ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

5А330203-“Прикладная информатика”

КУРСОВАЯ РАБОТА

Тема: Анализ программного обеспечения Web-технологий.

Работу выполнил:
студент группы 203-12
Зиядуллаев С.О.

Принял:

ТАШКЕНТ 2014

Содержание

I. Введение

World Wide Web – глобальная компьютерная сеть на сегодняшний день содержит миллионы сайтов, на которых размещена всевозможная информация. Люди получают доступ к этой информации посредством использования технологии Internet. Для навигации в WWW используются специальные программы – Web-браузеры, которые существенно облегчают путешествие по бескрайним просторам WWW. Вся информация в Web-браузере отображается в виде Web-страниц, которые являются основным элементом байтов WWW.

Web-страницы, поддерживая технологию мультимедиа, объединяют в себе различные виды информации: текст, графику, звук, анимацию и видео. От того, насколько качественно и красиво сделана та или иная Web-страница, зависит во многом ее успех в Сети.

Пользователю приятно посещать те Web-страницы, которые имеют стильное оформление, не отягощены чрезмерно графикой и анимацией, быстро загружаются и правильно отображаются в окне Web-браузера.

Создать качественную Web-страницу непросто, для этого потребуются не только навыки дизайнера для красивого и стильного оформления, но и опыт программирования. Сложность и объем программ, требуемых для создания того или иного сайта, зависит от множества факторов, таких, как структура, цель, требуемая функциональность, обеспечение надежности и т.п.

В своей курсовой работе я сделал попытку разобраться в том, что необходимо знать и уметь для создания Web-страницы, какое программное обеспечение является инструментарием создания Web-страниц и как его эффективно использовать.

II. Обзор технологий
1. Языки описания документов

1.1. Язык гипертекстовой разметки HTML

World Wide Web, или, что-то же самое Всемирная паутина, WWW составляет основную компоненту глобальной компьютерной сети Интернет. Зародившись еще в 60-е гг., Интернет долгое время использовалась лишь узким кругом специалистов для обмена информацией по электронной почте. Сеть управлялась операционной системой UNIX - для научных целей это подходило, но достаточно сложный текстовый интерфейс UNIX'a существенно ограничивал масштабы применения сетевых технологий. Годом рождения World Wide Web считается 1989 - в этом году был изобретен язык, ставший впоследствии основным языком web-документов - это язык - HTML (HyperText Markup Language - язык разметки гипертекста).

Автором языка HTML является Тим Бернерс-Ли (Tim Berners-Lee), выпускник Оксфордского университета, работавший в то время по контракту в Женеве, в Европейской лаборатории физики элементарных частиц (CERN, Conseil Europeen pour la Recherche Nucleaire) консультантом по программному обеспечению. CERN - достаточно большая организация, и потому, чтобы лучше ориентироваться в ее структуре, не хранить в памяти данные о большом количестве проектов, должностных лиц и пр., Тим Бернерс-Ли разработал для своего личного пользования программу Enquire, на основе которой через несколько лет для лаборатории было создано своего рода информационное пространство. Программа Enquire позволяла осуществлять так называемый "нелинейный" поиск документов - т.е. переходить от одного документа к другому без обращения к оглавлению или справочнику.

Язык HTML составляет основу технологии гипертекста. Гипертекстовый документ содержит так называемые гиперссылки.

Текстовые гиперссылки обычно выделяются цветом и подчеркиванием, хотя это необязательно. Гиперссылки позволяют пользователю немедленно перейти к нужному документу, независимо от места его физического расположения. Это существенно облегчает поиск информации, разбросанной по всему Земному шару. Технология гипертекста позволила связать всю совокупность опубликованных в Интернет документов в единую систему - WWW.

Язык разметки документов - это набор специальных инструкций, называемых тэгами, предназначенных для формирования в документах какой-либо структуры и определения отношений между различными элементами этой структуры. Тэги языка, или, как их иногда называют, управляющие дескрипторы, в таких документах каким-то образом кодируются, выделяются относительно основного содержимого документа и служат в качестве инструкций для программы, производящей показ содержимого документа на стороне клиента. В самых первых системах для обозначения этих команд использовались символы “<” и “>”, внутри которых помещались названия инструкций и их параметры. Сейчас такой способ обозначения тэгов является стандартным.

Когда осенью 1991 года Интернет впервые услышал позывные новой технологии, название которой легко уместилось в три буквы, почти никто не мог представить себе, что завоевания ее окажутся настолько глобальными. Популярность World Wide Web и неотъемлемой ее части, HTML, безусловно, стала причиной повышенного внимания к системам гипертекстовой разметки документов. Хотя понятие гипертекста было введено В.Бушем еще в 1945 году и, начиная с 60-х годов стали появляться первые приложения, использующие гипертекстовые данные.

Использование гипертекстовой разбивки текстового документа в современных информационных системах во многом связано с тем, что гипертекст позволяет создавать механизм нелинейного просмотра информации. В таких системах данные представляются не в виде непрерывного потока текстовой информации, а набором взаимосвязанных компонентов, переход по которым осуществляется при помощи гиперссылок.

Язык HTML не является собственно языком программирования; это есть средство описания структуры документа, его стиля и связей его с другими документами. Для просмотра Web-документов используются специальные программы - так называемые браузеры (англ.: to browse - 1) пастись, ощипывать побеги; 2) читать, заниматься беспорядочно, урывками.) Вообще-то, возможности браузеров много шире, но пока ограничимся их определением как средством просмотра web-документов. Именно браузерам Интернет обязана своей популярностью.

Создание языка HTML в 1989 году облегчило обмен информацией в пределах лаборатории CERN, однако это не решало проблем, связанных обменом информацией между сотрудниками лаборатории и их коллегами, работавшими в разных странах. Для того, чтобы такой обмен был возможен, необходимо было создать *децентрализованную информационную систему*, основанную на некоем *стандарте* обмена данными. Таким стандартом стал созданный Тимом-Бернерсом Ли в 1990 году протокол HTTP (HyperText Transfer Protocol, протокол передачи гипертекстовых файлов).

HTML является упрощенной версией стандартного общего языка разметки - SGML (Standart Generalised Markup Language), который был утвержден ISO в качестве стандарта еще в 80-х годах. Этот язык предназначен для создания других языков разметки, он определяет допустимый набор тэгов, их атрибуты и внутреннюю структуру документа. Контроль за правильностью использования дескрипторов осуществляется при помощи специального набора правил, называемых DTD- описаниями(более подробно о DTD мы поговорим чуть позже), которые используются программой клиента при разборе документа. Для каждого класса документов определяется свой набор правил, описывающих грамматику соответствующего языка разметки.

С помощью SGML можно описывать структурированные данные, организовывать информацию, содержащуюся в документах, представлять эту информацию в некотором стандартизованном формате. Но в виду некоторой своей сложности, SGML использовался, в основном, для описания синтаксиса других языков(наиболее известным из которых является HTML), и немногие приложения работали с SGML- документами напрямую.

Гораздо более простой и удобный, чем SGML, язык HTML позволяет определять оформление элементов документа и имеет некий ограниченный набор инструкций - тэгов, при помощи которых осуществляется процесс разметки. Инструкции HTML, в первую очередь, предназначены для управления процессом вывода содержимого документа на экране программы-клиента и определяют этим самым способ представления документа, но не его структуру. В качестве элемента гипертекстовой базы данных, описываемой HTML, используется текстовый файл, который может легко передаваться по сети с использованием протокола HTTP. Протокол HTTP позволил передавать по сети HTML-документы.

Появилась возможность организовать доступ многочисленных пользователей (клиентов) к HTML-документам, находящимся на так называемых *серверах* (англ.: to serve - служить, обслуживать). Ясно, что и на компьютере пользователя, с которого происходит обращение к серверу, и на сервере должно быть установлено специальное программное обеспечение:- сервер должен не только хранить документы, но и уметь быстро *находить* их по запросу клиента;- клиент (браузер) должен распознать HTML-код и *визуализировать* его,

представить в легко воспринимаемом виде.Тим-Бернерс Ли написал первый браузер (который он назвал World Wide Web) и первый web-сервер (info.cern.ch). В 1991 году это программное обеспечение стало доступно достаточно широкому кругу пользователей Интернет. Он также разработал *схему адресации* к web-документам в Интернет. Тим-Бернерс Ли назвал ее *Universal Resource Identifier* (*URI*, универсальный идентификатор ресурсов). (Сейчас ее назвают URL—Uniform Resource Locator, унифицированный указатель ресурса.) Таким образом, разработка языка HTML, протокола HTTP, web-сервера, браузера и системы адресации (URI) явились основой развития Всемирной паутины (WWW).

Отсутствие единого стандарта, которого придерживались бы разработчики браузеров и web-серверов, привело бы к тому, что невозможно было бы создать сайта, который одинаково выглядел бы в разных браузерах. Пришлось бы под каждый браузер писать свой сайт.

Во избежание этого по инициативе Тима-Бернерса Ли в июле 1994 года на базе Массачуссетского технологического института (Massachussets Institute of Technlogies, MIT) был создан World Wide Web consortium (или просто W3C) (Тим-Бернерс Ли возглавил его). Основная цель W3C - обеспечить как можно большую совместимость программного обеспечения web-публикаций. W3C не является административным органом, это нечто вроде форума для выработки компромиссных решений в области web-технологий. Консорциум принимает на рассмотрение любые проекты и предложения. Спецификации, разработанные W3C, не обязательны для применения, но консорциум ведет работу по их пропаганде. HTML был ратифицирован World Wide Web Consortium.

1.2. Язык гипертекстовой разметки XML

За короткий срок своего официального существования язык XML привлек к себе уже достаточно много внимания со стороны разработчиков и пользователей Интернет. Несмотря на то, что XML очень молод (международная организация W3C утвердила спецификацию "Extensible Markup Language(XML) 1.0" в начале февраля 1998г.), уже сегодня появляются новые языки, созданные на основе XML, возникают многочисленные Web-сервера, использующие эту технологию для организации хранящейся на них информации. Мир Интернет вокруг нас в очередной раз преображается, и мы можем стать участниками этого процесса уже сегодня

Для чего нужен новый язык разметки?

Самый популярный на сегодняшний день язык гипертекстовой разметки – HTML, был создан специально для организации информации, распределенной в сети Интернет, и является одной из ключевых составляющих технологии WWW. С использованием гипертекстовой модели документа способ представления разнообразных информационных ресурсов в сети стал более упорядочен, а пользователи получили удобный механизм поиска и просмотра нужной информации.

