Государственное образовательное учреждение

высшего профессионального образования Тюменской области

«ТЮМЕНСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ

МИРОВОЙ ЭКОНОМИКИ, УПРАВЛЕНИЯ И ПРАВА»

Факультет управления

Кафедра математики, информатики

 и естественных наук

 Годовая курсовая работа

на тему:

**Основы разработки компонентов для CMS Joomla 1.5**

Выполнил:

Студент 461 группы

Колмаков Р.А.

Проверил:

Преподаватель

Фокина Е.Н.

Тюмень 2010

**Содержание**

[Введение 4](#_Toc273557800)

[Глава I. Описание технических средств реализации компонента 6](#_Toc273557801)

[1.1. Базовая информация о сети Интернет 6](#_Toc273557802)

[1.2. PHP: процессор гипертекста 10](#_Toc273557803)

[1.3. СУБД MYSQL 15](#_Toc273557804)

[1.4. Веб-сервера 17](#_Toc273557805)

[1.5. Принципы построения клиент / серверных систем 24](#_Toc273557806)

[1.6. Системы управления содержимым 26](#_Toc273557807)

[1.7. Система управления содержимым Joomla CMS 36](#_Toc273557808)

[1.7.1. История возникновения Joomla CMS 36](#_Toc273557809)

[1.7.2. Базовые (стандартные) возможности 37](#_Toc273557810)

[1.7.3. Технические требования 39](#_Toc273557811)

[1.7.4. Интерфейс прикладного программирования Joomla 39](#_Toc273557812)

[Глава II. Разработка компонента для Joomla 1.5 41](#_Toc273557813)

[2.1. Общая структура компонента 41](#_Toc273557814)

[2.2. Создание точки входа компонента (файл hello.php) 43](#_Toc273557815)

[2.3. Создание контроллера компонента (файл controller.php) 45](#_Toc273557816)

[2.4. Создание модели компонента (файл /models/hello.php) 46](#_Toc273557817)

[2.5. Создание вида компонента (файл /views/hello/view.html.php) 48](#_Toc273557818)

[2.6. Создание шаблона данных (файл /views/hello/tmpl/default.php) 49](#_Toc273557819)

[2.7. Создание точки входа компонента для административного интерфейса (файл /admin/hello.php) 49](#_Toc273557820)

[2.8. Создание модели компонента для административного интерфейса (файл /admin/models/hellos.php) 50](#_Toc273557821)

[2.9. Создание вида компонента для административного интерфейса (файл /admin/views/hellos/view.html.php) 52](#_Toc273557822)

[2.10. Создание шаблона вида для административного интерфейса (файл /admin/views/hellos/tmpl/default.php) 53](#_Toc273557823)

[2.11. Создание установочного файла компонента (файл hello.xml) 54](#_Toc273557824)

[Заключение 57](#_Toc273557825)

[Список использованной литературы 58](#_Toc273557826)

# Введение

Интернет стал неотделимой частью современной цивилизации. Стремительно врываясь в сферы образования, торговли, связи, услуг, он порождает новые формы общения и обучения, коммерции и развлечений. «Сетевое поколение» – это настоящий социокультурный феномен наших дней. Для его представителей Интернет давно стал привычным и удобным спутником жизни. Человечество вступает в новый информационный этап своего развития, и сетевые технологии играют в нем огромную роль.

В период раннего развития Интернета, только человек, знающий язык разметки страниц HTML, мог создать и запустить свой собственный интернет-сайт. Изначально сайты представляли собой совокупность простых статических документов, расположенных на сервере.

С постепенным развитием Интернета, развивались и интернет-сайты, как неотъемлемая часть всемирной сети. В настоящее время, большинство сайтов являются динамическими и интерактивными. Теперь не нужно хранить на сервере сотни статичных страниц, достаточно установить и настроить так называемую систему управления содержимым (CMS), которая сама сгенерирует сайт, от пользователя потребуется лишь некоторое время на настройку систему и внесение содержимого сайта. Ему не требуется даже знаний языка HTML.

 В настоящее время разработано много систем управления содержимым, как бесплатных, так и коммерческих. Из всего разнообразия бесплатных систем выделяется система управления содержимым Joomla CMS, количество скачиваний которой достигло 4 миллионов. Joomla CMS распространяется бесплатно и для нее написано огромное количество компонентов, расширяющих стандартный функционал системы.

Однако, несмотря на большое обилие дополнительных компонентов, зачастую в силу некоторых специфических требований к сайту необходимо разрабатывать собственные компоненты, которые могут реализовать эти

требования на сайте, построенном с помощью Joomla CMS. Поэтому с каждым новым пользователем этой системы актуальность знаний основ проектирования и разработки компонентов для Joomla прямо пропорционально увеличивается.

**Объект** курсовой работы: система управления содержимым Joomla CMS.

**Предмет** работы: внутренний интерфейс программирования (API) Joomla CMS.

**Цель** работы: изучить методы и принципы проектирования и разработки компонентов для Joomla CMS, разработать собственный компонент, использующий основные функции встроенного интерфейса программирования системы.

Для достижения поставленной цели, следует выполнить следующие з**адачи**:

* Определить и изучить технические средства, применяемые для разработки компонентов.
* Изучить API и структуру компонентов Joomla CMS.
* Разработать простой компонент, использующий основные возможности API Joomla CMS.

# Глава I. Описание технических средств реализации компонента

## 1.1. Базовая информация о сети Интернет

Интернет – это всемирная компьютерная сеть, объединяющая миллионы компьютеров в единую информационную систему. Интернет предоставляет широчайшие возможности свободного получения и распространения научной, деловой, познавательной и развлекательной информации. Глобальная сеть связывает практически все крупные научные и правительственные организации мира, университеты и бизнес-центры, информационные агентства и издательства, образуя гигантское хранилище данных по всем отраслям человеческого знания. Виртуальные библиотеки, архивы, ленты новостей содержат огромное количество текстовой, графической, аудио- и видео-информации.

Интернет возник как воплощение двух идей – глобального хранилища информации и универсального средства ее распространения.

Человечество с давних пор стремилось упорядочить производимую им письменную информацию. Прообразы каталогов были еще в Александрийской библиотеке, а ранние религиозные тексты содержали развитый аппарат «параллельных мест», то есть указаний на места в рукописи, где описываются те же события.

С появлением научной литературы и лавинообразным накоплением самых различных сведений в письменной форме потребность в их систематизации еще более возросла. В 20 в. архивное дело, каталогизация и реферирование стали профессиями.

Американские ученые Ванневар Буш (Vannevar Bush) и Теодор Нельсон (Theodor Holm Nelson) искали способы автоматизации мыслительной деятельности человека. Они хотели избавить его от утомительного труда по поиску и обработке нужной информации. Буш даже придумал несколько гипотетических устройств, организующих ассоциативные связи в картотеке данных, а Нельсон разработал теорию «документарной вселенной», в которой все знания, накопленные человечеством, представляли бы единую информационную систему, пронизанную миллиардами перекрестных ссылок. Работы этих ученых носили скорее философский, чем практический характер, но их идеи легли в основу того, что мы сейчас называем гипертекстом.

Ванневар Буш немало сделал для того, чтобы наукой заинтересовались военные. Щедрое финансирование исследований в области кибернетики несомненно способствовало ее быстрому развитию. Немалую роль в формировании теоретической базы будущей глобальной информационной системы принадлежит Норберту Винеру. Его блестящие семинары в Массачусетском технологическом институте (MIT) привлекли в компьютерную отрасль немало талантливой молодежи.

В конце 1950-х министерство обороны США учредило Агентство перспективных исследовательских проектов ARPA (Advanced Research Projects Agency), которое занималось компьютерным моделированием военных и политических событий. Талантливый организатор и ученый-компьютерщик Джозеф Ликлайдер (J.C.R. Licklider) убедил руководство ARPA сосредоточить усилия на развитии компьютерной связи и сетей. В своей работе Симбиоз человека и компьютера он развил идеи распределенных вычислений, виртуальных программных средств, электронных библиотек, разработал структуру будущей глобальной сети.

В 1960-х компьютерные сети стали бурно развиваться. Множество фирм-разработчиков создавали программное обеспечение и оборудование для локальных сетей университетов, исследовательских центров, военных учреждений. Однако при передаче информации между сетями разных типов возникала проблема совместимости, когда компьютеры просто «не понимали» друг друга. Крупным недостатком больших сетей была их низкая устойчивость. Выход из строя одного участка мог полностью парализовать работу всей сети.

Перед агентством ARPA была поставлена задача решить эти проблемы, и наступило время воплотить в жизнь теоретические наработки. Поль Барен, Ларри Робертс и Винтсент Серф (Paul Baran, Larry Roberts, Vint Cerf) разработали и применили методы, ставшие основой дальнейшего развития сетевых технологий: пакетная коммутация, динамическая маршрутизация сообщений в распределенной сети, использование универсального сетевого протокола (то есть набора правил, по которым организуется и передается информация).

