Министерство образования и науки Республики Казахстан

Костанайский государственный университет им. А. Байтурсынова

Кафедра программного обеспечения

**В. Синицина**

**Web - технологии**

Учебно-методическое пособие

Костанай, 2014

УДК 32.973-01

С 38

Рецензенты:

Уаисова М.М., зав.кафедрой кафедры информационных технологий КИнЭУ им.М.Дулатова

Байманкулов А.Т., доцент кафедры программного обеспечения КГУ имени А. Байтурсынова, д.ф.-м.н.

Салыкова О.С., доцент кафедры программного обеспечения КГУ имени. А. Байтурсынова, к.т.н.

Составитель: Синицина В.Б., ст.преподаватель кафедры программного обеспечения

Синицина В.Б.

С 38 Web-технологии. Учебно-методическое пособие. - Костанай: КГУ имени А. Байтурсынова, 2014.- 122 с.

В рамках учебно-методического пособия рассматривается широкий спектр протоколов, стандартов и технологий, имеющих непосредственное отношение к разработке web-приложений. В курсе изучаются различные роли и ответственность клиентов и серверов для различных приложений в WWW, общие принципы работы клиентских и серверных языков и технологий в WWW, основные протоколы, необходимые для создания и работы web-приложений, основные принципы и подходы к web-интеграции приложений, разнородных компонент и систем, основные продукты и технологии Майкрософт, используемые для разработки web-контента и web-приложений.

Предназначено для студентов, изучающих основы web-технологий. Может быть использовано преподавателями, магистрантами для проведения учебных занятий.

Утверждено учебно-методическим советом Костанайского государственного университета имени А.Байтурсынова протокол от \_\_\_\_\_\_\_ № \_\_.

© КГУ имени А. Байтурсынова, 2014

© Синицина В.Б.

Содержание

|  |  |
| --- | --- |
| Тема. Введение в веб-технологии: структура и принципы Веб………….. | 4 |
| Тема. Введение в клиент-серверные технологии Веб. Протокол http…… | 17 |
| Тема. Клиентские сценарии и приложения……………………………….. | 31 |
| Тема. Серверные веб-приложения…………………………………………. | 41 |
| Тема. Языки разработки сценариев Perl и PHP……………………………. | 53 |
| Тема. Введение в C# и платформу Visual Studio.Net……………………… | 60 |
| Тема. Архитектура веб-приложений ASP.NET. Разработка веб-приложений на платформе .NET…………………………………………… | 67 |
| Тема. Интерфейсы взаимодействия веб-приложений с СУБД…………… | 72 |
| Тема. Введение в XML……………………………………………………… | 73 |
| Тема. Языки описания cхем XML………………………………………….. | 79 |
| Тема. DOM XML. Преобразование XML документов……………………. | 87 |
| Тема. Интеграция и взаимодействие в сети Веб…………………………... | 93 |
| Тема. Организация процесса разработки веб-контента.CMS/CMF……… | 100 |
| Тема. Синдикация и аггрегирование веб-контента……………………….. | 102 |
| Тема. Веб-порталы. Классификация веб-порталов………………………... | 105 |
| Тема. Введение в технологию AJAX. Разработка мобильных веб-приложений………………………………………………………………….. | 109 |
| Тема. Введение в Веб 2.0…………………………………………………… | 114 |
| Тема. Приложения для социальных сетей…………………………………. | 118 |

**Тема. Введение в веб-технологии: структура и принципы Веб**

***Что же такое Интернет?***

По сути, это самая большая в мире сеть, не имеющая единого центра управления, но работающая по единым правилам и предоставляющая своим пользователям единый набор услуг. Интернет можно рассматривать как "сеть сетей", каждая из которых управляется независимым оператором – поставщиком услуг Интернета (ISP, Internet Service Provider).

С точки зрения пользователей Интернет представляет собой набор информационных ресурсов, рассредоточенных по различным сетям, включая ISP-сети, корпоративные сети, сети и отдельные компьютеры домашних пользователей. Каждый отдельный компьютер в данной сети называется хостом (от английского термина host).

Сегодняшний Интернет обязан своему появлению объединенной сети ARPANET, которая начиналась как скромный эксперимент в новой тогда технологии коммутации пакетов (табл. 1.1). Сеть ARPANET была развернута в 1969 г. и состояла поначалу всего из четырех узлов с коммутацией пакетов, используемых для взаимодействия горстки хостов и терминалов. Первые линии связи, соединявшие узлы, работали на скорости всего 50 Кбит/с. Сеть ARPANET финансировалась управлением перспективного планирования научно-исследовательских работ ARPA (Advanced Research Projects Agency) министерства обороны США и предназначалась для изучения технологии и протоколов коммутации пакетов, которые могли бы использоваться для кооперативных распределенных вычислений.

Таблица 1.1 - Хронология развития Интернета (с 1966 по 2000 г.)

|  |  |
| --- | --- |
| Год | Событие |
| 1966 | Эксперимент с коммутацией пакетов управления ARPA |
| 1969 | Первые работоспособные узлы сети ARPANET |
| 1972 | Изобретение распределенной электронной почты |
| 1973 | Первые компьютеры, подключенные к сети ARPANET за пределами США |
| 1975 | Сеть ARPANET передана в ведение управления связи министерства обороны США |
| 1980 | Начинаются эксперименты с TCP/IP |
| 1981 | Каждые 20 дней к сети добавляется новый хост |
| 1983 | Завершен переход на TCP/IP |
| 1986 | Создана магистраль NSFnet |

Продолжение таблицы 1.1

|  |  |
| --- | --- |
| 1990 | Сеть ARPANET прекратила существование |
| 1991 | Появление Gopher |
| 1991 | Изобретение Всемирной паутины. Выпущена система PGP. Появление Mosaic |
| 1995 | Приватизация магистрали Интернета |
| 1996 | Построена магистраль ОС-3 (155 Мбит/с) |
| 1998 | Число зарегистрированных доменных имен превысило 2 млн. |
| 2000 | Количество индексируемых веб-страниц превысило 1 млрд. |

На рисунке 1.1 представлен график, показывающий динамику роста числа хостов (как формально зарегистрированных и так активно функционирующих).



Рисунок 1.1 - Динамика роста числа хостов в Интернет (взято с сайта www.netcraft.com).

Возможно ли централизованное управление в такой глобальной сети? Ответ на данный вопрос будет отрицательным, поскольку, во-первых, данная сеть является транснациональной и, во-вторых, в силу исторических предпосылок ее формирования.

Тем не менее, в Интернете могут проявляться опосредованные формы централизации в форме единой технической политики, согласованном наборе технических стандартов, назначении имен и адресов компьютеров и сетей, входящих в Интернет.

То есть Интернет является децентрализованной сетью, что имеет свои достоинства и недостатки.

1. Достоинства:

* Легкость наращивания Интернета путем заключения соглашения между двумя ISP.

1. Недостатки:

* Сложность модернизации технологий и услуг Интернета, поскольку требуются согласованные усилия всех поставщиков услуг.
* Невысокая надежность услуг Интернета.
* Ответственность за работоспособность отдельных сегментов этой сети возлагается на поставщиков услуг Интернета.

Существуют различные типы поставщиков услуг Интернета:

* просто поставщик услуг Интернета выполняет транспортную функцию для конечных пользователей – передачу их трафика в сети других поставщиков услуг Интернета;
* поставщик интернет-контента имеет собственные информационно-справочные ресурсы, предоставляя их содержание в виде веб-сайтов;
* поставщик услуг хостинга предоставляет свои помещения, каналы связи и серверы для размещения внешнего контента;
* поставщик услуг по доставке контента занимается только доставкой контента в многочисленные точки доступа с целью повышения скорости доступа пользователей к информации;
* поставщик услуг по поддержке приложений предоставляет клиентам доступ к крупным универсальным программным продуктам, например SAP R3;
* поставщик биллинговых услуг обеспечивает оплату счетов по Интернету;

***О роли стандартизации в Интернет***

Как следует из всего вышеизложенного, Интернет является очень сложной сетью, и соответственно такой же сложной является задача организации взаимодействия между устройствами сети. Для решения такого рода задач используется декомпозиция, т.е. разбиение сложной задачи на несколько более простых задач-модулей. Одной из концепций, реализующих декомпозицию, является многоуровневый подход. Такой подход дает возможность проводить разработку, тестирование и модификацию каждого отдельного уровня независимо от других уровней. Иерархическая декомпозиция позволяет, перемещаясь в направлении от более низких к более высоким уровням переходить к более простому представлению решаемой задачи.

Специфика многоуровневого представления сетевого взаимодействия состоит в том, что в процессе обмена сообщениями участвуют как минимум две стороны, для которых необходимо обеспечить согласованную работу двух иерархий аппаратно-программных средств. Каждый из уровней должен поддерживать интерфейс с выше- и нижележащими уровнями собственной иерархии средств и интерфейс со средствами взаимодействия другой стороны на том же уровне иерархии. Данный тип интерфейса называется протоколом (см. рисунок 1.2).

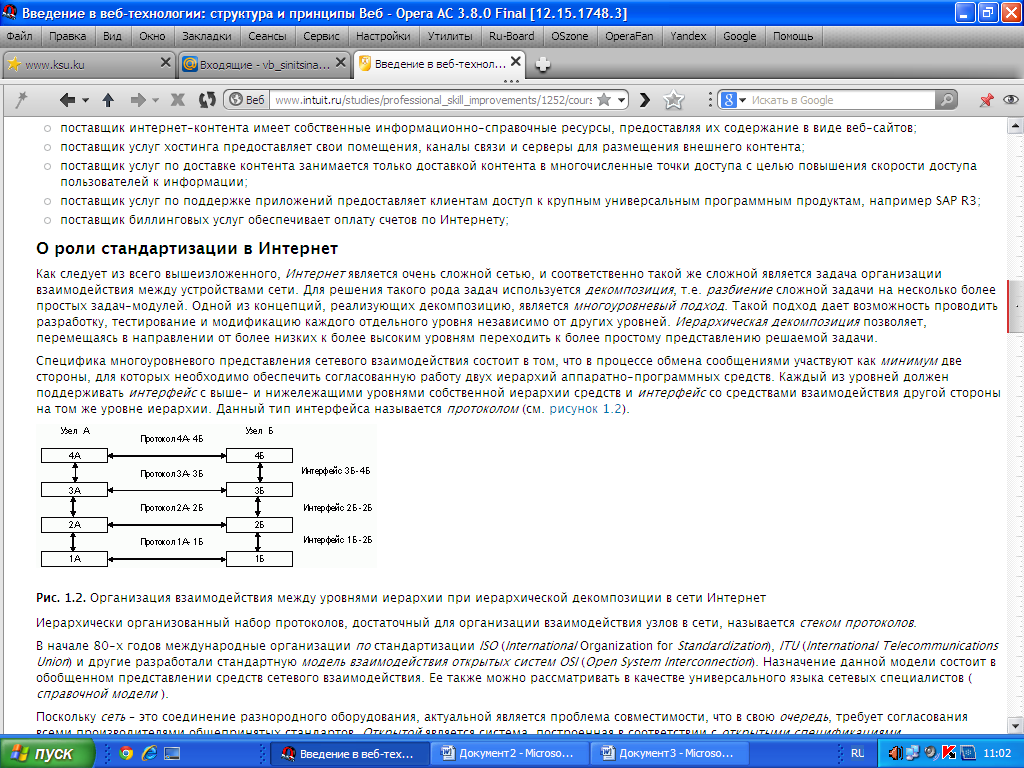


Рисунок 1.2 - Организация взаимодействия между уровнями иерархии при иерархической декомпозиции в сети Интернет

Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов в сети, называется стеком протоколов.

В начале 80-х годов международные организации по стандартизации ISO (International Organization for Standardization), ITU (International Telecommunications Union) и другие разработали стандартную модель взаимодействия открытых систем OSI (Open System Interconnection). Назначение данной модели состоит в обобщенном представлении средств сетевого взаимодействия. Ее также можно рассматривать в качестве универсального языка сетевых специалистов (справочной модели).

Поскольку сеть – это соединение разнородного оборудования, актуальной является проблема совместимости, что в свою очередь, требует согласования всеми производителями общепринятых стандартов. Открытой является система, построенная в соответствии с открытыми спецификациями.

Спецификация представляет собой формализованное описание аппаратных (программных) компонентов, способов их функционирования, взаимодействия с другими компонентами, условий эксплуатации, особых характеристик. Под открытыми спецификациями понимаются опубликованные, общедоступные спецификации, соответствующие стандартам и принятые в результате достижения согласия после всестороннего обсуждения всеми заинтересованными сторонами. Использование открытых спецификаций при разработке систем позволяет третьим сторонам разрабатывать для этих систем аппаратно-программные средства расширения и модификации, а также создавать программно-аппаратные комплексы из продуктов разных производителей.

Если две сети построены с соблюдением принципов открытости, это дает следующие преимущества:

* Возможность построения сети из аппаратных и программных средств различных производителей, придерживающихся стандарта;
* Безболезненная замена отдельных компонентов сети другими, более совершенными;
* Легкость сопряжения одной сети с другой.

В рамках модели OSI средства взаимодействия делятся на семь уровней: прикладной, представления, сеансовый, транспортный, сетевой, канальный и физический. В распоряжение программистов предоставляется прикладной программный интерфейс, позволяющий обращаться с запросами к самому верхнему уровню, а именно, - уровню приложений.

Сеть Интернет строилась в полном соответствии с принципами открытых систем. В разработке стандартов этой сети принимали участие тысячи специалистов-пользователей сети из вузов, научных организаций и компаний. Результат работы по стандартизации воплощается в документах RFC.

RFC (англ. Request for Comments) — документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и Стандарты, широко применяемые во Всемирной сети. В настоящее время первичной публикацией документов RFC занимается IETF под эгидой открытой организации Общество Интернета ( ISOC ). Правами на RFC обладает именно Общество Интернет. Формат RFC появился в 1969 г. при обсуждении проекта ARPANET. Первые RFC распространялись в печатном виде на бумаге в виде обычных писем, но уже с декабря 1969 г., когда заработали первые сегменты ARPANET, документы начали распространяться в электронном виде. В таблице 1.2 приведены некоторые из наиболее известных документов RFC.

Таблица 1.2 - Примеры популярных RFC-документов.

|  |  |
| --- | --- |
| Номер RFC | Тема |
| RFC 768 | UDP |
| RFC 791 | IP |
| RFC 793 | TCP |
| RFC 822 | Формат электронной почты, заменен RFC 2822 |
| RFC 959 | FTP |
| RFC 1034 | DNS — концепция |
| RFC 1035 | DNS — внедрение |
| RFC 1591 | Структура доменных имен |
| RFC 1738 | URL |
| RFC 1939 | Протокол POP версии 3 (POP3) |
| RFC 2026 | Процесс стандартизации в Интернете |
| RFC 2045 | MIME |
| RFC 2231 | Кодировка символов |
| RFC 2616 | HTTP |
| RFC 2822 | Формат электронной почты |
| RFC 3501 | IMAP версии 4 издание 1 (IMAP4rev1) |

Основным организационным подразделением, координирующим работу по стандартизации Интернет, является ISOC (Internet Society), объединяющее порядка 100 тысяч участников, которые занимаются различными аспектами развития данной сети. ISOC курирует работу IAB (Internet Architecture Board), включающую две группы:

* IRTF (Internet Research Task Force). Координирует долгосрочные исследовательские проекты, относящиеся к TCP/IP;
* IETF (Internet Engineering Task Force). Инженерная группа, определяющая спецификации для последующих стандартов Интернет.

Разработкой стандартов для сети Веб, начиная с 1994 года, занимается Консорциум W3C (World Wide Web Consortium), основанный и до сих пор возглавляемый Тимом Бернерсом-Ли.

Консорциум W3C — организация, разрабатывающая и внедряющая технологические стандарты для Интернета и WWW. Миссия W3C формулируется следующим образом: "Полностью раскрыть потенциал Всемирной паутины путем создания протоколов и принципов, гарантирующих долгосрочное развитие Сети". Две другие важнейшие задачи Консорциума — обеспечить полную "интернационализацию Сети" и сделать ее доступной для людей с ограниченными возможностями.

W3C разрабатывает для WWW единые принципы и стандарты, называемые " Рекомендациями ", которые затем внедряются разработчиками программ и оборудования. Благодаря Рекомендациям достигается совместимость между программными продуктами и оборудованием различных компаний, что делает сеть WWW более совершенной, универсальной и удобной в использовании.

Все Рекомендации W3C открыты, то есть, не защищены патентами и могут внедряться любым человеком без каких-либо финансовых отчислений Консорциуму.

Для удобства пользователей Консорциумом созданы специальные программы-валидаторы (англ. Online Validation Service), которые доступны по сети и могут за несколько секунд проверить документы на соответствие популярным Рекомендациям W3C. Консорциумом также созданы многие другие утилиты для облегчения работы веб-мастеров и программистов. Большинство утилит — это программы с открытым исходным кодом, все они бесплатные. В последнее время, повинуясь мировым тенденциям, Консорциум, в целом, гораздо больше внимания уделяет проектам с открытым исходным кодом.

В российском сегменте Интернета имеется своя организация - Российский НИИ Развития Общественных Сетей РОСНИИРОС (Russian Institute for Public Networks, RIPN). РОСНИИРОС занимается координацией российских исследований и разработок в Интернете.

Прежде чем перейти к описанию структуры, принципов работы и основных протоколов сети Веб, рассмотрим основной стек протоколов сети Интернет - стек TCP/IP.

***Стек протоколов TCP/IP***

Эти протоколы изначально ориентированы на глобальные сети, в которых качество соединительных каналов не идеально. Он позволяет создавать глобальные сети, компьютеры в которых соединены друг с другом самыми разными способами от высокоскоростных оптоволоконных кабелей и спутниковых каналов до коммутируемых телефонных линий. TCP/IP соответствует модели OSI достаточно условно и содержит 4 уровня. Прикладной уровень стека соответствует трем верхним уровням модели OSI: прикладному, представления и сеансовому.

В сети данные всегда передаются блоками относительно небольшого размера. Каждый блок имеет префиксную часть (заголовок), описывающую содержимое блока, и суффиксную, содержащую, например, информацию для контроля целостности передаваемого блока данных.

Название стека протоколов TCP/IP состоит из названий двух разных протоколов. Протокол IP (Internet Protocol) представляет собой протокол нижнего (сетевого) уровня и отвечает за передачу пакетов данных в сети. Он относится к так называемым протоколам датаграмм и работает без подтверждений. Последнее означает, что при его использовании доставка пакетов данных не гарантируется и не подтверждается. Не гарантируется также и то, что пакеты достигнут пункта назначения в той последовательности, в которой они были отправлены.

К протоколам сетевого уровня относится также протокол межсетевых управляющих сообщений ICMP (Internet Control Message Protocol), предназначенный для передачи маршрутизатором источнику информации об ошибках при передаче пакета.

Очевидно, что намного удобнее передавать данные по каналу, который работает корректно, доставляя все пакеты по порядку. Поэтому над протоколом IP работает протокол передачи данных более высокого (транспортного) уровня — TCP (Transmission Control Protocol). Посылая и принимая пакеты через протокол IP, протокол TCP гарантирует доставку всех переданных пакетов данных в правильной последовательности.

Следует отметить, что при использовании протокола IP обеспечивается более быстрая передача данных, так как не тратится время на подтверждение приема каждого пакета. Есть и другие преимущества. Одно из них заключается в том, что он позволяет рассылать пакеты данных в широковещательном режиме, при котором они достигают всех компьютеров физической сети. Что же касается протокола TCP, то для передачи данных с его помощью необходимо создать канал связи между компьютерами. Он и создается с использованием протокола IP.

Для идентификации сетевых интерфейсов используются 3 типа адресов:

* аппаратные адреса (или MAC-адреса);
* сетевые адреса (IP-адреса);
* символьные (доменные) имена.

В рамках IP протокола для создания глобальной системы адресации, не зависящей от способов адресации узлов в отдельных сетях, используется пара идентификаторов, состоящая из номера сети и номера узла. При этом IP-адрес идентифицирует не отдельный компьютер или маршрутизатор, а одно сетевое соединение в составе сети, в которую он входит; то есть конечный узел может входить в несколько IP-сетей.

***Система доменных имен DNS***

Несмотря на то, что аппаратное и программное обеспечение в рамках TCP/IP сетей для идентификации узлов использует IP-адреса, пользователи предпочитают символьные имена (доменные имена).

Первоначально в локальных сетях из небольшого числа компьютеров применялись плоские имена, состоящие из последовательности символов без разделения их на отдельные части, например MYCOMP. Для установления соответствия между символьными именами и числовыми адресами использовались широковещательные запросы. Однако для больших территориально распределенных сетей, работающих на основе протокола TCP/IP такой способ оказался неэффективным. Поэтому для установления соответствия между доменным именем и IP-адресом используется специальная система доменных имен (DNS, Domain Name System), которая основана на создаваемых администраторами сети таблиц соответствия.

В сетях TCP/IP используется доменная система имен, имеющая иерархическую (в виде дерева) структуру. Данная структура имен напоминает иерархию имен, используемую во многих файловых системах. Запись доменного имени начинается с самой младшей составляющей, затем после точки следует следующая по старшинству символьная часть имени и так далее. Последовательность заканчивается корневым именем, например: company.yandex.ru.

Построенная таким образом система имен позволяет разделять административную ответственность по поддержке уникальности имен в пределах своего уровня иерархии между различными людьми или организациями.

Совокупность имен, у которых несколько старших составных частей совпадают, образуют домен имен.

Корневой домен управляется центральными органами Интернета: IANA и Internic.

Домены верхнего уровня назначаются для каждой страны, а также для различных типов организаций. Имена этих доменов должны следовать международному стандарту ISO 3166. Для обозначения стран используются двухбуквенные аббревиатуры, например ru (Российская Федерация), us (США), it (Италия), fr (Франция).

Для различных типов организаций используются трехбуквенные аббревиатуры:

* net – сетевые организации;
* org – некоммерческие организации;
* com - коммерческие организации;
* edu – образовательные организации;
* gov – правительственные организации.

Администрирование каждого домена возлагается на отдельную организацию, которая делегирует администрирование поддоменов другим организациям.

Для получения доменного имени необходимо зарегистрироваться в соответствующей организации, которой организация InterNIC делегировала свои полномочия по распределению доменных имен.

Регистратором доменных имен в зоне ru до 2005 г. являлся Российский научно-исследовательский институт развития общественных сетей ( РосНИИРОС ). В настоящее время регистрация доменов осуществляется одним из действующих регистраторов.

В TCP/IP сетях соответствие между доменными именами и IP-адресами может устанавливаться как локальными средствами, так и централизованными службами. Первоначально соответствие задавалось с помощью создаваемого вручную на хосте файла hosts.txt, состоящего из строк, содержащих пару вида "доменное имя – IP-адрес". Однако с активным ростом Интернета такое решение оказалось немасштабируемым.

Альтернативное решение – централизованная служба DNS, использующая распределенную базу отображений "доменное имя – IP-адрес". Сервер домена хранит только имена, которые заканчиваются на следующем ниже по дереву уровне. Это позволяет распределять более равномерно нагрузку по разрешению имен между всеми DNS-серверами. Каждый DNS-сервер помимо таблицы отображения имен содержит ссылки на DNS-серверы своих поддоменов.

Существуют две схемы разрешения DNS-имен.

Нерекурсивная процедура:

* DNS-клиент обращается к корневому DNS-серверу с указанием полного доменного имени;
* DNS-сервер отвечает клиенту, указывая адрес следующего DNS-сервера, обслуживающего домен верхнего уровня, заданный в следующей старшей части имени;
* DNS-клиент делает запрос следующего DNS-сервера, который отсылает его к DNS-серверу нужного поддомена и т.д., пока не будет найден DNS-сервер, в котором хранится соответствие запрошенного имени IP-адресу. Сервер дает окончательный ответ клиенту.

Рекурсивная процедура:

* DNS-клиент запрашивает локальный DNS-сервер, обслуживающий поддомен, которому принадлежит клиент;
* Далее
* Если локальный DNS-сервер знает ответ, он возвращает его клиенту
* Если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу. После получения ответа сервер передает его клиенту.

Таким образом, при рекурсивной процедуре клиент фактически перепоручает работу своему серверу. Для ускорения поиска IP-адресов DNS-серверы широко применяют кэширование (на время от часов до нескольких дней) проходящих через них ответов.

***Структура и принципы WWW***

Сеть WWW образуют миллионы веб-серверов, расположенных по всему миру. Веб - сервер является программой, запускаемой на подключенном к сети компьютере и передающей данные по протоколу HTTP.

Для идентификации ресурсов (зачастую файлов или их частей) в WWW используются идентификаторы ресурсов URI (Uniform Resource Identifier). Для определения местонахождения ресурсов в этой сети используются локаторы ресурсов URL (Uniform Resource Locator). Такие URL-локаторы представляют собой комбинацию URI и системы DNS.

Доменное имя (или IP-адрес) входит в состав URL для обозначения компьютера (его сетевого интерфейса), на котором работает программа веб-сервер.

На клиентском компьютере для просмотра информации, полученной от веб-сервера, применяется специальная программа — веб-браузер. Основная функция веб-браузера - отображение гипертекстовых страниц (веб-страниц). Для создания гипертекстовых страниц в WWW изначально использовался язык HTML. Множество веб-страниц образуют веб-сайт.

***Прокси-серверы***

Прокси-сервер (proxy-server) — служба в компьютерных сетях, позволяющая клиентам выполнять косвенные запросы к другим сетевым службам.

Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс, расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кеша (если имеется). В некоторых случаях запрос клиента или ответ сервера может быть изменен прокси-сервером в определенных целях. Также прокси-сервер позволяет защищать клиентский компьютер от некоторых сетевых атак.

Чаще всего прокси-серверы применяются для следующих целей:

* обеспечение доступа с компьютеров локальной сети в Интернет;
* кеширование данных: если часто происходят обращения к одним и тем же внешним ресурсам, то можно держать их копию на прокси-сервере и выдавать по запросу, снижая тем самым нагрузку на канал во внешнюю сеть и ускоряя получение клиентом запрошенной информации.
* сжатие данных: прокси-сервер загружает информацию из Интернета и передает информацию конечному пользователю в сжатом виде.
* защита локальной сети от внешнего доступа: например, можно настроить прокси-сервер так, что локальные компьютеры будут обращаться к внешним ресурсам только через него, а внешние компьютеры не смогут обращаться к локальным вообще (они "видят" только прокси-сервер).
* ограничение доступа из локальной сети к внешней: например, можно запретить доступ к определенным веб-сайтам, ограничить использование интернета каким-то локальным пользователям, устанавливать квоты на трафик или полосу пропускания, фильтровать рекламу и вирусы.
* анонимизация доступа к различным ресурсам. Прокси-сервер может скрывать сведения об источнике запроса или пользователе. В таком случае целевой сервер видит лишь информацию о прокси-сервере, например, IP-адрес, но не имеет возможности определить истинный источник запроса. Существуют также искажающие прокси-серверы, которые передают целевому серверу ложную информацию об истинном пользователе.

***Протоколы Интернет прикладного уровня***

Самый верхний уровень в иерархии протоколов Интернет занимают следующие протоколы прикладного уровня:

DNS - распределенная система доменных имен, которая по запросу, содержащему доменное имя хоста сообщает IP адрес;

HTTP - протокол передачи гипертекста в Интернет;

HTTPS - расширение протокола HTTP, поддерживающее шифрование;

FTP (File Transfer Protocol - RFC 959) - протокол, предназначенный для передачи файлов в компьютерных сетях;

Telnet (TELecommunication NETwork - RFC 854) - сетевой протокол для реализации текстового интерфейса по сети;

SSH (Secure Shell - RFC 4251) - протокол прикладного уровня, позволяющий производить удаленное управление операционной системой и передачу файлов. В отличие от Telnet шифрует весь трафик;

POP3 – протокол почтового клиента, который используется почтовым клиентом для получения сообщений электронной почты с сервера;

IMAP - протокол доступа к электронной почте в Интернет;

SMTP – протокол, который используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю;

LDAP - протокол для доступа к службе каталогов X.500, является широко используемым стандартом доступа к службам каталогов;

XMPP (Jabber) - основанный на XML расширяемый протокол для мгновенного обмена сообщениями в почти реальном времени;

SNMP - базовый протокол управления сети Internet.

Рассмотрим более подробно некоторые из этих протоколов.

1. FTP

FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер; кроме того, возможен режим передачи файлов между серверами; FTP позволяет обмениваться файлами и выполнять операции над ними через TCP-сети. Данный протокол работает независимо от операционных систем. Исторически протокол FTP предложил открытую функциональность, обеспечивая прозрачный перенос файлов с одного компьютера на другой по сети. Это не так тривиально, как может показаться, так как у разнотипных компьютеров могут различаться размеры слов, биты в словах могут храниться в неодинаковом порядке или использоваться разные форматы слов.

1. Telnet

Название "telnet" имеют также некоторые утилиты, реализующие клиентскую часть протокола. Протокол telnet работает в соответствии с принципами архитектуры "клиент-сервер" и обеспечивает эмуляцию алфавитно-цифрового терминала, ограничивая пользователя режимом командной строки. Приложение telnet предоставило язык для общения терминалов с удаленными компьютерами. Когда появилась сеть ARPANET, для каждой компьютерной системы требовались собственные терминалы. Приложение telnet стало общим знаменателем для терминалов. Достаточно было написать для каждого компьютера программное обеспечение, поддерживающее "терминал telnet ", чтобы один терминал мог взаимодействовать с компьютерами всех типов.

1. SSH

Сходен по функциональности с протоколами telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH-клиенты и SSH-серверы имеются для большинства операционных систем.

1. Почтовые протоколы.