Однако современные приложения нуждаются не только в языке представления данных на экране клиента, но и в механизме, позволяющем определять структуру документа, описывать содержащиеся в нем элементы. HTML обладает несложным набором команд и вполне успешно справляется с задачей описания текстовой информации и отображением ее на экране программы просмотра - броузера. Однако сами отображаемые данные никак не связаны с теми тэгами, которые используются для форматирования, поэтому у программ-анализаторов нет возможности использовать тэги HTML для поиска нужных нам фрагментов документа. Т.е. встретив, например, такое описание

rose,

программа просмотра будет знать, каким цветом отобразить текст, содержащийся внутри тэгов и, вероятно, отобразит его правильно, но ей абсолютно безразлично, в каком месте документа встретился этот тэг, в какие другие тэги заключен текущий фрагмент, существуют ли вложенные в него фрагменты, правильно ли построены отношения между объектами. Такое "безразличие" к структуре документа приводит к тому, что поиск или анализ информации внутри него ничем не будет отличаться от работы со сплошным, не разбитым на элементы текстовым файлом. А это, как известно, не самый эффективный способ работы с информацией.

Другим существенным недостатком HTML можно назвать ограниченность набора его тэгов. DTD- правила для HTML определяют фиксированный набор дескрипторов и поэтому у разработчика нет возможности вводить собственные, специальные тэги. Хотя время от времени появляются новые расширения языка(на сегодняшний день последней версией HTML является HTML 4.0), но долгий путь их стандартизации, сопровождаемый постоянными разногласиями между основными производителями броузеров делают практически невозможной быструю адаптацию языка, его использование для отображения специализированной информации(например, мультимедийной, математических, химических формул и т.д.).

Подводя итог всему сказанному, можно утверждать, что HTML уже сегодня не удовлетворяет в полной мере требованиям, предъявляемым современными разработчиками к языкам подобного рода. И ему на смену был предложен новый язык гипертекстовой разметки, мощный, гибкий, и, одновременно с этим, удобный язык XML. В чем же заключается его достоинства?

XML (*Extensible Markup Language*) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Таким образом, если, например, мы считаем, что для обозначения элемента *rose* в документе необходимо использовать тэг <*flower*>;, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему:

<flower>rose</flower>

Набор тэгов может быть легко расширен. Если, предположим, мы хотим также указать, что описание цветка должно по смыслу идти внутри описания оранжереи, в которой он цветет, то просто задаем новые тэги и выбираем порядок их следования:

```
<conservatory>
<flower>rose</flower>
</conservatory>
```

Если мы хотим посадить туда еще несколько цветочков, то должны внести следующие изменения:

```
<conservatory>
<flower>rose</flower>
<flower>tulip</flower>
<flower>cactus</flower>
</conservatory>
```

Как видно, сам процесс создания XML документа очень прост и требует от нас лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру,

строит необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

Еще одним из очевидных достоинств XML является возможность использования его в качестве универсального языка запросов к хранилищам информации. Сегодня в глубинах W3C находится на рассмотрении рабочий вариант стандарта XML-QL(или XQL), который, возможно, в будущем составит серьезную конкуренцию SQL. Кроме того, XML-документы могут выступать в качестве уникального способа хранения данных, который включает в себя одновременно средства для разбора информации и представления ее на стороне клиента. В этой области одним из перспективных направлений является интеграция Java и XML - технологий, позволяющая использовать мощь обеих технологий при построении машинно-независимых приложений, использующих, кроме того, универсальный формат данных при обмене информации.

XML позволяет также осуществлять контроль за корректностью данных, хранящихся в документах, производить проверки иерархических соотношений внутри документа и устанавливать единый стандарт на структуру документов, содержимым которых могут быть самые различные данные. Это означает, что его можно использовать при построении сложных информационных систем, в которых очень важным является вопрос обмена информацией между различными приложениями, работающими в одной системе. Создавая структуру механизма обмена информации в самом начале работы над проектом, менеджер может избавить себя в будущем от многих проблем, связанных с несовместимостью используемых различными компонентами системы форматов данных.

1.3. Dynamic HTML

До появления версий 4.0 Internet Explorer и Netscape Navigator сценарии могли изменять содержание и внешний вид страниц, только используя метод `write.document` при загрузке страницы.

Любой сценарий, выполняемый позже, может посылать команды броузеру, такие как запрос на загрузку новой страницы, вывод окна предупреждения или изменение значения поля формы, но как только страница загружена, сценарий не может изменить HTML-код. Как же сделать Web- страницы интерактивными, если их невозможно изменить в ответ на действия пользователя?

Динамический HTML, частично реализованный в Netscape 4 и в большей степени в Internet Explorer 4, устраняет эти ограничения. С использованием динамического HTML сценарии могут вставлять блоки HTML, удалять и заменять их или изменять свойства объектов после отображения страницы на экране. Броузер автоматически обновляет новые свойства и (или) новый HTML-код. Динамический HTML строится на двух принципах.

- Объектная модель документа (DOM) описывает способ организации и названия объектов в броузере, а также определяет, какие объекты и свойства могут быть изменены и какие значения они могут принимать.
- Событийная модель описывает способ передачи управления сценариям - какие действия посылают сценарии на выполнение.

1.4. Macromedia Flash

World Wide Web (Всемирная паутина) развивается очень стремительно. Традиционные *Web-сайты (Web-sites) со* статическими информационными страницами уже не привлекают как прежде внимание конечных пользователей. Поэтому Web-дизайнеры и разработчики пытаются придать своим страницам неотразимый вид, вводя графику и

файлы анимационного формата GIF. Несмотря на то, что применение графики в традиционных форматах сжатия изображений GIF, JPEG и PNG придает сайту некоторый эффект, тем не менее, загрузка таких страниц может отнять много времени из-за больших размеров графических файлов. Альтернативным инструментом создания анимации является программный продукт— **Macromedia Flash**, который поможет добиться поставленных при Web-разработке целей.

По сравнению с другими графическими приложениями, предназначенными для публикации результатов в Web или вывода в файл, Flash обладает многими преимуществами, такими, как:

- Применение *векторной (vector)* графики, которая в действительности представляет собой ряд математических формул, описывающих размер, цвет и местоположение формы. Векторная графика состоит из линий и кривых, тогда как растровая *(bitmapped)* — из небольших *точек растра,* или *пикселей (pixels)*. Применение математических формул является главным преимуществом отображения графики и сохранения небольшого размера файлов, что особенно важно для Web.
- Возможность сжатия анимации, графики и звука. Такое сжатие файла обусловливает свойство *потоковой передачи (streaming),* которое позволяет отображать сайт в Web-броузере до полной загрузки его содержимого. Это означает, что одна векторная графика сайта отображается, в то время как другая графика, звук и анимация все еще загружаются.
- Интерактивность во Flash может состоять из самых разных свойств. Помимо неограниченных возможностей применения простой графики, анимации или крупных заголовков, в любой объект или область сайта можно вводить полноценные навигационные свойства: кнопки, меню или фрагменты анимации, с помощью которых конечный пользователь перемещается по сайту. А задавая действия *(actions),* можно создавать интерактивные фильмы. Для создания интерактивных элементов управления во Flash предоставляется **язык создания сценариев - ActionScript.**
- Изображения и тест, разработанные во Flash, всегда отображаются ясно и четко, что опять же обусловлено векторной графикой.
- Flash помогает пользователю рисовать. В частности, Flash может распознавать основные геометрические формы в процессе их создания. При этом она заменяет небрежно нарисованную форму правильной геометрической формой, которую, как предполагается, пользователь пытается нарисовать. Например, если быстро нарисовать овал или окружность, создав круглую форму, Flash сделает эту форму более плавной и круглой, доведя ее до идеального овала или окружности. Кроме того, Flash помогает создавать прямые и плавные линии.

В состав Flash входит инструмент, который создает основу для большинства качественных Flash-фильмов. Этот инструмент – **ActionScript** - событийно-управляемый язык, встроенный во Flash. ActionScript делает ваши страницы интерактивными. Вы можете реагировать на события с мышки или с клавиатуры, можете выполнить какие-либо действия при проигрывании определенного кадра.

Для того чтобы овладеть ActionScript в полной мере, желательно уже иметь опыт программирования (предпочтительно на C++, JavaScript, etc.). Однако одним из достоинств языка Flash является то, что вам не нужно быть профессионалом во Flash, или полностью знать ActionScript, чтобы писать на нем качественный код. Вы можете использовать лишь те возможности языка, которые сочтете необходимыми для своей работы.

2. Языки программирования клиент-машин

2.1. JavaScript

Web, как гипертекстовую систему, можно рассматривать с двух точек зрения. Во-первых, как совокупность отображаемых страниц, связанных гипертекстовыми переходами. Во-вторых, как множество элементарных информационных объектов, составляющих отображаемые страницы (текст, графика, мобильный код и т.п.). В последнем случае множество гипертекстовых переходов страницы - это такой же информационный фрагмент, как и встроенная в текст картинка.

При втором подходе гипертекстовая сеть определяется на множестве элементарных информационных объектов самими HTML-страницами, которые и выступают в роли гипертекстовых связей. Этот подход более продуктивен с точки зрения построения отображаемых страниц "на лету" из готовых компонентов.

При генерации страниц в Web возникает дилемма, связанная с архитектурой "клиент-сервер". Страницы можно генерировать как на стороне клиента, так и на стороне сервера. Последнее реализуется через механизм подстановок на стороне сервера (Server Site Includes). Компания Netscape распространила в 1995 году механизм управления страницами и на клиента, разработав язык программирования JavaScript.

Таким образом, **JavaScript** - это язык управления сценариями просмотра гипертекстовых страниц Web на стороне клиента. Если быть более точным, то JavaScript - это не только язык программирования на стороне клиента. Liveware, прародитель JavaScript, является средством подстановок на стороне сервера Netscape. Однако, наибольшую популярность JavaScript обеспечило программирование на стороне клиента.

К возможностям JavaScript можно, например, отнести следующее:

- отображать изменяющиеся данные, такие как текущее время или дата;
- программировать переменное содержание в зависимости от даты, броузера пользователя или других условий;
- изменять внешний вид элементов страницы, если пользователь щелкнул мышью или провел курсор мыши над элементом.

