «JAVA» стало самым модным в индустрии программного обеспечения и технологий Интернет. Язык JAVA захватил воображение не только технических специалистов. Способы его использования изучают компании из всех мыслимых секторов бизнеса – от финансов до транспорта.

Независимость JAVA от платформы позволяет писать мощные приложения для операционных систем.

JAVA предоставляет полный контроль за разработкой программисту, независимо от того, обладает ли он ограниченными техническими навыками или глубоким знанием компьютеров. Изучить JAVA, по сравнению с другими языками программирования, намного легче.

JAVA создан на основе двух языков Си и Си ++. Кроме того, JAVA позаимствовал из Objective C расширения, которые делают возможным чрезвычайно динамическое разрешение методов.

JAVA обладает высокой переносимостью, т.е. имеет характеристики, которые облегчают его использование на разнородных платформах. JAVA обладает готовностью и совместимостью для работы в сети, т.е. готов к использованию в сложных сетях и включает непосредственную поддержку таких распространенных сетевых протоколов, как FTP и HTTP, что обеспечивает совместимость при работе в сети.

JAVA избавляется от указателей, автоматически управляет памятью и даже включает процедуру сборки мусора, которая выполняется в фоновом режиме.

JAVA – самая безопасная среда программирования, которую только можно найти. Этот язык устраняет лазейки в системе безопасности, что делает его идеальным языком программирования для Web. Например, система динамической верификации кода проверяет программы при их компиляции и выполнении на JAVA и гарантирует, что они не содержат злонамеренных команд.

Среда JAVA времени выполнения обеспечивает жесткие правила выполнения программ, запущенных с удаленных машин. Такие программы не могут обращаться к локальной сети, получать доступ к файлам на локальной системе и запускать на пользовательской системе другие программы.

Создание языка программирования JAVA – результат согласованных усилий группы прогрессивных разработчиков. Сегодня JAVA – это самая горячая тема в Интернет. JAVA предназначен для распределенных сред и открытых систем. Это средство, которое позволит создавать для Web полностью интерактивные мультимедиа-презентации.

JAVA и интерпретируется, и компилируется, поэтому можно создавать программу на JAVA, используя простую структуру языка высокого уровня.

Для программирования в 90-х годах JAVA означает то же, что Си++ для программирования в 80-х, - гигантский скачок вперед. JAVA представляет собой следующий шаг в логической последовательности развития языков программирования и прочно опирается на самый популярный язык своего времени – Си++. JAVA дает не только вид и ощущение Си++, но и предлагает мощь языка, созданного специально для объектно-ориентированного программирования (ООП). Но в отличие от Си++, JAVA полностью разрывает связи с процедурным программированием и заставляет следовать надежным принципам ООП.

JAVA осуществляет автоматическую проверку граничных условий, благодаря чему устраняются многие трудности, с которыми приходится сталкиваться, работая на Си и Си++. В JAVA нет указателей, а управление памятью выполняется автоматически. Теперь нельзя сослаться на неверную область памяти и вызвать сбой программы, а также сделать ошибку при освобождении памяти и вызвать медленную ее утечку, что в конце концов приведет к исчерпанию памяти в системе.

В JAVA массивы инкапсулированы в структуре класса, и это также облегчает встроенную в него проверку граничных условий. Объекты легко использовать повторно, обновлять и поддерживать. Быстро и без особых усилий можно решить следующие задачи:

- Точно определить необходимую входную и выходную информацию для объекта.
- Найти зависимости между переменными.
- Локализовать последствия изменений.
- Внести изменения в случае необходимости.
- Создать подклассы на основе исходного объекта.

Поскольку процедуры освобождения памяти и общей очистки системы работают в фоновом режиме, программы на JAVA выполняются быстрее и эффективнее.

JAVA придает понятию библиотеки новый оттенок, используя для описания набора связанных классов термин *пакет*.

Язык программирования JAVA определяет четыре уровня управления доступом:

- Частные методы и переменные. Доступны для объектов текущего класса.

- Защищенные методы и переменные. Доступны для объектов текущего класса и его подклассов;
- Дружественные методы и переменные. Используются по умолчанию: доступны для всех классов текущего пакета;
- Публичные методы и переменные. Доступны для всех классов из любого пакета.

Основные характеристики JAVA:

- Независимость от архитектуры.
- Работа в распределенных средах.
- Динамическое исполнение.
- Интерпретация и компиляция.
- Множественные потоки выполнения.
- Ориентация на сетевое использование.
- Объектная ориентация.
- Переносимость.
- Устойчивость.
- Безопасность.

JAVA – апплеты

Апплет – это маленькая программа, которая выполняется внутри другой, более крупной принимающей (host) программы. Апплет имеет цель расширить функциональность принимающей программы и не может существовать вне ее. В случае с JAVA апплеты расширяют функциональность браузера Web. Браузер может отобразить текст, но, если добавить соответствующий апплет, то текст запрыгает по экрану как шарик от пинг-понга. Это все, что реально имеется в виду, когда идет речь об апплете – программе, которая выполняется внутри браузера.

Сосредоточенность JAVA на Сети заключена в пакетах java.net и java.applet (библиотеках классов). Пакет java.applet обеспечивает связь с Web через класс Applet – класс, который предоставляет приложению возможность появляться посреди документа HTML, когда документ просматривается браузером Web.

Революционной силой JAVA делает концепция размещения программ (апплетов) в документе HTML. Просматриваемый документ Web весьма статичен. Текст просто сидит на странице и ждет, когда пользователь его просмотрит. Апплеты же JAVA являются активными элементами. Страница Web с апплетами может изменяться в соответствии с любой из широкого диапазона побудительных причин. Причина для изменения может исходить непосредственно от пользователя, от другого апплета или процесса на той же самой рабочей станции или от какого-то процесса в Сети. Она может прийти из любого места и в образе невидимом и/или недешифруемом для пользователя.

Таким образом, вооружившись апплетами, можно создать страницу Web, которая ведет себя в значительной степени как пользовательский

интерфейс традиционного приложения, а не как обычная старая страница HTML.

Внутренне JAVA поддерживает два отдельных раздела памяти: один раздел, который интерпретатор JAVA использует для себя, и второй – для удовлетворения нужд апплетов. Апплеты не могут через сам JAVA получить доступ к памяти из раздела, предназначенного для JAVA. Это является результатом внутренних средств безопасности и конструкции JAVA, в которой отсутствуют указатели. Обеспечить подобный вид защиты в языках, которые поддерживают указатели, почти невозможно.

Все операционные системы, на которых выполняется JAVA, обеспечивают какие-то системные средства, автоматически защищающие одно приложение, в данном случае апплет JAVA, от воздействия любого другого приложения.

Большая часть риска, связанного с JAVA, заключена в способности апплетов выходить из JAVA «наружу» посредством родных методов и вызовов родных исполняемых модулей. JAVA предоставляет возможность апплетам обращаться к динамически связываемым библиотекам и исполняемым модулям, которые, как предполагается, существуют на рабочей станции. Насколько это касается JAVA, то родные методы и исполняемые модули совершенно небезопасны – их можно написать на любом языке, любым человеком, для любых целей как вредоносных, так и нет. Но, хотя JAVA и не защищает от злокачественных родных методов, но он и не оказывает писателям вирусов в родных методах никакой помощи.

Мир JAVA состоит из миллионов маленьких апплетов, созданных тысячами независимых разработчиков. Благодаря своей распределенной среде JAVA обеспечивает очень большое пространство имен для апплетов. Ограничение имен апплетов соглашением DOS «восемь точка три» дало бы приблизительно 30 в восьмой степени (26 букв плюс четыре допустимых символа пунктуации) или более двухсот миллиардов возможных уникальных имен апплетов. Такое ограничение было бы приемлемым, если бы пользователи не возражали называть свои апплеты RRRRXXHY или FRTYZZVC. Реальная проблема состоит в том, что из двухсот миллиардов возможных имен только очень немногие являются описательными. Это только те имеющие смысл слова английского (или китайского, или русского) языка, которые помещаются в восемь символов.

Не пользователь, а браузер загружает, выполняет, останавливает и выгружает апплеты. Апплет выполняется тогда, когда отведенное ему в документе HTML пространство видимо, и заканчивает выполняться тогда, когда по какой-либо причине это пространство перестает быть видимым. До тех пор, пока апплет не создается и не выполняется в своем собственном потоке, это верно. Дело в том, что простые апплеты выполняются в потоке, который создал и контролирует браузер. Когда апплеты JAVA создают свои потоки, то ответ на вопрос «когда

выполняется апплет?» усложняется. Технически, поскольку потоки апплетов теперь вне контроля браузера, то «железного» правила, когда выполняется апплет, нет. Теоретически, браузер ожидает, что хорошо написанные апплеты будут придерживаться базового правила работать, когда они видны на экране, и прекращать работу после ухода с экрана. Однако, реализует ли апплет эту теорию или нет, полностью зависит от программиста апплета.

JAVA-скрипты

На данный момент существуют два основных языка написания сценариев на Web: JavaScript и VBScript. Комитет ECMA (European Computer Manufactures Association – европейская ассоциация производителей компьютеров), в состав которой входят представители компаний Netscape, Microsoft и других поставщиков, утвердил стандартный вариант языка JavaScript. Реализация JScript компании Microsoft в Интернет Explorer 4.0 полностью совместима с новым стандартом.

Для создания Web-страниц в Интернет, которые должны быть выставлены на всеобщее обозрение, JavaScript предоставляет наибольшие возможности, и поддерживается в настоящее время браузерами Netscape и Microsoft.

Кроме того, синтаксис для управления программным потоком в JavaScript очень сходен с синтаксисом в таких языках, как Java, C++, которые знакомы многим авторам Web-страниц.

2.6.3. CGI (Common Gateway Interface)

CGI - Common Gateway Interface является стандартом интерфейса (связи) внешней прикладной программы с информационным сервером типа HTTP, Web сервер. Сегодня такие возможности, как гостевая книга, поиск по серверу, форма для отправки сообщений - неотъемлемый атрибут практически любого серьезного сайта. К сожалению, изучение HTML-исходников страниц конкурентов ничего, кроме ссылок на некий "cgi-bin", не дает, да еще в телеконференциях иногда встречается упоминание о каких-то cgi-скриптах.

Для начала надо разобраться с понятиями. CGI-скрипт - это программа, которая выполняется на Web-сервере по запросу клиента (то есть посетителя Web-сайта). Программа эта принципиально ничем не отличается от обычных программ, которые установлены на компьютере - будь то MS Word или игра.

CGI - это не язык программирования, на котором написан скрипт, а Common Gateway Interface - специальный интерфейс, с помощью которого и происходит запуск скрипта и взаимодействие с ним.

Правда, есть один довольно неприятный момент. На сервере, где находится сайт, должно быть разрешено выполнение cgi-скриптов. Дело в том, что скрипт, как и любая другая программа, может выполнять

системные команды на сервере, что представляет потенциальную угрозу безопасности. Так что если сайт размещен на бесплатном сервере, например, Chat.Ru, то запускать скрипты невозможно. Впрочем, некоторые бесплатные сервера допускают использование CGI. На платном сервере, как правило, использование cgi-скриптов разрешено.

Обычно гипертекстовые документы, извлекаемые из WWW серверов, содержат статические данные. С помощью CGI можно создавать CGI-программы, называемые шлюзами, которые во взаимодействии с такими прикладными системами, как система управления базой данных, электронная таблица, деловая графика и др., смогут выдать на экран пользователя динамическую информацию.

Программа-шлюз запускается WWW сервером в реальном масштабе времени. WWW сервер обеспечивает передачу запроса пользователя шлюзу, а она в свою очередь, используя средства прикладной системы, возвращает результат обработки запроса на экран пользователя. Программа-шлюз может быть закодирована на языках C/C++, Fortran, Perl, TCL, Unix Schell, Visual Basic, Apple Script. Как выполнимый модуль, она записывается в поддиректорий с именем cgi-bin WWW сервера.

Оригинал описания CGI интерфейса - инструмента связи программы-шлюз с WWW сервером находится в узле wist.ifmo.ru .

Для передачи данных об информационном запросе от сервера к шлюзу, сервер использует командную строку и переменные окружения. Эти переменные окружения устанавливаются в тот момент, когда сервер выполняет программу шлюза.

Информация шлюзам передается в следующей форме: имя=значение&имя1=значение1&.., где имя - имя переменной (из оператора FORM, например), и значение - ее реальное значение. В зависимости от метода, который используется для запроса, эта строка появляется или как часть URL (в случае метода GET), или как содержимое HTTP запроса (метод POST). В последнем случае, эта информация будет послана шлюзу в стандартный поток ввода.

Как работает CGI-скрипт? Посетитель страницы заполняет поля формы, например, для записи в гостевую книгу. После этого он нажимает кнопку "Submit", которая и запускает cgi-скрипт. Скрипт выполняет запрограммированные действия - в данном случае считывает данные из формы и пишет их в файл гостевой книги - и посылает в браузер посетителя обычный HTML-код, например, сообщение "Спасибо, что вы оставили запись в гостевой книге".

Преимуществ CGI-скриптов перед JavaScript и Java три, и они весьма значительны:

- так как программа выполняется сервером, нет никакого значения, какой у посетителя браузер - древний Lynx или новейший Интернет Explorer. Нет никаких сообщений об ошибках;

- cgi-скрипты позволяют реализовать гораздо более широкий набор функций;
- код cgi-скрипта закрыт для конкурентов.

На каком же языке может быть написана CGI-программа? Ответ вас приятно удивит: практически на любом. Главное, чтобы сервер мог выполнить эту программу, то есть на сервере должен быть установлен компилятор или интерпретатор соответствующего языка программирования. Для систем на базе Unix это обычно C/C++, Perl, Shell; для серверов под управлением Windows NT - те же Perl, C/C++ и любая Windows-система программирования, поддерживающая написание cgi-приложений, например, Visual Basic или Delphi.

Для того, чтобы наладить работу скриптов на сайте, знания языка программирования особенно не нужно. В Сети лежит очень большое количество абсолютно бесплатных скриптов на любой вкус - от гостевых книг до сложных баз данных. Все, что нужно - хотя бы начальные знания английского языка. Скачать любой скрипт можно с сайта CGI-Resources.Com. Это специализированный каталог, содержащий ссылки на тысячи скриптов на самых разных языках программирования.

2.6.4. SSI (Server Side Includes)

Основным, простейшим, но в то же время чрезвычайно мощным инструментом поддержки больших наборов документов является SSI (Server-Side Includes - включения на стороне сервера).

SSI – технология, тесно переплетенная с CGI. На основе макроязыка, очень напоминающего C, SSI позволяет реализовать такие возможности, как вывод в документе того или иного текста в зависимости от определенных условий или согласно заданному алгоритму, формировать файл HTML из динамически изменяющихся фрагментов, из заранее определенных отрывков или встраивать результат работы некоторого CGI сценария или программы в какой-либо его участок. Достоинства и недостатки SSI аналогичны достоинствам и недостаткам CGI.

Директивы SSI включаются в HTML документ в виде комментариев (это не мешает использовать обычные комментарии).

Элементы SSI:

config - контролирует различные аспекты сканирования.

Допустимые атрибуты:

errmsg - устанавливает сообщение, выводящееся при возникновении ошибки; в большинстве случаев целесообразно установить в пустую строку;

sizefmt - устанавливает формат, в котором будет выводиться размер файла. Формат соответствует передаваемому библиотечной функции *strftime;*

timefmt - устанавливает формат, в котором будет выводиться дата.

echo - выводит значение установленной переменной SSI.

Допустимый атрибут - *var*.

fsize - выводит размер файла в определенном с помощью *sizefmt* формате.

Допустимые атрибуты:

file - определяет путь к файлу, относительно сканируемого документа;

virtual - определяет стандартный кодированный URL, относительно сканируемого документа, или, при наличии в начале слеша (/) - относительно корня документов узла.

flastmod - выводит дату последней модификации файла в определенном с помощью timefmt формате.

Атрибуты аналогичны атрибутам *fsize*.

include - включает текст другого документа или файла в сканируемый файл. К включаемому файлу применяются все установленные правила ограничения доступа. Если для каталога, из которого включается файл, установлена опция *IncludesNOEXEC*, и включение данного документа привело бы к запуску программы, то документ не включается, и выводится сообщение об ошибке. CGI сценарии вызываются, как обычно с помощью URL, который может содержать кодированную строку запроса (query string).

Положение файла указывается с помощью атрибутов:

file - указывает путь, относительно сканируемого документа; путь не может содержать ../ и не может быть абсолютным путем; всегда предпочтительнее использовать атрибут *virtual*;

virtual - содержит кодированный URL, относительный или абсолютный; URL не может содержать имя протокола или имя хоста, и может содержать строку запроса.

printenv - выводит содержимое переменных окружения. Вызывается без параметров.

set - устанавливает значение переменной. Ее атрибутами являются *var*, определяющий имя переменной, и *value*, определяющий ее значение.

В дополнение к стандартным окружения CGI, модуль SSI делает доступными для директив и условий, а также для вызываемых через SSI сценариев следующие переменные:

DATE_GMT - текущее время по Гринвичу;

DATE_LOCAL - текущее локальное (для сервера) время;

DOCUMENT_NAME - имя файла (без каталогов) документа, запрошенного пользователем;

DOCUMENT_URI - декодированный URL запрошенного пользователем документа;

LAST_MODIFIED - дата последней модификации документа, запрошенного пользователем. То есть во вложенном SSI эта переменная будет содержать имя "главного" документа, а не вложенного.

Первым распространенным применением SSI является внедрение в документ некоего динамического куска разметки. Хрестоматийными примерами могут служить счетчики посещений, цитаты или баннеры рекламных сетей.

Программа или сценарий CGI, вставляемая с помощью SSI, ничем не отличается от стандартной CGI программы, за исключением того, что должен выдаваться не целый документ, а только кусок разметки.

Вторым распространенным применением SSI является формирование страницы из шаблона. В простейшем случае - это документ, в начало и конец которого вставляются верхний и нижний колонтитулы.

Только, используя SSI, можно в считанные минуты полностью изменить внешний вид или обновить систему навигации на узле, имеющем сотни или тысячи документов, и так, что это пройдет безболезненно для пользователей узла, т.к. вам только потребуется заменить несколько файлов, а остальное за вас сделает сервер.

2.6.5. CSS (Cascading Style Sheet)

Зачастую у web-дизайнера возникает необходимость применить в процессе создания html-документа сложное форматирование – от абзаца к абзацу менять шрифт, расположение текста, его цвет, формировать различные таблицы данных. Можно решить эту проблему с помощью стандартных средств HTML: описывать каждый абзац отдельным набором команд, но в этом случае итоговый документ будет иметь большой размер, да и само создание кода становится весьма трудоемкой работой.

Можно пойти другим путем: подключить к странице внешний файл, выполненный в стандарте CSS – Cascading Style Sheets (каскадные таблицы стилей), в котором с помощью специального макроязыка один раз жестко задать форматирование страницы. Другими словами, файл CSS выполняет роль некоего шаблона, применяемого для форматирования текста, таблиц и иных элементов в документе HTML. Есть возможность подключать один и тот же физический файл CSS к различным web-страницам сайта. CSS можно использовать практически на любом сервере без каких-либо ограничений.

Крупные недостатки у данной технологии также практически отсутствуют.

CSS в отличие от HTML использует несколько иной алгоритм описания элементов web-страниц. Один раз указав свойства каждого элемента в текстовом файле с расширением .css и назначив им свойства стиля, можно затем подключить этот файл к html-документу, заставив клиентский браузер считывать значения параметров каждого компонента web-страницы уже оттуда. Более того, поскольку стили описываются вами

в отдельном файле, вы можете подключить его к неограниченному количеству различных документов, раз и навсегда отказавшись от необходимости назначать свойства каждому конкретному объекту.

Для элементов HTML, описанных в файле CSS, справедлив принцип наследования.

Есть и еще одно неоспоримое преимущество использования CSS перед традиционным способом подготовки web-страниц, которое кажется неочевидным на первый взгляд: для того, чтобы изменить стиль оформления какого-либо элемента всех web-страниц вашего сайта, достаточно немного подправить всего одну строку кода в одном файле.

Метод описания стилей с помощью CSS является оптимальным для web-дизайна.

CSS является достаточно гибкой структурой, позволяющей описывать не только определенные параметры для каких-либо элементов таблицы стилей, но и назначать различные свойства одним и тем же элементам.

Поскольку стандарт CSS окончательно сложился значительно позже стандарта HTML, в полной мере его поддерживают далеко не все версии браузеров, а именно – Microsoft Интернет Explorer начиная с версии 4.0 и Netscape Navigator старше четвертой версии. Более того, интерпретаторы данных браузеров работают, как известно, с некоторыми отличиями, а потому одни и те же элементы, специфицированные согласно стандарту CSS, могут отображаться в них совершенно по-разному. Старые браузеры вообще могут обрабатывать команды CSS по абсолютно непредсказуемому алгоритму.

Поэтому с объективной точки зрения нецелесообразно использование для представления содержимого web-страниц каскадных таблиц стилей в полном объеме. Наоборот, рекомендуется применять лишь ограниченный набор элементов, заведомо поддерживаемый браузерами большинства версий. Такой набор директив CSS условно называется «безопасным».

Широкие возможности каскадных таблиц стилей можно использовать совместно с html-документом различными способами. Первый из них подразумевает использование нотаций CSS непосредственно в тегах кода web-страницы, помещаемых в тело документа.

Второй способ подразумевает включение в html-код web-страницы инструкций CSS наподобие подпрограммы. Он также не требует подключения к странице отдельного css-файла и применяется в основном в тех случаях, когда создание такого файла представляется нецелесообразным, например, если стили используются в пределах лишь одного документа HTML.

Третий метод использования в документе HTML элементов стилей подразумевает подключение к данному документу отдельного файла с расширением .css, содержащего описания всех элементов стилей и их свойств, который хранится на сервере отдельно от web-страниц.

Наконец, четвертый способ подключения файла CSS к web-странице, наиболее традиционный и часто используемый, подразумевает добавление в заголовок html-документа специального тега < LINK>.

CSS можно применять на всех без исключения типах http-серверов без ограничений. Те или иные ограничения на использование каскадных таблиц стилей может накладывать только клиентское программное обеспечение. Не требуется также никаких надстроек и дополнительных программных модулей для того, чтобы браузеры пользователей могли обрабатывать директивы CSS. Файлы CSS просто загружаются на сервер совместно с web-страницами, а при открытии в браузере посетителя вашего сайта они будут автоматически считаны с удаленного узла и запущены на исполнение.

2.6.6. PHP (Personal Home Page tools)

PHP – это скрипт-язык (scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере. Проще всего это показать на примере:

```
<html>
<head>
<title>Example</title>
</head>
<body>
<?php echo "Hi, I'm a PHP script!"; ?></body>
</html>
```

После выполнения этого скрипта получаем страничку, в которой будет написано:

Hi, I'm a PHP script!

Довольно просто.

Основное отличие от CGI-скриптов, написанных на других языках, типа Perl или C – это то, что в CGI-программах самостоятельно пишется выводимый HTML-код, а, используя PHP, вы встраиваете свою программу в готовую HTML-страницу, используя открывающий и закрывающий теги (в примере - <?php и ?>).

Отличие PHP от JavaScript, состоит в том, что PHP-скрипт выполняется на сервере, а клиенту передается результат работы, тогда как в JavaScript-код полностью передается на клиентскую машину и только там выполняется.

Любители Интернет Information Server найдут, что PHP очень похож на Active Server Pages (ASP), а энтузиасты Java скажут, что PHP похож на

Java Server Pages (JSP). Все три языка позволяют размещать код, выполняемый на Web-сервере, внутри HTML страниц.

Возможности PHP

На PHP можно сделать все, что можно сделать с помощью CGI-программ. Например: обрабатывать данные из форм, генерировать динамические страницы, получать и посылать куки (cookies).

Кроме этого в PHP включена поддержка многих баз данных (databases), что делает написание Web-приложений с использованием БД до невозможности простым.

Таблица 1.