Хотя telnet и FTP были (и остаются) полезными, первым приложением, совершившим переворот в сознании пользователей компьютеров сети ARPANET, стала электронная почта. До сети ARPANET существовали системы электронной почты, но все они были однокомпьютерными системами. В 1972 г. Рэй Томлинсон (Ray Tomlinson) из компании BBN написал первый пакет, предоставляющий распределенные почтовые услуги в компьютерной сети из нескольких компьютеров. Уже к 1973 г. исследования управления ARPA показали, что три четверти всего трафика сети ARPANET составляла электронная почта. Польза электронной почты оказалась столь велика, что все больше пользователей стремилось подключиться к сети ARPANET, в результате чего возрастала потребность в добавлении новых узлов и использовании высокоскоростных линий. Таким образом, появилась тенденция, сохраняющаяся и по сей день.

* POP3 (Post Office Protocol Version 3 - RFC 1939) — протокол, который используется почтовым клиентом для получения сообщений электронной почты с почтового сервера;
* IMAP (Internet Message Access Protocol - RFC 3501) — протокол доступа к электронной почте. Аналогичен POP3, однако предоставляет пользователю богатые возможности для работы с почтовыми ящиками, находящимися на центральном сервере. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без необходимости постоянной пересылки с сервера и обратно файлов с полным содержанием писем.
* SMTP (Simple Mail Transfer Protocol — RFC 2821) — протокол, предназначенный для передачи электронной почты. Используется для отправки почты от пользователей к серверам и между серверами для дальнейшей пересылки к получателю. Для приема почты почтовый клиент должен использовать протоколы POP3 или IMAP.

**Тема. Введение в клиент-серверные технологии Веб. Протокол HTTP**

Базовым протоколом сети гипертекстовых ресурсов Веб является протокол HTTP. В его основу положено взаимодействие " клиент-сервер ", то есть предполагается, что:

Потребитель- клиент инициировав соединение с поставщиком-сервером посылает ему запрос;

Поставщик- сервер, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.

При этом возможны два способа организации работы компьютера-клиента:

Тонкий клиент - это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с веб-приложениями.

Толстый клиент, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

Прежде чем перейти к конкретным клиент-серверным веб-технологиям, рассмотрим основные принципы и структуру базового протокола HTTP.

***Протокол http***

HTTP (HyperText Transfer Protocol - RFC 1945, RFC 2616) - протокол прикладного уровня для передачи гипертекста.

Центральным объектом в HTTP является ресурс, на который указывает URL в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя изначально данный протокол предназначен для передачи символьной информации. На первый взгляд это может показаться излишней тратой ресурсов. Действительно, данные в символьном виде занимают больше памяти, сообщения создают дополнительную нагрузку на каналы связи, однако подобный формат имеет много преимуществ. Сообщения, передаваемые по сети, удобочитаемы, и, проанализировав полученные данные, системный администратор может легко найти ошибку и устранить ее. При необходимости роль одного из взаимодействующих приложений может выполнять человек, вручную вводя сообщения в требуемом формате.

В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера. Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Например, клиентское веб-приложение, посылающее запросы, может отслеживать задержки ответов, а веб-сервер может хранить IP-адреса и заголовки запросов последних клиентов.

Все программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:

* Серверы - поставщики услуг хранения и обработки информации (обработка запросов).
* Клиенты - конечные потребители услуг сервера (отправка запросов).
* Прокси-серверы для поддержки работы транспортных служб.

Основными клиентами являются браузеры, например: Internet Explorer, Opera, Mozilla Firefox, Netscape Navigator и другие. Наиболее популярными реализациями веб-серверов являются: Internet Information Services (IIS), Apache, lighttpd, nginx. Наиболее известные реализации прокси-серверов: Squid, UserGate, Multiproxy, Naviscope.

"Классическая" схема HTTP-сеанса выглядит так.

* Установление TCP-соединения.
* Запрос клиента.
* Ответ сервера.
* Разрыв TCP-соединения.

Таким образом, клиент посылает серверу запрос, получает от него ответ, после чего взаимодействие прекращается. Обычно запрос клиента представляет собой требование передать HTML-документ или какой-нибудь другой ресурс, а ответ сервера содержит код этого ресурса.

В состав HTTP-запроса, передаваемого клиентом серверу, входят следующие компоненты.

* Строка состояния (иногда для ее обозначения используют также термины строка-статус, или строка запроса).
* Поля заголовка.
* Пустая строка.
* Тело запроса.

Строку состояния вместе с полями заголовка иногда называют также заголовком запроса.

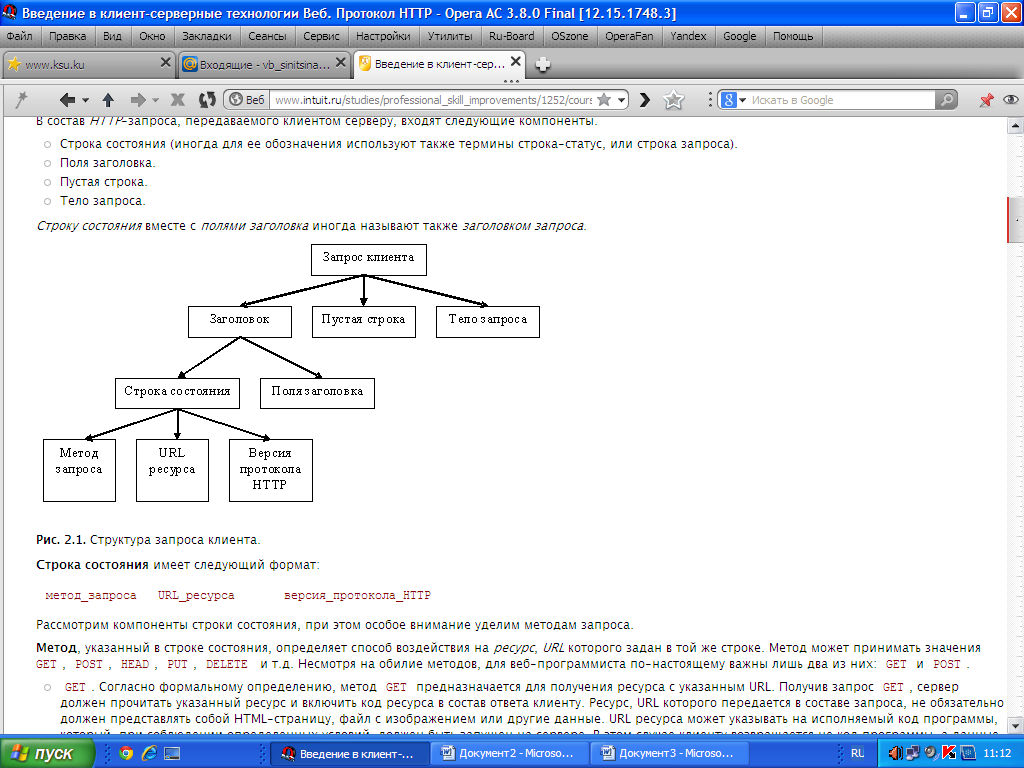


Рисунок 2.1 - Структура запроса клиента

Строка состояния имеет следующий формат:

метод\_запроса URL\_pecypca версия\_протокола\_НТТР

Рассмотрим компоненты строки состояния, при этом особое внимание уделим методам запроса.

Метод, указанный в строке состояния, определяет способ воздействия на ресурс, URL которого задан в той же строке. Метод может принимать значения GET, POST, HEAD, PUT, DELETE и т.д. Несмотря на обилие методов, для веб-программиста по-настоящему важны лишь два из них: GET и POST.

* GET. Согласно формальному определению, метод GET предназначается для получения ресурса с указанным URL. Получив запрос GET, сервер должен прочитать указанный ресурс и включить код ресурса в состав ответа клиенту. Ресурс, URL которого передается в составе запроса, не обязательно должен представлять собой HTML-страницу, файл с изображением или другие данные. URL ресурса может указывать на исполняемый код программы, который, при соблюдении определенных условий, должен быть запущен на сервере. В этом случае клиенту возвращается не код программы, а данные, сгенерированные в процессе ее выполнения. Несмотря на то что, по определению, метод GET предназначен для получения информации, он может применяться и в других целях. Метод GET вполне подходит для передачи небольших фрагментов данных на сервер.
* POST. Согласно тому же формальному определению, основное назначение метода POST - передача данных на сервер. Однако, подобно методу GET, метод POST может применяться по-разному и нередко используется для получения информации с сервера. Как и в случае с методом GET, URL, заданный в строке состояния, указывает на конкретный ресурс. Метод POST также может использоваться для запуска процесса.
* Методы HEAD и PUT являются модификациями методов GET и POST.

Версия протокола HTTP, как правило, задается в следующем формате:

HTTP/версия.модификация

Поля заголовка, следующие за строкой состояния, позволяют уточнять запрос, т.е. передавать серверу дополнительную информацию. Поле заголовка имеет следующий формат:

Имя\_поля: Значение

Назначение поля определяется его именем, которое отделяется от значения двоеточием.

Имена некоторых наиболее часто встречающихся в запросе клиента полей заголовка и их назначение приведены в таблице 2.1.

Таблица 2.1 - Поля заголовка запроса HTTP

|  |  |
| --- | --- |
| Поля заголовка HTTP -запроса | Значение |
| Host | Доменное имя или IP-адрес узла, к которому обращается клиент |
| Referer | URL документа, который ссылается на ресурс, указанный в строке состояния |
| From | Адрес электронной почты пользователя, работающего с клиентом |
| Accept | MIME-типы данных, обрабатываемых клиентом. Это поле может иметь несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Accept используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент |
| Accept-Language | Набор двухсимвольных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом |
| Accept-Charset | Перечень поддерживаемых наборов символов |
| Content-Type | MIME-тип данных, содержащихся в теле запроса (если запрос не состоит из одного заголовка) |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| Content-Length | Число символов, содержащихся в теле запроса (если запрос не состоит из одного заголовка) |
| Range | Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть |
| Connection | Используется для управления TCP-соединением. Если в поле содержится Close, это означает, что после обработки запроса сервер должен закрыть соединение. Значение Keep-Alive предлагает не закрывать TCP-соединение, чтобы оно могло быть использовано для последующих запросов |
| User-Agent | Информация о клиенте |

Во многих случаях при работе в Веб тело запроса отсутствует. При запуске CGI-сценариев данные, передаваемые для них в запросе, могут размещаться в теле запроса.

Ниже представлен пример HTML-запроса, сгенерированного браузером

GET http://oak.oakland.edu/ HTTP/1.0

Connection: Keep-Alive

User-Agent: Mozilla/4.04 [en] (Win95; I)

Host: oak.oakland.edu

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, \*/\*

Accept-Language: en

Accept-Charset: iso-8859-l,\*,utf-8

Получив от клиента запрос, сервер должен ответить ему. Знание структуры ответа сервера необходимо разработчику веб-приложений, так как программы, которые выполняются на сервере, должны самостоятельно формировать ответ клиенту.

Подобно запросу клиента, ответ сервера также состоит из четырех перечисленных ниже компонентов.

* Строка состояния.
* Поля заголовка.
* Пустая строка.
* Тело ответа.

Ответ сервера клиенту начинается со строки состояния, которая имеет следующий формат:

Версия\_протокола Код\_ответа Пояснительное\_сообщение

* Версия\_протокола задается в том же формате, что и в запросе клиента, и имеет тот же смысл.
* Код\_ответа - это трехзначное десятичное число, представляющее в закодированном виде результат обслуживания запроса сервером.
* Пояснительное\_сообщение дублирует код ответа в символьном виде. Это строка символов, которая не обрабатывается клиентом. Она предназначена для системного администратора или оператора, занимающегося обслуживанием системы, и является расшифровкой кода ответа.

Из трех цифр, составляющих код ответа, первая (старшая) определяет класс ответа, остальные две представляют собой номер ответа внутри класса. Так, например, если запрос был обработан успешно, клиент получает следующее сообщение:

HТТР/1.0 200 ОК

Как видно, за версией протокола HTTP 1.0 следует код 200. В этом коде символ 2 означает успешную обработку запроса клиента, а остальные две цифры (00) — номер данного сообщения.

В используемых в настоящее время реализациях протокола HTTP первая цифра не может быть больше 5 и определяет следующие классы ответов.

1 - специальный класс сообщений, называемых информационными. Код ответа, начинающийся с 1, означает, что сервер продолжает обработку запроса. При обмене данными между HTTP-клиентом и HTTP-сервером сообщения этого класса используются достаточно редко.

2 - успешная обработка запроса клиента.

3 - перенаправление запроса. Чтобы запрос был обслужен, необходимо предпринять дополнительные действия.

4 - ошибка клиента. Как правило, код ответа, начинающийся с цифры 4, возвращается в том случае, если в запросе клиента встретилась синтаксическая ошибка.

5 - ошибка сервера. По тем или иным причинам сервер не в состоянии выполнить запрос.

Примеры кодов ответов, которые клиент может получить от сервера, и поясняющие сообщения приведены в таблице 2.2.

Таблица 2.2 - Классы кодов ответа сервера

|  |  |  |
| --- | --- | --- |
| Код | Расшифровка | Интерпретация |
| 100 | Continue | Часть запроса принята, и сервер ожидает от клиента продолжения запроса |

Продолжение таблицы 2.2

|  |  |  |
| --- | --- | --- |
| 200 | OK | Запрос успешно обработан, и в ответе клиенту передаются данные, указанные в запросе |
| 201 | Created | В результате обработки запроса был создан новый ресурс |
| 202 | Accepted | Запрос принят сервером, но обработка его не окончена. Данный код ответа не гарантирует, что запрос будет обработан без ошибок. |
| 206 | Partial Content | Сервер возвращает часть ресурса в ответ на запрос, содержавший поле заголовка Range |
| 301 | Multiple Choice | Запрос указывает более чем на один ресурс. В теле ответа могут содержаться указания на то, как правильно идентифицировать запрашиваемый ресурс |
| 302 | Moved Permanently | Затребованный ресурс больше не располагается на сервере |
| 302 | Moved Temporarily | Затребованный ресурс временно изменил свой адрес |
| 400 | Bad Request | В запросе клиента обнаружена синтаксическая ошибка |
| 403 | Forbidden | Имеющийся на сервере ресурс недоступен для данного пользователя |
| 404 | Not Found | Ресурс, указанный клиентом, на сервере отсутствует |
| 405 | Method Not Allowed | Сервер не поддерживает метод, указанный в запросе |
| 500 | Internal Server Error | Один из компонентов сервера работает некорректно |
| 501 | Not Implemented | Функциональных возможностей сервера недостаточно, чтобы выполнить запрос клиента |
| 503 | Service Unavailable | Служба временно недоступна |
| 505 | HTTP Version not Supported | Версия HTTP, указанная в запросе, не поддерживается сервером |

В ответе используется такая же структура полей заголовка, как и в запросе клиента. Поля заголовка предназначены для того, чтобы уточнить ответ сервера клиенту. Описание некоторых из полей, которые можно встретить в заголовке ответа сервера, приведено в таблице 2.3.

Таблица 2.3 - Поля заголовка ответа веб-сервера

|  |  |
| --- | --- |
| Имя поля | Описание содержимого |
| Server | Имя и номер версии сервера |
| Age | Время в секундах, прошедшее с момента создания ресурса |
| Allow | Список методов, допустимых для данного ресурса |
| Content-Language | Языки, которые должен поддерживать клиент для того, чтобы корректно отобразить передаваемый ресурс |
| Content-Type | MIME-тип данных, содержащихся в теле ответа сервера |
| Content-Length | Число символов, содержащихся в теле ответа сервера |
| Last-Modified | Дата и время последнего изменения ресурса |
| Date | Дата и время, определяющие момент генерации ответа |
| Expires | Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей |
| Location | В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса |
| Cache-Control | Директивы управления кэшированием. Например, no - cache означает, что данные не должны кэшироваться |

В теле ответа содержится код ресурса, передаваемого клиенту в ответ на запрос. Это не обязательно должен быть HTML-текст веб-страницы. В составе ответа могут передаваться изображение, аудио-файл, фрагмент видеоинформации, а также любой другой тип данных, поддерживаемых клиентом. О том, как следует обрабатывать полученный ресурс, клиенту сообщает содержимое поля заголовка Content - type.

Ниже представлен пример ответа сервера на запрос, приведенный в предыдущем разделе. В теле ответа содержится исходный текст HTML-документа.

HTTP/1.1 200 OK

Server: Microsoft-IIS/5.1

X-Powered-By: ASP.NET

Date: Mon, 20 OCT 2008 11:25:56 GMT

Content-Type: text/html

Accept-Ranges: bytes

Last-Modified: Sat, 18 Oct 2008 15:05:44 GMT

ETag: "b66a667f948c92:8a5"

Content-Length: 426

<html>

<body>

<form action='http://localhost/Scripts/test.pl'>

<p>Operand1: <input type='text' name='A'></p>

<p>Operand2: <input type='text' name='B'></p>

<p>Operation:<br>

<select name='op'>

<option value='+'>+</option>

<option value='-'>-</option>

<option value='\*'>\*</option>

<option value='/'>/</option>

<select></p>

<input type='submit' value='Calculate!'>

</from>

</body>

</html>

Поля заголовка и тело сообщения могут отсутствовать, но строка состояния является обязательным элементом, так как указывает на тип запроса/ответа.

Поле с именем Content-type может встречаться как в запросе клиента, так и в ответе сервера. В качестве значения этого поля указывается MIME -тип содержимого запроса или ответа. MIME -тип также передается в поле заголовка Accept, присутствующего в запросе.

Спецификация MIME (Multipurpose Internet Mail Extension — многоцелевое почтовое расширение Internet) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем. Однако применение MIME не исчерпывается электронной почтой. Средства MIME успешно используются в WWW и, по сути, стали неотъемлемой частью этой системы.

Стандарт MIME разработан как расширяемая спецификация, в которой подразумевается, что число типов данных будет расти по мере развития форм представления данных. Каждый новый тип в обязательном порядке должен быть зарегистрирован в IANA (Internet Assigned Numbers Authority).

До появления MIME компьютеры, взаимодействующие по протоколу HTTP, обменивались исключительно текстовой информацией. Для передачи изображений, как и для передачи любых других двоичных файлов, приходилось пользоваться протоколом FTP.

В соответствии со спецификацией MIME, для описания формата данных используются тип и подтип. Тип определяет, к какому классу относится формат содержимого HTTP-запроса или HTTP-ответа. Подтип уточняет формат. Тип и подтип отделяются друг от друга косой чертой:

тип/подтип

Поскольку в подавляющем большинстве случаев в ответ на запрос клиента сервер возвращает исходный текст HTML-документа, то в поле Content-type ответа обычно содержится значение text/html. Здесь идентификатор text описывает тип, сообщая, что клиенту передается символьная информация, а идентификатор html описывает подтип, т.е. указывает на то, что последовательность символов, содержащаяся в теле ответа, представляет собой описание документа на языке HTML.

Перечень типов и подтипов MIME достаточно велик. В таблице 2.4 приведены примеры MIME -типов, наиболее часто встречающиеся в заголовках HTML-запросов и ответов.

Таблица 2.4 - MIME типы данных.

|  |  |  |
| --- | --- | --- |
| Тип/подтип | Расширение файла | Описание |
| application/pdf | .pdf | Документ, предназначенный для обработки Acrobat Reader |
| application/msexcel | .xls | Документ в формате Microsoft Excel |
| application/postscript | .ps, .eps | Документ в формате PostScript |
| application/x-tex | .tex | Документ в формате ТеХ |
| application/msword | .doc | Документ в формате Microsoft Word |
| application/rtf | .rtf | Документ в формате RTF, отображаемый с помощью Microsoft Word |
| image/gif | .gif | Изображение в формате GIF |
| image/ jpeg | .jpeg, .jpg, | Изображение в формате JPEG |
| image/tiff | .tiff, .tif | Изображение в формате TIFF |
| image/x-xbitmap | .xbm | Изображение в формате XBitmap |
| text/plain | .txt | ASCII-текст |
| text/html | . html , . htm | Документ в формате HTML |
| audio/midi | .midi, .mid | Аудиофайл в формате MIDI |
| audio/x-wav | .wav | Аудиофайл в формате WAV |
| message/rfc822 |  | Почтовое сообщение |
| message/news |  | Сообщение в группы новостей |
| video /mpeg | .mpeg, .mpg, .mpe | Видеофрагмент в формате MPEG |
| video/avi | .avi | Видеофрагмент в формате AVI |

Для однозначной идентификации ресурсов в сети Веб используются уникальные идентификаторы URL.

Единообразный идентификатор ресурса URI (Uniform Resource Identifier) представляет собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс. URI не указывает на то, как получить ресурс, а только идентифицирует его. Это дает возможность описывать с помощью RDF (Resource Description Framework) ресурсы, которые не могут быть получены через Интернет (имена, названия и т.п.). Самые известные примеры URI - это URL и URN.

URL (Uniform Resource Locator) - это URI, который, помимо идентификации ресурса, предоставляет еще и информацию о местонахождении этого ресурса.

URN (Uniform Resource Name) - это URI, который идентифицирует ресурс в определенном пространстве имен, но, в отличие от URL, URN не указывает на местонахождение этого ресурса.

URL имеет следующую структуру:

<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>

где:

* схема - схема обращения к ресурсу (обычно сетевой протокол);
* логин - имя пользователя, используемое для доступа к ресурсу;
* пароль - пароль, ассоциированный с указанным именем пользователя;
* хост - полностью прописанное доменное имя хоста в системе DNS или IP-адрес хоста;
* порт - порт хоста для подключения;
* URL-путь - уточняющая информация о месте нахождения ресурса.

Общепринятые схемы (протоколы) URL включают протоколы: ftp, http, https, telnet, а также:

* gopher — протокол Gopher;
* mailto — адрес электронной почты;
* news — новости Usenet;
* nntp — новости Usenet через протокол NNTP;
* irc — протокол IRC;
* prospero — служба каталогов Prospero Directory Service;
* wais — база данных системы WAIS;
* xmpp — протокол XMPP (часть Jabber);
* file — имя локального файла;
* data — непосредственные данные (Data: URL);

***Обеспечение безопасности передачи данных http***

Поскольку протокол HTTP предназначен для передачи символьных данных в открытом (незашифрованном) виде, то лица, имеющие доступ к каналу передачи данных между клиентом и сервером, могут без труда просматривать весь трафик и использовать его для совершения несанкционированных действий. В связи с этим предложен ряд расширений базового протокола направленных на повышение защищенности интернет-трафика от несанкционированного доступа.

Самым простейшим является расширение HTTPS, при котором данные, передаваемые по протоколу HTTP, "упаковываются" в криптографический протокол SSL или TLS, тем самым обеспечивая защиту этих данных. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443. Чтобы подготовить веб-сервер для обработки HTTPS соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера.

SSL (Secure Sockets Layer) - криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет. При его использовании создается защищенное соединение между клиентом и сервером. SSL изначально разработан компанией Netscape Communications. Впоследствии на основании протокола SSL 3.0 был разработан и принят стандарт RFC, получивший название TLS. Этот протокол использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надежность передачи данных за счет использования корректирующих кодов и безопасных хэш-функций. На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции различных протоколов (POP3, IMAP, SMTP или HTTP). Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнет передавать и получать данные.

Для доступа к веб-страницам, защищенным протоколом SSL, в URL вместо схемы http, как правило, подставляется схема https, указывающая на то, что будет использоваться SSL-соединение. Стандартный TCP-порт для соединения по протоколу https — 443. Для работы SSL требуется, чтобы на сервере имелся SSL -сертификат.

В сети Веб поддерживаются 3 типа аутентификации при клиент-серверных взаимодействиях:

Basic - базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках http-пакетов. Пароль при этом не шифруется и присутствует в чистом виде в кодировке base64. Для данного типа аутентификации использование SSL является обязательным.

Digest - дайджест-аутентификация, при которой пароль пользователя передается в хешированном виде. По уровню конфиденциальности паролей этот тип мало чем отличается от предыдущего, так как атакующему все равно, действительно ли это настоящий пароль или только хеш от него: перехватив удостоверение, он все равно получает доступ к конечной точке. Для данного типа аутентификации использование SSL является обязательным.

Integrated - интегрированная аутентификация, при которой клиент и сервер обмениваются сообщениями для выяснения подлинности друг друга с помощью протоколов NTLM или Kerberos. Этот тип аутентификации защищен от перехвата удостоверений пользователей, поэтому для него не требуется протокол SSL. Только при использовании данного типа аутентификации можно работать по схеме http, во всех остальных случаях необходимо использовать схему https.

***Cookie***

Поскольку HTTP-сервер не помнит предыстории запросов клиентов, то каждый запрос обрабатывается независимо от других, и у сервера нет возможности определить, исходят ли запросы от одного клиента или разных клиентов.

Если сервер будет проверять TCP-соединения и запоминать IP-адреса компьютеров-клиентов, он все равно не сможет различить запросы от двух браузеров, выполняющихся на одной машине. И даже если допустить, что на компьютере работает лишь одна клиент-программа, то никто не может утверждать, что в промежутке между двумя запросами она не была завершена, а затем запущена снова уже другим пользователем.

Тем не менее, если вы когда-нибудь пользовались почтовым ящиком на mail.ru или на другом сервере, предоставляющем почтовые услуги пользователям Веб, вспомните, как вел себя клиент после того, как вы создали для себя почтовый ящик на сервере. Когда вы в следующий раз обратились с того же компьютера к mail.ru, вы, вероятно, заметили, что после загрузки веб-страницы ваше регистрационное имя уже отображалось в соответствующем поле ввода.

Такие сведения позволяет получить дополнительное средство под названием cookie. Механизм cookie позволяет серверу хранить информацию на компьютере клиента и извлекать ее оттуда.

Инициатором записи cookie выступает сервер. Если в ответе сервера присутствует поле заголовка Set-cookie, клиент воспринимает это как команду на запись cookie. В дальнейшем, если клиент обращается к серверу, от которого он ранее принял поле заголовка Set-cookie, помимо прочей информации он передает серверу данные cookie. Для передачи указанной информации серверу используется поле заголовка Cookie.

Для того чтобы в общих чертах представить себе, как происходит обмен данными cookie, рассмотрим следующий пример. Предположим, что клиент передает запросы на серверы А, В и С. Предположим также, что сервер В, в отличие от А и С, передает клиенту команду записать cookie. Последовательность запросов клиента серверу и ответов на них будет выглядеть приблизительно следующим образом.

Передача запроса серверу А.

Получение ответа от сервера А.

Передача запроса серверу В.

Получение ответа от сервера В. В состав ответа входит поле заголовка SetCookie. Получив его, клиент записывает cookie на диск.

Передача запроса серверу С. Несмотря на то что на диске хранится запись cookie, клиент не предпринимает никаких специальных действий, так как значение cookie было записано по инициативе другого сервера.

Получение ответа от сервера С.

Передача запроса серверу А. В этом случае клиент также никак не реагирует на тот факт, что на диске хранится cookie.

Получение ответа от сервера А.

Передача запроса серверу В. Перед тем как сформировать запрос, клиент определяет, что на диске хранится запись cookie, созданная после получения ответа от сервера В. Клиент проверяет, удовлетворяет ли данный запрос некоторым требованиям, и, если проверка дает положительный результат, включает в заголовок запроса поле Cookie.

Таким образом, процедуру записи и получения cookie можно представить себе как своеобразный "запрос" сервера, инкапсулированный в его ответе клиенту. Соответственно, получение cookie также можно представить себе как ответ клиента, инкапсулированный в составе запроса тому же серверу.

Рассмотрим подробнее, какие данные передаются в поле заголовка Set-cookie и как они влияют на поведение клиента.

Поле Set-cookie имеет следующий формат:

Set-cookie: имя = значение; expires = дата; path = путь; domain = имя\_домена, secure

где

Пара имя = значение – именованные данные, сохраняемые с помощью механизма cookie. Эти данные должны храниться на клиент-машине и передаваться серверу в составе очередного запроса клиента.

Дата, являющаяся значением параметра expires, определяет время, по истечении которого информация cookie теряет свою актуальность. Если ключевое слово expires отсутствует, данные cookie нe удаляются по окончании текущего сеанса работы браузера.

Значение параметра domain определяет домен, с которым связываются данные cookie. Чтобы узнать, следует ли передавать в составе запроса данные cookie, браузер сравнивает доменное имя сервера, к которому он собирается обратиться, с доменами, которые связаны с записями cookie, хранящимися на клиент-машине. Результат проверки будет считаться положительным, если сервер, которому направляется запрос, принадлежит домену, связанному с cookie. Если соответствие не обнаружено, данные cookie не передаются.

Путь, указанный в качестве значения параметра path, позволяет выполнить дальнейшую проверку и принять окончательное решение о том, следует ли передавать данные cookie в составе запроса. Помимо домена с записью cookie связывается путь. Если браузер обнаружил соответствие имени домена значению параметра domain, он проверяет, соответствует ли путь к ресурсу пути, связанному с cookie. Сравнение считается успешным, если ресурс содержится в каталоге, указанном посредством ключевого слова path, или в одном из его подкаталогов. Если и эта проверка дает положительный результат, данные cookie передаются серверу. Если параметр path в поле Set-Cookie отсутствует, то считается, что запись cookie связана с URL конкретного ресурса, передаваемого сервером клиенту.

Последний параметр, secure, указывает на то, что данные cookie должны передаваться по защищенному каналу.

Для передачи данных cookie серверу используется поле заголовка Cookie. Формат этого поля достаточно простой:

Cookie: имя=значение; имя=значение; ...

C помощью поля Cookie передается одна или несколько пар имя = значение. Каждая из этих пар принадлежит записи cookie, для которой URL запрашиваемого ресурса соответствуют имени домена и пути, указанным ранее в поле Set-cookie.

**Тема. Клиентские сценарии и приложения**

Как правило, Веб-приложение - приложение, в котором клиентом выступает браузер, а сервером - веб-сервер.

Рассмотрим типы программ, обеспечивающих работу Веб и использующих HTTP-протокол.

Никакой HTTP-обмен невозможен без клиента и сервера. Однако помимо клиента и сервера в веб-сеансе могут участвовать и другие программы, которые и являются объектом веб-программирования.

Результатом работы веб-приложения является веб-страница, отображаемая в окне браузера. При этом само веб-приложение может выполняться как на компьютере клиента, так и на компьютере сервера.

Рассмотрим подробнее обе схемы.