В 1969 была создана сеть ARPANET, которая и стала основой будущего Интернета. 1969 традиционно считается годом его возникновения.

В 1976 Серф разработал универсальный протокол передачи данных TCP/IP (Transmission control protocol/ Internet protocol). Название IP означало просто межсетевой протокол. Он стал стандартом для межсетевых коммуникаций, а сети, использующие его, так и назывались – интернет-сети.

ARPANET стала основой для объединения локальных и территориальных сетей в единую глобальную систему, которая постепенно разрослась до масштабов всей Земли. Это гигантское объединение сетей и называют Интернетом с большой буквы или Сетью.

В 1980‑х Интернетом пользовались в основном специалисты. По сети передавалась электронная почта и организовывались телеконференции между научными центрами и университетами.

В 1990 программист Европейского центра ядерных исследований (CERN) в Женеве Тим Бернерс-Ли (Tim Berners-Lee) создал систему, реализующую идею единого гипертекстового пространства. Для описания гипертекстовых страниц служил специальный язык HTML (HyperText Markup Language), а для их пересылке по сети – протокол передачи HTTP (HyperText Transfer Protocol). Новый способ указания адресов с помощью URL (Uniform Resource Locator – универсальный указатель ресурсов) позволял легче запоминать их и лучше ориентироваться в информационном пространстве Интернета. Была написана также специальная программа отображения гипертекстовых страниц – первый браузер (browser – обозреватель). Бернерс-Ли назвал свой проект WWW – World Wide Web, то есть «Всемирная паутина».

Но по-настоящему популярным Интернет стал после выхода в свет графического браузера «Мозаика» (Mosaic), разработанного в 1992 сотрудником Иллинойского университета Марком Андресеном (Marc Andreesen). К этому времени возросла пропускная способность сетей, и появилась возможность быстро передавать цветные изображения, фотографии, рисунки. В Интернет хлынула не только научная, но и развлекательная информация. На рисунке 1.1 представлена динамика развития сети с августа 1995 по май 2009 года.

Рисунок 1.1 – Динамика развития интернет-ресурсов с августа 1995 г. по май 2009 г. по данным компании «Netcraft Ltd»

В настоящее время Интернет развивается экспоненциально: каждые полтора-два года его основные количественные показатели удваиваются. Это относится к числу пользователей, числу подключенных компьютеров, объему информации и трафика, количеству информационных ресурсов.

Границы применения Интернета в жизни человечества постоянно расширяются, появляются совершенно новые виды сетевого сервиса и использование телекоммуникационных технологий даже в бытовой технике.

Интернет меняет даже мировоззрение и психологию людей. Молодые люди, выросшие в эпоху передовых технологий, с детства, привыкшие к использованию компьютеров в образовании, развлечениях, на работе, живут в ином восприятии информационного пространства. Для них общение со сверстником из другой части света или мгновенное получение сведений по любому вопросу дело совершенно обыденное.

Жизнь современного общества становится все более компьютеризированной. Растут требования к оперативности и надежности информационных услуг, появляются новые их виды. Уже сейчас ученые разрабатывают принципиально новые формы глобальных информационных сетей. В недалеком будущем многие процессы сетевого проектирования, администрирования и обслуживания будут полностью автоматизированы.

## 1.2. PHP: процессор гипертекста

PHP – это язык программирования, созданный для генерации HTML‑страниц на веб-сервере и работы с базами данных. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров. Входит в LAMP – «стандартный» набор для создания веб-сайтов (Linux, Apache, MySQL, PHP (Python или Perl)).

Название PHP – рекурсивная аббревиатура, означающая «PHP: Hypertext Preprocessor» (ранее акроним расшифровывался как «Personal Home Page Tools»). Изначально PHP создавался в качестве надстройки над Perl для облегчения разработки веб-страниц.

В области программирования для Сети, PHP – один из популярнейших скриптовых языков (наряду с JSP, Perl и языками, используемыми в ASP.NET) благодаря своей простоте, скорости выполнения, богатой функциональности и распространению исходных кодов на основе лицензии PHP. PHP отличается наличием ядра и подключаемых модулей, «расширений»: для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF и т.п. Любой желающий может разработать своё собственное расширение и подключить его. Существуют сотни расширений, однако в стандартную поставку входит лишь несколько десятков хорошо зарекомендовавших себя. Интерпретатор PHP подключается к веб-серверу либо через модуль, созданный специально для этого сервера (например, для Apache или IIS), либо в качестве CGI‑приложения.

Кроме этого, он может использоваться для решения административных задач в операционных системах UNIX, GNU/Linux, Microsoft Windows, Mac OS X и AmigaOS. Однако в таком качестве он не получил распространение, отдавая пальму первенства Perl, Python и VBScript.

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

В 1994 году датский программист (ныне живущий в Канаде) Расмус Лердорф (Rasmus Lerdorf) написал набор скриптов на Perl/CGI для вывода и учёта посетителей его онлайн-резюме, обрабатывающий шаблоны HTML‑документов. Лердорф назвал набор Personal Home Page (Личная Домашняя Страница). Вскоре функциональности и быстроты Perl – интерпретатора скриптов – перестало хватать, и Лердорф написал на языке C новый интерпретатор шаблонов PHP/FI (англ. Personal Home Page / Forms Interpreter – «Личная Домашняя Страница / Интерпретатор форм». PHP/FI включал базовую функциональность сегодняшнего PHP: оформление переменных в стиле Perl ($имя\_переменной для вывода значения), автоматическую обработку форм и встраиваемость в HTML‑текст и многое другое. Новорожденный язык отличался от своего прародителя более простым и ограниченным синтаксисом.

В 1997 году после длительного бета-тестирования вышла вторая версия обработчика, написанного на C – PHP/FI 2.0. Её использовали около 1% (приблизительно 50 тысяч) всех интернет-доменов мира.

PHP 3.0 была первой версией, напоминающей PHP, каким мы знаем его сегодня. В 1997 году два израильских программиста Энди Гутманс (Andi Gutmans) и Зив Сураски (Zeev Suraski), разработчики из израильского института технологий (Technion), переписали код с нуля: разработчики сочли PHP/FI 2.0 непригодным для разработки приложения электронной коммерции, над которым они работали для проекта Университета расположенного в Хайфе, Израиль. Для совместной работы над PHP 3.0 с помощью базы разработчиков PHP/FI 2.0 Энди, Расмус и Зив решили объединиться и объявить PHP 3.0 официальным преемником PHP/FI, разработка же PHP/FI была практически полностью прекращена.

Одной из сильнейших сторон PHP 3.0 была возможность расширения ядра. Впоследствии интерфейс написания расширений привлёк к PHP множество сторонних разработчиков, работающих над своими модулями, что дало PHP возможность работать с огромным количеством баз данных, протоколов, поддерживать большое число API. Фактически, это и был главный ключ к успеху, но стоит добавить, что немаловажным шагом оказалась разработка нового, намного более мощного и полного синтаксиса с поддержкой ООП.

Абсолютно новый язык программирования получил новое имя. Разработчики отказались от дополнения о персональном использовании, которое имелось в аббревиатуре PHP/FI. Язык был назван просто PHP – аббревиатура, содержащая рекурсивный акроним (англ. PHP: Hypertext Preprocessor – «PHP: Препроцессор Гипертекста»).

К концу 1998 года PHP использовался десятками тысяч пользователей. Сотни тысяч веб-сайтов сообщали о том, что они работают с использованием этого языка. В то время PHP 3.0 был установлен приблизительно на 10% веб-серверов Интернета.

PHP 3.0 был официально выпущен в июне 1998 года после 9 месяцев публичного тестирования.

К зиме 1998 года, практически сразу после официального выхода PHP 3.0, Энди Гутманс и Зив Сураски начали переработку ядра PHP. В задачи входило увеличение производительности сложных приложений и улучшение модульности базиса кода PHP. Расширения дали PHP 3.0 возможность успешно работать с набором баз данных и поддерживать большое количество различных API и протоколов, но PHP 3.0 не имел качественной поддержки модулей и приложения работали неэффективно.

Новый движок, названный Zend Engine (от имён создателей, Зива и Энди, также основателей Zend Technologies), успешно справлялся с поставленными задачами и впервые был представлен в середине 1999 года. PHP 4.0, основанный на этом движке и принёсший с собой набор дополнительных функций, официально вышел в мае 2000 года, почти через два года после выхода своего предшественника PHP 3.0. В дополнение к улучшению производительности, PHP 4.0 имел ещё несколько ключевых нововведений, таких как поддержка сессий, буферизация вывода, более безопасные способы обработки вводимой пользователем информации и несколько новых языковых конструкций.

Обновления PHP 4 выпускались только до конца 2007 года. До этого же времени осуществлялась официальная поддержка четвертой версии. Далее до 8 августа 2008 года по мере необходимости будут появляться только критические обновления безопасности. С 9 августа четвертая версия PHP ушла в историю.