Перечень основных поддерживаемых баз данных

Adabas D	InterBase	Solid
dBase	mSQL	Sybase
Empress	MySQL	Velocis
FilePro	Oracle	Unix dbm
Informix	PostgreSQL	Access

Кроме того, PHP понимает протоколы IMAP, SNMP, NNTP, POP3 и даже HTTP, а также имеет возможность работать с сокетами (sockets) и общаться по другим протоколам.

Краткая история PHP

Началом PHP можно считать осень 1994 года, когда Rasmus Lerdorf решил расширить возможности своей Home-page и написать небольшой движок для выполнения простейших задач. Такой движок был готов к началу 1995 года и назывался Personal Home Page Tools. Умел он не очень много – понимал простейший язык и всего несколько макросов.

К середине 1995 года появилась вторая версия, которая называлась PHP/FI Version 2. Приставка FI присоединилась из другого пакета Rasmusa, который умел обрабатывать формы (Form Interpritator). PHP/FI компилировался внутрь Apache и использовал стандартный API Apache. PHP скрипты оказались быстрее аналогичных CGI – скриптов, так как серверу не было необходимости порождать новый процесс. Язык PHP по возможностям приблизился к Perl, самому популярному языку для написания CGI-программ. Была добавлена поддержка множества известных баз данных (например, MySQL и Oracle). Интерфейс к GD – библиотеке, позволял генерировать картинки на лету. С этого момента началось широкое распространение PHP/FI.

В конце 1997 Zeev Suraski и Andi Gutmans решили переписать внутренний движок, с целью исправить ошибки интерпретатора и повысить скорость выполнения скриптов. Через полгода, 6 июня 1998 года вышла новая версия, которая была названа PHP 3.

К лету 1999 года PHP3 был включен в несколько коммерческих продуктов. По данным NetCraft, на ноябрь 1999 PHP использовался в более чем 1 млн. доменах.

На декабрь 1999 выпущена версия PHP 4. Производительность этой версии в десятки раз выше, чем у существующей.

Почему нужно выбирать PHP

Известно, что web-страницы - это не только текст и картинки. Достойный внимания сайт должен поддерживать некоторый уровень интерактивности с пользователем: поиск информации, продажа продуктов, конференции и т.п. Традиционно все это реализовалось CGI-скриптами, написанными на Perl. Но CGI - скрипты очень плохо масштабируемы. Каждый новый вызов CGI требует от ядра порождения нового процесса, а это занимает процессорное время и тратит оперативную память. PHP предлагает другой вариант – он работает как часть Web-сервера, и этим самым похож на ASP от Microsoft.

Синтаксис PHP очень похож на синтаксис C или Perl. Люди, знакомые с программированием, очень быстро смогут начать писать программы на PHP. В этом языке нет строгой типизации данных и нет необходимости в действиях по выделению/освобождению памяти.

Программы, написанные на PHP, достаточно легкочитаемы. Написанный PHP – код легко зрительно прочитать и понять в отличие от Perl-программ.

Недостатки PHP

Основным недостатком PHP 3 является то, что по свой идеологии PHP изначально был ориентирован на написание небольших скриптов. PHP 3 был не пригоден для использования в сложных проектах – при обработке больших скриптов производительность системы резко падает. Однако этот недостаток ликвидирован в PHP 4, который может быть эффективно исрользован для работы в больших проектах.

PHP является интерпретируемым языком, и, вследствие этого, не может сравниться по скорости с компилируемым C. Однако при написании небольших программ, что, в общем-то, присуще проектам на PHP, когда весь проект состоит из многих небольших страниц с кодом, вступают в силу накладные расходы на загрузку в память и вызов CGI-программы, написанной на C.

2.6.7. ASP (Active Server Pages)

ASP (активные страницы сервера) – технология, аналогичная JavaScript и PHP. Для того, чтобы сделать web-страницу интерактивной с применением технологии ASP, необходимо встроить в ее код соответствующий скрипт, написанный на макроязыке, отдаленно напоминающем Java и C. Скрипт интерпретируется и исполняется

непосредственно на сервере, после чего пользовательскому браузеру отправляется уже готовый html-документ с результатами работы сценария ASP. Отсюда следует вполне справедливое заключение о том, что для страниц, содержащих ASP, не имеет значения, какое программное обеспечение установлено на пользовательском компьютере. Зато принципиальное значение имеет тип сервера, на котором вы планируете использовать ASP, поскольку отнюдь не все они поддерживают данную технологию.

Применение встраиваемых в ASP ActiveX-компонентов значительно облегчает задачу создания сайтов. Ведь создание громоздких функций, выполняющих сложные вычисления, да и простых подпрограмм гораздо удобнее с помощью известных всем языков программирования, таких как Visual Basic, Delphi или C++, а сгенерированный код (.dll или .ocx) будет работать во много раз быстрее, так как скомпилированный код модуля выполняется почти мгновенно, в то время как препроцессор ASP формирует HTML-страницу для каждого клиента гораздо дольше. Реализация этих функций средствами самого ASP во многих случаях будет нетривиальна, а в некоторых — попросту невозможна. Кроме того, многие полезные компоненты доступны со стороны так называемых третьих фирм (third-party components), которые зачастую распространяются бесплатно (или почти бесплатно).

Работать с ActiveX-компонентами в ASP достаточно просто. Как правило, компоненты, предоставляемые «третьими» фирмами, не являются частью самого ASP, а представляют собой так называемые черные ящики, настроить которые под конкретные нужды невозможно. Будучи «третьими» компонентами, они должны устанавливаться на серверы, а это, в свою очередь, означает, что понадобится скопировать файлы компонентов (DLL или OCX) на сервер и зарегистрировать их. Проблемы не возникают, если вы сами осуществляете хостинг вашего сайта. Однако если этим занимается какая-нибудь другая компания, то могут возникнуть проблемы с размещением и (или) регистрацией компонентов на сервере, обслуживающем ваш сайт.

Пара строк ASP-кода, обрабатывающая длинные строки символов, выглядит великолепно на сервере с небольшим числом клиентов, но может «поставить на колени» сервер с множеством клиентов.

Отсюда вывод: самые «узкие» места целесообразно разрабатывать на каком-нибудь языке программирования вроде C++, а ASP использовать в качестве связующего звена между быстро работающими ActiveX-компонентами, созданными с его помощью.

2.6.8. Macromedia Flash

В последнее время технология Flash завоевала прочные позиции на всемирном рынке инструментов по созданию Web-приложений. Элементы

Flash-приложений есть на многих сайтах известных компаний и корпораций, таких как "Pepsi","Adidas", "Disney".

Есть немало программ, разработанных специально для создания веб-анимации, векторной графики, интерактивных фильмов и даже полноценных сайтов. Но появилась необходимость в программе, с помощью которой можно было бы соединить не только вышеперечисленные элементы, а также низкочастотные звуки и музыку высшего качества, растр-графику и, что немаловажно, иметь при этом интерактивность. Примерно так можно охарактеризовать Macromedia Flash 5.

Macromedia Flash 5 - практически стандарт для использования в web векторных изображений. Обладает собственной средой разработки и позволяет создавать впечатляющую векторную анимацию. Его также называют "мультипликатором Интернета".

Веб - не единственная цель Flash 5. С помощью этой простой и в тоже время мощной программы, можно создавать анимационные презентации, простые 3D объекты, разнообразные игры и интерактивные анкеты.

К достоинствам программы следует отнести:

- Файл обычно загружается один раз, и поэтому при нажатии ссылки не нужно ждать каждый раз, что существенно ускоряет работу.
- Работает с векторный графикой.
- Можно управлять событиями через JavaScript.

У Flash имеется библиотека: использованные один раз элементы (любые объекты, рисунки, музыка) сохраняются в них. При необходимости использования элемента несколько раз, применяются копии этого элемента из библиотеки, и в этом состоит его главное достоинство. Если была использована кнопка в одном месте (например, она занимает 4,5 КБ), и нужно применить её кнопку в другом месте, размер общего файла увеличится не намного - меньше, чем на 100 байтов. Эта технология схожа с технологией SSI.

Технология Flash делает Web-страницу динамичной. Пользователь не только видит анимацию, которая повторяется каждую 4 секунд, но также может влиять на вид и содержание при нажатии ссылки отображаемой страницы (например, бабочка двигается за курсором пользователя, и это то время как курсор превращается в тюльпан и т.д.). У Flash нет проблем с кодировкой. Имеется ввиду то, что некоторые теги не распознаются различными браузерами. А если установить плагин Flash Player (примерно 300КБ), то проблем с работой не будет. Вся работа, сделанная на Flash сохраняется в один (SWF) файл расширением .swf. Это удобно, так как файл легко найти и использовать. Не нужны специальные программы для перекачки файла. Достаточно скопировать HTML и SWT файлы.

Web-страница имеет малый размер на Flash сравнительно с другими технологиями, что очень существенно при загрузки страницы у пользователя. Долго загружающиеся файлы раздражают клиента, и такую страницу стараются не посещать. Лучшие Flash-страницы выглядят качественно лучше, чем традиционные сайты. Создается впечатление, что находишься не на сайте, а смотришь фильм и находишься внутри него. Flash постоянно развивается и его недостатки могут быть преодолены.

К недостаткам Flash относятся следующие:

- Нужен специальный плеер-плагин (плеер (player) - это проигрыватель, который делает возможным просмотр файлов различных форматов (в данном случае формат swf); плагин – это специальная и дополнительная программа, которая настраивает браузер клиента таким образом, чтобы Flash-ролики и сайты показывались в вашем браузере), хотя последние версии Интернет Explorer (начиная с версии 5.0) оснащены такими плагинами.
- Большой недостаток - это работа с шрифтами. Когда впервые был создан Flash1.0, он не предназначался для того что бы делать полноценные веб-страницы, а являлся только приложением (помощником) для создания сайтов, в особенности для создания разных кнопок и мульт-роликов. Этот недостаток присущ и Flash 5.

По дизайну Flash удовлетворителен, но со стороны языка программирования нужны доработки. Он требует больше системных ресурсов, чем простой HTML файл. Flash также уе подходит для коммерческих сайтов (банковские переводы, переводы денег и т.д. через Интернет).

2.6.9. VBScript (Visual BASIC Script)

VBScript (Visual Beginners All-purpose Symbolic Instruction Code Script) или визуальный символический универсальный командный код для начинающих – очередная версия интерпретируемого языка, встраиваемого в html-документ с целью включения в состав web-страницы интерактивных элементов. Честь создания данной технологии принадлежит разработчикам компании Microsoft.

VBScript (Script версия языка Visual Basic) фирмы Microsoft - это язык сценариев, предусмотренный в браузере Microsoft Интернет Explorer, и наиболее очевидный вариант выбора для программирования Web страниц на базе ActiveX-элементов. Он представляет собой адаптированное для Web подмножество языка VBA (Visual Basic for Applications) с принятым в Microsoft синтаксисом Бейсика. Здесь используются обычные для языка Бейсик обозначения объект-параметр (перечисляемые через точку), процедуры, функции и структуры, пригодные для управления выполнением программы, а также широкий набор традиционных для этого языка функций. Все, кому ранее уже

приходилось работать с современным языком Бейсик, прекрасно справятся и с VBScript.

Если сравнить более распространенный стандарт JavaScript с VBScript, обнаружить серьезные различия очень трудно, поскольку мнемоника и синтаксис обоих языков во многом схожи. С помощью VBScript можно реализовать практически весь спектр возможностей, характерных для JavaScript. Обе технологии не зависят от типа сервера, на котором планируется опубликовать включающую их web-страницу. Однако VBScript в настоящее время менее распространен в Интернете, нежели его «конкурент», поскольку он поддерживается только браузерами производства Microsoft, а именно Интернет Explorer версии 3.0 и выше. Netscape Navigator не имеет интерпретатора этого языка, поэтому приверженцы этого браузера лишены возможности использовать интерактивные элементы, созданные с применением VBScript, в то время как JavaScript поддерживается и Интернет Explorer и Netscape Navigator.

Как и в VBA, предложение Option Explicit в VBScript предназначено для явного объявления переменных. Однако в отличие от VBA в VBScript вы не задаете явно типы переменных, все они - вне зависимости от того, строка это или число, - хранятся как тип Variant. Конечно, неявно заданные типы данных существуют - включая integer, long, single, double, string, numeric, date и financial, - но для системы это лишь формат представления данных. К счастью, имеются функции как для преобразования, так и для проверки типов переменных (например, VarType, IsNumeric), которые можно достаточно уверенно применять для Web страниц, оснащенных средствами для проверки допустимости типов данных.

В VBScript вы не найдете средств для ввода-вывода файлов и для прямого доступа к памяти. Такие ограничения, характерные также для Java и JavaScript, обусловлены необходимостью исключить появление программ с разрушительным или пагубным поведением. Удивительно, но среди средств VBScript даже не предусмотрен объект Debug. Обнаружилось, что во всех имеющихся инструментах практически полностью отсутствуют какие либо средства отладки.

Подобного рода упражнения - это как раз и есть конструирование Web страниц с помощью ActiveX-элементов, стоит отметить, что в VBScript управление объектами форм (form objects) HTML не отличается от манипуляций с текстовыми блоками и кнопками. Даже применение этих простых средств позволит вам обогатить разнообразными функциональными возможностями свою страницу. Обращения к объектам форм организуются в VBScript путем указания имени формы FORM NAME и параметра NAME любого объекта, имеющегося в этой форме.

DHTML (Dynamic HTML- Динамический HTML)

Всемирная Паутина (World Wide Web) спровоцировала революцию в информатике, предоставив любому пользователю возможность

публикации HTML-документов. До недавнего времени информация в этих документах была в большинстве случаев статической, что требовало реакции сервера на действия пользователя. С введением динамического HTML парадигма Web сместилась от взаимодействия с сервером в сторону создания интерактивных Web-узлов и Web –приложений. Поскольку динамический HTML обеспечивает возможность взаимодействия HTML-документов с пользователем и полного их изменения на клиентском компьютере, вы можете создавать Web-приложения с богатыми возможностями.

Динамический HTML построен на объектной модели, которая расширяет традиционный статический HTML-документ.

Динамический HTML обеспечивает API, необходимый для полного управления HTML-документом. Больше не требуется определять страницу во время загрузки. После загрузки любая часть страницы может быть сразу же изменена в динамическом режиме. Например, вы можете создать приложение, которое разворачивает или сворачивает список содержания документа. Когда пользователь разворачивает или сворачивает список, на экран мгновенно выводится или с него удаляется содержание. Представьте создание страниц, которые автоматически изменяются и настраиваются для пользователя. Все это возможно с помощью динамического HTML.

Способность изменения документа и автоматическое переформатирование документа является основным нововведением, которое используется в динамическом HTML. Традиционные браузеры были основаны на инструментах перехода по документу, которые отображали документ и затем ожидали, пока пользователь выберет новый документ. Когда требовалось внести изменение в документ, для создания новой страницы отправлялся запрос на сервер, который и генерировал ее на машине клиента.

Язык программирования используется для манипулирования объектной моделью динамического HTML, но динамический HTML разработан как независимый от платформы и языка программирования. Поэтому могут быть использованы языки программирования JavaScript, Jscript, VBScript, C++, Java, а также другие языки.

Если вы знакомы с языком JavaScript и существующей объектной моделью, то обнаружите, что расширения объектной модели динамического HTML совместимы со всеми предыдущими версиями. Все страницы, написанные для предыдущих версий Интернет Explorer или Netscape Navigator 3.0, будут выполняться в Интернет Explorer 5.0. Такая совместимость позволяет Web-мастерам использовать имеющиеся знания при изучении данных нововведений. Тем, кто не знаком с JavaScript, изучение методов программирования HTML-страницы позволит расширить и улучшить работу создаваемых страниц и приобрести некоторый опыт.

Динамический HTML снимает все ограничения на доступ к документу. Динамический HTML в Интернет Explorer 5.0 предоставляет в распоряжение разработчиков ряд новых элементов:

HTML 4.0 и расширенная поддержка CSS. Броузер Интернет Explorer 5.0 поддерживает последний стандарт HTML 4.0, CSSI и множество новых элементов в CSS. Эти стандарты HTML и CSS определяют элементы, представляемые объектной моделью динамического HTML.

Полный доступ к структуре документа. Доступ к элементам в документе поддерживается объектной моделью динамического HTML. Вы больше не ограничены программированием элементов форм. Стиль и содержание любого элемента могут быть изменены в динамическом режиме, и данные изменения будут немедленно отражены в документе.

Кроме того, улучшены внутренние элементы управления для поддержки HTML и CSS, что позволяет Web-мастеру манипулировать внешним видом данных элементов управления, включая установку цвета текста, фона и шрифта на кнопках и текстовых элементах управления. Объектная модель внутренних элементов управления сходна с объектной моделью документа и обеспечивает простой доступ к стилю и содержанию.

Динамический стиль. Таблицы стилей CSS документа могут быть изменены в любое время. Документ не требуется перезагружать из кэша или обращаться на сервер. Объектная модель разработана для предоставления возможности немедленного отображения любых изменений на странице. Например, внешний вид элемента может быть изменен при перемещении указателя мыши или нажатии кнопки мыши на элементе.

Динамическое содержание. Объектная модель позволяет обращаться к содержимому документа и изменять его. Кроме того, при этом не требуется взаимодействие с сервером и реакция будет мгновенной. Например, можно написать утилиту электронных часов в стандартном HTML. Больше не требуется использовать апплеты Java или элементы управления ActiveX для изменения содержания.

Мгновенный ответ пользователя. Динамический HTML предоставляет мощную новую модель событий, которая представляет все пользовательские действия на странице. Сценарии в документе могут реагировать на все действия пользователя внутри броузера. На основе действий пользователя любой апплет в содержании документа или стиле может быть изменен в динамическом режиме.

Web-страницы клиент/сервер. Интернет Explorer 4.0 добавляет расширения в HTML-элементы для создания таблиц со связыванием данных и форм с одиночной записью и отчетов. Данные асинхронно загружаются и воспроизводятся в документе, используя несколько базовых HTML-расширений. Данные могут быть кэшированы локально, позволяя

осуществлять поиск и сортировку на клиентской стороне без помощи сервера. Например, поисковые машины могут представлять множество результатов поиска одновременно. Напротив, поисковая машина может отправлять запросы клиенту, где они будут воспроизводиться по мере получения. Пользователь может мгновенно сортировать и отбирать данные полностью на локальной машине, не отправляя запросов на сервер.

Эффекты мультимедиа и анимации. Броузер Интернет Explorer 4.0 тесно интегрирует эффекты мультимедиа и анимации с содержанием документа. Данные эффекты включают фильтры, которые могут моделировать источники света, тени и другие эффекты, которые реализуются непосредственно в тексте или других элементах управления. Могут быть также добавлены эффекты перехода между изображениями и текстом, и даже между страницами.

Все эти элементы основаны на текущем обсуждении внутри рабочих групп консорциума W3C (World Wide Web Consortium). Объектная модель динамического HTML рассматривается рабочей группой объектной модели документа (Document Object Model). Цель этой группы заключается в определении объектной модели, которая является независимой от языка и платформы и удовлетворяет ряд требований для структурированных документов. Объектная модель, которая определена в Интернет Explorer 4.0, удовлетворяет всем требованиям, изложенным рабочей группой объектной модели документа.

HTML является приложением языка SGML (Standard Generalized Markup Language – стандартный обобщенный язык разметки). В документе

HTML/SGML теги определяют структуру содержания документа. Традиционный SGML-документ имеет три различных компонента: структура, стиль и содержание. С введением динамического HTML был добавлен четвертый компонент: модель поведения. Термин «модель поведения» (behavior) определяет взаимодействие между пользователем и HTML-страницей.

Для полноценного использования возможностей динамического HTML документ должен правильно отделять содержание и структуру от представления. Динамический HTML проще использовать и он работает более предсказуемо с действительными HTML-документами. Манипулирование недействительным HTML сложнее и может привести к непредсказуемым результатам.

XML и XHTML

XML (Extensible Markup Language)

XML(расширяемый язык разметки) - является принципиально новым стандартом, предложенным в 2000 году создателем языка HTML – консорциумом World Wide Web Consortium (W3C). Это новейшая технология изготовления web-страниц, и ее окончательная спецификация в настоящий момент еще находится в стадии разработки. По структуре XML

представляет собой не собственно язык разметки гипертекста, а так называемый метаязык, предназначенный для описания других языков более низкого уровня.

XHTML (Extensible HyperText Markup Language)

XHTML (расширяемый язык разметки гипертекста) - представляет собой промежуточный вариант между XML и HTML 4.0. Именно за счет его широкого применения W3C планирует осуществить постепенный переход от одного стандарта к другому. В основу спецификации XHTML положен принцип обратной совместимости. Иными словами, владельцам web-сайтов, страницы которых выполнены по технологии HTML 4.0 или более ранних версий, не придется каким либо образом изменять формат опубликованных в Интернете документов: все дополнения и расширения нового языка полностью включают в себя предыдущие стандарты. XML и XHTML значительно расширяют возможности HTML и позволяют web-мастерам использовать практически весь потенциал, заложенный в современный Интернет, на сто процентов, в первую очередь в сфере электронной коммерции. Поскольку XHTML подразумевает модульную архитектуру построения электронных документов, данный стандарт позволяет создавать механизмы взаимодействия с нетрадиционными для Интернета устройствами, такими, как факсы, сотовые телефоны и телевизоры. По мнению аналитиков, в не столь отдаленном будущем XML станет основным стандартом во Всемирной сети, постепенно включив в себя большинство других использующихся ныне форматов.

3. WWW и средства интерактивного взаимодействия

Цель данного вопроса познакомить пользователя с той частью WWW-технологий, которая связана с созданием интерактивных интерфейсов и предполагается, что пользователь знаком с основами WWW, HTML и C/C++.

В общем случае интерактивный интерфейс пользователя представляет собой систему, обеспечивающую взаимодействие пользователя и программы. Для WWW интерактивный интерфейс можно определить как последовательность HTML-документов, реализующих интерфейс пользователя. Можно также условно классифицировать принципы построения интерфейса по типу формирования HTML-документа:

- статический;
- динамический.

В первом случае источником интерфейса является HTML-документ, созданый в каком-либо текстовом или HTML-ориентированном редакторе. Следовательно, данный документ остается неизменным в течение использования. Во втором случае источником интерфейса является HTML-

документ, сгенерированный cgi-модулем. Следовательно, появляется некоторая гибкость в видоизменении интерфейса во время использования.

Таким образом, можно ввести понятие интерактивного интерфейса для WWW.

Интерактивный интерфейс для WWW представляет собой последовательность статически или динамически формируемых HTML-документов, реализующих интерфейс пользователя.

Практически любая задача, решающая проблему получения данных от клиента, связана с построением интерфейса. Наиболее интересным является построение интерфейсов к различным базам данных, доступ к SQL-серверу, получение информации от периферийных устройств, создание клиентских рабочих мест. Все это возможно посредством CGI (Common Gateway Interface).

Common Gateway Interface (CGI) является стандартом интерфейса внешней прикладной программы с WWW сервером.

Задача построения вышеназванных интерфейсов делится на две части:

- Клиентская часть.
- Серверная часть.

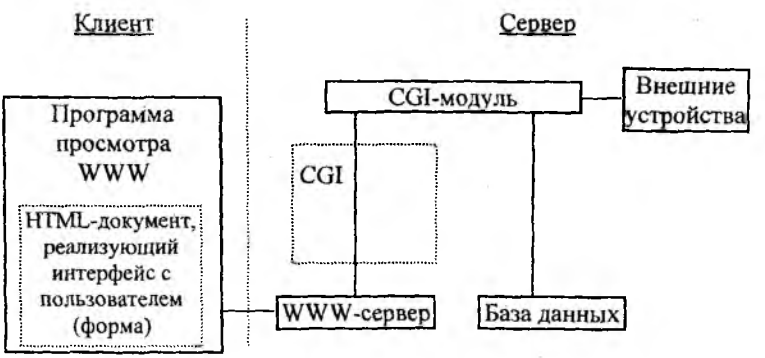


Рис. 3. Две части интерактивного интерфейса

3.1. Компоненты приложений клиент / сервер

Ключ к успешному выполнению программы клиент/сервер - баланс ее различных компонентов. Можно реализовать очень простую и недорогую архитектуру, эффективную для небольшого числа пользователей и / или простых приложений. Однако для поддержки большого числа пользователей или очень сложного приложения необходима более сложная архитектура.