Программы, выполняющиеся на клиент-машине

Одним из типов программ, предназначенных для выполнения на клиент-машине, являются сценарии, например, JavaScript (VBScript). Исходный текст сценария представляет собой часть веб-страницы, поэтому сценарий JavaScript передается клиенту вместе с документом, в состав которого он входит. Обрабатывая HTML-документ, браузер обнаруживает исходный текст сценария и запускает его на выполнение.

Ко всем программам, которые передаются с сервера на клиент-машины и запускаются на выполнение, предъявляется одно общее требование: эти программы должны быть лишены возможности обращаться к ресурсам компьютера, на котором они выполняются. Такое требование вполне обосновано. Ведь передача по сети и запуск Java-апплетов и JavaScript-сценариев происходит автоматически без участия пользователя, поэтому работа этих программ должна быть абсолютно безопасной для компьютера. Другими словами, языки, предназначенные для создания программ, выполняющихся на клиент-машине, должны быть абсолютно непригодны для написания вирусов и подобных программ.

Программы, выполняющиеся на сервере

Код программы, работающей на сервере, не передается клиенту. При получении от клиента специального запроса, предполагающего выполнение такой программы, сервер запускает ее и передает параметры, входящие в состав запроса. Средства для генерации подобного запроса обычно входят в состав HTML-документа.

Результаты своей работы программа оформляет в виде HTML-документа и передает их веб-серверу, а последний, в свою очередь, дополняет полученные данные HTTP-заголовком и передает их клиенту. Взаимодействие клиента и сервера в этом случае показано на рисунке 5.1.

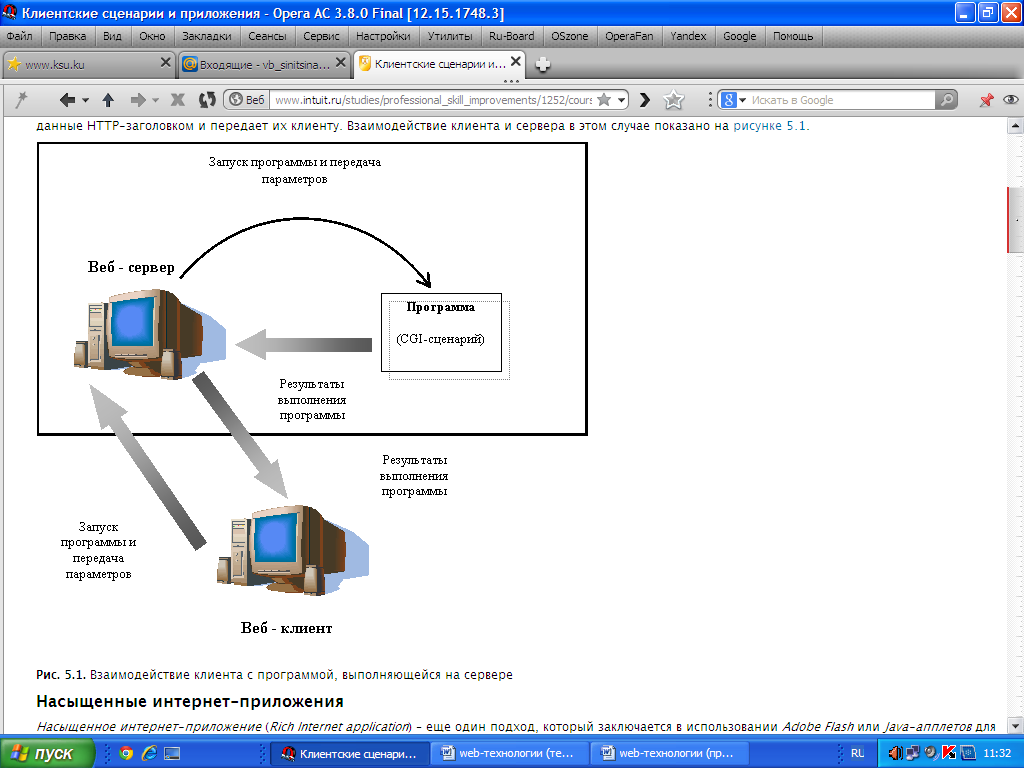


Рисунок 3.1 - Взаимодействие клиента с программой, выполняющейся на сервере

***Насыщенные интернет-приложения***

Насыщенное интернет-приложение (Rich Internet application) – еще один подход, который заключается в использовании Adobe Flash или Java-апплетов для полной или частичной реализации пользовательского интерфейса, поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов).

Возникновение данного подхода обусловлено тем, что в рамках веб-приложений с "тонким" клиентом взаимодействие пользователя с приложением реализуется в существенной степени через сервер, что требует отправки данных на сервер, получение ответа от сервера и перезагрузку страницы на стороне клиента.

При использовании Java-апплетов в состав HTML-документа включается специальный дескриптор, описывающий расположение файла, содержащего код апплета, на сервере. После того как клиент получает HTML-код документа, включающего апплет, он генерирует дополнительный запрос серверу. После того как сервер пересылает клиенту код апплета, сам апплет запускается на выполнение. Взаимодействие между клиентом и сервером при получении апплета показано на рисунке 5.2.

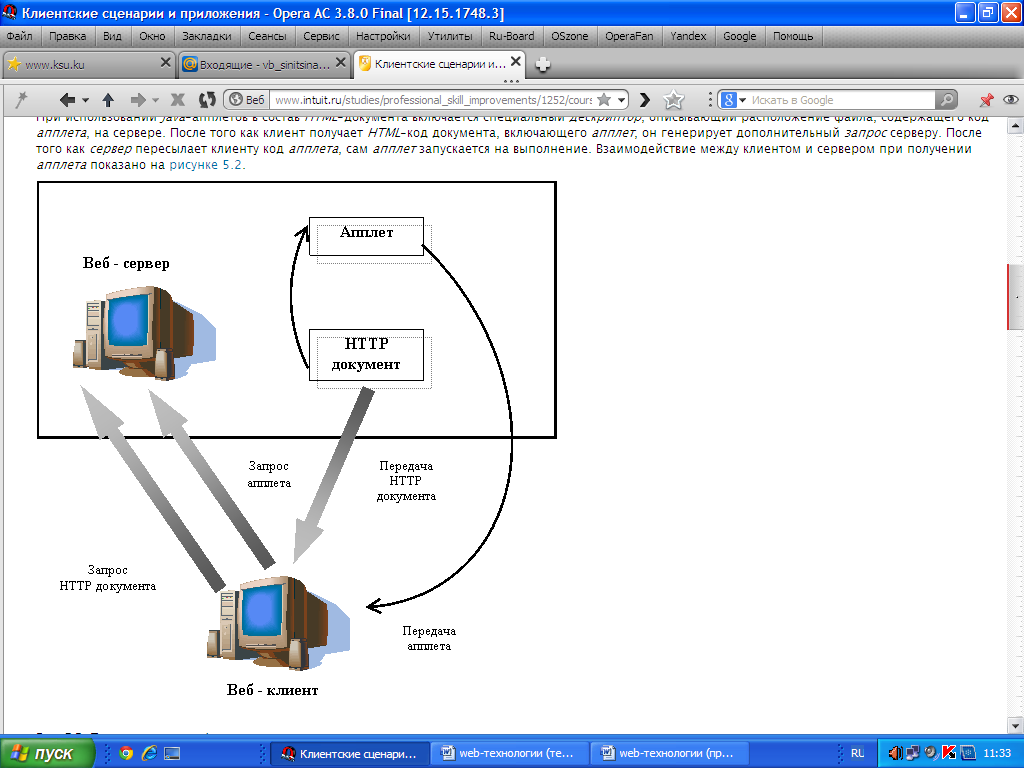


Рисунок 3.2 - Передача клиенту Java-апплета

При использовании насыщенных интернет-приложений приходится сталкиваться со следующими проблемами:

* необходимость обеспечения безопасной среды выполнения ("песочница");
* для исполнения кода должно быть разрешено исполнение сценариев;
* потеря в производительности (т.к. выполяется на клиентской стороне);
* требуется много времени на загрузку;

Для разработки насыщенных интернет-приложений используются пакеты Curl, Adobe Flex и Microsoft Silverlight.

***Введение в JScript***

JavaScript - интерпретируемый язык программирования, стандартизированный международной организацией ECMA в спецификации ECMA-262. Языки JavaScript, JScript и ActionScript являются расширением стандарта ECMA-262.

Название "ECMAScript" явилось фактически компромиссом между организациями, вовлеченными в процесс стандартизации, в частности Netscape и Microsoft. Хотя JavaScript и JScript стремились к совместимости с ECMAScript, они имеют ряд дополнительных возможностей не предусмотренных спецификацией ECMA.

Синтаксис JScript во многом аналогичен языку JavaScript, однако, помимо добавления клиентских скриптов на веб-страницы и некоторых других функций, JScript может использоваться и для других целей, например:

* автоматизация администрирования систем Microsoft Windows;
* создание страниц ASP.

Язык JScript получил дальнейшее развитие в виде языка JScript.NET, который ориентирован на работу в рамках платформы Microsoft.NET

JScript - интерпретируемый, объектно-ориентированный язык. Хотя он имеет существенно меньшее количество возможностей, чем такие объектно-ориентированные языки как C++ и Java.

Возможности языка существенно ограничены:

* язык не позволяет разрабатывать самостоятельные приложения;
* сценарии на JScript могут выполняться только при помощи интерпретатора, в частности веб-браузером.

JScript - язык без строгого контроля типов. Поэтому не требуется объявлять тип переменных явно. Кроме того, во многих случаях JScript исполняет преобразования автоматически, когда они необходимы. Например, при сложении строки и числа, число будет преобразовано в строку.

Код на JScript пишется в текстовом формате, и организован в инструкции, блоки, состоящие из связанных наборов инструкций, и комментариев. В пределах инструкции можно использовать переменные и данные, такие как строки, числа и выражения. Для объявления конца инструкции используется точка с запятой ;. Группа JScript-инструкций, заключенная в фигурные скобки {}, называется блоком.

Комментарием в JScript является текст, расположенный после // до конца строки. Многострочный комментарий начинается с /\*, и заканчивается \*/.

Знак равенства ( = ) используется в JScript как присваивание. Следующий код

Pi = 3.14;

подразумевает "Присвоить значение 3.14 переменной Pi ".

При сравнении двух значений на равенство применяется двойной знак равенства ( == ).

JScript выражения можно разделить на логические или числовые. Выражения содержат некоторые особенности, к примеру, символ " + " означает "добавить к...". Любая допустимая комбинация значений, переменных, операторов, и других выражений является выражением.

Объявление переменной перед использованием является необязательным. Для этого используется инструкция var. Инструкция var является обязательной при объявлении локальной переменной внутри функции. Разрешается объявление переменной неявно - без инструкции var. Однако, в выражениях применять необъявленные переменные не допускается. JScript различает регистр в имени переменной. Name и name рассматриваются как различные переменные.

***Типы данных***

JScript - язык с нестрогим контролем типов, переменные в JScript не имеют строго фиксированного типа. Переменные имеют тип, эквивалентный типу значения, которое они содержат. Однако, в некоторых случаях, необходимо принудительное преобразование переменной в определенный тип. Числа могут быть объявлены как строки, а строки необходимо преобразовать в числовой тип. Для этого применяют функции parseInt() и parseFloat().

В JScript используется шесть типов данных. Основные из них - числа, строки, объекты, логический. Остальные два - null и undefined (т.е. неопределенный).

Строки объявляются при помощи двойных кавычек или апострофов. Строка может состоять из нуля или более символов unicode. Когда количество символов равно нулю, это называется пустой строкой ( "" ).

JScript поддерживает числа как целые, так и с плавающей запятой. Также существуют специальные представления чисел, например NaN (не число).

Примеры чисел:

3.14 // Вещественное число

26 // Целое число

0177 // Восьмеричное число

0XA8 // Шестнадцатиричное число

Логический тип допускает значения - true и false. Любое выражение, равное 0, считается эквивалентным false, а любое выражение, равное числу, отличному от 0 будет эквивалентным true.

Undefined – означает, что тип не определен. Значение undefined имеет переменная после ее объявления и до присвоения ей какого-либо определенного значения.

Переменная типа null - не имеет никакого определенного значения.

***Операторы***

Язык поддерживает условные выражения if и if...else. При использовании нескольких условий одновременно можно использовать операторы || (ИЛИ ) или && (И).

В JScript поддерживается несколько типов циклов: for, for...in, while, do...while и switch. Также существует инструкция остановки выполнения цикла. Оператор завершения break может использоваться, чтобы остановить цикл, при выполнении какого-либо условия. Инструкция continue используется, чтобы немедленно перейти к выполнению следующей итерации, пропуская остальную часть выполнения кода текущей итерации, но обновляя переменную-счетчик.

***Функции и объекты***

В JScript имеется два вида функций: встроенные и определяемые. Программист имеет возможность создавать собственные функции. Определение функции состоит из объявления параметров и блока инструкций JScript.

Объекты в JScript, по-сути, являются совокупностями методов и свойств. Все объекты можно разделить на три вида: встроенные, созданные и браузерные. Обработка объектов и массивов идентична. Можно обратиться к любой части объекта (его свойствам и методам) либо по имени, либо по индексу. Нумерация индексов в JScript начинается с нуля.

Краткая характеристика VBScript

Visual Basic Scripting Edition (обычно просто VBScript) — сценарный язык программирования, интерпретируемый компонентом Windows Script Host. Он широко используется при создании скриптов в операционных системах семейства Microsoft Windows.

Язык был создан компанией Microsoft как замена устаревшему пакетному языку, интерпретируемому приложением command.com. Синтаксис VBScript является упрощенной версией синтаксиса языка Visual Basic.

Сценарии на языке VBScript чаще всего используются в следующих областях, использующих программные продукты Microsoft:

* автоматизация администрирования систем Windows;
* серверный программный код в страницах ASP;
* клиентские сценарии в браузере Internet Explorer.
* Java-апплеты

Java-апплет - это программа, написанная на языке Java и откомпилированная в байт-код. Выполнется в браузере с использованием виртуальной Java-машины ( JVM ). Апплеты используются для предоставления интерактивных возможностей веб-приложений, которые не возможны в HTML. Так как байт-код Java платформо-независим, то Java-апплеты могут выполняться браузерами на многих операционных платформах.

Java-сервлеты являются серверными приложениями, но они отличаются от апплетов языком, функциями и другими характеристиками.

Предназначены Java-апплеты для выполнения в безопасной среде с целью предотвращения их доступа к локальным ресурсам клиентского компьютера.

Код апплета загружается с веб-сервера, и браузер

* либо вставляет апплет в веб-страницу;
* либо открывает отдельное окно с собственным пользовательским интерфейсом апплета.

Апплет может быть внедрен в веб-страницу с помощью использования HTML тэга <applet>, или (что рекомендуется) тэга <object>.

Можно назвать следующие преимущества Java-апплетов:

* работают практически на большинстве операционных платформ;
* поддерживаются большинством браузеров;
* кэшируются в большинстве браузеров, что существенно ускоряет их загрузку при возвращении на веб-страницу;
* после первого запуска апплета, когда Java-машина уже выполняется и быстро запускается, выполнение апплетов происходит существенно быстрее;
* загружаются со скоростью, сопоставимой с програмами на других компилируемых языках, например C++, но во много раз быстрее чем на JavaScript.

При этом у Java-апплетов имеются и недостатки:

* требуется установка Java-расширения, которые доступны по умолчанию не во всех браузерах;
* проблемы реализации Java-расширений для 64-разрядных процессоров;
* не могут запускаться до первой загрузки виртуальной Java-машины, что может занимать значительное время;
* разработка пользовательского интерфейса с использованием апплетов является более сложной задачей по сравнению с HTML;
* не имеют прямого доступа к локальным ресурсам клиентского компьютера;
* некоторые апплеты привязаны к использованию определенной среды времени выполнения Java ( JRE ).

ActionScript – общая характеристика

ActionScript — объектно-ориентированный язык программирования, один из диалектов EcmaScript, который добавляет интерактивность, обработку данных и многое другое в содержимое Flash-приложений. ActionScript исполняется виртуальной машиной (ActionScript Virtual Machine), которая является составной частью приложения Flash Player. ActionScript компилируется в байткод, который включается в SWF -файл.

SWF -файлы исполняются Flash Player. Сам Flash Player существует в виде плагина к веб-браузеру, а также как самостоятельное исполняемое приложение. Во втором случае возможно создание исполняемых exe -файлов, когда swf -файл включается во Flash Player.

С помощью ActionScript можно создавать интерактивные мультимедиа-приложения, игры, веб-сайты и многое другое.

***XAML и Microsoft Silverlight***

XAML (eXtensible Application Markup Language) - язык интерфейсов платформы Windows Vista.

Модель приложений Vista включает объект Application. Его набор свойств, методов и событий позволяет объединять веб-документы в связанное приложение. Объект Application контролирует выполнение программы и генерирует события для пользовательского кода. Документы приложения создаются с помощью языка XAML, который описывает, прежде всего, пользовательский интерфейс. Логика приложения управляется процедурным кодом (С#, VB и др.). XAML включает основные четыре категории элементов: панели, элементы управления, элементы, связанные с документом и графические фигуры.

Microsoft Silverlight является официальным названием основанной на XML и .NET технологии под кодовым именем WPF/E (Windows Presentation Foundation Everywhere), являющейся альтернативной Adobe Flash. Представляет собой подмножество Windows Presentation Foundation, в котором реализованы векторная графика, анимация и средства воспроизведения видео. В версии 1.1 включает в себя полную версию .NET CLR - называемую CoreCLR, что позволит разрабатывать Silverlight приложения на любом из языков .NET. Silverlight v.1.0 содержит подключаемый модуль браузера для обработки XAML и кодеки для воспроизведения мультимедийного содержимого в форматах WMV, WMA и MP3.

Microsoft Silverlight представляет браузеру внутреннюю модель DOM, управляемую из JavaScript кода. Поскольку язык XAML основан на XML, то документ, определяющий загружаемый клиенту пользовательский интерфейс - текстовый и потому вполне пригоден для индексирования поисковыми системами. Используя модель DOM, JavaScript может динамически обновлять содержимое Silverlight, аналогично DHTML.

Также можно вызывать методы управления презентацией (запуска анимации или приостановки воспроизведения видео, например).

Silverlight -приложение начинается с вызова объекта Silverlight из HTML страницы, загружающего XAML файл. XAML файл содержит объект Canvas, выступающий подложкой для других элементов. Объекты XAML способны генерировать события, перехватываемые из JavaScript.

***Понятие о DOM***

DOM (Document Object Model) - объектная модель документа. Это независящий от платформы и языка программный интерфейс, позволяющий программам получать доступ к содержимому документов, а также изменять содержимое, структуру и вид документов.

В рамках DOM любой документ представляется в виде дерева узлов. Каждый узел представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы между собой находятся в отношении "родитель-потомок".

Изначально различные браузеры имели собственные модели DOM, не совместимые с остальными. Для того, чтобы обеспечить взаимную и обратную совместимость, консорциум W3C классифицировал эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM:

Уровень 0. Включает в себя все специфические модели DOM, которые существовали до появления Уровня 1, например document.images, document.forms. Эти модели формально не являются спецификациями DOM, опубликованными W3C, а скорее отражают то, что существовало до начала процесса стандартизации.

* Уровень 1. Базовые функциональные возможности DOM (HTML и XML) в документах, такие как получение дерева узлов документа, возможность изменять и добавлять данные.
* Уровень 2. Поддержка пространства имен XML, filtered views и событий.
* Уровень 3. Состоит из шести различных спецификаций:
* DOM Level 3 Core ;
* DOM Level 3 Load and Save ;
* DOM Level 3 XPath ;
* DOM Level 3 Views and Formatting ;
* Level 3 Requirements ;
* DOM Level 3 Validation.

Текущим уровнем спецификаций DOM является Уровень 2, но, тем не менее, некоторые части спецификаций Уровня 3 являются рекомендуемыми W3C.

***DHTML***

Динамический HTML или DHTML представляет собой набор технологий, которые совместно позволяют создавать интерактивные веб-сайты на основе статического языка разметки ( HTML ), языка создания клиентских сценариев ( JavaScript ), языка описания представления документа ( CSS ) и документной объектной модели ( DOM ).

DHTML позволяет сценарным языкам изменять переменные языка описания представления документа, таким образом, изменяя вид и поведение прежде статического содержимого HTML документа уже после полной загрузки документа и в процессе просмотра его пользователем. Таким образом, динамичность, привносимая DHTML, проявляет себя в процессе просмотра страницы, но не имеет никакого отношения к генерации содержимого страницы при каждой ее загрузке.

В противоположность DHTML, динамически генерируемая страница - более широкое понятие, подразумевающее, например генерацию содержимого веб-страницы индивидуально для каждого пользователя. Это достигается созданием страниц с помощью клиентских или серверных (например, на PHP или Perl) сценариев.

***Регулярные выражения***

Регулярные выражения — система поиска текстовых фрагментов в электронных документах, основанная на специальной системе записи образцов для поиска. Образец, задающий правило поиска, называется " шаблоном ". Применение регулярных выражений принципиально преобразило технологии электронной обработки текстов.

Многие языки программирования уже поддерживают регулярные выражения для работы со строками либо в виде отдельных функций, либо имеют уже встроенный в их синтаксис механизм обработки регулярных выражений, например, Perl и Tcl. Популяризации понятия регулярных выражений способствовали утилиты, поставляемые в дистрибутивах Unix.

С помощью регулярных выражений можно задавать структуру искомого шаблона и его позицию внутри строки (например, в начале или в конце строки, на границе или не на границе слова).

При описании структуры шаблона используются:

гибкая система квантификаторов (операторов повторения);

операторы описания наборов символов и их типа (числовые, нечисловые, специальные).

**Тема. Серверные веб-приложения**

Для расширения возможностей клиент-серверного взаимодействия в рамках протокола HTTP помимо создания на клиентской стороне расширений стандартных возможностей, предоставляемых языками разметки и браузерами, можно также разрабатывать на стороне веб-сервера приложения, плагины и сценарии, расширяющие возможности самого веб-сервера.

Плагин (plug - in) - независимо компилируемый программный модуль, динамически подключаемый к основной программе, предназначенный для расширения или использования ее возможностей. Обычно выполняются в виде разделяемых библиотек.

Сценарий (скрипт, script) - программа, которая автоматизирует некоторую задачу, которую пользователь выполняет вручную, используя интерфейсы программы.

***Стандарт CGI***

Круг задач, решаемых Web-сервером, ограничен. В основном он сводится к поддержке НТТР-взаимодействия и доставке клиенту Web-документов. Любые "нестандартные" действия реализуются с помощью специальной программы, которая взаимодействует с веб-сервером и клиентом. Это взаимодействие подчиняется определенным правилам.

Основной набор таких правил - стандарт CGI (Common Gateway Interface - интерфейс общего шлюза), который определяет порядок запуска программы на компьютере-сервере, способы передачи программе параметров и доставки результатов ее выполнения клиенту. Программа, написанная по правилам CGI, называется CGI-сценарием (script CGI), хотя это не означает, что на сервере не может выполняться двоичный файл.

Благодаря этому интерфейсу для разработки приложений можно использовать любой язык программирования, который располагает средствами взаимодействия со стандартными устройствами ввода/вывода. Такими возможностями обладают также сценарии для встроенных командных интерпретаторов операционных систем.

Выполнение любой программы (в том числе CGI-сценария) можно условно разделить на пять этапов.

* Запуск программы.
* Инициализация и чтение выходных данных.
* Обработка данных.
* Вывод результатов выполнения.
* Завершение программы.

Различия между CGI-сценарием и консольным приложением касаются первого, второго и четвертого этапов выполнения.

Каждый раз, когда веб-сервер получает запрос от клиента, он анализирует содержимое запроса и возвращает соответствующий ответ:

Если запрос содержит указание на файл, находящийся на жестком диске, то сервер возвращает в составе ответа этот файл ;

Если запрос содержит указание на программу и необходимые для нее аргументы, то сервер исполняет программу и результат ее работы возвращает клиенту.

CGI определяет:

* каким образом информация о сервере и запросе клиента передается программе в форме аргументов и переменных окружения;
* каким образом программа может передавать назад дополнительную информацию о результатах (например, о типе данных) в форме заголовков ответа сервера.

В подавляющем большинстве случаев запуск CGI-сценария осуществляется щелчком на кнопке Submit, сформированной с помощью дескриптора <input tyре = "submit">, который находится на HTML-странице между <form> и </form>. Не зная назначения атрибутов action и method, невозможно понять, как происходит вызов программы и передача параметров.

Значением атрибута action дескриптора <form> является URL файла, содержащего код CGI-сценария. Так, приведенное ниже выражение означает, что файл с кодом CGI-сценария находится на сервере www.myhp.edu в каталоге cgi-bin в файле script.рl.

<form action="http://www.myhp.edu/cgi-bin/script.pl" method="post">

Как веб-сервер различает, что надо сделать с файлом, на который указывает URL, — передать его содержимое клиенту или запустить файл на выполнение? Существует два способа распознавания файлов, содержащих тексты CGI-сценариев.

Первый способ заключается в том, что при установке веб-сервера один из каталогов специально выделяется для хранения сценариев. Обычно такой каталог получает имя cgi-bin (или Scripts для веб-сервера IIS). В этом случае, если клиент запрашивает файл из каталога cgi-bin, сервер воспринимает такой запрос как команду на запуск сценария. Файлы из других каталогов интерпретируются как HTML-документы.

Второй способ использует расширение файла. При настройке сервера указывается, что файлы с определенными расширениями содержат коды сценариев.

Идентификация по расширению используется относительно редко. Чаще всего все сценарии помещаются в cgi-bin, /Scripts или в другой каталог, специально выделенный для их хранения.

Вывод результатов выполнения CGI-сценария осуществляется чрезвычайно просто. Для того чтобы данные были переданы клиенту, достаточно вывести их в стандартный выходной поток. Однако, разрабатывая CGI-сценарий, не следует забывать о том, что он все же отличается от консольной программы и имеет следующие особенности.

Информация, передаваемая клиенту, должна соответствовать протоколу HTTP, т.е. состоять из заголовка и тела ответа. Как правило, получив данные от сценария, сервер самостоятельно добавляет первую строку заголовка.

НТТР/1.0 200 OK

Формирование информационных полей, входящих в состав заголовка, - задача сценария. Чтобы данные, переданные сценарием, были правильно интерпретированы клиентом, необходимо, чтобы в заголовке присутствовало как минимум поле Content-type. За заголовком должна следовать пустая строка. При отсутствии полей заголовка реакция браузера будет непредсказуемой. В подобных случаях браузер обычно пытается отобразить полученную информацию как текстовый файл.

Самый естественный формат для браузера - формат HTML. Результаты работы сценария обычно оформляются в виде веб-страницы, т.е. возвращаемые данные следует дополнить дескрипторами HTML. Таким образом, ответ CGI-сценария клиенту обычно выглядит так:

Content-type: text/html

<html>

<hеаd>

<titlе>ответ сценария</titlе>

</hеаd>

<body>

……………………

</body>

</html>

Обратите внимание на пустую строку после выражения Content-type: text/html. Она обязательно должна присутствовать в ответе, в противном случае клиент воспримет все последующие данные как продолжение заголовка.

После компиляции программы необходимо скопировать исполняемый файл в каталог cgi-bin (или в другой каталог, предназначенный для размещения исполняемых файлов) из которого он может запускаться веб-сервером на выполнение по запросу клиента.

Для вызова данного сценария достаточно включить в веб-страницу следующий фрагмент HTML-кода:

<form method="post" action="/cgi-bin/hello.exe">

<input type="submit">

</form>

Если сценарий вызывается из формы, ему передаются те данные, которые пользователь ввел с помощью интерактивных элементов, отображаемых на веб-странице - передача информации CGI-сценарию осуществляется в два этапа: сначала браузер передает данные веб-серверу, затем веб-сервер передает их сценарию.

В большинстве случаев кроме кнопки Submit форма содержит другие интерактивные элементы, каждый из которых имеет имя (атрибут NAME ) и значение (атрибут VALUE, либо последовательность символов, введенная пользователем). Из имен элементов и их значений формируется строка параметров, которая имеет следующий формат.

имя=значение&имя=значение& . . . &имя=значение

Каждый параметр представляет собой имя управляющего элемента и его значение, разделенные знаком равенства, а несколько таких пар объединяют строку с помощью символа " & ". Если в состав имени или значения входит символ " & " или " = ", то подобные символы кодируются последовательность знака процента " % ", за которым следуют две шестнадцатеричные цифры, определяющие код символа. Так, например, последовательностью " %21 " кодируется восклицательный знак " !". Как правило, при передаче параметров трехсимвольными последовательностями заменяются все знаки, кроме латинских букв, цифр и символа пробела (последний заменяется знаком " + ").

Таким образом, перед использованием строки параметров ее надо декодировать. Алгоритм декодирования чрезвычайно прост и включает в себя следующие действия:

* Выделить из строки параметров пары имя = значение.
* Выделить из каждой пары имя и значение.
* В каждом имени и каждом значении заменить символы " + " пробелами.
* Каждую последовательность из символа " % " и двух шестнадцатеричных и преобразовать в ASCII-символ.

Атрибут method дескриптора <form> имеет либо значение " GET ", либо значение " POST ". Значения " GET " и " POST " определяют два различных метода передачи параметров сценарию:

Если атрибут method имеет значение " GET ", строка параметров передается вместе с URL вызываемого сценария. Разделителем между URL и строкой параметров является символ " ?".

Если атрибут method имеет значение " POST ", строка параметров передается в теле HTTP-запроса.

Рассмотрим, как должен вести себя CGI-сценарий, чтобы правильно обработать данные в зависимости от метода, использованного при передаче данных, строка параметров доставляется CGI-сценарию различными способами.

Если атрибут METHOD дескриптора <FORM> имел значение " GET ", строка параметр передается серверу в качестве значения переменной окружения QUERY\_STRING.

При использовании метода POST данные доставляются сценарию по-другому. Они передаются через стандартный поток ввода (STDIN). Чтобы сценарий смог определить, сколько символов следует читать из стандартного ввода, веб-сервер устанавливает значение переменной окружения CONTENT\_LENGTH, равным длине строки параметров.