Пятая версия PHP была выпущена разработчиками 13 июля 2004 года. Изменения включают обновление ядра Zend (Zend Engine 2), что существенно увеличило эффективность интерпретатора. Введена поддержка языка разметки XML. Полностью переработаны функции ООП, которые стали во многом схожи с моделью, используемой в Java. В частности, введён деструктор, открытые, закрытые и защищённые члены и методы, окончательные члены и методы, интерфейсы и клонирование объектов.

Нововведения, однако, были сделаны с расчётом сохранить наибольшую совместимость с кодом на предыдущих версиях языка. На данный момент самыми стабильными и часто используемыми являются именно версии 5.xx, даже несмотря на то, что уже имеется dev‑версия PHP 6.

Шестая версия PHP находится в стадии разработки с октября 2006 года.

В ней уже сделано множество нововведений, как, например, исключение из ядра регулярных выражений POSIX и «длинных» суперглобальных массивов, удаление директив safe\_mode, php\_magic\_quotes и register\_globals из конфигурационного файла php.ini. Также много внимания уделено поддержке Юникода. Загрузить текущую версию разрабатываемого кода для GNU/Linux/BSD и скомпилированные версии для Microsoft Windows можно на сайте PHP Snapshots.

PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных. Преобразования между скалярными типами зачастую осуществляется неявно без дополнительных усилий (впрочем PHP предоставляет широкие возможности и для явного преобразования типов).

Массивы (array) поддерживают числовые и строковые ключи и являются гетерогенными. Массивы могут содержать значения любых типов, включая другие массивы. Порядок элементов и их ключей сохраняется.

PHP поддерживает широкие объектно-ориентированные возможности, полная поддержка которых была введена в пятой версии языка.

Класс в PHP объявляется с помощью ключевого слова class. Методы и поля класса могут быть общедоступными (public, по умолчанию), защищёнными (protected) и скрытыми (private). PHP поддерживает наследование (родительский класс указывается с помощью ключевого слова extends после имени класса) и интерфейсы (ставятся в соответствие с помощью implements). Разрешается объявление финальных, абстрактных методов и классов. Множественное наследование классов не поддерживается, однако класс может реализовывать несколько интерфейсов.

Код, созданный для более ранних версий языка, зачастую не работает или работает некорректно с более поздними версиями языка. В более поздних версиях исключаются конструкции, методики, функции, применявшиеся ранее. В результате, приложения, созданные несколько лет назад, практически теряют работоспособность для современных версий языка и требуют значительной модификации. Такие изменения обусловлены двумя факторами: устранение несогласованного синтаксиса и устранение конструкций, поощряющих создание небезопасного кода. Справедливости ради, следует отметить, что отсутствие обратной совместимости вообще характерно для современных интерпретируемых языков.

## 1.3. СУБД MYSQL

Программное обеспечение MySQL представляет собой очень быстрый многопоточный, многопользовательский надежный SQL‑сервер баз данных (SQL – язык структурированных запросов). Сервер MySQL предназначен как для критических по задачам производственных систем с большой нагрузкой, так и для встраивания в программное обеспечение массового распространения.

MySQL имеет двойное лицензирование. Это означает, что пользователи могут выбирать, использовать ли ПО MySQL бесплатно по общедоступной лицензии GNU General Public License (GPL) или приобрести одну из стандартных коммерческих лицензий MySQL AB.

MySQL является решением для малых и средних приложений. Входит в LAMP. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Благодаря открытой архитектуре и GPL‑лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года Sun Microsystems приобрела MySQL AB за $1 млрд.

MySQL возникла как попытка применить mSQL к собственным разработкам компании: таблицам, для которых использовались ISAM – подпрограммы низкого уровня. В результате был выработан новый SQL‑интерфейс, но API‑интерфейс остался в наследство от mSQL. Откуда происходит название «MySQL» – доподлинно не известно. Разработчики дают два варианта: либо потому, что практически все наработки компании начинались с префикса My, либо в честь девочки по имени My, дочери Майкла Монти Видениуса, одного из разработчиков системы.

MySQL портирована на большое количество платформ: AIX, BSDi, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, OpenBSD, OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Server 2003 и Windows Vista. Существует также порт MySQL к OpenVMS. Важно отметить, что компания MySQL AB предоставляет для свободной загрузки не только исходные коды СУБД, но и откомпилированные и оптимизированные под конкретные операционные системы готовые исполняемые модули, которые можно загрузить с зеркал, представленных на официальном сайте.

MySQL имеет API для языков C, C++, Java, Perl, PHP, Python, Ruby, Smalltalk и Tcl, библиотеки для языков платформы.NET, а также обеспечивает поддержку для ODBC посредством ODBC‑драйвера MyODBC.

В MySQL 5.0 значительно расширена функциональность, которая ставит MySQL в один ряд с коммерческими СУБД. Если раньше СУБД MySQL обвиняли в недостаточной поддержке стандарта SQL, то с появлением пятой версии этой популярной базы данных, появилась практически полная поддержка стандарта SQL. MySQL 5.0 содержит следующие нововведения: хранимые процедуры и функции; обработчики ошибок; курсоры; триггеры; представления; информационная схема (так называемый системный словарь, содержащий метаданные).

Версия MySQL 6.0 пока находится в стадии альфа-тестирования. Одно из главнейших нововведений версии 6.0 – новый механизм хранения данных Falcon, разработанный компанией MySQL AB, в качестве потенциальной замены для InnoDB компании InnoBase, приобретённой компанией Oracle.

## 1.4. Веб-сервера

Веб-сервер – это сервер, принимающий HTTP‑запросы от клиентов. Обычно в роли клиентов выступают веб-браузеры, и выдающий им HTTP‑ответы, вместе с которыми передаются HTML‑страницы, изображения, файлы, медиа-потоки или другие данные. Веб-серверы являются основой Всемирной паутины.

Фактически веб-сервером называют как программное обеспечение обеспечивающее все вышеперечисленные действия, так и компьютер, на котором это программное обеспечение работает. Клиенты получают доступ к веб-серверу по средствам единого указателя ресурсов – URL‑адресу нужной им веб-страницы или другого ресурса располагаемого на сервере.

Единый указатель ресурсов (англ. URL – Uniform Resource Locator) – это единообразный локатор (определитель местонахождения) веб-ресурса. URL был изобретён Тимом Бернерсом-Ли в 1990 году в стенах Европейского совета по ядерным исследованиям (фр. Conseil Européen pour la Recherche Nucléaire, CERN) в Женеве, Швейцария. URL стал фундаментальной инновацией в Интернете. Изначально URL предназначался для обозначения мест расположения ресурсов (чаще всего файлов) во Всемирной паутине.

Сейчас URL применяется для обозначения адресов почти всех ресурсов Интернета. Стандарт URL закреплён в документе RFC 1738, прежняя версия была определена в RFC 1630. Сейчас URL позиционируется как часть более общей системы идентификации ресурсов URI, сам термин URL постепенно уступает место более широкому термину URI. Cтандарт URL регулируется организацией IETF и её подразделениями.

Новая версия URI была определена в 1998 году в RFC 2396, тогда же слово Universal в названии было заменено на Uniform. В декабре 1999 года RFC 2732 ввёл в спецификацию URI небольшие изменения, обеспечив совместимость с IPv6. В августе 2002 года RFC 3305 анонсировал устаревание термина URL и приоритет URI. Текущая структура и синтаксис URI регулируется стандартом RFC 3986, вышедшим в январе 2005 года. Многие новейшие технологии семантической паутины (например, RDF) базируются на стандарте URI. Сейчас ведущая роль в развитии URI принадлежит Консорциуму Всемирной паутины.

На просторах сети Интернет в данный момент работает около 390 миллионов различных веб-серверов. Среди которых самыми популярными являются: Apache HTTP-Server от компании Apache и IIS от компании Microsoft. График роста количества веб-серверов представлен на рисунке 1.2.

Рисунок 1.2 – динамика внедрения веб-серверов в сети Интернет с Августа 2000 г. по Май 2009 г. по данным компании «Netcraft Ltd».

Появление Apache HTTP Server явилось одним из важнейших средств, стимулировавших развитие всемирной сети Интернет. Apache был первой бесплатной альтернативой веб-серверу от Netscape Communications (в настоящее время известен как Sun Java System Web Server). Приложение доступно для установки на многих платформах, таких как: Unix, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows и т.д. Apache распространяется как бесплатное программное обеспечение с открытым исходным кодом, что позволяет множеству разработчиков со всего мира участвовать в процессе доработки и выпускать дополнительные надстройки, выполняющие специфические, нужные для определенных целей, функции.

Авторы проекта всегда говорили о том, что его имя было выбрано не случайно. На это были две причины. Первой причиной стала дань уважения к коренным жителям Америки – индейскому племени Апаче, чья выносливость и неповторимое боевое мастерство известны во всем мире. Вторая причина более предсказуема, дело в том, что изначально, проект разрабатывался как серия многочисленных «патчей» (заплаток) к уже существующему веб-серверу NCSA httpd.