Для реализации компьютерного приложения с архитектурой клиент/сервер, в первую очередь необходимо понимание функций различных уровней, обычно существующие внутри этого приложения. На

рис.4 показаны некоторые потенциальные уровни любого компьютерного приложения.

Крайним слева на этом рисунке изображен конечный пользователь – клиент данной архитектуры – тот, кто запрашивает услуги приложения. При продвижение по схеме слева направо каждый из уровней, расположенных левее, можно рассматривать в качестве клиента для сервера, формируемого уровнями, расположеннее правее. Так, пользователь приложения через свой интерфейс запрашивает выполнение прикладной задачи, например, ввод данных о новом клиенте, оформление заказа или печатание расчетов. Прикладные задачи приложения должны соответствовать бизнес – правилам (покупатель не может заказать более того, что определенно рамками его кредита), которые, в свою очередь, подчиняются правилам целостности данных (заказ может ввести только определенный покупатель; величина заказа должна выражаться двузначным десятичным числом). Крайняя справа выборка данных обычно выполняется устройством управления базой данных, которое, в свою очередь, использует устройство доступа к файлам, обеспечиваемого операционной системой компьютера.

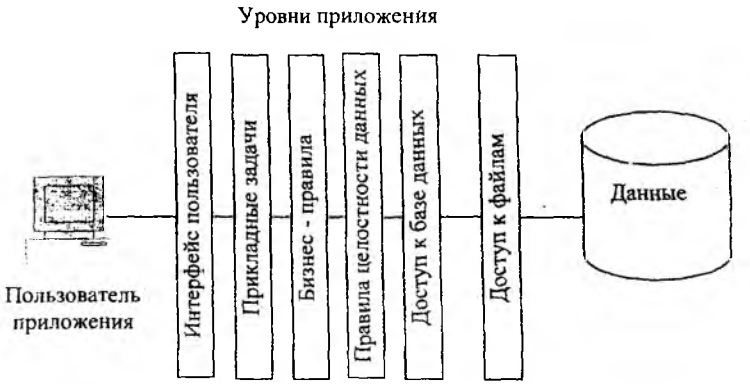


Рис. 4. Типичные уровни компьютерного приложения

3.2. Разделение клиента и сервера

Если все уровни приложения выполняются на одном компьютере, значит, это однопользовательское приложение. Для создания архитектуры клиент/сервер необходимо разделить сферы действия клиента и сервера по границе любого из уровней приложения. Самое простое разделение клиент/сервер можно получить, выбрав ближайшую границу к любому концу данной архитектуры.

3.2.1. Выполнение основной работы на сервере

Один из простых способов разделений функций клиента и сервера - поручить клиенту только интерфейс пользователя, а остальное предоставить серверу (рис.5).

Здесь в системе пользователя размещена часть приложения, ответственная за просмотр данных, ввод с клавиатуры или с помощью мыши. Остальная часть выполняется на большой распределенной системе. Иногда клиентный компьютер представляет собой не более чем "интеллектуальный" терминал данных. Подобная архитектура еще встречается в устаревших громоздких вычислительных система.

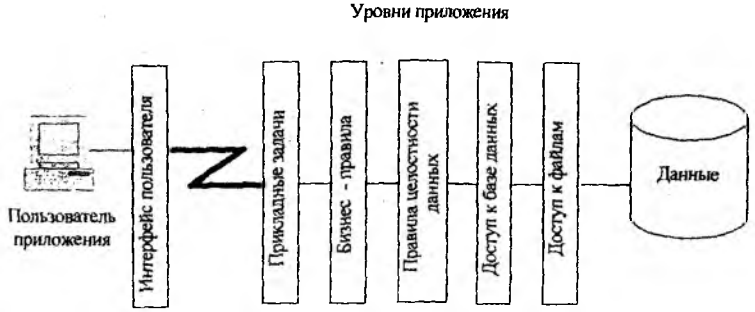


Рис. 5. Размещение большей части задач на сервере

Даже если пользователь работает с приложением такого типа на персональном компьютере, его компьютер только эмулирует терминал или дает "графическое" изображение более старой, основанной на символах системы. В клиентской части приложения выполняется, как максимум, проверка правильности простых данных.

Такая архитектура подобна архитектуре большинства диалоговых сервисных систем. Компьютер пользователя осуществляет связь и обеспечивает обзор программного обеспечения (например, систему навигации и просмотра информации в сети Web-сервер), а основную работу выполняют компьютеры сети, с которыми соединен пользователь. Другой пример – система X-Windows, работающая под операционной системой UNIX. В обоих случаях, несмотря на то, что клиент обеспечивает только представление и перемещение данных, его компьютера должен быть достаточно мощным, чтобы быстро оперировать со сложными потоками графических данных, идущими с сервера.

Иногда профессионалы в области компьютерной технологии полагают, что такой тип простого разграничения не определяет правильность приложения клиент/сервер. И все таки, если программное обеспечение компьютера или рабочей станции клиента выполняет важную работу по обеспечению пользователя усиленным графическим

интерфейсом и по отправлению запросов на сервер с помощью простого потока символьных данных, такой вид приложения соответствует духу вычислений типа клиент/сервер.

Преимущество такой архитектуры – все данные и большинство прикладных логических средств сосредоточены в одном месте. Большинство изменений или усовершенствований приложения можно выполнить на центральном сервере. Недостаток – центральный сервер должен быть очень мощным, если доступ к данному приложению необходим многим пользователям. Поскольку связь между клиентом и сервером может потребовать работы с большими томами графической информации, для получения достаточно быстрых ответов на запросы может понадобиться очень дорогая высокоскоростная сеть, связывающая клиента с сервером.

3.2.2. Выполнение основного объема работы у клиента

Противоположный описанному в предыдущем разделе вариант – разделение приложения таким образом, что сервер работает только как простое хранилище данных, или файловый сервер (рис.6). Все представления данных и прикладные логический средства размещены на клиентном компьютере. Такая архитектура чаще всего используется, когда для всех приложений применяются продукты оперативной базы данных типа Microsoft Access. Хотя таблицы данных легко переместить на сервер и установить соединение с множеством копий данных приложения, вся работа, исключая доступ к файлам базового уровня, выполняется на клиентных компьютерах.

Основное достоинство такой архитектуры – ее простота. Для совместного использования данных системой клиент/сервер достаточно переместить файлы данных на сервер управления файлами. Однако эта архитектура требует достаточно сложного аппаратного обеспечения клиента, и поскольку большинство файловых серверов осуществляет только первичное распределение данных на уровне файлов, она предусматривает только легкие загрузки доступа к данным или относительно малое число совместно работающих пользователей.

Даже, когда программное обеспечение клиента позволяет применять механизм управления доступом сервера, обеспечивающий совместное использование на уровне блоков (такую возможность дает Access), число пользователей может превзойти ресурсы даже самых мощных серверов, если только приложения не предназначено для запроса минимума данных для решения локальной задачи.

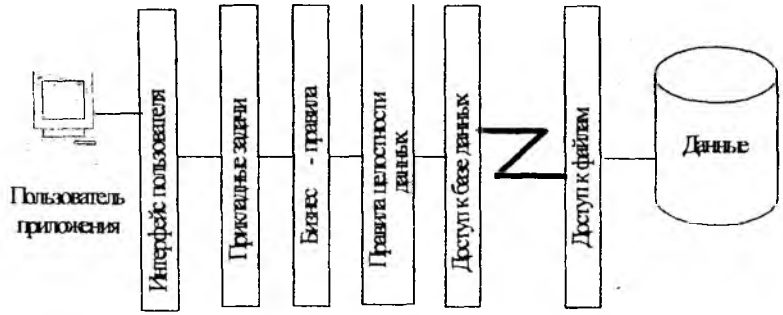


Рис. 6. Размещение большой части задач на стороне клиента

Установка приложения Access изначально, предназначенного для одного пользователя и перемещения его таблиц данных на совместно используемый сервер, необязательно является правильной реализацией архитектуры клиент/сервер. Такое приложение обычно предполагает локальность данных, поэтому представление данных и манипулирование ими проводятся со всеми доступными данными. Для эффективной работы архитектуры клиент/сервер клиентная часть приложения должна иметь особую конструкцию и запрашивать информацию только для решения отдельной текущей задачи. Приложение Access подобной конструкции легко "увеличивается " для использования нормального сервера базы данных. Если же приложение сконструировано иначе, его невозможно использовать в качестве приложения клиент/сервер независимо от количества работы, предназначаемой серверу.

3.2.3. Достижения Баланса

Для использования преимущества архитектуры клиент/сервер в полной мере нужно так сконструировать систему приложения, чтобы достичь баланса рабочей загрузки компьютерных систем клиента и сервера. Для этого, в первую очередь, следует реализовать не файловый, нормальный сервер базы данных, который способен на любую работу по выполнению запросов путем прямого доступа к памяти для хранения данных. В сеть должны возвращаться только данные, необходимые для выполнения конкретной задачи (это часто самый "медленный" компонент архитектуры клиент/сервер).

Сервер базы данных может также централизованно реализовывать все правила целостности и защиты данных. При этом целостность данных приложения обеспечивается независимо от целостности данных всех клиентов. Более того, мощные серверы базы данных (типа Microsoft SQL Server) позволяют выполнять бизнес – правила более высокого уровня, которые нельзя нарушить независимо от действий клиента (рис.7).

Реализация сбалансированной архитектуры требует дополнительной работы и планирования, но в результате достигается высокая степень

целостности данных и возможность манипулировать более сложными рабочими загрузками. Недостаток ее в том, что ни клиент, ни сервер не являются простыми реализациями.

Серверная часть состоит из исполняемого модуля, решающего основные задачи обработки данных, поступающих от клиентской части, формирования ответа в формате HTML и т.д. Такой модуль называется *cgi-модулем*.

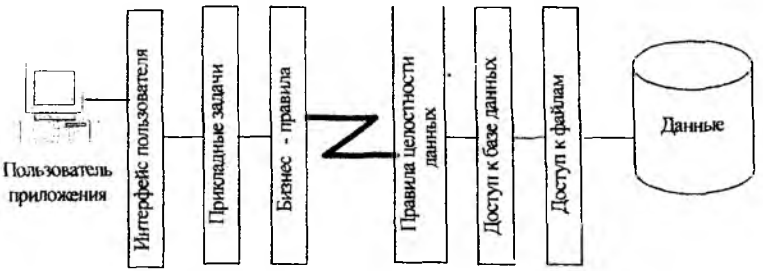


Рис. 7. Схема баланса задач клиента и сервера

3.3. Серверная часть

Механизм реализуется за счет наличия двух более или менее стандартизованных средств: возможности включения форм в документ, составленный с использованием языка гипермедийной разметки HTML, и возможности использования внешних по отношению к серверу Web программ, взаимодействие которых происходит через специфицированный протокол CGI (Common Gateway Interface) или внедренный позже API (Application Program Interface). (Хотя CGI называется "общим интерфейсом шлюзования", по сути дела, это одновременно некоторое подмножество протокола HTTP, и способ его соблюдения при взаимодействии сервера с внешней программой.)

При реализации на основе CGI общая схема реализации доступа к базе данных на стороне Web-сервера выглядит следующим образом:

- при просмотре документа клиент встречает ссылку на страницу, содержащую одну или несколько форм, предназначенных для запроса данных из базы данных;
- клиент запрашивает эту страницу; помимо незаполненных форм страница может содержать общую информацию о базе данных и о назначении предлагаемых форм;
- если клиента действительно интересует информация из базы данных, которую можно получить на основе предложенных форм, то он заполняет одну из форм и отправляет заполненную форму на сервер;

- получив заполненную форму, сервер запускает соответствующую внешнюю программу, передавая ей параметры и получая результаты на основе протокола CGI;
- внешняя программа преобразует запрос, выраженный с помощью заполненной формы, в запрос на языке, понятном серверу баз данных (обычно это язык SQL).
- внешняя программа взаимодействует с сервером баз данных; взаимодействие может быть прямым, если внешняя программа жестко привязана к конкретному SQL-серверу, или с использованием, например, протокола и соответствующего драйвера ODBC, если жесткая привязка отсутствует;
- после получения результатов запроса внешняя программа формирует соответствующую виртуальную или реальную HTML-страницу, передает ее серверу и завершает свое выполнение;
- сервер передает сформированную HTML-страницу клиенту, и на этом процедура доступа к базе данных завершается (как обычно, сервер разрывает транспортное соединение с клиентом).

На языке Web-мастеров любая внешняя программа, запускаемая Web-сервером в соответствии со спецификациями CGI, называется CGI-скриптом. CGI-скрипт может быть написан на языке программирования (Си, Си++, Паскаль и т. д.) или на командном языке (языки семейства shell, perl и т. д.). CGI-скрипт, выполняющий роль посредника между Web-сервером и другими видами серверов (например сервером баз данных), называется шлюзом или CGI-шлюзом. Наличие CGI-скриптов на стороне Web-сервера позволяет, в частности, перенести часть логики приложения из клиента в сервер. CGI-шлюзы представляют собой средство для организации трехзвенной (в общем случае, многозвенной) архитектуры клиент-сервер.

В спецификациях CGI предусмотрены четыре способа взаимодействия Web-сервера и CGI-скрипта (по крайней мере в среде ОС Unix). Первый способ состоит в использовании создаваемых сервером переменных окружения, через которые передается как общая информация, независящая от функциональных особенностей CGI-скрипта (например, имя и версия Web-сервера), так и специфические данные, определяющие поведение CGI-скрипта (скажем набор значений, введенных в форму на стороне клиента). Второй способ заключается в формировании параметров argc и argv, которые передаются функции main CGI-скрипта (как если бы CGI-скрипт вызывался командной строкой в интерактивном режиме). Этот способ применяется для реализации ограниченного класса запросов ISINDEX. Наконец, входные параметры могут передаваться CGI-скрипту через файл стандартного ввода, а CGI-скрипт может передавать Web-серверу результирующие данные через файл стандартного вывода.

Как видно, при использовании CGI вся интерпретация пользовательского запроса производится CGI-скриптом. Скрипт может быть предельно жестким, ориентированным на выполнение запроса к фиксированной таблице фиксированной базы данных, или относительно гибким, способным выполнить произвольный запрос к одной или нескольким таблицам базы данных, идентифицируемой в параметрах клиента.

Что касается API, то для людей, знающих одновременно Unix и операционные системы вообще, ситуация предельно ясна. Запуск независимого "тяжеловесного" процесса стоит немало. Загрузка и выполнение еще одной программы в уже существующем адресном пространстве по сравнению с предыдущем вариантом гораздо дешевле. Нет никаких новых идей. API - это, фактически, дешевый способ (хотя и небезопасный) выполнить в адресном пространстве сервера WWW программу, которая соответствует спецификациям на языке HTML (так как внешние программы незнакомы серверу). Такая программа должна быть заранее подготовлена и включена в библиотеку, из которой сервер может производить динамическую загрузку (например, такая библиотека может быть разделяемой - shared library). Таким образом. смысл не меняется, меняется только реализация.

3.4. Клиентская часть

Для создания клиентской части необходимо создать HTML-документ, в котором реализован интерфейс с пользователем. В языке HTML это возможно посредством форм.

Видимо, наиболее мощные средства обеспечения доступа к базам данных на стороне Web-клиента обеспечивает язык Java. Java - это объектно-ориентированный язык программирования, являющийся, по сути дела, "безопасным" подмножеством языка Си++. В частности, Java не содержит средств адресной арифметики, не поддерживает механизм множественного наследования и т.д. Поэтому утверждается, что корректность Java-программы можно проверить до ее реального выполнения. Различают:

- язык Java, как таковой, для которого существуют компиляторы в так называемый "мобильный код" (машинно-независимый код, который может интерпретироваться или из которого могут генерироваться машинные коды на разных платформах);
- язык JavaScript, который обычно используется для расширения возможностей языка HTML за счет добавления процедурной составляющей;
- программный продукт HotJava, являющийся, по сути, интерпретатором мобильных кодов Java.

Для обеспечения доступа к базам данных на стороне Web-клиента наиболее существенно наличие языка Java. Технология разработки HTML-

документа позволяет написать произвольное количество дополнительных Java-программ, откомпилировать их в мобильные коды и поставить ссылки на соответствующие коды в теле HTML-документа. Такие дополнительные Java-программы называются апплетами (Java-applets). Получив доступ к документу, содержащему ссылки на апплеты, клиентская программа просмотра запрашивает у Web-сервера все мобильные коды. Коды могут начать выполняться сразу после размещения в компьютере клиента или быть активизированы с помощью специальных команд.

Поскольку апплет представляет собой произвольную Java-программу, то, в частности, он может быть специализирован для работы с внешними базами данных. Более того, система программирования Java включает развитый набор классов, предназначенных для поддержки графического пользовательского интерфейса. Опираясь на использование этих классов, апплет может получить от пользователя информацию, характеризующую его запрос к базе данных, в том же виде, как если бы использовался стандартный механизм форм языка HTML, а может применять какой-либо другой интерфейс.

Для взаимодействия Java-апплета с внешним сервером баз данных разработан специализированный протокол JDBC, который, фактически, сочетает функции шлюзования между интерпретатором мобильных Java-кодов и ODBC, а также включает ODBC.

3.5. Сравнение клиентского и серверного подхода

Сравнивая достоинства и недостатки двух рассмотренных подходов, можно сделать следующие выводы. Использование CGI-скриптов на стороне Web-сервера позволяет иметь на стороне клиента только сравнительно простые программы просмотра. Вся хитроумная логика работы с базами данных (возможно, с обработкой полученных данных) переходит на сторону Web-сервера. Это легкий способ построения трехзвенной архитектуры приложения. В последнее время разгрузку клиента от логики приложения называют решением проблемы «толстого» клиента. В данном случае мы можем иметь предельно «тонкого» клиента при утолщении сервера. Отрицательным моментом является то, что при необходимости подключения нового CGI-скрипта, вообще говоря, требуется относительная модификация кода сервера.

Использование Java-апплетов, вообще говоря, обеспечивает более гибкое решение. Апплет - это часть HTML-документа. Для включения нового апплета нужно только перекомпоновать документ. Web-сервер трогать не нужно. С другой стороны, клиент станет «толще». В общем случае клиент должен быть достаточно "толстым", чтобы в приемлемое время справиться с интерпретацией всех апплетов. Но, конечно же, сервер по-прежнему должен быть "толще" клиента.

На самом деле и при применении первого подхода, и при использовании второго остается нерешенной одна организационно-

производственная проблема: кто должен проектировать, писать, отлаживать и сопровождать процедурный код? Web-мастера, создающие HTML-документы, обычно выполняют дизайнерскую работу и не умеют программировать. Здесь же требуется чисто программистская работа. Иметь же двух мастеров (дизайнера и программиста) на одном Web-site - это достаточно дорого. Выход может быть в том, чтобы обучать специалистов и дизайнерским навыкам и Web-программированию.

Устремления разработчиков Web-технологий в настоящее время направлены на создание удобного и развитого интерфейса интегрированных информационных систем, который бы сочетал в себе черты гипертекста и баз данных. Возможно, он будет называться "универсальный сервер баз данных".

3.6. Методы HTTP запроса

Для реализации взаимодействия "клиент-сервер" важно, какой метод HTTP запроса использует клиентская часть при обращении к WWW серверу. В общем случае, запрос - это сообщение, посылаемое клиентом серверу. Первая строка HTTP запроса включает в себя метод, который должен быть применен к запрашиваемому ресурсу, идентификатор ресурса (URI-Uniform Resource Identifier), и используемую версию HTTP-протокола. В данном нами случае, клиентская часть применяет методы запроса POST и GET. Метод POST используется для запроса серверу, чтобы тот принял информацию, включенную в запрос, как относящуюся к ресурсу, указанному идентификатором ресурса. Метод GET используется для получения любой информации, идентифицированной идентификатором ресурса в HTTP запросе.

Для WWW-сервера стандарта NCSA прикладные программы или CGI-модули, обрабатывающие поток данных от клиента или (и) формирующие обратный поток данных, могут быть написаны на таких языках программирования, как:

- C/C++;
- Любой UNIX shell;
- Fortran;
- Perl;
- Visual Basic;
- TCL;
- AppleScript.

4. Основы использования WWW-технологий для доступа к базам данных

4.1. Проблема баз данных и WWW-технологии

Многие компании и организации используют электронные базы данных (БД) для поддержки своих рабочих процессов. Часто это системы

на одного-двух пользователей, выполненные с использованием файлов таких баз данных, как Clipper, Dbase, FoxPro, Paradox, Access. Обычно они используется независимо друг от друга. Если информация, хранимая в таких БД, представляет интерес не только для непосредственных пользователей, то для ее дальнейшего распространения используются бумажные отчеты и справки, созданные системой управления базой данных.

С появлением локальных сетей, подключением их к Интернет, созданием корпоративных сетей появляется возможность с любого рабочего места организации получить доступ к информационному ресурсу сети. Однако, при попытке использовать существующие БД возникают проблемы, связанные с требованием к однородности рабочих мест (для запуска "родных" интерфейсов), сильнейшим трафиком в сети (доступ идет напрямую к файлам БД), загрузкой файлового сервера и невозможностью удаленной работы (например, командированных сотрудников). Решением проблемы могло бы стать использование унифицированного интерфейса WWW для доступа к ресурсам организации.

Технология World Wide Web получила столь широкое распространение из-за простоты своих пользовательских интерфейсов. Принцип "нажми на то, что интересно", лежащий в основе гипертекста, интуитивно понятен. В технологиях WWW все ключевые понятия просматриваемого документа: слова, картинки - имеют возможность "раскрыться" новым документом, развивающим это понятие. Из этих предпосылок возникает задача преобразования накопленных данных в гипертекстовые документы WWW, задача поддержки актуальности преобразованной структуры. Другими словами, задача предоставления WWW-доступа к существующим базам данных.

Базы данных содержат основные знания человечества. В конце двадцатого века с появлением технологии баз данных мы накопили больше информации, чем за всю предыдущую историю. Вся беда в том, что доступ к базам данных (даже к тем, которые содержат полностью открытую информацию) ограничен. Чтобы получить интересующую его информацию, пользователь должен иметь физический доступ к соответствующей СУБД, быть в курсе модели данных, знать схему базы данных и, наконец, уметь пользоваться соответствующим языком запросов. Что касается языка запросов, то проблему частично решает протокол ODBC, позволяющий направлять ограниченный набор операторов SQL (с промежуточной обработкой соответствующим драйвером ODBC) к произвольному серверу баз данных. Но это только частичное решение, поскольку оно никак не помогает пользователю понять схему базы данных (даже в терминах SQL) и, конечно, не способствует созданию унифицированного интерфейса конечного

пользователя (нельзя же заставить всех работать в строчном режиме на языке SQL).

Итак, что же имеется. Есть удобные средства разработки распределенных в Интернет гипермедийных документов, простые, удобные, развитые и унифицированные интерфейсы для доступа к информации WWW. Кроме того, мы имеем большое количество ценных баз данных, управляемых разнородными СУБД, а также желание сделать эти базы доступными всем людям (в случае публичных баз данных) или членам территориально-распределенной корпорации (в случае корпоративных баз данных). Возникает естественное желание скрестить эти две технологии и обеспечить доступ к базам данных в интерфейсе Web. Еще два года назад существовали только идеи такого скрещивания и не очень тщательно разработанные подходы к реализации. На сегодняшний день имеется несколько работающих механизмов, и далее они обсуждаются. В целом механизмы делятся на два класса: обеспечивающие доступ к базе данных (по запросу клиента) на стороне Web-сервера и работающие непосредственно на стороне клиента.

4.2. Компоненты технологии WWW для доступа к информационным ресурсам

Использование технологий WWW для обеспечения доступа к каким-либо информационным ресурсам подразумевает существование следующих компонент (рис. 8).