Получив управление, сценарий в первую очередь должен выяснить, с помощью какого метода выполнялась передача параметров. Эта информация содержится в переменной окружения REQUEST\_METHOD.

Таким образом, в простейшем случае, чтобы выполнить обработку строки параметров, достаточно знать назначение трех переменных окружения: REQUEST\_METHOD, QUERY\_STRING и CONTENT\_LENGTH.

Пример сценария на языке Perl, который возвращает клиенту строку параметров, приведен ниже. Сценарий определяет, какой метод использовался для передачи данных, читает строку параметров и передает ее клиенту, предварительно дополнив HTML-дескрипторами.

$method = $ENV{'REQUEST\_METHOD'};

if ($method eq "GET")

{ $pars = $ENV{'QUERY\_STRING'}; }

else

{ $length =$ENV{'CONTENT\_LENGTH'}; }

read (STDIN, $pars, $ length);

print "Content-type: text/html\n\n";

print "<HTML><BODY>\n";

print "<P>METHOD = ", $method;

print "<P>String of parameters: <P>\n";

print $pars;

print "</HTML></BODY>\n";

При разработке более сложных сценариев может потребоваться дополнительная информация. Информация о типах сервера и браузера, адресе клиент-машины и многие другие сведения передаются с помощью переменных окружения. Некоторые из них перечислены ниже

|  |  |
| --- | --- |
| REMOTE\_ADDR | IP-адрес узла, с которого поступил запрос |
| REMOTE\_HOST | Доменное имя узла, с которого поступил запрос |
| SERVER\_PORT | Номер порта, который использовался при обращении к серверу |
| SERVER\_SOFTWARE | Имя и версия сервера, посредством которого был запущен сценарий |
| SERVER\_NAME | Имя или адрес узла, на котором выполняется сервер |
| SERVER\_PROTOCOL | Название и версия протокола, с помощью которого был передан запрос |
| HTTP\_USER\_AGENT | Клиентская программа, отправившая запрос серверу |
| HTTP\_REFERER | URL документа, отображаемого браузером при вызове сценария |

***Сценарии***

К основным достоинствам разработки приложений на стороне веб-сервера в форме сценариев можно отнести следующие:

* поскольку сценарии не компилируются а интерпретируются, то ошибки в сценарии вызовут только диагностическое сообщение, но не приведут к дестабилизации веб-сервера или операционной системы.
* лучшие выразительные возможности. Язык сценариев как правило имеет собственный проблемно-ориентированный набор команд, и одна строка сценария может делать то же, что несколько десятков строк на традиционном языке. Как следствие, на этом языке может писать программист низкой квалификации.

Поддержка кроссплатформенности.

Поскольку сценарии интерпретируются из исходного кода динамически при каждом исполнении, они выполняются обычно значительно медленнее готовых программ, транслированных в машинный код на этапе компиляции.

В плане быстродействия сценарные языки можно разделить на:

Языки динамического разбора (например, command.com). Интерпретатор считывает инструкции из файла программы минимально требующимися блоками, и исполняет эти блоки, не читая дальнейший код.

Предварительно компилируемые (например Perl). Вначале считывается вся программа, затем компилируется либо в машинный код, либо в один из внутренних форматов, после чего получившийся код исполняется.

В рассмотрим кратко наиболее известные языки разработки сценариев для веб-приложений.

***Python***

Python — высокоуровневый язык программирования общего назначения с акцентом на производительность и читаемость кода. Язык Python сочетает в себе минимализм синтаксиса ядра и большой объем полезных функций в стандартной библиотеке.

Python поддерживает структурную, объектно-ориентированную, функциональную, императивную и аспектно-ориентированную парадигмы.

Его основные архитектурные черты:

* динамическая типизация
* автоматическое управление памятью
* полная интроспекция
* механизм обработки исключений
* поддержка многопоточных вычислений
* удобные высокоуровневые структуры данных

Код в Python организовывается в функции и классы, которые могут объединяться в модули (которые в свою очередь могут быть объединены в пакеты).

Для всех основных платформ Python имеет поддержку характерных для данной платформы технологий (например, Microsoft COM/DCOM). Cуществует даже специальная версия Python для виртуальной машины Java - Jython, что позволяет интерпретатору выполняться на любой системе, поддерживающей Java, при этом классы Java могут непосредственно использоваться из Питона и даже быть написанными на Python. Несколько проектов обеспечивают интеграцию с платформой Microsoft.NET, основные из которых - IronPython и Python.Net.

Стандартная библиотека Python имеет средства для работы со многими сетевыми протоколами и форматами интернета, например, модули для написания HTTP-серверов и клиентов, для разбора и создания почтовых сообщений, для работы с XML и т. п. Набор модулей для работы с операционной системой позволяет писать кросс-платформенные приложения. Существуют также модули для работы с регулярными выражениями, текстовыми кодировками, мультимедийными форматами, криптографическими протоколами, архивами, сериализации данных, поддержка юнит-тестирования и др.

Помимо стандартной библиотеки существует множество библиотек, предоставляющих интерфейс ко всем системным вызовам на разных платформах; Имеется большое количество прикладных библиотек для Python в самых разных областях (веб, базы данных, обработка изображений, обработка текста, численные методы, приложения операционной системы и т. д.).

***Ruby***

Ruby — интерпретируемый язык высокого уровня для быстрого и удобного объектно-ориентированного программирования. Ruby обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, " сборщиком мусора " и многими другими возможностями. Многие особенности синтаксиса и семантики языка Perl заимствованы в Ruby.

Первая общедоступная версия Ruby появилась в 1995 г.

Ruby - полностью объектно-ориентированный язык:

Все данные являются объектами, в отличие от многих других языков, где существуют примитивные типы.

Каждая функция является методом.

Переменные Ruby содержат не сами объекты, а ссылки на них.

Присваивание - это не передача значения, а копирование ссылки на объект.

В Ruby можно добавлять методы не только в любые классы, но и в любые объекты. Например, можно добавить к некоторой строке произвольный метод.

Массивы в Ruby могут автоматически изменять размер, могут содержать любые элементы и язык предоставляет мощные средства для их обработки.

Ruby поставляется с большой стандартной библиотекой. Это, прежде всего, библиотеки для работы с различными сетевыми протоколами на стороне сервера и клиента, средства для работы с различными форматами представления данных (XML, XSLT, YAML, PDF, RSS, CSV, WSDL). Также есть библиотеки для работы с архивами, датами, кодировками, матрицами, средства для системного администрирования, распределенных вычислений, поддержки многопоточности и т. д.

В языке Ruby также реализован простой и удобный механизм для расширения языка с помощью библиотек, написанных на Си, позволяющий легко разрабатывать дополнительные библиотеки. Для унифицированного доступа к базам данных разработана библиотека Ruby DBI.

К недостаткам интерпретатора Ruby можно отнести следующие:

Невысокая скорость работы.

Отсутствие поддержки потоков операционной системы (для Unix-подобных операционных систем есть поддержка процессов ОС), есть в экспериментальной версии 1.9.

Отсутствие встроенной поддержки юникода (возможна работа с использованием дополнительных библиотек, есть в экспериментальной версии 1.9).

Отсутствие компиляции в байткод. (При этом есть возможность компилировать Ruby в Java и .NET байткод, используя компилятор JRuby и Ruby.NET). В экспериментальную версию 2.0 входит виртуальная машина YARV, компилирующая Ruby в байткод и существенно ускоряющая исполнение.

***ASP***

ASP (Active Server Pages) — технология, разработанная компанией Microsoft, позволяющая легко создавать приложения для Веб.

Программирование на ASP дает разработчикам доступ к интерфейсу программирования приложений Internet Information Server с помощью языка сценариев VBScript и JScript.

ASP работает на платформе операционных систем линии Windows NT и на веб-сервере Microsoft IIS.

Архитектура ASP представлена ниже.

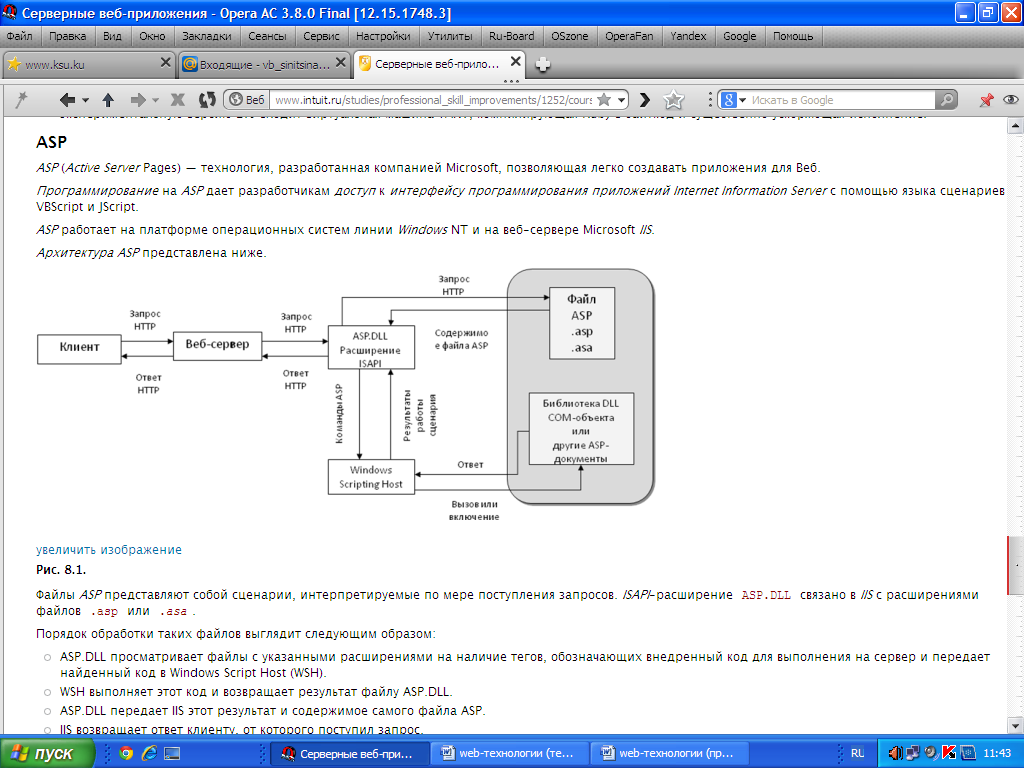


Рисунок 4.1 - Архитектура ASP

Файлы ASP представляют собой сценарии, интерпретируемые по мере поступления запросов. ISAPI-расширение ASP.DLL связано в IIS с расширениями файлов .asp или .asa.

Порядок обработки таких файлов выглядит следующим образом:

* ASP.DLL просматривает файлы с указанными расширениями на наличие тегов, обозначающих внедренный код для выполнения на сервер и передает найденный код в Windows Script Host (WSH).
* WSH выполняет этот код и возвращает результат файлу ASP.DLL.
* ASP.DLL передает IIS этот результат и содержимое самого файла ASP.
* IIS возвращает ответ клиенту, от которого поступил запрос.

Рассмотрим основы синтаксиса ASP.

IIS различает код, выполняющийся на сервере, и содержимое, отправляемое клиенту с помощью ASP.DLL, анализируя файл ASP на наличие начального " <% " и конечного " %> " тегов и выполняя код, расположенный между ними, с помощью WSH.

Рассмотрим пример:

<% Language=VBScript %>

<HTML>

<BODY>

<%

Response.Write("<p>Hello world!</p>")

%>

</BODY>

</HTML>

В примере первая строка кода <% Language=VBScript %> сообщает о необходимости использовать интерпретатор языка VBScript. Для вставки строки в документ был использован метод Write стандартного объекта Response.

Событие веб-запроса в ASP обрабатывается с помощью следующих объектов:

Response. Используется для записи данных в запрос HTTP, возвращаемый клиенту.

Application. Содержит параметры и конфигурации по настройке работы ASP для данного веб-сайта.

Request. Хранит содержимое HTTP-запроса и обеспечивает вспомогательные функции для обработки данных HTTP-запроса.

Server. Содержит информацию о веб-сервере, веб-сайте, а также обеспечивает поддержку вызывающей программы.

Session. Представляет собой состояние заданного веб-сеанса с заданным хостом клиентом.

***ISAPI***

Для веб-сервера IIS (Internet Information Server). был разработан специальный программный интерфейс для создания приложений расширяющих стандартные возможности веб-сервера.

ISAPI (Internet Server Application Programming Interface) – многозвенный API для IIS.

ISAPI также реализован в виде модуля mod\_isapi для веб-сервера Apache. Таким образом, серверные приложения, разработанные для MS IIS могут также выполняться в Apache и других веб-серверах.

В противоположность CGI - ISAPI-приложение загружается в том же адресном пространстве, что и веб-сервер IIS. Это позволяет повысить производительность приложений благодаря сокращению издержек на запуск отдельных процессов. Однако сбой ISAPI-приложения может привести к неустойчивой работе самого веб-сервера. В 6-ой версии IIS имеется возможность запуска приложений в рамках отдельного процесса.

ISAPI включает в себя 2 компоненты: расширения и фильтры.

Таким образом, все многообразие разрабатываемых ISAPI-приложений сводится только к этим двум типам. И фильтры и расширения компилируются в DLL файлы динамически запускаемые веб-сервером.

ISAPI приложения могут разрабатываться с помощью любых языков, поддерживающих экспорт стандартных С -функций, например С, С++, Delphi. Для разработки имеется ограниченное число библиотек для разработки ISAPI приложений, например Intraweb-компоненты Delphi Pascal, специальные MFC-классы, специальная С++ библиотека серверных технологий ATL.

К наиболее важным особенностям ISAPI-расширений можно отнести следующие:

* ISAPI -расширения имеют доступ ко всем функциональным возможностям IIS.
* Реализуются в виде DLL-модулей, загружаемых в пространстве процесса, контролируемого IIS.
* Клиенты могут обращаться к ISAPI -расширениям также как к статическим HTML страницам.
* ISAPI -расширения могут быть ассоциированы с отдельными расширениями файлов, с целыми каталогами или сайтами.

ISAPI-фильтры необходимы для изменения или совершенствования функциональности IIS. Они обычно работают с IIS-сервером и фильтруют каждый запрос. Фильтры применяются для анализа и модификации входящих и исходящих потоков данных.

Фильтры также как и расширения реализуются в виде DLL файлов.

Обычно ISAPI-фильтры используются для решения следующих задач:

* Изменение данных в запросе клиента (URL или заголовков).
* Управление отображением URL в физические файлы.
* Управление именами и паролями пользователей при анонимной или базовой аутентификации.
* Анализ и модификация запросов по завершении аутентификации.
* Модификация ответа веб-сервера.
* Ведение журналов и анализ трафика.
* Реализация собственной аутентификации.
* Управление шифрацией и сжатием.

Стоит отметить, что существуют реализации в виде ISAPI-расширений для таких инструментальных средств как:

* ASP (Active Server Pages )
* ASP.NET
* ColdFusion
* Perl ISAPI (Perlis)
* PHP

**Тема. Языки разработки сценариев Perl и PHP**

***Язык Perl***

Язык Perl (Practical Extraction and Report Language) — это язык программирования, сильными сторонами которого считаются его богатые возможности для работы с текстом, в том числе реализованные при помощи регулярных выражений. Также язык известен тем, что имеет огромную коллекцию дополнительных модулей CPAN.

Чтобы запустить программу на языке Perl на выполнение, ее компиляция не требуется, она вполне может выполняться под управлением интерпретатора. Чтобы файл с исходным текстом Perl можно было запускать на выполнение, надо чтобы первая его строка выглядела так:

#!путь\_к\_интерпретатору\_Perl

Основными типами данных в языке являются: скаляры, массивы (скалярные), хеш-таблицы (ассоциативные массивы), функции, файловые дескрипторы и константы.

Переменные разных типов отличаются знаком, который стоит перед именем переменной:

* $a - скаляр или указатель
* @b - скалярный массив
* %c - ассоциативный массив (хеш-таблица)
* &d - функция
* F - дескриптор ввода-вывода или константа

Скалярные переменные используются для хранения одиночных значений. Они могут содержать числа, строки и ссылки на другие объекты. Перед именем скалярной переменной необходимо ставить знак доллара '$'. Тип скалярной переменной не фиксирован и определяется динамически в зависимости от контекста.

Скалярный массив является упорядоченным списком скаляров. Каждый элемент массива имеет порядковый номер (индекс), с помощью которого к нему можно получить доступ. Нумерация элементов начинается с нуля.

Перед именем переменной типа скалярный массив указывается знак @, а для доступа к определенному элементу массива необходимо ставить знак $, так как определенный элемент массива является скаляром:

@winter = ("декабрь", "январь", "февраль");

print "Второй месяц зимы ", $winter[1], "\n";

Хеш-таблица представляет собой ассоциативный массив, позволяющий ассоциировать строку ( ключ ) со скаляром ( значение ). Строка при этом называется ключом, а скаляр в хеш-таблице - значением. Перед именем переменной-списка необходимо ставить знак процента %, а для доступа к определенному элементу массива ставят знак $.

Фактически хеш-таблица представляет собой массив, где в нечетных позициях находятся ключи, а на четных - значения.

Использование ассоциативных массивов напоминает использование массивов скалярных значений, однако индексация производится не целыми числами, а ключевыми словами. Кроме того, индексы заключаются не в квадратные, а в фигурные скобки.

Так, например, для того чтобы присвоить значение трем элементам массива %dict с индексами first, second и third, можно воспользоваться одним из двух способов, указанных ниже.

$dict { 'first' } = "первый";

$dict { 'second' } = "второй";

$dict { 'third' } = "третий";

или

%dict { 'first', 'second', 'third' } =

"первый", "второй", "третий";

Кроме того, существует способ одновременно записать в ассоциативный массив и ключевые слова, и их значения. Сделать это можно с помощью следующего выражения:

имя\_массива = ( ключ 1, значение 1, ключ 2, значение 2, ... );

Для примера, приведенного выше, это выражение будет выглядеть так:

%dict = ("first", "первый",

"second", "второй",

"third", "третий");

Рассмотрим приведенный следующий фрагмент программы на языке Perl.

while (<STDIN>)

{ print; }

Несмотря на то, что формально программа составлена правильно, она на первый взгляд может показаться бессмысленной. Однако при запуске она ведет себя точно так же, как и одна из программ "эхо". В языке Perl существует предопределенная скалярная переменная $\_, используемая по умолчанию. Именно в нее помещаются данные, прочитанные из стандартного ввода, и из нее берется значение для вывода в STDOUT.

Кроме $\_ в Perl имеются и другие предопределенные переменные:

$] - номер версии Perl.

$. - номер строки, прочитанной из файла последней.

$! - сообщение об ошибке.

$$ - идентификатор текущего процесса.

$^T - время в секундах с начала 1970 года до запуска данной программы.

$O - имя файла, в котором содержится выполняемая программа.

$1...$9 - фрагменты текста, отмеченные при выполнении операции сопоставления с шаблоном.

Подобно предопределенным скалярным переменным, в Perl существуют массивы, имеющие специальное значение. Наиболее важный из них - ассоциативный массив %ENV, содержащий текущие значения переменных окружения. Чтобы получить значение переменной окружения, надо обратиться к элементу данного массива, указав в качестве индекса имя переменной окружения. Так, приведенное ниже выражение записывает в скалярную переменную $path\_string значение переменной окружения PATH.

$path\_string = $ENV { 'PATH' };

Одна из первых строк CGI-сценария на Perl, может выглядеть так

$method = $ENV { 'REQUEST\_METHOD' };

Для работы с файлами и потоками в Perl предусмотрены специальные файловые дескрипторы.

Файловые дескрипторы представляет собой указатель на файл, устройство или PIPE канал, открытые для записи, чтения или для записи и чтения. Оператор " <> " в Perl называется бриллиантовым оператором (diamond operator). Он определяет операцию чтения строки из потока, дескриптор которого содержится в угловых скобках:

$str=<STDIN>; #чтение строки из дескриптора STDIN (стандартного потока ввода)

@lines=<F>; #чтение всех строк из связанного с дескриптором файла F.

print STDOUT $str; #печать в STDOUT (стандартный поток вывода)

Для связывания файла с файловым дескриптором используется функция open. Ниже приводятся варианты использования этой функции:

* open дескриптор\_потока > имя\_файла файл открывается для вывода данных. Если файл с указанным именем отсутствует, создается новый файл.
* open дескриптор\_потока >> имя\_файла файл открывается в режиме, позволяющем записывать данные в конец файла.
* open дескриптор\_потока +> имя\_файла открытый файл становится доступным для чтения и для записи.

Функция

сlose дескриптор\_файла

закрывает файл, связанный с указанным дескриптором.

В состав языка Perl входят средства поиска и замены, причем, задавая шаблон для поиска, можно использовать регулярные выражения. Это значит, что сложные операции, встречающиеся в специализированных приложениях, можно легко реализовать в любой Perl-программе.

Оператор поиска m// записывается следующим образом:

m/шаблон/

Если значение переменной $\_ содержит подстроку, соответствующую указанному шаблону, оператор поиска возвращает значение true.

Рассмотрим следующий пример:

$\_ = <INPUT>;

if (m/Scripts/)

{ print "В URL есть каталог Scripts \n"; }

else

{ print " В URL нет каталога Scripts \n"; }

Оператор замены s/// записывается следующим образом:

s/шаблон поиска/выражение для замены/[набор модификаторов]

При выполнении оператора s/// производится поиск соответствия шаблону, и если поиск завершается успешно, найденная подстрока заменяется указанным выражением. Подобно оператору m//, оператор s/// использует переменную $\_. Ниже приведен простейший пример применения оператора s///.

$\_ = "CGI-сценарий написан на языке С";

s/C$/Perl/;

print;

В результате выполнения сценария на консоль будет выведена следующая строка:

CGI-сценарий написан на языке Perl

Поскольку символ С содержится в аббревиатуре CGI, поэтому в шаблоне поиска указано, что он должен быть последним в строке.

За последним разделителем в операторе s/// могут следовать один или несколько модификаторов. Назначение некоторых модификаторов приведено ниже.

* g - глобальный поиск. Если этот модификатор не указан, после обнаружения первого соответствия оператор s/// закончит свою работу. Поэтому при отсутствии модификатора g будет произведено не более одной замены.
* i - указывает, что при поиске следует игнорировать регистр символов.
* e - указывает, что последовательность символов для замены следует интерпретировать не как подстроку, а как выражение Perl.

В выражении для подстановки могут присутствовать переменные $1 - $9, и в этом случае необходимо указать модификатор е. Так, например, если требуется интерпретировать десятичное число как код символа, можно воспользоватъся следующим выражением:

s/([0-9]+)/chr($1)/e;

Чтобы поиск или замена производились в строке, содержащейся в нужной переменной, надо использовать следующее выражение:

Переменная =~ оператор\_поиска\_или\_замены

Так, например, для преобразования шестнадцатеричных чисел, содержащихся в переменной $string, в десятичное представление можно использовать инструкцию:

$string =~ s/([0-9A-Fa-f]+)(H|h)/hex($1)/ge;

***Язык PHP***

Язык PHP (PHP :Hypertext Preprocessor) - один из наиболее популярных сценарных языков ввиду своей простоты, скорости выполнения, богатой функциональности и распространенности исходных кодов на основе лицензии PHP.

PHP состоит из ядра и набора подключаемых расширений: для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF и др. Возможна разработка своих собственных расширений с их последующим подключением. Хотя и существуют сотни расширений, однако в стандартную поставку входит лишь несколько десятков хорошо зарекомендовавших себя расширений.

Интерпретатор PHP подключается к веб-серверу либо через DLL модуль, созданный специально для этого сервера, либо в виде CGI -приложения.

В настоящее время PHP используется сотнями тысяч разработчиков. Порядка 20 миллионов сайтов сообщают о работе с PHP, что составляет более пятой доли доменов Интернета

Синтаксис PHP подобен синтаксису языка Си. При этом некоторые элементы, как например ассоциативные массивы и цикл foreach, заимствованы из языка Perl.

Для работы программы на PHP не требуется описывать какие-либо переменные, используемые модули, и т.п. Любая программа может начинаться непосредственно с оператора PHP.

<?php

echo 'Hello, world!';

?>

Помимо ограничителей <?php ?>, допускается использование дополнительных вариантов, таких как <? ?> и <script language="php"> </script>. Кроме того, до версии 6.0 допускается использование ограничителей языка программирования ASP <% %>.

Имена переменных начинаются с символа $, тип переменной объявлять не требуется. В отличие от имен функций и классов, имена переменных чувствительны к регистру. Переменные обрабатываются в строках, заключенных в двойные кавычки.

Инструкции завершаются точкой с запятой (;)

PHP поддерживает два типа комментариев:

в стиле языка С (ограниченные /\* \*/ ),

C++ (начинающиеся с // и идущие до конца строки)

PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных, равно как и самого объявления переменных. Преобразования между скалярными типами может осуществляться автоматически (хотя и имеются возможности для явного преобразования типов).

К скалярным типам данных относятся

целый тип ( integer ),

вещественный тип данных ( float, double ),

логический тип ( boolean ),

строковый тип ( string )

специальный тип NULL.

К нескалярным типам относится

* "ресурс" (resource),
* массив (array)
* и объект (object).

Тип NULL предназначен для переменных без определенного значения. Значение NULL принимают неиницализированные переменные, переменные инициализированные константой NULL, а также переменные, удаленные при помощи конструкции unset().

Ссылки на внешние ресурсы имеют тип resource. Переменные данного типа, как правило, представляют собой дескриптор, позволяющий управлять внешними объектами, такими как файлы, динамические изображения, результирующие таблицы базы данных и т. п.

Массивы поддерживают числовые и строковые ключи и являются гетерогенными. Массивы могут содержать значения любых типов, включая другие массивы. Суперглобальными массивами ( superglobal arrays ) в PHP называются предопределенные массивы, которые видны в любом месте исходного кода без использования ключевого слова global.

$GLOBALS - массив всех глобальных переменных (в том числе и пользовательских).

$\_SERVER - содержит множество информации о текущем запросе и сервере.

$\_ENV - текущие переменные среды. Их набор специфичен для каждой конкретной платформы, на которой выполняется сценарий.

$\_GET - ассоциативный массив с параметрами GET-запроса. В исходном виде эти параметры доступны в $\_SERVER['QUERY\_STRING'] и в $\_SERVER['REQUEST\_URI'] в составе URI.

$\_POST - ассоциативный массив значений полей HTML -формы при отправки методом POST.

$\_FILES - ассоциативный массив со сведениями об отправленных методом POST файлах. Каждый элемент имеет индекс идентичный значению атрибута " name " в форме и, в свою очередь, также является массивом со следующими элементами:

* ['name'] — исходное имя файла на компьютере пользователя.
* ['type'] — указанный агентом пользователя MIME - тип файла.
* ['size'] — размер файла в байтах.
* ['tmp\_name'] — полный путь к файлу во временной папке.
* ['error'] — код ошибки.

$\_COOKIE - ассоциативный массив с переданными агентом пользователя значениями cookie.

$\_REQUEST - общий массив вводных данных запроса пользователя как в массивах $\_GET, $\_POST, $\_COOKIE. Начиная с версии PHP 4.1 включается и содержимое $\_FILES.

$\_SESSION - информация о текущей сессии пользователя.

PHP поддерживает широкие объектно-ориентированные возможности, полная поддержка которых была введена в пятой версии языка. Класс в PHP объявляется с помощью ключевого слова class. Методы и поля класса могут быть общедоступными ( public, по умолчанию), защищенными ( protected ) и скрытыми ( private ). PHP поддерживает все три основные механизма ООП — инкапсуляцию, полиморфизм и наследование (родительский класс указывается с помощью ключевого слова extends после имени класса). Поддерживаются интерфейсы (ставятся в соответствие с помощью implements ). Разрешается объявление финальных, абстрактных методов и классов. Множественное наследование классов не поддерживается, однако класс может реализовывать несколько интерфейсов. Для обращения к методам родительского класса используется ключевое слово parent. Экземпляры класса создаются с помощью ключевого слова new, обращение к полям и методам объекта производится с использованием символов ->. Для доступа к членам класса из его методов используется переменная $this.

Среди наиболее часто используемых возможностей PHP стоит отметить следующие:

* имеется большой набор функций для работы со строками;
* работа с регулярными выражениями PCRE.
* работа с базами данных, осуществляемая посредством модулей:
* php5-mysql для MySQL,
* php5-pgsql для PostgreSQL
* и др.
* для PHP разработаны средства шаблонирования веб-страниц, позволяющие эффективно разделить представление от модели, например Smarty;
* имеется библиотека для работы с графическими изображениями GD, позволяющая производить преобразования с графическими файлами, и создавать изображения "на лету".

**Тема. Введение в C# и платформу Visual Studio.Net**

MicrosoftVisualStudio .NET - это интегрированная среда разработки для создания, документирования, запуска и отладки программ, написанных на языках .NET.

Эта среда разработки является открытой языковой средой. Наряду с языками программирования, изначально включенными в среду - C++, C#, J#, Visual Basic, - в нее могут добавляться любые языки программирования, компиляторы которых создаются сторонними разработчиками. Необходимым условием для включения языков в среду Visual Studio.Net является использование единого каркаса – платформы Framework.Net.

Платформа Framework. Net позволяет:

* легко использовать компоненты, разработанных на различных языках;
* разрабатывать единое приложение из нескольких частей на разных языках;

Платформа Framework.Net содержит две основных компоненты:

* FCL (Framework Class Library) - библиотеку классов каркаса;
* CLR (Common Language Runtime) - общеязыковую исполнительную среду.

В рамках данной платформы используется стандартная система типов Common Type System (CTS), которая полностью описывает все типы данных, поддерживаемые средой выполнения, определяет взаимодействие типов данных и их представление в формате метаданных .NET.

