ВВЕДЕНИЕ

Настоящий курс является введением в теорию и практику современных Интернет-технологий. Интернет-технологии — это технологии глобальной информационной сети Интернет, представляющие собой наиболее сложный и динамичный вид информационных технологий. Следует различать понятия «Интернет-технологии» в широком и в узком смысле слова.

Под понятием «Интернет-технологии» в широком смысле понимают технологии обмена данными, основанные на использовании стека протоколов TCP/IP. Основные положения стека протоколов TCP/IP рассмотрены в курсе «Аппаратно-программное обеспечение ЭВМ и сетей». Стек TCP/IP поддерживает два вида сервисов: сервисы технологий Интернет-коммуникаций и сервисы технологий прикладного уровня: Telnet, FTP, SMTP, POP, IMAP, HTTP, которые являются базовыми технологиями Интернет.

С понятием «Интернет-технологии» в узком смысле связаны Web-технологии, которые относятся к прикладному уровню стека TCP/IP и охватывают огромную, постоянно расширяющуюся сферу человеческой деятельности включая облачные технологии и технологии Big Data.

В этом курсе рассматриваются базовые технологии и сервисы Интернет, облачные технологии и сервисы, современные Web-технологии, а также технологии, связанные с разработкой корпоративных порталов и безопасностью в Интернет.

Лабораторный практикум данного курса ориентирован на освоение современных Web-технологии и включает технологии работы с HTML5, технологии работы с XML, технологию работы с облачной платформой и технологию создания корпоративного портала.

2. БАЗОВЫЕ ТЕХНОЛОГИИ И СЕРВИСЫ INTERNET

2.1. Базовые технологии прикладного уровня стека TCP/IP

Прикладной уровень стека TCP/IP играет особую роль в сетевых технологиях. Три нижних уровня этого стека по существу обеспечивают транспорт для прикладных программ. Для конечного пользователя технологии этих уровней скрыты.

Стек TCP/IP включает большое число протоколов прикладного уровня, ориентированных на самые разнообразные применения, в том числе и протоколы, предназначенные для реализации обеспечивающих (служебных) технологий. К таким служебным протоколам относятся протокол DNS (служба DNS для поиска IP адресов), SNMP (простой протокол управления сетью), протоколы безопасности.

К числу базовых протоколов прикладного уровня TCP/IP, на основе которых реализуются собственно прикладные технологии и информационные сервисы, относятся: TELNET, FTP, TFTP, SMTP, POP 3, IMAP 4, HTTP.

2.2. Технология Telnet

Понятие Telnet

Telnet – это ставшая классикой одна из самых старых информационных технологий Internet, обеспечивающая наиболее простой способ доступа к локальным или удаленным информационным ресурсам.

Понятие Telnet включает как протокол Telnet, так и разработанные на основе этого протокола программные средства (они входят в штатный состав операционных систем). Такие программные средства включают программу «клиент Telnet» (терминал, терминальное устройство, интерфейс пользователя) и программу «сервер Telnet» (терминал-ориентированный процесс).

Протокол Telnet

Протокол Telnet (RFC-854, RFC-855; 23 порт TCP) описывает стандартный метод взаимодействия терминального устройства ("user") и терминал-ориентированного процесса ("server"). Telnet базируется на трех фундаментальных положениях:

концепции сетевого виртуального терминала NVT (Network Virtual Terminal);

принципа договорных опций (согласование параметров взаимодействия);

симметрии связи "терминал-процесс".

Протокол Telnet может быть использован для организации взаимодействий:

"терминал-процесс",

"терминал-терминал" (связь),

"процесс-процесс" (распределенные вычисления).

При этом во всех случаях user" - это сторона, инициирующая соединение, а "server" - пассивная сторона.

Сетевой виртуальный терминал NVT задает стандартное описание наиболее широко используемых возможностей реальных физических терминальных устройств. Так что (реальные, основанные на протоколе Telnet) терминальная программа ("user") и взаимодействующий с ней процесс ("server"), преобразовывая характеристики физических устройств в спецификацию NVT, обеспечивают тем самым совместимость устройств с разными характеристиками и возможностями.

Принцип договорных опций позволяет согласовать параметры взаимодействия. NVT специфицирует минимально необходимый набор параметров, который позволяет работать по telnet даже самым древним устройствам. Принцип договорных опций позволяет использовать возможности современных устройств. Например, при взаимодействии «терминал-процесс» "user", (в такой схеме инициатор всегда--"user") используя команды договора, предлагает "server"у применять Esc-последовательности. Получив такую команду "server" начинает их вставлять. В режиме "терминал-терминал" каждая из сторон может выступать инициатором договорного процесса. При этом применяется принцип "прямого действия" (а не "запрос-подтверждение"). Состоящий в том, что если терминальная программа хочет расширить возможности представления информации, то она это делает (например, вставляет в информационный поток Esc-последовательности), если в ответ она получает информацию в новом формате, то это означает, что попытка удалась, в противном случае происходит откат к стандарту NVT.

Обычно процесс согласования форм представления информации происходит в начальный момент организации telnet-соединения. Каждый из процессов старается установить максимально возможные параметры сеанса. Однако эти параметры могут быть изменены и позже, в процессе взаимодействия (например, после запуска прикладной программы).

Симметрия взаимодействия по протоколу telnet позволяет в течение одной сессии программе-"user" и программе-"server" меняться местами, что принципиально отличает технологию telnet от традиционной схемы "клиент-сервер.

Сетевой виртуальный терминал NVT в протоколе Telnet определен как двунаправленное символьное устройство, состоящее из принтера (для отображения информации) и клавиатуры (для ввода данных).

Принтер имеет неограниченные ширину и длину страницы и может отображать все символы US ASCII (коды 32 - 127), расширенный набор символов (коды 128 - 255), а также распознает управляющие коды (0 - 31 и 127), среди которых имеются обязательные (см. табл. 2.1) и рекомендуемые (см. табл. 2.2) коды.

Таблица 2.1.Обязательные коды

Название кода	Код	Значение
NULL	0	Нет операции
Перевод строки	10	Переход на другую строку с сохранением текущей

Line Feed (LF)		позиции в строке
Возврат каретки Carriage Return (CR)	13	Устанавливает в качестве текущей первую позицию текущей строки

Таблица 2.2. Рекомендованные коды

Название кода	Код	Значение
Звонок (BEL)	7	Звуковой сигнал
Сдвиг на одну позицию назад (BACK SPACE)	8	Перемещает каретку на одну позицию назад в текущей строке
Горизонтальная табуляция Horizontal Tab (HT)	9	Перемещение к следующей позиции горизонтальной табуляции
Вертикальная табуляция Vertical Tab (VT)	11	Перемещение курсора к следующей позиции вертикальной табуляции
Прогон страницы Form Feed (FF)	12	Переход к новой странице

Клавиатура должна обеспечивать возможность ввода всех символов ASCII, а также может иметь возможность генерировать стандартные специальные функции управления терминалом. Стандарт telnet определяет пять функций управления терминалом (если на реальном терминале эти функции отсутствуют, то заменяются командой NO (No-Operation)).

Команда "Прервать процесс" (Interrupt Process - **IP**) реализует стандартный для многих систем механизм прерывания процесса выполнения задачи пользователя (Cntrl+C в Unix-системах или Cntrl+Break в MS-DOS).

Команда "Прервать процесс выдачи" (Abort Output - **AO**). В отличие от команды IP (при выполнении IP прерывается выполнение текущего процесса пользователя, но не происходит очистка буфера вывода, при этом вывод данных на экран или на принтер может продолжаться) происходит очистка буфера вывода, что прерывает выдачу данных.

Команда "Ты еще здесь?" (Are You There - **AYT**). Позволяет пользователю убедиться в том, что он не потерял связь с удаленной машиной.

Команда "Удалить символ" (Erase Character – **EC).** Команда EC стандартизирует так называемый символ "забой" или удаление последнего напечатанного символа.

Команда "Удалить строку" (Erase Line – **EL**). Данная команда аналогична EC, но удаляет целую строку ввода. Обычно выполнение этой команды приводит к очистке буфера ввода, т.к. при работе в режиме командной строки строка ввода только одна.

Команды telnet имеют свой формат. Команда – это 2-байтовая последовательность, состоящая из Esc-символа (255) IAC (Interpret as Command) и кода команды (240-255). Команды, связанные с процедурой согласования параметров сеанса, имеют 3-х байтовый формат: третий байт - ссылка на устанавливаемую опцию.

Технология Telnet позволяет использовать средства Internet для связи с базами данных, каталогами библиотек и другими информационными мировыми ресурсами. Доступ по Telnet предполагает регистрацию пользователя на удаленном хосте, так что доступными оказываются ресурсы только тех серверов, которые имеют специально зарезервированных пользователей (гостевые учетные записи) и таких серверов становится все меньше.

2.3. Протокол FTP

Первая спецификация **протокола FTP** (File Transfer Protocol) относятся к 1971 году (**RFC 114**). FTP подвергался многократной модификации (более15; **RFC 765** – переход на транспорт TCP; **RFC 959**, октябрь 1985 г., устранены ошибки в документации и добавлены новые команды).

Простейшая модель работы протокола **FTP** представлена на рис. 2.1. В протоколе FTP используются два канала:

канал управления (КУ), по которому осуществляется управление информационным обменом в стандарте протокола Telnet;

канал передачи данных (КПД), который может быть использован как для приема, так и для передачи данных.

FTP соединение инициируется интерпретатором протокола пользователя (ИПП). Команды FTP генерируются ИПП и передаются на интерпретатор протокола сервера (ИПС)**,** установить контакт, с которым пользователь может и другими средствами. Ответы сервера отправляются пользователю также по КУ. Сервер FTP «слушает» 21 порт TCP, находясь постоянно в состоянии ожидания соединения.

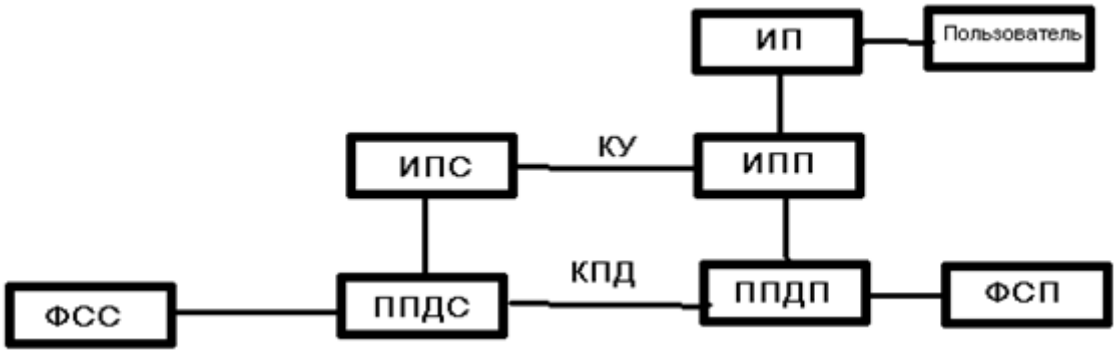


Рис. 2.1 Простая модель работы протокола FTP

Команды FTP определяют параметры канала и процесса передачи данных, а также характер работы с файловыми системами, а именно:

роль участников соединения: активный, пассивный;

порт соединения как для модуля Программа передачи данных пользователя (ППДП), так и для модуля Программа передачи данных сервера (ППДС);

тип передачи;

тип передаваемых данных;

структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить, например, сохранить, считать, добавить или удалить данные или файл и другие.

После согласования параметров канала передачи данных, один из участников соединения, который является пассивным (например, ППДП), становится в режим ожидания открытия соединения на заданный для передачи данных порт. После этого активный модуль (например, ППДС) открывает соединение и начинает передачу данных.

После окончания передачи данных, соединение между ППДС и ППДП закрывается, но управляющее соединение ИПС и ИПП остается открытым. Пользователь, не закрывая сессии FTP, может еще раз открыть канал передачи данных.

В случае передачи данных на третий компьютер пользователь организует канал управления с двумя серверами с прямым каналом данных между ними. Команды управления идут через пользователя, а данные напрямую между серверами (см. рис. 2.2).

Алгоритм работы такой схемы, на примере передачи данных от сервера 2 к серверу 1, сводится к следующему.

1. ИПП указал модулю ИПС 1 работать в пассивном режиме. После этого ИПС 1 отправил пользователю адрес и номер порта (N), который он будет слушать.

2. ИПП назначил ИПС 2 активным участником соединения, указав ему передавать данные на порт (N) ИПС 1.

3. ИПП подал ИПС 1 команду “сохранить поступившие данные в указанном файле”, а ИПС 2 — “передать содержимое заданного файла”.

4. Между ИПС 1 и ИПС 2 образуется поток данных, который управляется клиентским хостом.

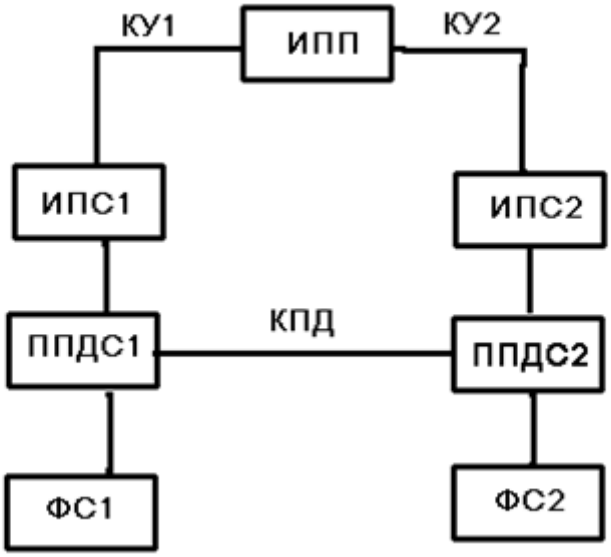


Рис. 2.2. Простая работы протокола FTP с передачей данных на третий компьютер.

Команды протокола FTP

Команды управления контролем передачи данных, которыми обмениваются ИПС и ИПП, можно разделить на три достаточно большие группы:

Команды управления доступом к системе.

Команды управления потоком данных.

Команды FTP-сервиса.

Команды управления доступом к системе.

USER. Как правило, эта команда открывает сессию FTP между клиентом и сервером. Аргументом команды является **имя (идентификатор)** пользователя для работы с файловой системой. Команда может подаваться не только в начале, но и в середине сессии, если, например, пользователь желает изменить идентификатор, от имени которого будут проводиться действия. При этом все переменные, относящиеся к старому идентификатору, освобождаются. Если во время изменения идентификатора происходит обмен данными, обмен завершается со старым идентификатором пользователя.

PASS. Подается после ввода идентификатора пользователя, в качестве аргумента содержит **пароль пользователя**. При этом данные аутентификации FTP передаются по сети открытым текстом.

CWD. Позволяет работать с различными каталогами удаленной файловой системы. Аргументом команды является строка, указывающая путь требуемого каталога удаленной файловой системы.

REIN. Команда реинициализации. Очищает все переменные текущего пользователя, сбрасывает параметры соединения, но позволяет завершить передачу данных с прежними параметрами.

QUIT. Закрывает управляющий канал. Если в момент подачи команды происходит передача данных, то канал закрывается после окончания передачи

данных.

Команды управления потоком данных

Команды управления потоком (устанавливают параметры передачи данных) могут подаваться в любом порядке, но все они должны предшествовать командам FTP-сервиса. К основным командам управления потоком данных относятся следующие:

PORT. Команда назначает адрес и порт хоста, который будет использоваться как активный участник соединения по каналу передачи данных. Аргументами команды являются 32-битный IP адрес и 16-битный номер порта соединения. Эти значения разбиты на шесть 8-битных полей и представлены в десятичном виде: h1, h2, h3, h4, p1, p2, где hN - байты адреса (от старшего к младшему), а pN - байты порта (от старшего к младшему).

PASV. Эта команда отправляется модулю, который будет играть пассивную роль в передаче данных (“слушать” соединение). Ответом на данную команду должна быть строка, содержащая адрес и порт хоста, находящиеся в режиме ожидания соединения в формате команды PORT — “h1, h2, h3, h4, p1, p2”.

Команды **TYPE, STRU, MODE** определяют, соответственно, тип передаваемых данных (**ASCII, binary, Image** и другие), структуру или формат передачи данных (**File, Record, Page),** способ передачи (**Stream, Block** и другие). Использование этих команд необходимо для организации взаимодействия в гетерогенных средах.

Команды FTP-сервиса

Это команды определяющие действия с файлами. Как правило, аргументом команд этой группы является путь к файлу. Синтаксис указанного пути должен удовлетворять требованиям формата файловой системы обработчика команды. К основным командам данной группы относятся следующие:

RETR. Команда указывает модулю “Программа передачи данных сервера” передать адресату копию файла, заданного параметром этой команды.

STOR. Указывает модулю “Программа передачи данных сервера” принять данные и сохранить их как файл, имя которого задано параметром этой команды. Если такой файл уже существует, он будет замещен новым, если нет, будет создан заново.

Команды **RNFR и RNTO** должны следовать одна за другой. Первая команда содержит в качестве аргумента старое имя файла, вторая — новое. Последовательное применение этих команд переименовывает файл.

ABOR. Команда предписывает серверу прервать выполнение предшествующей сервисной команды (например, передачу файла) и закрыть канал передачи данных.

Команда **DELE** удаляет указанный файл.

Команды **MKD и RMD**, соответственно, создают и удаляют указанный в аргументе каталог.

При помощи команд **LIST и NLST** можно получить список файлов в указанном каталоге.

Все команды FTP-протокола отправляются **интерпретатором протокола пользователя** (ИПП) в текстовом виде - по одной команде в строке. Каждая строка команды (идентификатор и аргументы) заканчиваются символами <CRLF>. Имя команды отделяется от аргумента символом пробела.

Обработчик команд возвращает код обработки каждой команды, состоящий из трех цифр. Коды обработки составляют определенную иерархическую структуру и, как правило, определенная команда может возвратить только определенный набор кодов. За кодом обработки команды следует символ пробела, затем следует текст пояснения. Например, строка успешного завершения операции выглядит следующим образом: “200 Command okay”.

2.4. Протокол TFTP

Тривиальный протокол передачи файлов TFTP (Trivial File Transfer Protocol , RFC 1350, июль 1992 г., п**о**рт 69 UDP) является упрощенной вариантом протокола FTP.

Протокол TFTP ориентирован на транспортные услуги UDP и поэтому не обеспечивает надежной передачи данных. Программная реализация TFTP обычно ис пользуется для начальной загрузки бездисковых сетевых рабочих станций (записывается в чипе ПЗУ рабочей станции), мостов и маршрутизаторов.

Протокол TFTP значительно проще протокола FTP. TFTP поддерживает 5 типов пакетов:

запрос чтения,

запрос записи,

данные,

подтверждения,

ошибки.

Формат пакетов описан в RFC 1350. Первый пакет, передаваемый от клиента сер веру, является управляющим. Этот пакет определяет имя файла, а также действия с ним на удаленной станции. Затем передаются пакеты данных, каждый размером 512 байт; при этом начальный пакет данных помечается номером 1. Номер каждого следующего пакета увеличивается на единицу. Принимающая сторона на каждый полученный пакет передает подтверждение. Любой пакет размером меньше 512 байт означает конец передачи данных.

Порядковая нумерация и подтверждение принятых пакетов реализуется TFTP, а не службой транспортного уровня UDP. (UDP, как нам известно, обеспечивает только ненадежную, не ориентированную на соединение службу). При этом TFTP не использует принцип скользящего окна (как в TCP), так что можно сказать, что TFTP пользуется окном размером 1. Протокол TFTP в отличие от FTP поддерживает толь одно соединение.

2.5. Почта Интернет

Почтовые системы

Электронная почта (E-mail, Electronic Mail) — это не только всем известное массовое средство электронных коммуникаций в Internet, но и средство доступа к таким информационным ресурсам, как FTP-архивы и многое другое.

Похожая по принципам организации на обычную почту, электронная почта превосходит ее по всем показателям (скорость и стоимость доставки, удобство создания письма и ответа на него и т.д.)

С технологической точки зрения почта Internet представляет собой службу пересылки сообщений между зарегистрированными почтовыми адресами.

Существующие системы электронной почты, в зависимости от используемых стандартов, можно условно разделить на три класса.

1. Системы на основе SMTP (стандарт де-факто).
2. Системы на основе Рек. ITU-T X.400 (модель OSI).
3. Системы на базе фирменных стандартов, например, MS Mail, Lotus cc:Mail.

По функциональному назначению почтовые системы (RFC 2821) подразделяются на:

Пользовательский агент (клиент, обеспечивающий конечному пользователю сервис по созданию и передаче писем по протоколу SMTP и получение по протоколу POP или IMAP).

Агент доставки (сервер, обеспечивающий пользовательскому агенту исходящий, входящий и транспортный сервис).

Relay (система, обеспечивающая меж узловую пересылку почты по протоколу SMTP).

Gateway (шлюз в другую почтовую систему, например, в UUCP).

В литературных источниках широко используются также термины: MTA (Mail Transport Agent) — транспортные агенты и MUA (Mail User Agent) — пользовательские агенты.

Почтовый адрес (адрес электронной почты) состоит из почтового псевдонима (Alias), оператора адреса «@» (знака «коммерческого эй» или «собаки») и имени почтового домена. Например:

zizi@mail.ru

Почтовый адрес может соответствовать одному человеку, группе людей, роботу-обработчику и т.д.

Имя почтового домена — это такое **доменное имя**, для которого в **базе данных DNS** существует запись типа **MX** (Mail Exchanger). Компьютер с сетевым интерфейсом типа MX является сервером (узлом, хостом) обмена почтой (но не наоборот, т.е. компьютер может быть почтовым сервером, но не иметь MX**).** Такие узлы могут иметь различное функциональное назначение (запись MX эту функциональность не идентифицирует**)**. Почтовых серверов в домене может быть несколько, в этом случае будет выбран сервер с

наименьшим значением приоритета. Доменное имя сетевого интерфейса «рядового» компьютера, участвующего в почтовых коммуникациях, имеет самый простой **тип записи** — A для Ipv4 и AAAA для Ipv6.

2.5. Протокол SMTP

Протокол обмена почтовыми сообщениями в Internet SMTP (Simple Mail Transfer Protocol, порт 25 TCP) описан в RFC 821 (август 1982 г.). Расширяющие дополнения содержатся в RFC 1870 (ноябрь 1995 г.9с.), в RFC 974 описано взаимодействие SMTP с системой доменных имен DNC.

Схема взаимодействия по протоколу SMTP следующая. Между отправителем (клиентом), который инициирует соединение, и получателем почтового сообщения (сервером), устанавливается двусторонняя связь (см. рис. 2.3). Далее между клиентом и сервером устанавливается информационный обмен, продолжающийся до тех пор, пока соединение не будет закрыто или прервано.

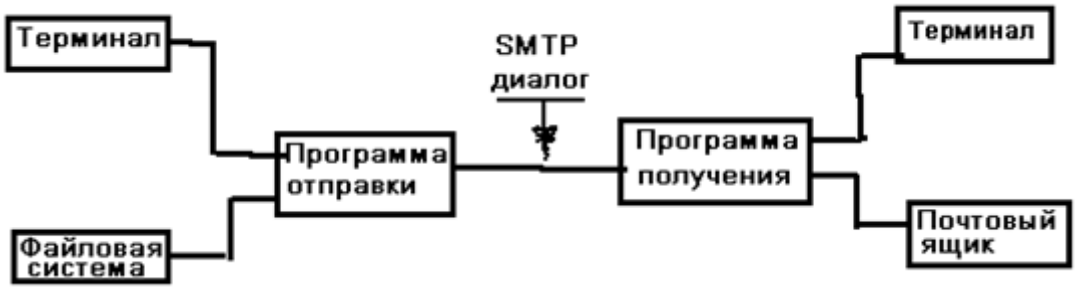


Рис. 2.3. Схема взаимодействия по протоколу SMTP.

При такой схеме взаимодействия (режим on-line) почта доставляется клиенту практически незамедлительно (секунды, возможно минуты из-за очереди). Это принципиально отличает протокол SMTP от протокола UUCP (Unix-Unix-CoPy), используемого в Unix-системах. В UUCP почта передается по цепочке: от одного сервера к другому, пока не достигнет машины абонента-получателя (так называемый принцип "stop-go").

Протокол SMTP предусматривает ряд процедур, которые реализуются соответствующими командами. К основным процедурам SMTP относятся:

передача почты (Mail Procedure);

форвардинг почты (Mail Forwarding), т.е. перенаправление почтового сообщения;

проверка имён почтового ящика и вывод списка почтовых групп (Verifying and Expanding);

открытие и закрытие канала передачи (Opening and Closing).

Команды SMTP состоят из ключевых слов, за которыми следует один или более параметров (SP). Ключевое слово состоит из 4-х символов и отделено от аргумента одним или несколькими пробелами. Каждая командная строка заканчивается символами CRLF (перевод строки).

Команды протокола SMTP

HELO *hostname*

Первая командой сеанса, *hostname* **-** доменное имя отправителя (вызывающего клиента).

MAIL FROM: *email_адрес_от_кого*
Обратный адрес (адрес отправителя).