Для языка высокого уровня JavaScript обладает довольно сильными возможностями. Он не позволяет работать на уровне машинных кодов, однако вы получаете доступ ко многим возможностям броузеров, Web-страниц, а иногда и системы, в которой работает броузер. В отличие от Java™ или C, программы на JavaScript обходятся без компиляции, а вашему броузеру не придется загружать виртуальную машину для выполнения программного кода. Программируй и загружай!

JavaScript также работает в объектно-ориентированной архитектуре, напоминающей Java или C++. Такие возможности языка, как конструкторы или наследование на базе прототипов, добавляют в схему разработки новый уровень абстракции, что способствует многократному использованию программного кода.

Одна из главных причин, по которой Web-разработчики приняли JavaScript, — возможность выполнения на стороне клиента многих функций, которые ранее выполнялись исключительно на стороне сервера. Лучшим примером является проверка форм. Программисты старой школы еще помнят, что несколько лет назад для проверки пользовательского ввода в формах HTML приходилось пересылать информацию на Web-сервер и передавать ее сценарию CGI, где и проходила проверка введенных данных.

Если данные не содержали ошибок, сценарий CGI продолжал работу. Однако при обнаружении ошибок сценарий возвращал пользователю сообщение с описанием проблемы. Хотя это решение

работает, представьте, сколько лишней работы при этом происходит. Для передачи формы необходим специальный запрос HTTP от сервера. После пересылки данных в Сети приходится заново выполнять сценарий CGI. Этот процесс повторяется каждый раз, когда пользователь допускает ошибку при заполнении формы. Пользователь узнает об ошибке лишь после того, как сообщение об ошибке вернется к нему.

Но вот на сцене появляется JavaScript. Теперь элементы формы можно проверить *до того,* как пользователь передаст информацию Web-серверу. Это приводит к уменьшению количества транзакций HTTP, а также заметному снижению вероятности ошибки при повторном заполнении формы. Кроме того, JavaScript позволяет читать и записывать cookie — когда-то эта операция выполнялась исключительно средствами Web-сервера для работы с заголовками.

Коротко о некоторых особенностях JavaScript

Тэг <SCRIPT> сообщает броузеру, что внутри HTML размещен код JavaScript. Тэг </SCRIPT> отменяет действие. <!— и —> тэги сообщают броузерам, которые не могут интерпретировать <SCRIPT> и </SCRIPT>, что строки кода следует рассматривать как комментарии. Двойная косая черта (//) перед тэгом --> - знак комментария в языке JavaScript; без него JavaScript интерпретирует --> как ошибочный оператор.

Если вы забудете точку с запятой, JavaScript сам подставит ее в конце строки, но проблем будет меньше, если вы сами проследите за пунктуацией.

При строгой типизации данных каждая переменная имеет один и только один тип данных, который не может быть изменен. Объявленное целым останется целым всегда, объявленное строкой останется строкой всегда. При *слабой* типизации данных, как в JavaScript, вы можете поместить любое значение в любую переменную, и переменная примет требуемый тип данных.

Document.write - метод, который прописывает HTML в Web-страницу как при программировании вручную.

Каждый JavaScript-оператор должен заканчиваться точкой с запятой. Отсутствие точки с запятой JavaScript считает ошибкой. Логические выражения должны быть заключены в круглые скобки, а блоки операторов — в фигурные скобки.

JavaScript использует знак == как логический оператор эквивалентности и знак = в качестве оператора присваивания. Попытка сравнения с использованием знака равенства (=) - вторая причина ошибок в JavaScript.

2.2. VBScript

В ответ на появление JavaScript Microsoft выпустила версию своего популярного языка программирования Visual Basic. В целом VBScript делает то же самое, что JavaScript, только программный код очень похож на Visual Basic.

Visual Basic Script - подмножество языка Visual Basic. VBScript позволяет решать задачи, связанные с Internet, а именно создавать сценарии (или скрипты) управления объектами (кнопками, списками, ниспадающими меню и т. д.) на Web-страничках. С помощью VBScript можно быстро создавать собственные страницы или даже писать игры. И все это размещается внутри HTML-документа.

Коротко о некоторых особенностях VBScript

VBScript отличается от JavaScript следующим:

- в конце оператора не ставится точка с запятой;
- знак равенства используется как для присвоения значений переменным, так и для выполнения операции сравнения;

• блоки операторов выделяются не фигурными скобками, а парами ключевых слов *if...endif, do...loop, u while...wend.*

Так же как в JavaScript, переменные вводятся по мере необходимости, а их тип определяется по контексту. VBScript поддерживает почти все встроенные функции Visual Basic. На Web-страницах VBScript выглядит следующим образом:

```
<script language="VBScript"> <!—
document.write("URL=" + document.location) —>
</script>
```

Тэги <SCRIPT> и </SCRIPT> практически такие же, как и раньше, а язык определяется как VBScript. Тэги <!— и —> - те же самые, но без знака комментариев //.

Основной недостаток VBScript состоит в том, что его поддерживает только Internet Explorer. Это уничтожает всякую надежду использовать одну и ту же страницу как для пользователей Netscape Navigator, так и для пользователей Internet Explorer. По этой причине использование VBScript для программирования на стороне клиента ограничено.

2.3. Java

Создание языка **Java** — один из самых значительных шагов вперед в области разработки сред программирования за последние 20 лет. Язык HTML был необходим для статического размещения страниц во “Всемирной паутине” WWW (World Wide Web). Язык Java потребовался для качественного скачка в создании интерактивных продуктов для сети Internet.

Три ключевых элемента объединились в технологии языка Java и сделали ее в корне отличной от всего, существующего на сегодняшний день.

- Java предоставляет для широкого использования свои **апплеты (applets)** — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения, встраиваемые в страницы Web. Апплеты Java могут настраиваться и распространяться потребителям с такой же легкостью, как любые документы HTML.
- Java высвобождает мощь объектно-ориентированной разработки приложений, сочетая простой и знакомый синтаксис с надежной и удобной в работе средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты.
- Java предоставляет программисту богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

История создания

Язык Java зародился как часть проекта создания передового программного обеспечения (ПО) для различных бытовых приборов. Реализация проекта была начата на языке C++, но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента - языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Апплеты Java

Каждый **апплет** — это небольшая программа, динамически загружаемая по сети — точно так же, как картинка, звуковой файл или элемент мультипликации. Главная особенность апплетов заключается в том, что они являются настоящими программами, а не очередным форматом файлов для хранения мультфильмов или какой-либо другой информации. Апплет не просто проигрывает один и тот же сценарий, а реагирует на действия пользователя и может динамически менять свое поведение.

Именно броские Web-страницы с анимацией привлекли большинство ранних приверженцев языка Java. Поскольку пользователи не сразу смогли полностью освоить наиболее революционные аспекты Java, этот язык часто сравнивался с другими технологиями для загрузки динамических изображений и простого взаимодействия с Web-клиентами.

Революционный язык программирования

Разработчики Java с самого начала хорошо понимали, что язык, предназначенный для решения проблем гетерогенных сред, также должен быть

- простым - его должны с легкостью использовать все разработчики
- ясным - разработчики должны без больших усилий выучить Java
- объектно-ориентированным - он использует все преимущества современных методологий разработки ПО и подходит для написания распределенных клиент-серверных приложений
- многопоточным - для обеспечения высокой производительности приложений, выполняющих одновременно много действий (например, в мультимедийных системах)
- интерпретируемым - для переносимости и большей динамичности

Язык должен был воплощать следующие качества: простоту и мощь, безопасность, объектную ориентированность, надежность, интерактивность, архитектурную независимость, возможность интерпретации, высокую производительность и легкость в изучении. Даже если вы никогда не напишете ни одной строки на языке Java, знать о его возможностях весьма полезно, поскольку именно перечисленные выше свойства языка придают динамику страницам Всемирной паутины.

- **Простота и мощь**

После освоения основных понятий объектно-ориентированного программирования вы быстро научитесь программировать на Java. В наши дни существует много систем программирования, гордящихся тем, что в них одной и той же цели можно достичь десятком различных способов. В языке Java изобилие решений отсутствует — для решения задачи у вас будет совсем немного вариантов. Стремление к простоте зачастую приводило к созданию неэффективных и невыразительных языков типа командных интерпретаторов. Java к числу таких языков не относится – для Вас вся мощность ООП и библиотек классов.

- **Безопасность**

В популярной литературе наших дней, особенно если речь заходит об Internet, стало модной темой обсуждение вопросов безопасности. Люди уверены, что использование Internet в коммерческой деятельности равносильно написанию номера своей кредитной карточки на стенке телефонной будки. Один из ключевых принципов разработки языка Java заключался в обеспечении защиты от несанкционированного доступа. Программы на Java не могут вызывать глобальные функции и получать доступ к произвольным

системным ресурсам, что обеспечивает в Java уровень безопасности, недоступный для других языков.

- **Объектная ориентированность**

Забавно наблюдать, как многочисленные новые диалекты старых языков безапелляционно объявляются объектно-ориентированными. Поскольку при разработке языка отсутствовала тяжелая наследственность, для реализации объектов был избран удобный прагматичный подход. Разработчики Java старались выдержать разумный компромисс между моделью пуристов — "все является объектами", и моделью хакеров — "уйди с моей дороги". Объектная модель в Java проста и легко расширяется, в то же время, ради повышения производительности, числа и другие простые типы данных Java не являются объектами.

- **Надежность**

Java ограничивает вас в нескольких ключевых областях и таким образом способствует обнаружению ошибок на ранних стадиях разработки программы. В то же время в ней отсутствуют многие источники ошибок, свойственных другим языкам программирования (строгая типизация, например). Большинство используемых сегодня программ "отказывают" в одной из двух ситуаций: при выделении памяти, либо при возникновении исключительных ситуаций. Java фактически снимает обе эти проблемы, используя сборщик мусора для освобождения незанятой памяти и встроенные объектно-ориентированные средства для обработки исключительных ситуаций.

- **Интерактивность**

Java создавалась как средство, которое должно удовлетворить насущную потребность в создании интерактивных сетевых программ. В Java реализовано несколько интересных решений, позволяющих писать код, который выполняет одновременно массу различных функций и не забывает при этом следить за тем, что и когда должно произойти. В языке Java для решения проблемы синхронизации процессов применен наиболее элегантный из всех когда-либо изобретенных методов, который позволяет конструировать прекрасные интерактивные системы.