Набор правил, определяющих подмножество общих типов данных, в отношении которых гарантируется, что они небезопасны при использовании во всех языках .NET, описывается в рамках спецификации Common Language Specification (CLS). Для того чтобы классы, разработанные на разных языках, можно было совместно использовать в рамках одного приложения, они должны удовлетворять определенным ограничениям, задаваемым CLS. Класс, удовлетворяющий CLS, называется CLS-совместимым. Он доступен для использования в других языках, классы которых могут быть клиентами или наследниками совместимого класса.

Платформа .NET предоставляет в распоряжение программиста библиотеку базовых классов, доступную из любого языка программирования .NET. Поскольку число классов библиотеки FCL достигает нескольких тысяч, то в целях структуризации функционально близкие классы объединяются в группы, называемые пространством имен ( Namespace ).

Основным пространством имен библиотеки FCL является пространство System, содержащее как классы, так и другие вложенные пространства имен. Например, в пространстве System.Collections находятся классы и интерфейсы, поддерживающие работу с колТемами объектов - списками, очередями, словарями. Пространство System.Windows.Forms содержит классы, используемые при создании windows-приложений.

Следует помнить, что C# генерирует код, предназначенный для выполнения только в среде выполнения .NET (управляемый код). Сам двоичный файл, содержащий управляемый файл, называется сборкой. Сборка содержит код на промежуточном языке MSIL (Microsoft Intermediate Language) или просто IL. Аналогично байт-коду Java IL-код компилируется в платформенно-специфические инструкции при непосредственном обращении среды выполнении .NET к блоку IL-инструкций. Двоичные модули .NET в дополнение к инструкциям содержат также метаданные. Метаданные описывают не только типы, используемые в сборке, но и саму сборку. Данная часть метаданных называется манифестом.

В большинстве случаев между двоичным файлом .NET и сборкой существует отношение " один-к-одному ". Однако сборка может состоять как из одного, так и из нескольких двоичных файлов.

Сборка из одного файла содержит и манифест, и метаданные, и инструкции IL.

Двоичные файлы, образующие совместно общую сборку называются модулями. При этом один из двоичных файлом должен содержать манифест сборки. Остальные модули могут содержать только метаданные типов и инструкции IL.

Многофайловые сборки позволяют среде исполнения избирательно загружать только те сборки, которые в данный момент работы приложения действительно необходимы, что позволяет сокращать сетевой трафик и увеличивать скорость работы программ.

***Основы C#***

Что нового принес язык C# по сравнению с уже существовавшими до него языками?

В программах на C#, как правило, нет необходимости в работе с указателями (при сохранении этой возможности), поскольку в нем реализовано автоматическое управление памятью.

Предусмотрены встроенные синтаксические конструкции для работы с перечислениями, структурами и свойствами классов.

Имеется полная поддержка программных интерфейсов. Использование двоичных модулей .NET позволяет передавать объекты (по ссылке или по значению) через границы программных модулей.

Полная поддержка объектно-ориентированных технологий.

Для разработки приложений в Visual Studio.Net используются проекты.

Проект (Project) - это основная единица, с которой имеет дело разработчик. Сначала он должен выбрать тип проекта, после чего Visual Studio создает каркас проекта в соответствии с выбранным типом. Проект состоит из классов, собранных в одном или нескольких пространствах имен. Пространства имен (Namespaces) позволяют структурировать проекты, содержащие большое число классов, объединяя в одну группу близкие классы.

Несколько проектов могут объединяться в решение (Solution), которое также может включать ресурсы, необходимые этим проектам.

С точки зрения разработчика конечным результатом его работы, получаемым после компиляции исходного программного кода, является решение, а с точки зрения CLR (Common Language Runtime - общеязыковой среды исполнения) – сборка ( assembly ), содержащая PE файл, т.е. модуль в формате исполняемого файла PE (Portable Executable) для 32-разрядной ОС Windows либо DLL (Dynamic Link Library) файл.

Visual Studio.Net предлагает большое разнообразие возможных типов проектов.

***Типы данных С#***

Стандарт языка C++ включает следующий набор фундаментальных типов.

Логический тип (bool).

Символьный тип (char).

Целые типы. Они могут отличаться размером: short, int, long, а также могут быть знаковыми (signed) или беззнаковыми (unsigned).

Типы с плавающей точкой. Они также могут отличаться размерами: float, double и long double.

Тип void указывает на отсутствие информации.

К конструируемым типам относятся следующие:

* Указатели (например, char\*).
* Ссылки (например, char&).
* Массивы (например, char[]).

Также язык позволяет разработчику конструировать собственные типы:

* Перечислимые типы (enum).
* Структуры (struct).
* Классы.

В языке C# все типы можно рассматривать и под другим ракурсом, разделив их на четыре категории:

* Типы-значения (value).
* Ссылочные (reference).
* Указатели (pointer).
* Тип void.

Для ссылочного типа значение задает ссылку на область памяти в "куче" (heap), где расположен соответствующий объект. Для типа-значения значением являются собственно данные, а память для них выделяется в стеке.

Логический, арифметический, структуры, перечисление относятся типам-значениям. Массивы, строки и классы относятся к ссылочным типам.

И ссылочные, и обычные типы являются производными от базового класса object. В тех случаях, когда обычный тип должен вести себя как объект, создается оболочка (wrapper), которую можно рассматривать как ссылочный объект, помещенный в кучу, и в нее копируется значение переменной обычного типа. Оболочка автоматически помечается таким образом, что система знает, какое значение она содержит. Этот процесс назывется упаковкой ( boxing ), а обратный процесс - распаковкой ( unboxing ).

Упаковка происходит автоматически, для этого нужно только присвоить значение обычного типа переменной типа object. Упаковка и распаковка позволяют обрабатывать любой тип как объект. Например, в выражении

3.ToString();

целое число 3 упаковывается путем вызова функции Int32.ToString().

Массивы в C# могут быть многомерными ( multidimensional ) или невыровненными ( jagged ). Более сложные структуры данных такие, как стек и хеш-таблица определены в пространстве имен System.Collections.

В языке C# определен класс сhar[], и его можно использовать для представления строк постоянной длины. Однако массив сhar[] - это обычный массив, поэтому его нельзя инициализировать строкой символов. В C# не определено преобразование из класса сhar[] в класс String. У String есть динамический метод ToCharArray, задающий подобное преобразование в сhar[].

Класс String не разрешает изменять существующие объекты. Класс StringBuilder позволяет исправить этот недостаток. Этот класс принадлежит к изменяемым классам и его можно найти в пространстве имен System.Text.

***Классы и методы в C#.***

В Visual Studio.Net, и в C# в частности, любая программная система рассматривается как совокупность классов, объединенных в проекты, пространства имен, решения.

Описание класса имеет следующий синтаксис:

[атрибуты][модификаторы]class имя\_класса[:список\_родителей] {тело\_класса}

В теле класса могут быть объявлены:

* константы;
* поля;
* конструкторы и деструкторы;
* методы;
* события;
* делегаты;
* классы (структуры, интерфейсы, перечисления).

Поля класса синтаксически являются обычными переменными (объектами) языка. Их описание удовлетворяет обычным правилам объявления переменных. Поля характеризуют свойства объектов класса.

Методы класса синтаксически являются обычными процедурами и функциями языка. Методы содержат описания операций, доступных над объектами класса. Методы, называемые свойствами являются специальной синтаксической конструкцией, предназначенной для обеспечения эффективной работы с классами.

Конструктор представляет собой специальный метод класса, позволяющий создавать объекты класса. Его имя должно совпадать с именем класса. Если разработчик не определяет конструктор класса, то к классу автоматически добавляется конструктор по умолчанию - конструктор без аргументов.

Делегат в C# представляет собой описание случая класса и задает определение функционального типа (класса) данных. Экземплярами класса являются функции. Каждый делегат описывает множество функций с заданной сигнатурой. Каждая функция (метод), сигнатура которого совпадает с сигнатурой делегата, может рассматриваться как экземпляр класса, заданного делегатом. Синтаксис объявления делегата имеет следующий вид:

[<спецификатор доступа>] delegate <тип результата> <имя класса> (<список аргументов>);

***Выражения и операторы C#.***

Выражения строятся из операндов - констант, переменных, функций, - объединенных знаками операций и скобками. При вычислении выражения определяется его значение и тип.

В таблице ниже приведен список операций C#.

|  |  |
| --- | --- |
| Категория операций | Операции |
| Арифметические | + – \* / % |
| Логические (boolean и побитовые) | & | ^ ! ~ && || |
| Строковые | + |
| Инкремент и декремент | ++ -- |
| Сдвиг | >> << |
| Сравнение | == != < > <= >= |
| Присвоение | = += –= \*= /= %= &= |= ^= <<= >>= |
| Обращение к члену класса | . |
| Индексация | [] |
| Приведение типа (Cast) | () |
| Условие | ?: |
| Создание объекта | new() |
| Информация о типе | is sizeof typeof |
| Управление исключениями | checked unchecked |
| Косвенности и адресации | \* –> [] & |

Имя и тип переменной задаются при ее объявлении и остаются неизменными в течение всего времени ее жизни. Особенностью языка C# является требование обязательной инициализации переменной до начала ее использования. Попытка использовать неинициализированную переменную приводит к ошибкам, обнаруживаемым еще на этапе компиляции.

По используемым выражениям и операторам C# похож на C++. Так в программах на C# используются такие операторы как:

* Оператор присваивания ( = )
* Составной оператор ( {} )
* Операторы выбора: if-else и switch
* Операторы цикла: for, while
* Операторы break и continue
* Оператор return
* Оператор перехода goto

Кроме того, введены несколько новых инструкций. Например, оператор foreach позволяет получить доступ ко всем элементам массива или коллекции поочередно, в порядке возрастания индексов. Его синтаксис:

foreach (тип идентификатор in контейнер) оператор

***Интерфейсы C#***

Интерфейс представляет собой полностью абстрактный класс, все методы которого абстрактны. Однако методы интерфейса объявляются без указания модификатора доступа, и класс, наследующий интерфейс, обязан полностью реализовать все методы интерфейса. В этом - отличие от класса, наследующего абстрактный класс, где потомок может реализовать лишь некоторые методы родительского абстрактного класса, оставаясь абстрактным классом.

Интерфейс позволяет описывать некоторые желательные свойства, которыми могут обладать объекты разных классов.

Среди интерфейсов, встроенных в библиотеку базовых классов .NET, можно особо выделить такие как:

* IEnumerable (для работы с наборами объектов, в т.ч. с использованием оператора foreach).
* IClonable (копирование объектов).
* IComparable (для сравнения и сортировки объектов).

Пространство имен System.Collections, предназначенное для работы с наборами объектов, поддерживает интерфейсы:

* ICollection (определяет общие характеристики класса набора элементов).
* IComparer, IDictionary (позволяет представлять содержимое объекта в виде пар имя-значение).
* IDictionaryEnumerator (нумерация содержимого объекта, поддерживающего IDictionary).
* IEnumerable, IEnumerator.
* IHashCodeProvider (возвращает хэш-код с помощью выбранного алгоритма хэширования).
* IList (обеспечивает методы добавления, удаления и индексирования элементов в списке объектов).

***Сериализация объектов***

При работе с программной системой зачастую возникает необходимость в сериализации объектов. Под сериализацией понимают процесс сохранения объектов в долговременной памяти (файлах) в период выполнения системы. Под десериализацией понимают обратный процесс - восстановление состояния объектов, хранимых в долговременной памяти.

Механизмы сериализации C# и Framework.Net поддерживают два формата сохранения данных - в бинарном файле и XML файле. В первом случае данные при сериализации преобразуются в бинарный поток символов, который при десериализации автоматически преобразуется в нужное состояние объектов. Другой возможный преобразователь запоминает состояние объекта в формате XML.

Если класс объявить с атрибутом [Serializable], то в него встраивается стандартный механизм сериализации, поддерживающий глубокую сериализацию. Если по каким-либо причинам стандартная сериализация разработчика не устраивает, то класс следует объявить наследником интерфейса ISerializable, реализация методов которого позволит управлять процессом сериализации.

**Тема. Архитектура веб-приложений ASP.NET.**

**Разработка веб-приложений на платформе .NET**

Платформа .NET Framework предоставляет возможность разработки и интеграции веб-приложений. ASP.NET является одной из составляющих инфраструктуры .NET Framework и фактически является платформой для создания веб-приложений и веб-сервисов, работающих под управлением IIS.

ASP.NET внешне во многом напоминает более старую технологию ASP, но в то же время внутреннее устройство ASP.NET существенно отличается от ASP. Компания Майкрософт ASP.NET построила на базе CLR (Common Language Runtime), который является основой всех приложений .NET. Разработчики могут создавать код для ASP.NET, используя языки программирования, входящие в .NET Framework: C#, Visual Basic.NET, JScript.NET и другие.

***Рассмотрим более подробно, чем отличается ASP.NET от ASP.***

Классический ASP имеет следующие недостатки:

* Используются только языки сценариев, которые дают большой проигрыш в производительности (из-за их интерпретируемости) и не поддерживают многие возможности объектно-ориентированного программирования.
* Логика представления (в виде кода HTML) не отделена от бизнес-логики (исполняемого кода), что приводит перемешиванию в одном файле кода HTML с кодом сценария.
* Невозможно повторно использовать готовые решения в других проектах (возможно только копирование кода сценариев).

В файлах ASP.NET включается код на таких языках программирования как C#, JScript.NET, VisualBasic.NET, что позволяет применять непосредственно в веб-приложениях возможности объектно-ориентированного программирования. Также существенно сокращается объем кода, написанного вручную за счет применения серверных объектов, автоматически генерирующих код элементов управления HTML. Возможно использование стандартной среды разработки Visual Studio.NET, т.е. ASP.NET имеет преимущество в скорости по сравнению со сценарными технологиями, так как при первом обращении код компилируется и помещается в специальный кеш, а впоследствии только исполняется, не требуя затрат времени на парсинг, оптимизацию, и т. д.

Несмотря на возможность совместной работы ASP и ASP. NET на одном веб-сервере, они не могут использовать общий сеанс. Файлы ASP .NET обрабатываются библиотекой aspnet\_isapi.dll (а не asp.dll ), которая, в свою очередь, использует для выполнения кода технологию .NET.

Библиотека базовых классов .NET содержит пространства имен 3 основных групп:

* элементы web-приложений (протоколы, безопасность и др.);
* элементы графического интерфейса ( WebForms ) ;
* web -службы.

Как уже указывалось ранее, ASP.NET использует возможности стандартной среды разработки Visual Studio.Net, и в частности классы библиотеки FCL (Framework Class Library).

Разработчику веб-приложений на ASP.NET доступны классы, входящие в следующие пространства имен:

|  |  |
| --- | --- |
| Пространство имен | Содержание |
| System.Web | Организация взаимодействия web-клиента (браузера) с web-сервером (запрос-ответ, cookie и и др.) |
| System.Web.Caching | Поддержка кэширования при работе web-приложений |
| System.Web.Configuration | Настройка web-приложения в соответствии с файлами конфигурации проекта |
| System.Web.Security | Реализация системы безопасности web-приложений |
| System.Web.Services | Организация работы web-сервисов |
| System.Web.Services.Description |  |
| System.Web.Services.Discovery |  |
| System.Web.Services.Protocols |  |
| System.Web.UI | Построение графического интерфейса пользователей web-приложений |
| System.Web.UI.WebControls |  |
| System.Web.HtmlControls |  |

В свою очередь пространство имен System.Web включает в себя пространства имен, названия которых знакомы разработчикам веб-приложений на ASP:

|  |  |
| --- | --- |
| Пространство имен | Содержание |
| HttpApplication | Данный класс определяет общие для всех web-приложений члены |
| HttpApplicationState | В данном классе содержится общая информация web-приложения для множества запросов, сеансов и каналов передачи данных |
| HttpBrowserCapabilities | Этот класс используется для получения информации о возможностях клиентского браузера, обращающегося к web-серверу |
| HttpCookie | Поддержка механизма безопасной работы с объектами HTTP cookie |
| HttpRequest | Предоставляет доступ к информации, переданной web-клиентом |
| HttpResponse | Используется для формирования HTTP-ответа сервера |

В основу разработки веб-приложений на ASP.NET положена модель разделения кода представления и кода реализации, рекомендуемая Майкрософт при создании динамических документов с помощью программных кодов. Это делается путем размещения программного кода либо в отдельный файл, либо внутри специального тэга для сценариев. Файл такого рода обычно имеет расширение \*.aspx.cs ( \*.aspx.vb ) и имеет имя, совпадающее с именем основного ASPX файла. В принципе такой подход позволяет веб-дизайнеру сконцентрироваться работе с кодом разметки документа с минимальными изменениями программного кода, в обычном ASP внедряемого непосредственно в код разметки.

Взаимодействие пользователя с веб-приложением, реализованном на ASP.NET включает в себя следующие процессы:

При запросе страницы ASPX инициируется событие Page\_Init, производящее начальную инициализацию страницы и ее объекта.

Далее инициируется событие Page\_Load, которое может быть использовано, например для установки начальных значений для элементов управления. При этом также можно определить была ли загружена страница впервые или обращение к ней осуществляется повторно в рамках обратной отсылки в ответ на события, связанные с элементами управления, размещенными на странице; т.е. проверить свойство Page.IsPostBack.

Далее выполняется проверка валидности элементов страницы с точки зрения корректности введенных пользователем данных.

И, наконец, следует обработка всех событий, связанных с действиями пользователя с момента последней обратной отсылки.

Для сохранения данных веб-страницы в промежутках между обращениями к ней в ASP.NET используются состояния отображения ( view state).

Если данные, введенные в веб-форму, необходимо сделать доступными другим веб-формам того же приложения, эти данные необходимо сохранить в объектах Application и Session. Объекты Application доступны всем пользователям приложения и могут рассматриваться как глобальные переменные, обращение к которым возможно из любых сеансов. Объекты Session доступны только в рамках одного сеанса, и поэтому они оказываются доступными только одному пользователю.

***Серверные элементы управления ASP.NET***

Важной особенностью ASP.NET является использование серверных элементов управления на веб-странице (элементы WebForm ), которые являются фактически тэгами, понятными веб-серверу. Эти элементы определены в пространстве имен System.Web.UI.WebControls.

Принято выделять три типа серверных элементов управления:

* Серверные элементы управления HTML – обычные HTML тэги.
* Элементы управления веб-сервера – новые тэги ASP.NET.
* Серверные элементы управления для проверки данных (валидации) – применяются для валидации входных данных от клиентского приложения (обычно веб-браузера).

Преимущества от использования таких элементов при разработке веб-приложений:

* Сокращается количество кода, написанного вручную (что особенно заметно в для сложных элементов документа). Элемент просто "перетаскивается" из панели инструментов, после чего выполняется настройка его параметров в специальном окне. При этом все изменения автоматически заносятся непосредственно в \*.aspx файл.
* С программной точки зрения каждому из этих элементов управления соответствует определенный класс в библиотеке базовых классов .NET, что позволяет писать для них такой же код как и для любых других классов.
* Для любого элемента управления WebForm определен набор событий, обрабатываемых на веб-сервере.
* Для любого элемента управления WebForm предоставляется возможность для проверки ввода данных пользователем.

По умолчанию серверные элементы управления HTML в ASP.NET файлах рассматриваются как текст. Для их программирования требуется добавление атрибута runat="server" в соответствующий HTML элемент. Кроме того, все серверные элементы управления HTML должны быть размещены внутри области действия тэга <form>, также имеющего атрибут runat="server".

Подобно серверным элементам управления HTML элементы управления веб-сервера также создаются на веб-сервере и предполагают добавление атрибута runat="server". Однако они могут и не соответствовать конкретным элементам HTML, но представлять более сложные элементы.

Общий синтаксис для описания таких элементов:

<asp:тип\_элемента id="идентификатор" runat="server"/>

Серверные элементы валидации применяются для проверки вводимых пользователем данных.

Имеют следующий синтаксис:

<asp:тип\_элемента id="идентификатор" runat="server" />

Работа с источниками данных в ASP.NET

В ASP.NET используются два элемента управления WebForm для управления отображением данных, получаемых из источника данных:

DataGrid - элемент управления, отображающий содержимое объекта ADO.NET DataSet в виде таблицы.

DataList - элемент управления для выбора значений, заполняемых из источника данных.

Если необходимо отобразить данные, полученные по запросу пользователя из источника данных, в виде таблицы на веб-странице, то ASP.NET предоставляет в распоряжение веб-программиста удобный элемент управления DataGrid.

**Тема. Интерфейсы взаимодействия веб-приложений с СУБД**

Сегодня большинство информационных систем в той или иной степени используют базы данных. Не составляют исключение и системы, основанные на веб-технологиях. Поэтому организация взаимодействия веб-приложений с СУБД является неотъемлемой составной частью веб-технологий.

До начал 90-х годов существовало несколько разных поставщиков баз данных, каждый из которых имел собственный интерфейс. Если приложению было необходимо обмениваться данными с несколькими источниками данных, для взаимодействия с каждой из баз данных было необходимо написать отдельный код. С целью решения этой проблемы Майкрософт и ряд других компаний создали стандартный интерфейс для получения и отправки данных источникам данных различных типов. Этот интерфейс получил название open database connectivity ( ODBC ).

C помощью ODBC прикладные программисты смогли разрабатывать приложения с использованием единого интерфейса доступа к данным, не учитывая тонкости взаимодействия с различными источниками данных. Это достигается благодаря тому, что поставщики различных баз данных разрабатывают драйверы, учитывающие специфику конкретных источников данных при реализации стандартных функций из ODBC API. При этом приложения используют функции такого API, реализованные в соответствующем конкретному источнику данных драйвере.

По-сути, интерфейс ODBC является обычным процедурным API. ODBC поддерживается большим количеством операционных систем.

Имеются также ODBC-драйверы и для нереляционных данных, таких как электронные таблицы, текст и XML файлы.

Типичный сценарий работы веб-приложения с источником данных выглядит следующим образом:

* Установление соединение и подключение к источнику данных.
* Выполнение запросов, необходимых для выборки, вставки или изменения наборов данных источника.
* Отключение от источника данных.

Компанией Майкрософт был предложен интерфейс программирования приложений для доступа к данным, разработанный и основанный на технологии компонентов ActiveX - ADO (ActiveX Data Objects), который позволяет представлять данные из разнообразных источников (реляционных баз данных, текстовых файлов и т. д.) в объектно-ориентированном виде. Компоненты ADO нашли применятся при разработке приложений на таких языках как VBScript в ASP и Visual Basic.

В рамках Microsoft .NET основной моделью доступа приложений к источникам данных является ADO.NET. Она не является развитием ADO и представляет собой совершенно самостоятельную технологию.Компоненты ADO .NET входят в поставку .NET Framework .

ADO.NET включает в себя две основные части:

* Dataprovider - набор классов для доступа к источникам данных. Каждый из источников данных имеет свой собственный набор объектов, однако все они имеют общее множество классов: Connection, Command, Parameter, DataAdapter, DataReader.
* DataSets объекты - группа классов, описывающих простые реляционные базы данных, размещаемы в памяти. Содержит иерархию таких классов как: DataTable, DataView, DataColumn, DataRow, DataRowView, DataRelation, Constraint.

Объект DataSet заполняется данными из БД с помощью объекта DataAdapter, у которого заданы свойства Connection и Command. DataSet может сохранять свое содержимое также в XML (опционально вместе с XSD схемой) или получать данные из XML.

ADO.NET поддерживает работу с отсоединенными наборами данных, что крайне важно при использовании масштабируемых веб-приложений. Такая возможность реализуется с помощью класса DataSet совместно с классом DataAdapter.

Одной из важнейших составляющих технологии ADO.NET является поставщик данных. По-сути, это набор классов, предназначенных для взаимодействия с источником данных определенного типа. Использование разных поставщиков данных делает ADO.NET очень гибкой и расширяемой.

**Тема. Введение в XML**

В 1986 году, задолго до того, как идея создания сети Веб была воплощена в жизнь, универсальный стандартизированный язык разметки SGML (Standardized Generalized Markup Language) был утвержден в качестве международного стандарта (ISO 8879) определения языков разметки, хотя SGML существовал еще с конца шестидесятых. Он использовался для того, чтобы описывать языки разметки, предоставляя при этом автору возможность давать формальные определения каждому элементу и атрибуту языка.

Язык HTML первоначально был всего лишь одним из SGML-приложений. Он описывал правила, по которым должна быть подготовлена информация для World Wide Web. Таким образом, язык HTML - это набор предписаний SGML, сформулированных в виде определения типа документа (DTD), объясняющих, что именно обозначают тэги и элементы. Схема DTD для языка HTML хранится в веб-браузере.

К недостаткам языка HTML можно отнести следующие:

* HTML имеет фиксированный набор тэгов. Нельзя создавать свои тэги, понятные другим пользователям.
* HTML - это исключительно технология представления данных. HTML не несет информации о значении содержания, заключенного в тэгах.
* HTML - "плоский" язык. Значимость тэгов в нем не определена, поэтому с его помощью нельзя описать иерархию данных.

В качестве платформы для приложений используются браузеры. HTML не обладает достаточной мощью для создания веб-приложений на том уровне, к которому в настоящее время стремятся веб-разработчики. Например, на языке HTML невозможно разработать приложение для профессиональной обработки и поиска документов.

Большие объемы трафика сети. Существующие HTML-документы, используемые как приложения, перегружают Интернет большими объемами трафика в системах клиент-сервер. Примером может служить пересылка по сети большого по объему документа, в то время как необходима только небольшая часть этого документа.

Таким образом, с одной стороны, язык HTML является очень удобным средством разметки документов для использования в веб, а с другой - документ, размеченный в HTML, имеет мало информации о своем содержании. Если тот или иной документ несет достаточно полную информацию о своем содержании, появляется возможность сравнительно легко провести автоматическую обобщенную обработку и поиск в файле, хранящем документ. Язык SGML позволяет сохранять информацию о содержании документа, однако вследствие особой сложности он никогда не использовался так широко, как HTML.

Группа экспертов по языку SGML, возглавляемая Джоном Боузэком (Jon Bosak) из компании Sun Microsystems, приступила к работе по созданию подмножества языка SGML, которое могло бы быть принято Web-сообществом. Решено было удалить многие несущественные возможности SGML. Перестроенный таким образом язык назвали XML. Упрощенный вариант оказался значительно более доступным, чем оригинал, его спецификации занимали всего 26 страниц по сравнению с более чем 500 страницами спецификаций SGML.

Рассмотрим более детально структуру и особенности этого языка.

XML (eXtensible Markup Language) - рекомендованный W3C язык разметки. XML - текстовый формат, предназначенный для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе специализированных языков разметки. XML является упрощенным подмножеством языка SGML .

Язык XML имеет следующие достоинства:

* Это человеко-ориентированный формат документа, он понятен как человеку, так и компьютеру.
* Поддерживает Юникод.
* В формате XML могут быть описаны основные структуры данных - такие как записи, списки и деревья.
* Это самодокументируемый формат, который описывает структуру и имена полей также как и значения полей.
* Имеет строго определенный синтаксис и требования к анализу, что позволяет ему оставаться простым, эффективным и непротиворечивым.
* Широко используется для хранения и обработки документов;
* Это формат, основанный на международных стандартах;
* Иерархическая структура XML подходит для описания практически любых типов документов;
* Представляет собой простой текст, свободный от лицензирования и каких-либо ограничений;
* Не зависит от платформы;
* Является подмножеством SGML, для которого накоплен большой опыт работы и созданы специализированные приложения;

К известным недостаткам языка можно отнести следующие:

* Синтаксис XML избыточен.
* Размер XML документа существенно больше бинарного представления тех же данных (порядка 10 раз).
* Размер XML документа существенно больше, чем документа в альтернативных текстовых форматах передачи данных (например JSON, YAML) и особенно в форматах данных, оптимизированных для конкретного случая использования.
* Избыточность XML может повлиять на эффективность приложения. Возрастает стоимость хранения, обработки и передачи данных.
* Для большого количества задач не нужна вся мощь синтаксиса XML, и можно использовать значительно более простые и производительные решения.
* Пространства имен XML сложно использовать и их сложно реализовывать в XML парсерах.
* XML не содержит встроенной в язык поддержки типов данных. В нем нет понятий "целых чисел", "строк", "дат", "булевых значений" и т. д.
* Иерархическая модель данных, предлагаемая XML, ограничена по сравнению с реляционной моделью и объектно-ориентированными графами.

Вообще говоря, XML можно рассматривать не только как новый язык разметки, но и как основу для целого семейства технологий:

|  |  |
| --- | --- |
| XML | Технические рекомендации об использовании XML |
| DTD | Определение типа документа (схема) |
| XDR | Формат XML Reduced (схема Microsoft) |
| XSD | Определение схемы XML (схемы W3C) |
| Пространство имен | Метод определения имен элементов и атрибутов |
| XPath | Язык путей XML |
| XLink | Язык ссылок XML |
| XPointer | Язык указателей XML |
| DOM | API для объектной модели документа |
| SAX | Простой API для XML |
| XSL | Расширяемый язык таблиц стилей |
| XSL-FO | Объекты форматирования XSL |
| XSLT | Язык преобразований XSL |
| XInclude | Синтаксис XML Include |
| XBase | Синтаксис XML Base URI |

По-сути, XML служит метаязыком для описания структуры других языков. Взаимосвязь между SGML, XML, HTML и некоторыми другими языками показана на следующей диаграмме:

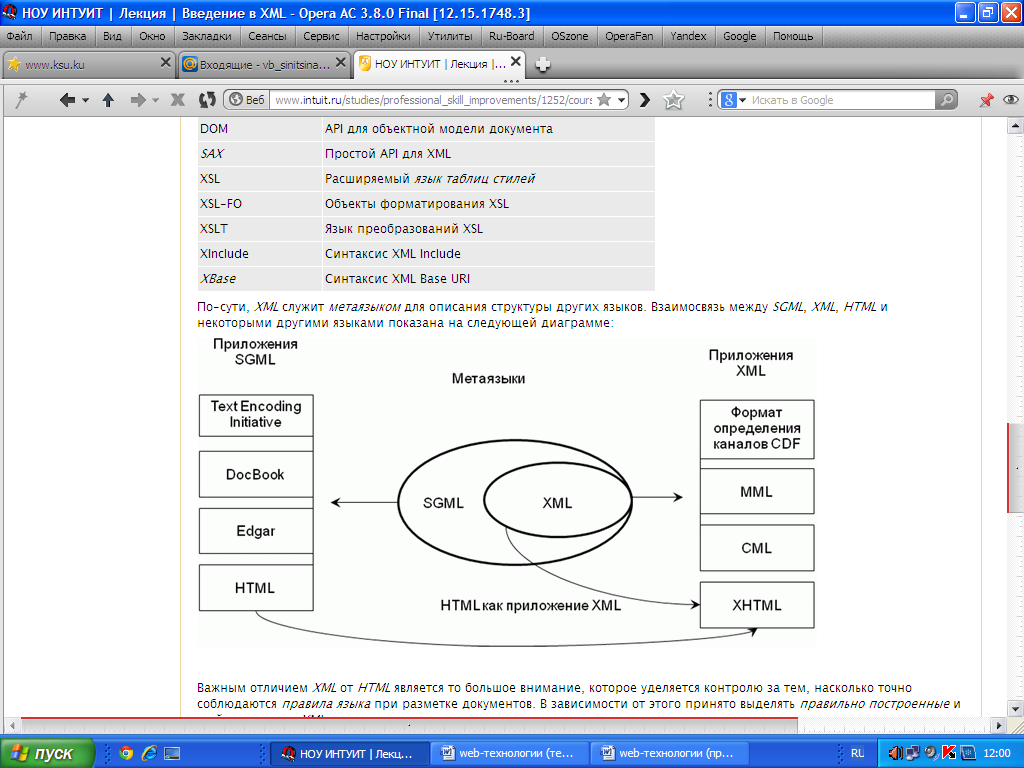


Рисунок 9.1 - Диаграмма

Важным отличием XML от HTML является то большое внимание, которое уделяется контролю за тем, насколько точно соблюдаются правила языка при разметке документов. В зависимости от этого принято выделять правильно построенные и действительные XML документы.