RCPT TO: *email_адрес_кому*
Адрес получателя (в случае нескольких адресатов команда повторяется для каждого адресата).

DATA
Команда вводится без параметров и обозначает начало ввода сообщения **(б**азовая структура сообщения определена в RFC-822). Сервер посылает промежуточный положительный отклик 354, после чего воспринимает все последующие строки как сообщение. Концом ввода сообщения является новая строка, состоящая из одной точки в первой позиции. Сообщение состоит из заголовка (который регламентируется RFC-822) и тела. Между заголовком и телом сообщения должна быть *одна пустая строка*. Сообщение укладывается в конверт, который не виден получателю.

RSET
Сброс сеанса к начальному состоянию (состояние, как после ввода HELO).

VRFY *email_адрес*
Команда серверу: проверить подлинность введенного *email_адреса.* В случае успеха выдается положительный отклик (250,251 или 252), иначе выдается отклик 550. При этом адекватность положительного отклика (существование *email_адреса)* гарантирована только для локальных адресов на сервере.

EXPN *email_addr*
Команда по которой сервер выводит локальные адреса списка рассылки, в котором содержится и адрес *email_addr.* Если *email_addr не локальный адрес, то* поведение команды не определено (просто выдается отклик 250). Действия этой команды в стандарте четко не определены. Ее реализация не является обязательной. Команда по соображениям секретности может не поддерживаться сервером

SEND FROM: *email_адрес*
Используется вместо команды MAIL, указывая на то, что почта должна быть доставлена на терминал пользователя.

SOML FROM: *email_адрес*
Комбинации команд SEND или MAIL (или на терминал или в ящик).

SAML FROM: *email_адрес*
Комбинации команд SEND и MAIL (успех, если хотя бы в ящик).

HELP
Запросить у сервера помощь о переданной в качестве аргумента команде.

NOOP

На эту команду сервер должен дать положительный ответ. Команда ничего не делает и никак не влияет на указанные до этого данные.

QUIT

Конец связи.

Существуют также команды Расширенного SMTP **(ESMTP**). Однако не все серверы их поддерживают или поддерживают некоторое подмножество ESMTP-команд, включая и нестандартные команды (это можно выяснить, если вместо HELO ввести команду EHLO).

Стандарт MAIL (RFC 822)

Базовая структура сообщения электронной почты специфицирована в **RFC 822** (1982 г.). Расширяющие дополнения содержатся в RFC 1870 (1995 г.).

Сообщение, согласно **RFC 822,** состоит из двух частей: заголовка и тела. При пересылке сообщения добавляется еще и третья часть – конверт (пользователю он не виден), который формируется в процессе SMTP-сеанса (**RFC 821).** Следует иметь в виду, что все эти три части иногда также называют сообщением (это сообщение в широком смысле, а без конверта — сообщение в узком смысле).

Заголовок всегда находится перед телом сообщения и отделен от него пустой строкой. Заголовок состоит из полей. Поле включает имя поля (ключевое слово) и содержание поля. Имя поля отделено от содержания символом **":".**

Стандартный заголовок содержит следующие поля.

Date: Дата, которая проставляется автоматически.

FROM: Email адрес отправителя (например, oris@yahoo.com)

TO: Email адрес получателя (например zizi@mail.ru).

CC: Email адреса получателей копии сообщения. Адреса разделяются точкой с запятой и пробелом.

Subject: Тема сообщения.

Заголовок может содержать дополнительные поля.

Sender: Адрес первоисточника (или источника) содержания данного письма (если отправитель не является автором содержания).

Reply-To: Пользователь, которому отвечают .

Comment: Комментарий.

In-Reply-To: Используется в письмах относящихся к типу "В ответ на Ваше сообщение, отвечающее на сообщение,

Bcc: Получатель (получатели), указанный в этом поле будет невидим остальным получателям, адреса которых указаны в "To:" и "Cc:" , т.е. они (остальные) не будут знать, что копия письма отправлена этому для них невидимому получателю.

Message-ID: Уникальный идентификатор сообщения, генерируемый MTA-отправителем; для восприятия человеком не предназначен.

Received: Заголовок, который добавляется каждым транспортным агентом, через которого проходит сообщение. Содержит информацию кем, от кого, когда и каким образом получено сообщение.

В заголовке могут также использоваться нестандартные поля, определяемые пользователем**: X-Special-action.**

Адреса на конверте и адреса в заголовках "**From**:" и "**To**:" могут совпадать. Но если письмо адресовано нескольким получателям в разные почтовые домены, то у получателей они (адреса) будут различаться.

Тело сообщения в стандарте RFC 822 представляет собой текст в формате ASCII (7-битовая кодировка), что не позволяет передавать даже размеченный текст, не говоря уже о передаче графики, аудио, видео. Проблемы RFC 822 были разрешены в спецификации MIME.

Стандарт MIME

Стандарт MIME (Multipurpose Internet Mail Extension*;* RFC 2045 - RFC 2049,) описывает многоцелевые расширения почты Интернет (MAIL, RFC 822). Стандарт MIME ориентирован на устранение недостатков **RFC 822**. Встречающиеся в литературных источниках версии MIME RFC 1341 (1992 г.) и RFC 1521 (1993 г.) являются устаревшими.

Для представления различных видов информации в стандарте MIME зарезервированы следующие специальные поля заголовка почтового сообщения:

Поле версии MIME (MIME-Version), которое используется почтовыми программами для идентификации стандарта сообщения (программа должна отличать MAIL от MIME).

Поле описания типа данных (Content-Type**), которое** предназначено для описания типа данных содержащихся в теле почтового сообщения.

Поле типа кодирования информации (Content-Transfer-Encoding)**,** которое указывает на применяемый тип процедуры кодирования. Технология MIME по существу посредством специального кодирования использует принцип инкапсуляции типов данных в сообщения стандарта RFC 821, в котором, как известно, для передачи данных используется код us-ascii. Обрабатывающие почтовые программы должны знать используемый код.

Два дополнительных поля:

Content-ID определяющее уникальный идентификатор сообщения (это поле синтаксически идентично полю Message-ID RFC 822)

Content-Description для комментария содержания.

Стандарт MIME разработан как расширяемая спецификация, число типов данных в которой будет расти. На июль 2003 года количество типов и подтипов данных превышало 150. MIME-типы используются не только в электронной почте, но и в WWW.

Основные типы данных

text

Текстовые данные. Наиболее распространенные подтипы: **plain** (обычный текст) и **html**. Возможный параметр: "charset= *название_кодировки_символов*"; наличие параметра необязательно. (Примеры кодировок символов: us-ascii, кириллица: koi8-r, windows-1251, iso8859-5.) Если не указан charset, считается, что это us-ascii. Пример заголовка: Content-Type: text/plain; charset=koi8-r.

image

Неподвижные изображения; примеры подтипов: jpeg, gif. Пример заголовка (параметры необязательны): .Content-Type: image/jpeg; name="portrait.jpg"

audio

Звук; примеры подтипов: mpeg, x-realaudio. Пример заголовка (параметры необязательны): Content-Type: audio/x-realaudio; name="song.ra"

video

Видео; примеры подтипов: **mpeg, quicktime**. Пример заголовка (параметры необязательны): Content-Type: video/mpeg; name="movie.mpeg"

application

Приложения. Любой другой тип, не попадающий в названные выше. Обычно используется для передачи двоичных данных (потока байт) прикладных программ. Существует много различных подтипов (например, **postscipt, msword, zip, x-javascript**). Если неизвестно, как интерпретировать данные, то используется общий подтип **octet-stream**. Пример заголовка (параметры необязательны): Content-Type: application/msword; name="my_file.doc"

multipart

Составное (смешанное) сообщение, состоящее из нескольких разделов, каждый из которых имеет свои заголовки и тело. Наиболее распространенный подтип - **mixed,** позволяющий любому разделу письма содержать данные любого зарегистрированного типа. Разделы отделяются друг от друга уникальным набором символов, который формируется пользовательским агентом и указывается в заголовке как значение обязательного параметра **"boundary".**

message

Сообщение, тело которого в свою очередь является email-сообщением, состоящим из одного или нескольких частей. Такой тип может применяться при пересылке (форвардинге) сообщений. Основной подтип: RFC 822. Пример заголовка: Content-Type: message/rfc822.

Пользователь может использовать также нестандартные типы и подтипы данных их наименования должны начинаться с символов "**x-**" .

Заголовок поле типа "Content-Transfer-Encoding:" определяет способ представление данных в теле сообщения (раздела).

Возможные значения:

7bit

Текст в формате us-ascii. Данные помещаются в тело сообщения, как они есть.

8bit

Восьмибитовый текст (например, кириллица) Данные помещаются в тело сообщения (раздела) как они есть.

binary

Двоичные данные. Помещаются в тело сообщения (раздела) как они есть. Это представление обычно не используется, из-за ограничения на длину строки (данные должны состоять из строк длиной не более 1000 символов).

quoted-printable

Восьмибитовый текст в закодированном виде. Выполняется посимвольное кодирование по следующему принципу. Символ с ascii-кодом от 32 до 127 включительно передается, как он есть; символ с другим кодом передается в виде тройки символов, состоящей из символа "=", за которым следует шестнадцатеричное значение кода символа. Символ "=", являющийся частью текста, кодируется как "=3D".

Такое представление данных используется для текстов состоящих в основном из символов латиницы, цифр **и** знаков препинания, с незначительным количеством других символов.

base64

Произвольные двоичные данные, закодированные по алгоритму base64. Принцип кодирования по этому алгоритму состоит в следующем: 3 октета (24 бита) входного потока разбиваются на четыре 6-ти битовых группы. Каждое возможное значение 6-битовой группы (0-63) ставится в соответствие одному из символов 7-битвой таблицы us-ascii (0='A', 1='B', ..., 26='a', 27='b', ..., 52='0', ..., 62='+', 63='/'). В результате такого кодирования 3 входных октета преобразуются в четыре семибитовых символа us-ascii, которые стандартно обрабатываются почтовыми программами. Выходной поток символов разбивается на строки длиной не более 76 символов.

Стандарт MIME допускает использование нестандартных представлений данных, которые должны начинаться с символов "**x-**".

2.6. Протокол POP 3

Протокол **POP 3** (Post Office Protocol v.3; RFC 1939, май 1996; TCP, порт 110) используется для получения почты с сервера на рабочую станцию пользователя (для передачи используется SMTP). Протокол **P**OP 2 (RFC 937) устарел и по набору команд несовместим с POP 3.

После установления соединения с сервером POP 3 появляется строка, начинающаяся с символов "+OK". Затем сервер POP3 переходит в стадию авторизации, в процессе которой пользователю необходимо ввести имя и пароль. После успешной авторизации, сервер POP-3 входит в транзакционное состояние, временно блокируя почтовый ящик пользователя для внешних транзакций.

В этом состоянии у пользователя (посредством клиента POP-3) появляется возможность узнать количество сообщений в его почтовом ящике, принять сообщения,

удалить сообщения из почтового ящика. Сообщения после их приема клиентом удаляются из почтового ящика.

После посылки со стороны клиента команды **quit,** сеанс связи будет закрыт. Затем на сервере будет выполнен необходимый процесс *обновления* (удаление сообщений и т.п.).

Команды протокола POP 3.

USER *имя_пользователя.*
Имя пользователя (идентификатор почтового ящика).

PASS *пароль.*
Пароль пользователя.

STAT.
Команда выводит два числа: число сообщений и их общий объем в байтах.

LIST *n.*
Если *n* указано, то выводит размер **n-го** сообщения в байтах. Если *n* не указано, то выводит список из двух колонок, содержащих номера сообщений и размер сообщений в байтах.

RETR *n.*
Выводит сообщение под номером *n* .

DELE *n.*
Удаляет из почтового ящика ***сообщение n*** без изменения нумерации сообщений.

TOP *n m.*
Выводит заголовок и ***m*** первых строк сообщения номер ***n***.

RSET.
Отменяет удаление всех сообщений, удаленных в данном сеансе.

NOOP.
Нет операции (то же, что и в SMTP),

QUIT. Конец связи.

2.7. Протокол IMAP 4

Протокол IMAP 4 (Internet Mail Access Protocol; **RFC 2060**, декабрь 1996 г; 143 порт TCP.) является протоколом, поддерживающим возможность управления электронной почтой непосредственно на почтовом сервере. MAP4, обеспечивая те же функции что и POP3, дополнительно позволяет:

выборочно загружать с сервера отдельные сообщения или фрагменты сообщений;

просматривать заголовки сообщений и выбрать для загрузки только избранные сообщения;

осуществлять поиск по ключевым словам в заголовках письма, и в самом сообщении;

и т.д.

При этом все сообщения, а также организованная пользователем структура папок, останутся на сервере неизменными до тех пор, пока не будут удалены или изменены пользователем явным образом.
IMAP обеспечивает надежный механизм идентификации пользователя, поддерживает Kerberos и другие протоколы безопасности, включает поддержку адресной книги и ссылок на отдельные электронные документы.

Программные средства почты Интернет

В Unix-системах стандартом де-факто является почтовая система **sendmail.** Поддерживает протоколы SMTP и UUSP, позволяет обеспечивать поддержку списка адресов-синонимов, списка адресов рассылки, автоматической рассылки почты через шлюзы, поддержку очередей сообщений для повторной рассылки почты, работу в качестве SMTP-сервера и т.д.

В качестве POP-сервера в Unix-системах может быть использована программа **qpopper.** Почтовыми клиентами являются программы: mail, pine, различные MailTools под X-Windows и др. К почтовым клиентам со встроенным POP-клиентом (Unix, Windows) относятся: Netscape, Eudora, The Bat, Mozilla Mail др.

В Windows-системах работают почтовые серверы **Netscape Messaging Server** и **Microsoft Exchange**. Они содержат все необходимые функции, включая **POP3-сервер**, систему администрирования почтовых ящиков, псевдонимов, групп и списков рассылки. Однако эти серверы по гибкости и универсальности уступают **sendmail.**

К почтовым клиентам под Windows относятся. **Microsoft Outlook Express**, который по известным причинам является одним из наиболее популярных почтовых клиентов (входит в состав ОС Windows). **Microsoft Outlook** (полная версия Microsoft Outlook), который поставляется в комплекте с Microsoft Office, и имеет в своем составе электронный органайзер. Обе почтовые программы поддерживает протоколы SMTP, POP3, IMAP.

Почтовый клиент **The Bat!** Является лидером среди почтовых клиентов. **The Bat!** считается самым удобным, функциональным и безопасным почтовым клиентом. Компактен и быстр, обладает богатыми многопользовательскими возможностями, поддерживает все основных стандарты обмена почтой и системы кодирования писем, удобен в управлении почтой, является полностью российской разработкой.

Eudora Mail. Популярная в прошлом программа. Имеет довольно удобный интерфейс, поддерживает стандартный набор почтовых протоколов (POP3, SMTP, IMAP).

Кроме почтовых программ, реализованных в виде отдельной программы, существуют почтовые клиенты, встроенные в браузеры. Наиболее популярным среди них являются почтовый клиент браузера **Opera.** . Этот клиент обладает минимальными почтовыми функциями (поддерживает лишь протоколы SMTP и POP3), имеет скромные средства поиска. Достоинством является поддержка различных операционных систем.

Кроме Opera, почтовый клиент имеет и другой популярный браузер — **Netscape Navigator**. Он, также как и клиент Opera поддерживает только самые основные стандарты, но по ряду показателей ему уступает.

2.8. Протокол HTTP

Версии протокола HTTP

HTTP (Hypertext Transfer Protocol: протокол пересылки гипертекста) используется в World Wide Web начиная с 1990 года. Первая версия HTTP (HTTP/0.9) представляла собой простой протокол для передачи необработанных данных. В версии HTTP/1.0 (RFC 1945, 1996 г.) введены MIME формат и модифицированная семантика запросов/ответов. Дальнейшее развитие протокол HTTP получил в версиях HTTP/1.1 (RFC 2068; RFC 2616, 1999 г., 176 с.). Версия RFC 2068 считается устаревшей.

RFC 2616 содержит 176 страниц, так что здесь приведены лишь общие сведения о протоколе HTTP/1.1, которые следует рассматривать как введение в этот протокол. Детальное описание HTTP/1.1 в интерпретации RFC 2068, можно найти в книге Ю.А. Семенова: Телекоммуникационные технологии (Гипертекстный протокол HTTP).

Принципы работы HTTP

Клиент посылает серверу запрос, содержащий в общем случае **метод** запроса, **URI**, версию протокола HTTP, MIME-подобное сообщение, содержащее модификаторы запроса, клиентскую информацию, и, возможно, тело запроса. Сервер отвечает строкой состояния, включающей версию протокола сообщения, код успешного выполнения или код ошибки, MIME-подобное сообщение, содержащее информацию о сервере, метаинформацию объекта, и, возможно, тело объекта.

Метод — это команда, которую нужно применить к ресурсу, идентифицированному запрашиваемым URI. В RFC описаны следующие методы:

GET, HEAD, POST (RFC 1945);
OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE (RFC 2068);
OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT (RFC 2616).

URI (Uniform Resource Identifier, универсальный идентификатор ресурса) — это общий термин для всех допустимых форматов схем адресации, включая и URL (универсальный локатор ресурсов, который является общепринятым в World Wide Web. Синтаксис URI в нотации Бекуса-Наура приведен в RFC 2616

Большинство HTTP соединений инициализируется агентом пользователя. В самом простом случае, запрос может быть выполнен посредством одиночного соединения между агентом пользователя и первоначальным сервером (рис 2.4).

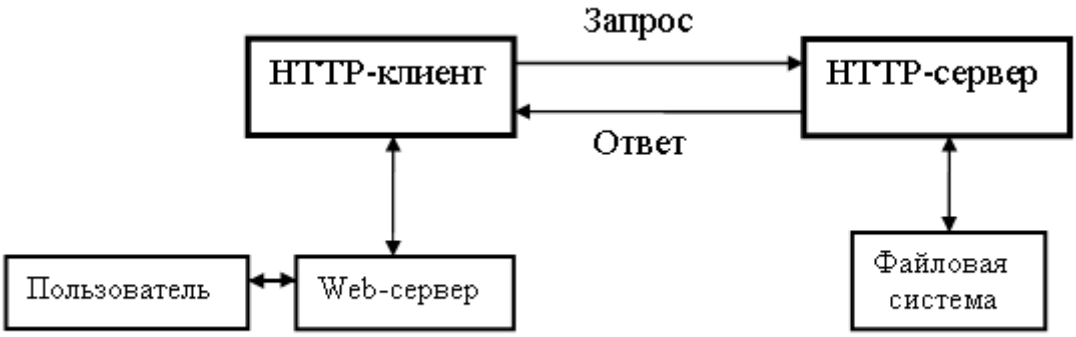


Рис. 2.4. Простое соединение между пользователем и сервером.

Более сложная ситуация возникает, когда в цепочке запросов/ответов присутствует один или несколько посредников (рис 2.5).

Цепочка запросов ---------->
UA -----v----- **A** -----v----- **B** -----v----- **C** -----v----- **O**
<----------Цепочка ответов

Рис. 2.5. Цепочки запросов/ответов с посредниками.

На рис. 4.5 между **UA** (агентом пользователя) и **O** (базовым сервером) показаны три посредника: A, B, и C, так что запрос или ответ, двигаясь по цепочке, должен пройти четыре различных соединительных сегмента v. Существуют три основных разновидности посредников: **прокси-сервера**, **шлюзы**, и **туннели**.

Прокси-сервер является агентом-посредником, который получает запросы на некоторый **URI**, и отсылает измененный запрос серверу, идентифицированному URI. Так что прокси-сервер выступает и как сервер (по отношению к конечному пользователю), и как клиент.

Шлюз — это принимающий агент, действующий как посредник для некоторого другого сервера и, в случае необходимости, транслирующий запросы в протокол сервера.

Туннель действует как реле между двумя соединениями, не изменяя сообщения; туннели используются тогда, когда связь нужно производить через посредника, который не понимает содержание сообщений.

Любой участник обмена в схеме рис 4.5, если только он не является туннелем, может воспользоваться кэшем, что позволяет снизить требования к пропускной способности канала.

Запрос клиента состоит из трех частей**: строки запроса, раздела заголовка и тела.**

При вводе URL Web-браузер, используя по умолчанию порт 80, устанавливает соединение с сервером, указанным в URL, по протоколу TCP. Далее браузер посылает **строку запроса**. Синтаксис **з**апроса:

МЕТОД<SP>**URI**<SP>**HTTP/версия**<CRLF>,

где <SP> — пробел,
<CRLF> — перевод на новую строку.

HTTP/версия: HTTP/1.0, HTTP/1.1. Может использоваться и старая версия HTTP/0.9. В этом случае в запросе должна присутствовать только строка запроса; такой запрос называется **Простым**. Он содержит метод доступа и запрос ресурса. Например:

GET http://sampo.karelia.ru/.

Здесь слово GET обозначает метод доступа GET, а http://sampo.karelia.ru – запрос ресурса.

Клиенты, которые способны поддерживать версии выше протокола HTTP/0.9 должны пользовать полной формой запроса. Полная форма запроса содержит заголовок запроса (Request Header), заголовок тела запроса (Entity-Header) и само тело запроса

Методы HTTP

Метод GET служит для получения информации, указанной в URI запроса. Он изменяется на условный GET если в заголовке присутствует поле с именем If-Modified-Since. Метод GET является наиболее распространенный метод поиска документов с помощью браузеров. Результатом запроса GET может быть файл, доступный для сервера, результат выполнения программы или CGI-сценария, выходная информация аппаратного устройства и т.д. Если клиент пользуется в своем запросе методом GET,то сервер отвечает строкой состояния, заголовками и затребованными данными. Если же сервер не может обработать запрос вследствие ошибки или отсутствия полномочий, то он, как правило, посылает в информационном разделе ответа текстовое пояснение.

Метод HEAD аналогичен методу GET, за исключением того, что сервер не посылает в информационную часть ответа (не возвращает тело ответа). Метод HEAD запрашивает только информацию о файле или ресурсе и используется тогда, когда клиент хочет найти информацию о документе, не получая сам документ. Например, клиент может затребовать информацию о времени изменения документа, о типе и размере документа и т.д.

Метод POST позволяет посылать на сервер данные в запросе клиента. Эти данные направляются в программу обработки данных, к которой сервер имеет доступ. Метод POST может использоваться для передачи входных данных сетевым службам; для выполнения операций в базах данных и т.д.. Данные, посылаемые на сервер, находятся в теле содержимого запроса клиента. По завершении обработки запроса POST и заголовков сервер передает тело содержимого в программу, заданную URL.

Метод OPTIONS представляет собой запрос информации о коммуникационных опциях, доступных в цепочке запрос/отклик. Метод позволяет клиенту определить опции и/или требования, связанные с ресурсами, или возможности сервера, не прибегая к операциям по извлечению и пересылке каких-либо файлов.

Метод PUT – служит для сохранения в указанном URI тела запроса. Если такой URI существует то тело запроса должно рассматриваться как модифицированная версия данного URI.

Метод DELETE служит для удаления данных, находящиеся на сервере по заданному URL.

Метод TRACE позволяет проследить цепочку (трассу) прохождения запроса.

Метод CONNECT (RFC 2616) предназначен для использования с прокси-серверами, которые способны динамически переключаться на работу в качестве туннеля. Изначально этот метод был рассчитан на работу с HTTPS протоколом (HTTP Secure – защищенный протокол). Однако этот метод может быть использован и для других целей, в результате появились CONNECT proxy, которые позволяют «обходить» блокировки корпоративного прокси.

HTTP ответ

В ответ на запрос клиента сервер посылает ответ, который состоит из **Строки состояния**, **Общего заголовка** (General-Header), **Заголовка ответа** (Response-Header), **Заголовка тела ответа** (Entity-Header) и **Тела ответа**.

3. ТЕХНОЛОГИИ WORLD WIDE WEB (WWW)

3.1. Компоненты начального этапа технологии WWW

На начальном этапе WWW включала следующие компоненты:

1. Язык гипертекстовой разметки документов HTML (HyperText Markup Language).

2. Универсальный способ адресации ресурсов в сети URL (Universal Resource Locator).

3. Протокол обмена гипертекстов**ой** информацией HTTP (HyperText Transfer Protocol).

4. Универсальный интерфейс шлюзов CGI (Common Gateway Interface).

Первые три компонента предложил Тим Бернерс-Ли**,** ему же принадлежит и название World Wide Web (1989 г.; проект «Гипертекст для ЦЕРН»). Последний четвертый компонент добавлен командой NCSA (Национальный центр суперкомпьютерных приложений).

Язык гипертекстовой разметки HTML предназначен для формирования гипертекстовой информации на основе структурированных документов. HTML является упрощенной версией языка разметки SGML (Standard Generalized Markup Language), утвержденного ISO в 1986 г. (ISO 8879).

Идея построения WWW на гипертекстовой основе оказалась революционной. Значимость гипертекстовой технологии такова, что в некоторых источниках ее ставят в один ряд с книгопечатанием.

Идею гипертекста предложил в 1945 году В. Буш, а сам термин в 1965 году ввел Т. Нельсон. Работу гипертекстового интерфейса впервые продемонстрировал в 1968 году изобретатель манипулятора "мышь" Д. Енжильбард. В 1975 году на атомном авианосце "Карл Винстон" была установлена гипертекстовая информационная система внутреннего распорядка ZOG (в коммерческом варианте известная как KMS).