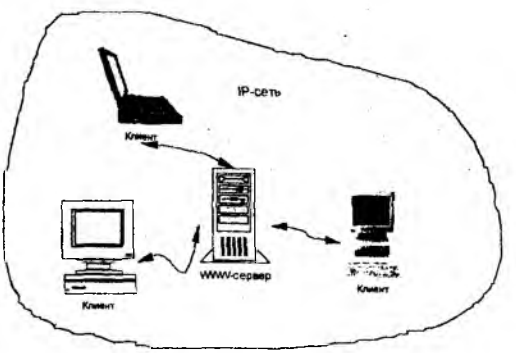


Рис. 8. Компоненты IP-сети

IP - сети с поддержкой базового набора услуг по передаче данных с единой политикой нумерации и маршрутизации, работающим сервисом имен DNS.

Выделенного информационного сервера - WWW-сервера, обеспечивающего предоставление гипертекстовых документов через IP - сеть в ответ на запросы WWW - клиентов.

Передаваемые гипертекстовые документы оформляются в стандарте HTML - языке описания гипертекстовых документов. Эти документы могут либо храниться в статическом виде (совокупность файлов на диске), либо динамически компоноваться в зависимости от параметров запроса специальным программным обеспечением. Для динамической компоновки HTML-документов, WWW-сервер использует специальным образом оформленные программы.

4.3. Варианты www-доступа к базам данных

В состав специфики конкретной БД входят как технологические основы, такие как тип СУБД, вид интерфейсов, связи между таблицами, ограничения целостности, так и организационные решения, связанные с поддержкой актуальности баз данных и обеспечением доступа к ней.

При обеспечении WWW-доступа к существующим БД, возможен ряд *путей - комплексов* технологических и организационных решений. Практика использования WWW-технологии для доступа к существующим БД предоставляет широкий спектр технологических решений, по-разному связанных между собой - перекрывающих, взаимодействующих и т.д. Выбор конкретных решений при обеспечении доступа зависит от специфики конкретной СУБД и от ряда других факторов, как то: наличие специалистов, способных с минимальными издержками освоить определенную ветвь технологических решений, существование других БД, WWW-доступ к которым должен осуществляться с минимальными дополнительными затратами и т.д.

WWW - доступ к существующим базам данных может осуществляться по одному из трех основных сценариев. Ниже дается их краткое описание и основные характеристики.

Вариант 1. Однократное или периодическое преобразование содержимого БД в статические документы.

В этом варианте содержимое БД просматривает специальная программа, создающая множество файлов - связных HTML-документов (рис. 9). Полученные файлы могут быть перенесены на один или несколько WWW-серверов. Доступ к ним будет осуществляться как к статическим гипертекстовым документам сервера.

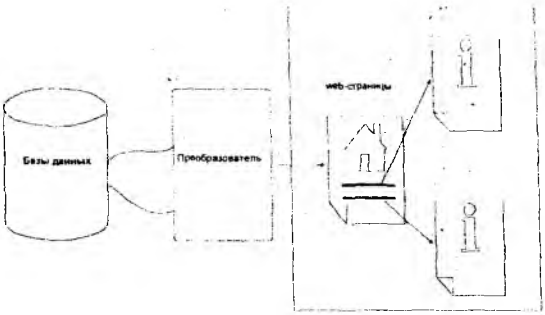


Рис. 9. Однократное или периодическое преобразование содержимого базы данных в статические документы

Этот вариант характеризуется минимальными начальными расходами. Он эффективен на небольших массивах данных простой структуры и редким обновлением, а также при пониженных требованиях к актуальности данных, предоставляемых через WWW. Кроме этого, очевидно полное отсутствие механизма поиска, хотя возможно развитое индексирование.

В качестве преобразователя может выступать программный комплекс, автоматически или полуавтоматически генерирующий статические документы. Программа-преобразователь может являться самостоятельно разработанной программой, либо быть интегрированным средством класса генераторов отчетов.

Вариант 2. Динамическое создание гипертекстовых документов на основе содержимого БД

Рис. 10. Динамическое создание гипертекстовых документов на основе содержимого базы данных

В этом варианте доступ к БД осуществляется специальной CGI-программой, запускаемой WWW-сервером в ответ на запрос WWW -

клиента. Эта программа, обрабатывая запрос, просматривает содержимое БД и создает выходной HTML-документ, возвращаемый клиенту (рис.10).

Это решение эффективно для больших баз данных со сложной структурой и при необходимости поддержки операций поиска. Показаниями также являются частое обновление и невозможность синхронизации преобразования БД в статические документы с обновлением содержимого. В этом варианте возможно осуществлять изменение БД из WWW-интерфейсов.

К недостаткам этого метода можно отнести большое время обработки запросов, необходимость постоянного доступа к основной базе данных, дополнительную загрузку средств поддержки БД, связанную с обработкой запросов от WWW - сервера.

Для реализации такой технологии необходимо использовать взаимодействие WWW-сервера с запускаемыми программами CGI - *Common Gateway Interface*. Выбор программных средств достаточно широк - языки программирования, интегрированные средства типа генераторов отчетов. Для СУБД с внутренними языками программирования существуют варианты использования этого языка для генерации документов.

Вариант 3. Создание информационного хранилища на основе высокопроизводительной СУБД с языком запросов SQL. Периодическая загрузка данных в хранилище из основных СУБД

В этом варианте предлагается использование технологии, получившей название "информационного хранилища" (ИХ). Для обработки разнообразных запросов, в том числе и от WWW-сервера, используется промежуточная БД высокой производительности. Информационное наполнение промежуточной БД осуществляется специализированным программным обеспечением на основе содержимого основных баз данных (рис. 11).

Этап 1. Перезагрузка данных

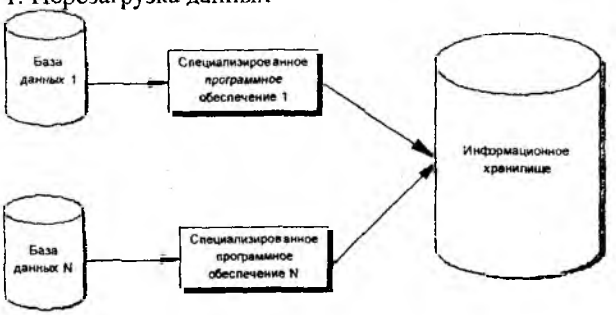


Рис. 11. Перегрузка данных

Этап 2. Обработка запросов

Рис. 12. Обработка запросов

Данный вариант свободен ото всех недостатков предыдущей схемы. Более того, после установления синхронизации данных информационного хранилища с основными БД возможен перенос пользовательских интерфейсов на информационное хранилище, что существенно повысит надежность и производительность, позволит организовать распределенные рабочие места.

Несмотря на кажущуюся громоздкость такой схемы, для задач обеспечения WWW-доступа к содержимому нескольких баз данных накладные расходы существенно уменьшаются.

Основой повышения производительности обработки WWW-запросов и резкого увеличения скорости разработки WWW-интерфейсов является использование внутренних языков СУБД информационного хранилища для создания гипертекстовых документов.

Для загрузки содержимого основной БД в информационное хранилище могут использоваться все перечисленные решения (языки программирования, интегрированные . средства), а также специализированные средства перегрузки, поставляемые с SQL-сервером и продукты поддержки информационных хранилищ.

Трудоемкость обеспечения WWW-доступа к базам данных, очевидно, складывается из трудоемкости работ при реализации одного из вышеприведенных вариантов. Реализация первого варианта связана с последовательным преобразованием всех данных, находящихся в исходной БД. Разработка средств вывода содержимого таблицы в формате HTML с необходимым форматированием и текстовым сопровождением будет занимать порядка 1-3-х дней для одного разработчика. Разработка средств построения индексной структуры к выводимым данным является более творческой работой, и может занять 1-3 недели для одного разработчика.

Трудоемкость построения интерфейсов для вариантов 2, 3, в общем случае, эквивалентна трудоемкости построения этих интерфейсов при создании исходной информационной системы (т.е. той, для которой обеспечивается WWW-доступ) с использованием традиционных средств разработки (не - CASE). В третьем варианте дополнительные трудозатраты пойдут на перегрузку данных в информационное хранилище. При перегрузке данных без изменения структуры и имен можно исходить из оценки трудозатрат: 1-2 таблицы в 1-2 дня для одного разработчика, в зависимости от сложности и объема таблиц, при условии отладки технологии перегрузки.

При использовании различных средств разработки интерфейсов к БД, трудозатраты могут существенно различаться. Ранжированный по уменьшению трудозатрат на разработку интерфейсов список будет выглядеть так:

- библиотеки и функции на языке C;
- язык Perl;
- пакеты WOW и Cold Fusion.

4.4. Средства доступа к базам данных на стороне сервера

- CGI
- API
- FastCGI

CGI

Common Gateway Interface - это спецификация интерфейса взаимодействия Web-сервера с внешними прикладными программами. Главное назначение CGI - обеспечение единообразного потока данных между сервером и работающим на нем приложением. CGI определяет:

- порядок взаимодействия сервера с прикладной программой, в котором сервер выступает инициирующей стороной;
- механизм реального обмена данными и управляющими командами в этом взаимодействии, что не определено в протоколе HTTP. Такие понятия, как метод доступа, переменные заголовка, MIME, типы данных, заимствованы из HTTP и делают спецификацию прозрачной для тех, кто знаком с самим протоколом.

Обычно гипертекстовые документы, возвращаемые по запросу клиента WWW сервером, содержат статические данные. CGI обеспечивает средства создания динамических Web-страниц на основе данных, полученных от пользователя. Программы, написанные в соответствии со спецификацией CGI, называются CGI-скриптами или шлюзами. Шлюз - это CGI-скрипт, который используется для обмена данными с другими информационными ресурсами Интернет или приложениями-демонами такими, как, например, система управления базами данных. Обычная CGI-

программа запускается Web-сервером для выполнения некоторой работы, возвращает результаты серверу и завершает свое выполнение (рис.13).

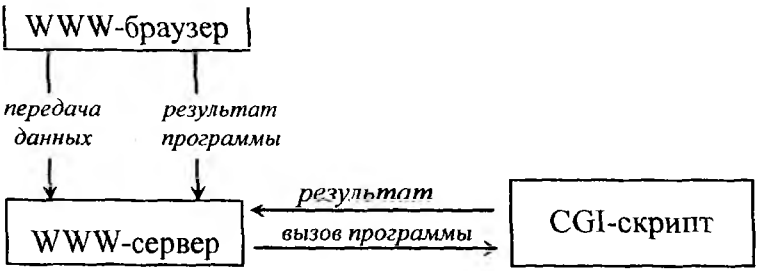


Рис. 13. Схема взаимодействия CGI-скрипта

Шлюз выполняется точно также, только, фактически, он инициирует взаимодействие в качестве клиента с третьей программой (рис.14). Если эта третья программа является сервером БД, то шлюз становится клиентом СУБД, который посылает запрос по определенному порту соединения с системой управления базами данных, а после получения ответа пересылает его WWW-серверу.

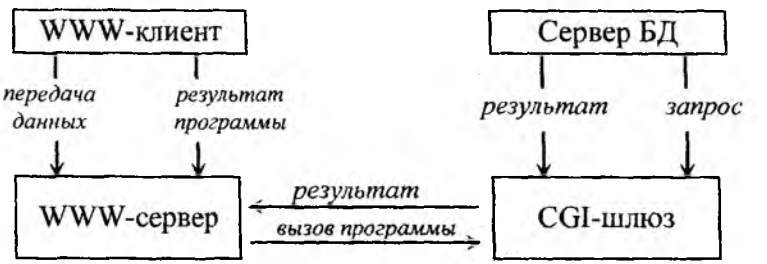


Рис.14. Схема взаимодействия CGI-шлюза

Обмен данными по спецификации CGI реализуется обычно через переменные окружения и стандартный ввод/вывод. Выбор механизма передачи параметров определяется методом доступа, который указывается в форме в атрибуте METHOD. Если используется метод GET, то передача параметров происходит с помощью переменных окружения, которые сервер создает при запуске внешней программы. Через них передается приложению как служебная информация (версия программного обеспечения, доменное имя сервера и др.), так и сами данные (в переменной QUERY_STRING). При методе POST для передачи используется стандартный ввод. А в переменных окружения фиксируются тип и длина передаваемой информации (CONTENT_TYPE и CONTENT_LENGTH).

Стандартный вывод используется скриптом для возврата данных серверу. При этом вывод состоит из заголовка и собственно данных. Результат работы скрипта может передаваться клиенту без каких-либо преобразований со стороны сервера, если скрипт обеспечивает построение полного HTTP-заголовка, в противном случае сервер модифицирует заголовок в соответствии со спецификацией HTTP. Обязательным для скриптов при генерировании документов "на лету", когда реального документа в файловой системе сервера не остается является только HTTP-заголовок *Content-type*, в котором указывается тип возвращаемого документа для правильной интерпретации браузером. Обычно в Content-type указывают текстовые типы text/plain и text/html. При использовании такого вида скриптов следует учитывать, что не все серверы и клиенты отрабатывают так, как представляется разработчику скрипта. Так, при указании Content-type: text/html, некоторые клиенты не реализуют сканирования полученного текста на предмет наличия в нем встроенной графики.

- При применение спецификаци CGI для обмена данными с внешними прикладными программами можно выделить следующие преимущества:
- Прозрачность использования.
- "Языковая" независимость - CGI-программы могут быть написаны на любом языке программирования или командном языке, имеющим средства работы со строками.
- Процессная изолированность - при запуске CGI-программы на сервере порождается отдельный процесс и ошибочный CGI-скрипт не может сломать Web-сервер или получить доступ к закрытой информации.
- Открытость стандарта - CGI интерфейс применим на каждом Web-сервере.
- Архитектурная независимость - CGI не зависит от особенностей реализации архитектуры сервера (однопоточности, многопоточности и т.д.). Но CGI имеет также и существенные недостатки. Главная проблема заключается в затратах на выполнение CGI-приложений: поскольку на сервере для каждого очередного запроса порождается новый процесс, который завершается после его выполнения, то это приводит к невысокому быстродействию CGI-скрипта и снижает эффективность работы сервера. При использовании CGI-программ для доступа к базам данных из-за неподдержки непрерывного соединения Web-сервера и соответствующей СУБД очень сложно произвести процесс "ведения" пользователя базой данных, так как каждый раз при генерации очередного запроса требуется новое подключение. Но в то же время закрытие соединения после обработки каждого запроса сильно осложняет деятельность взломщиков, так как при отсутствии постоянного подключения к БД проникнуть в нее гораздо сложнее. Другое достоинство этого "недостатка" состоит в том, что связь с Web-сервером устанавливается только на короткий промежуток

времени, в результате чего он не перегружается и может выполнять другие задачи.

CGI также ограничен по способности функционирования - спецификация предусматривает только простую "ответную" роль скрипта при генерации результата на запрос пользователя. CGI-программы не имеют взаимосвязей с установлением аутентификации пользователя и проверки его входных данных.

API

В ответ на ограничения и недостатки спецификации CGI была разработана спецификация прикладных модулей API, встроенных в сервер. Данное расширение Web-сервера запускается как динамическая библиотека и выполняет обработку каждого вызова сервера по отдельной структуре памяти, что значительно проще, чем создание отдельного процесса для каждого клиентского запроса. Наиболее известны два API-интерфейса - NSAPI компании Netscape и ISAPI компании Microsoft. Свободно распространяемый популярный Unix-сервер Apache также имеет модуль PHP, реализующий данный интерфейс. Приложения, работающие через API, соединяются с сервером значительно быстрее, чем CGI-программы, так как API выполняется в основном процессе сервера и постоянно находится в состоянии ожидания запросов, поэтому время на запуск программы и порождение нового процесса не требуется. API-интерфейс предоставляет и большую функциональность, чем CGI. Можно написать дополнительные процедуры, осуществляющие контроль доступа к файлам, получающие доступ к log-файлам сервера и связывающиеся с другими этапами обработки запроса сервером.

Тем не менее спецификация API не имеет преимуществ CGI-интерфейса и поставщики API-модулей тоже сталкиваются с целым рядом проблем:

- "Языковая" зависимость - прикладные программы могут быть написаны только на языках, поддерживаемых в данном API (обычно это C или/C++); Perl, наиболее популярный язык для CGI-скриптов, как правило, не используется в существующих поставляемых API-модулях.
- Неизолированность процесса - так как приложения выполняются в адресном пространстве сервера, то ошибочные программы могут "уронить" сервер или какое-либо приложение. Таким образом, вполне возможно (намеренно или нет) сломать систему безопасности сервера.
- Ограниченность применения - написанные программы в соответствии с данным API могут использоваться только на данном сервере.
- Архитектурная зависимость - API-приложения зависимы от архитектуры сервера: если сервер поддерживает однопоточность, то многопотоковые приложения не получают никакого преимущества в быстродействии при выполнении. Также при изменении производителем

архитектуры сервера, модуль API обычно тоже подвергается изменениям, и прикладные программы соответственно тоже требуют переделки или даже могут быть написаны заново.

FastCGI

Интерфейс FastCGI сочетает в себе наилучшие аспекты спецификаций CGI и API. Взаимодействие в соответствии с FastCGI происходит сходным образом с CGI. FastCGI-приложения запускаются отдельными изолированными процессами. Отличие состоит в том, что эти процессы являются постоянно работающими и после выполнения запроса не завершаются, а ожидают новых запросов. Вместо использования переменных окружения операционной системы и стандартных потоков ввода/вывода протокол FastCGI объединяет информацию среды, стандартный ввод, вывод и сообщения об ошибках в единственное дуплексное соединение. Это позволяет FastCGI-программам выполняться на удаленных машинах, используя TCP-соединения между Web-сервером и FasstCGI-модулем.

Таким образом, преимущества FastCGI состоят в следующем:

- Быстродействие - благодаря постоянному функционированию FsatCGI-процессов обеспечивается обслуживание одним процессом многих запросов, что решает задачу и связанные с ней проблемы порождения новог о процесса на отдельный клиентский запрос.
- Простота применения и легкость миграции из CGI.
- "Языковая" независимость - как и CGI, FastCGI-приложения могут быть написаны на любых языках программирования или командных языках.
- Изолированность процессов - "неисправные" FastCGI-программы не могут разрушить ядро сервера или какие-либо другие приложения, а также получить секретную служебную информацию.
- Совместимость - FastCGI поддерживается во всех открытых продуктах, включая коммерческие серверы Netscape и Microsoft, NCSA сервер и свободно распространяемый Apache.
- Архитектурная независимость - FastCGI интерфейс не зависит от особенностей реализации серверной архитектуры и прикладные программы могут быть как одно-, так и многопоточными.
- Распределенность - FastCGI обеспечивает возможность выполнять приложения удаленно, что используется для распределенной загрузки и управления внешними Web-сайтами.

4.5. Доступ к базам данных на стороне клиента. Java-технология

Для обеспечения доступа к базам данных на стороне Web-клиента применяется Java-технология. Java - это современный объектно-ориентированный язык программирования для разработки приложений,

созданный специально для распределенных сред. Технология Java позволяет создавать полноценные приложения для работы с компьютерной графикой, файловыми системами и компьютерными сетями.

Одно из важных свойств Java-технологии - это мобильность, суть которой заключается в том, что написанный на Java код может исполняться на любой компьютерной платформе. Java-приложения компилируются в особый код (так называемый байт-код), исполняемый на виртуальной машине (Java Virtual Machine). Байт-код является универсальным форматом программы, единым для всех аппаратных платформ - и для рабочих станций, и для больших универсальных ЭВМ, и для персональных компьютеров. Java-технология обеспечивает быстрый цикл компиляции и отладки программ. Еще на стадии компиляции проводится выявление многих ошибок и частичная оптимизация программ. Средства разработки, содержащие виртуальную машину внутри себя, обеспечивают контроль приложений на стадии исполнения (переполнение стека, отслеживание границ массивов, поиск резервов для оптимизации и др.).

Пользователю готовых Java-приложений достаточно иметь клиентскую программу, имитирующую работу виртуальной машины. Виртуальная машина представляет собой довольно компактный интерпретатор байт-кода Java. Перед первым запуском нового приложения виртуальная машина проверяет его код на принадлежность к байт-коду (на правильность инструкций Java), безопасность команд для компьютера и локальной сети, соответствие разрешенным операциям, а также на целый ряд дополнительных условий. Это необходимо, поскольку приложения, распространяемые по сети, создаются разными людьми с различными намерениями, причем дурные намерения тоже не исключены. Непосредственно перед запуском виртуальная машина производит сборку модулей и устанавливает связи между именами, при этом поиск недостающих модулей производится не только в системе, но и на серверах Интернет. Затем, собственно, и начинается работа приложений.

Технология разработки HTML-документа позволяет написать произвольное количество дополнительных Java-программ, откомпилировать их в мобильные коды и поставить ссылки на соответствующие коды в теле HTML-документа. Такие дополнительные Java-программы называются апплетами (Java-applets). Получив доступ к документу, содержащему ссылки на апплеты, клиентская программа просмотра запрашивает у Web-сервера все мобильные коды. Коды могут начать выполняться сразу после размещения в компьютере клиента или быть активизированы с помощью специальных команд.

Поскольку апплет представляет собой произвольную Java-программу, то, в частности, он может быть специализирован для работы с внешними базами данных. Опираясь на использование классов, апплет

может получить от пользователя информацию, характеризующую его запрос к базе данных, в том же виде, как если бы использовался стандартный механизм форм языка HTML, а может применять какой-либо другой интерфейс.

Для взаимодействия Java-апплета с внешним сервером баз данных разработан специализированный протокол JDBC, который, фактически, сочетает функции шлюзования между интерпретатором мобильных Java-кодов и интерфейсом ODBC (Open Data Base Connectivity). JDBC - это разработанный JavaSoft прикладной программный SQL интерфейс API JDBC к базам данных. Этот API позволяет использовать стандартный набор процедур высокого уровня для доступа к различным БД.

JDBC базируется на интерфейсе уровня вызовов X/Open SQL CLI - основе ODBC. Прикладной программный интерфейс JDBC реализуется поверх других SQL-API, включая ODBC. То есть, все базы данных, допускающих работу с ODBC, будут взаимодействовать с JDBC без изменений. При использовании JDBC Интернет-пользователи подключаются к Web-серверу и загружают HTML-документ с апплетом. Апплет выполняется на клиентской ЭВМ в среде браузера и устанавливает связь с сервером базы данных. Механизм связи с базами данных является классом Java.

Архитектура JDBC состоит из двух уровней: JDBC API, который обеспечивает связь между приложением и менеджером JDBC, и драйвер JDBC API, который поддерживает связь между JDBC менеджером и драйвером (рис.15). Разработчики имеют возможность взаимодействовать напрямую с ODBC посредством моста JDBC-ODBC.

JDBC API представляет собой набор классов (пакет или package) для установки соединений с базами данных, создания SQL-выражений, обработки результатов. JDBC-драйвера могут быть написаны на Java и загружены как часть апплета или быть написаны используя "родной" код компьютера (как шлюз к существующим библиотекам СУБД API). Примером такого шлюза является JDBC-ODBC мост (JDBC-ODBC bridge). Он транслирует JDBC запросы в вызовы ODBC, что позволяет получить доступ к огромному множеству существующих ODBC драйверов.

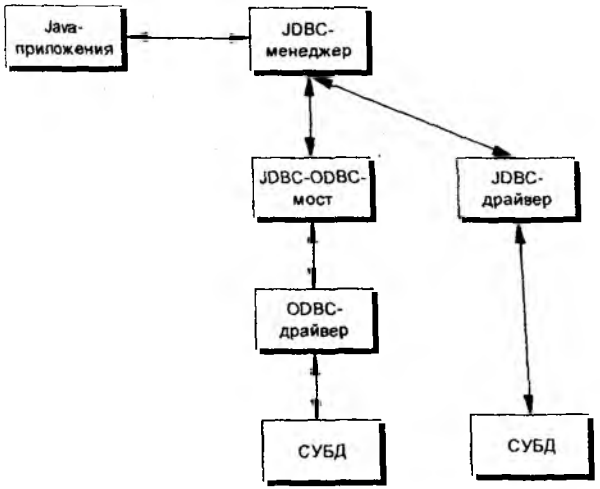


Рис. 15. Схема взаимодействия Java-приложений с сервером БД

Использование Java-апплетов в общем обеспечивает более гибкое решение. Так как апплет - это часть HTML-документа, то для включения нового апплета нужно всего-навсего перекомпоновать документ без привлечения Web-сервера. Апплеты передаются по сети только в тот момент, когда их нужно выполнять. При этом они могут загружаться из разных мест, и после загрузки взаимодействовать друг с другом.