Документ XML считается правильно построенным, если он соответствует всем синтаксическим правилам XML.

Проверка действительности документа предполагает выполнение следующих действий:

* Проверка использования только заданного набора дескрипторов.
* Проверка полного соответствия порядка следования элементов и атрибутов содержанию документа или определенным правилам.
* Контроль типов данных (достигается при использовании соответствующей схемы).
* Контроль целостности данных для обеспечения оптимального обмена информацией через Веб с помощью транзакций.

Рассмотрим теперь основные синтаксические правила построения XML документов.

* XML документ содержит один и только один корневой элемент, содержащий все остальные элементы
* Дочерние элементы, содержащиеся в корневом элементе, должны быть правильно вложены.
* Имена элементов подчиняются правилам:
* Имя начинается с буквы, знака подчеркивания или двоеточия.
* После первого символа в имени могут быть буквы, цифры, знаки переноса, подчеркивания, точка или двоеточие.
* Имена не могут начинаться с буквосочетания XML.

XML документ имеет следующую структуру:

* Первая строка XML документа называется объявлением XML. Это необязательная строка, указывающая версию стандарта XML (обычно это 1.0). Также здесь может быть указана кодировка символов и внешние зависимости.
* Комментарий может быть размещен в любом месте дерева. XML комментарии размещаются внутри пары тегов <!-- и заканчиваются -->. Два знака дефис ( -- ) не могут быть применены ни в какой части внутри комментария.
* Остальная часть этого XML-документа состоит из вложенных элементов, некоторые из которых имеют атрибуты и содержимое.
* Элемент обычно состоит из открывающего и закрывающего тегов, обрамляющих текст и другие элементы.
* Открывающий тег состоит из имени элемента в угловых скобках;
* Закрывающий тег состоит из того же имени в угловых скобках, но перед именем еще добавляется косая черта.
* Содержимым элемента называется все, что расположено между открывающим и закрывающим тегами, включая текст и другие (вложенные) элементы.
* Кроме содержания у элемента могут быть атрибуты - пары имя=значение, добавляемые внутрь открывающего тега после названия элемента.
* Значения атрибутов всегда заключаются в кавычки (одинарные или двойные), одно и то же имя атрибута не может встречаться дважды в одном элементе.
* Не рекомендуется использовать разные типы кавычек для значений атрибутов одного тега.
* Для обозначения элемента без содержания, называемого пустым элементом, необходимо применять особую форму записи, состоящую из одного тега, в котором после имени элемента ставится косая черта "/".

К сожалению, описанные выше правила позволяют контролировать только формальную правильность XML документа, но не содержательную. Для решения второй задачи используются так называемые схемы.

Схема четко определяет имя и структуру корневого элемента, включая спецификацию всех его дочерних элементов. Программист может задать, какие элементы и в каком количестве обязательны, а какие – необязательны. Схема также определяет, какие элементы содержат атрибуты, допустимые значения этих атрибутов, в т.ч. значения по умолчанию.

Чаще всего для описания схемы используются следующие спецификации:

* DTD (Document Type Definition) - язык определения типа документов.
* XDR (XML Data Reduced) – диалект XML, разработанный Майкрософт.
* XSD (язык определения схем XML) – рекомендована консорциумом W3C.

XML документ отличается от HTML документа также и тем, как он отображается в веб-браузере. Без использования CSS или XSL XML-документ отображается как простой текст в большинстве веб-браузеров. Некоторые веб-браузеры, такие как Internet Explorer, Mozilla и Firefox отображают структуру документа в виде дерева, позволяя сворачивать и разворачивать узлы с помощью нажатий клавиши мыши.

Наиболее распространены три способа преобразования XML-документа в отображаемый пользователю вид:

* Применение стилей CSS.
* Применение преобразования XSLT.
* Написание на каком-либо языке программирования обработчика XML-документа.

**Тема. Языки описания cхем XML**

Идея создания собственных тэгов, имеющих специальное значение и помогающих описать содержание документа, сама по себе просто замечательна. Но если каждый пользователь может создавать свои собственные описания, каким образом их распознавать? С этой целью в спецификации XML для описания подобных "самодеятельных" тэгов используются схемы. Они необходимы для того, чтобы:

* описать, что именно является разметкой;
* описать точно, что означает разметка.

Наиболее известными языками описания схем являются следующие:

* DTD (Document Type Definition) - язык определения типа документов, который первоначально использовался в качестве язык описания структуры SGML-документа.
* XDR (XML Data Reduced) – диалект схемы XML, разработанный Microsoft, который поддерживался в Internet Explorer 4 и 5 версий.
* XML Schema или просто XSD ( язык определения схем XML) – рекомендация консорциума W3C с 2001 года.

Рассмотрим подробнее первые два из них. Третий язык описания схем рассматривается в лабораторной работе 11.

***DTD схема***

Схема DTD предоставляет шаблон разметки документа, в котором указываются наличие, порядок следования и расположение элементов и их атрибутов в документе XML.

В рамках DTD модель содержимого XML документа можно описать следующим образом:

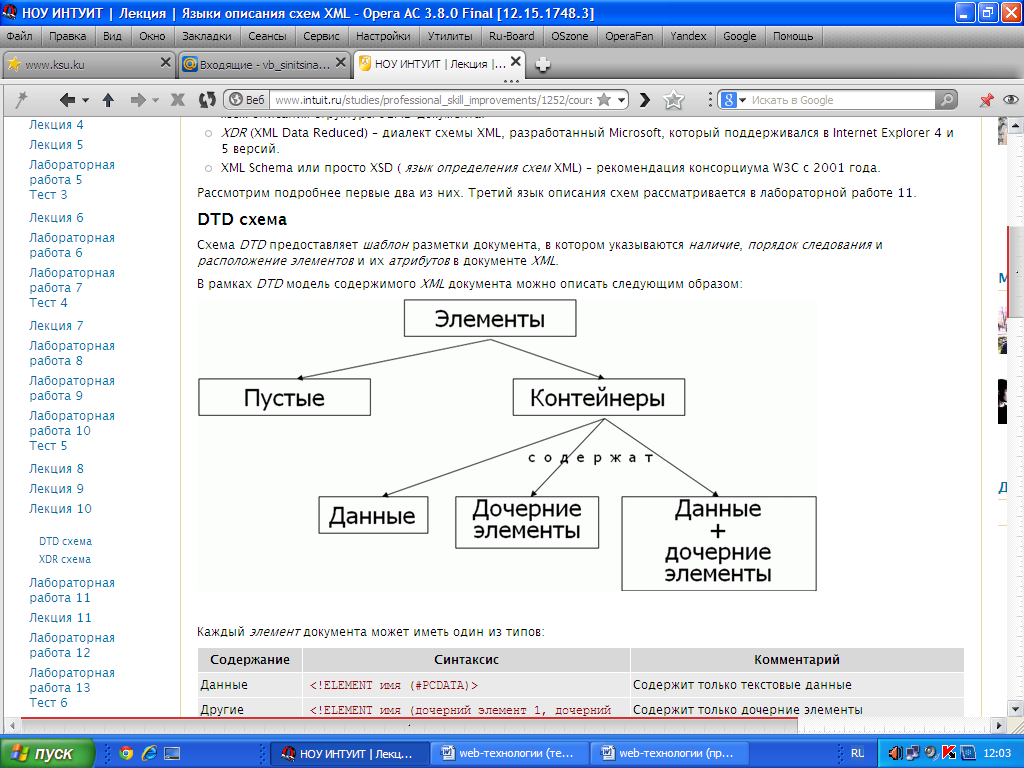


Рисунок 10.1 – Модель содержимого XML документа

Каждый элемент документа может иметь один из типов:

|  |  |  |
| --- | --- | --- |
| Содержание | Синтаксис | Комментарий |
| Данные | <!ELEMENT имя (#PCDATA)> | Содержит только текстовые данные |
| Другие элементы | <!ELEMENT имя (дочерний элемент 1, дочерний элемент 2)> | Содержит только дочерние элементы |
| Смешанное | <!ELEMENT имя (#PCDATA, дочерний элемент)\*> | Содержит комбинацию текстовых данных и дочерних элементов |
| EMPTY | <!ELEMENT имя EMPTY> | Ничего не содержит |
| ANY | <!ELEMENT имя ANY> | Может содержать текстовые данные или дочерние элементы |

Атрибуты, находящиеся внутри тэгов документа, описываются отдельно с помощью синтаксиса:

<!ATTList

имя\_элемента имя\_атрибута1 (тип) значение\_по\_умолчанию

…………………………………………………………………………………...

имя\_элемента имя\_атрибутаN (тип) значение\_по\_умолчанию >

При этом атрибут в DTD может иметь один из трех типов:

* Строка
* Маркированные атрибут
* Атрибута с перечислением

Кроме типа атрибута можно также задавать и его модальность:

|  |  |
| --- | --- |
| Значение | Описание |
| #REQUIRED | Атрибут обязательно должен быть указан |
| #FIXED | Значение атрибута не должно отличаться от указанного |
| #IMPLIED | Необязательное значение |

Рассмотрим в качестве примера описание атрибутов строкового типа для элемента, описывающего некоторое сообщение:

<!ATTLIST message

number CDATA #REQUIRED

date CDATA #REQUIRED

from CDATA #FIXED

status CDATA #IMPLIED>

Если этот элемент содержит атрибуты с перечислением, то их описание может выглядеть, например, следующим образом:

<!ATTLIST message

number ID #REQUIRED

from CDATA #REQUIRED

alert (low | normal | urgent) "normal">

Маркированных атрибуты элемента могут быть четырех типов:

|  |  |
| --- | --- |
| Значение | Описание |
| ID | Уникальный идентификатор элемента (начинается с буквы, двоеточия или подчеркивания) |
| IDREF | Ссылка на элемент, содержащий атрибуты ID |
| ENTITIES | Ссылка на внешний элемент |
| NMTOKEN | Содержит буквы, цифры, точки, знаки подчеркивания, переносы, двоеточия, но не пробелы |

И, наконец, в DTD можно использовать следующие индикаторы вхождения последовательностей:

|  |  |  |
| --- | --- | --- |
| Символ | Пример | Описание |
| , | (a, b, c) | Последовательное использование элементов списка |
| | | (a | b | c) | Используется один из членов списка |
|  | date | Используется один и только один элемент |
| ? | subject? | Необязательное использование (0 или 1 раз) |
| + | paragraph+ | Используется один или несколько раз |
| \* | brother\* | Используется ноль или несколько раз |

В качестве примера приведем DTD схему, описывающую структуру электронного почтового ящика:

<!ELEMENT mailbox (message\*)>

<!ELEMENT message (head, body)>

<!ATTLIST message uid CDATA #REQUIRED>

<!ELEMENT head ( from,to+, subject?, CC\*, notify?) >

<!ELEMENT from (#PCDATA)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT subject (#PCDATA)>

<!ELEMENT CC (#PCDATA)>

<!ELEMENT notify EMPTY>

<!ELEMENT body (#PCDATA)>

Исходный XML документ, удовлетворяющий данной схеме, может выглядеть, например, так:

<?xml version="1.0" ?>

<!DOCTYPE mailbox SYSTEM "mailbox.dtd">

<mailbox>

<message uid="1">

<head>

<from>user1@myhp.edu</from>

<to>user2@myhp.edu</to>

<subject>Re:</subject>

</head>

<body>

What's up!

</body>

</message>

<message uid="2">

<head>

<from>user3@myhp.edu</from>

<to>user2@myhp.edu</to>

<subject>Remind</subject>

<CC> user1@myhp.edu </CC>

<notify/>

</head>

<body>

Remind me about meeting.

</body>

</message>

</mailbox>

Обратите внимание на 2-ю строчку документа, в которой указывается внешняя ссылка на файл, содержащий DTD схему.

В принципе, DTD допускает два способа использования в XML документе.

Объявление внутренней схемы:

<!DOCTYPE корневой\_элемент [

<!ELEMENT корневой\_элемент (модель содержания)>

]>

Объявление внешней схемы:

<!DOCTYPE корневой\_элемент SYSTEM "name.DTD">

В заключение укажем на следующие недостатки DTD схем:

* Не являются экземплярами XML. Требуется изучение совершенно другого языка.
* Не предоставляют контроль за типами данных, за исключением самых простых текстовых данных.
* Не являются экземплярами XML, поэтому их нельзя легко расширить или преобразовать к другим языкам разметки – HTML или DHTML.
* Не обеспечивают поддержки пространств имен XML.

***XDR схема***

XML-Data – полное имя языка описания схем, предложенного Майкрософт, а XML-DataReduced – это "часть" полной рекомендации. Схема XDR - это экземпляр XML, т.е. соответствует всем синтаксическим правилам и стандартам XML.

Реализуя проверки данных на уровне документа с помощью схемы, приложения, генерирующие и принимающие транзакции, можно оптимизировать для обеспечения максимального быстродействия. Соответствие полей и правильность записей проверяются на уровне экземпляров XML.

Корневым элементом в схеме XDR всегда является элемент Schema:

<Schema

name="имя\_схемы" xmlns="urn:schemas-microsoft-com:xml-data"

xmlns:dt="urn:schemas-microsoft-com:datatypes">

<-- Объявления других элементов -->

</Schema>

Элемент ElementType имеет синтаксис:

<ElementType

content="{empty | texOnly | eltOnly | mixed}">

dt:type "datatype"

model="{open | closed}"

name = "idref"

order="{one | seq | many}"

>

Элемент ElementType может иметь следующие атрибуты:

|  |  |
| --- | --- |
| Имя атрибута | Описание |
| name | Имя элемента |
| content | Содержание элемента. Допустимые значения: empty (пустой элемент), eltOnly (может быть только контейнером для других элементов), textOnly (только текстовые данные), mixed (смешанные данные). |
| dt:type | Тип данных элемента |
| model | Может принимать значения:  Open – разрешено использовать элементы, не определенные в схеме  Closed – запрещено использовать элементы, не определенные в схеме |
| order | Порядок следования дочерних элементов в экземпляре XML. Допустимые значения:  one – предполагается наличие одного документа  many – любое количество элементов в любом порядке  seq – элементы указываются в строго заданном порядке. |

В качестве дочерних элементов для ElementType можно использовать следующие:

|  |  |
| --- | --- |
| Имя элемента | Описание |
| element | Объявляет дочерний элемент |
| description | Обеспечивает описание элемента ElementType |
| datatype | Обеспечивает тип данных элемента ElementType |
| group | Определяет порядок следования элементов |
| AttributeType | Определяет атрибут |
| attribute | Определяет сведения о дочернем элементе AttributeType |

Для объявления атрибутов используется синтаксис:

<AttributeType

default="default-value"

dt:type="primitive-type"

dt:values="enumerated-values"

name="idref"

required="{yes|no}"

>

В свою очередь элемент AttributeType может иметь атрибуты:

|  |  |
| --- | --- |
| Значение | Описание |
| default | Значение по умолчанию |
| dt:type | Один из следующих типов:  entity, entities, enumeration, id, idref, nmtoken, nmtokens, notation, string |
| dt:values | Допустимые значения |
| name | Имя атрибута |
| required | Указывает на обязательное наличие атрибута в описании |

Синтаксис для описания элемента attribute выглядит следующим образом:

<attribute

default="default-value"

type="attribute-type"

[required="{yes|no}"]

>

а его возможные значения могут быть такими:

|  |  |
| --- | --- |
| Значение | Комментарий |
| default | Значение по умолчанию |
| type | Имя элемента AttributeType, определенного в данной схеме. Должно соответствовать атрибуту name элемента AttributeType |
| required | Указывает на обязательное наличие атрибута в описании |

В отличие от DTD схем XDR поддерживает типы данных. Элемент Schema имеет следующий атрибут:

Xmlns:dt="urn=schemas-microsoft-com:datatypes"

С полным списком типов данных можно ознакомится на странице по адресу: http://msdn.microsoft.com/en-us/library/ms256121(VS.85).aspx

Индикаторы вхождения в схемах XDR имеют синтаксис:

<element

type="element-type"

[minOccur="{0|1}"]

[maxOccur="{1|\*}"]

>

XDR схема позволяет определять группы содержания. Так, в элементе ElementType может содержаться элемент group, имеющий синтаксис:

<group order="(one|seq|many)" minOccur="(0|1)" maxOccur="(1|\*)">

<element type="ElementType/">

<element type="ElementType/">

<element type="ElementType/">

<element type="ElementType/">

</group>

В заключение приведем пример XSD схемы, описывающей структуру XML документа, содержащего письма электронной почты:

<?xml version = "1.0"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">

<xsd:element name="m\_box">

<xsd:complexType>

<xsd:sequence>

<xsd:element ref="message" minOccurs="0"

maxOccurs="unbounded"/>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name="message">

<xsd:complexType>

<xsd:sequence>

<xsd:element ref="head" minOccurs="1" maxOccurs="1"/>

<xsd:element ref="body" minOccurs="1" maxOccurs="1"/>

</xsd:sequence>

<xsd:attribute name="uid" use="required" type="xsd:string"/>

</xsd:complexType>

</xsd:element>

<xsd:element name="head">

<xsd:complexType>

<xsd:sequence>

<xsd:element ref="to" minOccurs="1" maxOccurs="unbounded"/>

<xsd:element ref="from" minOccurs="1" maxOccurs="1"/>

<xsd:element ref="date" minOccurs="1" maxOccurs="1"/>

<xsd:element ref="subject" minOccurs="1" maxOccurs="1"/>

<xsd:element ref="cc" minOccurs="0" maxOccurs="unbounded"/>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name="to" type="xsd:string"/>

<xsd:element name="from" type="xsd:string"/>

<xsd:element name="date" type="xsd:string"/>

<xsd:element name="subject" type="xsd:string"/>

<xsd:elememt name="cc" type="xsd:string"/>

</xsd:schema>

Для проверки действительности XML документа можно использовать специальные валидаторы, например W3C валидатор (http://validator.w3.org/).

Для проверки схем также существуют специальные валидаторы, например XML Schema валидатор (http://www.w3.org/2001/03/webdata/xsv).

Согласно спецификации W3C XML программа должна прекратить обработку XML документа, как только будет обнаружена ошибка в этом документе.

**Тема. DOM XML. Преобразование XML документов**

Для программной обработки XML документов используется модель XML DOM, которая определяет объекты и свойства всех XML элементов и методы (интерфейс) для доступа к ним. Иначе говоря, XML DOM описывает каким образом необходимо получать, изменять, добавлять и удалять XML элементы.

Согласно DOM:

* все, что содержится внутри XML документа, является узлом;
* весь документ представляется узлом документа ;
* каждый XML элемент – узел элемента ;
* текст внутри XML элементов - текстовый узел ;
* каждый атрибут - узел атрибута ;
* комментарии - узлы комментариев.

XML документ в соответствии с моделью XML DOM представляется как дерево из узлов, при этом:

Все узлы дерева находятся в определенных отношениях друг с другом.

Все узлы доступны через дерево. Их содержимое может быть изменено, удалено; новые элементы могут быть добавлены в дерево.

Дерево начинается с корневого узла и разветвляется вниз вплоть до текстовых узлов на самом низшем уровне дерева.

Все узлы находятся в иерархических отношениях между собой.

Эти отношения описываются с помощью понятий родитель, дочерний и потомок (все дочерние на одном уровне).

Альтернативным интерфейсом для обработки XML документов является SAX.

SAX (Simple API for XML) - прикладной программный интерфейс для парсера с последовательным доступом к XML. Этот интерфейс предоставляет механизм чтения данных из XML документа.

SAX парсер является поточным и управляемым событиями. Задачей пользователя SAX API заключается в описании методов, вызываемых событиями, возникающими при анализе документа.

Такими событиями могут быть следующие:

* текстовый узел;
* узел элемента XML;
* инструкция обработки XML;
* комментарий XML.

События вызываются появлением как открывающего тэга, так и закрывающего тэга любого из этих элементов документа. Атрибут XML также рассматривается как событие.

Анализ документа является однонаправленным (т.е. без возвратов по дереву).

В отличие от DOM формальной спецификации для SAX не существует. В качестве нормативной рассматривается Java реализация SAX.

Следует отметить следующие достоинства и недостатки SAX.

Достоинства:

* Затраты памяти существенно меньше (зависит от максимальной глубины дерева документа и количества атрибутов в узле элемента), чем в случае DOM (требуется хранить в памяти все дерево документа).
* Скорость работы выше за счет сокращения затрат времени на выделение памяти для элементов дерева в случае DOM.
* Потоковое чтение данных с диска в случае DOM невозможно. Если для размещения всего документа в памяти недостаточно места, то использование SAX является безальтернативным.

Недостатки:

* Процедура проверки правильности предполагает доступ ко всему документу одновременно.
* Это также требуется и в случае XSLT преобразования.

Если загрузить "чистый" XML документ в веб-браузер, то можно будет увидеть древовидную структуру этого документа:

В этом как раз и заключается главное отличие между XML и HTML, а именно разделение структуры документа и его представления в браузере. Конкретный вид XML документа описывается отдельно с помощью CSS или XSL.

CSS и XSL — принципиально разные технологии, имеющие лишь частичное пресечение областей применения. CSS-форматирование применяется к HTML-документу браузером на клиентской стороне, а XSL-преобразование выполняется, как правило, на сервере, после чего результат отправляется браузеру клиента. XSL базируется на XML, благодаря чему XSL более гибок и универсален. У разработчиков имеется возможность использовать средства контроля за корректностью составления стилевых списков (используя схемы XML).

С помощью XSL можно преобразовать XML-документ в формат HTML, WML, RTF, PDF, SQL, SWF, а так же в другой XML и XSL документ. XSL указывает как будет оформлен документ, где и каким образом должны размещаться данные.

Cпецификация XSL состоит из трех частей:

* XSLT (XSL Transformations), язык для преобразования XML;
* XPath - язык путей и выражений, используемый в XSLT для доступа к отдельным частям XML-документа;
* XSL-FO (XSL Formatting Objects), язык для верстки XML.

Наиболее распространенным механизмом XSLT преобразований для систем работающих на платформе Microsoft Windows является MSXML ; для систем на основе GNU — xsltproc.

Для того, чтобы обработать XML документ c помощью XSL, необходимо в XML документе написать следующую инструкцию:

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="my-style.xsl"?>

<root>

< !-- ... -- >

</root>

Возвращаясь к рассмотренному ранее примеру, добавив в XML файл ссылку на XSL файл, получим следующий код разметки:

<?xml version="1.0" encoding="windows-1251" ?>

<!DOCTYPE mailbox SYSTEM "mailbox.dtd">

<?xml-stylesheet href="mailbox.xsl" type="text/xsl" ?>

<mailbox>

<message uid='1'>

<body>Проверка почтового ящика</body>

<head>

<to>user2@myhp.edu</to>

<to>user3@myhp.edu</to>

<from>user1@myhp.edu</from>

<subject>Test</subject>

<cc></cc>

<notify></notify>

</head>

</message>

<message uid='2'>

<body>Это письмо не является спамом</body>

<head>

<to>user2@myhp.edu</to>

<from>user4@myhp.edu</from>

<subject>Интересное предложение</subject>

<notify></notify>

</head>

</message>

</mailbox>

После загрузки данного документа в веб-браузере его вид кардинально изменится:

Содержимое XSL файла mailbox.xsl приводится ниже:

<?xml version="1.0" encoding="windows-1251"?>

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match = "mailbox">

<h2 align="center" color="red">Почтовый ящик</h2>

<xsl:apply-templates/>

</xsl:template>

<xsl:template match = "message">

<table align="center" border="0" width="80%" bgcolor="gray">

<tr>

<td width="20%"><b>From:</b></td>

<td>

<xsl:value-of select="head/from"/>

</td>

</tr>

<tr>

<td width="20%"> <b>To:</b> </td>

<td> <xsl:for-each select="head/to">

<xsl:apply-templates/>,

</xsl:for-each>

</td>

</tr>

<tr>

<td width="20%"><b>Subject:</b></td>

<td><xsl:value-of select="head/subject"/></td>

</tr>

<tr>

<td width="20%"><b>CC</b></td>

<td><xsl:value-of select="head/cc"/></td>

</tr>

<tr>

<td width="20%"><b>Notify</b></td>

<td><xsl:value-of select="head/notify"/></td>

</tr>

<tr>

<td colspan="2">

<textarea cols="120%" rows="3">

<xsl:value-of select="body"/>

</textarea>

</td>

</tr>

</table>

<br/><hr/><br/>

</xsl:template>

</xsl:stylesheet>

XSLT и XPath

***Спецификация XSLT является рекомендацией W3C.***

В результате применения таблицы стилей XSLT, состоящей из набора шаблонов, к XML-документу (исходное дерево) образуется конечное дерево, которое может быть другой XML-структурой, HTML-документом или обычным текстом. Правила выбора данных из исходного дерева записываются на языке запросов XPath. XSLT применяется в основном в веб-программировании и для генерации отчетов.

Благодаря XSLT реализуется отделение данных от их представления в рамках парадигмы MVC ( Model-view-controller ).

XPath (XML Path Language) - язык запросов к элементам XML-документа. XPath был разработан для организации доступа к частям документа XML в файлах трансформации XSLT и является стандартом консорциума W3C. В языке XPath используется компактный синтаксис, отличный от принятого в XML. Начиная с версии 2.0, XPath является составной частью языка XQuery. XPath призван помочь обходить всевозможные деревья, получать необходимые элементы из другой ветви относительно точки обхода, распознавать предков, потомков, атрибуты элементов. Это полноценный язык навигации по дереву.

Для нахождения элемента(ов) в дереве документа используются пути адресации.

Каждый шаг адресации состоит из трех частей:

* оси, например child::;
* условия проверки узлов, например имена элементов документа body, html ;
* предиката, например attribute::class .

Дополнением к ядру языка является набор функций, которые делятся на 5 групп: системные функции, функции с множествами, строковые функции, логические функции, числовые функции.

***XSL-FO***

XSL-FO (eXtensible Markup Language Formatting Objects) - рекомендованный W3C язык разметки предпечатных материалов. По-сути , XSL-FO - это унифицированный язык представления. Он не имеет семантической разметки (как в HTML) и сохраняет все данные документа внутри себя (в отличие от CSS, который модифицирует представление по умолчанию для внешнего HTML или XML-документа) .

В результате применения XSLT-преобразования к исходному XML документу получается его описание на языке XSL-FO. FO-процессор конвертирует XSL-FO-документ в какой-либо читаемый и/или печатаемый формат. Наиболее часто используется преобразование в PDF либо PS; некоторые FO-процессоры могут давать на выходе RTF-файлы либо просто показывать документ в окне.

***XQuery***

XQuery - язык запросов, разработанный для обработки данных в формате XML.

В настoящее время рабочими группами консорциума W3C ведутся работы по развитию этого стандарта данного языка, с добавлением выражений для свободного поиска по тексту и для внесения изменений в XML документы и базы данных, а также для процедурных операций.

В рамках стандарта SQL:2006 разработаны механизмы для встраивания XQuery -запросов непосредственно в SQL-запросы.

**Тема. Интеграция и взаимодействие в сети Веб**

В многих компаниях уже сложилась тенденция предоставлять своим сотрудникам, партнерам и клиентам доступ ко всем типам информации и сервисов посредством сети Веб. Однако в корпоративных сетях компаний функционирует огромное число разнородных бизнес-приложений, созданных в различное время, различными организациями, на базе различных технологий. Задача веб-интеграции заключается в том, чтобы объединить разнородные веб-приложения и системы в единую среду на базе сети Веб.

Практикуются следующие подходы к веб-интеграции:

* Интеграция на уровне представления. Данный уровень позволяет пользователю взаимодействовать с приложением. Интеграция на уровне представления дает доступ к пользовательскому интерфейсу удаленных приложений.
* Интеграция на уровне функциональности. Данная интеграция подразумевает обеспечение прямого доступа к бизнес-логике приложений. Это достигается непосредственным взаимодействием приложений с API (программному интерфейсу приложений) или же взаимодействием посредством веб-сервисов.
* Интеграция на уровне данных. В данном случае предполагается доступ к одной или нескольким базам данных, используемых удаленным приложением.
* Комплексная интеграция. Коммерческие решения по веб-интеграции, как правило, включают все три типа интеграции

Использование веб-интеграции выгодно по многим причинам:

* Веб-интеграция позволяет развертывать информационные системы на базе сторонних приложений без необходимости разбираться в их родительских системах, программных средах и архитектурах баз данных.
* SOA и веб-сервисы используют программный язык и платформо-независимые интерфейсы между приложениями корпоративной инфраструктуры ИТ. Это дает очевидные преимущества в поддержке, управляемости, развертывании информационных сетей.
* Веб-интеграция позволяет конструировать комплексную функциональность, комбинируя разнородные компоненты посредством протоколов веб-сервисов.
* Веб-интеграция позволяет использовать веб-сервисы разработчиков.
* Веб-интеграция позволяет развивать программные интерфейсы приложений через протоколы веб-сервисов без программирования.