Второй компонент: универсальный способ адресации ресурсов в сети URL (RFC 1738, RFC 1808) позволяет обеспечивать доступ как ресурсам WWW.

Назначение протокола HTTP в WWW то же, что и протокола SMTP в электронной почте; при этом и сообщения в обоих протоколах передаются в подобных форматах. Текущая версия HTTP/1.1 (RFC 2616) имеет расширенную функциональность за счет введения открытого набора методов, основанных на использовании Универсального Идентификатора Ресурса (URI; местоположение URL или имя URN). Стандартизацией HTML занимается World Wide Web Consortium (W3C).

Спецификация CGI (Common Gateway Interface) предназначена для расширения возможностей WWW путем подключения внешних программ (CGI-скриптов и Шлюзов)**,** запускаемых на Web-сервере. Спецификация CGI расширяет возможности Web-технологии, придавая ей динамический характер.

CGI-скрипт **—** это программа, написанная в соответствии со спецификацией CGI на любом языке программирования (Perl, C, C++, Java, Visual Basic, Delphi и

т.д.). Шлюз — это такой CGI-скрипт, который инициирует взаимодействие в качестве клиента с третьей программой.

3.2. Архитектура начального этапа технологии WWW

Архитектура взаимодействия программного обеспечения технологии WWW на начальном этапе приведена на рис. 3.1.

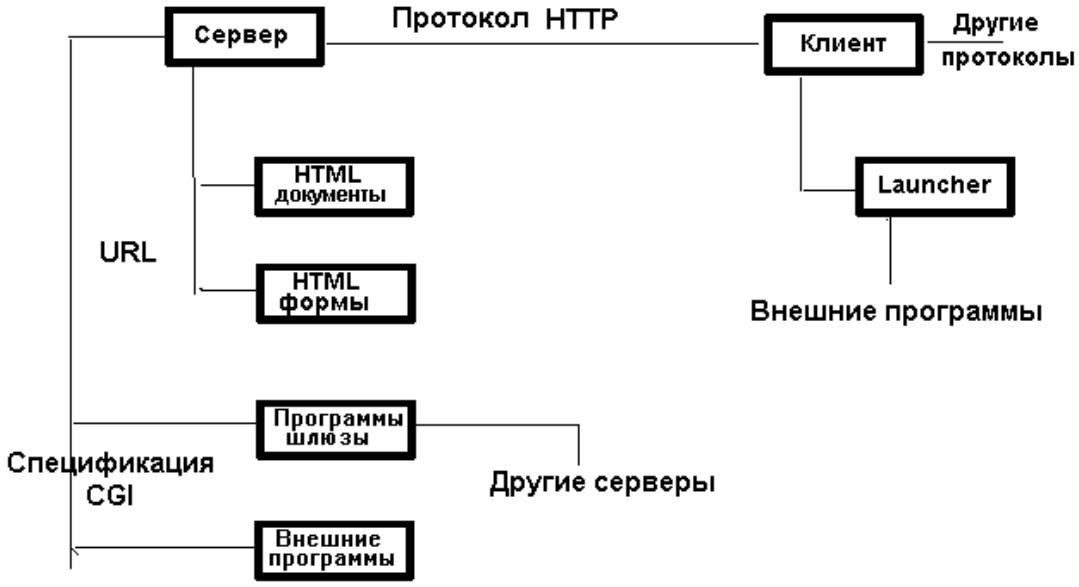


Рис. 3.1. Архитектура WWW-технологии на начальном этапе.

Клиент выполняет функции интерфейса пользователя и обеспечивает доступ к ресурсам Internet (не только по протоколу HTTP, но и по протоколам Telnet, FTP, SMTP и др.). К основным функциям клиента относятся: размещение текста на экране, обмен информацией с сервером, запуск внешних программ (посредством программы Launcher) для работы с документами в форматах, отличных от HTML

Web-сервер (сервер протокола HTTP) по запросу клиента осуществляет взаимодействие с этим клиентом, а также выполняет необходимую обработку информации с использованием базы данных документов, CGI-скриптов и шлюзов.

3.3. Второй этап развития Web-технологии

К середине 1996 года в концептуальной схеме Web-технологии произошли следующие основные изменения (см. рис.3.2).

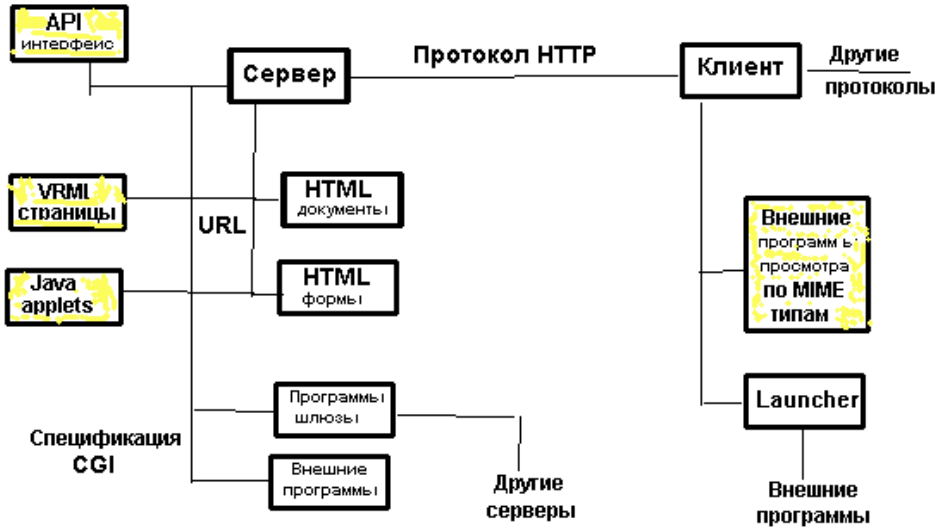


Рис. 3.2. Архитектура второго этапа Web-технологии.

Использование технологии API обусловлено тем, что спецификации API (Application Programming Interface; ISAPI для IIS Microsoft и NSAPI для серверов Netscape) обеспечивает большую эффективность, нежели интерфейс CGI, которому присущи ограниченные возможности масштабирования и невысокое быстродействие. При любом соединении с приложением CGI загружается новый экземпляр программы, так что при обращении 100 клиентов будет загружено 100 программ.

В технологии API приложение оформляется в виде динамически связываемой библиотеке (DLL). Приложение API загружается один раз, и все клиенты обращаются к одному экземпляру. Разработка приложений API сложнее, чем CGI. Приложение API пишутся на C, C++, Visual C++.

Язык виртуальной реальности VRML (Virtual Realty Modeling Language, первая версия появилась в 1994 г.) позволяет в рамках технологии WWW описывать трехмерные сцены и создавать смешанные базы данных. Например, можно создать архив в виде книжной библиотеки; выбранная в таком архиве книгу и тематику, которая затем представляются в формате документа HTML.

Концепция Java-applet'ов разработана специально для использования в WWW. При обработке HTML-документа, содержащего теги APPLET, апплеты (мобильные коды) будут подгружены, после чего они могут быть и выполнены. Тег APPLET введенный в HTML содержит имя апплета и параметры его вызова.

3.4. Третий этап развития технологии Web-технологии

Дальнейшее развитие архитектуры WWW связано с применением технологий ASP, XML, ASP.NET (см. рис. 3.3).

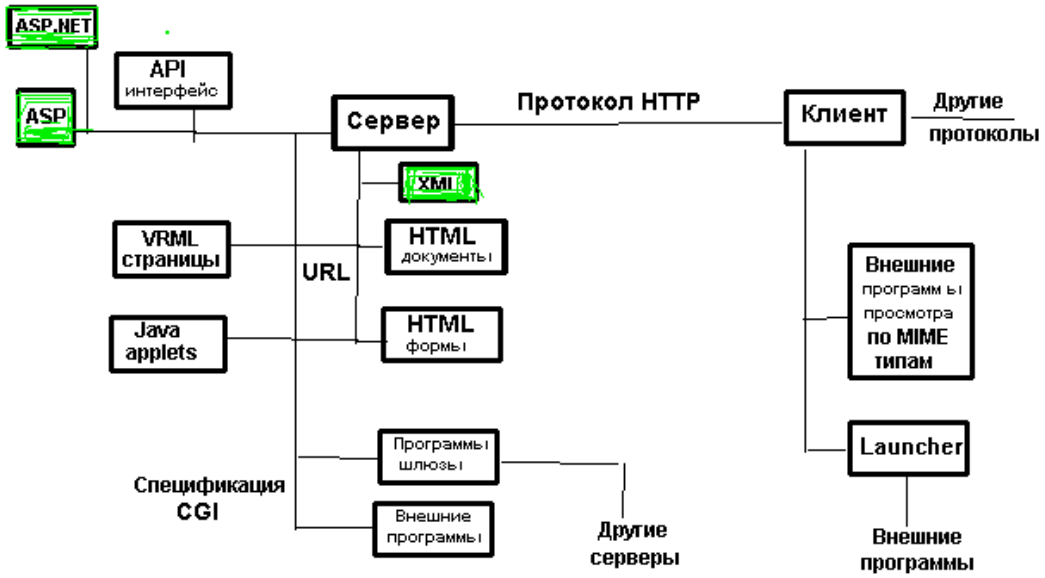


Рис. 3.3. Архитектура третьего этапа Web-технологии.

ASP (Active Server Pages, 1997 г.) — это активные страницы сервера. В IIS (Internet Information Server**)** Windows 2000 включена поддержка страниц ASP. Страницы ASP являются средой создания
серверных сценариев для разработки динамических интерактивных приложений Web -серверов.

Успех технологии ASP обусловлен простым скриптовым языком Visual Basic Script (используются по умолчанию) или Java Script. Программы можно писать на любом скриптовом языке, установленном в системе. Созданная программа помещается в файл на сервере (файлы с программами имеют расширение .asp). При запросе этого файла браузером клиента файл интерпретируется сервером и посылается клиенту в виде HTML-кода. ASP можно рассматривать и как приложение API. ASP, как и API выполняется в том же адресном пространстве.

Разработка языка XML (Extensible Markup Language, 1998 г.) обусловлена недостатками языка HTML, который не связан с содержанием документа и поэтому не обеспечивает возможность работы над содержимым сайта независимо от его внешнего представления. Язык XML является простым подмножеством SGML, в котором используются лишь реально необходимые в Internet возможности SGML. Язык XML в отличие от HTML, который является приложением, а не подмножеством SGML, позволяет определять структурные элементы документа (собственные теги и атрибуты), а также создавать сложные документы. XML представляет собой метаязык, то есть язык, на базе которого можно создавать новые языки. Он ориентирован не только для создания средств, служащих для организации обмена данными в Web, но и для распознавания семантики этих данных. Сам по себе XML не содержит никаких тэгов для разметки, он просто определяет порядок их создания

Технология ASP**.**NET похожа на ASP, но внутренняя реализация ASP.NET переделана; ASP.NET включает поддержку Web Forms, что позволяет

разрабатывать web-приложение как обычное приложение для Windows. Основное же достоинство технологии ASP.NET состоит в том, что она обеспечивает межплатформаенность путем использования промежуточного языка MSIL (Microsoft Intermediate Language) для которого у каждой аппаратной платформы и программной среды должен быть свой промежуточный компилятор.

3.5. Информационно-поисковые системы Internet

Поиск информации в Internet осуществляется путем использования: поисковых машин, метапоисковых систем, индексированных каталогов, онлайновые энциклопедий и справочников. Поисковые порталы содержат не только набор поисковых средств, но и обеспечивают многочисленные дополнительные услуги (сервисы). По-видимому, наиболее продвинутым порталом в области поисковых сервисов является Google, который является и лидером по качеству поиска.

Поисковые машины, осуществляя постоянное сканирование с помощью программы робота (spider; паук) доступных узлов Интернета, скачивают найденные страницы в базу данных и формируют специальную базу данных, в которой хранится индексированная информация о скаченных страницах

При поступлении запроса поисковая машина, используя индексированную информацию, выдает список документов, ранжированных по местоположению ключевых слов в поисковом запросе, их частоте в тексте и другим параметрам. Имея схожий принцип работы, поисковые машины, тем не менее, различаются по используемым алгоритмам и принципам поиска, которые к тому же постоянно совершенствуются; поэтому результаты поиска у разных машин отличаются.

Наиболее популярными среди нескольких сотен различного вида поисковиков являются поисковые средства следующих порталов.

Google (http://www.google.com/) мировой лидер по объему проиндексированных документов и скорости обработки запросов. Google самая популярная поисковая система, обеспечивает поиск независимо от языка источника, проста в использовании, имеет хороший язык запросов и простой интерфейс.

Yahoo! (http://www.yahoo.com/). Как поисковая система является наиболее авторитетным справочником ресурсов Интернета. Выдает по поисковой теме максимальное количество зарубежных сайтов; обеспечивает поиск на русском языке.

Baidu (http://www.baidu.com/). Лидер китайских поисковых систем. По количеству обрабатываемых запросов стоит на 3 месте в мире.

Jandex (http://www.yandex.ru/) крупнейший российский портал, предлагающий пользователям многочисленные услуги, включая поисково-информационные (12 служб). Робот поисковой системы Яндекс постоянно сканирует Интернет, автоматически отслеживая изменения; результаты поиска упорядочиваются в соответствии с установленными критериями релевантности

(степени соответствия искомого и найденного). Поисковая система обеспечивает возможность расширенного поиска, позволяющего детализировать многочисленные параметры поиска.

Rambler (http://www.rambler.ru/) российский поисковый портал. Как профессиональная информационно-поисковая система Rambler существует с 1996 года. Обеспечивает различные виды поиска, включая расширенный поиск по комбинации различных параметров с учетом морфологии русского языка.

Aport! (http://aport.ru/) российский поисковый портал Апорт имеет расширенные возможности по формулированию запросов; обеспечивает поиск графических изображений и мультимедийных файлов.

4. ОБЛАЧНЫЕ ТЕХНОЛОГИИ И СЕРВИСЫ

4.1. Понятие облачной технологии

Облачные технологии (облачные вычисления; Cloud Computing) представляют собой быстро развивающуюся сферу человеческой деятельности; являются стратегическим направлением компьютерной индустрии.

Понятие Cloud Computing, как и любое другое понятие имеет три аспекта: содержание понятия (совокупностью отличительных признаков), объем понятия (совокупность объектов, отображенных в понятии) и дефиниция (определение понятия). При этом между содержанием и объемом понятия существует обратная зависимость, т.е. чем больше признаков включено в содержание понятия, тем меньше объектов оно охватывает и тем в более узком смысле оно трактуется.

Дефиниция дает лишь самое общее представление о понятии, поэтому обычно прибегают к рассмотрению основных свойств понятия и его классификации (таксономии). Рассмотрим некоторые существующие определения Cloud Computing и свойства этого понятия.

Cloud computing – это программно-аппаратное обеспечение, доступное пользователю через Интернет или локальную сеть в виде сервиса, позволяющего использовать удобный интерфейс для удаленного доступа к выделенным ресурсам (вычислительным ресурсам, программам и данным).

Облачные вычисления представляют собой динамически масштабируемый способ доступа к внешним вычислительным ресурсам в виде сервиса, предоставляемого посредством Интернета, при этом пользователю не требуется никаких особых знаний об инфраструктуре «облака» или навыков управления этой «облачной» технологией.

Облачные вычисления - это новый подход, позволяющий снизить сложность ИТ-систем, благодаря применению широкого ряда эффективных технологий, управляемых самостоятельно и доступных по требованию в рамках виртуальной инфраструктуры, а также потребляемых в качестве сервисов. Переходя на частные облака, заказчики могут получить множество преимуществ, среди которых снижение затрат на ИТ, повышение качества предоставления сервиса и динамичности бизнеса».

«Облако» является новой бизнес-моделью для предоставления и получения информационных услуг. Эта модель обещает снизить оперативные и капитальные затраты. Она позволяет ИТ департаментам сосредоточиться на стратегических проектах, а не на рутинных задачах управления собственным центром обработки данных.

Облачные вычисления – это не только технологическая инновация в ИТ, но и способ создания новых бизнес-моделей, когда у небольших производителей ИТ-продуктов, в том числе и в регионах, появляется возможность быстрого предложения рынку своих услуг и мало затратного способа воплощения своих бизнес-идей. Поддержка облачных вычислений в сочетании с инвестициями в молодые компании создают быстро развивающуюся экосистему инновационных производств.

Облачные вычисления являются рыночным ответом на систематическую специализацию и усиление роли аутсорсинга в ИТ. По сути, переход к облачным вычислениям означает аутсорсинг традиционных процессов управления ИТ-инфраструктурой профессиональными внешними поставщиками. Большинство современных поставщиков решений сферы облачных вычислений

предоставляет возможность не только использовать существующие облачные платформы, но и создавать собственные, отвечающие технологическим и юридическим требованиям заказчиков.

Подводя итог приведенным определениям отметим, что облачные технологии представляют собой агрессивный высокотехнологичный прорыв, затрагивающий практически все IT-сферы человеческой деятельности. Основное достоинство облачных технологий состоит в том, что они позволяют доставить пользователю требуемую информационную технологию в качестве сервиса так, как этого требуют правила логистики: нужный товар, в нужном месте, нужного количества, нужного качества, в нужное время, с минимальными затратами. Пользователю, который может находиться в любой географической точке, необходим только тонкий клиент и высокоскоростной канал; при этом нужный товар (сервис) предоставляется практически мгновенно с требуемой виртуальной инфраструктурой и программным обеспечением. Пользователь, заказывая необходимые аппаратно-программные средства, платит только за хостинг, при этом вопросы модернизации, поддержки и лицензирования являются прерогативой поставщика сервиса.

Если сказать еще короче, то основное достоинство облачных технологий состоит в том, что они обеспечивают пользователю, при прочих равных условиях, значительно большее пространство технологической свободы.

4.2. Классификация облачных технологий

Главными классификационными признаками облачных технологий являются содержание предоставляемых услуг и принадлежность или местоположение сервис центра.

Различают следующие виды облачных технологий (моделей развертывания Cloud Computing):

Частное облако (Private Cloud),
Публичное облако (Public Cloud),
Общее облако (Community Cloud),
Гибридное облако (Hybrid Cloud).

Облачные технологии (вычисления) характеризуются следующими основными видами предоставляемых услуг (сервисов):

Инфраструктура как сервис (Infrastructure as a Service; IaaS);
Платформа как сервис (Platform as a Service; PaaS);
Программное обеспечение как сервис (Software as a Service; SaaS).

Наиболее широкие технологические возможности обеспечивает сервисная модель IaaS, так как в этом случае сервис предоставляется на уровне инфраструктуры. На базе IaaS реализуются сервисы PaaS и SaaS, а также другие сервисы.

Модель IaaS позволяет запускать практически любое приложение. Это как раз то, что необходимо для специальной подготовки IT-специалиста. Сервис IaaS может быть реализован как в публичном, так и в частном облаке.

Модель PaaS, обеспечивает доступ к технологическим платформам (к операционным системам и прикладным программам).

Модель облачных приложений SaaS представляет собой наиболее распространенный вид облачных сервисов. По существу все приложения, которые установлены не у пользователя, а в «облаке» публичном, частном, общем или гибридном, относятся к модели SaaS.

В заключение отметим, что облачные вычисления базируются не только на новациях в области аппаратных средств, но и являются результатом синтеза ряда современных технологий и подходов (см. рис. 4.1).

Рис. 4.1. Технологии и подходы облачных вычислений.

Важнейшей технологией в области облачных вычислений является технология виртуализации, но виртуализация не единственная технология, лежащая в основе современной облачной парадигмы. Важную роль сыграли и другие технологии, такие как сервис- ориентированная архитектура (Service-Oriented Architecture, SOA), технология ASP, технологии Web 2.0 и др.

По существу современные облачные технологии и начинаются с SOA компании Amazon, которая в 2006 году запустила платформу Amazon Web Service; в 2008 году анонсированы облачные платформы Windows Azure и Google App Engine. Официальный релиз Windows Azure состоялся в 2010 году (с 2014 года — это Microsoft Azure).

4.3. Аппаратная база облачной технологии

Оптимизация тройки: мощность, производительность, цена процессора привели, в конечном счете, к необходимости создания многопроцессорных систем (дальнейшее увеличение мощности процессора требовало дорогостоящих технологий его охлаждения), а затем и многоядерных вычислительных систем.

Названные технологические новации потребовали новых подходов к размещению, электропитанию, охлаждению и обслуживанию серверов. В результате создан новый тип серверов XXI века — сервер-лезвие (blade-сервер). Первые модели blade-серверов разработаны в 2001 г. По сравнению с обычными серверами равной производительности blade-серверы занимают в два раза меньше места, потребляют в три раза меньше энергии и обходятся в четыре раза дешевле, так что их описывают с помощью правила «1234».

Blade-сервер («лезвие») представляет собой модульную одноплатную компьютерную систему, содержащую процессор и память**.** Лезвия вставляются в специальное шасси с объединительной панелью (backplane), обеспечивающей подключение к сети и подачу электропитания.

Шасси с лезвиями составляет Blade-систему. Шасси представляет собой стандартную 19-дюймовую стойку, которая, в зависимости от модели и производителя,

занимает 3U, 6U или 10U (здесь U - unit, или монтажная единица, равная 1,75 дюйма).

Лидерами производства блейд-систем являются компании Hewlett-Packard, IBM, Dell, Fujitsu Siemens Computers, Sun.

Основные преимущества Blade-серверов

1. Снижение энергопотребления и повышение надежности, сокращение занимаемой площади и кабельного хозяйства.

2. Высокая управляемость и гибкость. Для управления системой не требуется клавиатура, видео и мышь; управление осуществляется удаленно с помощью централизованного модуля и специального процессора.

3. Масштабируемость. Увеличение производительных мощностей осуществляется посредством дополнительных лезвий.

4. Повышенная надежность. Блейд-системы имеет встроенные средства резервирования. Наличие нескольких блоков питания (при выходе из строя одного блока питания) обеспечивает бесперебойную работу серверов, расположенных в шасси. Дублируются и охлаждающие компоненты. Наконец при выходе из строя одного из серверов лезвие будет заменено на новое.

5. Снижение эксплуатационных расходов. Применение блейд-архитектуры приводит к уменьшению энергопотребления и выделяемого тепла, а также к уменьшению занимаемого объема. При этом инфраструктура блейд систем является более простой в управлении.

4.4. Технологии виртуализации

Понятие «виртуализация» имеет широкий спектр применений: виртуальная реальность, виртуальная память, виртуальный канал и т.д. В компьютерной индустрии под виртуализацией понимают разнообразные методы, служащие для абстрагирования от различных физических ресурсов компьютеров.

Термин «виртуализация» в компьютерных технологиях появился в шестидесятых годах прошлого века вместе с термином «виртуальная машина». Виртуальная машина (ВМ) представляет собой изолированный программный контейнер с собственной операционной системой и приложениями. Так что ВМ действует так же, как и физический компьютер. ВМ содержит собственные виртуальные ОЗУ, жесткий диск и сетевой адаптер.

Начало виртуализации положила компания IBM в 60-х годах прошлого века, но настоящий интерес к виртуализации возник в конце девяностых годов прошлого века. В 1999 г. компания VMware представила технологию виртуализации систем на базе x86, затем в игру вступили компании Parallels, Oracle и Citrix Systems. Корпорация Microsoft выпустив свой первый продукт Virtual PC для настольных ПК в 2003 г.

Внедрение технологий виртуализации обусловлено тем, что средний уровень загрузки серверов в традиционных технологиях не превышает 10%, а это — повышенное энергопотребление, дополнительные площади, лицензии. Современные же технологии виртуализации обеспечивают загруженность серверов до 85%. При этом технология виртуализации позволяет: сократить расходы на инфраструктуру; снизить затраты на программное обеспечение; обеспечить высокую управляемость инфраструктуры и непрерывную работу предприятия. Технология виртуализации обеспечивает: возможность работы на одном компьютере

несовместимых приложений; миграцию программного обеспечения и данных на новые платформы; возможность сохранения аппаратной и программной среды для унаследованных приложений.

Различают следующие основные разновидности виртуализации:

виртуализация на уровне машины,

виртуализация на уровне операционных систем,

виртуализация приложений,

виртуализация рабочего стола.

Виртуализация на уровне машины

В основе виртуализации на уровне машин лежит монитор виртуальных машин (Virtual Machine Monitor; VMM), который отвечает за создание и изоляцию ВМ, сохранение ее состояния, а также за ограничение доступа к системным ресурсам. Схема VMM зависит от архитектуры конкретного процессора.

Различают три вида реализации VMM:

VMM работает над операционной системой хоста (как Java VM);

Гибридная реализация, в которой VMM работает как равноправный с операционной системой модуль (Virtual Server 2005 R2);

VMM работает как гипервизор (Microsoft Hyper-V).

При реализации VMM для создания интерфейса между ВМ и виртуальными системными ресурсами используются три метода: полная виртуализация, собственная виртуализация и паравиртуализация.