С другой стороны, байт-код Java исполняется интерпретатором, а не является откомпилированной на данной платформе программой. Отсюда возникает первый очевидный недостаток - это скорость выполнения кода, так как интерпретатор работает гораздо медленнее откомпилированной программы. Собственно, и другие свойства технологии (объектная ориентируемость, использование многопоточности, отсутствие адресной арифметики и т.п.) в большинстве случаев при стандартной комплектации оборудования, только тормозят выполнение программы.

Основной протокол обмена апплетами - HTTP. Это значит, что они передаются по сети точно также, как и другие ресурсы Website, и приобретают свое особое значение только в момент получения их браузером. Учитывая неэффективность реализации HTTP поверх TCP, можно сказать, что и обмен апплетами тоже неэффективен, если для каждого обмена устанавливается свое TCP-соединение. В любом случае загрузка страниц с Java происходит гораздо медленнее, чем без Java.

Java-технология является еще технологией, находящейся в стадии разработки. Интерфейс взаимодействия прикладных программ CGI имеет уже достаточный опыт в применении для обработки данных и доступа к БД. По сравнению с Java, он является не только отлаженным механизмом,

но и более простым и удобным средством для разработчиков CGI-программ, так как они могут быть созданы на любом языке, имеющим средства работы со строками.

Выбор средства доступа к базам данных во многом определяется не только эффективностью того или иного механизма, но также и наилучшим его сопряжением с соответствующей СУБД. От того, какие средства предоставляет сама СУБД для доступа к своим базам данных из внешних прикладных программ может зависеть выбор предпочтения. Сейчас разработано достаточно много коммерческих СУБД, но все же хочется обратить внимание на свободно распространяемые продукты, которые часто оказываются не менее эффективными, но из-за "неизвестности" не достаточно широко используются. Одним из таких некоммерческих продуктов является СУБД POSTGRES95, которая устанавливается на большинстве существующих платформ -DEC Alpha AXP on OSF/1 2.0, HP PA-RISC on HP-UX 9.0, i386 Solaris, SUN SPARC on Solaris 2.4, SUN SPARC on SunOS 4.1.3, DEC MIPS on Ultrix 4.4, Intel x86 on Linux 1.2 and Linux ELF, BSD/OS 2.0, 2.01, 2.1, IBM on AIX 3.2.5, SGI MIPS on IRIX 5.3, DG/UX 5.4R3.10 и др.

4.6. Программные средства технологии WWW-базы данных

4.6.1. WWW - сервер NCSA HTTPD

Одним из ключевых элементов технологии WWW является WWW-сервер. Стандартом де-факто для Unix-систем стало программное обеспечение (ПО) WWW-сервера Национального Центра по Суперкомпьютерным Приложениям (NCSA) Иллинойского Университета. Все вновь создаваемые продукты поддерживают полную совместимость с ПО NCSA по режимам работы и форматом данных. Сервер NCSA является постоянно совершенствуемым продуктом, отражающим последние веяния WWW-технологии. Созданная относительно недавно "Apache Group" разрабатывает свое программное обеспечение WWW - сервера на базе продукта NCSA HTTPD.

4.6.2. SQL - сервер фирмы Oracle

При реализации сценария 3 встает вопрос о выборе качественной платформы для создания информационного хранилища. Реляционная система управления базами данных фирмы Oracle является лидером на рынке СУБД. По производительности, надежности хранения данных, развитию семейства интерфейсов, объему серверных платформ продукты Oracle возглавляют многочисленные рейтинги. Гибкость использования, развитые средства управления доступом и распределенная архитектура делают сервер Oracle чрезвычайно привлекательным для технологии информационных хранилищ, а возможность работы на свободно распространяемых Unix-платформах расширяет его возможности в некоммерческой среде.

Существенным ограничением использование Oracle в сфере науки и образования является достаточно высокая цена и низкое бюджетное финансирование. Однако с 1996 года фирма Oracle объявила о специальной программе для российских университетов, что позволяет за относительно небольшие деньги приобрести любой набор продуктов Oracle.

4.6.3. Библиотеки и функции на языке Си

Одной из основных технологий создания CGI-модулей для реализации функций "преобразователя" и "обработчика" сценариев 1-3 является язык Си. Язык Си - наиболее распространенный язык программирования. При решении описанных задач язык Си можно использовать для создания следующих программ:

- преобразователя, однократно преобразующего содержимое БД в сеть гипертекстовых документов;
- обработчика, динамически обрабатывающего запрос от WWW-сервера к БД;
- перегрузчика из существующих БД в информационное хранилище;
- обработчика запросов от WWW-сервера к информационному хранилищу.

Для поддержки этих функций создано большое количество библиотек и функций языка Си, готовых приложений в исходных текстах.

4.6.4. Язык программирования Perl

Язык Perl был создан для повышения эффективности обработки текстовых документов. Он ориентирован на обработку строк. В настоящее время язык получил большое распространение как инструмент создания исполняемых модулей WWW-сервера. Существующие пакеты расширения обеспечивают доступ к SQL-серверам непосредственно из Perl-программы. Это позволяет использовать его для решения всех задач, возникающих при обеспечении WWW-доступа к базам данных. Perl эффективен также при обработке произвольных структур данных: существующих отчетов, списков, карточек в электронном виде.

4.6.5. Пакет Web - Oracle - Web

Пакет WOW является свободно распространяемым программным средством, предназначенным для создания интерактивных WWW-интерфейсов с СУБД Oracle. Пакет WOW был первым и наиболее простым средством, выпущенным фирмой Oracle. В настоящее время существует набор продуктов, развивающих функциональность WOW'a - Oracle Web Server версий 1, 2, Oracle Web Arcitecture.

Все перечисленные продукты позволяют использовать процедурное расширение языка SQL - PL/SQL, разработанное фирмой Oracle для динамического создания гипертекстовых документов. Высокая скорость

разработки достигается за счет резкого упрощения доступа к БД - программы на PL/SQL исполняются самим сервером Oracle.

Основной областью использования WOW является обработка запросов от WWW-сервера к SQL-серверу Oracle в среде Unix. В предложенных сценариях пакет WOW позволит организовать эффективный WWW доступ к информационному хранилищу, построенному на базе сервера баз данных Oracle.

4.6.6. Пакет Cold Fusion фирмы Allaire Corp

Пакет предназначен для использования под ОС Windows и позволяет обращаться к различным базам данных, поддерживающим интерфейс ODBC через WWW-интерфейсы. Пакет имеет коммерческий статус, его "evaluation copy" является свободно-распространяемой. Для доступа к базам данных используются конструкции языка DBML - расширения языка HTML, дополненного средствами доступа к БД через ODBC. Документы на языке DBML обрабатываются на серверной части, в результате чего создается HTML-документ.

Пакет может эффективно использоваться в качестве обработчика запросов WWW к исходным базам данных.

5. Технологии ActiveX

5.1. Основы технологии ActiveX

В последнее время все большую популярность получает это средство создания интерактивных Web-страниц. Основным его достоинством является возможность формирования страниц на основании интерактивного процесса «клиент-сервер». Сами же программы, написанные на ASP, что программирование с их помощью доступно даже неопытным разработчикам.

Применение встраиваемых в ASP ActiveX-компонентов значительно облегчает задачу создания сайтов. Для этого необходимо подготовить шаблон базы данных с соответствующими полями для заголовка, аннотации, рубрики, даты публикации и текста самой статьи, а также HTML-файла и иллюстрации, загружаемой как в виде текста с HTML-формы, так и в виде HTML-файла. В последнем случае полностью сохраняется формат статьи, что достаточно удобно. Кроме того, статья на сайте может быть снабжена иллюстрацией, которую тоже необходимо загрузить с формы на сервер и отображать по усмотрению пользователя на главной странице сайта.

Затем необходимо организовать ввод всех полей формы, включая локальные пути (на компьютере клиента) к загружаемым файлам (в нашем случае с HTML-версией статьи и с иллюстрацией), загрузить их на сервер в определенный каталог и присвоить соответствующим полям базы данных значения.

Для реализации данной задачи необходим персональный компьютер с Microsoft Windows NT или Windows 2000 (можно и Workstation или Server), установленный IIS (Internet Information Server), HTML-редактор (хороший вариант - использовать Macromedia Dreamweaver), Microsoft Access (версии 95, 97 или 2000) и самый обычный текстовый редактор.

Далее необходимо прописать нашу базу данных в соответствующем разделе источников данных систем. Для этого:

- запустим программу-конфигуратор источников данных (Data Sources ODBC) – Start->Settings->Control Panel->Administrative Tools->Data Sources ODBC;
- перейдем во вкладку "System DSN" и создадим новый источник данных, нажав на "Add...";
- в появившемся списке драйверов выберем драйвер баз данных Microsoft Access – "Microsoft Access Driver (*.mdb)" и нажмем на "Finish";
- в строке "Data Source Name" зададим имя нашей базы данных, например "Name_database" (это то имя, по которое служит для дальнейшего обращения к ней);
- нажмем на "Select...", выберем подготовленный нами файл " Name_database.mdb" и нажмем "OK".

Появляется строка в списке источников данных в вашей системе: “Name_database” “Microsoft Access Driver (.mdb)”.

В сети Интернет можно найти множество Active-X-компонентов как свободно распространяемых, так и платных различного назначения. Например, для загрузки файла с HTML-формы можно использовать компонент AspUpload, находящийся на сайте http://www.persits.com.

К наиболее часто применяемые компоненты можно отнести следующие:

Компоненты загрузки - позволяют реализовать загрузку файлов на сервер, управлять многопользовательской системой для загрузки файлов через браузер на сервер

Почтовые компоненты - предназначены для отправления почтовых сообщений через SMTP-сервер, для соединения с POP сервером и просмотра почтовых сообщений.

Компоненты баз данных - для создания DSN(ODBC) на лету, например из формы. Такие компоненты имеют массу различных возможностей.

Компоненты графического отображения – могут генерировать gif - изображения с текстом "на лету", создавать динамические картинки, используя при этом формат GIF, JPEG, PNG или BMP, создавать анимированные gif-изображения, добавлять к изображению прозрачный текст, добавлять градиентные заполнения, иметь средства рисования геометрических фигур, обрезать, изменять размеры изображений, создавать кнопки, эффекты осветления и затемнения, загружать

изображения посредством OLE, создавать Gif - изображения из таблиц Excel, создавать clickable-карты с полной поддержкой навигации по карте.

Компоненты для авторизации – могут обеспечивать регистрацию пользователя на сервере, позволять манипулировать правами доступа, могут содержать средства для интернет-провайдеров, которым требуется разграничивать доступ пользователям для администрирования их собственных баз данных, управлять web-доступом, ограничивать доступ для каждого конкретного пользователя, обеспечивать разграничение прав к различным страницам сервера. В последнем случае пользователю потребуется авторизация перед тем, как они получат доступ к запрашиваемой странице.

Компоненты для преобразований - позволяют форматировать числа, даты и т.д. в соответствии с локальными настройками пользователя.

Компоненты для Чатов и Форумов - позволяют создавать "комнаты переговоров" в сочетании с HTML кодом.

Навигационные компоненты - позволяют создавать так называемые Tree View (списки типа дерева).

Сетевые компоненты – позволяют производить прямые и обратные запросы DNS, записывать в системный лог дополнительную информацию из ASP-программ, позволяют ASP-программе производить запросы к другим web-серверам, возвращая строку результата, позволяют получать страницы с других web-сайтов для их последующего разбора или преобразований. В последнем случае объект поддерживает два типа запросов: GET и POST, а также поддерживает переадресацию и возвращает строку для ее дальнейшей обработки.

Файловые компоненты – могут превращать обычный браузер в мощное средство управления файлами и папками на web-сервере, позволяют изменять системную дату и время целого ряда файлов проекта, осуществлять запись файлов на клиентскую машину.

Другие компоненты – существует еще ряд полезных компонент. К ним можно отнести следующие: позволяющие из ASP-скриптов оценивать текущую производительность системы, позволяющие создать "монитор производимости" сервера, позволяющая из ASP-приложений регистрировать и разрегистрировать ActiveX компоненты на сервере.

Более подробно компоненты будут рассмотрены ниже.

5.2. Клиентская технология ActiveX (Active Desktop)

Клиентская технология ActiveX (Active Desktop) ActiveX реализуется на машине-клиенте с помощью библиотек, поставляемых вместе с Internet Explorer. В дальнейшем эти библиотеки будут дополняться и переписываться, в частности, наиболее значимые обновления этих библиотек на клиенте следует ожидать после выхода NetShow, продукта, предназначенного для оптимальной передачи по сети данных мультимедиа.

Программные компоненты ActiveX могут быть установлены автоматически на компьютер пользователя по сети с удаленного сервера, причем будет загружен код, подходящий для конкретной платформы клиента, будь то Macintosh, Windows или Unix. Разработчик Web-страниц может либо сам запрограммировать элементы ActiveX, используя популярные языки программирования Visual C++, Visual Basic или Java, либо использовать существующие.

Примеры готовых программных элементов можно найти по адресу: http://www.microsoft.com/activeplatform/default.asp .

Используя языки сценариев ActiveX, программисты могут обеспечить взаимодействие различных элементов ActiveX, Java, других программ на клиентском компьютере и различных частей самого Internet Explorer. Например, программный элемент синхронизации может обновлять страницу Web через определенные промежутки времени. Можно также периодически запускать программный элемент, привлекающий внимание пользователя. Имеются реализации Visual Basic Scripting Edition, являющегося подмножеством Visual Basic и JScript. Кроме того, разработчик может написать интерпретатор собственного языка сценариев и добавить его в систему.

С ActiveX Documents знаком каждый, кто работал с составными документами. С помощью Internet Explorer можно работать, например, с таблицами Microsoft Excel и файлами других офисных приложений. Это делает программу просмотра универсальным средством, способным не только отображать файлы в формате HTML и осуществлять переходы по ссылкам, но и поддерживающим работу с документами любых приложений и даже запуск программ.

5.3. Серверная технология ActiveX (Active Server)

Серверная часть технологии ActiveX реализована с помощью Microsoft Internet Information Server 3.0. С помощью ActiveX можно писать программы на языках сценариев (сейчас это VBScript), выполняющиеся на сервере. Если раньше разработчикам приходилось использовать такие средства, как Microsoft Visual C++ для написания динамически загружаемых библиотек, использующих специальные вызовы Internet Server API, то теперь возможно написание приложений на языке сценариев. Это существенно упрощает разработку, сокращает время написания программы и минимизирует затраты.

Программы, основанные на технологиях Active Server на порядок производительнее программ, основанных на Common Gateway Interface (CGI). Это достигается оптимизацией процессов ActiveX на сервере, учитывающей архитектуру Windows NT.

С помощью языков сценариев на сервере можно осуществлять доступ к системам управления базами данных, поддерживающим стандарт ODBC, и использовать механизм транзакций.

Поскольку подход к использованию технологий ActiveX на сервере стандартизован, программисты могут не только разрабатывать приложения, способные выполняться на серверах, но и реализовывать свои схемы взаимосвязи серверных приложений и сервисов, создавать собственные интерпретаторы серверных языков сценариев. Для этого требуется предварительное приобретение лицензии у Open Group.

5.4. Программные элементы ActiveX

"Оживление" Интернет с помощью технологий ActiveX

Microsoft Интернет Explorer - многоплатформенный универсальный клиент Интернет, поддерживающий технологию ActiveX. ActiveX, в свою очередь, является стандартом, позволяющим программным компонентам взаимодействовать друг с другом по сети независимо от языка программирования, на котором они были написаны. С помощью ActiveX можно "оживить" страницы Web, используя эффекты мультимедиа, интерактивные объекты или сложные приложения, взаимодействующие с пользователем. ActiveX играет роль некоего скрепляющего раствора, с помощью которого отдельные программные компоненты на разных компьютерах склеиваются в единую распределенную систему. Большинство пользователей Web будут иметь дело с программными компонентами, активными документами и макроязыками на основе ActiveX, но вообще ActiveX более многогранна, она включает в себя клиентскую, серверную части и библиотеки для разработчика, а именно: программные компоненты ActiveX - компоненты, работающие на компьютере-клиенте, но загружаемые в первый раз с сервера Web. С их помощью можно показывать разнородную информацию, включающую звук и видео без запуска дополнительных программ. Более того, эти программные компоненты могут использоваться в приложениях, написанных на любых популярных языках программирования, включая Java, Visual Basic, Visual C++. Active Scripting поддерживает любой популярный макроязык, включая Visual Basic(r) Script и JScript. Макроязыки могут использоваться для объединения на одной странице нескольких программных элементов ActiveX или Java, обеспечивая их взаимодействие между собой.

Документы ActiveX позволяют открыть документ любого формата в окне Microsoft Интернет Explorer и способны поддерживать полноценную работу с ним. Можно, например, открыть файл Microsoft Excel или Word с помощью проводника Интернет. Наследует давно знакомую концепцию активных документов OLE (OLE Active Documents). Виртуальная машина Java(tm) позволяет любому проводнику Интернет, поддерживающему технологию ActiveX (например, Интернет Explorer 3.0) выполнять программные компоненты Java и обеспечивать их взаимодействие с программными компонентами ActiveX. ActiveX Server Framework

обеспечивает серверные функции ActiveX, сюда входят поддержка безопасных соединений, доступ к базам данных и другие.

Средства разработки позволят использовать знакомые средства разработки Microsoft или третьих фирм при создании программных компонент на Web-сервере. Эти средства включают в себя Visual Basic, Visual C++, Macromedia Shockwave, Adobe Photoshop, Borland Delphi, средства программирования Sybase и Borland, другие средства, способные создавать программные компоненты Java и т. д.

5.5. Программные компоненты ActiveX

Сегодня существует очень много приложений, которые активно используются в повседневной работе. С помощью технологии ActiveX эти программы могут легко использоваться и в глобальных сетях. При создании технологии ActiveX, специалисты Microsoft переписали OLE таким образом, чтобы эти библиотеки удовлетворяли таким требованиям (важным при работе в Интернет), как высокая скорость выполнения и небольшой размер кода. Также как и OLE, ActiveX базируется на технологии COM (и ее расширении Distributed COM). Таким образом, ActiveX представляет собой ни что иное, как программные библиотеки, полностью совместимые по вызовам (API) с OLE, но оптимизированные для работы в глобальных сетях, поэтому любой ранее написанный программный элемент OLE (OLE Control) будет работать с библиотеками ActiveX. При использовании новой технологии ActiveX пользователю больше не придется вручную загружать отдельные программы для просмотра страниц, содержащих видео, звук, анимационные эффекты... Они могут быть загружены автоматически с самой страницы, причем будет загружен код, подходящий для конкретной платформы клиента, будь то Macintosh, Windows или Unix. Использую популярные языки программирования: Visual C++, Visual Basic или Java, Web-мастера могут легко создавать программные компоненты и помещать их на свои страницы. Это также просто, как работать с компонентами в Visual Basic или Microsoft Access.

Кроме того, используя макроязыки ActiveX, программисты могут обеспечивать взаимосвязь различных компонент. Несколько примеров, как программные компоненты помогут оживить страницы Web: динамическое обновление страниц в зависимости от действий пользователя, внешних событий или состояния самого Web-сервера. Например, программная компонента "Новости" может применяться для показа информации, актуальной до определенного времени или компонента "Часы" (Timer) может использоваться для синхронизации работы других компонент или для привлечения внимания пользователя после продолжительного периода неактивности.

Снижение сетевого трафика с помощью графического отображения данных. Например, программная компонента ActiveX Chart (ее можно

загрузить с www.microsoft.com/ie), используется для построения графика по числовым значениям. Вместо того, чтобы все время для отображения графиков передавать картинки с помощью тега IMG, можно использовать эту программную компоненту, которая загружается только один раз и сохраняется на локальном диске, а затем запускается каждый раз по мере надобности. Вместо этого, каждый раз при рисовании нового графика по сети передаются только числовые данные, а не сам рисунок, что экономит время. Способность отображения текста под любым углом с использованием эффектов анимации. Текст, расположенный по диагонали или вертикально в некоторых случаях может быть более информативным и привлекательным. Чтобы посмотреть примеры применения этих программных компонентов, можно заглянуть в галерею ActiveX (http://www.microsoft.com/activex/gallery).

Простота загрузки программных компонент ActiveX

При просмотре страниц со ссылками на программные элементы ActiveX, эти элементы устанавливаются автоматически без какого-либо вмешательства пользователя. В отличие от компонент plug-in, введенных Netscape, не нужно отдельно запускать программу установки этого программного компонента, а затем перезапускать программу просмотра. Кроме того, применяя программные компоненты ActiveX с цифровой подписью, можно быть уверенным, что данный код не был изменен в процессе передачи по Интернет. Если программный элемент еще не установлен на компьютере, Microsoft Explorer проверит цифровую подпись, которая передается вместе с кодом.

Доступность программных элементов ActiveX

Много программных элементов ActiveX доступны уже сейчас.

Галерея компонент ActiveX

Она находится по адресу http://www.microsoft.com/activex/gallery. Галерея представляет собой набор программных элементов и примеров страниц с их использованием. Разработаны эти компоненты не только Microsoft, но и различными третьими фирмами. Галерея будет расширяться по мере написания дополнительных программных элементов.

а) Программная компонента ActiveMovie

С помощью ActiveMovie можно помещать на страницы видео - и аудиоролики, записанные в популярных форматах: AVI, QuickTime, MPEG, WAV, AU, AIFF, или MIDI. Это означает, что Web-мастера могут помещать любые ролики на свои страницы без всякого опасения, что их не сможет воспроизвести программа просмотра.

Расширяемая архитектура ActiveMovie позволяет добавлять поддержку других форматов, менее популярных сейчас или будущих разработок. Этот элемент можно загрузить с http://www.microsoft.com/ie/download.

б)Видеопроигрыватель для PowerPoint

Новый видеопроигрыватель Microsoft PowerPoint(r) Animation Player for ActiveX - самое легкое средство оживить статическую страницу без изучения сложных языков программирования или покупки сложной аппаратуры и программ создания заставок мультимедиа. Это средство могут использовать все те, кто умеет работать с PowerPoint, все анимационные эффекты, использующиеся в PowerPoint для Windows 95, могут быть перенесены на страницу Web: движущиеся объекты, выплывающий текст, музыкальное оформление и так далее.

в) Программный элемент поддержки VRML

Этот программный элемент реализует поддержку языка моделирования виртуальной реальности (Virtual Reality Modeling Language - VRML), с его помощью можно наблюдать и исследовать 3-хмерные объекты и виртуальные пространства в Интернет. Этот программный элемент поддерживает расширения VRML 1.0 , и позволяет загружать фоновые рисунки, встраивать виртуальные пространства VRML на страницы Web, показывать объемные объекты. После того, как содержимое страницы VRML загрузится, пользователь может выбрать, как ему перемещаться по виртуальному миру, он может идти прямо, поворачивать, перемещаться вбок, наклоняться в любую сторону. Перемещение по 3-хмерному миру может осуществляться с помощью мыши, клавиатуры или джойстика. В следующей версии будет реализована поддержка VRML версии 2.0.

Программный элемент ActiveMovie можно загрузить с http://www.microsoft.com/ie/download/ieadd.htm .