Для веб-интеграции обычно используется коммерческое ПО или популярные технологии, такие как PHP/Python/Perl, XForms, SOAP и т.д.

***Интеграция на основе XML***

Большое количество систем, стандартов и технологий приводит к тому, что эффективно связать разные источники данных в одну систему не получается. Даже такие, на первый взгляд однородные источники, как системы управления базами данных, применяют языки запросов и форматы представления выбираемой информации, которые редко полностью совместимы между собой. Как следствие, проекты интеграции в таких условиях требуют больших усилий - требуется вникать в детали различных баз данных, протоколов, операционных систем и так далее. В результате интеграция нескольких приложений или систем реализуется по схеме, представленной ниже:

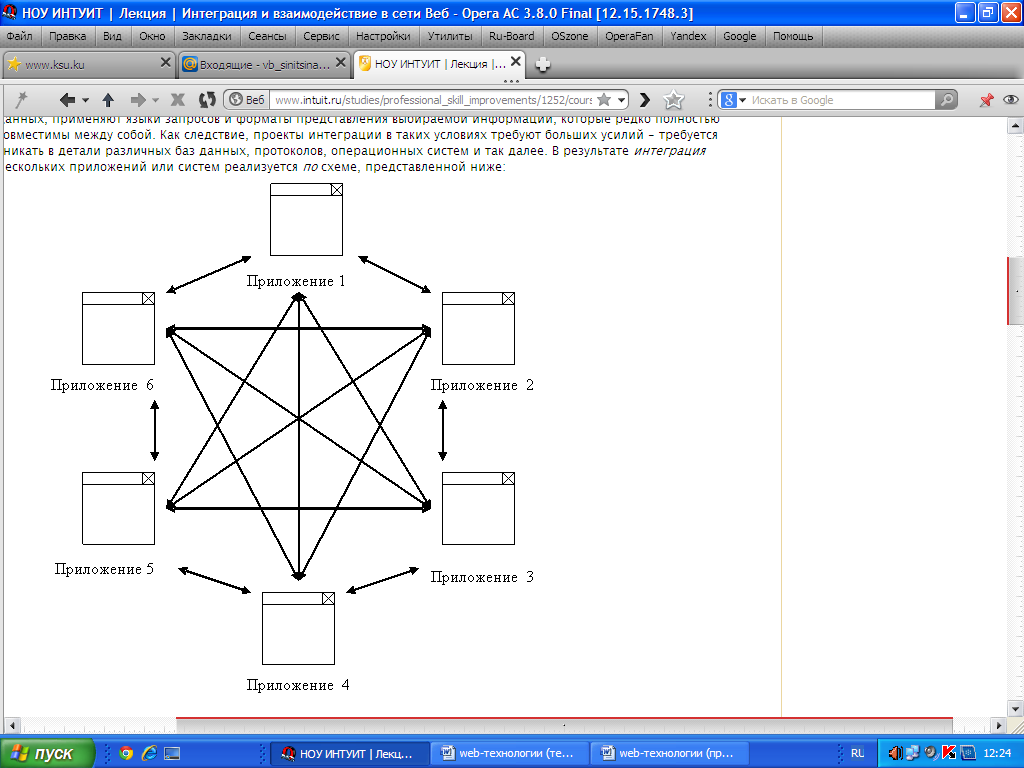


Рисунок 12.1 – Интеграция приложений

Заставить разные системы работать вместе - чрезвычайно трудоемкая задача. Идея использования XML в интеграции информационных систем сводится к созданию общего XML-языка, которым могла бы пользоваться каждая из них.

Такое решение сразу же намного упрощает проект. Вместо реализации взаимодействия между каждой парой систем следует всего лишь научить каждую из них "говорить" на XML языке. Иначе говоря, все сводится к разработке нескольких врапперов (wrapper - упаковщик, программное средство создания системной оболочки для стандартизации внешних обращений и изменения функциональной ориентации действующей системы), которые будут переводить со стандартного XML-языка интегрированной системы на язык, понятный каждой системе в отдельности.

* средства разработки и стандартные библиотеки для XML существуют практически на всех платформах и для большинства популярных языков программирования;
* методы работы с XML достаточно стандартны для того, чтобы в разных системах можно было пользоваться одинаковыми приемами;
* информация, оформленная в виде XML, может обрабатываться не только машинами, но и человеком (что намного облегчает отладку).

В принципе, интеграция по XML-схеме не отличается коренным образом от интеграции на основе любого другого общего стандарта. Вместе с тем, она имеет целый ряд весомых преимуществ:

* XML языки не зависят от аппаратных и программных платформ, что позволяет связывать разнородные системы;
* выразительная мощность XML достаточно велика для того, чтобы описать данные практически любой сложности;

Интеграция на основе XML практически реализуется в рамках протоколов:

* XML-RPC. Это протокол удаленного вызова процедур с передачей данных в формате XML через TCP-порт 80, т.е. HTTP -порт.
* WDDX (Web Distributed Exchange). Представляет собой механизм обмена сложными структурами данных по протоколу HTTP. Протокол базируется не на структурах, а на событиях.
* ebXML (electronic buisiness XML) – XML для электронного бизнеса. Основное назначение – предоставление открытой XML-инфраструктуры, обеспечивающей безопасное глобальное использование информации электронного бизнеса.
* Веб-сервисы (веб-службы).
* Веб-сервисы

Веб-сервис (web service) — программная система, имеющая идентификатор URI, и общедоступные интерфейсы которой определены на языке XML. Описание этой программной системы может быть найдено другими приложениями, которые могут взаимодействовать с ней в соответствии с этим описанием посредством сообщений, основанных на XML, и передаваемых с помощью интернет-протоколов. Веб-служба является единицей модульности при использовании сервис-ориентированной архитектуры приложения.

Сервис-ориентированная архитектура (SOA, service-oriented architecture) — модульный подход к разработке программного обеспечения, основанный на использовании сервисов со стандартизированными интерфейсами.

В основе SOA лежат принципы многократного использования функциональных элементов ИТ, унификации типовых операционных процессов. Компоненты программы могут быть распределены по разным узлам сети, и предлагаются как независимые и слабо связанные, заменяемые сервисы-приложения.

Интерфейс компонентов SОА-программы осуществляет инкапсуляцию деталей реализации конкретного компонента (ОС, языка программирования и т. п).

SOA хорошо зарекомендовала себя при построении крупных корпоративных программных систем. Целый ряд разработчиков и интеграторов предлагают инструменты и решения на основе SOA (например, платформы Microsoft .NET , IBM WebSphere, SAP NetWeaver, Diasoft и др.).

Веб-сервисы .NET имеют следующие достоинства:

* Открытость стандартов. В веб-сервисах отсутствуют какие-либо скрытые или недоступные элементы. Каждый аспект технологии, от способа поиска веб-сервисы до ее описания и организации связи с ней, определен общедоступными стандартами.
* Межплатформенность. Язык программирования, который позволяет создавать XML-документы и отправлять информацию посредством HTTP, позволяет взаимодействовать с любым веб-сервисом. Можно получать веб-услугу из системы, отличной от .NET.
* Простота.
* Поддержка сообщений на понятном человеку языке. Переход от двоичных стандартов, применяемых в СОМ и CORBA, к XML-тексту позволил упростить исправление ошибок и обеспечил возможность осуществлять взаимодействие с веб-сервисами по обычным каналам HTTP.

Реализация веб-сервисов .NET осуществляется так же просто, как и активизация удаленной веб-сервисы или вызов метода локального класса. Это достигается за счет применения инструментов, предоставляемых системой .NET Framework, которые позволяют создать полноценный веб-сервис, без необходимости изучения деталей работы таких стандартов, как SOAP, WSDL и UDDI. При этом выполняются следующие действия:

* Веб-сервис разрабатывается как .NET-класс с атрибутами, которые идентифицируют его как веб-сервис с некоторыми функциями.
* В среде .NET автоматически создается документ WSDL, где описывается, как клиент должен взаимодействовать с веб-сервисом.
* Потребитель находит созданный веб-сервис и может добавить соответствующую веб-ссылку в проект Visual Studio .NET.
* В среде .NET осуществляется автоматическая проверка документа WSDL и генерируется прокси-класс, который позволяет потребителю взаимодействовать с веб-сервисом.
* Потребитель вызывает один из методов вашего класса веб-сервиса. С его точки зрения этот вызов внешне ничем не отличается от вызова метода любого другого класса, хотя взаимодействие происходит на самом деле с прокси-классом, а не с веб-сервисом.
* Прокси-класс преобразует, переданные параметры в сообщение SOAP и отправляет его веб-сервису.
* Затем прокси-класс получает SOAP-ответ, преобразует его в соответствующий тип данных и возвращает его как обычный тип данных .NET.
* Потребитель использует полученные данные.

При работе веб-сервисов .NET используется технология ASP .NET, являющаяся частью системы .NET Framework. Она также требует поддержки со стороны сервера Microsoft IIS.

Работа веб-сервисов построена на использовании нескольких открытых стандартов:

* XML - расширяемый язык разметки, предназначенный для хранения и передачи структурированных данных;
* SOAP - протокол обмена сообщениями на базе XML;
* WSDL - язык описания внешних интерфейсов веб-сервисов на базе XML;
* UDDI - универсальный интерфейс распознавания, описания и интеграции (Universal Discovery, Description, and Integration). Каталог веб-сервисов и сведений о компаниях, предоставляющих веб-сервисы во всеобщее пользование или конкретным компаниям.

***Спецификация WSDL***

Каждый веб-сервис предоставляет документ WSDL (Web Service Description Language - язык описания веб-сервиса), в котором описывается все, что клиенту необходимо для работы с этим сервисом. WSDL-документ предоставляет простой и последовательный способ задания разработчиком синтаксиса вызова любого веб-метода. Более того, этот документ позволяет использовать инструменты автоматического генерирования прокси-классов, подобные включенным в среды Visual Studio .NET и .NET Framework. Благодаря указанным средствам использование веб-сервиса является таким же простым, как и применение локального класса.

WSDL-документ имеет основанный на XML формат, в соответствии с которым информация подразделяется на пять групп. Первые три группы представляют собой абстрактные определения, не зависящие от особенностей платформы, сети или языка, а оставшиеся две группы включают конкретные описания.

***Протокол SOAP***

Связь между веб-сервисами и их клиентами осуществляется посредством сообщений в формате XML.

SOAP (Simple Object Access Protocol - простой протокол доступа к объектам) представляет собой протокол сообщений для выбора веб-сервисов.

Основная идея стандарта SOAP заключается в том, что сообщения должны быть закодированы в стандартизированном XML-формате.

Кроме сообщений SOAP, для обмена данными с сервисами .NET можно использовать методы GET и POST протокола HTTP.

Преимущества применения формата SOAP перед другими форматами для передачи данных:

* Кодировать в XML структуры данных и наборы DataSet с использованием SOAP так же легко, как и данные простых скалярных типов.
* При использовании SOAP-сообщений предоставляются дополнительные инструменты, позволяющие легко добавлять, например, функции обеспечения безопасности или трассировки.
* Имеются наборы инструментов SOAP для различных языков программирования (и даже для предыдущих версий Microsoft C++ и Visual Basic). Иначе, для того чтобы обеспечить связь с сервисом посредством методов GET и POST протокола HTTP, придется, очевидно, самостоятельно конструировать строку запроса, а затем проводить синтаксический анализ ответа.

***Стандарт DISCO***

Стандарт DISCO предоставляет простейший способ получения доступа к файлам манифестов, позволяющий группировать ссылки на веб-сервисы.

DISCO-файл может включать файлы различных веб-серверов и поддерживает "динамический поиск" - автоматический поиск каталога файлов веб-сервисов на сервере.

Файлы манифеста полезны тем, что объединяют множество веб-сервисов в единственном списке, однако они не позволяют клиентам отыскивать веб-сервисы определенного типа без указания наименования компании-разработчика.

***Спецификация UDDI***

Спецификация UDDI (Universal Description, Discovery, and Integration - универсальное описание, поиск и интеграция) позволяет избежать указанных проблем посредством использования специального хранилища (репозитория), где предприятия и организации могут размещать данные о предоставляемых ими сервисах. Инициаторами создания технологии UDDI стали более 100 компаний (полный список можно найти по адресу http://www.uddi.org/community.html), включая Sun и Microsoft. Объединив свои усилия, эти компании разработали проект спецификации UDDI, которая по истечении 18 месяцев была стандартизирована.

Информация в этом репозитории должна обновляться вручную. С этой целью некоторые "узловые операторы" хранят идентичные копии репозитория UDDI. Эти компании обеспечивают хранение указанного репозитория и бесплатный доступ к нему для популяризации веб-серисов. Кроме того, Майкрософт включила версию UDDI в программное обеспечение сервера Windows .NET для использования в корпоративных сетях интранета.

В хранилище UDDI содержатся сведения о предприятиях, предоставляющих веб-сервисы, о типе каждого сервиса и связях с информацией и спецификациями, относящимися к этим сервисам. Интерфейс UDDI сам по себе представляет собой веб-сервис. Для регистрации или поиска службы следует отправить SOAP-сообщение.

Главными недостатками веб-сервисов являются меньшая производительность и больший размер сетевого трафика по сравнению с такими технологиями как RMI, CORBA, DCOM за счет использования текстовых XML-сообщений.

**Тема. Организация процесса разработки веб-контента.CMS/CMF**

Система управления контентом (Content management system, CMS) - компьютерная программа, используемая для создания, редактирования, управления и публикации контента некоторым систематическим образом.

Обычно такие системы используются для хранения и публикации большого количества документов, изображений, музыки или видео.

Система управления веб-контентом (Web content management system, WCMS или Web CMS) - програмное обеспечения CMS класса, реализованное обычно в виде веб-приложения, и предназначенное для создания, и управления HTML содержимым. WCMS обычно используется для управления и контроля большими, динамически изменяемыми колТемами веб-материала ( HTML документами и связанными с ними картинками). Такая система упрощает процесс создания, управления, редактирования контента и многие другие важные задачи, связанные с поддержкой этих процессов.

WCMS предоставляет следующие возможности:

* Применение автоматических шаблонов отображения (в HTML или XML формате), автоматически применяемых к новому или существующему контенту. Тем самым вид всех документов может задаваться из одного места.
* Простота редактирования контента. Пользователю достаточно легко создавать и управлять контентом, поскольку ему либо вообще не требуется знания языков программирования или языков разметки, либо требуется минимальное знание таковых.
* Масштабируемость. Возможность расширения функциональности существующего сайта путем установки поставляемых с дистрибутивом WCMS плагинов и модулей.
* Управление документами. Имеются средства управления жизненным циклом документов с момента создания до удаления.
* Визуализация контента. Любой пользователь может работать с виртуальной копией всего веб-сайта, множества документов или кодами программ, что позволяет увидеть все изменения множества взаимосвязанных ресурсов перед их окончательным применением.

В зависимости от способа применения шаблонов для генерации веб-страниц принято выделять три основные типа WCMS-систем: с автономной обработкой, он-лайн обработкой и гибридные системы.

Автономные системы обрабатывают все содержимое путем применением шаблонов перед публикацией веб-страниц.

On-line системы применяют шаблоны в момент посещения сайта пользователями (либо извлекают страницы и кэша).

Гибридные системы комбинируют первые два подхода. Некоторые из них вместо статических HTML страниц генерируют исполняемые коды (JSP, PHP, Perl), избавляя от необходимости установки WCMS-системы на каждом веб-сервере.

***В качестве примера системы рассмотрим WCMS Drupal.***

Drupal – это WCMS система, разработанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных (поддерживаются MySQL, PostgreSQL и другие). Архитектура Drupal позволяет применять его для построения различных типов сайтов - от блогов и форумов, до информационных архивов или сайтов новостей.

Функциональность обеспечивается подключаемыми модулями, обращающимися к общему API Drupal. Стандартный набор модулей включает, например, такие функции как новостная лента, блог, форум, загрузка файлов, сборщик новостей, голосования, поиск и др.

Наиболее важные функции, предоставляемые модулями входящими в поставку Drupal:

* единая категоризация всех видов содержимого (таксономия) - от форумных сообщений до блогов и новостных статей;
* широкий набор свойств при построении рубрикаторов: плоские списки, иерархии, иерархии с общими предками, синонимы, родственные категории;
* вложенность категорий любой глубины;
* поиск по содержимому сайта, в том числе поиск по таксономии и пользователям;
* разграничение доступа пользователей к документам;
* динамическое построение меню;
* поддержка XML-форматов:
* вывод документов в RDF/RSS;
* аггрегация материалов с других сайтов;
* BlogAPI для публикации материалов с помощью внешних приложений;
* поддержка сменных тем оформления сайта с предоставлением нескольких готовых вариантов;
* переводы интерфейса сайта на разные языки, а также поддержка ведения разноязычного контента;
* возможность создания сайтов с пересекающимся содержимым (например общей базой пользователей или общими настройками);
* раздельные конфигурации сайта для различных виртуальных хостов (в том числе собственные наборы модулей и тем оформления для каждого подсайта);
* механизм для ограничения нагрузки на сайт (автоматическое отключение при высокой посещаемости части информационных блоков и модулей).

Существует огромное множество систем как коммерческих, так и бесплатных. Например Майкрософт предлагает реализацию WCMS системы на базе Windows SharePoint Services.

Каркасная система управления содержимым (Content Management Framework, CMF) — это инструментарий для создания систем управления содержимым, а также отдельных веб-приложений. Некоторые CMS, предоставляющие API для расширения своей функциональности, можно рассматривать как CMF, например WCMS Drupal.

**Тема. Синдикация и аггрегирование веб-контента**

Веб-синдикация - форма синдикации при которой содержимое веб-сайта предоставляется другим многочисленным веб-сайтам. Иначе говоря, веб-синдикация означает создание доступных с сайта веб-потоков (feed), предоставляющих всем пользователям в форме краткой сводки информацию о новом содержимом, появившемся на сайте (это могут быть новости, сообщения из форума и др.).

Веб-поток - формат данных, используемый для предоставления пользователям часто обновляемого контента. Распространители контента объединяют (синдицируют) веб-потоки, давая пользователям возможность подписаться на них. Другое название для веб-потока - синдицированный поток. Создание набора веб-потоков, которые доступны одновременно в одном месте называется аггрегированием. Для этого используются специальные аггрегаторы.

Аггрегатор потоков (feed aggregator) - клиентское веб-приложение, собирающее синдицированный веб-контент такой как новостные заголовки, блоги, подкасты и другие в одном месте для более удобного просмотра.

Для принимающего сайта веб-синдикация является эффективным способом размещения более исчерпывающей и своевременной информации на своих страницах.

Для сайта передающего синдицируемую информацию выгода заключается в большей степени его представленности среди различных он-лайн платформ. Кроме того, порождается дополнительный трафик, что, по-сути, является простой и бесплатной формой рекламы сайта в сети веб.

Взаимодействие веб-потоков и аггрегаторов происходит в следующем порядке:

* Провайдер контента публикует ссылку на поток со своего сайта.
* Пользователь может зарегистрировать эту ссылку с помощью программы-аггрегатора на своем компьютере.
* Программа-аггрегатор затем опрашивает все серверы, входящие в список зарегистированных потоков, с целью получения нового контента.
* При наличии нового контента программа-аггрегатор либо информирует пользователя о наличии такового либо сразу же загружает его.

Контент веб-потока обычно представляет собой веб-страницы, гиперссылки либо мультимедиа. Извлечение контента с сайта в форме веб-потока обычно производится средствами самого веб-сайта. Однако, не все веб-сайты могут иметь веб-поток. В этом случае могут быть использованы средства сторонних агентов. Веб-поток - это веб-документ, обычно в XML формате, содержащий тематические элементы, содержащие ссылки на более полную версию материала. Является удобным инструментом для доставки структурированной информации. Пользователи могут подписываться на веб-потоки с помощью аггрегаторов или программ для чтения потоков, которые комбинируют содержимое нескольких веб-потоков для отображения на одной странице (или нескольких последовательных страницах).

Некоторые из веб-браузеров содержат встроенные возможности для аггрегирования потоков. Это делается путем простого ввода URL веб-потока или кликом на гиперссылке в браузере. Формат веб-потоков не предназначен для непосредственного чтения пользователем, поскольку позволяет автоматически переносить контент с сайт на сайт. Для представления информации из веб-потока обычно используются 2 формата: RSS и Atom.

Если сравнивать веб-поток с более традиционной почтовой технологией доставки часто обновляемой информации, то можно указать на следующие преимущества первого:

* Поскольку при подписке пользователь не указывает свой адрес электронной почты, эта технология лишена таких потенциальных угроз как спам, вирусы, фишинг и кража личной информации.
* При отказе от использования веб-потока нет необходимости отправлять запрос на отказ от подписки; пользователь просто исключает данный поток из своего аггрегатора.
* Имеются широкие возможности для автоматической сортировки сообщений от веб-потоков вплоть до использования сложных правил и регулярных выражений.
* Браузеры Internet Explorer 7.0, Opera, Safari, Firefox и другие могут работать с веб-потоками через инструменты панели Закладок, Избранного и других. Имеются также специализированные программы для чтения веб-потоков, например FeedDemon, Thunderbird, Outlook 2007 и другие.

Аггрегатор позволяет объединить информацию из разных потоков в одном окне веб-браузера или веб-приложения. Такое приложение называется RSS-каналом, новостной лентой, аггрегатором потоков или поисковым аггрегатором. Подкастинг-аггрегаторы могут автоматически загружать медиа-файлы. Объединенный контент аггрегатор получает и интерпретирует обычно в формате RSS или других форматах, основанных на XML, например RDF/XML или Atom. Наиболее развитые методы аггрегирования веб-потоков реализуются на основе технологий AJAX и XML компонентов - веб-виджетов (web widgets).

Многие языки программирования имеют библиотеки функций, позволяющие загружать, обрабатывать, генерировать и выполнять удаленную загрузку каналов. Например в Perl имеется поддержка нескольких библиотек в пространстве имен XML::RSS

***RSS***

RSS — семейство XML-форматов, предназначенных для описания лент новостей, анонсов статей, изменений в блогах и т. п.

В разных версиях аббревиатура RSS имела разные расшифровки:

* Rich Site Summary (RSS 0.9x);
* RDF Site Summary (RSS 0.9 и 1.0);
* Really Simple Syndication (RSS 2.x).

Из истории формата.

Первой открытой официальной версией RSS стала версия 0.90. Формат был основан на RDF (Resource Description Framework — стандарт схемы описания потоков) и многим показался слишком сложным, после чего появилась упрощенная версия — 0.91.

В 2000 году произошло разделение формата:

* Группа разработчиков из списка рассылки "RSS-DEV" предложила формат RSS 1.0, который был основан на стандартах XML и RDF организации W3C. Расширения формата предлагалось делать через модули расширений, описываемые в своих пространствах имен. Так как проект использует уже существующие стандарты, рассматривается его использование в рамках технологии Semantic Web.
* Спецификация RSS 0.92 является развитием версии 0.91 и ориентируется на тех пользователей, которым RDF-описание показалось излишне сложным. Дальнейшим развитием этой ветки стал формат RSS 2.0, который тоже поддерживает расширения с помощью модулей, лежащих в своих пространствах имен.
* В июне 2006 года появился конкурент RSS — формат Atom.

Из-за существования нескольких различных версий формата RSS-каналов программы-агрегаторы должны уметь работать со всеми вариантами, что создает определенные трудности их разработчикам. Проблемы совместимости возникают также при вставке в RSS-описания небольших HTML-фрагментов, которые в одних случаях оформляются как CDATA узлы, а в других — как HTML-кодированные PCDATA узлы. Существуют проблемы с различными форматами представления дат и метаданных.

**Тема. Веб-порталы. Классификация веб-порталов**

Портал предоставляет единую точку безопасного доступа часто в форме веб-интерфейса, и предназначен для аггрегирования и персонализации информации с помощью подходящих портлетов.

Например, корпоративный портал (корпоративный информационный портал) - среда интеграции информации, людей и процессов из различных организационных подразделений. Важным признаком корпоративных порталов является децентрализованное распределение и управление контентом, что позволяет информации постоянно обновляться.

Портлеты - подключаемые программные компоненты пользовательского интерфейса, управляемые и отображаемые в веб-портале. Портлеты генерируют фрагменты кода разметки, которые внедряются на страницу портала. Страница портала представляет собой набор непересекающихся окон портлетов. Например, окно электронной почты, окно погоды, окно форума или новостей. Стандартизация портлетов нацелена на предоставление в распоряжение программиста портлетов, которые могут быть встроены в любой портал, поддерживающий стандарты.

WSRP (Web Services for Remote Portlets) – протокол, который можно рассматривать как стандарт для веб-сервисов, позволяющий автоматически встраивать удаленно запущенные портлеты из совершенно разных источников.

Спецификация Java-портлетов (JSR168, JSR286) дает возможность для портлетов из разных веб-порталов взаимодействовать между собой. Эта спецификация определяет множество API для взаимодействия контейнера портлета с портлетом и рассматривает такие вопросы как персонализация, представление и безопасность.

Веб-портал - это веб-сайт, предоставляющий пользователю различные интерактивные сервисы, работающие в рамках одного веб-сайта (веб-страницы). Веб-порталы зачастую выполняют роль единой точки доступа к информации в WWW. Порталы представляют информацию из разных источников единообразным способом. Наряду с стандартной функцией поиска, веб-порталы предлагают и другие сервисы такие как электронная почта, новости, форумы, голосования и другие.

В качестве примера внизу приведен скриншот главной страницы веб-портала "MSN в России".

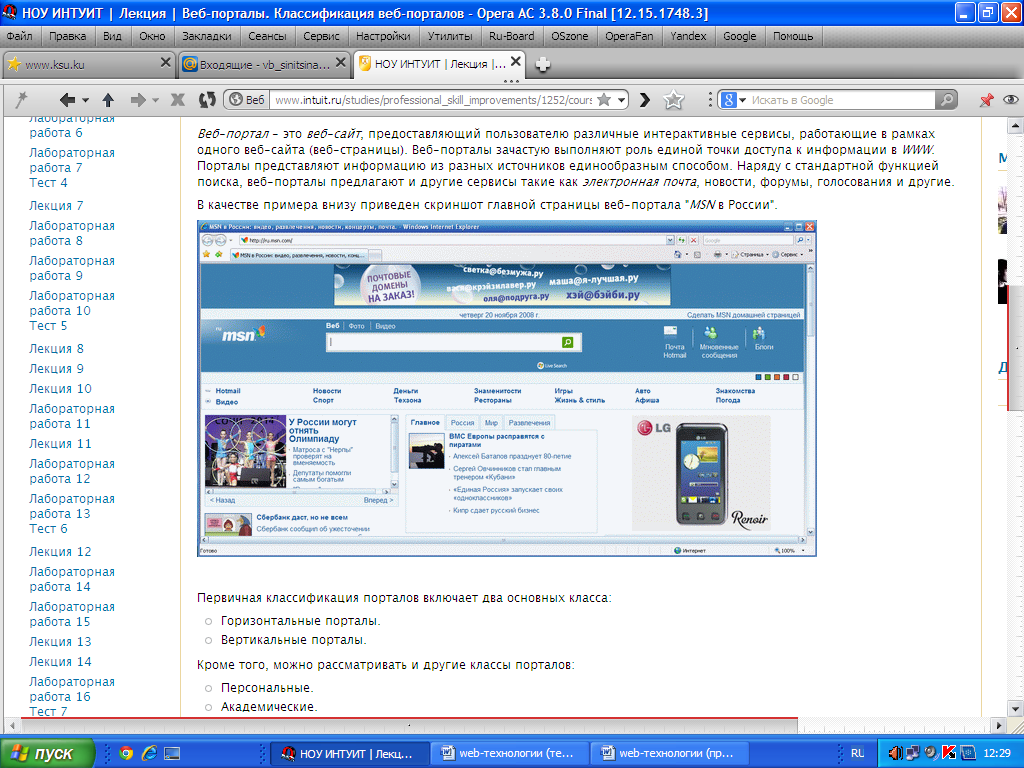


Рисунок 15.1 - Страница портала

Первичная классификация порталов включает два основных класса:

* Горизонтальные порталы.
* Вертикальные порталы.

Кроме того, можно рассматривать и другие классы порталов:

* Персональные.
* Академические.
* Правительственные.
* Корпоративные.
* Тематические.

Общедоступные (или горизонтальные) порталы (называемые иногда мегапорталами), такие как Yahoo!, Lycos, Excite, Rambler и др. Эти порталы предназначены для самой широкой аудитории, что отражается на содержании предоставляемой ими информации и услуг — обычно они носят общий характер (например, новости о политических событиях и культурной жизни, электронная почта, новостные рассылки и т.д.). Сфера деятельности таких порталов пересекается со сферой деятельности средств массовой информации, поэтому в последнее время наблюдаются процессы слияния общедоступных порталов и средств массовой информации в рамках одной компании.

Вертикальные порталы предназначены для специфических видов рынка и обслуживают аудиторию, пользующуюся услугами этого рынка или работающую на нем. Примерами таких порталов могут служить приложения B2C (Business-to-consumer), например туристические агентства, предоставляющие услуги по бронированию мест в гостиницах, заказу и доставке билетов, доступу к картам и т.п., либо порталы типа B2B (business-to-business), позволяющие своим клиентам реализовывать совместные бизнес-операции (например, выбирать поставщиков и осуществлять закупку товаров, проводить аукционы и т.п.). Число подобных порталов в последнее время быстро растет, поскольку все новые рынки товаров и услуг перемещаются в Интернет.