- **Независимость от архитектуры ЭВМ**

Вопрос о долговечности и переносимости кода важнее религиозных войн между ПК и Макинтошами. Создатели Java наложили на язык и на среду времени выполнения несколько жестких требований, которые на деле, а не на словах позволяют, однажды написав, всегда запускать программу в любом месте и в любое время (где существует виртуальная Java-машина – броузеры на всех платформах, OS/2, Netware).

- **Интерпретация плюс высокая производительность**

Необычайная способность Java исполнять свой код на любой из поддерживаемых платформ достигается тем, что ее программы транслируются в некое промежуточное представление, называемое байт-кодом (bytecode). Байт-код, в свою очередь, может интерпретироваться в любой системе, в которой есть среда времени выполнения Java. Большинство ранних систем, в которых пытались обеспечить независимость от платформы, обладало огромным недостатком — потерей производительности (Basic, Perl). Несмотря на то, что в Java используется интерпретатор, байт-код легко переводится непосредственно в "родные" машинные коды (Just In Time compilers) "на лету". При этом достигается очень высокая производительность (Symantec JIT встроен в Netscape Navigator).

- **Простота изучения**

Язык Java, хотя и более сложный чем языки командных интерпретаторов, все же неизмеримо проще для изучения, чем другие другие языки программирования, например C++. Черты языка станут казаться вам естественным путем для решения тех или иных задач и будут способствовать отработке хорошего стиля программирования. Поскольку объектная модель в Java одновременно проста и выразительна, вы скоро освоитесь с объектно-ориентированным стилем создания программ.

- **Богатая объектная среда**

Среда Java — это нечто гораздо большее, чем просто язык программирования. В нее встроен набор ключевых классов, содержащих основные абстракции реального мира, с которым придется иметь дело вашим программам. Основой популярности Java являются встроенные классы-абстракции, сделавшие его языком, действительно независимым от платформы. Библиотеки, подобные MFC/COM, OWL, VCL, NeXTStep, Motif и OpenDoc прекрасно работают на своих платформах, однако сегодня главной платформой становится Internet.

2.4. VRML

Язык VRML (Virtual Realty Modelling Languagy) предназначен для описания трехмерных изображений и оперирует объектами, описывающими геометрические фигуры и их расположение в пространстве.

Vrml-файл представляет собой обычный текстовый файл, интерпретируемый браузером. Поскольку большинство браузеров не имеет встроенных средств поддержки vrml, для просмотра Vrml-документов необходимо подключить вспомогательную программу - Vrml-браузер, например, Live3D или Cosmo Player.

Как и в случае с HTML, один и тот же vrml-документ может выглядеть по-разному в разных VRML-браузерах. Кроме того, многие разработчики VRML-браузеров добавляют нестандартные расширения VRML в свой браузер.

Существует немало VRML-редакторов, делающих удобней и быстрее процесс создания Vrml-документов, однако несложные модели, рассматриваемые в данной статье, можно создать при помощи самого простого текстового редактора.

3. Языки программирования серверов

С помощью сценариев для сервера можно получить доступ к файлам, базам данных и другим ресурсам, хранимым на сервере, а также к централизованным ресурсам сервера, таким как электронная почта или факс-служба.

Функционирование в непротиворечивой и управляемой среде - еще одно преимущество выполнения сценариев на сервере. Ваш код выполняется только на одной версии единственного сервера, а не на множестве версий множества броузеров.

Однако и для использования сценариев на стороне сервера имеется три основных препятствия.

- Запуск скриптов на сервере зачастую требует получения специальных прав от Web-мастера или системного администратора.
- Для взаимодействия со сценариями, выполняющимися на сервере, пользователь должен щелкнуть мышкой на ссылке или на кнопке на странице, а затем ожидать, когда сервер выполнит сценарий и перешлет ответ. Взаимодействие с использованием динамического HTML происходит быстрее.
- Для тестирования сценариев для сервера требуется иметь собственный WWW-сервер, предпочтительно того же типа, что и промышленный вариант.

Программирование на стороне сервера в настоящее время является необходимым условием для решения широкого спектра задач. Оно позволяет:

a) получать и обрабатывать на сервере данные, введенные пользователем при помощи формы;

b) динамически создавать web-документы, не зависящие ни от платформы, ни от браузера клиента;

c) обеспечивать динамический доступ к данным, находящимся на сервере, в частности, к серверным базам данных (при таком способе доступа HTML-документ автоматически изменится, как только изменятся хранящиеся на сервере данные);

d) использовать серверные компоненты, предназначенные для решения типовых задач (таких, например, как циклическая смена рекламных баннеров и др.);

e) осуществлять аутентификацию пользователя;

f) получать информацию о браузере клиента;

g) создавать и читать ключики на стороне клиента;

3.1. CGI: Технология «клиент-сервер»

Большое количество World Wide Web приложений основано на использовании внешних программ, управляемых Web сервером. Использование данных программ позволяет строить Web приложения с динамически обновляемой информацией, хранящейся в базах данных или генерирующейся в зависимости от бизнес-правил решаемых задач. Для связи между Web сервером и вызываемыми программами широко используется Common Gateway Interface (CGI), имеющий реализации как для Windows-ориентированных программ, так и для приложений, функционирующих в среде Unix.

CGI - Common Gateway Interface является стандартом интерфейса (связи) внешней прикладной программы с информационным сервером типа HTTP, Web сервер. 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.

Т.о., 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 обычно записывается в поддиректорий с именем cgi-bin WWW сервера.

Интернет вообще и WWW в частности работает по технологии «клиент-сервер», то есть все программное обеспечение разделяется на клиентскую и на серверную части. Также между ними разделены и функциональные обязанности. Важным для понимания моментом является то, что клиент не знает и не обязан знать принципы работы и реализацию внутренних алгоритмов сервера, а сервер не вмешивается в дела клиента. Для взаимодействия этих частей разработан специальный *протокол* (в частном случае — протокол HTTP), и все взаимодействие между клиентом и сервером осуществляется исключительно в рамках данного протокола. Вашему броузеру все равно, какое программное обеспечение стоит на сервере, какая там операционная система, где физически лежат запрашиваемые документы на сервере (и лежат ли вообще, ведь они могут и генерироваться на лету специальными программами). Сервер тоже не вмешивается в дела вашего броузера, серверу абсолютно все равно, что сделает клиент с переданной информацией, как он ее будет отображать, сохранит на диске или проигнорирует — серверу до этого дела нет. Взаимодействие клиента и сервера происходит по принципу «запрос-ответ». Клиент посылает запрос, сервер обрабатывает его и посылает ответ (рис. 1.1-1):

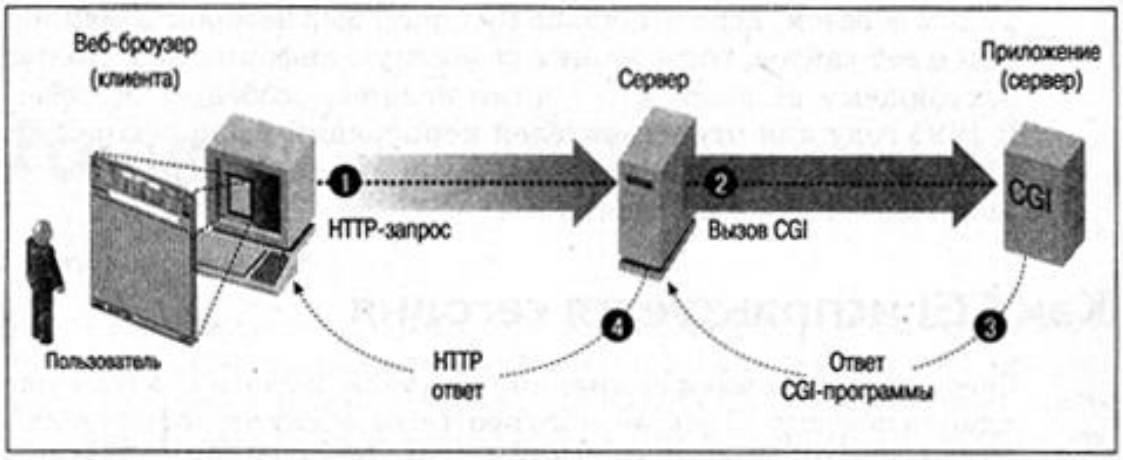


Рис. 1-1. Как запускается CGI-приложение

Рассмотрим более детально все этапы соединения по протоколу HTTP:

1. Формирование запроса клиентом. (Броузер формирует запрос из URL, набранного пользователем, из щелчка на ссылке либо из данных формы.)
2. Установка соединения с сервером. (Если установить соединение не удается, то на этом HTTP-транзакция закончится и клиент выдаст пользователю сообщение об ошибке.)
3. Посылка запроса и ожидание ответа от сервера. (Все, что требуется от клиента, это чтобы запрос был в корректном формате.)
4. Сервер принимает запрос. (Об этом и следующем этапе клиенту ничего не известно.)
5. Сервер обрабатывает запрос.
6. Генерация ответа.
7. Прием ответа клиентом.
8. Разрыв соединения.

9. Обработка данных клиентом. (Вывод или сохранение данных.)

Обычно под запросом к серверу понимается URL (это унифицированная форма «заказа» данных на сервере). К собственно URL могут еще «прилагаться» некоторые данные, чаще всего это данные форм (вспомните, как вы вводите ключевое слово в поисковике). Формирование HTTP-запроса будет детально рассмотрено в одном из следующих уроков.

При установке соединения с сервером сначала происходит трансляция символьного доменного имени, такого, как www.siemens.com, в IP-адрес, а затем осуществляется непосредственно создание TCP/IP-соединения с данным IP. Когда данные HTTP-запроса посланы серверу, клиент просто ожидает, пока не придет ответ.

Пока нет обращений от клиентов, сам HTTP-сервер просто «спит» в ожидании запросов. Когда клиент устанавливает соединение, сервер «просыпается» и, приняв данные запроса, приступает к их обработке. Что именно сервер делает с запросом — известно только самому серверу. Единственный результат всех хитрых манипуляций — это выдача ответа, которого и ожидает клиент.

После того как сервер выдал ответ, он разрывает соединение и вновь «погружается в сон». Естественно отметить, что в случае возникновения ошибки HTTP-транзакция может закончиться на любом из этих этапов.