г) Программный элемент HTML Layout Control

Этот элемент позволяет просматривать достаточно сложные страницы, где для указания месторасположения объектов используются координаты, объекты можно накладывать друг на друга и указывать степень их прозрачности. Объекты можно располагать, используя смещение относительно верхнего и левого краев страницы, указывать их размеры и порядок перекрытия (z-order). HTML Layout Control реализует будущее дополнение к стандарту HTML по размещению на страницах Web плоских объектов. Элемент HTML Layout Control может использоваться как основа для работы и размещения на страницы других программных компонент ActiveX. Эту элемент можно загрузить с http://www.microsoft.com/ie/download/ieadd.htm.

д) Программный элемент ActiveX RealAudio

Позволяет прослушивать записи RealAudio, помещенные на странице Web. Программный элемент ActiveX Marquee позволяет просматривать страницу Web, когда она сама прокручивается на экране. Регулируются скорость прокрутки и параметры отображения.

Другие программные компоненты любой Web-мастер или пользователь может легко загрузить из галереи ActiveX по адресу http://www.microsoft.com/ie/appdev/controls/default.htm любой программный элемент и использовать ее для улучшения своего Web-сервера.

е) Label

Этот программный элемент позволяет отобразить текст под углом или расположенный каким-либо другим образом. Эта возможность знакома людям, работающим с издательскими программами, а для HTML она нова, но вместе с тем и привлекательна. При использовании макроязыков можно управлять этим объектом: двигать его, заставлять его расти или уменьшаться, менять цвета и т. п. Preloader. Этот элемент применяется при необходимости загрузить страницу большого объема. Она инициирует загрузку перед тем моментом, когда эта информация действительно понадобится, таким образом, часто удается уменьшить время ожидания. Также этот программный элемент можно использовать для предотвращения перехода на другую страницу перед тем, как полностью загрузится текущая.

ж) Timer

Программный элемент используется для синхронизации страницы, для синхронизации одновременно работающих других компонентов. Например, он может использоваться для периодического обновления страницы, запроса действий пользователя при долгой его неактивности и т. п. ViewTracker. Позволяет обновлять страницу по мере того, как пользователь прокручивает ее содержимое.

з) StockTicker

Обновляет страницу через определенные промежутки времени, позволяя отображать изменяющуюся во времени информацию Popup Menu. Позволяет отобразить выпадающее меню с несколькими вариантами выбора. Chart. Позволяет графически отображать табличную информацию в различных вариациях и разными графиками. Более подробная информация о программных компонентах ActiveX Controls, содержится на странице http://www.microsoft.com/workshop/author/cpad/ .

5.6. *Макроязыки ActiveX*

Поскольку Microsoft Интернет Explorer 3.0 поддерживает Visual Basic(r) Script и JScript, эта программа просмотра предлагает уникальные возможности по управлению содержимым Web-сервера с помощью макроязыков. Используя макроязыки, можно создавать страницы, на которых пользователю будут задаваться вопросы, проверяться введенные пользователем данные, т.е. страницы, активно взаимодействующие с пользователем. С помощью макроязыков можно создавать страницы, на которых будет осуществляться взаимодействие программных компонентов ActiveX, Java, других программ на клиентском компьютере, различных

частей самого Интернет Explorer, например, взаимодействие документа HTML с панелью управления или меню. Кроме того, можно написать свой макроязык и добавить его интерпретатор в Интернет Explorer с помощью динамически загружаемой библиотеки DLL.

а) Visual Basic Scripting Edition

VB Script является подмножеством языка Visual Basic и предназначен для программирования страниц Web. С его помощью можно заставить взаимодействовать разные объекты на странице, в том числе программные компоненты. Этот язык полностью совместим с Visual Basic и Visual Basic for Applications. Microsoft откроет VBScript для бесплатного лицензирования третьим компаниям, производящим программы просмотра ресурсов Интернет. С помощью VBScript программисты могут отслеживать события, получаемые от программных компонентов ActiveX, активизировать методы и изменять свойства компонентов ActiveX. Код, написанный на VBScript, легко изменять, поскольку он находится на самой странице. Выполняться он может на всех платформах, где есть или будут интерпретаторы этого языка.

б) Java Scripting

JScript обеспечивает такую же функциональность, как и VBScript. Реализация Microsoft этого языка позволяет связывать и синхронизировать объекты на странице, включая программные компоненты ActiveX и Java, создавать страницы, способные реагировать на действия пользователей.

5.7. Документы ActiveX

ActiveX Documents позволяет открыть окно другого приложения внутри Интернет Explorer. Это означает, что с помощью Интернет Explorer можно открывать и работать, например, с таблицами Microsoft Excel, после чего можно вернуться на Web-страницу, с которой была ссылка на эту таблицу. Как и все спецификации ActiveX, ActiveX Documents являются открытой спецификацией, доступной для всех. Поддержка проводником Интернет спецификаций ActiveX Documents позволяет ей стать программой-оболочкой, способной показывать не только страницы Web, но и любые документы. Это очень ценное свойство для применения в интрасетях, поскольку все сотрудники организации могут использовать одну программу для работы с разнородными документами, тогда как сетевые администраторы могут связывать обычные офисные документы с помощью страниц HTML, сами же эти документы будут оставаться в форматах тех приложений, которые использовались для их создания. Интернет Explorer может служить не только не только клиентом ActiveX Document, но быть и сервером ActiveX Document. Это означает, что любое приложение может использовать Интернет Explorer для просмотра файлов формата HTML. Спецификация ActiveX Documents - дополнение к спецификации OLE Documents, технологии составного документа OLE.

Также, как и активные документы OLE, приложения, поддерживающие ActiveX Documents, могут работать как контейнеры (клиенты), обеспечивающие показ документов ActiveX, или как серверы, так, что любая другая программа может их использовать для своей работы.

5.8. Средства разработки компонентов ActiveX

Программисты могут создавать компоненты ActiveX, используя любой язык программирования, включая давно знакомые Visual Basic, Visual C++ или новые средства программирования, такие как Java. С помощью Visual Basic 5.0 программисты смогут писать компоненты ActiveX и конвертировать существующий код в VBScript простой манипуляцией мышью. Все макросы, которые использовались в приложениях Microsoft Office, могут быть легко перенесены на страницы Web. Если программист использует новый язык Visual J++, он увидит снова знакомый интерфейс Microsoft Developer's Studio-привычный отладчик и редактор - и сможет создавать машинно-независимые программные компоненты Java, способные выполняться в Интернет Explorer. Если пользователь хочет быстро с нуля разрабатывать компоненты ActiveX, он может использовать Microsoft ActiveX Development Kit (MADK). Он включает в себя все средства и информацию, необходимые для этого. Web-мастера могут управлять своими серверами с помощью Microsoft FrontPage. Этот продукт распространяется с большим количеством мастеров (wizards), позволяющих быстро создавать страницы с таблицами, фреймами и другими интересными элементами оформления, включая программные элементы ActiveX.

6. Введение в Web-дизайн

6.1. Дизайн или содержание?

Этот вопрос сам собой возникает у большинства Web-мастеров. Мнения делятся почти поравну, но большая половина достается содержанию. Действительно, ведь ради содержания - информации и существует Интернет- дизайн производная, но, безусловно, важная часть любого сайта. Web-дизайн порождает новые технологии, по сути, Web-дизайн имеет не только свои принципы, стандарты, но и свой рынок. Этот рынок растет, развивается, но все-таки стоит помнить, что дизайн без хорошего содержания не представляет интереса для большинства людей, пользующихся глобальной сетью.

Очень важный вопрос - с чего начать создание сайта? Одно из больших заблуждений, при создании сайта, заключается в том, что начинать нужно с дизайна. Но это совсем не так, и едва ли большую часть времени, ушедшего на создание сайта, будут занимать дизайн и верстка.

6.2. Содержание

Приступать к работе следует с тщательного продумывания концепции и составления структуры будущего сайта. И уже на основе этой информации подбирать необходимые материалы. Следующим шагом будет представление всей информации удобным и понятным образом, чтобы вашу информацию узнал бы и ваш посетитель. При отборе информации, жизненно важно ее структурировать, уметь выделить наиболее значительное. Нередко можно слышать возражение, что на сайте важна вся информация, и необходимо акцентировать ее всю. Но, на самом деле, так не бывает, и даже в самой важной информации есть порция наиболее важной. Хорошая структурированность сайта обеспечивает половину успеха при его создании. Ошибки, допущенные на этом этапе, впоследствии могут привести к большим затратам сил на исправление.

Ключевым моментом содержания является тема. Выбор темы в основном зависит от ваших интересов. Создавая свою домашнюю страницу, вы как бы ищете людей, имеющих такие же увлечения или работу. Не стоит распыляться. Сделав сайт, перегруженный всевозможными темами, Вы вводите посетителей в замешательство, сбиваете с толку не только их, но и себя. Даже, если вас интересуют очень многие вопросы, выберете лишь несколько самых важных из них. Этим вы четче обрисуете тему проекта. Именно к конкретизации темы и надо стремиться, создавая новый сайт. Например, типовая структура сайта небольшой компании может быть следующей.

- Новости компании.
- Информация о фирме.
- Услуги фирмы.
- Каталог товаров.
- Контактная информация.

Теперь эту структуру необходимо скорректировать для конкретной фирмы. Возможно, что у компании есть несколько одинаковых по объему областей деятельности и в этом случае, конечно же, имеет смысл добавить еще несколько пунктов меню в верхний уровень. При составлении структуры будущего сайта, не помещайте все пункты меню на одну страничку. Человек комфортно воспринимает не более 7-ми пунктов, и их большее количество вызывает подсознательное отторжение при просмотре. Слишком сложное, не вызывает желания разбираться.

Для того, чтобы этого избежать, и используется иерархическое построение меню. Правильное структурирование информации позволяет выделить основные разделы сайта, а все второстепенные разделы будут доступны после выбора одного из основных. Описанное замечание относится также и к меню 2-го уровня - при большом количестве пунктов имеет смысл ввести меню 3-го уровня. Не рекомендуется создавать меню с четвертым и более уровнем.

Если же информации все же настолько много, что не получается структурировать ее в три уровня меню, то имеет смысл выделить отдельные подсайты, каждый из которых будет отвечать за свою область. Так, например, часто выносят электронный магазин из структуры основного сайта, и он приобретает достаточно самостоятельный характер.

6.3. Подготовка текстов

Кроме разработки структуры, важной частью данного этапа является подготовка текстов. Оригинальные тексты, например, взятые из буклета компании и выложенные в неизменном виде будут читаться плохо, ведь они разрабатывались совершенно для другого стиля восприятия информации. И на сайте такой текст будет смотреться отчужденно - это заметно. Сайт может быть красив, но лишь умные тексты позволят удержать своего посетителя – об этом надо помнить.

Публикация текстов в Сети имеет свою специфику и буквальное повторение структуры буклета "О компании" будет ошибкой. Буклет просматривают линейно - сверху вниз и слева направо - по страничкам. Соответственно и пишется текст, который последовательно излагает всю информацию. Для web-страничек же более логичным является способ структурированного представления информации. При этом варианте - в основной статье излагаются общие вопросы, а более подробное их освещение вынесено на отдельные странички, переход к которым осуществляется по ссылке. Ссылка может быть размещена как в конце странички. Таким образом она будет служить указателем на следующую страничку (это аналогично линейному просмотру информации). Ссылка может находиться и непосредственно внутри текста, указывая посетителю способ узнать о данном вопросе подробнее.

В чем преимущество именно такого способа представления информации? А в том, что посетитель читает лишь то, что ему нужно. Некоторым достаточно краткого обзора, в то время как другие предпочитают углубляться в суть вещей. Структурируя большую статью, мы разбиваем ее на ряд небольших фрагментов, каждый из которых содержит один из аспектов рассматриваемого вопроса. И читатель волен выбирать путь, по которому идти - отбирая лишь ту информацию, которая ему действительно интересна. А второе преимущество является следствием первого - небольшие странички очень быстро грузятся, что немаловажно.

Все сказанное выше, относится исключительно к чтению информации он-лайн. Если же предполагается, что текст будет распечатываться, то стоит подумать о создании специальных "страничек для печати", которые должны содержать полные тексты статей.

Стиль изложения информации является не менее важной частью, чем стиль дизайна и столь же уникален для хорошего сайта. На корпоративных сайтах, как правило, стиль изложения достаточно официален, но даже в

этом случае тексты желательно подготовить, а не выкладывать всем надоевшую, информацию из пресс-релиза.

Структура готова, тексты написаны - вот теперь, пожалуй, и можно приступать к созданию набросков дизайна

6.4. Дизайн

На этой стадии очень важно выбрать инструмент, с которым будет идти работа. Первоначальный вариант можно набросать и карандашом на листке бумаге, но его придется переносить в электронный вид. Многие web-дизайнеры предпочитают использовать для целей макетирования растровые или векторные редакторы, которые позволяют свободно манипулировать объектами на будущей страничке. Результатом этого этапа должна быть готовый эскиз.

Нужно понять, какой круг людей может заинтересоваться данным сайтом. Какими браузерами пользуются эти люди? Версии браузеров? Хотя бы приблизительно должны быть ясны ответы на эти вопросы. Исходя из этого, надо пытаться сделать что-то максимально гибкое и совместимое между Netscape и Explorer. Версии влияют на поддержку Java, JavaScript, CSS, DHTML и т.п. В крайнем случае, на страничку помещаются требования к браузеру. Необходимо помнить про разрешение экрана. Самый распространенный монитор на данный момент имеет диагональ 15 дюймов, следовательно, самыми популярными режимами являются 800x600 и 1024x768. После этих слов появляется много причин для того, чтобы протестировать свою страничку на разных машинах.

Отметим, что до сих пор речь не шла о HTML-кодировании, потому что на самом деле это далеко не первая по важности вещь и все вышеперечисленные шаги могут занять большую часть, отведенную на создание сайта, времени. Ведь, в конце концов, написание HTML-кода есть дело техники. Есть некоторые нестыковки со стандартами, но все это преодолимо, в то время, как вышеописанные шаги есть процесс творческий.

6.5. Понятие стиля сайта

Стильный web-сайт - это когда, каждая страничка публикации имеет ярко выраженную принадлежность ко всему web-сайту, когда легко ориентироваться и поиск информации не сопряжен с опасностью заблудиться и потерять время, когда странички загружаются менее, чем за минуту.

Стильность web-сайта достигается несколькими приемами. Можно выделить следующие элементы, участвующие в создании стиля:

- шрифт - в пределах публикации он должен иметь одинаковые характеристики - такие, как гарнитура (начертание), кегль (высота), цвет.

- абзац - желательно, чтобы преобладал какой-нибудь один из видов выравнивания на страничке, например, публикация сделана с отступом от левого края и выравниванием влево.
- цветовая схема web-сайта - она начинается с выбора тех трех цветов страницы, которые используются для представления обычного текста, ссылок и посещенных ссылок. Все эти параметры указываются в теге <body>. Ниже, для примера, приведена строчка, задающая цвета на этих страничках:

<body bgcolor="#669900" text="#333333" link="#669900" vlink="#666666" alink="#000000">

Цветовая схема должна повторяться на всех страничках публикации, это создает у посетителя ощущение связности сайта. Цвета ссылок надо стараться выбирать таким образом, чтобы, с одной стороны, посетитель видел, что это ссылка, а с другой стороны, она бы не мешала ему читать основной текст. По поводу ссылок есть два полезных замечания: первое - как бы не хотелось сделать цвета ссылок и посещенных ссылок одинаковыми (без веских на то причин), лучше придать им немного различия, для этого цвета уже посещенных ссылок, делаются чуть темнее; и второе - так уж получилось, что подчеркнутый текст в Web символизирует ссылку, поэтому не стоит использовать подчеркнутый текст в публикации, предпочтительнее воспользуйтесь другим способом выделения.

Графическое оформление сайта - во-первых, оно должно укладываться в общую цветовую схему; во-вторых, должна быть продумана общая концепция графического оформления. Все графические элементы можно разделить на два больших класса: рисованные и фотореалистические. Не стоит смешивать эти два типа в оформлении. Кроме этого, в случае, если используются на сайте фотографии в качестве иллюстраций, то перед использованием они должны быть обработаны - это тоновая и цветовая коррекция, кадрирование, выбор примерного размера фотографии в публикации, обработка края фотографии. Затем данное оформление используется по всей публикации. К фотографиям очень желательно писать пояснения в параметре ALT тэга IMG - это будет восприниматься как подпись к фотографии и, кроме того, избавит пользователя от ожидания в случае, если его не интересует данная страничка.

Навигация по сайту - именно она не дает посетителю заблудиться в дебрях сайта. Должна всегда быть возможность для посетителя перехода на главную страничку публикации. Кроме этого, очень много людей попадают на странички через поисковые системы, т.е. не на первую страницу, и хороший сайт должен позволить читателю перейти на первую страничку. То есть, навигационная система должна быть продублирана на всех страничках. В случае, если навигационная панель выполняете

графическими средствами, то обязательно надо сделать ее текстовую копию и поместить где-нибудь снизу (текст в любом случае загружается быстрее графики).

6.6. Программы, используемые в веб-мастеринге

Чтобы разработать профессиональный сайт, необходимо владеть несколькими профессиональными программами. Рассмотрим некоторые из них.

6.6.1. Программы обработки растровой графики

Эти программы нужны при сканировании фотографий и их коррекции. При их помощи можно создавать те разнообразные эффекты, которые обязательно присутствуют в Интернете, рекламах и т.д. К наиболее полезным можно отнести следующие навыки для этого класса программ:

- Сканирование фотографий.
- Коррекция отсканированных и готовых фотографий, в том числе - цветовая коррекция.
- Ретушь фотографий.
- Умение кадрировать.
- Понимание различий в форматах графических файлов.
- Грамотное использование фильтров.
- Работа со слоями.

В качестве конкретных программ можно привести следующие:

Adobe PhotoShop 6.0 - данная программа является лидером в области графических программ такого рода, но она требует и соответствующих ресурсов от компьютера.

Paint Shop Pro 4.0-5.0 – одна из лучших shareware-программа, которая, к тому же, поддерживает фильтры от Adobe PhotoShop и очень быстро работает с объемными (>20M) фотографиями. Может импортировать и экспортировать изображения в 40-50 разных форматов.

Для некоторых работы придется использовать и другие редакторы. Идеальных редакторов нет, некоторые лучше делают одно, некоторые - другое.

6.6.2. Программы обработки векторной графики

Еще одна группа из важных для дизайнера программ. Она позволяет создавать с нуля или с использованием клипарта различные логотипы, кнопки, эффектные надписи и т.п. вещи. Принципы векторных редакторов сильно отличаются от растровых, поэтому осваивать их придется отдельно; зато, освоив их, можно изготавливать для себя фирменные визитки, бланки, брошюры (как побочный эффект). Для работы с векторной графикой потребуются следующие навыки умения работать:

- с графическими объектами (группировка, наложение, получение нестандартных объектов)

- с кривыми и узлами кривых;
- с направляющими и сеткой;
- с цветовыми моделями (RGB, CMYK, HSB);
- с текстом (разместить его на любой кривой);
- с градиентми и заливками;
- с эффектами.

Типичными представителями данного класса являются *Corel DRAW 7.0-10.0* и *Adobe Illustrator* - они обе являются лидерами в своих областях и, соответственно, располагают самыми последними достижениями в области векторной графики. В противовес этим тяжеловесам можно порекомендовать симпатичную и быструю программку Corel Xara!; пусть Вас не вводит в заблуждение слово Corel - данная программа разработана фирмой XARA, которая была куплена корпорацией, как один из конкурентов. Сейчас доступна уже ее вторая версия. Данная программа, в отличие от двух предыдущих, очень быстрая и маленькая, но некоторые эффекты и команды, доступные в других программах, в ней сделать или трудно, или вообще невозможно.

6.6.3. Программы просмотра web-страничек

Это - программы-браузеры. В настоящее время наиболее используемыми являюся три браузера - "Microsoft Internet Explorer", "Netscape Navigator", «Opera». Сейчас приближаются уже пятые версии данных программ. Поэтому для контроля внешнего вида страничек используются они. Причем на компьютере должны стоять обе версии. Хорошая WEB-страничка должна одинаково выглядеть в любом из этих броузеров.

6.6.4. Простой текстовый редактор

Он понадобится для ручного исправления и добавления HTML-кода, т.к. существующие визуальные редакторы не могут полностью контролировать процесс создания web-странички (а если делается страничку со сложным дизайном, то, возможно, придется весь код писать в редакторе). В качестве примера подойдет обычный "*Блокнот*" из стандартной поставки Windows или один HTML-редакторов, которые имеют встроенные команды на проверку правильности тэгов и структуры документов, например, *HoTMetaL*.

6.6.5. Визуальные редакторы

Они позволяют быстро разрабатывать несложные web-странички и корректировать уже написанные, но с ними нужно быть осторожными, т.к., благодаря именно им, Ваша страничка может плохо глядеться в другом браузере.

Наиболее известные кандидаты - это "*Microsoft Front Page*" и "*Netscape Composer*" (или в ранней версии - редактор, встроенный в поставку "Netscape Navigator Gold"), HomeSide, DreamWeaver 3. Иногда

придется вручную исправлять код, сгенерированный данными программами.

6.6.6. Текстовый процессор

Его возможности проверки орфографии - очень полезен для набора текста и исправления ошибок в распознанных текстах. Как пример - обычный *"Microsoft Word"*.

6.6.7. Программы распознавания текста

Они помогут сэкономить массу времени, избавляя от ручного набора напечатанных текстов. Программ по распознаванию русского языка всего две - это *CuneiForm и FineReader*. Лучше использоватье последние версии. По качеству распознавания они примерно одинаковы.

6.6.8. Некоторые специальные программы

Эти программы позволяют выполнять некоторые эффекты и справиться с такими задачами, которые другими способами трудновыполнимы или не выполнимы вообще. Ниже перечислены некоторые из этих программ:

Ulead GIF Animator - программа, позволяющая создавать анимированные GIFы. Обеспечивает полный контроль над выходным файлом. Обладает очень мощными средствами оптимизации.

Фильтры для *Adobe PhotoShop* - их количество просто огромно, но реально понадобится немного. Они способны существенно повысить работоспособность и двумя-тремя нажатиями создать впечатляющие эффекты.

Macromedia Flash - практически стандарт для использования в web векторных изображений. Обладает собственной средой разработки и позволяет создавать впечатляющую векторную анимацию.

Программы для создания VRML-миров или 3D-программы, позволяющие экспортировать в данный формат. В качестве примера могу порекомендовать неплохую программу создания VRML-миров *Internet3D*.

Space Builder. Программы для обработки звука - могут понадобиться, если того требует страничка. Это программы вообще отдельного класса, но для простой обработки звука, например, подойдет *CoolEdit*.

6.7. Оптимальный размер страницы

Размер странички складывается из размеров текстовой и графической информации. Можно просто сказать, что размер всего сайта это размер дизайна плюс размер содержания. Web-мастера должны волновать размеры всех отдельных элементов дизайна и размеры кодов страниц. Сложив размеры всех подключаемых к странице файлов: картинок, апплетов, таблиц стилей, включенных CGI-скриптов, и размер самого html-файла получим размер всей страницы. Каким, в идеале, должно быть это количество? Размер должен орпделяться килобайтами.

Страница размером более одного мегабайта большинству людей будет не интересна, какой бы дизайн и содержание она в себе не несла. Страницы менее 1K страдают противоположным явлением. За оптимальный можно принять размер от 15K до 100K. Страницы, перегруженные всевозможными кнопками и баннерами, как правило, не догружаются пользователями. В этом плане, за пример сайтов с оптимальным размером можно рассматривать все самые популярные сайты. Таких как Yahoo! или Яndex.

6.8. Обзор сайтов

В качестве примера можно привести один из популярнейших серверов Yahoo! sites. Статистика посещений этого узла полностью оправдывает отсутствие всевозможных скриптов и фреймов на главной страничке. На сайт каждый месяц заходят 37 миллионов посетителей! По этой причине разработчики сайта уделяют много внимания совместимости и размеру страниц. Сайт Microsoft ежемесячно посещают всего 22 миллиона человек и видимо поэтому дизайн располагает множеством табличных тегов и скриптов. Недавно появились даже Flash-баннеры, которые пока размещены в нижней части страниц. Из сайтов Рунета можно назвать победителями в отношении содержание и дизайна поисковые системы: Яndex, Апорт.. Сервер фирмы Macromedia содержит много полезной информации, касающейся технологий, применяемых в современном дизайне.