В полной виртуализации (Full, Native Virtualization) используются не модифицированные экземпляры гостевых ОС. Такая технология применяется, в частности, в VMware Workstation, VMware Server, Parallels Desktop, Parallels Server, MS Virtual PC, MS Virtual Server, Virtual Iron. К достоинствам данного подхода – относительная простота реализации, универсальность и надежность решения. Недостатки — высокие дополнительные накладные расходы на используемые аппаратные ресурсы, отсутствие учета особенностей гостевых ОС, меньшая гибкость в использовании аппаратных средств.

При собственной виртуализации процессор имеют в своей аппаратной части новые режимы, инструкции и структуры данных (серии процессоров AMD-V и Intel VT). Такой вид реализации имеет ряд преимуществ: упрощение архитектуры VMM, повышение производительности, более высокой плотности разделов в системе. Hyper-V использует этот метод для эмуляции работы устаревших систем.

В случае паравиртуализации модифицируется ядро гостевой операционной системы путем включения нового набора API, позволяющего бесконфликтно и напрямую работать с аппаратурой. При этом функции управления реализует специальная система, называемая гипервизором (hypervisor). Этот вид виртуализации применяется в VMware ESX Server, Xen, Microsoft Hyper-V.

Виртуализация на уровне ОС

Для создания независимых параллельно работающих операционных сред используется ядро хостовой ОС. Для гостевого ПО создается собственное сетевое и аппаратное окружение. Такой вариант используется в Virtuozzo (для Linux и Windows), OpenVZ (бесплатный вариант Virtuozzo) и Solaris Containers. Достоинства — высокая эффективность использования аппаратных ресурсов, низкие накладные расходы, отличная управляемость, минимизация расходов на приобретение лицензий. Недостатки: возможность реализация только однородных вычислительных сред.

Виртуализация приложений

Виртуализируется каждый экземпляр приложения и его основные компоненты. Приложение исполняется без инсталляции, может запускаться с внешних носителей (с флэш-карт или из сетевых папок). Преимущества виртуализации приложений: быстрое развертывание настольных систем; сведение к минимуму конфликтов между приложениями. Данная технология позволяет использовать в одной и той же операционной системе несколько несовместимых между собой приложений одновременно. Такой вариант виртуализации используется в Sun Java Virtual Machine, Microsoft Application Virtualization.

Виртуализация рабочего стола

При виртуализации рабочего стола пользователь видит удалённый рабочий стол и запущенные на нём приложения, которые выполняются на удаленном сервере. Пользователи дистанционно подключаются к виртуальной машине своей настольной среды. На локальных компьютерах пользователей в качестве удаленного настольного клиента могут применяться терминальные клиенты и старое оборудование. Виртуализация рабочих столов обычно реализуется при помощи инфраструктуры Virtual Desktop Infrastructure (VDI), которая составляет комбинацию аппаратного и программного обеспечения виртуализации, а также инструментов управления. Перспективной технологией виртуализации рабочего стола является технология тонких терминалов. Все команды пользователя и изображение сеанса на мониторе эмулируются с помощью ПО управления тонкими клиентами. Применение этой технологии позволяет централизовать обслуживание клиентских рабочих мест и резко сократить расходы на их поддержку.

4.4. Облачные сервисы

Классические облачные сервисы IaaS, PaaS и SaaS рассмотрены в п.4.2. Кроме классических поставщики услуг могут представлять и другие облачные сервисы, например, DaaS (Desktop as a Service) — доступ клиента к рабочему столу с предустановленной операционной системой и необходимым набором программного обеспечения; HSS (Hosted Security as a Service) — облачная безопасность; WaaS (Workspace as a Service) — рабочее окружение как сервис (в отличие от DaaS пользователь получает доступ только к ПО при этом все вычисления происходят на его машине) и др.

Наиболее распространенным видом облачных сервисов, как уже отмечалось, является SaaS. Сервисы модели SaaS практически необозримы; они постоянно и динамично развиваются. В качестве примера можно привести сервисы Google, в частности служба Google Docs позволяет создать или экспортировать файлы из офисных программ Microsoft Office и OpenOffice (текстовые документы, таблицы, презентации), причем интернет страницей могут работать несколько человек одновременно. Качество данного сервиса таково, что многие компании на западе полностью переходят на работу с документами в Google Docs. В силу достоинств SaaS все больше компаний переходят от «коробочных» решений к сервисам SaaS, На обеспечение облачных офисных технологий ориентированы продукты Microsoft Office 365 и SharePoit Foundation.

Общемировой тенденцией в настоящее время является проведение вебинаров по технологии SaaS. Вебинар (webinar) является удобным и многофункциональным средством при проведении научных онлайн конференций и семинаров, презентаций и рекламных акций, онлайн консультаций, технической поддержки и др.

Все большую значимость приобретают вебинары и в обучении. Технологии вебинара позволяют проводить занятия в реальном времени с использованием экрана и

доски; слушатели по ходу обучения могут задавать вопросы (как голосом, так и чате), слышать и видеть комментарии коллег. Практические занятия (тренинг) выполняются с использованием удаленного рабочего стола, на котором развернута необходимая виртуальная инфраструктура (такая же как, и у слушателей в классе), преподаватель видит действия слушателя и может управлять его рабочим столом. Так что в данном случае вебинар, реализованный по технологии SaaS, включает электронный тренинг на основе IaaS.

Облачный сервис позволяет хранить данные в интернете (фото, видео, музыку, контакты, приложения и многое другое), можно получить доступ к любому файлу с любого компьютера, читать ленты новостей, обмениваться мгновенными сообщениями и использовать социальными закладками и т.д. Существует большое количество облачных сервисов, которые позволяют даже совместно (по электронной почте) редактировать документы (google docs). Детальное ознакомление с облачными сервисами (см., например, **Сервисы Google)** которые постоянно и динамично развиваются, выходит за рамки данного курса.

4.5. Сервис ориентированная архитектура

Сервис-ориентированная архитектура (Service-Oriented Architecture, SOA), – это не только одна из концептуальных технологий облачных вычислений, но и современный подход построения корпоративных систем, в основе которого лежит идея интеграции слабосвязанных веб-сервисов. Веб-сервисы имеют стандартные интерфейсы и могут взаимодействовать как друг с другом, так и с приложениями, созданными на основе SOA. Такой подход обеспечивает необходимую гибкость, позволяющую своевременно реагировать на изменение рыночных условий или условий ведения бизнеса.

В обобщенном виде SOA включает поставщика сервиса, потребителя сервиса и реестр сервисов (рис. 4.2.). Поставщик сервиса регистрирует сервисы в реестре, потребитель обращается с запросом требуемого сервиса к реестру.

Сервис, который описывается на языке описания веб-сервисов WSDL (Web Services Description Language), в данной схеме представляет собой функцию, которая четко определена, самодостаточна и не завит от контекста или состояния других сервисов. Сервис может иметь любой размер, но должен представлять собой законченный функциональный объект, прячущий внутреннюю технологию. Взаимодействие внешних программ с сервисом осуществляется по простому протоколу доступа к объектам SOAP (Simple Object Access Protocol), который основан на HTTP запросах. Протокол SOAP отправляет серверу HTTP запросы, содержащие SOAP запрос в виде XML схемы. Результат работы сервис возвращает по HTTP протоколу, в который «завернут» SOAP ответ в виде XML схемы.

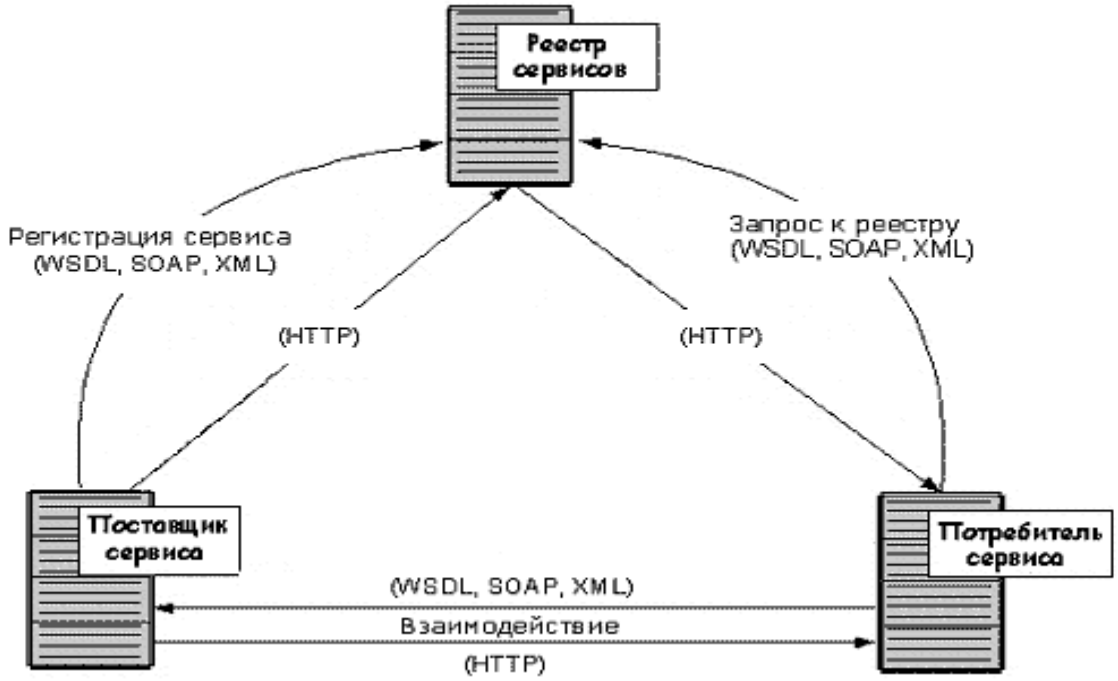


Рис. 4. 2. Обобщенная схема SOA.

Основными поставщиками решений для SOA являются IBM, Oracle, Sun, Microsoft. Однако десятилетняя попытка внедрения дорогостоящих программных комплексов, построенных на нисходящей модели SOA, оказалась недостаточно успешной. В настоящее время применяется восходящая модель SOA, которая предполагает отказ от дорогостоящей глобализации, с внедрением сервисов начиная с уровня рабочих групп. При этом могут быть использованы веб-сервисы SaaS, имеющие открытые API.

5. СОВРЕМЕННЫЕ WEB-ТЕХНОЛОГИИ

5.1. Основные компоненты Web-технологии

Перечислим основные компоненты и технологии WWW, приведенные в разделе 3. Начальный этап WWW включал: HTML, URL, HTTP и CGI; на втором этапе появились: API, VRML, Java applets; на третьем — XML, ASP и ASP.NET. Это далеко не полный перечень средств WWW. Необходимо назвать языки программирования PHP, Perl, Ruby, Python, JAVA; а также платформы Adobe Flash, SilverLight, JavaFX и фреймворк Ruby on Rails.

Названные этапы часто связывают с понятиями: эра WWW или Web-платформа. Различают платформы: Web 0 (эра PC), Web 1, Web 2, Web 3.

С платформой **Web 0** (до 1990 г.) связаны Telnet, FTP, Email, USENET, IRC, BBS («Bulletin Board System»), Gopher и др.;

Web 1 (1990 – 2000) — HTTP, HTML, CGI, SOAP, Java, Flash др. ;

Web 2 (2000 – 2010) — XML, RDF, RSS, AJAX;

Web 3 (начиная с 2010; Semantic Web) — OWL, SPARQL и др.

Основополагающая база Веб-технологий (**Web 1.0)** разработана в первом десятилетии Интернет; контент Интернет-ресурсов, как правило, формировался профессионалами, пользователи сети Интернет представляли собой обычных читателей.

Для Web 2.0 характерно наличие развитых пользовательских интерфейсов, позволяющих пользователям создавать контент и управлять содержимым приложения. Термин Web 2 получил распространение в 2004 году с подачи Тима О'Рейли. Ключевыми для Web 2 являются: Web-сервисы, Социальное ПО, AJAX, Веб-синдикация (одновременное распространение информации на различные страницы или Web-сайты) и др. Концепция (технология) **AJAX** (Asynchronous Javascript and XML — «асинхронный JavaScript и XML») базируется на следующих основных принципах: динамического обращения к серверу «на лету» (без перезагрузки) и использование DHTML для динамического изменения содержания страницы. AJAX уменьшает нагрузку на сервер, сокращает трафик и ускоряет интерфейсную реакцию.

Платформу Web 3.0 связывают с технологиями семантической обработки информации. Семантический Веб (Semantic Web) или Семантическая паутина является глобальной концепцией развития Интернет, которая ориентирована на машинную работу с метаданными Всемирной паутины.

Основной акцент концепции семантическая паутина (Semantic Web) — работа с метаданными, однозначно характеризующими свойства и содержание ресурсов Всемирной паутины. Термин впервые ввел Тим Бернерс-Ли в 2001 г. В семантической паутине предполагается использование, унифицированных идентификаторов ресурсов (URI), онтологий и языков описания метаданных.

Текущая структура и синтаксис URI (Uniform Resource Identifier) регулируется стандартом RFC 3986 (январь 2005 г.). URI вобрал как возможности URL, так и URN (Uniform Resource Name; унифицированное имя

ресурса), что позволяет указывать как местонахождение ресурса (URL), так и идентифицировать его в пространстве имен (URN). Если не указывать местонахождение, то можно описывать произвольные ресурсы (личности, книги и т.п.).

Традиционная схема использования URI сводится к установке ссылок, ведущих на объект, им адресуемый. Таким объектом может быть веб-страница, файл произвольного содержания, фрагмент веб-страницы, а также неявное указание на обращение к реально существующему физическому ресурсу по протоколу, отличному от HTTP. Концепция семантической паутины расширяет это понятие, включая в него ресурсы, недоступные для скачивания. Адресуемыми с помощью URI ресурсами могут быть, например, люди, города и т. д. К идентификатору предъявляются простые требования: он должен быть уникальной строкой определённого формата, адресующей реально существующий объект. В семантической паутине используются форматы описания, доступные для машинной обработки RDF, RDF Schema, язык описания онтологий OWL и язык запросов к RDF SPARQL.

5.2. **Технология HTML**

Работу по созданию спецификации HyperText Markup Language (HTML) - гипертекстовый язык разметки осуществляет World Wide Web Consortium (W3C). Существуют спецификации 2.0, 3.0, 3.2 (1997 г.) , 4.0 (1997 г.), 4.01(1979 г.) и 5.0 (2014 г.). HTML является приложением SGML и соответствует стандарту ISO 8879. XHTML же является приложением XML.

Остановимся на краткой характеристике стандартных версий HTML. Стандартные версии начинаются с HTML 2.0 (1995 г.; RFC 1866), которая включила оригинальные материалы предыдущих версий (иногда называемых HTML 1.0) и содержала новые возможности. Версия HTML 2.0 была стандартом в области дизайна вебсайтов до 1997 года.

Версия HTML 3.0 предложена Консорциумом W3C в марте 1995; обеспечивала много новых возможностей (создание таблиц, «обтекание» изображений текстом, отображение математических формул). Версия 3.1 официально никогда не предлагалась. В версии 3.2 были опущены многие нововведения версии 3.0 и добавлены нестандартные элементы, поддерживаемые браузерами Netscape Navigator и Mosaic.

В версии HTML 4.0 в связи с использованием таблицы стилей CSS многие элементы, связанные с форматированием, были отмечены как устаревшие и не рекомендованные.

Спецификация HTML 4.01 (W3C Recommendation 24 December 1999) в дополнение к HTML 4.0 обеспечивает поддержку большего количество опций мультимедиа, скриптов, каскадных таблиц стилей, улучшенные возможности печати и большую доступность документов для людей с ограниченными возможностями. Спецификация HTML 4.01 представляет собой многостраничный документ, в котором достаточно подробно рассмотрены различные аспекты HTML, включая понятийный аппарат, историю и др.

Интенсивное и экстенсивное развитие информационных технологий с широким применением мобильных средств породило многочисленные проблемы, в числе которых и стратегическая проблема: необходимость автоматизации семантической обработки информации. Сложился разрыв между потребностями практики и состоянием информационной индустрии. Широко распространённые технологии HTML 4, ориентированные в основном на работу с формой, уже не отвечали возросшим требованиям.

Попытки внедрения консорциумом W3C (World Wide Web Consortium) в HTML 4 технологий XML (версии XHTML 1.0, XHTML 1.1, XHTML Basic, XHTML MP и, наконец, незавершённой версии XHTML 2.0) оказалась тупиковой, поскольку веб-разработчики, поддерживаемые производителями браузеров, не хотели принимать жесткие синтаксические правила, унаследованные из XML.

Противостояние известных производителей браузеров и консорциума W3C, по поводу развития HTML в нужном для пользователей направлении, привело в 2004 году к образованию группы WHATWG (Web Hypertext Application Technology Working Group), которая в противовес W3C, разработала ряд спецификаций, объединенных в проект Web Applications 1.0 (2006 год). В 2006 году консорциум W3C принял решение работать совместно с WHATWG над развитием HTML, переименовав при этом Web Applications 1.0 в HTML5. В 2011 году WHATWG отказалась от упоминания версии HTML5, заменив ее простым названием: HTML и зарезервировало его на все последующие версии.

Стандарт WHATWG HTML (https://html.spec.whatwg.org/multipage/) называется «живым» (Living Standard), поскольку в него постоянно вносятся изменения. Такой подход к названию HTML существует параллельно стандарту HTML5.

Рекомендация HTML5 опубликована консорциумом W3C 28 октября 2014 года; в настоящее время W3C работает над версией HTML5.1 (предполагаемый выход рекомендации – 2016 г.).

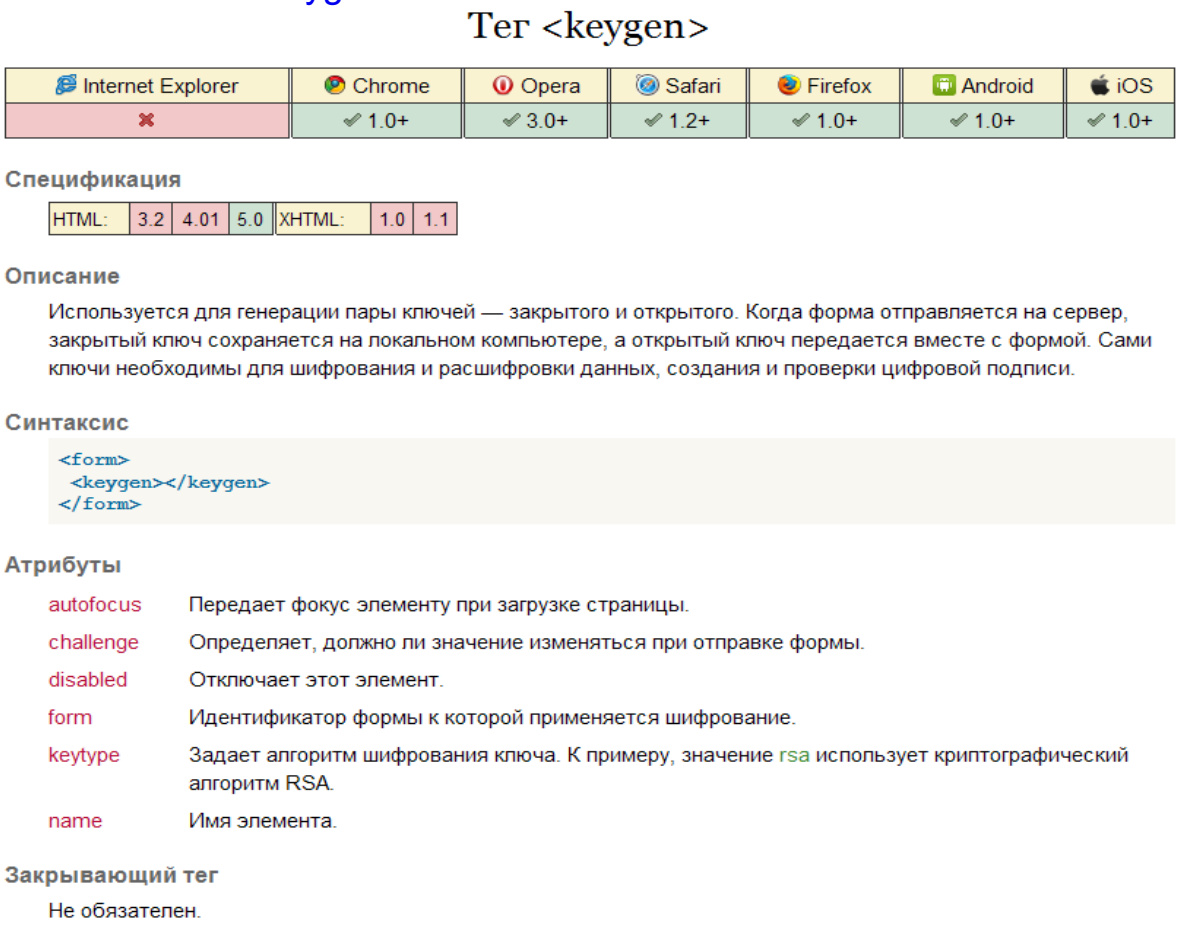
Спецификация HTML5 (http://www.w3.org/TR/html5/); по объему значительно превышает спецификацию HTML 4.01 и содержит многочисленные изменения и дополнения. Значительная часть спецификации посвящена вопросам разработки браузеров и совместимости различных реализаций HTML5.

Спецификация, сохраняя обратную совместимость, дополняет, развивает и модифицирует HTML 4.1 и XHTML 1.0. Единый язык позволяет использовать как HTML-синтаксис, так и XML-синтаксис. Большинство элементов HTML 4.1 поддерживается и в HTML5; часть элементов HTML 4.1 переопределены или доработаны; добавлены многочисленные новые элементы и атрибуты.

Значительна часть элементов и атрибутов HTML 4.1 в HTML5 объявлены как устаревшие (*deprecated),* но они поддерживается ради обратной совместимости. Список таких тегов приведен в справочнике по HTML Мержевича В.В. (http://htmlbook.ru/metki/spravochnik). Полный список изменений в HTML5 приведен здесь: http://www.w3.org/TR/html5-diff/ .

Вот, например, только не полный список новых элементов HTML5: header, footer, nav, section, article, aside, hgroup, figure, figcaption, video, audio, embed, mark, meter, time, canvas, command, details, keygen, output. этом справочники и

другие источники. Используя справочник по HTML Мержевича В.В. (http://htmlbook.ru/metki/spravochnik) получим, например, описание наиболее простого элемента keygen:



Тег <keygen>

Internet Explorer	Chrome	Opera	Safari	Firefox	Android	iOS
✖	✔ 1.0+	✔ 3.0+	✔ 1.2+	✔ 1.0+	✔ 1.0+	✔ 1.0+

Спецификация

HTML:	3.2	4.01	5.0	XHTML:	1.0	1.1

Описание

Используется для генерации пары ключей — закрытого и открытого. Когда форма отправляется на сервер, закрытый ключ сохраняется на локальном компьютере, а открытый ключ передается вместе с формой. Сами ключи необходимы для шифрования и расшифровки данных, создания и проверки цифровой подписи.

Синтаксис

```
<form>
  <keygen></keygen>
</form>
```

Атрибуты

autofocus	Передает фокус элементу при загрузке страницы.
challenge	Определяет, должно ли значение изменяться при отправке формы.
disabled	Отключает этот элемент.
form	Идентификатор формы к которой применяется шифрование.
keytype	Задает алгоритм шифрования ключа. К примеру, значение rsa использует криптографический алгоритм RSA.
name	Имя элемента.

Закрывающий тег

Не обязателен.

Рисунок. 5.1. Описание элемента keygen.

Как видно, на рисунке приведено не только описание тега keygen, синтаксис его написания, допустимые атрибуты и пример использования, но и таблица браузеров, которые его поддерживают.

Элементы HTML5 video, audio и canvas имеют API, позволяющие создавать видео, аудио, графику и управлять мультимедийными и графическими объектами без использования сторонних API и плагинов. К API HTML5 относятся также API технологий SVG, MathML, Grad and Drop, GeoLocation, Web Storage, Web Workers.

SVG (Scalable Vector Graphics; масштабируемая векторная графика) — язык разметки масштабируемой векторной графики, основанный на XML. Используется в HTML5 для создания векторной графики, в спецификацию HTML5 не входит.

MathML (Mathematical Markup Language; язык математической разметки) — язык разметки для представления математических символов и формул в документах HTML5, основанный на XML. В спецификацию HTML5 не входит.

Технология перетаскивания Grad and Drop является частью спецификации HTML5; обеспечивает перетаскивание практически любого элемента на веб-странице с возможностью дополнительной обработки перетаскиваемого элемента с помощью специальных событий перетаскивания.

Технология геолокации (GeoLocation) предназначена для определения местоположения пользователя; GeoLocation API не входит в спецификацию HTML5.

Технология Web Storage используется для локального хранения данных. Различают локальное хранилище и сессионное хранилище (для каждого открытого окна создается новая сессия, которая прекращает своё существование при закрытии окна).

Технология Web Workers реализует фоновые вычисления вне основного потока браузера, что снижает «притормаживание» веб-приложений.

Документ HTML5 состоит из элементов. Элементы могут быть пустыми (они так и называются), содержать текст, содержать другие элементы (дочерние элементы), либо текст и элементы.