Все девять этапов HTTP-соединения показаны на **рис. 1.2.**

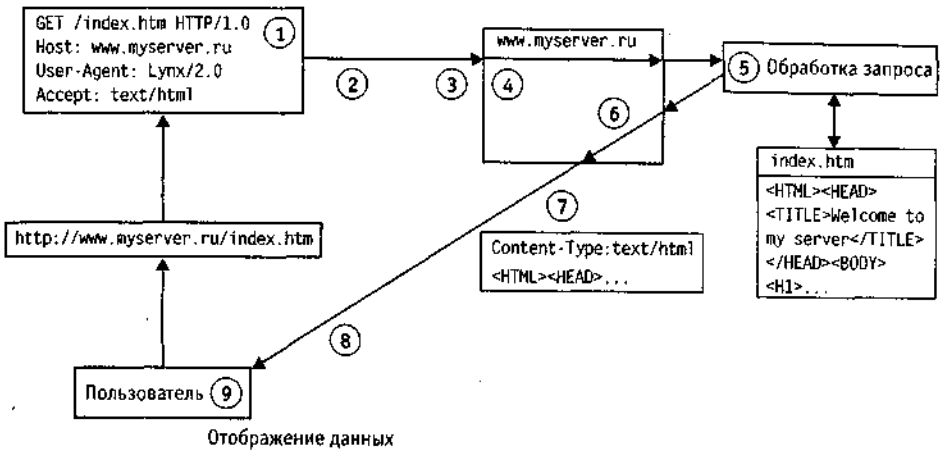


Рис. 1.2. Этапы HTTP-соединения

Если документ не найден или если для доступа к нему у вас нет прав (доступ к ресурсу может быть ограничен), то выдается код ошибки. А если все нормально, то информация, содержащаяся в документе, включая сопутствующие данные о его типе, выдается в виде ответа.

Схема взаимодействия «клиент-сервер» для случая, когда URL указывает на CGI-обработчик, показана на рис. 1.3.

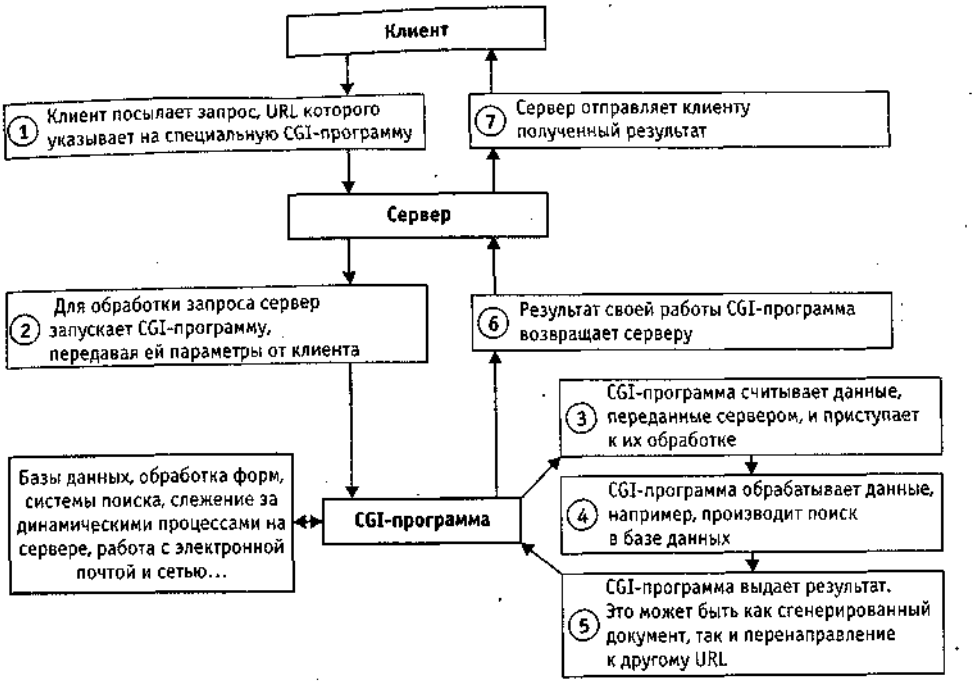


Рис. 1.3. Взаимодействие «клиент-сервер» при вызове CGI-обработчика

3.2. Технология SSI

SSI (Server Side Includes, включения на стороне сервера) - это директивы, вставляемые в HTML-код и служащие для передачи указаний серверу.

SSI позволяют "вставлять" фрагменты одних документов в другие. Конечно, это можно сделать непосредственно в текстовом редакторе, но если, например, в несколько документов вставляется один и тот же фрагмент, к тому же часто изменяемый, использовать SSI-вставки много удобнее.

Сервер интерпретирует SSI-директивы и выполняет соответствующие действия. Использование SSI-вставок позволяет динамически формировать странички в зависимости от различных параметров(например, типа браузера).

Преимущества SSI проявляются тем сильнее, чем больше по объему сайт, имеющий повторяющиеся элементы кода на разных страничках.
Для того, чтобы сервер знал, что страничка не обычная, а содержит SSI-директивы, используется специальное расширение: shtml или shtm. (Вообще-то, конфигурация сервера может быть настроена и на другое расширение, но shtml воспринимается всегда (если только на сервере не отключено применение SSI вообще).

Для того, чтобы указать серверу, какой блок нужно вставить и в каком месте странички, используется специальная форма записи в виде комментария:

```
<!--#команда параметр="значение" -->
```

При просмотре сформированного исходника HTML-файла пользователь не увидит никаких признаков SSI, т.к. браузер получает уже готовый HTML-код.

Первое преимущество SSI с точки зрения дизайнера заключается в том, что при таком подходе web-мастеру, занимающимуся поддержкой сайта, можно не бояться случайно испортить дизайн. Элементы сложной верстки скрыты за счет использования SSI, и поддержка содержимого страничек становится гораздо более легким и приятным делом. Второе, не менее важное преимущество, - это возможность мгновенной замены дизайна сайта, не требующая переделывания страничек. Для смены дизайна достаточно переписать SSI-вставки, формирующие внешний вид сайта.

Не является ли SSI альтернативой CGI? Скорее, это дополнение (и очень ценное) к CGI, предоставляющее web-программисту множество удобств.

3.3. ISAPI

Подобно CGI, спецификация **ISAPI (Internet Server Application Programming Interface)** определяет правила взаимодействия между Web-сервером и дополнительными программами. Для того чтобы понять, зачем понадобился альтернативный подход к созданию программ, выполняющихся на сервере, поговорим о преимуществах и недостатках стандарта CGI.

Несомненным преимуществом CGI является универсальность. CGI-сценарии могут быть написаны на разных языках и выполняться на компьютерах с различной архитектурой. Если при написании сценария вы учли все правила, то можете быть уверены, что созданная вами программа будет корректно взаимодействовать с любым Web-сервером. Простота CGI также способствует широкому распространению этого стандарта.

Однако, наряду с преимуществами, CGI-сценарии обладают некоторыми недостатками; главным из которых считается неэффективное использование ресурсов. Каждый из запросов клиента, предполагающий вызов сценария, порождает отдельный процесс на компьютере-сервере. Как известно, для выполнения независимого процесса требуется гораздо больше ресурсов, чем для работы потока в составе процесса.

Стремление повысить производительность и снизить затраты ресурсов привело к созданию корпоративных стандартов. Наиболее известными из них являются NSAPI и ISAPI.

Основное отличие ISAPI-программы от CGI-сценария состоит в том, что ISAPI-программа представляет собой не исполняемый файл, а динамическую библиотеку (DLL). Благодаря этому появилась возможность запускать программу не как отдельный процесс, а как поток, принадлежащий Web-серверу. Для выполнения потока требуется значительно меньше ресурсов, чем для работы независимого процесса. Поток использует адресное пространство породившего его процесса и работает намного быстрее, чем отдельный процесс.

По окончании выполнения процесс выгружается из памяти, а при поступлении следующего запроса снова загружается с диска. Это также не способствует эффективной работе CGI. При интенсивном поступлении запросов необходимость постоянного обмена с диском создает дополнительную нагрузку на сервер. В отличие от CGI-сценария, ISAPI-программа, окончив свою работу, как правило, остается резидентной в памяти и используется для обработки последующих запросов.

Однако иногда преимущества ISAPI-программ оборачиваются недостатками. (Это не удивительно. Ведь если бы ISAPI-программы были свободны от недостатков, они бы быстро вытеснили CGI-сценарии.)

Если за универсальность CGI-сценариев пришлось платить недостаточной производительностью, то в качестве платы за высокую производительность создатели ISAPI-программ жертвуют универсальностью. Арсенал языков, имеющихся в распоряжении разработчиков ISAPI-программ, значительно беднее, чем у их коллег, создающих CGI-

сценарии. По сути, тому, кто хочет быстро написать достаточно большую ISAPI-программу, не остается иного выбора, кроме C++.

Второй недостаток ISAPI гораздо серьезнее первого. Поскольку программа данного типа выполняется как поток, порожденный сервером, она использует адресное пространство сервера. Следовательно, ошибка ISAPI-программы может не только вызвать ее аварийное завершение, но и вывести из строя сервер.

Программы, использующие спецификацию ISAPI, делятся на две категории.

* *Расширения.* Функционально они мало чем отличаются от CGI-сценариев. Подобно CGI-сценариям, расширения реализуют дополнительные возможности Web-сервера, но строятся совершенно по-другому. Рассмотрению ISAPI-расширений посвящена данная глава.

* *Фильтры.* Это особый класс программ. Если расширения *дополняют* возможности Web-сервера, то фильтры *изменяют* его поведение. Принцип работы и реализация ISAPI-фильтров будут рассмотрены в следующей главе.

ISAPI-расширение представляет собой динамическую библиотеку (DLL), которая связывается с Web-сервером в процессе его работы. В данной библиотеке должны *экспортироваться,* т.е. быть доступными для вызова, две функции: GetExtensionVersion() и HttpExtensionProcf).

Особенности выполнения ISAPI-расширений, а также основные отличия программ данного типа от CGI-сценариев проще всего выяснить, рассматривая процедуру вызова расширения, а также назначение указанных выше функций и параметров, передаваемых при их вызове.

В отличие от расширения, которое вызывается лишь тогда, когда в запросе клиента явно указан URL программы, ISAPI-фильтр получает управление при каждом обращении клиента к серверу. Программа-фильтр загружается в память при запуске Web-сервера и остается резидентной в течение всего времени работы сервера.

3.4. Язык программирования Perl

Perl - интерпретируемый язык, приспособленный для обработки произвольных текстовых файлов, извлечения из них необходимой информациии и выдачи сообщений. Он также удобен для написания различных системных программ. Этот язык прост в использовании, эффективен, но про него трудно сказать, что он элегантен и компактен.