7. Первые шаги в создании Web-страницы на HTML

С появлением Интернет проблема межплатформной несовместимости файловых форматов стала ощущаться особенно остро: ведь тем, кто размещает информацию в Сети, хочется, чтобы она была доступна как можно более широкому кругу пользователей без особенных затрат. Очевидно, что самый простой путь решения — разработка общепринятого формата файлов, под который и будут «подстраиваться» вновь создаваемые приложения. В WWW таким стандартом стал HTML.

Файлы HTML состоят из команд форматирования, текста и ссылок на другие файлы или объекты (графика, звуки, программы). Программа для просмотра HTML-документов (браузер) интерпретирует код HTML, содержащийся в файле, и согласно командам форматирования собирает готовую Web-страничку.

Сам процесс создания домашней Web - страницы похож на программирование: создается «программа», описывающая размещение текста и иллюстраций на странице и связи между страницами. Программа-браузер интерпретирует эту «программу» и выдает в окне результат — отформатированный заданным образом текст вместе с иллюстрациями.

Таким образом, выполнение этой «программы» происходит автоматически. Если подготовлены исходные файлы, то страница готова к

немедленному использованию. Все, что необходимо, — это научиться пользоваться тегами языка HTML, чтобы с их помощью так оформить Web - страницы, чтобы ею было удобно и приятно пользоваться.

Минимальный набор для работы – это текстовый редактор (например, WordPad, ME8) и браузер.

Из чего состоят файлы HTML? Из обычного текста: исходный код файла можно просмотреть в любом текстовом редакторе, включая Блокнот Windows (Notepad). Но в этом случае вы увидите не web-страничку, а лишь текст в сочетании с метками HTML, из которых программа-браузер ее генерирует. Редактировать web-документы вы также можете, используя самый обычный Блокнот. Именно так и поступают многие опытные web-мастера, несмотря на то, что сейчас существует большое количество визуальных редакторов web-страниц. Эти редакторы зачастую обладают очень богатыми средствами и предоставляют пользователю возможность создавать великолепно выглядящие страницы, используя многочисленные шаблоны и библиотеки картинок, как правило, входящие в комплект поставки редактора. Но такие страницы не отличаются оригинальностью, так как подготовлены с помощью шаблонов. Кроме того, HTML-код, который генерируют визуальные редакторы, зачастую оставляет желать лучшего: избыточность кода и ошибки в форматировании не улучшают скорость загрузки и внешний вид документа.

Самыми популярными визуальными редакторами являются Microsoft FrontPage, Netscape Composer, NetObjects Fusion, Dreamweaver, HotDog Professional 5.5 (www.sausage.com)

7.1. Основные элементы языка HTML

На компьютерах, работающих в операционной системе Windows, такие документы хранятся в файлах с расширением .htm. На компьютерах, работающих в других операционных системах, расширение имени может быть из четырех букв — .html.

Таким образом, необходимо сохранять файлы в текстовом формате с расширение htm или html.

7.2. Как записываются теги HTML

Все теги HTML записываются в специальных угловых скобках, которые образуются символами “<” и “>”. Первое слово, которое идет после открывающего символа “<”, — это и есть собственно тег, а дальше до закрывающего символа “>” могут указываться параметры данного тега.

Некоторые теги являются парными, то есть влияют на все, что заключено между открывающим и закрывающим тегами. Закрывающий тег обычно выглядит так же, как открывающий, но содержит перед названием дополнительный символ “/” (косая черта), например <BODY> и </BODY>.

7.3. Структура документа HTML

Все документы HTML имеют одну и ту же структуру, которая выглядит следующим образом:

```
<HTML> <TITLE>Заголовок документа</TITLE>
<BODY>
Тело документа
</BODY>
</HTML>
```

Тег <HTML> является признаком того, что данный файл содержит документ HTML. С него начинаются все такие документы, и, обнаружив этот тег, браузер знает, что далее ему необходимо будет обрабатывать другие теги HTML.

Тег <TITLE> служит для задания заголовка документа. Текст, помещенный между тегами <TITLE> и </TITLE>, выводится в строке заголовка программы-браузера.

Тег <TITLE> служит для задания заголовка документа. Текст, помещенный между тегами <TITLE> и </TITLE>, выводится в строке заголовка программы-браузера.

Тег <TITLE> служит для задания заголовка документа. Текст, помещенный между тегами <TITLE> и </TITLE>, выводится в строке заголовка программы-браузера.

Тег <BODY> задает основную часть документа — его "тело". Информация, размещенная между тегами <BODY> и </BODY>, выводится в окне браузера и является его содержанием.

7.4. Как будет выглядеть текст, размещенный в теле документа

Отличительной особенностью языка HTML является то, что он не накладывает жестких ограничений на внешний вид текста в окне браузера. Вместо этого обычно задаются только общие принципы размещения информации. В частности, если не задать больше никаких тегов, кроме определяющих весь документ, то текст будет аккуратно отформатирован с учетом размеров окна.

Каждый абзац будет выравниваться по левому краю и иметь ширину, соответствующую ширине окна. При изменении размеров окна текст будет автоматически переформатироваться, с тем, чтобы оставаться удобочитаемым.

Однако, несмотря на автоматизированность и универсальность этой системы, у нее есть некоторые недостатки. Дело в том, что любой текст, все-таки выглядит лучше при определенном размере окна. Поэтому рекомендуется выбрать для себя определенный размер окна и задать внутри текста теги, "привязывающие" текст к указанной ширине.

Если страница всегда будет демонстрироваться на одной и той же машине, то можно рассчитывать на строго определенный размер окна браузера, что практически не встречается.

7.5. Начальный этап в создании страницы

Таблица 2.

Примеры команд форматирования текста в HTML

Команда	Результат
<I>Этот текст будет отображаться как наклонный.</I>	Этот текст будет отображаться как наклонный.
<I>Этот текст будет отображаться как наклонный. <U>А этот как наклонный и подчеркнутый.</U></I>	Этот текст будет отображаться как наклонный. А этот, как наклонный и подчеркнутый.
На странице этот текст будет располагаться в две строки.	На странице этот текст будет располагаться в две строки.

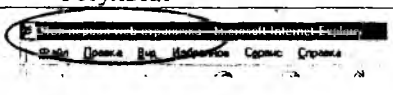
В WWW существует правило, согласно которому первый файл, загружаемый браузером с какого-либо сервера по умолчанию, должен называться index.html. Поэтому, если явно не указывается имя документа, который требуется, а пишется только имя сервера, например www.yahoo.com, автоматически будет загружен файл index.html, находящийся на этом сервере, т. е. фактически будет выполнена команда http://www.yahoo.com/index.html. Итак, первый файл который создается, а потом размещается в Интернет, будет называться index.html. Для этого потребуются лишь несколько команд форматирования и оформления текста в формате HTML.

7.6. Заголовок HTML-документа

HTML-файл состоит из двух основных частей: «головы» (заголовка) и «тела». «Тело» HTML-документа заключает в себе все информативное содержание, тогда как в заголовке указывается тип документа, его кодировка, язык, имя автора и прочая дополнительная информация. Тело документа отделяется от заголовка с помощью меток (тегов) HTML.

Таблица 3.

Пример команды заголовка страницы в HTML

Команда	Результат
<TITLE>Моя первая web-страничка </TITLE>	

Рассмотрим другие теги, входящие в заголовок страницы.

<META> - Этот тег используется для указания подробной информации о документе. Употребляется в сочетании с описаниями атрибутов документа и не требует закрытия.

Атрибутами дескриптора <META> может быть любая информация, которую вы хотите сообщить. Существует несколько стандартных атрибутов, необходимых для указания браузеру способа кодировки документа, имени создателя и информации для поисковых систем.

<META HTTP-EQUIV="Content-Type"CONTENT="text/html; CHARSET=Windows-1251">

Атрибут CONTENT (содержание) указывает броузеру, что содержанием документа является HTML в кодировке (CHARSET) CP-1251(Windows). Информация о том, что документ размечен в формате HTML, фактически дублирует метку <HTML>, открывающую документ. А вот данные о кодировке документа нужны броузеру, чтобы сгенерировать конечный файл с тем набором символов, который принят в операционной системе пользователя. Если кодировка загружаемого документа отличается от системной, то браузер конвертирует текст документа, иначе — оставляет нетронутым.

Общая структура тега <META> такова:

<META NAME </TITLE>

<META HTTP-EQUIV="Content-Type"CONTENT="text/html; charset=windows-1251">

<META NAME="Author"CONTENT="Dadabaeva,Hodiev,Ruzmetova">

<META NAME="Keywords"CONTENT="HTML, курс HTML, информационные технологии, Flash5, создание web-страниц">

</HEAD>

</="Имя определяемого параметра" CONTENT="значение параметра">

Атрибут NAME содержит имя определяемого параметра, а CONTENT — значение параметра, определенного атрибутом NAME. Например, если нужно дать сведения об авторе документа, это можно сделать следующим образом:

<META NAME="Author" CONTENT="имя создателя странички">

Можно разместить ключевые слова в теге <META>:

<META NAME="Keywords" CONTENT="ключевые слова">

Ключевые слова необходимы для индексации страницы в поисковых системах. Чем больше слов, соответствующих содержанию страницы будут помещены в этой строке, тем больше вероятность того, что страницу посетят.

В заголовок WWW-шаблона поместим следующую информацию:

HTML>

<HEAD>
<TITLE>Первая web-страничка HTML>

Если набрать это в текстовом редакторе и сохранить как текстовый документ с именем index.htm (либо другим, но с расширением .htm), открыв этот файл в браузере, то увидим озаглавленную, но абсолютно чистую страницу.

Следующий этап - сделать «тело» страницы.

В открывающий тег <BODY> могут входить элементы, определяющие параметры страницы, такие как используемый фоновый рисунок, цвет и размер основного шрифта и гиперссылок, цвет фона и т. п. Цвета этих элементов web-страницы указываются в шестнадцатеричном исчислении.

7.7. Цвет в документах HTML

Пусть цвет фона и шрифта будут соответственно белым и черным. Строка с указанием этих параметров страницы будет содержать следующие элементы:

<BODY BGCOLOR="#FFFFFF" TEXT="#000000">

При таком варианте описания раздела <BODY> явно указываются только цвета фона и основного текста. Цвет гиперссылки и посещенной гиперссылки устанавливаются автоматически, в соответствии с настройками браузера. Для гиперссылок установим нестандартные цвета, например зеленый для непосещенной и голубой для посещенной.

<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#008000" VLINK="#0080FF">

7.8. Фоновый рисунок в HTML

Для того чтобы оформить страничку фоновым рисунком, в теге <BODY> нужно указать параметр BACKGROUND="имя файла". Естественно, графический файл с фоновым рисунком также должен находиться на вашем Интернет-сервере. В качестве фонового рисунка вы можете использовать только файлы формата GIF или JPEG. Фоновый рисунок на нашей тестовой страничке задается файлом GRNFOL2.gif, а цвета текста и гиперссылок не определены, т. е. устанавливаются согласно настройкам браузера.

Таким образом, тег <BODY> на нашей страничке выглядит так:

<BODY BACKGROUND="GRNFOL2.gif">

При выполнении фон странички будет выглядеть, как показано на рис.16.

Рис. 16. Фоновый рисунок на странице HTML

Важно, что регистр символов в имени файла имеет значение. После создания тега <BODY> с указанием цветов мы можем приступить к наполнению странички информационным содержимым.

7.9. Шрифты в HTML-документах

Для оформления и форматирования текста в HTML-документах применяется небольшое количество тегов. Если текст не оформлен никакими тегами, то отображаться он будет в соответствии с настройками браузера пользователя.

Для управления шрифтовым оформлением WEB-страниц предназначен тег <FONT>. Этим тегом указываются параметры шрифта, такие, как размер, цвет и название. Параметры шрифта в теге <FONT> указываются в кавычках в любом порядке.

Определяя шрифт документа, нужно иметь ввиду, что при просмотре странички будут использоваться те шрифты, которые установлены на компьютере пользователя. И если будет использованы на страничках необычные шрифты, другие пользователи их могут не увидеть.

В настоящее время стандартными шрифтами, использующимися в WWW, являются Times New Roman и Courier New. Также можно пользоваться и шрифтом Arial, так как он распространен не менее широко, чем указанные. Пользоваться другими шрифтами нет смысла, поскольку они недостаточно популярны.

Строка, указывающая, что данный фрагмент текста необходимо будет отображать с установленным по умолчанию шрифтом размером 6 единиц и буквами фиолетового цвета, будет выглядеть так:

```
<FONT SIZE="6" COLOR="#8080C0">Текст.</FONT>
```

Если используется для текста нестандартный шрифт, в теге <FONT> необходимо указать параметр FACE, т. е. для отображения той же строки шрифтом Arial тег будет выглядеть так:

```
<FONT FACE="Arial" SIZE="6" COLOR="#8080C0">Текст.</FONT>
```

В параметре FACE можно указывать несколько шрифтов в порядке предпочтения. Текст будет отображаться тем шрифтом, который установлен на компьютере пользователя, в порядке предпочтения, указанном в теге <FONT>.

<FONT FACE="Verdana, Tahoma, Arial">Текст.<FONT>

Если на компьютере пользователя, просматривающего вашу страничку, не найден ни один из указанных вами шрифтов, текст будет отображаться шрифтом, указанным в настройках браузера пользователя, и, скорее всего, им станет Times New Roman.

Для управления начертанием символов, т. е. для установки параметров Жирный (Bold), Курсив (Italic) и Подчеркнутый (Underlining), используются теги <B>, <I>, <U>. Эти теги требуют закрытия и могут использоваться, перекрывая действие друг друга.

7.10. Команды форматирования абзацев

Как известно, на странице текст размещается в абзацах. Для определения абзаца в HTML используется тег <P> </P>. Он имеет параметр ALIGN, управляющий выравниванием текста параграфа с помощью четырех аргументов: LEFT, RIGHT, CENTER и JUSTIFY:

<P ALIGN=LEFT> — выравнивание текста по левому краю экрана;

<P ALIGN=RIGHT> — выравнивание текста по правому краю экрана;

<P ALIGN=CENTER> — выравнивание текста по центру экрана;

<P ALIGN=JUSTIFY> — полное выравнивание текста по обоим краям экрана.

Для размещения любого объекта по центру строки существует тег <CENTER>. Он вполне применим также и к тексту. Если необходимо расположить по центру строки лишь несколько слов или заголовок, тег <CENTER> будет являться более удобным вариантом, чем тег <P ALIGN=CENTER>

<P> и <CENTER> требуют закрывающего тега. В случае, если весь текст, находящийся на странице после открытия одного из этих тегов, должен быть размещен с тем же выравниванием, вы можете их не закрывать. Для удобства форматирования текста на странице существует также дескриптор
, переносящий следующий за ним текст на следующую строку.

7.11. Дескриптор <HR>

Это очень простой тег. В браузере он отображается как горизонтальная линия. Для управления внешним видом отображаемых

линий у тега <HR> имеются четыре параметра: SIZE, WIDTH, ALIGN и NOSHADE.

Таблица 4.

Параметры дескриптора <HR>

SIZE=N(пикселов)	Устанавливает толщину линии
WIDTH=N(пикселов или процентов)	Устанавливает ширину линии. Ширина линии может быть жестко указана в пикселах или в процентах относительно ширины окна браузера;
ALIGN=LEFT (RIGHT или CENTER)	Устанавливает выравнивание отображаемой линии относительно окна браузера. При отсутствии параметра ALIGN линия всегда центруется относительно окна браузера;
NOSHADE	Отключение трехмерного режима отображения линий. При указании этого параметра линия будет отображаться как простая черная полоса.

Теперь можно создать свой шаблон, запомнить с именем index.htm,открыть созданный шаблон и внесите в заголовок информацию о кодировке, авторе, заголовок окна, ключевые слова. После этого можно перейти к наполнению «тела» документа содержанием. Работу надо периодически сохранять.

7.12. Картинки в HTML-документах

Для вставки в текст картинки используется одиночный маркер <IMG SRC="адрес графического файла"> с одним обязательным атрибутом SRC. Адрес задается так же, как и адрес ссылки. У маркера много необязательных атрибутов, из которых самые важные:

ALT="текст" - текст, который будет выведен на экран, если не выведена картинка. Если не использовать этот атрибут, то пользователи, сидящие на медленных линиях, окажутся в трудном положении, так как у них не будет ни картинки, ни соответствующей надписи вместо нее.

HEIGHT=число - вертикальный размер картинки в пикселях.

WIDTH=число - горизонтальный размер картинки. Смысл этих двух параметров в том, чтобы браузер при выведении на экран оставил место для изображения, и смело помещал дальнейший текст, который можно читать, пока грузится картинка. При несоответствии размеров, картинка будет промасштабирована.

Атрибуты HSPACE, VSPACE, BORDER и ALIGN задают расположение текста относительно изображения. Цветок фейхоа<IMG SRC="feijoa1.jpg" ALT="изображение цветка фейхоа" HEIGHT=162 WIDTH=220 BORDER=5> <P ALIGN=right><IMG SRC="feijoa3.jpg" ALT="изображение плодов фейхоа" HSPACE=100 VSPACE=30 HEIGHT=126 WIDTH=159 ALIGN=TEXTTOP>плоды фейхоа</P> цветок фейхоа, плоды фейхоа.

Не стоит перегружать свои страницы графикой, из-за того, что файлы большого размера медленно передаются по сетям. По этой же причине, фотографии объектов живой природы лучше хранить в формате *.jpg, а рисованные или чертежи с небольшим числом цветов -- в *.gif.

7.13. Размещение страницы в Сети

Как известно, web-странички располагаются на Интернет-серверах, т.е. компьютерах, подключенных к Интернет и выполняющих серверные приложения. Соответственно, чтобы созданную страницу могли увидеть другие люди, необходимо разместить ее на Интернет-сервере. После этого она будет доступна любому пользователю Интернет. Существует большое количество серверов, которые позволяют бесплатно разместить на них web-сайты. Как правило, провайдеры также предоставляют своим пользователям место под web-страницы.

Список бесплатных WEB-серверов можно найти по адресу http://www.homepage.techno.ru/ilya/servers.htm. Список некоторые из них показан на рис.17.

При выборе сервера, на котором предполагается расположить страницу, следует руководствоваться скоростью доступа к нему. Если вы находитесь в Узбекистане и страничку будут просматривать люди, преимущественно живущие в этом же регионе, вряд ли имеет смысл располагать вашу страницу на сервере где-нибудь в Австралии. Гораздо более удобным решением будет разместить её на сервере провайдера. Но если однажды сменится провайдер, то придется переносить страницу на другой сервер, и соответственно адрес ее изменится. Это неудобно. Поэтому за неимением собственного сервера страницы лучше располагать на бесплатных серверах. Для того чтобы бесплатно получить место на сервере для размещения своих страничек, достаточно просто зарегистрироваться, т. е. заполнить небольшую анкету.

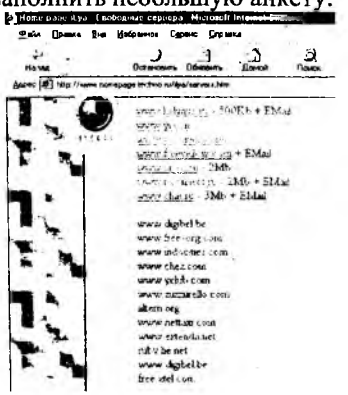


Рис. 17. Перечень серверов Интернет, где можно бесплатно разместить страницу

8. Графика в Web-дизайне

8.1. Форматы данных, используемые для Web-страниц

В Интернет особенную важность приобретают не художественные достоинства картинки, а скорость загрузки. Обычно, если загрузка картинки идет порядка 3-5 минут, посетитель сайта не ждет, а уходит на другие участки Интернет. Так, что одно из главных достоинств картинок для Сети — минимальный размер.

Для размещения изображений в WWW в данное время используются два графических формата — GIF и JPEG. Остальные типы графических файлов WWW-броузеры без применения специальных дополнительных программ (plug-in) не поддерживают.

Основное различие между форматами GIF и JPEG состоит в том, что применяются они для хранения разных видов графики. GIF преимущественно используется для сохранения рисованных изображений, векторной графики, картинок без полутонов, градиентов (переходов от цвета к цвету) и большого количества мелких деталей разного цвета, тогда как JPEG — для фотоизображений и полутоновой графики, живописи, градиентов и изображений с множеством мелких разноцветных деталей.

Причина в том, что формат GIF позволяет сохранять изображение, содержащее не более 256 цветов. Цвета могут быть любыми и в любой комбинации, но общее их количество ограничено этой цифрой. Поэтому в данном формате удобно хранить изображения с небольшим количеством цветов (меньше 256), так как это позволяет за счет сокращения объема информации о цветах значительно уменьшить размер файла, а следовательно, и время загрузки. Помимо того, формат GIF используется для сохранения анимированных картинок и изображений с прозрачными частями — в JPEG такие возможности просто не заложены.

Область же применения формата JPEG определяется тем, что он сохраняет полную цветовую палитру 24-битного изображения (True Color), которая может содержать миллионы цветов. При таком объеме информации размер файла получается очень большим, но главной особенностью формата JPEG является его способность хранить изображение в сжатом, как бы в заархивированном виде. Степень архивации изображения, сохраненного в формате JPEG, может быть очень большой: нормальным считается сжатие картинки в 10–20 раз без видимой потери качества.

При размещении изображения в WWW необходимо помнить о том, что просматривать его будут разные люди и компьютеры у них будут разные. Наверняка попадутся и такие, у которых разрешение монитора будет 640x480 точек (пикселов). Так что есть смысл ограничить размер картинки до максимальной ширины в 600 и длины в 400 точек. Тем более, что такая картинка будет весьма немаленькой по размеру файла и соответственно времени загрузки. Размер картинки можно узнать, открыв

ее в Adobe Photoshop (именно эту программу наиболее удобно для использовать для работы с изображениями) и щелкнув мышкой при нажатой клавише ALT по участку статусной строки слева от черного треугольника. Изменить размеры изображения в Photoshop можно с помощью команды Image Size в меню Image. При этом надо установить флажок в окошко Resample Image.

После изменения размеров изображение обычно несколько теряет резкость и приходится ее восстанавливать. Сделать это можно при помощи фильтра Unsharp Mask в разделе Sharpen, меню Filter.

Поработав с фильтром и установив галочку в окошко Preview, нужно добиться приемлемой резкости изображения, затем нажать OK.

После изменения размеров и корректировки резкости изображения необходимо сохранить картинку. Вариантов сохранения два — GIF и JPEG. Теперь предстоит выбрать, какой формат больше подходит для картинки. Рассмотрим на примере несколько изображений и варианты их сохранения для публикации в WWW.

8.2. Полутоновые изображения (фотографии)

Для фотоизображений характерно большое количество цветов и полутонов, поэтому для них оптимально использование формата JPEG.

При сохранении изображения необходимо отключить опцию Save Thumbnail в меню Save, т. к. это увеличивает размер файла. Сделать это раз и навсегда можно в диалоговом окне Saving Files, которое находится в разделе Preferences, меню File, установив в выпадающем списке Image Preview опцию Never Save.

При сохранении файла используйте имя, состоящее не более чем из восьми латинских символов без заглавных букв и пробелов. Расширение файла также должно быть набрано строчными буквами — например, «photo1.jpg».

После того как набрано имя файла и его расширение, щелкаем кнопку Save, и будет предложено выбрать степень сжатия изображения и тип файла JPEG.

Степень сжатия указывается в разделе Image Options с помощью выпадающего списка и движка Quality. Для большинства изображений опция Medium (3) представляет оптимальное соотношение между качеством и размером файла. Иногда может понадобиться меньшая степень сжатия, а иногда большая – здесь нужен эксперемент. Лучше сохранить несколько вариантов файла с разной компрессией и просмотрите их в браузере. Это позволит выбрать для данного изображения наилучшее соотношение качество / размер файла.