Непустой элемент как структура состоит из открывающего тега, опциональных (иногда и обязательных) атрибутов. Как правило, элемент содержит контент, а в конце его стоит закрывающий тег. Атрибуты могут быть глобальные (они применимы к любому HTML5 элементу), специфические (применяемые к отдельным элементам) и пользовательские (в HTML5 разработчик сам может определить любой атрибут, предваряя его префиксом *data-)*. Как правило, атрибуты имеют значения.

Простой HTML5-документ выглядит так:

```
<!DOCTYPE html>
<html>
 <head>
   <title> заголовок страницы </title>
    <meta charset=”utf-8”>
 </head>
 <body>
     Содержание документа
  </body>
</html>
```

Элемент `<!DOCTYPE html>` указывает тип документа (DTD: document type definition, описание типа документа). В HTML5 он легко запоминаем, и выглядит проще, чем в других версиях. Например, в HTML 4.1 он выглядит так:

```
     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

В контейнере <html>…</html> содержится текст документа, который состоит из заголовка, вложенного в теги <head>…</head>, и тела, выделенного тегами <body>…</ body >. Тело включает предназначенный для отображения текст и разметку документа. Кроме обычного текста тело документа может содержать десятки встроенных тегов.

Элемент-заголовок <head>…</head> содержит информацию о документе (его название, метаданные, списки стилей и JavaScript) и может содержать более десятка тегов. В данном простом документе содержатся: в тегах <title > ... </title> заголовок страницы и тег <meta**>** кодировки страницы, который записывается проще, чем в HTML 4.1, где он выглядит так:

<`meta http-equiv="content-type" content="text/html; charset=UTF-8">.`

Далее, используя теги и атрибуты, таблицы стилей CSS3 и технологию JavaScript можно создавать, как отмечают авторы книг по HTML5, сногсшибательные сайты и приложения для любых настольных и мобильных платформ.

Каскадные таблицы стилей CSS (Cascading Style Sheets) используются для задания цветов, шрифтов, расположения и других аспектов представления веб-документа. Они обеспечивают отделение содержимого документа от его оформления. Значительная часть элементов признанных в HTML5 устаревшим относятся к разметке, задачи которой в HTML5 возложены на CSS3.

Таблицы стилей строятся согласно определенным правилам. Общий синтаксис описания правила выглядит так:

селектор {свойство: значение;}

Селектор — это элемент, к которому применяются стили. Элементом может быть тег, класс или идентификатор объекта гипертекстового документа. Свойство определяет одну или несколько характеристик селектора. Свойства задают формат отображения селектора: отступы, шрифты, выравнивание, размеры и т.д. Значения — это фактические числовые или строковые константы, определяющие свойство селектора.

На рисунке 5.2 приведен пример изменения цвета текста и выравнивания текста параграфа, взятый из справочника по CSS Мержевича В. В. (http://htmlbook.ru/metki/spravochnik).

Пример HTML5 CSS 2.1 IE Cr Op Sa Fx

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Селекторы тегов</title>
  <style>
   p {
    text-align: justify; /* Выравнивание по ширине */
    color: green; /* Зелёный цвет текста */
   }
  </style>
 </head>
 <body>
  <p>Более эффективным способом ловли льва в пустыне
  является метод золотого сечения. При его использовании пустыня делится
  на две неравные части, размер которых подчиняется правилу золотого
  сечения.</p>
 </body>
</html>
```

В данном примере изменяется цвет текста и выравнивание текста параграфа. Стиль будет применяться только к тексту, который располагается внутри контейнера <p> (рис. 1).

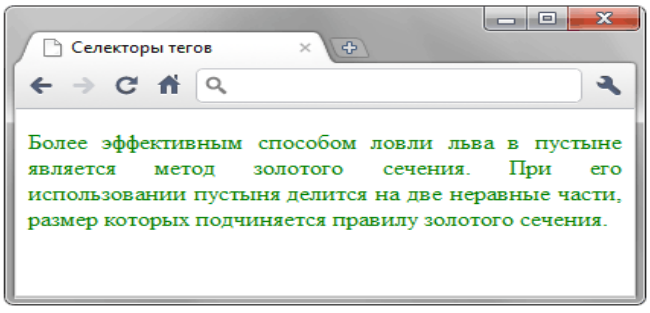


Рис. 1. Применение стиля к селектору p

Рисунок. 5.2. Применение стиля к тегу <p>.

Это один из способов внедрения правил CSS в документ: путем включения тегов <style>…</style> в секцию <head>. Существуют и другие способы внедрения CSS. Например, можно сделать так

```
<body>
Кафедра <span style='font-style:italic'>ИТАС</span>
<body>
```

Здесь использовано строчное правило CSS (свойство style элемента), в результате слово ИТАС будет напечатано курсивом.

Следующий способ, наиболее популярный. Все правила CSS сохраняются в отдельном документе, который затем посредством тега <link>, подключается одной строкой внутри тега <head>.

Наконец, с использованием атрибута класс, CSS внедряется так:

```
<!DOCTYPE html>
<html>
 <head>
   <title> Атрибут класс </title>
    <meta charset="utf-8">
   <style>
   .italic { font-style:italic; }
   </style>
 </head>
 <body>
   Кафедра <span class='italic'>ИТАС</span>
  </body>
</html>
```

В данном случае результат тот же, что и при использовании строчного правила CSS, но этот вариант предпочтителен, поскольку осуществлено отделение стиля от документа.

Спецификация CSS3, которая является будущим в оформлении веб-страниц, не является составной частью стандарта HTML5; поэтому при изучении HTML5-технологий необходимо пользоваться как справочником по HTML, так и справочником по CSS (http://htmlbook.ru/metki/spravochnik).

Третьим важнейшим компонентом HTML5 в широком смысле является язык JavaScript, который изначально разрабатывался специально для «оживления» веб-страниц. Язык дважды менял название, он специфицирован как ECMAScript (версия ECMAScript5), так как имя JavaScript является зарегистрированным товарным знаком компании Oracle Corporation.

Программу на языке JavaScript с помощью секции <script>...</script> можно встраивать в любое место документа HTML5, например, так

```
  <!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <script>
    alert('Нажмите «ОК», если готовы.');
  </script>
  </body>
```

```
</html>
```

Здесь глобальный метод alert выводит на экран окно с сообщением и приостанавливает выполнение скрипта до нажатия кнопки «ОК».

Так встраиваются только самые простые скрипты; сложные же выносят в отдельный файл, который подключается в HTML с использованием полного URL или абсолютного пути к файлу.

Язык JavaScript может выполняться не только в браузере, но и везде, где имеется его интерпретатор. Однако браузер, в целях безопасности, накладывает на исполнение кода JavaScript ограничения, которые отсутствуют при выполнении кода вне браузера, например на сервере. В браузере же JavaScript позволяет: менять стили элементов, прятать и показывать элементы; реагировать на действия пользователя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру; выполнять простые вычисления на веб-странице, посылать запросы на сервер и загружать данные без перезагрузки страницы и многое другое.

5.3. Технологии XML

5.3.1. Ключевые стандарты Web-сервисов

Технологии XML играют важную роль в современном подходе к построению корпоративных систем. Применение языка XML позволяет сократить технологические издержки на интеграцию, поскольку построенные по модульному принципу XML-ориентированные Web-сервисы представляют собой платформо и программно независимые, многократно используемые элементы, которые имеют идентификаторы URI, и общедоступные интерфейсы на языке XML.

Ключевыми стандартами Web-сервисов являются:

XML (eXtensible Markup Language) — расширяемый язык разметки;

SOAP (Simple Object Access Protocol) — протокол обмена сообщениями между потребителем и поставщиком веб-сервиса;

UDDI (Universal Description, Discovery and Integration) —универсальный интерфейс распознавания, описания и интеграции, используемый для формирования каталога веб-сервисов и доступа к нему;

WSDL (Web Services Description Language) язык описания Web – сервисов.

Протокол SOAP определяет формат запросов к web –сервисам. Сообщение (запрос или ответ) между пользователем и web –сервисом пакуется в SOAP-конверт. Конверт и его содержимое закодированы на языке XML. SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTPS; обычно — поверх HTTP.

UDDI является сетевым реестром (службой каталогов), представляющим данные и метаданные о веб-сервисах; доступен по адресу http://uddi.xml.org/services. UDDI опирается на стандарты HTTP, XML, XML Schema, SOAP и WSDL. Функциональное назначение реестра UDDI — представление данных и метаданных о веб-службах. Он может использоваться как в сети общего пользования, так и в пределах внутренней инфраструктуры организации.

Язык описания веб-сервисов WSDL (WSDL 2.0) предназначен для унифицированного представления внешних интерфейсов веб-службы. Посредством языка WSDL (основанного на XML) веб-сервис описывает в WSDL-документе синтаксис работы с данным сервисом.

Приведенные выше стандарты относятся к первому поколению Web-служб; современные Web-службы располагают обширным набором дополнительных стандартов и протоколов, связанных с безопасностью, гарантированной передачей сообщений и т. д.

Технологии XML охватывает огромную, сферу включающую множество стандартов и технологий.

С XML связан стандарт метаданных RDF (Resource Description Framework – спецификация описания ресурсов). Стандарт RDF поддерживает теги, позволяющие определять любые понятия (формально теги XML оторваны от определения их смыслового наполнения). При их использовании отпадает необходимость анализировать все остальное содержание Web-страницы в поисках нужной информации. Данным в формате RDF присваиваются дескрипторы, которые могут определяться в отдельных файлах определения типов документов Document Type Definitions (DTD). RDF наряду с XML является одной из ключевых технологий нового поколения World Wide Web, названного Тимом Бернерсом-Ли «семантической паутиной» (Semantic Web).

RDF – это оболочка описания ресурсов, хранящихся на Web-сайте, предназначенная для обмена машиночитаемой информацией между Web-системами. Стандарт RDF ориентирован в основном для автоматизированных систем поиска информации. Находя RDF-файл на целевом ресурсе, “поисковик” извлекает информацию о нем из предопределенных полей этого метафайла (“заголовок”, “автор”, “описание”, “издательство” и т.п.).

На основе XML и RDF разработан формат RSS, предназначенный для обмена контентом между сайтами (организации информационной коммуникации между серверами). RSS — это формат данных и технический стандарт для интегрированного доступа к новостной информации сайтов (любой материал, который делится на части, можно публиковать с помощью RSS). После преобразования сайтовой информации в формат RSS (существует несколько различных форматов RSS), программа, поддерживающая этот формат (агрегатор), может загружать и отображать новостную информацию Web-сайтов.

5.3.2. Базовые технологии XML

5.3.2.1. Истоки языка XML

Язык XML (Extensible Markup Language) разработан рабочей группой W3C (World Wide Web Consortium). Ныне действующей является Спецификация XML 1.0 (Второе издание) (XML 1.0 (Second Edition)). В основу XML положен язык SGML (Standardized Generalized Markup Language; ISO 8879, 1986 г.). Сам же SGML построен на основе GML, разработанном IBM в 1960 г., который «пришел» из издательского дела (маркировка рукописных текстов для наборщиков).

Язык SGML является сложным (содержит более 500 страниц трудночитаемых спецификаций) и ориентированным на допотопную технику. Язык SGML находит применение в издательском деле и других областях, где требуются большие объемы структурированного текста. Используют SGML-технологию крупные компании, которые могут позволить себе купить дорогостоящее программное обеспечение и нанять высокооплачиваемых специалистов.

Язык XML является простым подмножеством SGML, в котором используются лишь реально необходимые в Internet возможности SGML (по объему примерно в 10 раз меньше спецификации SGML). Язык XML в отличие от HTML, который является приложением, а не подмножеством SGML, позволяет определять структурные элементы документа (собственные теги и атрибуты), а также создавать сложные документы. XML представляет собой метаязык, то есть язык, на базе которого можно создавать новые языки. Он ориентирован не только для создания средств, служащих для организации обмена данными в Web, но и для распознавания семантики этих данных. Сам по себе XML не содержит никаких тэгов для разметки, он просто определяет порядок их создания.

XML-документ представляет собой обычный текстовый файл, содержащий данные, последовательность и вложенность которых определяет структуру документа и его содержание. Основным достоинством XML документов является то, что при относительно простом способе создания и обработки (обычный текст может редактироваться любым тестовым процессором и обрабатываться стандартными XML анализаторами), они позволяют создавать структурированную информацию, которую хорошо "понимают" компьютеры .

Главное же принципиальное отличие XML от HTML состоит в том, что теги XML включают содержание документа (информацию), а не то как документ долен быть отформатирован, т.е. выглядеть на экране.

5.3.2.2. Спецификации технологии XML

Технология XML включает набор следующих основных спецификаций:

XML (Extensible Markup Language) спецификация, определяющая базовый синтаксис XML.

XSL (Extensible Style Language) — спецификация, визуального оформления страницы посредством применения к документу стилей (style sheets), определяющих конкретные атрибуты форматирования;

XLL (Extensible Linking Language**)** — спецификация, определяющая представление ссылок на другие ресурсы.

Пространства имен XML 1.0 (Namespaces in XML 1.0) — Рекомендация консорциума W3C. Описывается механизм универсального обозначения имен элементов и атрибутов в XML-документах. Для исключения совпадений имен каждому словарю с использованием специального синтаксиса назначается свой маркер.

XML Base. Спецификация XML Base определяет способ ассоциирования XML-элементов с унифицированными идентификаторами ресурса.

XML Infoset (XML Information Set). Рекомендация консорциума W3C. Определяется абстрактный способ описания XML-документа в виде набора объектов, называемых **информационными единицами (**information items**),** обладающих специальными свойствами.

XPath XML Path Language (XPath) 1.0 Рекомендация консорциума W3C. Определяются синтаксис и модель данных для адресации частей XML-документа; является основой XSLT **.**

5.3.2.3. Спецификация XML

Основным исходным документом для изучения XML является спецификация XML; при этом официальной нормативной версией считается только версия на английском языке. Найти документацию по XML можно по адресу http://www.w3.org/; последней является пятая редакция XML 1.0. Спецификация XML 1.0 содержит 6 разделов: Введение. Документы. Логические структуры. Физические структуры. Соответствие. Условные обозначения.

Спецификация XML является основой для построения языков разметки; сам же XML лишь условно можно отнести к таким языкам. Существует много языков разметки, являющихся подмножествами XML. К таким языкам можно отнести: язык беспроводной разметки WML (Wireless Markup Language); язык описания химических соединений CML (Chemical Markup Language); язык описания и обработка математических формул MathML (Mathematical Markup Language) и др.

Кроме XML-приложений для описания определенных классов документов имеются XML-приложения, которые применяются для XML-документов любого типа, облегчая создание документа и улучшая его качество. К таким приложениям относятся.

XML Schema. Позволяет разрабатывать схемы для XML-документов с использованием стандартного синтаксиса XML; является более мощной альтернативой DTD.

XML Linking Language (XLink). Позволяет связывать XML-документы; поддерживает множественные целевые ссылки и другие полезные функции.

XML Pointer Language (XPointer). Позволяет определять гибкие целевые ссылки и при совместном использовании с XLink организовывать ссылки на любое место в целевом документе.

5.3.2.4. Основные понятия XML. Правила создания XML-документов

Основным понятием XML является **XML-документ**, которым становится любой объект данных, соответствующий требования корректности **XML**; такой XML-документ называется **корректным** (well-formed). Корректный XML-

документ является также и **действительным** (валидным), если он удовлетворяет дополнительным требованиям.

Любой XML-документ имеет **логическую** и **физическую** структуру. Физически документ состоит из элементов. Логическая структура XML-документа формируется из **деклараций**, **элементов**, **комментариев**, **ссылок на символы** и **инструкций обработки.**

Простой XML-документ состоит из двух частей: Объявления и корневого элемента. Объявление содержит указание на то, что это XML-документ и номер версии XML. Пролог может также содержать необязательные компоненты.

XML-документ может содержать комментарии, начинающиеся с символов " <!--" и заканчивающиеся символами "-->". Комментарий может содержать любой текст, за исключением символов "--".

XML-документ создается в любом текстовом редакторе, сохраняется как текстовый файл с расширением .xml. В дальнейшем такой документ будет открываться двойным щелчком в Internet Explorer.

Документы XML должны удовлетворять следующим требованиям:

1. В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация. Например, `<?xml version=”1.0”>`

2. В отличие от HTML каждый открывающий тэг, определяющий некоторую область данных, обязательно должен иметь закрывающий тег.

3. Необходимо учитывать чувствительность элементов XML к регистру.

4. Атрибуты, используемые в определении тэгов, должны иметь значения в кавычках.

5. Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов.

6. Данные, располагающиеся между начальным и конечным тэгами, рассматривается в XML как данные, поэтому учитываются все символы форматирования.

7. XML содержит зарезервированные символы (элементы синтаксиса). В тексте эти символы заменяются *сущностями* (*entties):*

&lt;	< - меньше, чем
&gt;	> - больше, чем
&amp;	& - амперсанд
&apos;	` - апостроф
&quot;	" - двойная кавычка

5.3.2.5. Синтаксис XML

Любой XML-документ может состоять: из **пролога, тела документа** и **эпилога**.

Пролог может включать **объявление XML, комментарии, команды обработки.** В версии 1.0 объявление XML может быть опущено, в версии 1.1

оно обязательно. Объявление (XML Declaration) пролога заключенное символами <?...?> содержит:

Метку xml и номер версии (version) спецификации XML.

Указание на кодировку символов (по умолчанию *encoding="UTF-8"*) .

Параметр *standalone,* который может принимать значения *"yes"* или *"no"* (по умолчанию *standalone="yes").*Значение "yes" указывает на то, что в документе содержатся все требуемые декларации, а если "no", то нужны внешние определения DTD.

Пример объявления:

```
<?xml version ="1.0" encoding-"windows-1251" standalone="yes"?>.
```

В объявлении XML обязательным является только атрибут *version*, остальные атрибуты можно опустить, но если они приводятся, то необходимо соблюдать указанный порядок.

В объявлении XML может содержаться необязательное объявление типа документа DTD (Document Type Declaration), которое заключено между символами <!DOCTYPE...> и может занимать несколько строк. В DTD объявляются теги, используемые в документе, или дается ссылка на файл, в котором записаны объявления.

После объявление типа документа в прологе могут следовать: комментарии, команды обработки и символы пустых пространств.

Тело документа

Элементы XML-документе определяют его логическую структуру и несут в себе информацию, содержащуюся в этом документе. Типичный элемент состоит из начального тега, содержимого элемента и конечного тега. Содержимым элемента могут быть:

вложенные элементы;

символьные данные (текст, выражающий информационное содержание элемента);

ссылки на общие примитивы и ссылки на символы,

разделы CDATA (текстовый блок, в котором можно свободно размещать любые символы, за исключением комбинации]]>);

инструкции по обработке (содержат информацию, необходимую для XML-приложений);

комментарии (они игнорируются процессором);

пустые элементы.

XML-документ, если он содержит более одного элемента, должен представлять собой иерархическое дерево, содержащее корневой элемент, в который вкладываются остальные элементы.

Имя корневого элемента, которое является и именем всего XML-документа, указывается во второй части пролога после слова Doctype. Имена элементов XML-документа в пределах документа должны быть уникальны.

Любой элемент начинается открывающим тегом, после которого следует необязательное содержимое элемента, затем обязательный закрывающий

тег. Допускается записывать элементы без содержания (пустой элемент) в следующей сокращенной форме:

<имя_элемента/>.

Включение в текст XML-документа зарезервированных символов или таких символов, которые отсутствуют в раскладке клавиатуры, осуществляется путем вставки знака амперсанд «**&**», за которым следует символ «#», код вставляемого символа в *Unicod (или в шестнадцатеричном коде с символ "x" в начале); завершается такая вставка точкой с запятой:*

&# код_символа_в_Unicode;.

&#xШестнадцатеричный_код_символа;.

Ссылки на сущности, которые позволяют включать любые строковые константы в содержание элементов или значение атрибутов начинаются с амперсанда, после которого следует имя сущности и в завершение точкой с запятой:

&имя_сущности;.

Ссылка на сущность указывают программе-анализатору на необходимость подстановки заранее заданных строки и символов. заранее заданную в определении типа документа.

Секция CDATA (может находиться везде, где размещаются символьные данные) используется для задания области документа, которую анализатор должен передать без изменений; так что секцию CDATA анализатор не игнорирует как комментарий, а передает далее без изменений.

Секция CDATA

Секция CDATA формируется следующим образом:

<![CDATA[содержание секции]]>.

Секция CDATA начинается комбинацией "<![CDATA", содержит внутренние квадратные скобки с содержанием секции, и заканчивается символами "]>":.

Инструкции по обработке

Инструкции по обработке заключаются между символами <? … ?> и содержат указания программе анализатору документа XML. Вначале (за вопросительным знаком) записывается имя программы, которой предназначена инструкция. Затем, через пробел, приводится инструкция, передаваемая программному модулю. Инструкция не должна содержать набор символов "?>", означающий конец инструкции. Например, инструкция

<?xml-stylesheet type="text/css" href=" Inventory01.css"?>

предписывает Internet Explorer использовать CSS-таблицу из файла Inventory01.css.

Атрибуты

Открывающие теги или теги пустых элементов в XML могут содержать атрибуты, в виде пары: *имя=значение*. В одном открывающем теге разрешается использовать только один экземпляр имени атрибута. Атрибуты могут содержать ссылки на объекты, ссылки на символы и текстовые символы. В XML значения атрибутов обязательно заключаются апострофами (') или кавычками (") :

```
имя_атрибут='значение_атрибута',
имя_атрибута="значение_атрибута".
```

Атрибуты в XML можно определять самостоятельно.

В рекомендациях XML 1.0 определены два специальных атрибута: `xml:space` и `xml:lang`.

Атрибут `xml:space` указывает приложению на необходимость сохранять пустые пространства, значения атрибута применяется не только к содержащему элементу, но и ко всем его потомкам. Атрибут `xml:space` может принимать значения `"preserve"` и `"default"`. Значение `"preserve"` предписывает сохранять пробельные символы без изменения, `"default"` — оставлять пробельные символы на усмотрение программы-обработчика.

Атрибут `xml:lang` используется для идентификации языка записи содержимого и значений атрибутов любого элемента.

Атрибут *xml:lang* необходимо продекларировать в DTD. Данный атрибут может принимать значения определенные в документе RFC 1766 или наследующих его стандартах. Значение атрибута применяется не только к содержащему его элементу, но и ко всем его потомкам и атрибутам.

Комментарии

Комментарий оформляют следующим образом:

<!-- …текст комментария… >

Комментарий позволяет сделать фрагмент документа "невидимым" для программы-анализатора.

Пустые пространства

В языке XML пустым пространством считаются четыре символа: горизонтальная табуляция (HT), перевод строки (LF), возврат каретки (CR), символ пробела ASCII. Спецификация XML требует, чтобы анализатор передавал все символы, в том числе и символы пустых пространств в содержании элемента, приложению. Символы же пустых пространств в тегах элементов и значениях атрибутов могут быть удалены.

Эпилог XML

Эпилог XML может включать **комментарии, инструкции по обработке и/ или пустое пространство**.

В языке XML индикатор конца документа не определен. Большинство приложений для этой цели используют завершающий тег корневого элемента документа.

Пример простого документа XML

Проиллюстрируем особенности синтаксиса XML на простом примере, в котором описаны три книги из нашего курса по технологии SharePoint. Ниже приведен листинг примера *Books.xml.* На рисунке. 5.3. показан вид *Books.xml в* MS Internet Explorer.

Листинг 5.1. Пример *Books.xml*

```
<?xml version="1.0"?>
    <root>
       <books>
       <book pages="832">
           <title>Microsoft SharePoint 2007</title>
           <author>Michael Noel</author>
       </book>
       <book pages="448">
           <title>Windows SharePoint Services 3.0</title>
           <author>Ted Pattison</author>
        </book>
        <book   pages="554">
            <title>Microsoft SharePoint Designer 2007</title>
            <author>Penelope Coventry</author>
        </book>
      </books>
  </root>
```

Документ XML не нарушающий правила 1-7 создания XML-документов называется **правильным**; в противном случае он не может считаться документом XML (парсер такой документ классифицирует как фатальную ошибку).

Рисунок. 5.3. Файл *Books.xml* в окне MS Internet Explorer.