Первая версия была выпущена Робертом МакКулом в 1994 году. После ее выхода сформировалась команда, позже ставшая знаменитой Apache Group. Вторая версия заметно отличалась от предшествующей. Она была выпущена только в 2002 году. Имея модульную структуру, новый API для разработчиков, поддержку IPv6 и улучшенную поддержку не-Unix платформ (например Windows) она стала самым используемым http‑сервером в сети Интернет. На данный момент последним релизом является версия 2.2.8, выпущенная 19 января 2008 года.

Веб-сервер Apache имеет модульную структуру ядра, что помогает разработчику, определившись с целями проекта, использовать именно те средства разработки, которые он планирует, отключив все остальные возможности. Данная архитектура ядра делает программный продукт более безопасным и быстрым. Веб-сервер поддерживает различные модули интерпретаторов, например: perl, python, php, tcl. Кроме этого веб-разработчик может использовать модули безопасности и криптографии. В Apache встроена поддержка аутентификации, реализован протокол шифрования SSL, имеется прокси-модуль и модуль создающий отчеты. Кроме этого в возможности веб-сервера входит подключение внешнего модуля сжатия gzip.

Удобная конфигурация позволяет администратору использовать один веб-сервер для обслуживания нескольких узлов одновременно. В настоящее время Apache входит в комплекты поставки различных решений для производства, Например в Novell NetWare он является веб-сервером по умолчанию, как и в IBM WebSpere. Для организации собственных решений сервер используют Oracle и Borland. Apache также используют и для реализации других задач различной сложности. Одним из примеров является обмен файлов. Пользователь, на своем персональном компьютере устанавливает копию программы и помещая документы в определенную папку, делает их открытыми для всех остальных пользователей зашедших на страницу созданную его веб-сервером.

В апреле 1996 г. Apache стал самым популярным HTTP‑сервером во Всемирной сети. Но к Ноябрю 2006 г. доля веб-серверов работающих на ПО от Apache Group значительно снизилась, почти половина всех веб-ресурсов перешла на сервер Microsoft Internet Information Services или сторонние разработки. Анализируя данные за Май 2008 г. можно сделать вывод о том, что доля узлов работающих на веб-серверах Apache составляет 52% от числа всех ресурсов сети.

Вторым по полярности является Microsoft Internet Information Services. Пользуясь данными, собранными компанией Netcarft можно сделать вывод – доля веб-ресурсов, обслуживаемых веб-серверами IIS, составляет 35% от общего количества узлов.

Выпуск IIS 7.0 совпадает с десятилетней годовщиной выпуска первой версии IIS в составе Windows NT® 4.0 в 2001 г. Четыре года спустя, IIS 5.0 стал самым распространенным сервером в Интернете, хотя спустя несколько месяцев он пал жертвой печально известных червей Code Red и Nimda. IIS 6.0, выпущенный в Windows Server® 2003, был серьезной переработкой сервера, в которой все усилия были сосредоточены на повышении безопасности, надежности и производительности. С тех пор IIS 6.0 доказал, что он является в высшей степени защищенным веб-сервером, обеспечивающим высокую надежность и безопасность. После его выпуска был издан только один важный бюллетень по безопасности исправляющий ошибку, которой невозможно воспользоваться в удаленном режиме.

IIS 7.0 должен был унаследовать от базового кода IIS 6.0 скорость, надежность и безопасность, и преобразовать его в высшей степени расширяемую и управляемую платформу веб-сервера, достаточно мощную для выполнения современных веб-приложений. В результате создан наиболее перспективный веб-сервер Майкрософт, содержащий самое большое в истории IIS количество архитектурных усовершенствований.

В основе выпуска IIS 7.0 лежит полностью модульный веб-сервер, включающий более 40 компонентов, которые можно объединять в компактные веб-серверы, оптимизированные для необходимой роли в топологии приложения. Эти компоненты создаются на основе нового слоя расширяемости, что позволяет разработчикам расширять или замещать практически любую функцию сервера в машинном коде или с помощью Microsoft®.NET Framework.

IIS 7.0 предлагает расширяемость компонентов выполнения, управления и рабочих компонентов, облегчая создание комплексных решений в соответствии с конкретными потребностями. Диспетчер служб IIS 7 представлен на рисунке 1.3.

Рисунок 1.3 – Диспетчер служб IIS 7

В платформе IIS 7.0 по сравнению с IIS 6 решаются многие проблемы, связанные с управляемостью и эксплуатацией сервера. Она обладает принципиально новой системой настройки, обеспечивающей полностью делегированное управление узлами и, в конечном итоге, делающей реальностью развертывание веб-приложений с использованием xcopy.

Новые интерфейсы API для управления и диагностические компоненты делают процедуры развертывания, администрирования и устранения неполадок сервера значительно проще и удобнее, чем когда-либо прежде.

Веб-сервер IIS поддерживает несколько различных технологий создания веб-приложений:

ASP.NET – для систем Windows это основное, на сегодняшний день, средство создания веб-приложений и веб-служб. Поддержка ASP.NET встроена в IIS 6.0; для более ранних версий необходимо отдельно загрузить и установить.NET Framework.

ASP – устаревшая технология создания динамических веб-страниц на основе сценариев. Входит в поставку IIS начиная с версии 3.0.

CGI – стандартная технология создания динамических веб-страниц.

ISAPI – для Windows систем это наиболее мощная технология, предоставляющая полный доступ ко всем возможностям IIS.

При помощи CGI и ISAPI к веб-серверу IIS могут подключаться сторонние средства поддержки веб-приложений, например, PHP и Perl.

Одним из интересных и перспективных нововведений в IIS 7.0 является пакет IIS Media Pack. Два дополнительных бесплатных модуля позволят превратить веб-сервер в современный инструмент медиа-вещания. Новые технологии Microsoft позволяют оптимизировать и грамотно управлять цифровым потоком медиа-данных. Сервер позволяет производить вещание данных в форматах: ASF, AVI, FLV, M4V, MOV, MP3, MP4, RM, RMVB, WMA, WMV.

Еще одним нововведением является встроенная поддержка новой технологии Silverlight, Silverlight 2 и Silverlight 3. Это новая технология представления данных в Интернете, предназначенной для запуска на различных платформах. Она позволяет создавать насыщенные, визуально привлекательные веб-страницы, работающие в различных обозревателях, устройствах и настольных операционных системах (например Apple Macintosh).

## 1.5. Принципы построения клиент / серверных систем

Для разработки клиент / серверных систем имеется два подхода:

* построение систем на основе двухзвенной архитектуры;
* построение систем на основе трехзвенной архитектуры.

Двухзвенная архитектура состоит из клиентской и серверной части. Как правило, серверная часть представляет собой сервер БД, на котором расположены общие данные. А клиентская часть представляет приложение, которое связывается с сервером БД, осуществляет к нему запросы и получает ответы. Такие системы используются в локальных сетях, т. к. нет затруднений с установкой клиентской части. Также системы с такой архитектурой более безопасны, т. к. могут использовать собственные протоколы передачи данных, не известные злоумышленникам. Поэтому многие крупные компании, которые располагаются не в едином месте и для соединения подразделений используют глобальную сеть Интернет, выбирают именно такую архитектуру построения клиент / серверных систем.

При разработке информационных систем, рассчитанных на широкую аудиторию, возникают проблемы с использованием двухзвенной архитектуры. Во-первых, пользователю необходимо иметь в наличии клиентскую часть, а, во-вторых, у неопытного пользователя, могут возникнуть проблемы с конфигурированием такой системы. Поэтому в последнее время, более часто разрабатывают приложения на базе трехзвенной архитектуры.

Трезвенная архитектура также состоит из двух частей: клиента и сервера. Но серверная часть в этой архитектуре представляет собой сервер приложений и сервер БД. А в качестве клиента выступает web‑браузер. Такая система очень проста для пользователя. Ему необходимо знать только адрес сервера приложения и наличие web‑браузера на рабочем компьютере. Все данные представляются в виде html‑разметки с использование графики (jpeg, gif, flash), каскадных слоев CSS и JavaScript. Передача запросов от клиента к серверу приложений происходит по средствам CGI‑интерфейса. Сервер приложений общается с сервером БД, используя другой интерфейс, зависящий от того, на основе каких средств строится конкретная информационная система. Недостатками такой архитектуры является использование общеизвестных протоколов передачи данных. Злоумышленник может осуществить взлом системы, если она будет не достаточно хорошо проверять поступившие запросы от клиента. Трехзвенная архитектура показана на рисунке 1.4.

Рисунок 1.4 – Трехзвенная архитектура.

При разработке клиент / серверных приложений необходимо учитывать:

* на каких пользователей будет рассчитана данная информационная система;
* какие требования предъявляются к безопасности.