Perl был создан в 1986 году как инструмент для администрирования и конфигурирования системных ресурсов сети, состоящей из Unix-компьютеров. Он сочетает в себе лучшие черты C, shell, sed и awk, поэтому для тех, кто знаком с ними, изучение Perl-а не представляет особого труда. Синтаксис выражений Perl-а близок к синтаксису C. В отличие от большинства утилит ОС UNIX Perl не ставит ограничений на объем обрабатываемых данных и если хватает ресурсов, то весь файл обрабатывается как одна строка. Рекурсия может быть произвольной глубины. Хотя Perl приспособлен для сканирования текстовых файлов, он может обрабатывать так же двоичные данные и создавать .dbm файлы, подобные ассоциативным массивам. Perl позволяет использовать регулярные выражения, создавать объекты, вставлять в программу на C или C++ куски кода на Perl-е, а также позволяет осуществлять доступ к базам данных.

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.

Хотя CGI-приложения можно писать практически на любом языке, Perl и CGI-программирование стали синонимами для многих программистов. Как сказал Хасан Шрейдер (Hassan Shroeder), первый вебмастер Sun, «Perl - это артерия Интернета». Perl - самый широко используемый язык для CGI-программирования, и для этого есть много веских причин:

- Perl легко выучить: его синтаксис напоминает другие языки (например C), потому что он «многое прощает», - при ошибке выдается подробное сообщение, помогающее быстро локализовать проблему.
- Perl способствует быстрой разработке, так как это интерпретируемый язык; исходный код не надо компилировать перед запуском.
- Perl доступен на многих платформах с минимальными изменениями.
- Perl содержит очень мощные функции для обработки строк со встроенной в язык поддержкой поиска и замены по регулярным выражениям.
- Perl обрабатывает двоичные данные так же легко, как и текст.
- Perl не требует четкого разделения на типы: числа, строки и логические выражения являются обычными скалярами.
- Perl взаимодействует с внешними приложениями очень просто и обеспечивает собственные функции для работы с файловыми системами.
- Для Perl есть много свободно доступных модулей от CPAN, начиная с модулей для создания динамической графики до интерфейсов с Интернет-серверами и системами управления базами данных. За подробной информацией по CPAN обратитесь к приложению B.

Perl действительно очень быстрый: считывая исходный файл, он тут же компилирует его в низкоуровневый код, который потом исполняет. Обычно компиляция и исполнение в Perl не воспринимаются как отдельные шаги, поскольку выполняются вместе: Perl запускается, читает исходный файл, компилирует его, запускает и затем завершает работу. Этот процесс повторяется каждый раз, когда запускается сценарий Perl, в том числе CGI-сценарии. Поскольку Perl так эффективен, этот процесс происходит достаточно быстро, чтобы обрабатывать все запросы не на самых загруженных серверах. Однако следует обратить внимание, что в системах Windows это гораздо менее эффективно из-за необходимости создания новых процессов.

3.5. PHP

PHP изобретен Расмусом Лердорфом в конце 1994 года. Первая версия выпущена в 1995 году под именем «Инструментарий Персональных Домашних Страниц», затем она была переработана и названа PHP/FI Version 2 (FI — модуль обработки данных для форм). Также была добавлена поддержка баз данных mSQL. С этого момента в разработке стали принимать участие добровольцы.

Статистика используемости PHP приблизительна, но, согласно исследованию, проведенному Netcraft, в начале 2001 года PHP использовался на более чем 5 300 000 сайтах по всему миру. Для сравнения: в это время число IIS серверов было примерно таким же (5 млн). Разработка интерпретатора PHP приняла форму организованного командного процесса, ядро интерпретатора разрабатывает компания Zend.com. При этом PHP распространяется свободно:

его последнюю версию можно загрузить с сайта PHP.net. Модули PHP поставляются в комплекте с сервером Apache, в комплектах систем Linux.

Изначально аббревиатура **PHP** означала ***Preprocessor of Home Pages* — препроцессор домашних страниц**. Это язык внедряемых в HTML-страницы сценариев, исполняемых на сервере. По большей части его синтаксис заимствован из таких языков, как C, Perl, Java, и при этом добавлена масса возможностей, которых этим языкам недостает. Проще говоря, синтаксис PHP — это разумная альтернатива и строгости C, и «беспредельности» Perl.

PHP наделен практически полным набором функциональности, о которой (до появления PHP) мог только мечтать web-программист. Его *цель* — позволить максимально быстро создавать динамически генерируемые web-страницы. С полным основанием можно заявить, что изучение и использование PHP будет выгодно как начинающим, так и профессиональным программистам.

Основными конкурентами PHP являются технологии JSP (J[ava] Server Pages и Java Scriptlets), ASP (Active Server Pages), Perl, SSI (Server Side Includes), Cold Fusion Server Pages.

Рассмотрим те недостатки, которые присущи указанным технологиям. **JSP** — достаточно сложный для изучения и использования язык. **ASP**, основанный на синтаксисе VBScript (Visual Basic), имеет всего несколько десятков собственных функций и поэтому вынужден использовать COM-объекты; кроме того, он ориентирован исключительно на работу под Windows. **Perl** — язык головоломный, и его вольности затрудняют его понимание. **SSI** позволяет всего лишь компоновать HTML-страницу из нескольких файлов. **CF** — коммерческий продукт, что является его основным недостатком.

Основные *достоинства* PHP:

- бесплатен; постоянно совершенствуется; работает на UNIX и Windows платформах;
- допускает работу с большинством СУБД;
- имеет широкий набор функций (более 3 тыс.);
- допускает объектно-ориентированное программирование;
- способен использовать протоколы HTTP, FTP, IMAP, SNMP, NNTP, POP3, net sockets и другие;
- позволяет выполнять все операции, что и перечисленные его конкуренты, и даже работать с файлами графики. Можно также запускать PHP-скрипты как интерпретируемые файлы и компилировать исполняемые приложения (в том числе с поддержкой графического интерфейса GTK).

Если вы обнаружите, что PHP не способен на что-то (или работает не так, как вам хотелось бы), никто не будет препятствовать вам вносить в исходный код PHP (написанный на C) желаемые изменения. PHP является программным продуктом с открытым исходным кодом, и внесение в него улучшений и дополнений путем создания собственных модулей расширения всегда приветствуется.

3.6. ASP и ASP.NET в составе Microsoft.NET

В конце 1997 г. Microsoft реализовала относительно простую среду периода выполнения пля Web — **Active Server Pages (ASP)** как часть сервера Internet Information Server (IIS), включенного в Windows NT 4 Option Pack. IIS обслуживает Web-страницы, запрашиваемые пользователем. ASP позволяет программистам реализовывать алгоритмы динамического создания страниц на IIS, состоящих из статического HTML и кода сценариев. Когда пользователь запрашивает ASP-страницу, IIS должен ее найти и активизировать ASP-процессор. ASP-процессор должен прочитать страницу и один к одному скопировать содержащиеся на ней HTML-элементы в выходную страницу. В нашем примере атрибут *style* устанавливает голубой цвет текста. При этом также интерпретируются элементы сценариев, расположенные между ограничителями <% %>.

Этот код , должен выполнять алгоритм, выдающий в качестве результата HTML-строки, которые ASP-процессор должен скопировать в выходную страницу в те места, где были элементы сценария. Результирующая страница, собранная из статических HTML-элементов и HTML, динамически сгенерированного сценарием, должна быть передана клиенту. Для простых задач ASP применять относительно легко, что является признаком качества этой технологии.

```
<html style="color:#OOOOFF;">
The time is: <% =time X> on <X =date X>
</html>
```

Технология Microsoft ASP представляет собой совокупность серверных средств для динамического создания Web-документов.

Активные серверные страницы (Active Server Pages, ASP), созданные Microsoft для собственного веб-сервера, сейчас доступны для многих серверов. Сервер ASP интегрирован в веб-сервер и не требует отдельного процесса. Он позволяет программистам совмещать код и HTML-страницы вместо того, чтобы писать отдельные программы. Для ASP существуют модули, позволяющие делать то же самое, используя CGI. ASP поддерживают различные языки программирования, самый популярный из которых Visual Basic, хотя JavaScript также поддерживается. Кроме того, существует версия Perl от ActiveState, которую можно использовать в Windows с ASP.

Microsoft .NET — готовая инфраструктура для решения общих проблем Интернет-приложений. Это прикомпоновываемая среда периода выполнения, работающая в ОС Windows 2000.

Сервис, обеспечиваемый .NET:

• **.NET Framework** — среда периода выполнения, облегчающая написание полноценного надежного кода в сжатые сроки, управление, развертывание и модификацию этого кода. Написанные вами программы и компоненты выполняются в этой среде. Она дает программистам в период выполнения такие классные возможности, как автоматическое управление памятью (сборка мусора) и упрощенный доступ ко всем службам ОС. Она добавляет массу вспомогательных функций вроде простого доступа к Интернету и базам данных. Кроме того, она обеспечивает новый механизм повторного применения кода — более простой в использовании и в то же время более мощный и гибкий, чем COM. Развертывать .NET Framework проще, так как она не требует настройки реестра. Она также поддерживает на системном уровне стандартизированный механизм управления версиями. Все это доступно программистам на любом .NET-совместимом языке. .NET Framework мы обсудим в главе 2.

• **ASP.NET (следующая версия Active Server Pages)** — это новая среда, работающая на Internet Information Server (US), заметно упрощающая написание кода для создания HTML-страниц. ASP.NET предлагает новый, не зависимый от языка способ создания кода и привязки его к запросам Web-страниц, — .NET Web Forms — управляемую событиями программную модель взаимодействия с элементами управления. Она делает программирование Web-страниц аналогичным программированию форм Visual Basic. ASP.NET содержит развитые средства управления сеансами и функции защиты. Она надежнее, и производительность ее значительно выше в сравнении с ASP. Microsoft .NET предлагает новый набор служб, позволяющих серверу предоставлять свои функции любому клиенту на любой машине с любой ОС.

По мере расширения Web и увеличения потребностей пользователей Web-разработчикам потребовались совершенствование двух ключевых свойств исполняющей среды: простоты

программирования и качества выполнения. ASP.NET и явилось таким усовершенствованием. ASP.NET похожа на оригинальную ASP и большая часть кода может быть переведена на нее практически без изменений. Но внутренняя реализация ASP.NET полностью переделана с тем, чтобы задействовать возможности .NET Framework.