Внесем изменение в листинг *Books.xml,* нарушающее правило 2. На рисунке. 5.4. прокомментирована фатальная ошибка в фале *Books.xml.*

Листинг 5.2. Изменение в листинге 1, нарушающее правило 2

```
<?xml version="1.0"?>
   <root>
      <books>
      <book pages="832">
          <title>Microsoft SharePoint 2007</title>
          <author>Michael Noel</author>
      </book>
      <book pages="448">
          <title>Windows SharePoint Services 3.0</title>
          <author>Ted Pattison</author>
       </book>
       <book   pages="554">
           <title>Microsoft SharePoint Designer 2007</title>
           <author>Penelope Coventry</author>
       </books>
  </root>
```

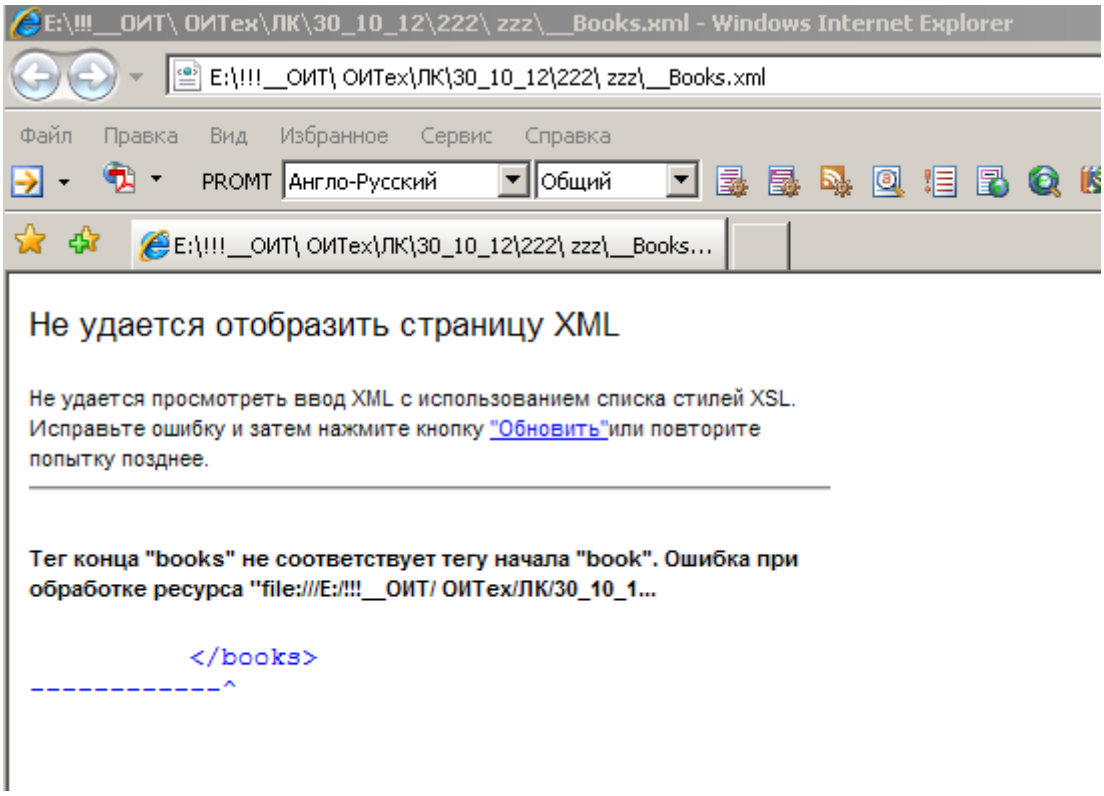


Рис. 5.4. Фатальная ошибка в фале *Books.xml*

5.3.2.6. Пространство имен XML

Спецификация Namespaces in XML 1.0 (третья редакция, 2009 г.) предназначена для разрешения проблем, связанных с конфликтами имен. Проблема разрешения имен (в отличие от HTML, где теги фиксированы) возникает в связи с тем, что каждый разработчик может создавать собственные теги. При подготовке сложных XML-документов, например, использующих несколько DTD, некоторые теги, разные по своему назначению, могут получить одинаковые имена.

В этой связи были разработаны пространства имен XML. Для различения схем документов схеме документа ставится в соответствие специальный уникальный идентификатор ресурса URI. В результате схемы будут считаться

тождественными только в том случае, если их URI будут совпадать. По этому, в качестве идентификатора чаще всего используется адрес своего ресурса.

Пространство имен обычно формируются согласно спецификации URL (URL — это такой вид URI, который кроме идентификации ресурса, указывает ещё и местонахождение этого ресурса), которые должны обеспечить уникальность пространства имен.

Пространства имён объявляются с помощью XML атрибута `xmlns`, значение которого должно быть ссылкой URI. Поскольку используется лишь алгоритм образования адреса, то нет необходимости в выборе адреса реально существующего Web-сайта, и поэтому адрес можно составить совершенно произвольно, например:

```
xmlns=http://www.w3.org/1999/xhtml.
```

В данном случае XML парсер обрабатывает URL как простую строку, а не как адрес в сети. По адресу http://www.w3.org/1999/xhtml находится просто справочник по пространству имён xhtml.

С одинаковым успехом адрес можно представить в и виде *xmlns=http://www.xur.com/vanja/,* и это будет работать. Главное в данном случае уникальность, именно поэтому большинство пространств имен выглядят как URL.

5.3.2.7. Объявления типа документа. Валидные документы

Действительным (валидным; *valid)* называется такой правильный XML-документ, который содержит необходимые объявления типа документа (Document Type Definitions; DTD). При этом, данный документ должен соответствовать ограничениям, налагаемым данным объявлением. Каждый действительный документ должен начинаться с информационного заголовка содержащего:

Описание правил структурирования, которым должен соответствовать данный документ.

Список внешних ресурсов или сущностей, составляющих какую-либо часть документа.

Объявления внутренних ресурсов или внутренних сущностей.

Условные обозначения или же ресурсы, проставляемые в форматах, отличных от XML, которые должны определять требуемые вспомогательные приложения.

Списки ресурсов, представляемых в форматах отличных от XML, имеющихся в документе.

Объявления типа документа (DTD) может быть внешним или внутренним или тем и другим. В последнем случае во внутренних DTD могут быть заданы новые декларации или переписанные из внешних (анализаторы сначала читают внутренние DTD).

Для связывания декларации DTD с экземпляром документа в версии XML 1.0 предлагается специальная декларация `DOCTYPE`.

Декларация DOCTYPE содержит ключевое слово DOCTYPE, за которым следует имя корневого элемента документа, а затем конструкция с декларациями содержания.

Можно написать внешнее подмножество деклараций в отдельном файле DTD, включить внутреннее подмножество в тело декларации DOCTYPE или сделать то и другое. В последнем случае во внутренних DTD могут быть заданы новые декларации или переписаны те, что содержатся во внешних (по определению спецификации XML анализаторы сначала читают внутреннее подмножество, и потому содержащиеся там декларации пользуются приоритетом).

Блок внутренней декларации разметки тега DOCTYPE состоит из левой квадратной скобки, списка деклараций и правой квадратной скобки:

```
<! DOCTYPE root_element_name
[… декларации …]
```

Для внешних DTD декларация DOCTYPE состоит из обычного и ключевого слова и имени корневого элемента. Далее следует ключевое слово SYSTEM или PUBLIC, обозначающее источник внешнего определения DTD.

Допустимое в XML-документе содержание определяется четырьмя приведенными в нижеследующей таблице типами декларации разметки в DTD.

Конструкция DTD	Значение
ELEMENT	Декларация типа элемента XML
ATTLIST	Декларация, назначаемых конкретным элементам, а также разрешенных значений этих атрибутов
ENTITI	Декларация повторно используемого содержания
NOTATION	Декларация форматирования для внешнего содержания не предназначенная для анализа

Первые два типа декларации связаны с элементами и атрибутами содержания документа. Последние два типа ориентированы на поддержку технологии; они включают содержание, которое часто используется в DTD. Нотации описывают содержание, разработанное не на языке XML.

В документе XML должен быть описан каждый элемент. Объявление элемента начинается с символов <!ELEMENT, далее через пробел записывается имя элемента и его содержание. Заканчивается объявление символом ">".

По содержанию элементы делятся на четыре группы:

1. Пустой элемент. Может иметь атрибуты, но не содержит текст или порожденные элементы, например,

```
<! ELEMENT element_name EMPTY>.
```

Здесь после имени элемента необходимо указать ключевое слово EMPTY.

2. **Элемент содержит только порожденные элементы.** После имени элемента в скобках через запятую перечисляются все вложенные элементы, например,

```
<ELEMENT element_name (elem_1,elem_2)>.
```

Примечание. Вложенные элементы в XML документе должны следовать в том же порядке, что и в объявлении.

3. **Элемент содержит не только порожденные элементы, но и текст.** В данном случае после имени элемента в скобках указывается ключевое слово #PCDATA, после которого через запятую перечисляются все (если они имеются) вложенные элементы:

```
<ELEMENT element_name (#PCDATA, elem_1, elem_2)>
```

4. **Элемент, открытый для любого содержания.** В этом случае после имени элемента указывается ключевое слово ANY**:**

```
<ELEMENT element_name ANY>.
```

В случае, если из нескольких вложенных элементов (или списков) может встретиться только один, то в их перечислении применяется вертикальная черта (|). Так, например в объявлении:

```
<!ELEMENT element_name (elem_1,(elem_2|elem_3))>
```

элемент *element_name* должен содержать элемент *elem_1*, а затем либо elem_2, либо *elem_3*. Элементы должны появляются именно в таком порядке.

Если вложенный элемент можно записать в объявляемом элементе несколько раз, то необходимо использовать знаки, приведенные в нижеследующей таблице

Символ	Функция в объявленном элементе
?	Элемент или список может встретиться нуль или один раз
*****	Элемент или список может встретиться нуль или несколько раз
+	Элемент или список может встретиться один или несколько раз

Например:

```
<!ELEMENT productList (computer)+ >
<!ELEMENT computer (hdd?, processor*, morherboard) >
<!ELEMENT hdd (#PCDATA) >
<!ELEMENT processor (#PCDATA) >
<!ELEMENT motherboard (#PCDATA) >
```

Объявления атрибутов элемента. За объявлением элемента следует объявления его атрибутов. Все атрибуты одного элемента объявляются сразу же, одним списком. Список начинается с ключевого слова <!ATTLIST, после которого через пробел следует имя элемента, затем имя атрибута, его тип или

список значений, которые он может принимать (значения перечисляются через вертикальную черту, в скобках), признак обязательности присутствия атрибута в элементе или значение по умолчанию.

Тип атрибута записывается одним из ключевых слов, приведенных в нижеследующей таблице:

Ключевое слово	Тип атрибута
CDATA	Строка символов
ID	Уникальный идентификатор, однозначно определяющий элемент, в котором встретился этот атрибут
IDREF	Идентификатор, содержащий одно из значений атрибутов типа ID, используется в качестве ссылки на другие элементы
IDREFS	Идентификатор, содержащий набор значений атрибутов типа ID, перечисленных через пробелы; используется в качестве ссылки сразу на несколько элементов
ENTITY	Имя непроверяемой анализатором сущности, объявленной в этом же описании DTD
ENTITIES	Имена непроверяемых сущностей
NMTOKEN	Слово, содержащее только символы, применяемые в именах
NMTOKENS	Слова, перечисленные через пробелы
NOTATION	Обозначение, расшифрованное в описании DTD

Признак обязательности записывается с использование ключевых слов**:**

`#REQUIRED` (запись атрибута в элементе обязательна);

`#IMPLIED` (атрибут необязателен, значения по умолчанию не имеет);

`#FIXED` (атрибут имеет только одно значение, которое записывается здесь же, через пробел).

Рассмотрим пример DTD, содержащееся непосредственно в самом XML-документе:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE DOC [
<!ELEMENT DOC (SUBJECT, DATE, ADDRESS, MEMO)>
<!ELEMENT SUBJECT (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
```

```
<!ELEMENT ADDRESS (#PCDATA)>
<!ELEMENT MEMO (#PCDATA)>
<!ENTITY PUBLISHER "The Coriolis Group">
]>
<DOC>
<SUBJECT>Today's Memo</SUBJECT>
<DATE>August 1, 2000</DATE>
<ADDRESS>200 West 34th Suite 953, Anchorage, AK</ADDRESS>
<MEMO>This memo is to alert you to the new XML
Black Book has now been printed. Published by
PUBLISHER;, this book outlines everything you
need to know about XML.</MEMO>
</DOC>
```

Прокомментируем основные компоненты этого XML-документа:

`<?XML version="1.0" encoding="UTF-8" standalone="no"?>` — это объявление указывает на то, что данный документ является XML-документом версии 1.0; используется 8-битная кодировка Unicode. Далее, объявление отдельного документа (`standalone document declaration`) “`no`" указывает процессору, что он может обрабатывать любые внешние объявления (если они имеются). По умолчанию `standalone` принимает значение "`yes`", означающее, что в данном документе не используется внешнее DTD или же какие-либо внешние параметрические сущности.

`<!DOCTYPE DOC [` — указывает, где размещено DTD (в данном случае в самом документе).

`<!ELEMENT DOC (SUBJECT, DATE, ADDRESS, MEMO) >` — Определяет список элементов для корневого элемента DOC; сообщает анализатору о том, что корневой элемент DOC содержит элементы `SUBJECT, DATE, ADDRESS и MEMO`, которые должны появляться в документе в этом же порядке, иначе анализатор выдаст сообщение об ошибке.

`<!ELEMENT SUBJECT (# PCDATA)>; <!ELEMENT DATE (#PCDATA)>; <!ELEMENT ADDRESS (#PCDATA)>; <!ELEMENT MEMO (#PCDATA)>` —определяет элементы `SUBJECT, DATE, ADDRESS, MEMO` и указывает, что они будет содержать символьные данные, которые подлежат обработке анализатором.

`<!ENTITY PUBLISHER "The Coriolis Group">` — определяет простую сущность `PUBLISHER` и указывает, что значением этой сущности является "`The Coriolis Group`".

`]>` – указывает на конец DTD.

Объявлением типа документа `<!DOCTYPE DOC` служит для связи XML-документа с DTD; выражение, следующее за `<!DOCTYPE`, является именем DTD. В случае внутреннего DTD за объявлением `<!DOCTYPE` следует список элементов и атрибутов. В объявлении типа документа XML-документа указывает, является ли данное DTD общим (public), или же частным (private). Далее, после закрывающей скобки указывается либо само DTD, либо локатор ссылки (reference locator) на его местоположение.

Внешние DTD также указываются в объявлении типа документа, но в отличие от внутренних DTD в данном случае указывается лишь внешний файл,

в котором хранится сам DTD. При этом одно DTD может использоваться в нескольких документах. В этом заключается одна из наиболее мощных функциональных возможностей XML, позволяющих создавать стандартизованные способы представления информации.

В случае внешнее объявление DTD в объявлении размещается ключевое слово `SYSTEM`, за которым следует адрес и имя файла, в котором хранится DTD:

```
<!DOCTYPE book SYSTEM "http://www.site.com/dtds/book.dtd">
```

В XML, в отличие от SGML, применение DTD не является обязательным. DTD необходимы для документов большого объема. При помощи внешнего DTD можно стандартизовать данные, что сделает этот документ логически более последовательным, поскольку применение DTD предполагает обязательное следование определенным правилам. Малые документы не требуют использования внешних DTD. Не следует создавать DTD и для простых документов (служебных записок и факсов размером в одну страницу). Применение внешних DTD вызывает дополнительную нагрузку каналов связи.

XML-процессоры, не проверяющие валидность XML-документов, не требуют DTD. Если XML-процессор проверяет только правильность XML-документа, но не валидность, то нет необходимости применять внешнее DTD. Следует отметить, что Internet Explorer проверяет лишь на соответствие документа формальным правилам, но не на валидность.

5.3.2.8. Спецификации XML Schema

В настоящее время в XML-технологиях вместо DTD широко используется язык XSD **(**XML Schema), что вызвано такими недостатками DTD как отсутствие поддержки пространств имён и XML синтаксиса, а также требуемой поддержки типов.

Язык описания XML Schema (XSD) обеспечивает возможность описание типов, на которые можно ссылаться как на экземпляр, что похоже на традиционные объектно-ориентированные отношения между классами и объектами. Эта особенность XSD позволяет эффективно использовать в технологии XML объектно-ориентированные языки программирования. Кроме того, XSD расширяем, и позволяет подключать уже готовые словари для описания типовых задач, например веб-сервисов, таких как SOAP.

Язык XSD **(**XML Schema Definition) специфицирован в трех частях:

Введение **(Пример),** расположенное на w3.org/TR/xmlschema-0, которое предназначено для начального ознакомления с данной технологией.

Стандарт на **структуру документа** (w3.org/TR/xmlschema-1), который иллюстрирует, как определять структуру XML-документов;

Стандарт на **типы данных** (w3.org/TR/xmlschema-2), который определяет некоторые общие типы данных и правила создания новых типов.

Начинать изучение XSD следует со спецификации XML Schema Part 0 (w3.org/TR/xmlschema-0), в которой отмечается, что Документ, "*XML-схема. Часть 0: Пример*", являясь учебником для начинающих, и должен

использоваться наряду с описанием языка, содержащегося в документах "*XML-схема. Часть 1: Структуры"* и "*XML-схема. Часть 2: Типы данных"».*

Обзор наиболее важных усовершенствований в XML Schema 1.1, в сравнении с XML Schema 1.0, приведен в XML Схема 1.1. .

Детальное рассмотрение спецификации XML Schema выходит за рамки данного курса. Ниже рассмотрены лишь отдельные вопросы, технологии XSD.

Наиболее существенные свойства XML-схем:

Поддержка типизации данных, что позволяет описывать разрешенное содержание документа, проверять правильность данных, работать с базами данных, задавать ограничения на данные и форматы данных, преобразовывать данные различных типов.

XML-схем пишутся на языке XML, что позволяет использовать XML-редактор, XML-парсер, работать с XML-схемами используя XML DOM и преобразовывать схемы посредством XSLT.

XML-схемы расширяемы. Это позволяет встраивать схемы в другие схемы, создавать из стандартных типов свои собственные типы данных, ссылаться на несколько схем.

В схемах XML поддерживаются наиболее распространенные в языках программирования типы данных:

```
xs:string
xs:decimal
xs:boolean – true/false
xs:float
xs:date (1950-03-26)
xs:time (15:30:01)
```

и др.

Элементы в схемах XML могут быть как простыми, так и сложными. Простые элементы не могут иметь атрибуты и содержимое смешанного типа. Содержимое смешанного типа, которое могут иметь сложные элементы, позволяет использовать внутри элемента дочерние элементы и текст.

Простые элементы определяются с помощью вложенного элемента

<xs:simpleType>,

а сложные определяются так

<xs:complexType>.

Ограничения на значение элемента устанавливать с помощью элемента <xs:restriction>, например:

```
<xs:element name=”возраст”>
<xs:simpleType>
<xs:restriction base=”xs:integer”> <!-- ограничения:-->
<xs:minInclusive value=”20”> <!—мин. Значение -->
<xs:maxInclusive=”30” ><!—макс. Значение -->
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Ограничения на значения элемента устанавливаются с помощью <xs:restriction> и <xs:enumeration>. Например:

```
<xs:element name="Processor">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="P4"/>
<xs:enumeration value="Intel"/>
<xs:enumeration value="CoreDuo"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Для того, чтобы можно было использовать ограничение вне определения элемента Processor, например, еще и в элементе "Platform", необходимо модифицировать определение элемента Processor:

```
<xs:element name="Processor" type="processorType">
<xs:simpleType name="processorType">
<xs:restriction base="xs:string">
<xs:enumeration value="P4"/>
<xs:enumeration value=" Intel"/>
<xs:enumeration value="CoreDuo"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

В отличие от простых элементов сложные элементы могут иметь атрибуты и содержимое смешанного типа. Последовательность появления дочерних элементов определяется с помощью конструкции

<xs:sequence>.

Например, сложный элемент Computer с установленной последовательностью появления: элементов HDD, Motherboard; определяется так:

```
<xs:complexType name="Computer">
<xs:sequence>
<xs:element name="HDD" type="xs:string"/>
<xs:element name="Motherboard" type="xs:string"/>
</xs:sequence>
<xs:attribute name="model" type="xs:string" fixed="NewModel"/>
<xsd:complexType>
```

Подключить XML документ к его XSD схеме можно разными способами: подать файлы со схемой на вход анализатора; задать

файлы со схемой как свойство анализатора; указать прямо в документе XML. Например, подключить XML-Schema к XML документу можно следующим так:

```
< root_element_name xmlns:xs="http://www.w3.org/2001/XMLSchema-
instance"
xs:noNamespaceSchemaLocation=" file_name.xsd ">
…
</ProductList>
```

Вместо «*file_name.xsd*» может быть любой допустимый URL, а вместо *root_element_name* любой корневой элемент XML документа.

Если элементы документа относятся к определенному пространству имен, то применяется атрибут `schemaLocation`, затем через пробел перечисляются пространство имен и расположение файла со схемой, описывающей это пространство имен. В данном случае записан атрибут `noNamespaceSchemaLocation`, указывающий расположение файла со схемой в форме URI, так как элементы документа не принадлежат пространству имен.

5.3.2.8. Технология XSLT

Язык преобразования XML-документов **XSLT** (eXtensible Stylesheet Language Transformations) входит в состав **XSL** и является рекомендацией W3C. В семейство XSL (eXtensible Stylesheet Language) кроме XSLT входят также **XPath** (язык путей и выражений, используемый в XSLT) и **XSL-FO** (XSL Formatting Objects; язык разметки типографских макетов и предпечатных материалов).

Технология XSLT по существу представляет собой web-шаблонизацию: XML-документ посредством XSLT-преобразования (таблицы стилей) преобразуется в HTML-документ (или в другую XML-схему); такой процесс часто называют *рендерингом*.

Отображение XML-документа с использованием таблицы стилей предполагает:

1. **Создание файла XSL-таблицы стилей**, который является XML-документом и отвечает правилам XSL; XSL-таблица стилей содержит простой текст, который можно создать с помощью текстового редактора.

2. **Связывание XSL-таблицы стилей с XML-документом** посредством включения в XML-документ инструкции по обработке `xml-stylesheet`, которая имеет следующую обобщенную форму:

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

Здесь `XSLFilePath` представляет собой URL местонахождение файла таблицы стилей; можно в принципе использовать и полный URL:

```
<?xml-stylesheet type="text/xsl"
href="http:/www.my_domain.com/Inventory.xsl"?>.
```

Таблица стилей должна размещаться в той же папке что и XML-документ, с которым она связана. Обычно инструкция по обработке `xml-stylesheet` добавляется в пролог XML-документа вслед за объявлением XML.

XSL-таблица стилей включает один или несколько шаблонов, каждый из которых содержит информацию для отображения в определенной ветви элементов в XML-документе.

На листинге 5.3 приведена XSL-таблица стилей для XSL-документа, представленного на листинге 5.2.

Листинг 5.2. XSL-документ

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
```

```
<?xml-stylesheet type="text/xsl" href="1XslDemo01.xsl"?>
<BOOK>
   <TITLE>Microsoft SharePoint 2007</TITLE>
   <AUTHOR>
      <FIRSTNAME>Майкл</FIRSTNAME>
      <LASTNAME>Ноэл</LASTNAME>
   </AUTHOR>
   <BINDING>Твердый переплет </BINDING>
   <PAGES>832</PAGES>
   <PRICE>$20.95</PRICE>
</BOOK>
```

XSL-документ на листинге 5.2, за исключением второй строки, не требует комментариев. Вторая строка этого листинга:

```
<?xml-stylesheet type="text/xsl" href="1XslDemo01.xsl"?>
```

является ссылкой на XSL-файл, приведенный на листинге 5.3.

Листинг 5.3. XSL-таблица XSL-документа

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
   <xsl:template match="/">
      <H2>Описание книги</H2>
      <SPAN STYLE="font-style:italic">Author: </SPAN>
      <xsl:value-of select="BOOK/AUTHOR"/><BR/>
      <SPAN STYLE="font-style:italic">Title: </SPAN>
      <xsl:value-of select="BOOK/TITLE"/><BR/>
      <SPAN STYLE="font-style:italic">Price: </SPAN>
      <xsl:value-of select="BOOK/PRICE"/><BR/>
      <SPAN STYLE="font-style:italic">Binding type: </SPAN>
      <xsl:value-of select="BOOK/BINDING"/><BR/>
    <SPAN STYLE="font-style:italic">Number of pages: </SPAN>
      <xsl:value-of select="BOOK/PAGES"/>
   </xsl:template>
</xsl:stylesheet>
```

Вторая строка файла содержит тег элемента `xsl:stylesheet`. Атрибутами этого элемента являются номер версии и ссылка на пространство имен. Атрибуты элемента `xsl:stylesheet` являются обязательными. Для XSL-файлов ссылка на пространство имен является стандартной. Элемент `xsl:template match`. вызывается всякий раз, когда имя `xml-узла` совпадает со значением атрибута xsl:template match. Атрибут `match` аналогичен селектору в правиле CSS.

Приведенная выше XSL-таблица содержит два вида XML-элементов:

XML-элементы, **представляющие HTML-разметку**: *<H2>Book Description</H2>,* который отображает заголовок второго уровня; `<SPAN STYLE="font-style:italic">Author: </SPAN>` — отображает блок текста, набранного курсивом и `<BR/>` — создает пустую строку.

XSL-элементы. Примером такого XSL-элемента в приведенной таблице стилей являются элемент *xsl:value-of*:

```
<xsl:value-of select="BOOK/AUTHOR"/>
```

XSL-элемент `value-of` добавляет текстовое содержимое соответствующего XML-элемента. В рассмотренной выше XSL-таблице, например, атрибуту `select` элемента `value-of` присвоен образец `"BOOK/AUTHOR"`; это приводит к выводу текстового содержимого элемента `AUTHOR` XML-документа.

При этом следует иметь в виду, что элементы XSL-таблицы, представляющие HTML-разметку, должны быть корректно сформированными XML-элементами (теги должны быть закрытыми), а также стандартными HTML-элементами.

На рисунке 5.5 показано отображение XML-документа (листинг 5.2) в IE в соответствии с таблицей стилей (листинг 5.3).