Если информационная система должна быть общедоступной и рассчитана на широкую аудиторию, то необходимо использовать трехзвенную архитектуру.

Если информационная система используется внутри предприятия, доступ имеют к ней ограниченные пользователи и требуется создать максимально безопасную и защищенную систему, то следует отдать предпочтение двухзвенной архитектуре.

## 1.6. Системы управления содержимым

В последние годы стали популярны системы управления контентом (или как их еще принято называть CMS – от англ. «Content Managment System»). С помощью подобных систем опытный пользователь может при необходимости легко создать базовый веб-проект (без дополнительных трудозатрат), а веб-интегратор получает возможность развернуть масштабный проект в приемлемые сроки (модульность подобных систем расширяет их функционал).

Во времена, когда системы управления контентом практически не применялись – разработка очередного веб-проекта была сопряжена с созданием кода разметки каждой страницы, либо с тяжелым программированием и интеграцией графического оформления в каждую страницу. Процесс занимал огромное количество времени, а результат выполнял ограниченное количество функций и тяжело модернизировался.

Современные системы управления контентом снимают необходимость постоянного программирования. Достаточно выбрать готовый компонент, из тысячи ранее созданных и протестированных. Интеграция в систему не займет много времени, так как все дополнения делаются по единому стандарту.

Для того, чтобы оценить достоинства систем управления контентом следует заглянуть в прошлое и посмотреть, как и с помощью каких инструментальных средств создавались веб-ресурсы до появления подобных систем, и как они создаются сегодня.

Начало развитие веб-среды можно обозначить серединой 90‑ых г. Этот период характеризовался относительно невысоким уровнем развития веб технологий, а многие динамические веб-проекты (т.е. способные на интерактивное взаимодействие) могли создаваться только программистами.

Вся динамика реализовывалась через CGI и другие сложные технологии, поэтому обычной пользователь, создающий собственный проект, мог рассчитывать только на статичный базовый проект масштаба сайта и не более того. Такой сайт обычно представлял собой набор статичных html‑страниц, подготовленных в WYSIWYG-редакторах, которые начали появляться примерно с середины 1995 года. После набора, страницы объединялись ссылками (для осуществления возможности межстраничных переходов) и размещались на сервере. Вся работа по обновлению информации и проверке работоспособности проекта перекладывалась на его автора. Так, например, если требовалось изменить ссылку на странице, автор должен был найти эту страницу среди остальных, затем внести в нее изменения и снова загрузить страницу на сервер. Если же требовалось не просто изменить существующую страницу, а добавить новую, то приходилось также решать вопросы, связанные с логическим и физическим внедрением последней в весь проект. И если объем страниц возрастал, то «справляться» с ними становилось еще сложнее. А изменять и расширять динамические проекты было еще более проблематично.

Систем автоматизации всех этих процессов на тот момент практически не было, а точнее не было доступных и легких систем, не было выбора между открытыми и коммерческими системами, а популярность немногочисленных коммерческих систем была такова, что об их существовании практически никто не знал. Однако такое положение длилось относительно недолго, и в дополнение к статичным html‑страницами CGI программированию появились более «дружественные» технологии – Asp (конец 1996 г.), ColdFusion (июнь 1995 г.), а позже и PHP (2‑ая версия вышла в 1997 г.). Новые технологии позволили совместить разметку html‑страниц и несложный программный код, сделав тем самым пассивные html‑страницы активными. Активность последних позволила легко организовать интерактивное взаимодействие с пользователями, ведь каждый раз при обращении к одной и той же активной странице пользователь мог получать новые данные. В то же время стали создаваться активные страницы для автоматизации определенных действий, например, процесса загрузки файлов на сервер или же процесса создания новых страниц. Такие полезные страницы постепенно собирались во вспомогательные пакеты. Подобные пакеты применялись в типовых задачах, однако их возможностей не всегда хватало и многие клиенты предпочитали заказывать индивидуальные системы управления под свой собственный проект. Эти заказные системы изначально не были универсальными – когда требовалось изменять либо расширить их функциональность, заказчику приходилось снова обращаться к разработчикам. Вскоре и сами разработчики пришли к решению о необходимости создания универсальных систем. Таким образом, появились первые универсальные коммерческие системы управления.

Утвердилась и аббревиатура CMS, которая была заимствована и русскоязычными пользователями. Позже появились и открытые системы управления. Дальнейшая эволюция этих систем привела всю область к такому уровню развития, что нынешнее словосочетание «система управления содержанием» уже не совсем актуально, т. к. функциональность современных систем значительно расширилась, теперь они способны работать с темами, компонентами, а также управлять другими элементами. Современные системы уже можно смело именовать «системами управления веб-проектом (класса n)».

Сейчас системы управления это не только удобная оболочка-менеджер для пользователя, но и мощный инструмент для веб-разработчика (последнее справедливо далеко не для всех систем). Благодаря таким системам, все реже возникает необходимость в разработке веб-проектов «с нуля» – подготовленному пользователю или интегратору достаточно просто выбрать, установить и настроить существующую систему, чтобы быстро и без дополнительных финансовых затрат получить приемлемый результат профессионального уровня.

К сожалению, некоторая часть существующих веб-сайтов создана без систем управления. Это так называемые «пустышки».

Многие клиенты попросту не знают о существовании подобных систем, а разработчики не утруждают себя их внедрением. Если у заказчика веб-проекта возникнет необходимость во внесении корректировок или изменений в проект, то ему придется самостоятельно выполнять сложные манипуляции с гипертекстом и программным кодом, либо потребуется обратиться к разработчикам проекта и за дополнительную оплату решать вопросы с которыми он и сам бы мог справиться, имея систему управления (а во многих случаях разработчики даже заинтересованы в подобной зависимости собственных клиентов).

По объективным оценкам на данный момент существует пока немалая часть веб-проектов без систем управления, которые в ближайшее время будут переводиться на подобные системы.

Теперь следует рассмотреть, как системы управления представляются различным пользователям. Фактически к пользователям систем управления можно причислить всех тех, кто хоть как-то с ними взаимодействует. К примеру, пользователями обычно становятся следующие лица:

* будущие и настоящие посетители разрабатываемого / разработанного веб-проекта (гости, зарегистрированные и незарегистрированные пользователи)
* контент-менеджеры, модераторы, редакторы, администраторы (владельцы);
* программисты и интеграторы (пользователи cms «как продукта»).

Посетители – это самая многочисленная и самая значимая группа пользователей. Без них любой проект перестал бы развиваться (а возможно и вообще существовать), поэтому за их внимание и ведется столь жесткая борьба. Посетитель, он никогда не вникает в технические подробности – ему абсолютно безразлично, на базе какой системы управления функционирует разрабатываемый проект, ему важен результат – информация и способ ее получения. Систему управления посетитель не видит в таком виде, в каком ее видят доверенные участники вашего проекта, вся система для него – это страницы с материалами, навигационные элементы, формы обратной связи, подсказки и т.д., но не более того. Посетитель – это типичный потребитель, и чем проще и быстрее интересная информация «извлекается» из вашего проекта, тем меньше он задумывается о других, альтернативных проектах. Поэтому недостаточно создать удобный и хорошо настроенный проект, необходимо также позаботиться и о его содержании – оно должно быть интересным и актуальным, так вы сможете привлечь больше посетителей.

Контент-менеджеры, модераторы, редакторы, администраторы – это доверенные лица, которым делегирован набор дополнительных прав. Пользователям такого уровня система управления представляется в расширенном виде – к интерфейсу системы добавляются дополнительные элементы управления или целые административные разделы. В таких разделах администратор может легко и быстро добавлять например новых пользователей, а модератор может вносить правки в оставленные посетителями сообщения и делать другие разрешенные администратором действия.

Для лучшего понимания предметной области, а также установления порядка, была создана такая методика как классификация. Классификация – это разделение общего множества на частные по выбранным критериям, т.е. попросту деление на группы по общим признакам. Классификация также требуется и в области систем управления, где она должна сделать четкие разграничения между системами.

На данном этапе будут рассмотрены два наиболее очевидных вида классификации:

* классификация по «степени открытости»;
* классификация по «разработчику системы».

**а) классификация по «степени открытости»**

Многие пользователи уже успели попробовать такие продукты как OpenOffice, Firefox, Linux, а также другие известные программы и системы, которые распространяются не просто бесплатно, но и с открытыми исходными кодами (в рамках движений OpenSource, GNU, FSF и других). Последнее означает, что каждый желающий может вносить изменения в исходный код таких программ и даже распространять их в модифицированном виде. Однако открытые проекты – это не просто программное решение и набор все дозволяющих лицензий – это еще и социальное явление, базирующееся на принципах взаимопомощи, а также движение, объединяющее своих участников по идейным интересам.