ASP.NET отделяет HTML от алгоритмов, создавая *фоновый код* (code-behind). Вместо того чтобы перемешивать HTML с кодом, код пишется в отдельном файле, на который есть ссылка на ASP-странице. В результате такого разделения Microsoft смогла усовершенствовать среду разработки и отладки Visual Studio.NET, которая используется при разработке Web-приложений.

3.7. JAVA-servlets

Сервлеты - это высокопроизводительные платформо-независимые server-side-приложения, написанные на Java и составляющие реальную конкуренцию таким технологиям, как CGI, PHP3, Perl, и уж конечно ASP.

Java-сервлеты были созданы в Sun. Сервлеты похожи на CGI-сценарии тем, что это код, создающий документы. Тем не менее, сервлеты, поскольку они используют Java, должны быть скомпилированы перед запуском как классы, которые динамически загружаются веб-сервером при запуске сервлетов. Интерфейс отличается от CGI. JavaServer Pages или JSP - это другая технология, позволяющая разработчикам встраивать Java в веб-страницы, наподобие ASP.

К преимуществам сервлетов можно отнести:

a) Исключительно высокая скорость работы.

Быстродействие сервлетов объясняется тем, что они, во-первых, представляют собою уже скомпилированный и оптимизированный код (а в случае с JIT-ом - ещё и преобразованный в машинный) и, во-вторых, выполняются в единожды загруженной и инициализированной Java-машине.

Таким образом, экономятся ресурсы на запуск обработчика/парсера скрипта, необходимые, например, для Perl или PHP3 (в некоторых ОС, в частности, в OS/2 - это очень серьезная экономия), и ресурсы (как память, так и время), затрачиваемые на непосредственно предкомпиляцию (интерпретацию) кода (что необходимо для тех же Perl, PHP, REXX).

Реально обе этих проблемы сразу не решаются, практически, нигде. Наибольший эффект даёт, пожалуй, внедрение транслятора скриптового языка непосредственно в веб-сервер, например, пресловутые .asp-скрипты в серверах от Microsoft, или модули **mod_perl** или **mod_php** для apache. *(Последний вариант - PHP3, внедренный в апач - является, наверное, самым производительным из всего вышеперечисленного).*

b) Переносимость.

В данном случае принцип "write once run everywhere" действует безотказно. Сервлеты, написанные в соответствии со спецификацией от Sun и не использующие какие-то особенности конкретного веб-сервера, работают безо всякой переделки или перекомпиляции под любыми, порой весьма далёкими друг от друга платформами, будь то Solaris, FreeBSD или OS/2. В связи с этим разработчик может совершенно свободно выбирать, в какой системе ему удобнее работать - он ни коим образом не привязан ни к серверу, ни к будущей целевой платформе.

c) Работа с базами данных.

Работа с реляционными СУБД из Java унифицирована (для этого существует специальный пакет **java.sql**), удобна и отвязана от специфичных для конкретной СУБД тонкостей. Всё, что Вам нужно - это найти для своей СУБД JDBC-драйверы (а они сейчас существуют практически для всех современных баз данных, зачастую даже по нескольку разновидностей), и далее можно пользоваться совершенно стандартными механизмами.

А при переходе на другую СУБД, например, с MySQL на Oracle, достаточно будет просто добавить в CLASSPATH новый драйвер и поменять URL для подключения к другой базе. Ни одного изменения в коде

d) **Перспективность, современность технологий.**

Конечно, есть у этой технологии и недостатки. Как технические: например, высокие требования к системным ресурсам - в основном, к памяти (под OS/2, например, запущенная Java-машина занимает 15-20 мегабайт оперативной памяти) или необходимсть в качественной устойчивой реализации Java для выбранной платформы, так и иного плана: такие как отсутствие должной квалификации как у разработчиков, так и, зачастую, у тех, кто принимает решения, их устоявшиеся предубеждения и многое другое...

Технология работы сервлет-сервера.

Итак, как же работают сервлеты. Рассмотрим это на примере модуля JServ к веб-серверу apache.

В момент старта сервера вместе с ним стартует и ява-машина с так называемым servlet-wrapper'ом или средой, в которой в дальнейшем и предстоит исполняться сервлетам. Строго говоря, JServ - это и есть та самая среда. Он целиком написан на Java и занимается непосредственно загрузкой и исполнением сервлетов, следуя спецификации Sun, а также обменом данными с собственно веб-сервером. В последнем для этого должен присутствовать специальный модуль mod_jserv (его необходимо добавить при компиляции и сборке apache, или подключить в виде внешнего модуля).

При получении запроса на документ, приходящийся на специально оговоренный URL или каталог (обычно это что-нибудь вроде /servlets/), apache с помощью модуля mod_jserv передает этот запрос JServ'у, который определяет, какой сервлет должен этот запрос обработать, загружает этот сервлет (если он ещё не был загружен) и затем возвращает веб-серверу тот текст или поток данных, который был сформирован в результате работы сервлета.

Изначально сервер "пуст" - при его старте сервлеты обычно не загружаются (хотя есть возможность принудительно инициализировать нужные сервлеты при старте сервера). При появлении запроса нужный сервлет ищется в списке уже загруженных и, при необходимости, стартуется и инициализируется. После этого он остается постоянно загруженным в Java-машине (и предкомпилированным, если Java-машина содержит JIT) и при последующих запросах просто вызывается соответствующий его метод для их обработки. Преимущества такой идеологии очевидны. Функционально это аналогично вызову простой подпрограммы внутри обычного сервера и проходит очень быстро и эффективно. Кроме того, заметный выигрыш дают такие вещи, как единожды проведенная инициализация, возможность хранения глобальных данных или поддержка множественных клиентских сессий, ведущаяся самим **сервером** (а не сервлетами, разработчики которых в значительной степени избавлены от изобретания велосипедов). Например, можно установить одно единственное соединение с базой данных, и пользоваться им при обработке запросов - немалая экономия, учитывая то, что из тех же скриптов на perl или php приходится каждый раз создавать новое соединение, восстанавливать параметры сессии и т.п.

Конечно же, существует возможность принудительной **выгрузки** отдельных сервлетов из памяти в случае необходимости, а также возможность автоматического распознавания изменения сервлетов и их перезагрузки. Иными словами, при обновлении того или иного сервлета нет необходимости перезагружать весь веб-сервер или JServ, достаточно просто положить новую версию на место старой, и она будет автоматически загружена в память при следующем запросе (естественно, при этом будет сначала произведено корректное завершение работы старой версии, путём вызова специального метода, а затем загрузка и инициализация новой).

3.8. Пакет Cold Fusion от Macromedia

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 или информационному хранилищу.

ColdFusion от Macromedia в большей степени чем PHP различает страницы с кодом и HTML-страницы. В HTML-страницах могут быть дополнительные теги, вызывающие функции ColdFusion. В ColdFusion доступны несколько стандартных функций, и разработчики могут создавать собственные функции как расширения. ColdFusion был первоначально написан для Windows, но теперь доступны версии и для Unix. Интерпретатор ColdFusion встроен в веб-сервер.

4. Технология ActiveX

ActiveX - технология Microsoft, предназначенная для написания сетевых приложений. Она предоставляет программистам наборы стандартных библиотек, значительно облегчающих процесс кодирования. Если раньше при написании программ использовались механизмы OLE (OLE Automation, OLE Documents, OLE Controls,...), основанные на компонентной объектной модели (COM - Component Object Model), то теперь библиотеки OLE переписаны так, чтобы обеспечивать функциональность, достаточную для написания сетевых приложений. Таким образом, теперь при написании программ используется DCOM (Distributed Component Object Model) - распределенная компонентная объектная модель, а реализуют ее библиотеки ActiveX, которые по объему оказались гораздо меньше, чем библиотеки OLE, а по скорости - быстрее. Сохранилась и совместимость - любой программный компонент OLE будет работать с библиотеками ActiveX.

4.1. Понятие COM

Все технологии OLE и ActiveX, построены на основании, обеспеченном COM. Итак, что же такое COM? Чтобы ответить на этот вопрос, зададимся сначала другим: "Каким образом одна часть программного обеспечения должна получать доступ к сервисам, предоставляемым другой частью? " На сегодняшний день ответ зависит от того, что представляют собой эти части:

- Приложения, например, скомпонованные с библиотекой, могут пользоваться ее сервисами, вызывая функции из этой библиотеки.
- Приложение также может использовать сервисы другого — являющегося совершенно отдельным процессом. В этом случае два таких локальных процесса взаимодействуют посредством некоего механизма связи, который обычно требует определения протокола между этими приложениями (набор сообщений, позволяющий одному приложению выдавать запросы, а другому соответствующим образом отвечать на них).
- Еще пример — приложение, использующее сервисы операционной системы. Здесь приложение обычно выполняет системные вызовы, обрабатываемые операционной системой.
- Наконец, приложению могут понадобиться сервисы, предоставляемые программным обеспечением, выполняемым на другой машине, доступ к которой осуществляется по сети. Получить доступ к таким сервисам можно множеством способов, таких как обмен сообщениями с удаленным приложением или вызовы удаленных процедур.

В принципе проблема одна: одна часть программного обеспечения должен получить доступ к сервисам, предоставляемым другой частью. Но в каждом отдельном случае механизм доступа разный: вызовы локальных функций, передача сообщения средствами связи между процессами, системные вызовы (которые с точки зрения программиста выглядят практически так же, как и вызовы функций) или какая-то разновидность сетевых коммуникаций. Зачем все это? Не проще ли определить один общий способ доступа ко всем видам программных сервисов независимо от способа их реализации?

Этим и занимается **COM** - *она определяет стандартный механизм, с помощью которого одна часть программного обеспечения предоставляет свои сервисы другой и который работает во всех описанных выше случаях***.** Общая архитектура сервисов в библиотеках, приложениях, системном и сетевом программном обеспечении позволяет COM изменить подход к созданию программ.

В начале 1996 года Microsoft ввела в оборот новый термин — ActiveX. Поскольку самым динамично развивающимся направлением в компьютерной индустрии является Internet, именно здесь наиболее естественно могут найти свое место программы, написанные с использованием технологии ActiveX. Не случайно в последнее время понятия ActiveX и Internet часто встречаются рядом. В то же время технология ActiveX имеет значительно более универсальную область использования.