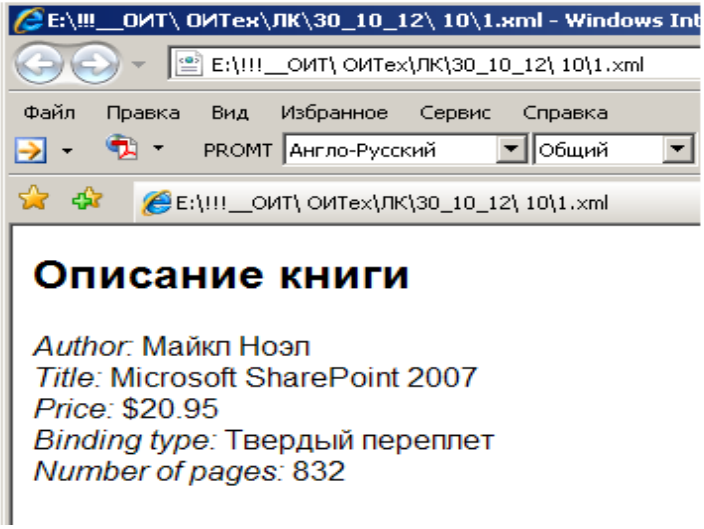


Рисунок. 5.5 Отображение XML-документа в IE.

Рассмотрим более сложный пример. Выведем данные, содержащиеся в XML-документе, который приведен на листинге 5.4, в виде таблицы.

Листинг 5.4. XML-документ

```
<?xml version="1.0"?>
<root>
         <books>
            <book>
               <pages>832</pages>
                 <title>Microsoft SharePoint 2007</title>
                   <author>Michael Noel</author>
            </book>
            <book>
                 <pages>448</pages>
                  <title>Windows SharePoint Services 3.0</title>
                    <author>Ted Pattison</author>
            </book>
             <book>
                  <pages>554</pages>
               <title>Microsoft SharePoint Designer 2007</title>
                     <author>Penelope Coventry</author>
             </book>
        </books>
</root>
```

Шаблон преобразования этого XML-документе приведен на листинге 5.5, а на листинге 5.6 приведен XML-документ с адресом данного шаблона.

Листинг 5.5. XSL-таблица XSL-документа

```
<?xml version="1.0" encoding="WINDOWS-1251" ?>
<xsl:stylesheet version="1.0"xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<table border="1">
<tr bgcolor="#CCCCCC">
<td align="center"><strong>Страниц</strong></td>
<td align="center"><strong>Название</strong></td>
<td align="center"><strong>Автор</strong></td>
</tr>
<xsl:for-each select="root/books/book">
<tr bgcolor="#F5F5F5">
<td><xsl:value-of select="pages"/></td>
<td align="right"><xsl:value-of select="title"/></td>
<td><xsl:value-of select="author"/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

Листинг 5.6. XSL-документ содержащий адрес шаблона.