Рассмотрим наиболее очевидные преимущества и возможности открытых продуктов:

1. Главным преимуществом открытых продуктов является их доступность – открытый продукт можно не только свободно получать, но и быстро обновлять, ведь новые версии таких продуктов выходят с завидной регулярностью, а исправление найденных ошибок не выливается в очередное ожидание заплаток или сервис паков, как это обычно происходит с коммерческими решениями. Открытые продукты предоставляют свободу выбора и действий, а также дают возможность с минимальными издержками осуществлять миграцию с одного решения на другое, если существующее не удовлетворяет требованиям заказчика.

2. Открытые продукты благодаря заимствованиям стороннего открытого кода, а также взаимоподдержке развиваются порой быстрей своих коммерческих конкурентов. Благодаря свежим идеям и неустанной поддержке со стороны сообщества, открытые продукты способны не только конкурировать с коммерческими решениями, но и занимать доминирующие позиции на рынке. Примечательно, что сейчас уже и коммерческие разработчики присоединились к подобной инициативе – многие из них уже включают в состав своих продуктов открытые решения.

3. Любой желающий, от простого пользователя до программиста, может стать участником открытого проекта, ведь большим интернациональным открытым проектам всегда требуются локализаторы (переводчики), тестировщики и другие специалисты.

4. Программист / организация, выбирая за основу открытые решения, избавляет себя от необходимости подстраивания под чужой корпоративный «черный ящик». Также из открытого продукта можно позаимствовать новые и / или интересные идеи, что не только не запрещается, а даже приветствуется. Открытые проекты и продукты практически всегда ориентируются на стандарты, а не идут вразрез с ними. Это значительно упрощает процесс доработки продукта, а также гарантирует совместимость с другими продуктами, поддерживающими стандарты.

5. Любой специалист в своей области, получая исходные тексты открытых программ, имеет возможность провести анализ продукта и вынести свое заключение относительно его надежности, безопасности, совместимости и т.д.

6. Открытый продукт можно «пересобрать» с учетом своих собственных требований и тем самым получить максимальный результат.

Возвращаясь к классификации по «степени открытости» все продукты можно разделить на две группы. В первую группу следует отнести все открытые продукты, во вторую – остальные (к остальным относятся коммерческие и бесплатно распространяемые продукты).

**б) классификация по «разработчику системы»**

Разработчик системы, как правило, является автором, и сопроводителем системы. В роли разработчика может выступать как открытое сообщество со своими идейными лидерами, так и целая коммерческая организация с серьезным штатом сотрудников. Для удобства отнесем проекты сообществ к первой группе, а проекты организаций, соответственно, ко второй. Для того чтобы понять, чьи продукты все же лучше, следует рассмотреть типичные сценарии появления новых cms (хотя это актуально и не только для cms).

Начнем с систем управления первой группы. В мире существует довольно много свободных профессионалов (высококлассных программистов, дизайнеров и т.д.), ценящих свой труд и свободу. Они никогда не соглашаются на роль эксплуатируемого звена, они предпочитают самостоятельно находить заказы и работать без посредников (а для больших проектов даже собирать целые команды) и, как результат, получать за свой труд достойное вознаграждение. В своем большинстве это идейные лидеры, которые не желают мириться с текущим положением вещей. Так или иначе, но в своей практике им приходится применять различные «решения» (готовые идеи и продукты). Со временем они понимают, что существующие решения обладают рядом недостатков (ограниченная функциональность, закрытость и т.д.) либо малопригодны в среде новых технологий, и тогда они берут на себя смелость создать нечто новое и более доступное. Так появляется новая идея. После воплощения задуманного, на свет появляется и «решение», пока еще, возможно и сырое, но уже перспективное. Далее автор принимает решение сделать все наработки общедоступными – так открывается новый проект. Это приводит к тому, что в скором времени, у проекта, появляются и свои последователи – образуется сообщество. Именно благодаря сообществу и его мощной поддержке в дальнейшем, все последующие «решения» будут эволюционировать (идея при этом останется неизменной). Как итог: уже через несколько лет полностью готовое и стабильное «решение» начнет вытеснять коммерческие продукты, а новый проект превратится в достояние сообщества.

Ко второй группе были отнесены коммерческие системы. Системы такого класса разрабатываются преимущественно веб-студиями и крупными интеграторами.

Следует также заметить, что веб студии бывают различных типов – на роль веб-студий могут претендовать как команды из 2–3 человек, так и вполне серьезные организации. Причем, в первом случае, такая веб-студия может представлять собой самый примитивный «сайто-строительный» полигон и не более того. Теперь вернемся к коммерческим CMS. Многие из ныне существующих коммерческих систем, т.е. систем второго потока, «зародились» в периоде между 1999 г. и 2003 г. – именно тогда начали стремительно появляться все популярные на данный момент веб-студии, которые в первые годы своего существования и стали создавать собственные системы управления. Схема разработки и поддержки систем управления контентом представлена на рисунке 1.5.

Рисунок 1.5 – Различия в поддержке между открытыми и закрытыми системами управления контентом

Современная CMS система должна позволять, не прибегая к дополнительному программированию, выполнять: редактировать содержимое страниц, включая добавление / удаление графики; добавление новых страниц; изменение структуры сайта и различных мета данных; настройку регистрационных форм; управление опросами, голосованиями и форумами; вывод статистики посещений; распределение прав по управлению сайтом среди пользователей.

## 1.7. Система управления содержимым Joomla CMS

Joomla (произносится джу́мла) — система управления содержимым, написанная на языке PHP, использующая в качестве хранилища базу данных MySQL. Является свободным программным обеспечением, распространяемым под лицензией GNU GPL. Название «Joomla» фонетически идентично слову «Jumla», которое в переводе с языка суахили означает «все вместе» или «единое целое», что отражает подход разработчиков и сообщества к развитию системы.

Joomla CMS включает в себя различные инструменты для изготовления веб-сайта. Важной особенностью системы является минимальный набор инструментов при начальной установке, который дополняется по мере необходимости. Это снижает загромождение административной панели ненужными элементами, а также снижает нагрузку на сервер и экономит место на хостинге.

Joomla CMS позволяет отображать интерфейс фронтальной и административной части на любом языке. Каталог расширений содержит множество языковых пакетов, которые устанавливаются штатными средствами администрирования. Доступны пакеты русского, украинского и ещё некоторых языков стран СНГ.

### 1.7.1. История возникновения Joomla CMS

Система управления содержимым Joomla! является ответвлением широко известной CMS Mambo. Команда независимых разработчиков отделилась от проекта Mambo по причине несогласия в экономической политике. 16 сентября 2005 года в свет вышла первая версия Joomla, являющаяся по сути переименованной Mambo 4.5.2.3 и включающая в себя исправления найденных на тот момент ошибок и уязвимостей.

В настоящее время актуальна версия 1.5. В разработке находится версия 1.6. Версия 1.0 на сегодняшний день считается устаревшей, её официальная поддержка прекращена 1 июля 2009 года.

К лету 2009 года по числу ежедневных скачиваний Joomla обогнала Wordpress и заняла первое место со значительным отрывом от других подобных систем.

В 2006 году Joomla заняла первое место в категории Best Linux/Open Source Project на выставке UK LinuxWorld 2006.

Так же стоит отметить награды Joomla в популярном издании Packt Publishing:

2006 г. - признана лучшей бесплатной системой управления сайтами;

2007 г. - признана лучшей бесплатной CMS основанной на PHP;

2008 г. - разделила 2-е место с CMS Made Simple в номинации лучшая бесплатная CMS основанная на PHP.

### 1.7.2. Базовые (стандартные) возможности

Благодаря стандартному функционалу система подходит как для небольшого, так и для крупного корпоративного сайта или информационного проекта. При необходимости, расширить стандартный функционал системы можно с помощью более чем 3000 созданных расширений. Стандартные возможности системы:

* Возможность создавать неограниченное количество страниц
* Четкая организация структуры сайта
* Для каждой динамической страницы можно создать свое описание и ключевые слова в целях повышения рейтинга в поисковых системах
* Начало и окончание публикации любых материалов можно запрограммировать по календарю
* Возможность ограничить доступ к определенным разделам сайта только для зарегистрированных пользователей
* Секции новостей, продукции или услуг легко редактируемы и управляемы
* Полностью настраиваемые схемы расположения элементов по 5 областям, что позволяет сделать оригинальный дизайн
* Опросы и голосования для эффективной обратной связи
* Различные компоненты - такие как последние новости, счетчик посещений, подробная статистика посещений, гостевая книга, форум и т.д.
* Возможность создания не одной, а нескольких форм обратной связи для каждого контакта.
* Генератор показа случайной новости.
* Компонент приёма от удалённых авторов новостей, статей и ссылок.
* Иерархия объектов - количество секций, разделов, подразделов и страниц, зависит от вашего желания.
* Менеджер рассылки новостей.
* Менеджер архива для старых материалов сайта
* Возможность распечатать или отправить другу на e-mail любую статью с сайта.
* Выбор из 3 визуальных редакторов, что упрощает редактирование материалов до уровня редактирования текста в программе Word.
* Предварительный просмотр перед окончательным размещением.
* Возможность легкой смены дизайна.
* Экономное использование места на сервере за счет использование базы данных MYSQL.
* Возможность использования адресов страниц адаптированных для лучшей индексации всеми поисковыми системами.