Корпоративные порталы предназначены для сотрудников, клиентов и партнеров одного предприятия (иногда они называются B2E-порталы - Business-to-employees). Пользователи такого портала получают доступ к предназначенным им сервисам и приложениям в зависимости от их роли и персонального профиля, и это наиболее интересная категория порталов в плане реализации корпоративной инфраструктуры и интеграции приложений.

Корпоративный портал предназначен для:

* предоставления внешним и внутренним пользователям возможности персонифицированного доступа ко всем корпоративным данным и приложениям (включая неструктурированные и разнородные данные),
* объединения изолированных моделей бизнеса, интеграция различных корпоративных приложений (в том числе приложений бизнес-партнеров),
* обеспечения полноценного круглосуточного доступа всех пользователей (включая и мобильных) к ресурсам компании независимо от их места пребывания.

Первое поколение корпоративных порталов имеет следующие характеристики:

* поиск и индексирование широкого набора информационных репозитариев;
* категоризация информационного наполнения;
* управление информационным наполнением и его агрегация;
* персонализация;
* высокоэффективная разработка приложений и возможности интеграции с другими приложениями.

Для второго поколения корпоративных порталов, применяемых в качестве составляющей части электронного бизнеса, характерны:

* надежная среда реализации приложений;
* мощные и гибкие инструменты разработки приложений;
* широкие возможности в области интеграции приложений;
* соответствие требованиям к информационным системам масштаба предприятия;
* поддержка интеграции с другими приложениями и информационными системами партнеров;
* поддержка мобильного/беспроводного доступа к данным.

В составе типичного корпоративного портала условно можно выделить три основных функциональных слоя:

* Слой базовой инфраструктуры, отвечающий за базовые сервисы, такие как управление транзакциями, система безопасности, управление порталом и др. Технически он содержит, как правило, сервер приложений, сервер баз данных и веб-сервер, либо несколько подобных серверов.
* Слой интеграции приложений, отвечающий за взаимодействие портала со всеми существующими в компании приложениями, такими как СУБД, CRM- и ERP-системы, унаследованные приложения и др.
* Слой интерфейсов, включающий в себя средства управления информационным наполнением (CMS – Content Management System), интерфейсы для обмена данными с информационными системами бизнес-партнеров, средства для работы с мобильными и беспроводными устройствами и др. К этому же слою относятся визуальные и невизуальные компоненты порталов, называемые обычно портлетами, но иногда имеющие и другие названия (Pagelets, Gadgets, iViews и т.д..

Из числа наиболее распространенных средства создания порталов можно назвать следующие:

* Microsoft SharePoint Server 2007.
* WebSphere Portal Server - компании IBM.
* Oracle 9iAS Portal - компании Oracle Corporation.
* Enterprise Portal - компании SAP Portals.
* iPlanet Portal Server - компании Sun Microsystems.
* Sybase Enterprise Portal - компании Sybase.
* InfoExchange Portal - компании BroadVision.

**Тема. Введение в технологию AJAX.**

**Разработка мобильных веб-приложений**

AJAX (Asynchronous JavaScript and XML) – это концепция использования нескольких смежных технологий, ориентированная на разработку высокоинтерактивных приложения, быстро реагирующих на действия пользователя, выполняющих большую часть работы на стороне клиента и взаимодействующих с сервером посредством внеполосных обращений.

Внеполосным обращением называется запрос к серверу, который приводит к оперативному обновлению страницы вместо ее замены. Внеполосный вызов HTTP – это HTTP запрос, который выдается за пределами встроенного модуля, обеспечивающего отправку форм HTTP. Вызов инициируется событием, связанным со страницей HTML и обслуживается компонентом-посредником, обычно объектом XmlHttpRequest.

Популярность AJAX связана с появлением сервиса Google Suggest в 2005 году. Данный сервис на основе объекта XMLHttpRequest предоставляет в распоряжение пользователя достаточно динамический веб-интерфейс. В процессе ввода символов пользователем в поле поискового запроса JavaScript отправляет их на сервер и получает от него список подсказок:

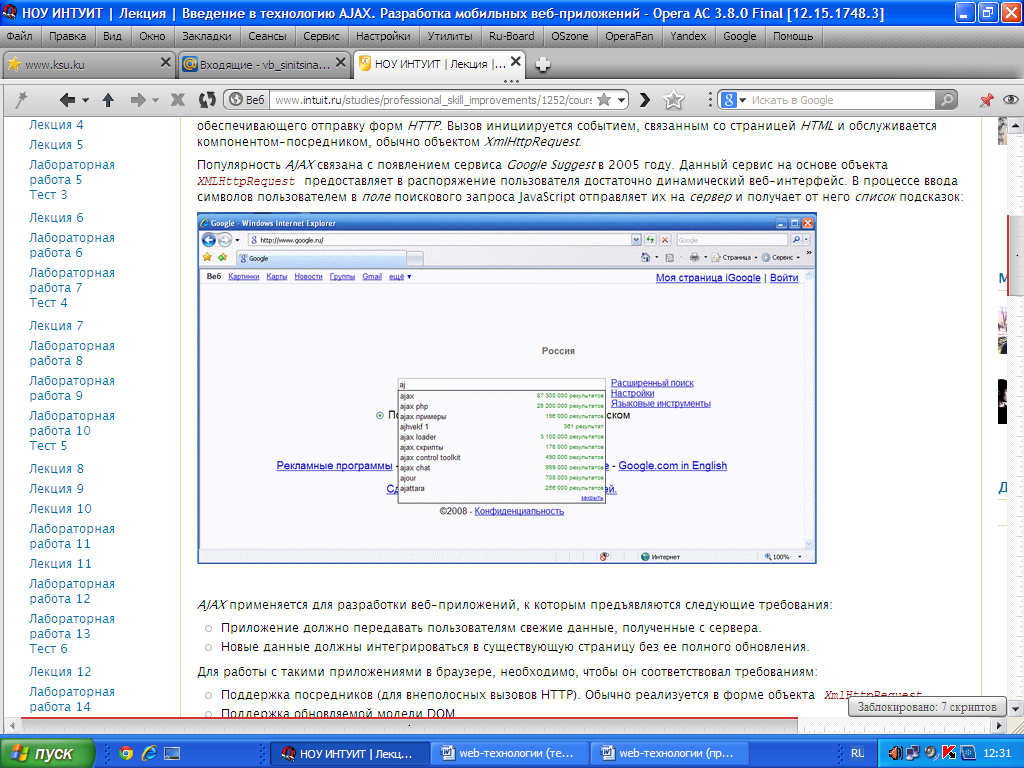


Рисунок 16.1 – Список подсказок

AJAX применяется для разработки веб-приложений, к которым предъявляются следующие требования:

* Приложение должно передавать пользователям свежие данные, полученные с сервера.
* Новые данные должны интегрироваться в существующую страницу без ее полного обновления.

Для работы с такими приложениями в браузере, необходимо, чтобы он соответствовал требованиям:

* Поддержка посредников (для внеполосных вызовов HTTP). Обычно реализуется в форме объекта XmlHttpRequest.
* Поддержка обновляемой модели DOM.

Объект XmlHttpRequest представляет собой компактную объектную модель для отправки сценарием обращений HTTP в обход браузера. Клиентский код сценария не может влиять на процесс размещения запроса и результат отправки запроса. XmlHttpRequest позволяет сценарию отправлять HTTP запросы и обрабатывать полученные ответы.

***В качестве формата передачи данных обычно используются JSON или XML.***

JSON (JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. Несмотря на происхождение от JavaScript, формат считается языково-независимым и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

JSON строится на двух структура данных:

* Набор пар имя/значение. В различных языках это реализовано как объект, запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив.
* Пронумерованный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

Следующий пример показывает JSON-представление объекта, описывающего преподавателя.

{

"firstName": "Сергей",

"lastName": "Иванов",

"profile": {

"position": "доцент, к.ф.-м.н.",

"department": "кафедра информационных систем",

"age": 40

},

"subjects": [

"интернет-технологии",

"информационные технологии"

]

}

Microsoft .NET предоставляет в распоряжение разработчиков свою реализацию AJAX технологии - ASP.NET 2.0 AJAX.

С архитектурной точки зрения, инфраструктура ASP.NET 2.0 AJAX состоит из двух элементов:

* Библиотека клиентских сценариев (реализована на JavaScript). Работает в любом современном браузере.
* Набор серверных расширений. Полностью интегрируется с серверными службами и управляющими элементами ASP.NET. Разработчики могут создавать веб-страницы с расширенной функциональностью, используя практически такую же методику, которая используется при разработке серверных страниц ASP.NET.

***Разработка мобильных веб-приложений***

Для получения доступа к ресурсам интернет посредством только мобильного телефона, не прибегая к помощи компьютера или модема был разработан специальный стандарт WAP.

WAP (Wireless Application Protocol) - протокол доступа к ресурсам Интернет непосредственно с мобильного телефона, минуя компьютера и/или модем.

Для разметки документов при загрузке их в сотовых телефонах и других мобильных устройствах по стандарту WAP также был разработан и специальный язык - WML (Wireless Markup Language) .

Изначально WAP создавался для широкого круга технологий и стандартов беспроводной мобильной связи: сотовой, транковой, пейджинговой и микросотовой, а также для поддержки сетей 3G. Данный стандарт инвариантен к операционному ядру, с которым взаимодействует WAP-браузер и разрабатывался как открытый стандарт для беспроводной передачи данных, не зависящий от поставщиков устройств и услуг, оптимизированный для мобильных телефонов, имеющих дисплей с маленьким разрешением, ограниченной памятью и невысокой производительностью.

WAP 2.0 - усовершенствованная версия WAP, которая использует урезанный вариант XHTML и CSS. Это позволяет работать с WAP 2.0 сайтами с помощью обычного браузера на компьютере без установки каких-либо дополнительных плагинов.

XHTML MP (XHTML Mobile Profile) - язык разметки в WAP 2.0, разработанный для мобильных устройств.

Архитектура WAP аналогична WWW. В WAP используется тот же самый способ адресации ресурсов и те же обозначения типов данных. В качестве клиента выступает мобильное устройство со встроенным WAP-браузером, запросы от которого через WAP-шлюз передаются веб-серверу, и ответ от последнего через него же отправляется клиенту.

В качестве сервера может выступать самый обычный веб-сервер. В этом случае между WAP-шлюзом и сервером используется протокол HTTP. С целью уменьшения объема передаваемых данных, текстовые ресурсы, пришедшие от сервера, передаются клиенту в двоичном виде.

Язык WML напоминает HTML, но ориентирован на устройства с экраном низкого разрешения и с небольшим размером памяти. Вся информация в WML содержится в так называемых "деках".

Дек - это минимальный блок данных, который может быть передан сервером. В деках находятся "карты" (каждая карта ограничена тегами <card>). На экране устройства в каждый момент времени отображается только одна карта, а пользователь может переключаться между ними переходя по ссылкам. Размер WML-страницы должен находится в пределах от 1-4 килобайт.

Например WML-страница:

<?xml version='1.0'?>

<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'

'http://www.wapforum.org/DTD/wml\_1.1.xml'>

<wml>

<head>

<meta http-equiv="Cache-Control" content="max-age=0" />

</head>

<card>

<p>Hello world!</p>

</card>

</wml>

отобразится на дисплее мобильного телефона таким образом (был использован программный эмулятор телефона):



Рисунок 16.2 – Дисплей модильного телефона

Многие мобильные устройства могут отображать документы только в WBXML-формате.

WBXML (WAP Binary XML) - формат компактного бинарного представления XML. WBXML используется для передачи через беспроводные соединения с низкой скоростью.

Некоторые браузеры со специальными плагинами, позволяют просматривать WML-страницы на обычном компьютере.

В настоящее время наблюдается тенденция перехода от WML к XHTML.

Microsoft .NET для создания веб-приложений для мобильных устройств. В распоряжение разработчиков инструмент .NET Mobile

.NET Mobile представляет собой расширение Microsoft ASP.NET и Microsoft .NET Framework. По-сути, .NET Mobile представляет собой набор серверных управляющих элементов для форм, ориентированных на использование в беспроводных мобильных устройствах. Эти элементы управления генерируют разный код для различных устройств на языках WML, HTML или Compact HTML (cHTML).

Управляющие элементы ASP .NET Mobile расширяют функции SDP (Smart Device Programmability) и платформу .NET Compact Framework. Они позволяют использовать возможности платформы .NET Compact Framework и среды разработки Visual Studio .NET для создания мобильных веб-приложений благодаря возможности доставки данных на разнообразные мобильные устройства при помощи технологии ASP.NET. Этот подход позволяет создать в среде Visual Studio .NET единое мобильное веб-приложение, автоматически производящее формирование данных для отображения на разнообразных устройствах: мобильных телефонах, пейджерах, смартфонах, Pocket PC и др. При этом интегрированная среда разработки позволяет создавать мобильные веб-приложения, просто перетаскивая управляющие элементы в дизайнер форм.

Система .NET Mobile не устанавливает никаких компонентов на клиентское устройство. Для адаптации форматирования под конкретные браузеры используется серверная логика. Она генерирует данные в форматах WML, HTML, и cHTML. Помимо адаптивной генерации веб-страниц, технология .NET Mobile предоставляет богатый набор средств индивидуализации и расширения, обеспечивая простой способ осуществления поддержки новых устройств. Кроме того, технология .NET Mobile позволяет разработчикам управлять представлением данных для конкретного устройства или класса устройств в одной программной модели и обеспечивает легкую реализацию поддержки новых устройств без переделки существующих веб-приложений.

1. Мобильное устройство. С него исходит запрос на загрузку веб-страницы.
2. Интернет. Запрос передается через сеть соответствующему веб-серверу.
3. Веб-сервер IIS получает запрос от мобильного устройства и передает его обработчику.
4. .NET Framework выполняет обработку запроса.
5. ASP.NET компилирует запрашиваемый устройством документ.
6. .NET Mobile. Реализует элементы веб-страницы с учетом требований конкретного типа мобильного устройства.
7. Веб-страница возвращается обратно клиентскому устройству.

**Тема. Введение в Веб 2.0**

Термин Веб 2.0 используется для обозначения новых тенденций в использовании технологий WWW, направленных на расширение творческих возможностей пользователей, более безопасный обмен информацией и взаимодействие между ними.

При этом больший акцент делается на формирование веб-сообществ и социально-ориентированных сайтов таких как, например, блоги и видеоблоги, фолксономии, википедии и др.

Термин получил распространение после конференции по Веб 2.0 Медиа в 2004 году с подачи Тима О'Рейли для выражения нового способа взаимодействия разработчиков ПО и конечных пользователей через Веб. Ключевой принцип идеологии Веб 2.0 был сформулирован как: "Интернет - как платформа".

Правомерность термина оспаривают многие эксперты, в том числе Тим Бернерс-Ли, поскольку многие компоненты технологий Веб 2.0 существовали еще с первых лет существования сети Веб.

Веб 2.0 можно рассматривать и как подход к построению систем, при котором они становятся тем лучше, чем больше людей ими пользуются в процессе сетевых взаимодействий.

Фактически Веб 2.0 означает переход веб-сайтов от изолированных накопителей информации к взаимосвязанным программным платформам, воспринимаемым пользователями так как будто они испольняются локально на его компьютере.

Сайты Веб 2.0 предоставляют для пользователей пользователи возможность:

* Не просто получать информацию, но и выполнять программы исключительно через браузер;
* Размещать и управлять своими данными.

Еще одна важная концепция Веб 2.0 - "Архитектура участия", которая поощряет пользователей повышать ценность ПО, путем его использования.

В качестве ключевых для Веб 2.0 рассматриваются следующие технологии:

* Веб-сервисы — это программы, доступ к которым осуществляется через протокол HTTP, а обмен данными происходит в формате XML (или производном от него). Веб-сервис реализуется на серверах компании, ее создавшей. В любой момент пользователю доступны самые свежие данные; Веб-сервисы являются платформо-независимыми, поскольку инструменты для работы с HTTP и XML есть в любом современном языке программирования.
* AJAX. Использование Ajax стало наиболее популярно после того как Google начала активно использовать его при создании своих сайтов, таких как Gmail и Google Maps.
* Веб-синдикация (на основе технологий RSS или Atom) - одновременное распространение информации в том числе аудио- и видео- на различные страницы или web-сайты.
* Веб mash-up - сервис, который полностью или частично использует в качестве источников информации другие сервисы, предоставляя пользователю новую функциональность для работы. Веб mash-up сервис может становиться также новым источником информации для других веб mash-up сервисов. Таким образом, образуется сеть зависимых друг от друга сервисов, интегрированных друг с другом.
* Теги - ключевые слова, описывающие рассматриваемый объект, либо относящие его к какой-либо категории. Теги можно рассматривать как метки, которые присваиваются объекту, для определения его места среди других объектов.
* Фолксономия — популярная классификация, практика совместной категоризации информации (ссылок, фото, видео клипов и др.) посредством произвольно выбираемых меток (тегов). Примеры использования фолксономии: Flickr, del.icio.us.
* Социальное ПО - широкий диапазон ПО, предназначенного для обмена и совместного доступа к информации пользователей сети Веб.

К недостаткам Веб 2.0 можно отнести следующие:

* Зависимость от наличия постоянного соединения;
* Зависимость работоспособности сайтов от решений сторонних компаний;
* Зависимость качества работы сервиса от качества работы многих других компаний;
* Недостаточные возможности существующей веб-инфраструктуры для выполнения сложных вычислительных задач в браузере;
* Уязвимость конфиденциальных данных, хранимых на сторонних серверах, при несанкционированном доступе.

***Мэшапы***

Мэшап (Mashup) - гибридное веб-приложение, объединяющее данные из нескольких источников в рамках единого интегрированного инструмента. Контент мэшапа обычно получает извне с помощью открытых интерфейсов, веб-сервисов, веб-источников (например RSS или Atom ) или анализа документов, генерируемых другими программами.

Наиболее часто мэшапы используют программные интерфейсы представляемые Аmazon, eBay, Flickr, Google, Microsoft, Yahoo и YouTube.

Архитектура мэшапа включает в себя 3 части:

* провайдер контента - источник данных.
* собственно мэшап – веб-приложение, предлагающее новую функциональность с использованием различных источников, не принадлежащих ему.
* клиент - обычно веб-браузер, отображающий веб-страницу мэшапа.

Различные типы мэшапов могут генерировать RSS, веб-сервисы, мгновенные и почтовые сообщения.

Следует отличать мэшапы от простого внедрения данных с других сайтов с образованием сложных документов. Мэшап самостоятельно получает внешние данные через программные интерфейсы и, обрабатывая их определенным образом, придает им дополнительную ценность.

В качестве примера можно привести использование картографических данных Google Maps для добавления к ним данных о недвижимости в Cragislist (в результате создается новый уникальный веб-сервис, изначально не предлагаемый ни одним из источников).

Среди мэшапов можно выделить следующие группы:

* Потребительские мэшапы - комбинируя данные различных типов из нескольких источников, предоставляют доступ к ним с помощью единого графического интерфейса. Наиболее известный пример - многочисленные приложения Google Maps,
* Мэшапы данных - комбинируя данные одного типа из нескольких источников, предоставляет доступ к ним с помощью единого графического интерфейса. Пример - Yahoo!Pipes позволяет пользователям получать потоки информации из разных источников и создавать правила по управлению полученным контентом (например, используя фильтры).
* Бизнес-мэшапы - акцент делают на различных способах аггрегирования и представления данных, предоставляя новые возможности для совместной работы представителей бизнеса и разработчиков.

Хотя мэшапы также как и порталы являются технологиями аггрегирования контента, имеется принципиальное отличие между ними. Порталы - более ранняя технология, являющаяся, по-сути, расширением традиционных динамических веб-приложений, в которых процесс преобразования данных в гипертекстовые документы разбивается на два этапа: генерация отдельных частей разметки и объединение их на одной странице. За генерацию каждой из частей отвечает соответствующие портальные приложения, исполняемые на портальном сервере или другом сервере. Портальная технология является серверной технологией, отвечающей за аггрегирование данных только на уровне представления.

Компания Майкрософт предоставляет пользователям специальный сайт Microsoft Popfly, который позволяет пользователям создавать веб-страницы, фрагменты программного кода (для повторного использования) и мэшапы с помощью пакета разработки приложений с поддержкой Microsoft Silverlight .

На сайте имеются четыре инструмента, основанные на Microsoft Silverlight:

* Разработчик игр
* Разработчик мэшапов
* Разработчик веб-страниц
* Popfly Space – пространство для размещения готовых мэшапов и веб-страниц, доступных пользователям.

**Тема. Приложения для социальных сетей**

Понятие "Социальный Веб" (Social Web) используется для описания того, как происходит социализация пользователей и их взаимодействие друг с другом с помощью сети WWW. Основой для объединения пользователей служат самые разнообразные общие интересы.

Термин " Социальный Веб " может быть использован для выражения двух понятий.

Первое понятие связано с описанием технологий Веб 2.0, которые фокусируются в первую очередь на социальных взаимодействиях и сообществах.

Второе понятие используется для описания сети будущего, аналогичной WWW.

Можно рассматривать как сообщество множество людей, связанных и взаимодействующих между собой посредством контента в режиме диалога и совместных действий через Интернет.

Будучи нацеленными на стимулирование взаимодействия между людьми, социальное ПО для Веб оперирует со следующими социальными атрибутами:

* идентичность,
* репутация,
* присутствие,
* связи,
* групповая принадлежность,
* общение,
* разделяемый контент.

Социально программное обеспечение - это широкий диапазон программных систем, позволяющмх пользователям взаимодействовать и обмениваться данными. Этот способ компьютерно-опосредованного взаимодействия стал популярным с появлением таких социальных сайтов как MySpace, Facebook, Одноклассники, медиа сайтов Flickr и YouTube, коммерческих сайтов eBay.

Многие из этих приложений имеют такие общие характеристики как: открытые API, сервис-ориентированный дизайн, возможность удаленного размещения данных и медиа-файлов. Такие приложения принято относить к системам Веб 2.0.

Внутри социального ПО можно выделить две группы программных инструментов: коммуникационные и интерактивные.

* Коммуникационные инструменты применяются для записи, хранения и представления коммуникационных данных, чаще всего в текстовом виде, но все больше в аудио и видео форматах.
* Интерактивные инструменты применяются для поддержки опосредованного данными (различных медиаформатов) взаимодействия между отдельными пользователями и их группами. В отличие от коммуникационных инструментов акцент делается на поддержке связности пользователей и механизмов общения между ними.
* В противоположность коммуникационным инструментам, которые являются обычно асинхронными, интерактивные инструменты преимущественно синхронны, позволяя взаимодействовать пользователям в режиме реального времени (как в случае интернет-телефонии, видеочатов и т.п.) либо почти синхронно (службы мгновенных сообщений, текстовые чаты и т.п.).

Можно назвать следующие примеры программных систем, которые относится к социальному ПО:

* Системы обмена мгновенными сообщениями (IM - Instant messaging) позволяют общаться с другим пользователем через сеть в режиме реального времени (в относительно защищенном режиме). К наиболее популярным из них можно отнести Skype, ICQ, Yahoo! Messenger, MSN Messenger, AOL Instant Messenger, Miranda IM. К системам, ориентированным на бизнес, можно отнести IBM Lotus Sametime, Microsoft Messenger и Jabber.
* Интернет-чаты (IRC - Internet Relay Chat) позволяют одновременно нескольким пользователям общаться в режиме реального времени.
* Интернет-форумы пришли на смены электронным конференциям (возникшим до появления WWW). Пользователь форума может создавать новую "тему", доступную для других. Другие пользователи могут просматривать тему и оставлять свои комментарии в режиме последовательной записи.
* Веб-блоги (web logs), или кратко блоги, можно рассматривать как личные он-лайн журналы отдельных пользователей. Владелец блога может размещать сообщения в своем журнале, в то время как другие пользователи (читатели) могут оставлять к ним свои комментарии.
* Вики-справочники (wiki), или просто вики, - по-сути, веб-сайты, содержимое которых может редактироваться посетителями сайта. Наиболее известный пример - Википедия.
* Сервисные социальные сети позволяют пользователям объединяться в он-лайн режиме вокруг общих для них интересов, увлечений или по различным поводам. Например, некоторые сайты предоставляют сервисы, с помощью которых пользователи могут размещать для общего доступа персональную информацию, необходимую для поиска партнеров. Примеры: LinkedIn, В контакте.
* Социальные сети прнятия решений - веб-сети для обсуждения с целью принятия решений. Используются для установления постоянной связи отдельных людей с правительством.
* Коммерческие социальные сети ориентированы на поддержку бизнес-транзакций и формирование доверия людей к брендам на основе учета их мнений о продукте, о том как сделать его лучше и т.п., тем самым позволяя потребителям участвовать в продвижении продукта и расширяя их осведомленность.
* Социальные закладки (social bookmarking). Некоторые веб-сайты позволяют пользователям предоставлять в распоряжение других список закладок или популярных веб-сайтов. Такие сайты также могут использоваться для поиска пользователей с общими интересами. Пример: del.icio.us.
* Социальные каталоги (social cataloging) напоминают социальные закладки, но ориентированы на использование в академической сфере, позволяя пользователям работать с базами данных цитат из научных статей. Примеры: Academic Search Premier, LexisNexis Academic University, CiteULike, Connotea.
* Социальные библиотеки представляю собой приложения, позволяющие посетителям оставлять ссылки на их коллекции, книги, аудиозаписи и т.п., доступные другим. Предусмотрена поддержка системы рекомендаций, рейтингов и т.п. Примеры: discogs.com, imdb.com.
* Многопользовательские сетевые игры (Massively Multiplayer Online Games) имитируют виртуальные миры с различными системами подсчета очков, уровней, состязательности, победителей и проигравших. Пример: World of Warcraft.

***Фолксономия***

Фолксономия (folksonomy) — практика и методика совместной категоризации контента (ссылок, фото, видео клипов и т.п.) посредством произвольно выбираемых меток (тегов). Она основана на спонтанном сотрудничестве группы людей с целью организации контента и полностью отличается от традиционных формальных методов классификации на основе индексных терминов. Как правило, этот феномен возникает только в неиерархических сообществах, например на общедоступных веб-сайтах. Так как участники фолксономии контента обычно являются и основными же ее потребителями, использование методики фолксономии приводит к результатам, более точно отражающим совместную концептуальную модель контента всей группы.

Основными проблемами фолксономии, приводящими к ненадежности и несогласованности результатов, являются:

* Наличие форм множественного числа.
* Полисемия.
* Синонимия.
* Глубина (специфичность) использования меток.

Фолксономию можно рассматривать в качестве одного из ключевых элементов в развитии Семантической веб-сети, в рамках которой все веб-страницы содержат машинно-ориентированные метаданные, описывающие содержимое страниц. Эти метаданные должны значительно улучшать точность работы поисковых систем.Однако во избежание трудностей, вызванных ненадежностью и несогласованностью в работе больших сообществ авторов страниц, им рекомендуется использовать стандарты метаданных, например Дублинское ядро (Dublin Core).

***Семантическая веб-сеть***

Семантическая веб-сеть (Semantic Web) — часть глобальной концепции развития сети Интернет, целью которой является реализация возможности машинной обработки информации, доступной в сети WWW. Основной акцент в этой концепции делается на работе с метаданными, однозначно характеризующими свойства и содержание ресурсов WWW, вместо используемого в настоящее время текстового анализа документов.

Термин был введен Тимом Бернерсом-Ли в мае 2001 года.

В семантической веб-сети предполагается повсеместное использование

* универсальных идентификаторов ресурсов (URI),
* онтологий и языков описания метаданных.

Концепция семантической веб-сети была принята и продвигается W3С. Для ее внедрения предполагается создание сети документов, содержащих метаданные о ресурсах WWW, и существующей параллельно с ними. Тогда как сами ресурсы предназначены для восприятия человеком, метаданные используются машинами (поисковыми роботами и другими интеллектуальными агентами) для получения однозначной информации о свойствах этих ресурсов с помощью механизмов логического вывода.

Техническую часть семантической паутины составляет семейство стандартов на языки описания, включающее XML, XML Schema, RDF, RDF Schema, OWL и др. Необходимость описания метаданных так или иначе приводит к дублированию информации. Каждый документ должен быть создан в двух экземплярах: размеченным для чтения людьми, а также в машинно-ориентированном формате.

***Онтология***

Онтология - это попытка всеобъемлющей и детальной формализации некоторой области знаний с помощью концептуальной схемы. Обычно такая схема состоит из иерархической структуры данных, содержащей все релевантные классы объектов, их связи и правила (теоремы, ограничения), принятые в этой области.

Современные онтологии обычно состоят из экземпляров, понятий, атрибутов и отношений.

Для описания онтологий Веб был разработан специальный язык - OWL (Web Ontology Language), построенный на основе XML. Язык OWL может быть использован для описания классов и отношений между ними. В основе языка — представление действительности в модели данных "объект — свойство". Язык применим не только для описания веб-страниц, но и любых объектов действительности и рассматривается в качестве одной из фундаментальных технологий, необходимых для построения Семантической веб-сети.

***Семантические веб-сервисы***

В то время как совокупность ресурсов и их метаданных можно считать статической частью семантической паутины, ее динамическую часть представляют семантические веб-сервисы - законченные элементы программной логики с однозначно описанной семантикой, доступные через Веб и пригодные для поиска, композиции и выполнения.

Технически, семантический веб-сервис отличается от обычного веб-сервиса наличием не только описания интерфейса (обычно на языке WSDL ) в терминах типов данных, передаваемых сервису, возвращаемых значений и генерируемых ошибок, но и наличием семантического описания всех его характеристик.

Потенциальная выгода от использования семантических веб-сервисов заключается в возможности автоматического поиска (а также композиции) программными агентами подходящих сервисов для решения поставленных задач.

Тем не менее, сложность этой задачи в ее общей формулировке пока позволяет добиваться некоторых положительных результатов только в узкоспециализированных отраслях, явным образом выигрывающих от внедрения сервисно-ориентированной архитектуры, например в интеграции корпоративных приложений.