```
xml version="1.0"?>
xml-stylesheet type='text/xsl' href='7ex03-1.xsl'?>
oot>
      <books>
         <book>
            <pages>832</pages>
              <title>Microsoft SharePoint 2007</title>
                 <author>Michael Noel</author>
         </book>
          <book>
              <pages>448</pages>
               <title>Windows SharePoint Services 3.0</title>
                 <author>Ted Pattison</author>
         </book>
          <book>
              <pages>554</pages>
               <title>Microsoft SharePoint Designer 2007</title>
                 <author>Penelope Coventry</author>
           </book>
      </books>
</root>
```

На рисунке. 5.6 показана таблица, содержащая данные XSL-документа, приведенного на листинге 5.6.

Страниц	Название	Автор
832	Microsoft SharePoint 2007	Michael Noel
448	Windows SharePoint Services 3.0	Ted Pattison
554	Microsoft SharePoint Designer 2007	Penelope Coventry

Рисунок 3.6. Таблица, содержащая данные XSL-документа.

В заключение следует отметить, что XSLT представляет собой мощную технологию, предоставляющую высокоуровневые средства для манипуляции данными, хранимыми в виде XML. В этом плане XSLT сравним с языком SQL. Однако этот достаточно простой язык имеет свои технологические границы. Существуют задачи, решения которых с помощью XSLT либо не целесообразно, либо вообще не существуют. В этих случаях прибегают к XSLT- расширениям, позволяющим использовать традиционные языки программирования.

3.3.3. Инструментальные средства XML

В настоящее время существует большое количество инструментальных средство для работы с XML:

Программы для разработки и проверки XML-карт сайтов (Google SiteMap Generator, Gsite Crawler, Automapit sitemap validator, XML sitemaps validator и др.).

Генераторы RSS-фидов (**IceRocket RSS builder, Feedity, RSSPect**).

Генераторы XML-схем (**Trang от ThaiOpenSource, XMLBeans, XML for ASP BuildXMLSchema**).

Программы для проверки корректности XML (**XML Validation.com, DecisionSoft.com Schema Validator, W3C XML validator)**.

Программы форматирования XML (**XMLIndent.com, X01 online xml formatter).**

Редакторы XML (**Xerlin XML Editor, Jaxe Editor, XMLFox)**.

Инструментальные средства для работы с XML (**Tiger XSLT Mappe, Kernow, и др.)**.

Инструментальные средства с открытыми исходными кодами для работы с XML (**XML IDE iXedit**, **Rinzo XML Editor**, **XPontus XML Editor**).

Интегрированные среды разработки XML (**XMLSpy, XML Marker, Liquid XML Studio и др.)**.

Средства сравнения XML (**Liquid XMLDiff, ExamXML, DeltaXML**).

5.4. Технология XHTML

5.4.1. Недостатки HTML

Язык HTML обеспечивает представление информации (размещение на странице текста, картинок и кнопок) ни как не связанное с содержанием документа, что отражается на сложности разработки современных Web-сайтов, которые должны обеспечивать работу над содержимым сайта независимо от его внешнего представления (принимать от пользователей заказы, пересылать требуемую информацию и т. д.).

Другими словами, современные приложения нуждаются не только в языке представления данных на экране клиента (с этим успешно справляется HTML), но и в механизме, позволяющем определять структуру документа, описывать содержащиеся в нем элементы, для поиска нужных фрагментов документа. Так что на определенном этапе развития Internet-технологий сложился определенный разрыв между потребностями практики и состоянием

в области Web-технологий, на устранение которого и направлена технология XML.

Для обеспечения плавного перехода с технологии HTML на технологию XML разработан стандарт XHTML. В XHTML используются стандартные HTML теги и некоторые собственные теги, правила интерпретации, для которых заданы во внешних файлах, либо встроены в клиент (браузер). Начиная с MS Office 2003 г. в состав MS Office входит редактор XML, с помощью которого пользователи, не имея навыков в работе с языком, могут легко создавать XML-документы.

5.4.2. Версии XHTML

В 1998 году Consortium W3C начал работу над новым языком разметки. Этот новый язык получивший название **XHTML** основан на **HTML 4**, но соответствует синтаксису языка **XML**. Версия XHTML 1.0, одобрена в качестве Рекомендации консорциума W3C в январе 2000 года.

Язык разметки web-страниц XHTML (Extensible Hypertext Markup Language) является преемником HTML, в отличие от HTML соответствует спецификации XML. Существует следующие версии XHTML:

XHTML 1.0,

XHTML 1.1,

XHTML Basic,

XHTML MP,

Спецификация XHTML 1.0 (одобрена Консорциумом W3C в январе 2000 года; вторая редакция этой спецификации опубликована в августе 2002 г.) определяет три типа документов:

XHTML1.0 Transitional (переходный). Этот тип имеет минимальные отличия от HTML, которые в основном направлены на приведение в соответствие с XML. В частности, тип XHTML1.0 Transitional требует, чтобы тэги были правильно вложены, семантически развиты, записывались строчными буквами, а атрибуты, включая численные, заключены в кавычки; все теги были закрыты.

XHTML1.0 Strict (строгий) полностью отделяет содержание документа от оформления. Оформление задаётся только через CSS. Многие атрибуты такие как, например, align, font, center и др. не поддерживаются.

XHTML 1.0 Frameset (фреймовый). Используется, если необходимо разделить окно браузера на несколько фреймов.

XHTML 1.1 — последняя (старшая) версия XHTML1.0 Strict.

XHTML Basic; облегчённая версия XHTML для устройств, с не полным набором элементов XHTML (для смартфонов, коммуникаторов, мобильных телефонов, КПК; замена WML и C-HTML).

XHTML (Mobile Profile): спецификация XHTML, основанная на XHTML Basic, добавляет специфические элементы для мобильных телефонов

5.4.3. Основные правила перехода от HTML к XHTML

Основные правила, которые необходимо соблюдать при переходе от HTML к XHTML:

1. Строгое форматирование документа

Все элементы должны иметь закрывающий тег (например, * ;*в HTML закрывающий тег не является обязательным) или быть написаны в специальной форме (например, <br />).

Должна соблюдаться корректная вложенность элементов.

ПРАВИЛЬНО: вложенные элементы
выделяем это слово.

НЕПРАВИЛЬНО: элементы перекрываются
выделяем это слово. .

2. Имена элементов и атрибутов должны записываться в нижнем регистре (строчными буквами)

Документы XHTML должны использовать нижний регистр для всех имен элементов и атрибутов HTML, поскольку XML чувствителен к регистру, например, <ul> и <UL> это различные теги.

3. Строгое соблюдение правил написания значений атрибутов

Все значения атрибутов, даже цифровые должны быть заключены в двойные ("… ") или одинарные('… ') кавычки.

ПРАВИЛЬНО: атрибут в кавычках
<td colspan="8">

НЕПРАВИЛЬНО: атрибут не в кавычках
<td colspan=8> .

Не допускается минимизация атрибутов.

ПРАВИЛЬНО: атрибут написан полностью
<dl compact="compact">

НЕПРАВИЛЬНО: атрибут минимизирован.
<dl compact>.

4. Недопустимо отсутствие тегов у непустых элементов

Все элементы, за исключением объявленных, в определении типа документа, должны иметь теги.

ПРАВИЛЬНО: закрытые элементы
Здесь какой-то текст. Еще параграф с текстом.

НЕПРАВИЛЬНО: незакрытые элементы
Здесь какой-то текст. Еще параграф с текстом.

5. Пустые элементы обязаны иметь конечный тег

Пустые элементы обязаны иметь конечный тег, или стартовый тег должен заканчиваться />. Например,
 или
</br>. Для совместимости с устаревшими браузерами необходим пробел перед /> в одиночном пустом элементе.

6. Элементы с атрибутами "id" и "name"

В XHTML 1.0 атрибут id определён как атрибут типа ID. Чтобы быть уверенным, что документы XHTML 1.0 являются правильно структурированными, документы XHTML 1.0 должны использовать атрибут id для определения идентификаторов фрагмента даже в тех элементах, которые исторически имели атрибут name. В XHTML 1.1 для элементов a и map атрибут name удален, вместо него используется атрибут id.

7. Символы < и & в тексте документа

Символы < и & в тексте документа должны быть заменены соответствующими последовательностями < и &.

5.5. Технология ASP.NET

Технология ASP**.**NET похожа на оригинальную ASP, но внутренняя реализация ASP.NET полностью переделана. К тому же ASP**.**NET позволяет отделять HTML-код от алгоритмов сценариев. ASP.NET включает поддержку Web Forms, что позволяет разрабатывать web-приложение как обычное приложение для Windows (так что можно написать целый сайт прямо в Visual Basic).

Основная идея технологии **.NET**, состоит в следующем.

Программа записывается на промежуточном языке MSIL (Microsoft Intermediate Language), для которого у каждой аппаратной платформы и программной среды должен существовать свой (соответствующий промежуточному языку) компилятор. Разработанная программа храниться на компьютере не в готовом к исполнению машинном коде, а в виде PE-файла (Portable Executable File). Компиляция PE-файла осуществляется компилятором соответствующей платформы (программной среды) только в момент запуска приложения, так что программа в виде PE-файла может работать на различных платформах.

Цена за эту универсальность более медленное выполнение приложения и необходимость дополнительной надстройки над операционной системой. Такая надстройка, получившая название Common Language Runtime (CLR), предоставляет приложению необходимые интерфейсы и обеспечивает управление исполняемым кодом.

В операционных системах от Microsoft CLR представлен пакетом **.**NET Framework (В Windows 2003 и выше эта среда является неотъемлемой частью ОС), который является надстройкой над ОС Windows. В отличие от традиционных Windows-приложений, реализованных в виде машинного кода, который напрямую взаимодействует с функциями Windows (Win API, COM-компоненты, системные сервисы), .NET-приложения выполняются через функциональный слой .NET Framework.

Программные компоненты в рамках архитектуры .NET реализуются в виде байт-кода на внутреннем языке MSIL, который "исполняется" в среде CLR с применением единых библиотек стандартных классов, средств доступа к удаленным сервисам, а также средств ОС (Win API, COM и т.п.). В отличие от Java байт-код в CLR выполняется не в режиме интерпретации, а путем предварительной компиляции.

Кроме языков программирования от Microsoft (VB.NET, C#, JScript, J#), существуют десятки других языков (Perl, Cobol, FORTRAN, Smalltalk и т.д.), которые обеспечивают работу .NET-приложений (путем преобразование исходного кода в MSIL). Такая унификация фактически нивелирует языковые различия, так что выбор инструментального языка в этой технологии теперь определяться лишь привычкой программиста к тому или иному синтаксису (можно использовать как VB.NET, так и C#.NET).

Пакет инструментальных средств Visual Studio .NET базируется на архитектуре .NET Framework. В нем реализована интегрированная среда разработки IDE (Integrated Development Environment) для всех .NET-языков (C++, VB.NET, C#, JScript и J#) . При этом только C++ позволяет создавать программы на "родном" машинном коде, все же остальные используют MSIL+CLR.

5.6. Системы управления контентом

Система управления контентом CMS (Content Management Systems) представляет собой программный комплекс, который автоматизирует управление контентом системы. Под контентом (content), понимают информационное наполнение сайта (web-страницы, документы, программы, аудио-файлы, фильмы и т.д.). Управление контентом включает: размещение и удаление материалов на сервере, организацию и реорганизацию материалов и отслеживания их состояния.

В связи с развитием технологии корпоративных порталов понятие CMS оказалось расширенным, включающим управление совместной (коллективной) работой пользователей и другие аспекты, связанные со сложностью управления и мониторинга современного портала. Так что с известной условностью можно выделить три основных вида систем управления контентом:

Разработки крупных компаний, ориентированные на создания корпоративных порталов, такие как IBM WebSphere Portal, Microsoft SharePoint и др.

Разработки небольших компаний (коммерческие CMS), ориентированные на создание различного вида сайтов (Bitrix, NetCat, и др.).

Системы с открытым исходным кодом, такие как Drupal, Mambo, Joomla и другие.

Портальные технологии рассматриваются в следующем разделе курса; ниже дана краткая характеристика некоторых CMS.

Joomla. Популярная универсальная CMS. Имеет широкую сферу применения. Для этой CMS разработано большое количество модулей и шаблонов. На Joomla можно создавать самые различные сайты от простых до

сложных. В качестве недостатков можно отметить проблемы с ошибками и торможение при большой посещаемости сайта.

Wordpress. Популярная, хорошо русифицированная, имеющая множество дополнительных модулей и шаблонов, CMS. Позволяет создавать сайты различного типа: информационные, новостные и т. п. Является лидером среди движков для блогов. Характеризуется теми же недостатками, что и Joomla.

Drupal. CMS предназначена для создания форумов, блогов, сайтов сообществ. Drupal является одной из самых надежных систем. К недостаткам использования Drupal можно отнести некоторую сложность (необходимы начальные знания веб-программирования).

Среди коммерческих CMS лидером является CMS **Bitrix,** которая является наиболее мощной среди русскоязычных CMS. К конкурентам **Bitrix** можно отнести: **NetCat**, **ABO.CMS**, **Amiro.CMS**, **UMI.CMS**, **Host.CMS,** а также **DataLife Engine.**

Bitrix. Профессиональная система для создания и управления корпоративными сайтами, интернет-магазинами, информационными порталами, интернет-сообществами, социальными сетями и другими сайтами. Имеет хорошую документацию на русском языке.

NetCat. Профессиональная CMS, предназначенная для создания веб-сайтов различного уровня сложности; нетребовательна к системным ресурсам.

DataLife Engine. Многопользовательский новостной движок, предназначенный для организации собственных СМИ в интернете.

6. ПОРТАЛЬНЫЕ ТЕХНОЛОГИИ

6.1. Понятие портала

Компания Gartner Group (один из ведущих мировых аналитических и консалтинговых центров) определяет портал как «инфраструктуру сетевого программного обеспечения, которая дает доступ (и обеспечивает взаимодействие) определенным целевым аудиториям к значимым информационным базам (например, информация/контент, программные приложения и бизнес процессы), базам знаний и пр., доставляемых высоко персонифицированным способом».

Корпоративный портал должен обеспечивать:

- единую точку доступа к корпоративной информации и корпоративным приложениям,
- аутентификацию и персонализацию,
- управления пользовательским интерфейсом,
- управление администрированием портала,
- управление безопасностью портала.

При этом важно отметить, что корпоративный портал являясь интегратором данных и приложений, не заменяет другие ИТ-системы и приложения корпорации. Внедрение корпоративного портала позволяет повысить уровень управления и поддержки принятия решений; обеспечить экономию времени и усилий сотрудников; сократить расходы и издержки компании; повысить качество работы и уровень взаимодействия сотрудников компании; повысить качество обслуживания клиентов и конкурентоспособность компании.

6.2. Классификация порталов

Упрощенная классификация порталов приведена на рисунке 6.1.

Рисунок 6.1. Классификация порталов.

Публичные порталы часто называют горизонтальными порталами. Примерами публичных универсальных порталов (мегапорталов) являются Google, AltaVista, Rambler и др. Это Internet-порталы (Web-порталы), которые являются в известном смысле аналогом традиционных публичных библиотек. В отличие от названных порталов, публичные специализированные порталы ориентированы на определенную предметную область или аудиторию, например, портал для женщин iVillage.com.

Корпоративный портал — это Web-портал, который представляют собой единую Web-точку доступа к корпоративной информации, сервисам и приложениям, доступную для определенной группы (групп) пользователей. В зависимости от отношения пользователей этих групп к самой корпорации (организации) корпоративные порталы подразделится на B2C, B2B, B2E порталы. Корпоративный портал может быть и многоцелевым, интегрированным. В таком портале B2C, B2B, B2E порталы объединены (интегрированы) как «несминаемые» секции. Иногда для таких объединений вводят специальные обозначения, например, **B2B2C (**интеграция B2B и B2C**).**

B2C (Business-to-Customer) — бизнес-потребитель. Взаимодействие продавца и покупателя (приобретение клиентом товаров, услуги, приобретение страховок и пр.). B2C-порталы, позволяют компаниям наладить контакт с розничными потребителями товаров и услуг.

B2B (Business-to-Business) — бизнес-бизнес. Взаимодействие одного бизнеса с другим (организация поставок, заказы, финансовые потоки, координация действий, совместные мероприятия).

B2E (Business-to-Employee) — бизнес-сотрудник. Взаимодействие корпорации с сотрудниками (справочная информация, табельный учет, управление льготами, формирование отчетов, подбор работы внутри организации и т.д.). Продвинутый B2E портал позволяет решать управленческие и организационные задачи, а также координировать групповую работу сотрудников.

Кроме названных выше корпоративных порталов находят применение и другие виды порталов. Например, с концепцией Электронного Правительства (E-Government), согласно которой государство (правительство и государственные службы) рассматривается, как крупная специфическая корпорация связаны понятия:

G2C (Government-to-Citizens) — Правительство-Граждане (доступ граждан к государственной информации).

G2B (Government-to-Business) — Правительство-Бизнес. (отношения правительства и бизнеса).

G2G (Government-to-Government) - Правительство-Правительство (отношения государственных органов между собой).

6.2. Архитектура корпоративного портала

Для корпоративных порталов не существует единого промышленного стандарта. Тем не менее, корпоративных порталы содержат обязательный набор следующих компонентов (и этим они отличаются от Web-сайтов):

Web-интерфейс.
Управление пользовательским интерфейсом.
Механизм доступа к внешним данным.
Сервисы управления данными.

Средства поиска.
Аутентификация и персонализация.
Обеспечение безопасности.
Средства разработки портала.
Средства управления и администрирование портала.

Функционально корпоративный портал может содержать следующие подсистемы:

1. Подсистема управления задачами, поручениями и контроля их выполнения,
2. Подсистема управления взаимоотношениями с клиентами и партнерами (CRM): база компаний партнеров и клиентов, планирование продаж, отслеживание процесса продаж.
3. Подсистема управления и планирования использованием ресурсов (ERP): планирование нагрузки персонала, учет рабочего времени, планирование и учет трудозатрат, учет производственных расходов и издержек.
4. Подсистема автоматизации управления персоналом: обучение и тестирование персонала, оперативная связь с сотрудниками.
5. Подсистема управления знаниями: автоматизация документооборота, WiKi подсистема, файловый архив.
6. Информационная подсистема: поиск сотрудников, социальные сети, блоги, форумы и опросы.
7. Подсистема сервиса: заказ пропусков и мест на парковке, заказ транспорта, заказ курьерской доставки и переговорных комнат.

В составе корпоративного портала условно можно выделить три основных функциональных уровня (см. рисунок 6.2):

Уровень 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 Web-сервер, либо несколько подобных серверов.

Уровень интеграции приложений, отвечающий за взаимодействие портала со всеми существующими в компании приложениями, такими как СУБД, CRM- и ERP-системы, унаследованные приложения и др.

Уровень интерфейсов, включающий средства управления информационным наполнением, интерфейсы для обмена данными с информационными системами бизнес-партнеров, средства для работы с мобильными и беспроводными устройствами и др. К этому же уровню относятся и портлеты.

Рисунок 6.2. Функциональная архитектура корпоративного портала

Как правило, порталы обладают открытой архитектурой, позволяющей расширять их функциональность за счет добавления сторонних приложений или дополнительных компонентов.

6.3. Технологии порталов

В развитии технологии порталов (начиная с 1997 г.) можно выделить пять этапов.

Для первого этапа характерны персонифицированный доступ к информации, унифицированный поиск информации, управление интерфейсом на базовом уровне.

Технологический фундамент второго этапа дополнился надежной расширяемой средой приложений, их базовой интеграцией и зачатками функций коллективной работы.

Третий этап характеризуется интеграцией процессов, а также базовой поддержкой Web-служб.

Четвертый этап обеспечивает поддержку интегрированных порталов, композитных приложений, стандартов портлетов (Java Specification Request, JSR-169) и Web-служб для удаленных портлетов (Web Services for Remote Portlets, WSRP).

Портлет (портальный аплет) — это миниатюрное Web-приложение (Web-компонент), выполняющееся внутри страницы портала. Портлет способен обрабатывать запросы и генерировать динамическое содержимое (контент).

Пятый этап развития портальной технологии связан с применением сервис-ориентированной архитектуры. Порталы пятого поколения обладают такими преимуществами, как гибкое объединение и легкая модификация автономных

подсистем и приложений. Примерами названной портальной технологии является технологии SharePoint Server 2007, 2010, 2013, 2016.

Создание корпоративного портала может быть реализовано путем покупки готовых решений, разработки с нуля (заказные разработки) и с применением комбинированного подхода. Разработка и внедрение корпоративного портала, как и создание любой другой ИС с большим количеством пользователей, требует профессионального промышленного подхода к разработке и внедрению портала (проектирование, разработка, тестирование, обучение, ввод в эксплуатацию). На стадии внедрения корпоративного портала на передний план выдвигаются административные проблемы, вопросы информационной безопасности и передачи знаний от разработчика к конечным пользователям.

Внедрение корпоративных порталов, как правило, связано со следующими проблемами:

- необходимость изменения бизнес-процессов компании и перераспределения ответственности;
- отсутствие формализации процессов и их документирования, не позволяющей произвести четкую постановку задачи;
- привычка работать на "личном контакте";
- низкая культура использования компьютеров в компаниях;
- сопротивление со стороны персонала компаний при внедрении системы;
- недостаточная твердость руководства при внедрении системы в компании;
- низкий уровень понимания потенциальных выгод от использования корпоративных информационных систем руководством и сотрудниками компании;
- сложность с формализацией распределения прав доступа к информации;
- попытка переноса имеющегося в компании информационного беспорядка в автоматизированный беспорядок посредством портала;
- отсутствие опыта эксплуатации подобных систем, низкий уровень понимания возможных вариантов использования портала.

Мировыми лидерами в области "порталостроения" являются: IBM, Microsoft, Oracle, SAP**.**

6.4. **Портальные технологии от Microsoft**

Технологии SharePoint включают: Windows SharePoint Services 2.0 и 3.0, SharePoint Services 2016; SharePoint Server (2003, 2007, 2010, 2013, 2016); SharePoint Designer (2007, 2016).

Windows SharePoint Services (WSS) 2.0 - это компонент Windows Server 2003, предоставляющий набор служб для создания узлов совместного использования информации и взаимодействия с другими пользователями. Компонент WSS 3.0 является бесплатным дополнительным продуктом, который может быть установлен на Windows Server 2003 (вместо WSS 2.0).

Предоставляя основные возможности управления документами, совместной работы и поиска WSS 3.0 вместе с этим не обеспечивает полного набора средств, необходимых крупным организациям. Расширенные возможности обеспечивает Microsoft SharePoint Server 2007. Наиболее же широкие возможности обеспечивает SharePoint Server 2016, позволяющий работать в облачной среде с применением мобильных и сенсорных устройств.

SharePoint Server 2007 имеет открытую масштабируемую архитектуру с поддержкой Web-служб и стандартов совместимости, включающих XML и Simple Object Access Protocol (SOAP). Этот сервер также обладает открытыми интерфейсами программирования приложений и обработчиками событий для списков и документов. Портал SharePoint Server 2007 расширяет функциональный набор, предоставляемый приложениями Microsoft Word, Excel, PowerPoint и др., а также обеспечивает работу разнообразных сервисов, которые расширяют функциональность базового портала.

Программа Microsoft Office SharePoint Designer 2007 частично основана на FrontPage 2003 и частично на Microsoft Visual Studio 2005. SharePoint Designer 2007 позволяет подключаться к узлу WSS 3.0 или к SharePoint Server 2007 и работать со страницами этого узла в визуальном режиме.

Технология Windows SharePoint Services (WSS) 3.0 основана на принципе фермы. Ферму составляет один или нескольких серверов. В простом случае ферма WSS состоит из одного сервера, на котором одновременно реализуется веб-сервер и сервер баз данных. Более сложная ферма WSS включает несколько веб-серверов и серверов баз данных.

При инсталляции WSS создаются два приложения:

Центр администрирования WSS 3.0.

Веб-приложение WSS для доступа конечных пользователей к WSS-узлам через стандартный для HTTP 80 порт FTP.

В пределах фермы могут быть созданы дополнительные веб-приложения, Процесс создания дополнительных веб-приложений будет рассмотрен ниже.

При загрузке центра администрирования WSS 3.0. появляется страница WSS 3.0, открытая на вкладке ***Домашняя*** (см. рисунок. 6.3)

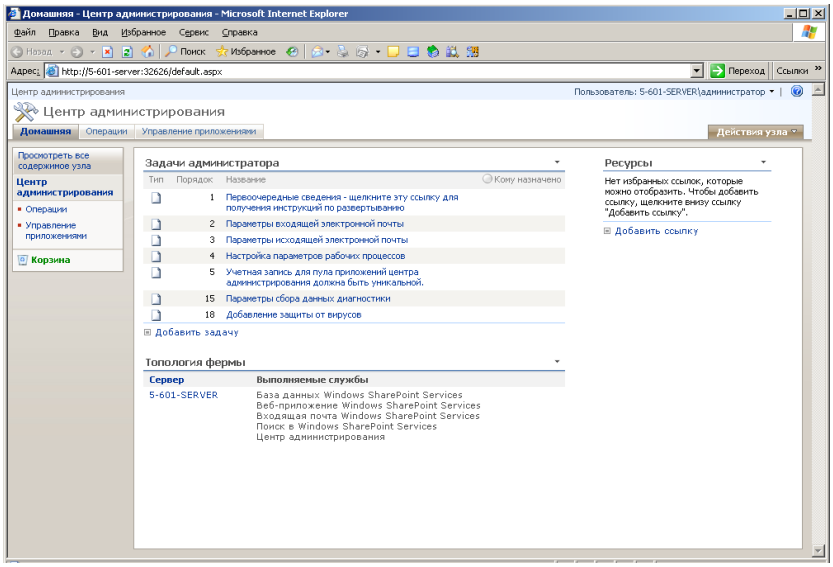


Рисунок.6.3. Домашняя страница Центра администрирования.

Домашняя страница имеет закладки: ***Задачи администратора*** *(*инструкции по развертыванию WSS 3.0, сведения о параметрах настройки SharePoint, инструкции по их настройке), ***Ресурсы*** (сведения об используемых ресурсах) ***и Топология фермы*** (выполняемые службы на сервере).

Вкладка ***Операции*** (см. рисунок. 6.4) содержат закладки: *топология и службы, глобальная конфигурация, настройка безопасности, ведение журналов и составление отчетов, резервное копирование и восстановление, конфигурация данных*, которые используются для изменения топологии фермы, указания служб, запускаемых на каждом сервере, изменения параметров серверов или приложений.

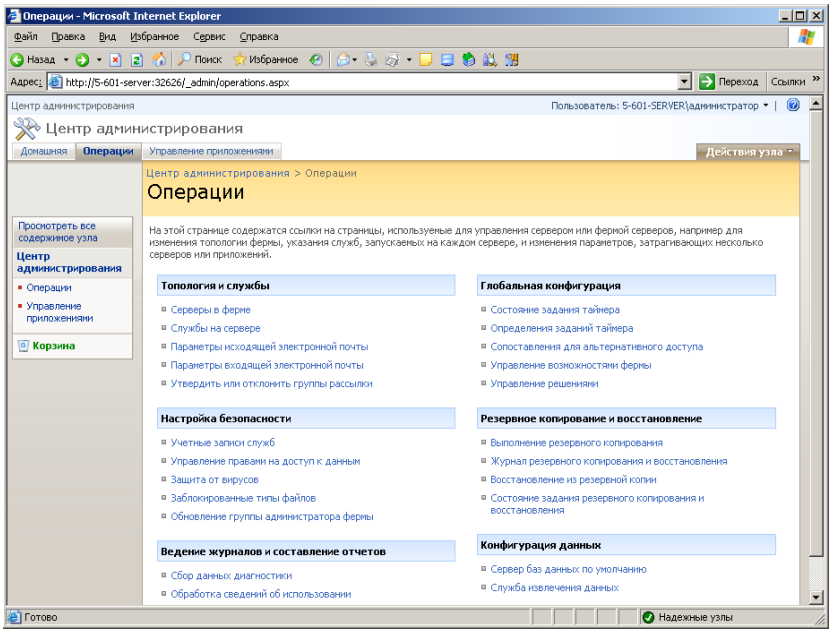


Рисунок.6.4. Страница вкладки *Операции* Центра администрирования.

Страница вкладки ***Управление приложениями*** (см. рисунок 6.5) содержит закладки:

Управление веб-приложениями SharePoint.
Управление узлами SharePoint .
Безопасность приложений.
Управление рабочими процессами.
Подключения к внешним службам.

Закладка ***Управление веб-приложениями*** содержит ссылки на следующие страницы:

создание или расширение веб-приложения,
удаление sharepoint с веб-узла IIS,
удаление веб-приложения,
определение управляемых путей,
параметры исходящей электронной почты веб-приложения,
общие параметры веб-приложений,
базы данных содержимого,
управление возможностями веб-приложений,

список веб-приложений.

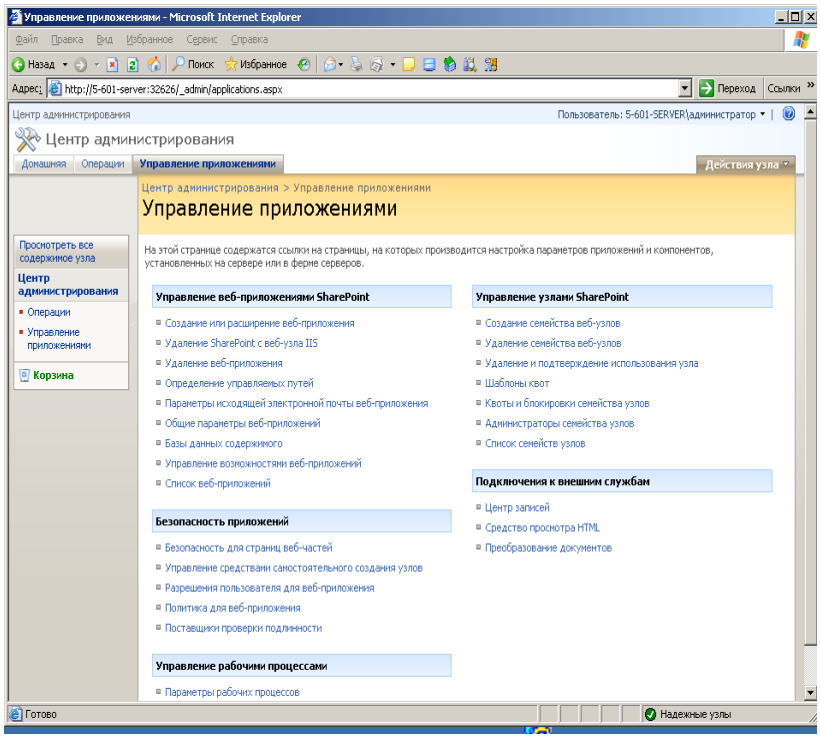


Рисунок. 6.5. Страница вкладки *Управление приложениями* Центра администрирования

Для создания нового приложения SharePoint необходимо выбрать страницу: ***Создание или расширение веб-приложения*** (см. рисунок 4.6) и на этой странице выбрать вкладку: **создать *веб-приложение***, см. рисунок. 6.7.

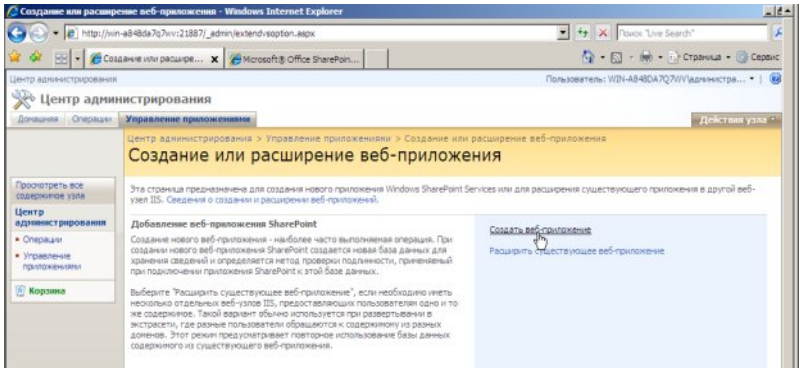


Рисунок 4.6. Страница: *Создание или расширение веб-приложения*

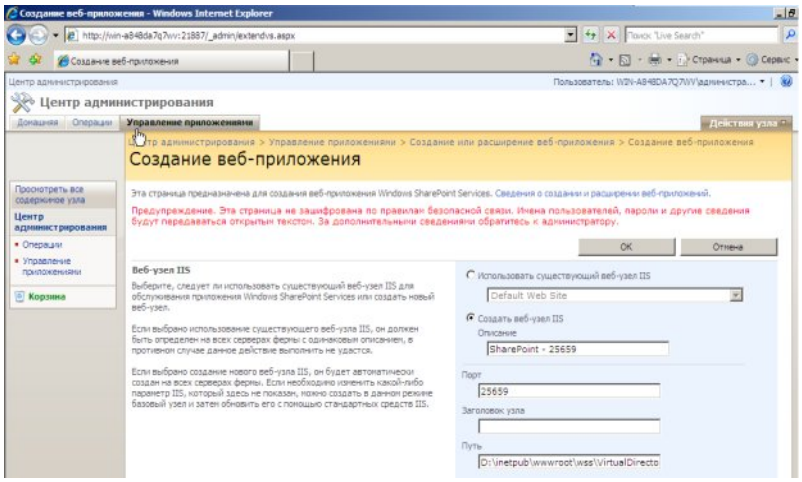


Рисунок.4.7. Страница ***Создание веб-приложения.***

После определения требуемых параметров (на рисунке 6.7 показана только часть страницы) необходимо нажмите кнопку **ОК** и дождаться сообщения об успешном создании ***веб-приложения***.

После создания нового ***веб-приложения SharePoint*** необходимо создать **семейство веб-узлов** в рамках этого приложения. Одно ***веб-приложение*** может иметь одно или несколько ***семейств веб-узлов***, которые, в свою очередь, включают в себя отдельные ***веб-узлы***. Каждая ступень иерархии "*веб-приложение – семейство веб-узлов – веб-узел* верхнего уровня – *дочерние веб-узлы*" может иметь своего собственного администратора, который вправе делегировать полномочия администраторам более низких ступеней.

Для создания **семейства веб-узлов** необходимо на странице ***Управление приложениями*** выбрать на вкладке ***Управление узлами*** SharePoint (см. рисунок 6.5) *закладку* ***Создание семейства веб-узлов.*** В результате на экране появится страница ***Создание семейства веб-узлов*** (см. рисунок 6.8).

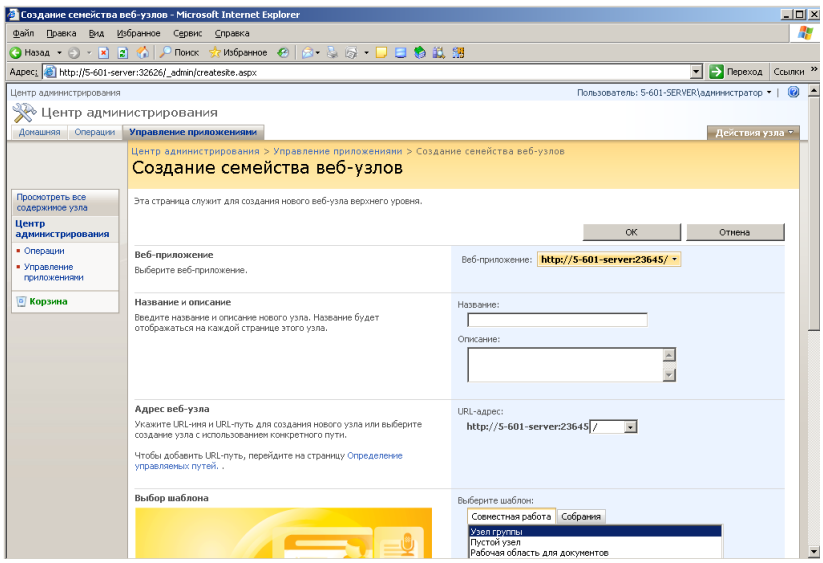


Рисунок.6.8**.** Страница *Создание семейства веб-узлов.*

После задания требуемых параметров (на рис. 6.8 показана часть страницы) следует нажать кнопку **ОК** и дождаться успешного создании ***семейства веб-узлов***.

Следует иметь в виду, что новое приложение SharePoint и семейство веб-узлов создается и конфигурируется на сервере сетевым администратором. Отдельные же веб-узлы этого приложения создаются и управляются с клиентских компьютеров уполномоченными членами рабочей группы. При этом администратор сервера уже может не иметь доступа ни к созданию, ни к изменению, ни даже к просмотру отдельных узлов.

7. ТЕХНОЛОГИЯ БЕЗОПАСНОСТИ В INTERNET

7.1. Проблема безопасности в Internet

Безопасность в Internet является частным, но наиболее сложным аспектом информационной безопасности. Проблема безопасности в Internet усугубляется тем, что Internet разрабатывалась как открытая система, при этом вопросам безопасности стека протоколов TCP/IP уделялось очень мало внимания.

Под информационной безопасностью понимают защищенность информации от случайных или преднамеренных воздействий естественного или искусственного характера, связанных с нанесением ущерба информационной системе.

Безопасность в Internet является комплексной проблемой, неразрывно связанной с защитой современной корпоративной сети. Защита же сети является одним из наиболее важных и сложных аспектов сетевой технологии. Особенность этого аспекта (в плане профессиональной подготовки) — высокие требования к специалисту по информационной безопасности (к специалистам отдела информационной безопасности, а в небольших организациях — к системному администратору). Специалист по информационной безопасности должен обладать не только глубокими специальными знаниями, должен не только регулярно отслеживать состояние в области безопасности, но и уметь руководствоваться здравым смыслом.

7.2. Политика безопасности

Важным этапом обеспечения информационной безопасности является разработка политики безопасности, формирование которой требует системного подхода, включающего многочисленные аспекты.

В политике безопасности должны быть отражены**:**

1. Предмет политики (предметная область, терминология, цель и причины разработки политики**)**.

2. Позиция руководства организации (решение руководства по данной политике, разрешения и запреты по использованию ресурсов).

3. Применимость политики (где, как, когда, кем и к чему применяется данная политика).

4. Роли и обязанности должностных лиц (ответственные лица и их обязанности в плане разработки и внедрения политики безопасности).

5. Реализация политики (регламент, нарушения и наказание).

6. Справочная информация и консультанты по безопасности.

7.3. Защита сети и средства безопасности

Защита сети охватывает следующие уровни:

1. Физическая защита.
2. Контроль действий пользователей.
3. Программная защита.

Физическая защита предполагает ограничение доступа пользователей к серверам, маршрутизаторам, брандмауэрам и другой сетевой аппаратуре путем ограничение доступа в помещение, в котором расположена аппаратура, или/и посредством идентификационных карт, карточек-ключей и т. п.

Контроль действий пользователей предполагает аутентификацию, аудит и управление рабочей средой пользователей. Неправильно сформированная в этой части политика безопасности или ее несоблюдение (записанный на бумажке пароль или незаблокированная администратором консоль) может свести на нет все усилия по обеспечению физической или программной защиты сети.

Программная защита предполагает анализ уязвимости программной среды, идентификацию и устранение потенциальных точек воздействия, закрытие дыр, черных ходов (back doors) и защиту от атак. В сети, требующей высокого уровня защиты, значительная часть работы связана с отслеживанием новых эксплойтов (документированных способов проникновения в систему) и осуществлением превентивных мер, направленных на устранение возможности их применения в сети.

Технология защиты сети базируется на использовании определенного комплекса средств обеспечения безопасности. К основным средствам обеспечения безопасности относятся: антивирусные программы, антишпионские программы, анализаторы сетевых протоколов, брандмауэры, сканеры сетевой безопасности.

Несмотря на сложность и многоплановость самой проблематики защиты сети потеря работоспособности сети во многих случаях обусловлена тривиальными причинами. Так по данным Международного общества компьютерной безопасности ICSA (International Computer Security Association), более 90% времени, стоимости и усилий, направленных на нейтрализацию последствий потери данных или служб в сетях, затрачиваются вследствие таких банальных причин, как вирусы, нарушение энергопотребления и злонамеренные действия персонала организаций. При этом в 70% проникновение в сети и системы осуществляется самими сотрудниками организаций.

Сложность проблематики информационной безопасности трудно переоценить. Эта сложность обусловлена не только спецификой сетевых технологий и конфликтным характером атакующей и противоборствующей сторон, но и факторами организационного, мотивационного, психологического, финансового и другого характера. При этом здесь очень важна правильная позиции высшего руководства организации и должная оплата труда специалистов, обеспечивающих безопасность системы.

7.4. Брандмауэры

Брандмауэр (firewall) представляет систему, которая в целях безопасности накладывает ограничения на проходящий через нее поток данных. Брандмауэры являются самым распространенным средством защиты сети.

Брандмауэр разделяет сеть на две или более частей посредством реализации набора правил, определяющих условия прохождения пакетов из

одной части в другую. Как правило, эта граница проводится между локальной сетью предприятия и Internet, но может быть проведена и внутри локальной сети предприятия.

Брандмауэры получили признание в начале 1990-х в связи быстрым развитием Internet.

Брандмауэры можно подразделить на следующие категории:

брандмауэры *с* фильтрацией пакетов (packet-filtering firewall);
шлюзы сеансового уровня (circuit-level gateway);
шлюзы прикладного уровня (application-level gateway);

Наибольшее распространение получили брандмауэры с фильтрацией пакетов, которые реализуются на маршрутизаторах и конфигурируются таким образом, чтобы фильтровать входящие и исходящие пакеты.

Фильтры пакетов просматривают поля поступающих IP-пакетов, а затем про пускают или удаляют их в зависимости, например, от IP-адресов отправителя и получателя, номеров портов отправителя и получателя протоколов TCP или UDP и других параметров. Фильтр сравнивает полученную информацию со списком правил фильтрации для принятия решения о разрешении или запрещении передачи пакета.

Технология фильтрации пакетов является самым «дешевым» способом реа лизации брандмауэра. Такой брандмауэр может проверять пакеты различных протоколов, причем с большой скоростью, так как он анализирует на сетевом уровне модели OSI только заголовок пакета.

В настоящее время, несмотря на то, что фильтры пакетов получили широкое распространение они мало подходят для внешней защиты сети; но хорошо подходят для обеспечения безопасности внутри сети (с их помощью можно разбить сеть на защищенные сегменты).

К достоинствам брандмауэров *с* фильтрацией пакетов относятся:

относительно невысокая стоимость,
гибкость в определении правил фильтрации,
небольшая задержка при прохождении пакетов,

Недостатки у данного типа брандмауэров следующие:

локальная сеть видна (маршрутизируется) из Internet;
правила фильтрации пакетов трудны в описании, требуются очень хорошие знания технологий TCP и UDP ;
при нарушении работоспособности брандмауэра все компьютеры за ним становятся полностью незащищенными либо недоступными;
аутентификацию с использованием IP-адреса можно обмануть использованием IP-спуфинга (атакующая система выдает себя за другую, используя ее IP-адрес);
отсутствует аутентификация на пользовательском уровне.

Шлюзы сеансового уровня представляет собой транслятор TCP соединения. Пользователь образует соединение с определенным портом на брандмауэре,

после чего последний производит соединение с местом назначения по другую сторону от брандмауэра. Во время сеанса этот транслятор копирует байты в обоих направлениях**.**

Такой тип брандмауэра позволяет создавать транслятор для любого определенного пользователем сервиса, базирующегося на TCP, осуществлять контроль доступа к этому сервису, сбор статистики по его использованию.

К достоинствам шлюзов сеансового уровня следует отнести их низкую стоимость и незначительное влияние на скорость маршрутизации, а также невидимость компьютеров локальное сети.

Основной недостаток — не позволяют осуществлять фильтрацию отдельных пакетов.

Брандмауэры прикладного уровня (шлюзы прикладного уровня, или прокси-брандмауэры) используют сервера конкретных сервисов - Telnet, FTP, SMTP, POP3, HTTP и т.д., запускаемые на брандмауэре и пропускающие через себя весь трафик, относящийся к данному сервису. Таким образом, между клиентом и сервером образуются два соединения: от клиента до брандмауэра и от брандмауэра до места назначения.

К преимуществам брандмауэров прикладного уровня следует отнести:

локальная сеть невидима из Internet;

при нарушении работоспособности брандмауэра пакеты перестают проходить через брандмауэр, тем самым не возникает угрозы для защищаемых им машин;

защита на уровне приложений позволяет осуществлять большое количество дополнительных проверок, снижая тем самым вероятность взлома с использованием дыр в программном обеспечении;

аутентификация на пользовательском уровне может быть реализована система немедленного предупреждения о попытке взлома.

Недостатками брандмауэра этого типа являются:

более высокая, чем для пакетных фильтров стоимость;

невозможность использовании протоколов RPC и UDP;

производительность ниже, чем для пакетных фильтров.

7.5. Система защиты информации

Процесс создания система защиты информации, как и любой другой сложной системы, представляет собой поэтапный последовательно-циклический процесс. Наиболее полно этот процесс описан в регламентирующих документах и стандартах, например, в ISO 17779, ISO 27001 RFC 2196 и др. Кроме названных стандартов, которые относятся к уровню менеджмента, существует и множество других международных стандартов по информационной безопасности (стандарты на криптосистемы, стандарты защищенной передачи данных и т.д.).

Понятийный базис в области информационной безопасности содержится "Оранжевой книге": стандарт Министерства обороны США (Department of Defense Trusted Computer System Evaluation Criteria, 1985 г.). Позднее была выпущена **"**Радужная серия", наиболее значимым документом, которой

является "Интерпретация "Оранжевой книги" для сетевых конфигураций" (Trusted Network Interpretation; 1987 г.). Этот документ содержит важнейшие концептуальные понятия (включая криптографические аспекты), и описание сервисов безопасности в области сетевых конфигураций.

Одной из наиболее удачных технологий создания современных систем безопасности считается разработанная компания Cisco Systems стратегия безопасности, получившая название SAFE, которая включает следующие этапы:

1. Политика безопасности.
2. Средства обеспечения политики безопасности.
3. Мониторинг.
4. Тестирование
5. Управление и улучшение.

Разработка политики безопасности (см. п. 7.2) организации (компании) предполагает создание документа, регламентирующего принципы информационной безопасности, которыми должен руководствоваться каждый сотрудник организации. На данном этапе целесообразно использовать стандарты ISO 17799 RFC ISO 27001 RFC 2196.

На втором этапе согласно разработанной политике безопасности проектируется система комплексного обеспечения безопасности. Используются межсетевые экраны, системы обнаружения атак, устройства шифрования и другое необходимое оборудование, а также реализуются организационные и физические методы обеспечения безопасности.

Третий этап предполагает внедрение систем постоянного мониторинга и анализа активности в сети компании на базе информации, полученной от систем обнаружения атак, серверов SNMP, а также различных систем регистрации.

Четвертый этап связан с вопросами тестирования существующей сети на предмет ее уязвимости путем использования специализированных сетевых сканнеров, которые позволяют обнаружить слабые места и элементы в системе защиты, и выдают некоторые рекомендации по их устранению.

На пятом, завершающем, этапе реализуется управление всеми устройствами обеспечения безопасности, а также оптимизация параметров элементов защиты в существующей системе. В последующем можно проводить и модернизацию существующей системы безопасности.