Следующий шаг - оптимизация цветового баланса изображения. Для этого выбирается в нижней части диалогового окна в разделе Format Options переключатель Baseline Optimized. Можно также использовать

новый формат Progressive JPEG, выбрав переключатель Progressive и параметр Scans.

Изображение, сохраненное в формате Progressive JPEG, при загрузке странички в браузер отображается сначала в низком разрешении, а затем постепенно, по мере загрузки, прорисовывается окончательно со всеми деталями. Этот формат обладает рядом существенных недостатков:

- не поддерживается старыми браузерами;
- файл получается несколько более объемным;
- для просмотра требуется большее количество оперативной памяти.

8.3. Рисунки, иллюстрации, фотографии с небольшим количеством цветов

Для публикации подобных изображений в WWW больше подходит формат GIF, т. к. если иллюстрация содержит не слишком много цветов, то, используя возможности этого формата и сохраняя ее в режиме Indexed color, с неполной цветовой палитрой, мы можем получить значительно меньший объем файла, чем при сохранении иллюстрации в формате JPEG. Рассмотрим процедуру сохранения иллюстрации в формате GIF с одновременным уменьшением цветовой палитры изображения и размера файла.

Для сохранения файла в формате GIF в пакете Adobe Photoshop существует специальный модуль GIF89a Export, находящийся в разделе Export, меню File. Использование этого модуля позволяет визуально подобрать наилучшее соотношение между качеством картинки и размером файла, не выходя из диалогового окна Export. Модуль экспорта GIF89a Export также позволяет сохранять изображения с прозрачными элементами. Для этого надо открыть файл в Photoshop и убедиться, что изображение находится в режиме RGB color (меню Image/Mode), а также его размеры и разрешение приемлемы (меню Image/Image Size). Откроем диалоговое окно GIF89a Export.

В данный момент не обращаем внимания на верхнюю часть окна с разделом Transparency From Mask — она понадобится нам позже, а сейчас мы будем работать в нижней части диалогового окна с параметрами Palette и Colors. Параметр Palette управляет выбором палитры цветов, которая будет создана для данного изображения. Можно выбрать один из трех режимов: Exact, Adaptive и System.

В режиме Exact для создания палитры используются цвета, содержащиеся в изображении. В диалоговом окне доступ к этому режиму появляется только, если картинка содержит не более 256 цветов. Exact позволяет получить оптимальное качество изображения при приемлемом размере файла.

Режим Adaptive используется в тех случаях, когда число цветов в изображении больше 256, и потому режим Exact недоступен. Для создания палитры вы указываете количество цветов, из которых следует ее

сформировать. Оно может варьировать от 1 до 256 Этот параметр устанавливается в окошке Colors.

System — режим создания палитры из таблицы цветов, используемых в данный момент операционной системой компьютера. При использовании этого режима нельзя предсказать, как будет выглядеть сохраненное изображение на мониторах компьютеров, находящихся под управлением других ОС. Так что пользоваться этим вариантом сохранения не стоит.

Если выбрана палитра Exact, можно смело нажимать на кнопку OK и сохранять изображение. При сохранении файла следует использовать имя, состоящее не более чем из восьми латинских символов без заглавных букв и пробелов. Расширение файла также должно быть набрано строчными буквами — например, «menu1.gif». Если сохраняется изображение с адаптивной палитрой, то нужно выбрать или впечатать в окошко Colors минимальное количество цветов, которыми можно передать все необходимые детали вашего изображения. Влияние уменьшения количества цветов в палитре на размер файла зависит от конкретной картинки, ее размеров и цветового содержимого. Для очень маленьких изображений существенное сокращение количества цветов не приведет к значительному уменьшению размеров файла.

После выбора палитры и количества цветов вы можете посмотреть, как при таких параметрах сохранения изображение будет выглядеть в Web-броузере. Для этого щелкните по кнопке Preview. Для просмотра изображения выберите инструмент «Рука» и поводите им по изображению при нажатой кнопке мыши. Если ваше изображение слишком мало, вы можете увеличить масштаб просмотра с помощью инструмента «Лупа». Для уменьшения масштаба используйте лупу с нажатой клавишей ALT. Если получилось, щелкайте OK и попробуйте еще немного уменьшить количество цветов в палитре изображения. Затем снова нажмите Preview и оцените качество картинки. Достигнув наименьшего приемлемого количества цветов, нажмите OK и сохраните изображение.

9. Технология Macromedia Flash

9.1. Понятие технологии Macromedia Flash

Эта технология появилась примерно два-три года тому назад, как набор средств для создания и просмотра так называемых Flash Movie. Средством создания является редактор Flash или Director, просматривать же клипы позволяет Flash Plug-in, автоматически скачивающийся при заходе на страницу, содержащую клип. Один Flash Movie может полностью заменить по функциональности HTML - форматированную страницу любой сложности и даже расширить возможности HTML и JavaScript. Визуальное отличие состоит в целостности, в качестве и анимации близкой к видео. Нечто подобное можно добиться средствами

Java, но это не так. По сравнению с Java-апплетами Flash Movie гораздо легче создавать. Все чаще Flash Movie заменяют gif-файлы в роле баннеров, так как выигрывают у последних по размеру и анимации.

9.2. Векторная графика

Здесь кроется причина столь малых размеров правильно собранного Flash Movie. Этот секрет состоит в том, что Macromedia Flash изначально задумывалась, как технология для показа векторной графики на Web-страницах. Преимущества векторной графики очевидны! Представьте, что каждое изображение состоит из простых геометрических фигур: прямые, окружности, многоугольники... Сколько же байт нам надо для представления одной простой фигуры? Рассмотрим окружность, взяв целый тип integer для хранения каждого числа. Всего нам понадобится шесть чисел: радиус, координата центра X, координата центра Y, цвет заливки, цвет рамки и какое-либо служебное число-идентификатор окружности. По приблизительным подсчетам, потребуются 24 байта. Теперь, проанализировав то количество памяти, которое может потребоваться для хранения пусть даже сотни простых фигур, становится понятно, что эта сотня угольников и окружностей потребует гораздо меньше байт, чем такое же GIF или JPG изображение. Заметьте, что в векторной графике размеры фигур не играют роли.

9.3. Flash Movie

В основу Flash Movie положен принцип видеоклипа, т.е. Movie разбивается на кадры. Однако кадров, как таковых, было бы недостаточно для реализации ссылок, форм и анимации близкой к видео. Имеются еще и ключевые (главные) кадры, слои, сцены и объекты: кнопка, клип, картинка. Кадры можно, а иногда необходимо, объединять в так называемые действия (motion tween). Они позволяют ускорить процесс создания движения/изменения объекта и оптимизировать клип за счет сокращения числа ключевых кадров. С помощью действий можно, разбив объект на пиксели, симулировать его превращение в другой объект. Допускаются вложенные клипы, клипы в кнопках и т.п. Поддерживаются формы. Важной частью является встроенный язык программирования схожий с JavaScript. Все эти средства позволяют сделать привлекательный баннер, клип или полностью оформить Web-страницу.

9.4. Редактор Flash 5

Является основным инструментом для создания Flash Movie на сегодняшний день. Внешний вид Flash 5 напоминает простой графический редактор: панель рисования, линейка меню, но на этом сходства заканчиваются. Стандартная палитра предлагает набор цветов, имеющих максимальное сходство с теми, что использует браузер. На первый взгляд, невзрачная полоса кадров на самом деле является основным элементом редактора. В ней происходит управление кадрами, действиями, слоями и

пользуетесь средствами встроенного языка программирования. Незаменимыми в работе являются меню insert, control, modify и отдельное окошко для управления сценами и объектами. Редактор позволяет озвучивать кадры, конвертируя затем все звуки в формат mp3 с указанными параметрами. В общем, редактор Macromedia Flash 4 позволяет реализовать все аспекты современной Flash технологии.

10. Полезные советы при создания Web-страниц

10.1. Совет первый. "Побольше мелких деталей"

Дело в том, что крупные объекты в составе любой композиции смотрятся довольно неважно. Нужно быть большим мастером, чтобы дизайн, основанный на таких объектах, выглядел хорошо. Аршинные буквы в заголовках, кнопки навигации высотой в 40 пикселей, верстка в одну колонку шириной в 600 точек, разделитель одного цвета, растянутый на весь экран, - все это придает дизайну непрофессиональный вид.

Если же добавить в элементы дизайна мелкие детали, а крупные объекты визуально разбить на более мелкие (например, применив градиентную заливку) - картина значительно улучшится.

Вот что конкретно можно предпринять для "размельчения" дизайна:

1) если позволяет содержание сервера, заголовки, набранные шрифтом большого размера, дополните подзаголовками, выполненные более мелким кеглем;

2) по возможности применяйте верстку в несколько колонок. При этом "разбивайте" колонки и по вертикали, выделяя их части, например, разными цветами (или оттенками одного цвета);

3) не заливайте большие объекты (например, линии-разделители) одним цветом. Выберите градиентную заливку или просто разделите объект на несколько частей с помощью линий и других графических примитивов;

4) не делайте кнопки навигации, заголовки колонок и другие подобные объекты однотонными. Придумайте какую-нибудь интересную рамку, визуально выделите часть объекта (например, угол), добавьте тень;

5) добавьте там, где, на ваш взгляд, это требуется, декоративные элементы: пиктограммы, горизонтальные полоски или даже что-нибудь типа орнамента.

Естественно, бросаться в крайности не нужно. Чувство меры - такое же необходимое качество дизайнера, как и буйная фантазия. Не делайте так, чтобы от мелких деталей у зрителя рябило в глазах. Следите, чтобы все заголовки, кнопки навигации, текст читались хорошо. Добавленные вами мелкие детали не должны сбивать с толку читателя: он не должен принимать их, например, за элементы навигации и пытаться ткнуть в них мышью.

10.2. Совет второй. "Повышенное внимание к мелочам"

Если в жизни результатом пренебрежения мелочами становятся человеческие жертвы и многомиллиардные убытки, то в Интернете беспечность дизайнера, хотя и имеет гораздо менее катастрофические последствия, тоже не приносит ничего хорошего. Отзывы разочарованных посетителей сайта и насмешки коллег по цеху в конечном итоге создадут дизайнеру дурную репутацию. Поэтому каждый профессиональный дизайнер уделяет повышенное внимание мелочам и никогда не позволяет себе халатного отношения к делу. Продумывайте до конца все элементы дизайна.

Необходимо следить, чтобы каждый из элементов был на своем месте и хорошо сочетался с другими. Не ленитесь потратить лишний час на дизайн кнопки размером 15x15 пикселей - вам обязательно воздастся. Не пожалейте времени, чтобы выяснить, какой именно тег в 10 килобайтах вашего HTML-кода вызывает смещение таблицы на всего лишь один ничтожный пиксель. Из тщательно "вылизанных" в графическом редакторе кнопок и пиктограмм, грамотно составленного текста, отлаженного HTML-кода, как из кирпичиков, и складывается профессиональный дизайн.

10.3. Совет третий. "Гармоничное сочетание цветов"

Сочетание цветов на сайте должно быть таково, чтобы, во-первых, текст, размещенный на веб-страницах, читался хорошо, а во-вторых, общий цветовой фон не должен вызывать у пользователя раздражения.

Критерий оценки допустимости того или иного сочетания цветов для использования их в качестве цветов фона и текста таков: чтение текста не должно вызывать у пользователя никакого дискомфорта. Критерий достаточно субъективный, но если текст на странице выглядит недостаточно контрастно, или для его чтения нужно всматриваться в экран, то данное сочетание цветов не годится. Однако, дело не только в контрастности. Например, в ярко-зеленом тексте на темно-синем фоне контраста - хоть отбавляй, но любой профессиональный дизайнер такое сочетание использовать не будет. Если данный способ применить не получается, то надо сделайть скриншот страницы, а затем в Photoshop переведите ее в режим grayscale (градации серого): так можно проверить читаемость текста. Большие тексты (например, статьи) лучше размещать на светлом фоне, так как в этом случае их читать гораздо легче. Для отдельных же предложений и слов можно выбрать и темный фон.

Теперь о выборе цветов для дизайна в целом. С одной стороны, зеленый с розовым явно не сочетаются, но, с другой стороны, в Сети полно зелено-розовых сайтов. То есть многие дизайнеры в принципе не способны определить, какие цвета сочетаются, а какие нет.

Имеется "научная" методика выбора цветов, которые хорошо сочетаются друг с другом. Она включает в себя выбор тона по цветовому

кругу, соотношение котраста и яркости. Но для web-страниц она не вполне подходит. Дело в том, что на веб-странице самые, казалось, невероятные сочетания цветов могут смотреться органично. Главное – надо экспериментировать.

Сначала выбераются какие-нибудь несколько цветов - пусть даже те, которые уже давно надоели. Нарисуйте ими какую-нибудь простую композицию (надпись на фоне нескольких прямоугольников, например). Далее сохраните получившуюся картинку в GIF или JPEG и загрузите в Internet Explorer. Теперь - финальный шаг: нажмите Ctrl+A или выберите из меню команду пункт "Правка-Выделить все". Выделение инвертирует цвета, и получаются довольно интересные сочетания. А теперь, если понравилось, делайте скриншот и используйте найденные цвета на здоровье, а если нет - продолжайте эксперименты.

10.4. Совет четвертый. "Гармоничное сочетание шрифтов"

Повышенное внимание к шрифтам со стороны любого профессионального дизайнера вполне закономерно. Текст - важнейший элемент дизайна всех веб-сайтов, на которые ходят посетители - ведь ходят они за информацией, представленной в большинстве случаев в текстовой форме. Исполнение логотипа, заголовков, основного текста во многом определяют внешний вид сайта.

Как и в случае с подбором цветов, здесь нужно брать в расчет несколько важных моментов.

Во-первых, не стоит в дизайне одного сайта использовать много разных шрифтов. Вполне достаточно одного-двух. Особенно хорошо действие этого правила заметно в полиграфии. В подавляющем количестве печатных изданий и других публикаций используется один шрифт с засечками для набора основного текста и рубленый (без засечек) шрифт для заголовков. Да взять хотя бы текстовый редактор MS Word: в качестве стандартных стилей он предлагает шрифт Times New Roman для обычного текста и Arial - для заголовков.

Естественно, совершенно необязательно набирать весь текст шрифтом одного и того же размера и начертания. Можно экспериментировать. Только, как всегда, нужно придерживаться "логического единообразия": если решено выделять важные участки текста полужирным, то не нужно какой-нибудь из абзацов непонятно почему делать подчеркиванием. А если для текстового меню навигации применено <FONT SIZE-1>, то ни в коем случае не нужно половину пунктов меню делать в <FONT SIZE+1>.

Во-вторых, не следует использовать декоративные и стилизованные шрифты - например, рукописные, имитирующие штамповку на военный манер или буквы на LCD-дисплее и т.п. О них нужно забыть и вспоминать только тогда, когда понадобится сделать именно стилизацию под конкретную тематику или временную эпоху.

В основном профессиональные дизайнеры не любят использовать декоративные гарнитуры, потому что эти шрифты сами по себе слишком сложны, чтобы являться всего лишь средством для передачи информации, опубликованной на сайте. Декоративная гарнитура представляет собой законченную композицию, тогда как в большинстве случаев она сама должна являться составной частью дизайна. В результате применения таких шрифтов трудно добиться хорошей сочетаемости всех элементов, формирующих внешний вид сайта, так как гарнитура буквально лезет на передний план. Пожалуй, единственная возможность избежать негативных последствий выбора декоративной гарнитуры - это использовать ее при оформлении небольших по размеру и второстепенных по значению элементов.

Кроме того, шрифты, как и многие другие вещи, подвержены влиянию моды, и декоративные шрифты - особенно. Большинство из них уже устарело и выглядит точно также, как одежда, которая пользовалась успехом 10 лет назад - достаточно нелепо. Многие же классические шрифты популярны уже десятки лет - например, Futura был создан в 1928 году, а Garamond ведет свою историю вообще с XV века.

Тем не менее, даже если для разработки дизайна использована пара каких-нибудь классических шрифтов, это еще не гарантирует того, что их сочетание будет смотреться органично. Например, однотипные, но немного разные шрифты (скажем, рубленые Arial и Futura) в паре выглядят плохо. Здесь нужно полагайтеся на свой художественный вкус и интуицию.

Заключение

Новые системы коммуникаций и цифровые технологии до неузнаваемости изменяют наш образ жизни. Теперь вполне обычным кажется то, о чем десять лет назад нельзя было и мечтать. Можно просмотреть фонды местной библиотеки - или библиотеки на другой стороне земного шара; работать со своим банковским счетом, делать покупки через Интернет, обучиться, не выходя из собственного дома, и все это с помощью компьютера и модема.

Снижение стоимости в сочетании с повышением производительности приносят все более и более мощные компьютеры и цифровые системы все большему и большему количеству людей. Сеть, с ее быстро растущей коллекцией баз данных и прочих источников информации, уже не ограничивается промышленно развитыми странами Запада, - она раскинулась нынче от одной стороны земного шара до другой. А цены компьютеров и модемов, которые к ней подключаются, продолжают снижаться, и все больше и больше людей могут это себе позволить.

Гиперпространство стало жизненно важной частью ежедневного распорядка миллионов людей. По электронной связи завязываются знакомства, находят друзей, партнеров, спонсоров, а все начинается с контакта в гиперпространстве, этом иллюзорном месте, поглощающем границы государств и наций. Коммерческие сделки выполняются в коде ASCII. На электронных связях начинаются политические и общественные движения, объединяющие людей за тысячи миль друг от друга.

И это только начало. Мы живем в веке коммуникаций, но средства этих коммуникаций остаются сильно разделенными. Но когда-нибудь наш телефон, телевизор, факс и персональный компьютер будут заменены единым "информационным процессором", привязанным к всемирной Сети оптоволоконными соединениями.

На кончиках пальцев будет не только доступ к базам данных и библиотекам, но и мощные социальные связи. Имея взаимоотношения с тысячами, с миллионами людей, вы сможете участвовать в социальных и политических движениях во всей стране и во всем мире.

Как это произойдет? Частично - от новой технологии. Высокого разрешения телевизоры потребуют разработки недорогих компьютеров, которые смогут обрабатывать объемы информации, доступные сегодня рабочим станциям. Телефонные и телеграфные компании в сотрудничестве, а иногда и конкурируя, проведут в ваш дом оптоволоконные линии.

Создание сети "информационных суперхайвеев" сравнимо по масштабу с системой автодорожных хайвеев пятидесятых годов.

Сейчас мы имеем компьютерно-сетевой эквивалент начала пятидесятых, как раз перед созданием массивной сети хайвеев. Несомненно, интересного очень много. Но чтобы до него добраться, надо петлять по «двухрядным дорогам» и иметь хорошую карту.

Создание этой новой Сети потребует не только высокоскоростных каналов и коммутационного оборудования, - потребуется новая концепция коммуникаций - Сеть как служба информации. Сеть остается достаточно сложным и загадочным местом. Чтобы получить сегодня что-то от сети, нужно потратить приличное время либо общаясь с ее ветеранами, либо читая учебник, либо приобретая свой опыт на собственных ошибках.

Системные администраторы Интернет начали уже понимать, что не все желают изучать тонкости операционных систем, и что отказ это делать не ставит людей вне морали. Мы уже имеем простые интерфейсы, которые позволяют миллионам людей воспользоваться всей мощью Сети. Можно заметить эту тенденцию в браузерахз Всемирной Сети (World Wide Web), которые не требуют изощренных навыков обращения с компьютером, но открывают ворота к тысячам источников информации.

О службах Интернет. На каждую базу данных, к которой есть доступ из Интернет, сегодня приходится три или четыре, к которым такого

доступа нет. Правительственные ведомства только теперь начинают подсоединять свои хранилища информации к Сети. Некоторые коммерсанты - от владельцев баз данных до владельцев книжных магазинов, предоставляют доступ к своим ресурсам через Сеть.

Сейчас неплохо используются одно из самых интересных приложений Сети. Есть стандарт, называемый MIME, который позволяет пересылать по Сети звуковые и графические файлы в виде сообщения. По электронной почте можно получить звуковое сообщение или фотографию. Наконец, с помощью этого стандарта по сети можно будет передавать даже небольшие видеосюжеты.

Все это требует расширения возможностей Сети, которая должна будет работать и с миллионами новых пользователей, и с новыми приложениями, которые им понадобятся. Воспроизведение движущегося образа на экране компьютера требует колоссального количества бит компьютера и соответствующих мощностей, чтобы с ними работать.

Из всего этого формируется Национальная Информационная Инфраструктура, способная передавать в секунду миллиарды бит информации - это столько, сколько нужно, чтобы присоединить информационные "рукава" к каждому предприятию и к каждому дому.

По мере роста этих "суперхайвеев" будут развиваться и "развязки", "ответвления", потому что мало пользы от высокоскоростных магистралей, если на них нельзя въехать. Цены на модемы, похоже, будут падать, как и на компьютеры. Все больше людей сможет позволить себе высокоскоростные модемы. И в конце концов, дома могут быть присоединены прямо к национальной цифровой сети. Большая часть дальних телефонных разговоров уже передается в цифровой форме по оптоволоконным линиям. Телефонные компании думают о том, чтобы провести такие линии и на "последней миле" - подводу к дому. Фонд Электронной Границы (Electronic Frontier Foundation) работает над тем, чтобы сделать цены на такие связи приемлемыми.

Помимо технических вопросов, возникают острые вопросы социального, политического и экономического характера. Кому следует предоставить доступ к таким услугам, и сколько они будут стоить? Если мы живем в информационном веке, не придется ли нам сегодня сеять семена, из которых вырастет информационно подавленный класс, который не сможет конкурировать со субъектами, имеющими достаточно денег и умения, чтобы манипулировать новыми каналами коммуникаций?

Каковы законы электронной границы? Там, где теряются в гиперпространстве национальные и государственные границы?

Информационные технологии являются мощным катализатором возможного создания действительно интегрированной мировой экономики и общества в целом. Информационная революция только начинается...

Использованная литература

1. Гуломов С.С., Шермухамедов А.Т., Бегалов Б.А.. Иктисодий информатика. – Т.:Узбекистон, 1999.
2. Информационные технологии в бизнесе. Под ред. Железны М. - С.-П.: «Питер», 2002.
3. Смирнов С.Н. Электронный бизнес. . - М.: АйТи, 1993.
4. .Вескес Джон Л, Гандерлоу Майк, Чипмен Мэри. Access и SQL Server. . - М.: "Лори", 1997.
5. Холмогоров В.. Интернет-маркетинг. Краткий курс. - С.-П.: «Питер», 2002.
6. Холмогоров В. Основы Web-мастерства. Учебный курс.- С.-П.: «Питер», 2001.
7. Чупалов А. Как зарабатывать деньги в Internet.- С.-П.: «Питер», 1997.
8. Нортон П., Станек У. Программирование на JAVA. . - М.: "СК Пресс", 1996.
9. Фролов А., Фролов Г. Базы данных в Интернете. - М.:,"Русская редакция", 2000.
10. Козье Д.. Электронная коммерция. - М.:, Изд."Русская редакция". 1999.
11. Фролов А.В., Фролов Г.В.. Глобальные сети компьютеров. - М.:,Изд. "Диалог-МИФИ", 1996.
12. Волков С., Достов В. Платежные механизмы современного Интернета. Мир Internet 2000, №5, с. 22-28
13. Медведовский И.Д., Семьянов П.В., Леонов Д.Г. Атака на Internet. - М.: Изд. ДМК., Москва, 1999.
14. Мельников В.В. Защита информации в компьютерных системах. – М.:Финансы и статистика, 1997. -364 с.

Содержание

Contents

Баходыр Юнусович Ходиев,
Татьяна Ивановна Сарсатская

ТЕХНОЛОГИИ ИНТЕРНЕТ
Учебное пособие

Редактор Мадумарова Г.
Компьютерная верстка Сафиев Р.

Подписано в печать 30. 10.2003. Формат 60 x80 1/16 Объем 6,9 п.л.
Тираж2000экз. Заказ № 180 Цена договорная

Отпечатано в типографии ТГЭУ на множительной технике «RISO»