Стандарт ActiveX позволяет программным компонентам взаимодействовать друг с другом по сети независимо от языка программирования, на котором они написаны. С помощью ActiveX можно "оживить" страницы Web эффектами мультимедиа, интерактивными объектами или сложными приложениями. ActiveX обеспечивает некий "скрепляющий раствор", с помощью которого отдельные программные компоненты на разных компьютерах "склеиваются" в единую распределенную систему.

ActiveX включает в себя клиентскую и серверную части, а также библиотеки для разработчика:

- **программные элементы ActiveX** - компоненты, работающие на компьютере-клиенте, но загружаемые в первый раз с сервреа Web. С их помощью можно демонстрировать разнородную информацию, включающую видео и звук без запуска дополнительных программ. Более того, эти программные компоненты могут использоваться в приложениях, написанных на любых популярных языках программирования, включая Java (Visual J++), Visual Basic, Visual C++.
- **Active Scripting** поддерживает любой популярный макроязык, включая Visual Basic Script и JScript (реализация компанией Microsoft языка сценариев JavaScript). Макроязыки могут использоваться для объединения на одной странице нескольких программных элементов ActiveX или Java, обеспечивая их взаимодействие между собой.
- **Документы ActiveX** позволяют открыть и обрабатывать в окне Microsoft Internet Explorer документ любого формата (например, файл Microsoft Excel или Word).
- **Виртуальная машина Java** позволяет любой программе просмотра Internet, поддерживающей технологию ActiveX (например, Internet Explorer 3.0) выполнять программные компоненты Java и обеспечивать их взаимодействие с программными компонентами ActiveX.
- **ActiveX Server Framework** обеспечивает серверные функции ActiveX, включая поддержку безопасных соединений, доступ к базам данных и другие.
- **Средства разработки** позволят использовать знакомые системы программирования Microsoft и других фирм для создания компонентов ActivX. К их числу относятся Visual Basic, Visual C++, Macromedia Shockwave, Adobe Photoshop, Borland Delphi, средства программирования Sybase и другие.

Основные преимущества использования технологии ActiveX:

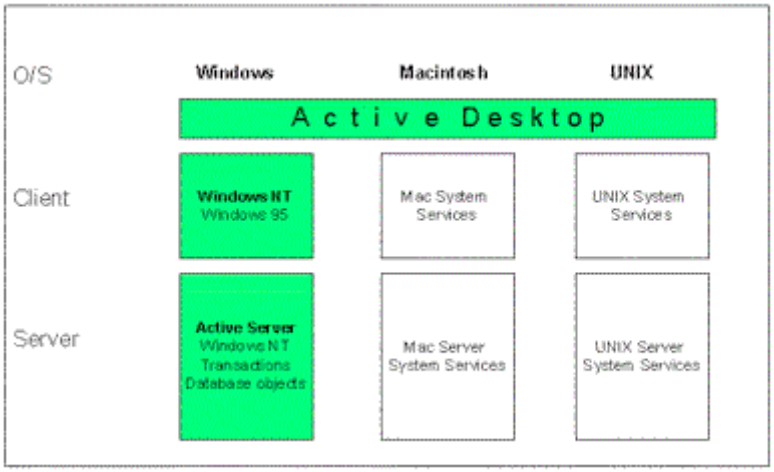
- **Быстрое написание программного кода.** Программирование сетевых взаимодействий становится очень похожим на программирование для отдельного компьютера.
- **Открытость и мобильность.** Спецификации технологии недавно были переданы в Open Group как основа открытого стандарта. Кроме того, Microsoft совместно с компаниями Metrowerks и Bristol заканчивает реализацию технологий ActiveX для платформ Macintosh и UNIX.
- **Возможность написания приложений с использованием знакомых средств разработки.** Программные элементы ActiveX могут быть созданы с помощью

Visual Basic, Visual C++, Borland Delphi, Borland C++, любых средств разработки на Java.

- **Большое количество уже существующих программных элементов ActiveX,** которые бесплатно могут применяться на серверах Web и в приложениях независимых разработчиков. Кроме того почти любой программный компонент OLE совместим с технологиями ActiveX и может применяться без модификаций в сетевых приложениях.
- **Стандартность.** Технология ActiveX основана на широко используемых стандартах Internet (TCP/IP, HTML, Java) с одной стороны и стандартах, введенных в свое время Microsoft и необходимых для сохранения совместимости (COM, OLE).

4.2. Клиентская технология ActiveX (Active Desktop)

ActiveX реализуется на машине-клиенте с помощью библиотек, поставляемых вместе с Internet Explorer 3.0. 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.

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 работы с документами любых приложений и даже запуск программ.

4.3. 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. Поддержка состояния

HTTP - это протокол без сохранения состояния, он определяет, как веб-клиенты и серверы общаются друг с другом, чтобы предоставлять пользователям документы и другие ресурсы. К сожалению, HTTP не обеспечивает прямой способ идентификации клиентов, чтобы отслеживать их при запросе нескольких страниц. Однако есть способы отслеживать пользователей непрямыми методами, которые мы рассмотрим в этой главе.

Веб-разработчики называют отслеживание пользователей ***поддержкой состояния***. Ряд взаимодействий определенного пользователя с нашим сайтом - это *сессия.* Информация, которую мы собираем для пользователя, это *информация сессии.*

Для чего необходима поддержка состояния? Если вы уважаете приватность, то отслеживание пользователей способствует этому. Хотя отслеживание пользователей можно использовать в сомнительных целях, есть законные ситуации, когда вы должны это использовать. Возьмем онлайн-магазин: чтобы покупатели могли просматривать продукты, добавлять что-то в корзину и затем расплачиваться за все выбранное, сервер должен обеспечить каждому пользователю собственную корзину. В этом случае сбор отдельных элементов из информации сессии не только допустим, но и приветствуется.

Строки запроса и дополнительная информация о пути

Можно добавить идентификатор в строку запроса или как дополнительную информацию внутри URL-документа. Когда пользователи перемещаются по сайту, CGI-приложение на лету генерирует документы, передавая идентификатор из документа в документ. Это позволяет нам отслеживать все документы, запрошенные каждым пользователем, и порядок, в котором они были запрошены. Броузер посылает эту информацию нам через строку статуса.

Скрытые поля

Скрытые поля форм позволяют встраивать «невидимую» информацию в виде имя-значение в формы так, чтобы пользователь не увидел ее, не посмотрев исходный код HTML-страницы. Как и обычные поля форм и значения, эта информация посылается CGI-приложению, когда пользователь нажимает кнопку отправки. Обычно мы используем эту технологию, чтобы учесть выбор и предпочтения пользователей, если участвует несколько форм. Также мы увидим, как CGI.pm может сделать большую часть этой работы для нас. Броузер посылает нам э информацию в строке статуса или в теле сообщения, в зависимости типа запроса (GET или POST соответственно).

Cookie на стороне клиента

Все современные броузеры поддерживают cookie на стороне клиент что позволяет хранить информацию на машине клиента и перед ' вать ее обратно на сервер при каждом запросе. Можно использовать эту технологию для хранения данных на стороне клиента, которы будут доступны нам, когда в дальнейшем пользователь запросит ресурсы с сервера. Cookie посылаются обратно клиентом в строке заголовка HTTP *Cookie.*

Преимущества и недостатки этих подходов отражены в таблице 11-1 Мы рассмотрим каждую технологию по отдельности, и если что-то в таблице останется неясным, вы сможете потом вернуться к ней. Обычно cookie на стороне клиента - это самый мощный способ поддержки состояния, но он требует что-то и от клиента. Другие технологии работают независимо от клиента, но у обеих есть ограничения на количество страниц, которые можно отследить.

Таблица 11-1. Технологии, используемые для поддержки состояния

Технология	Область применения	Надежность и производительность	Требования к клиенту
Строки запроса и дополни-тельная информация о пути	Может быть настроена для определенных групп страниц или веб-сайта целиком. Но информация о состоянии теряется, если пользователь уходит с веб-сайта, а потом возвращается	Сложно достоверю разобрать все ссылки в документе; приходится значительно расплачиваться произво-дительностью при передаче статического содержимого через CGI-сцена-рии	Не требует какого-либо особенного поведения от клиента
Скрытые поля	Работает только для нескольких отправок формы	Легко реализуется; не влияет на производительность	Не требует какого-либо особенного поведения от клиента
Cookie на стороне клиента	Работает всюду, даже если пользователь уходит на другой сайт и потом возвращается	Легко реализуется; не влияет на производительность	Требуется поддержка (и принятие) cookie клиентом

III. Заключение

Глобальная информатизация общества приводит к тому, что потребность в информации, растет с каждым новым пользователем сети. При этом задачей специалистов в области информационных технологий обеспечить пользователей полной и достоверной информацией путем простого и удобного для пользователей доступа к накопленным массивам данных.

В данной курсовой работе был рассмотрен широкий, однако далеко не полный ряд всевозможных методов, используемых в настоящее время для вышеуказанных целей. Очевидно, что задача создания полномасштабного web-приложения уже давно вышла за рамки возможностей одного человека. Спектр узких квалификаций в этой области многократно вырос за последние несколько лет, и продолжает расширяться.

Таким образом можно положить, что будущее развитие Интернета во многом определяется консолидацией усилий и слаженностью действий специалистов и разработчиков, задействованных в данной отрасли.

Список использованной литературы

1. Джейсон Мейнджер. Java: основы программирования :Пер. с англ. - К.: Издательская группа BHV,1997.-320с.
2. Симкин Стив, Бартлет Нейл, Лесли Алекс. Программирование на Java. Путеводитель :Пер. с англ. – К. НИПФ «ДиаСофт Лтд», 1996. 736 с.
3. Кристиансен Т., Торкингтон Н. Perl: Библиотека программиста :Пер. с англ.- СПб.: Издательство «Питер», 2000. – 736с.: ил.
4. Холзнер Стивен. Perl: специальный справочник :Пер. с анг. – СПб.: Питер, 2000. – 496с.: ил.
5. Хейл, Бернард Ван. JDBC: Java и базы данных :Пер. с англ. М.,1999.-320с.
6. Эферган М. Java: справочник. – СПб.: Питер, 1998. -448с.: ил.
7. http://www.citforum.ru
8. http://www.xpoint.ru
9. http://www.by.iatp.org.ua
10. http://phpclub.unet.ru
11. http://www.webmasteram.ru
12. http://kek.ksu.ru
13. http://inftech.webservis.ru
14. http://dit.vov.ru
15. http://chip.ua