### 1.7.3. Технические требования

Минимальные технические требования Joomla CMS к хостингу:

* Apache-вepcии 1.13 (или выше, желательно 2.x);
* РНР-версии 4.3.1 (или выше, желательно 5.x);
* MySQL-версии 3.23 (или выше, желательно 5.x).

Что же касается места на диске, то для установки системы понадобится как минимум 50 Мб, но лучше использовать хостинг с местом от 100 Мб — все зависит от того, какая информация планируется к размещению на сайте.

### 1.7.4. Интерфейс прикладного программирования Joomla

Интерфейс прикладного программирования (англ. Application Programming Interface, API) — набор готовых классов, функций, структур и констант, предоставляемых системой для использования во внешних программных продуктах.

В Joomla 1.5 разработчиками реализован собственный API, работающий по принципу Model-View-Controller.

Model-View-Controller (в дальнейшем - MVC) является схемой-паттерном для создания гибких и легко расширяемых приложений с пользовательским интерфейсом. Эта схема была разработана создателями Smalltalk-80 ещё в 1980 г.

В компоненте, построенном с помощью технологии MVC, используются три основные части:

Модель (Model) - это часть компонента, которая оперирует данными приложения. В общем случае модель содержит методы добавления, удаления, и обновления информации. Таким образом, если программа сохраняет данные в файлы, а нужно сохранять их в БД, то для этого нужно изменить только модель, не трогая остальной код.

Вид (View) – это часть компонента, которая берет данные из модели и выводит их в пользовательский интерфейс. Для простых веб-приложений, Вид генерирует html-страницу, с входными данными. Т.е. вид берет данные из модели, и вносит их в шаблон, который в конечном итоге и видит пользователь. Вид не изменяет данные, он только отображает их из модели.

Контроллер (Controller) - отвечает за действия пользователя. По сути, действие пользователя - это запрос страницы. Контроллер в зависимости от запроса, вызывает модель, для управления данными, и управляет переходом Модели на Вид. Таким образом, контроллер не отображает данные, а лишь вызывает модель, которая изменяет данные, а затем передает модель в вид, который отображается на экране. Схема принципа работы технологии MVC представлена на рисунке 1.6.

Рис. 1.6 – Принцип работы технологии Model-View-Controller

Такое разделение удобно для групповой работы над проектом, дисциплинирует и дает отличное качество получаемого кода.

MVC позволяет избежать зависимости модели от интерфейса, появилась дополнительная возможность к повторному использованию кода без его изменения.

В Joomla 1.5, технология MVC использует три класса: JModel , JView и JController .

# Глава II. Разработка компонента для Joomla 1.5

В этой главе будут рассмотрены основные принципы проектирования и реализации компонентов для Joomla CMS версии 1.5 на примере разработки простого компонента “Hello”, использующего основные методы и функции внутреннего интерфейса программирования системы.

## 2.1. Общая структура компонента

 Каждый компонент Joomla, реализованный по технологии MVC, имеет структуру расположения файлов, представленную на рисунке 2.1.

Рис. 2.1 – Начальная структура компонента Joomla 1.5

Директория «admin» содержит файлы по настройке и управлению компонента из административной панели Joomla CMS.

Директория «site» содержит файлы для лицевой части компонента, видимой посетителями сайта. Рассмотрим наиболее важные файлы этой директории:

* /site/{componentname}.php

Точка входа (или точка отправления) для лицевой части компонента. Является обязательным исполнительным файлом.

* /site/controller.php

Является основным управлением компонента, либо подключает дополнительные файлы, в зависимости от переданных внешних данных.

* /site/views/{viewname}/view.html.php

Обработчик полученных данных. После обработки передает данные для вывода на сайте.

* /site/views/{viewname}/tmpl

Шаблон вывода полученных данных от обработчика.

* /site/models

Директория содержит модели, используемые компонентов в процессе работы.

* /site/controllers

Директория содержит расширяющие контроллеры для основного, либо набор необходимых компоненту контроллеров.

Файловая структура директории «admin» в точности повторяет структуру лицевой части компонента, описанной выше. Необходимо учитывать, что директории «site» и «admin» не пересекаются в процессе работы компоненты. По большому счету, эти две директории можно рассматривать как два различных подкомпонента одного большого компонента.

Joomla CMS обрабатывает компонент в следующей последовательности:

* Joomla интерпретирует полученные значения в URL.
* Ищет составляющую таблицу компонента в базе данных.
* Затем определяет директорию компонента.
* В этой директории запускает исполнительный файл компонента. В данном случае – это файл hello.php.
* Исполняет этот файл.

### 2.2. Создание точки входа компонента (файл hello.php)

Joomla всегда обрабатывает ссылку в корневом файле index.php для страниц лицевой части сайта или administrator/index.php для страниц административного интерфейса системы. Функция обработки URL загружает требуемый компонент, основанный на значении 'option' в URL (используется метод передачи данных GET) или переданных данных методом POST.

Для разрабатываемого компонента, URL будет выглядеть следующим образом:

*index.php? option=com\_hello&view=hello*

Эта ссылка запускает выполнение файла, являющего точкой входа в компонент: components/com\_hello/hello.php.

Ниже приведен листинг файла hello.php:

*<?php*

*// Защита от прямого обращения к скрипту*

*defined( '\_JEXEC' ) or die( 'Restricted access' );*

*// Подключение файла контроллера.*

*require\_once( JPATH\_COMPONENT.DS.'controller.php' );*

*// Проверка на дополнительные контроллеры*

*if($controller = JRequest::getVar( 'controller' )) {*

 *require\_once( JPATH\_COMPONENT.DS.'controllers'.DS.$controller.'.php' );*

*}*

*// Создание класса контроллера компонента*

*$classname = 'HelloController'.$controller;*

*$controller = new $classname( );*

*// Выполнить задачу запроса*

*$controller->execute( JRequest::getVar( 'task' ) );*

*// Переадресация*

*$controller->redirect();*

Рассмотрим код более подробно:

1) *defined( '\_JEXEC' ) or die( 'Restricted access' );*

Первая строка предотвращает прямой доступ к исполнительному файлу в целях общей безопасности сайта:

2) *require\_once( JPATH\_COMPONENT.DS.'controller.php' );*

Следующая строка подключает основной контроллер компонента.

JPATH\_COMPONENT - абсолютный путь к текущему компоненту, в данном случае - это components/com\_hello.

DS - автоматический выбор слеша (разделителя директорий) '\' или '/', в зависимости от ОС. Таким образом, разработчик не должен волноваться о том, на какой системе установлена Joomla.

3) *if($controller = JRequest::getVar( 'controller' ))*

Проверка на наличие вспомогательных контроллеров для компонента. Если таковые присутствуют, то подключаются из директории /controllers компонента. В рассматриваемом случае контроллер всего один.

4) *$classname = 'HelloController'.$controller;*

 *$controller = new $classname( );*

Создание класса основного контроллера компонента.

5) *$controller->execute( JRequest::getVar( 'task' ) );*

Получение контроллером текущей задачи из полученного URL вида: index.php?option=com\_hello&task=Задача. Если значение task в переданном URL пустое, по умолчанию используется задача «display», предназначенная для вывода данных. При использовании задачи сохранения или удаления страницы контроллер компонента вызывает перенаправление на другую страницу с помощью строки *$controller->redirect();*

Фактически, точка входа компонента передает управлению контроллеру компонента для выполнения задачи, которая была определена в полученном запросе.

### 2.3. Создание контроллера компонента (файл controller.php)

Поскольку разрабатываемый компонент предназначен лишь для представления основ реализации компонентов под Joomla, контроллер этого компонента – очень простой. В некоторых больших компонентах число строк контроллера достигает до нескольких тысяч. В контроллере компонента «Hello» используется только один метод - display(). Большая часть необходимых функций уже встроена в класс JController, так что все, что требуется, это вызвать метод JController::display() .

Ниже представлен листинг файла controller.php:

*<?php*

*defined( '\_JEXEC' ) or die( 'Restricted access' );*

*jimport('joomla.application.component.controller');*

 *class HelloController extends JController*

*{*

*// метод вывода данных компонента*

*function display()*

*{*

*parent::display();*

*}*

*}*

display () - это первый по приоритету метод, он вызывает родительский конструктор. JController :: display () метод определяет название шаблонов вывода запроса и загрузки, которые определяют и устанавливают выбор ветки в дереве задач.

### 2.4. Создание модели компонента (файл /models/hello.php)

Модель - часть компонента, которая передает данные Виду компонента по его запросу, отправленному через Контроллер компонента. Такой метод часто освобождает от рутинной работы и от хаоса в коде, предоставляет возможность управлять данными удобным способом в дополнение к коду, который посылает запрос данных из Модели.

В Joomla 1.5, Модель содержит классы функций: добавить, удалить и модернизировать информацию в таблицах базы данных. Также содержит методы восстановления списка таблиц базы данных.

Наименование моделей строятся следующим образом: название компонента, model, название модели.

В рассматриваемом случае это название приобретет вид HelloModelHello.

Другими словами, основная структура доступа к данным должна быть кратко описана в модели.

Таким способом, если обработку запроса данных необходимо изменить, Модель компонента будет единственным элементом, в который будут внемены изменения , не затрагивая код Вида или Контроллера компонента.

При переносе точки запроса данных в общем алгоритме компонента, изменения будут внесены только в код Вида компонента.

В модели разрабатываемого компонента, будет использовано моделирование события компонента "Hello", которое сгенерирует приветствие. Таким образом, в компоненте будет содержаться один запрос к модели getGreeting () и БД, который возвратит строку "Hello, World!".

Листинг файла /models/hello.php:

*<?php*

*// Защита от прямого обращения к скрипту*

*defined( '\_JEXEC' ) or die( 'Restricted access' );*

*jimport( 'joomla.application.component.model' );*

*class HelloModelHello extends JModel*

*{*

 *function getGreeting()*

 *{*

 *$db =& JFactory::getDBO();*

 *$query = 'SELECT greeting FROM #\_\_hello';*

 *$db->setQuery( $query );*

 *$greeting = $db->loadResult();*

 *return $greeting;*

 *}*

*}*

Функция jimport используется, чтобы загрузить файлы из API Joomla, которые требуются для компонента. В данном случае загрузится файл /libraries/joomla/application/component/model.php. Точки используются как разделители директорий. Все файлы загружаются относительно директории библиотек - libraries.

В рассматриваемом случае подключается файл model.php из библиотеки системы. Это специальный файл, который описывает класс JModel, он необходим, потому что модель использует этот класс.

Для подключения к базе данных сайта используется статический класс JFactory и метод getDBO. Данный метод уже содержит все параметры подключения, поэтому ничего дополнительного указывать не нужно.

$db->loadResult() выполняет запрос из соответствующей таблицы БД и возвращает полученное значение в заданную переменную, в данном случае это переменная $greeting.

### 2.5. Создание вида компонента (файл /views/hello/view.html.php)

Задача этого файла разрабатываемого компонента очень проста – получить данные из модели компонента, которые будут отображаться на сайте и передать их в шаблон для конечного оформления вывода. Данные передаются в шаблон с помощью метода JView::assignRef() класса JView.

Листинг файла /views/hello/view.html.php:

*<?php*

*// Защита от прямого обращения к скрипту*

*defined( '\_JEXEC' ) or die( 'Restricted access' );*

*jimport( 'joomla.application.component.view');*

*class HelloViewHello extends JView*

*{*

 *function display($tpl = null)*

 *{*

 *$model =& $this->getModel();*

 *$greeting = $model->getGreeting();*

 *$this->assignRef( 'greeting', $greeting );*

 *parent::display($tpl);*

 *}*

*}*

### 2.6. Создание шаблона данных (файл /views/hello/tmpl/default.php)

Поскольку компонент очень прост и предназначен для вывода приветствия, в этом файле выводится значение переменной из Вида компонента.

Листинг файла /views/hello/tmpl/default.php:

*<?php*

*defined('\_JEXEC') or die('Restricted access'); ?>*

*<h1><?php echo $this->greeting; ?></h1>{*

### 2.7. Создание точки входа компонента для административного интерфейса (файл /admin/hello.php)

Основная среда административной части компонента очень похожа на среду лицевой части. Основной точкой входа также является файл hello.php. Этот файл идентичен файлу hello.php, который используется на сайте, кроме того, что имя загружаемого контроллера, должно быть изменено.Файл Контроллера по умолчанию также называется controller.php, и является идентичным контроллеру по умолчанию в лицевой части компонента, с тем отличием, что контроллер также должен иметь другое название.

Листинг файла /admin/hello.php:

*<?php*

*defined( '\_JEXEC' ) or die( 'Restricted access' );*

*require\_once( JPATH\_COMPONENT.DS.'controller.php' );*

*if($controller = JRequest::getWord('controller')) {*

 *$path = JPATH\_COMPONENT.DS.'controllers'.DS.$controller.'.php';*

 *if (file\_exists($path)) {*

 *require\_once $path;*

 *} else {*

 *$controller = '';*

 *}*

*}*

*$classname = 'HellosController'.$controller;*

*$controller = new $classname( );*

*$controller->execute( JRequest::getVar( 'task' ) );*

*$controller->redirect();*

### 2.8. Создание модели компонента для административного интерфейса (файл /admin/models/hellos.php)

Единственным действием модели Hellos является возможность получения списка приветствий из базы данных. Это действие выполняется с помощью метода getData().

Класс JModel имеет встроенный защищенный метод \_getList(). Этот метод может использоваться для упрощения задачи получения списка записей из базы данных. Достаточно лишь просто выполнить запрос, и он возвратит список записей.

Позже может возникнуть необходимость использовать запрос в другом методе. Для этого используется защищенный метод \_buildQuery(), который возвращает запрос, передаваемый методу \_getList(). Это облегчает изменение запроса, поскольку он вызывается из одного и того же места.

Метод \_buildQuery() просто возвращает запрос. Это выглядит примерно следующим образом:

 *function \_buildQuery()*

*{*

 *$query = ' SELECT \* '*

 *. ' FROM #\_\_hello '*

 *;*

 *return $query;*

*}*

Метод getData() получает запрос и извлекает записи из базы данных.

Листинг файла /admin/models/hellos.php:

*<?php*

*defined('\_JEXEC') or die();*

*jimport( 'joomla.application.component.model' );*

*class HellosModelHellos extends JModel*

*{*

 *function \_buildQuery()*

 *{*

 *$query = ' SELECT \* '*

 *. ' FROM #\_\_hello '*

 *;*

 *return $query;*

 *}*

 *function getData()*

 *{*

 *if (empty( $this->\_data ))*

 *{*

 *$query = $this->\_buildQuery();*

 *$this->\_data = $this->\_getList( $query );*

 *}*

 *return $this->\_data;*

 *}*

*}*

### 2.9. Создание вида компонента для административного интерфейса (файл /admin/views/hellos/view.html.php)

При реализации Вида компонента используется метод get() класса JView. Общий порядок работы Вида: первая строка берет данные из модели, вторая помещает данные в шаблон, третья отображает данные.

Таким образом листинг файла /admin/views/hellos/view.html.php имеет следующий вид:

*<?php*

*defined('\_JEXEC') or die();*

*jimport( 'joomla.application.component.view' );*

*class HellosViewHellos extends JView*

*{*

 *function display($tpl = null)*

 *{*

 *JToolBarHelper::title( JText::\_( 'Hello Manager' ), 'generic.png' );*

 *JToolBarHelper::deleteList();*

 *JToolBarHelper::editListX();*

 *JToolBarHelper::addNewX();*

 *$items =& $this->get( 'Data');*

 *$this->assignRef( 'items', $items );*

 *parent::display($tpl);*

 *}*

*}*

### 2.10. Создание шаблона вида для административного интерфейса (файл /admin/views/hellos/tmpl/default.php)

Как отмечалось выше, шаблон получает данные от Вида и осуществляет их вывод.

В разрабатываемом компоненте отобразим данные через цикл в виде простой таблицы:

*<?php defined('\_JEXEC') or die('Restricted access'); ?>*

*<form action="index.php" method="post" name="adminForm">*

*<div id="editcell">*

 *<table class="adminlist">*

 *<thead>*

 *<tr>*

 *<th width="5">*

 *<?php echo JText::\_( 'ID' ); ?>*

 *</th>*

 *<th>*

 *<?php echo JText::\_( 'Greeting' ); ?>*

 *</th>*

 *</tr>*

 *</thead>*

 *<?php*

 *$k = 0;*

 *for ($i=0, $n=count( $this->items ); $i < $n; $i++)*

 *{*

 *$row =& $this->items[$i];*

 *?>*

 *<tr class="<?php echo "row$k"; ?>">*

 *<td>*

 *<?php echo $row->id; ?>*

 *</td>*

 *<td>*

 *<?php echo $row->greeting; ?>*

 *</td>*

 *</tr>*

 *<?php*

 *$k = 1 - $k;*

 *}*

 *?>*

 *</table>*

*</div>*

*<input type="hidden" name="option" value="com\_hello" />*

*<input type="hidden" name="task" value="" />*

*<input type="hidden" name="boxchecked" value="0" />*

*<input type="hidden" name="controller" value="hello" />*

*</form>*

### 2.11. Создание установочного файла компонента (файл hello.xml)

Joomla позволяет устанавливать компоненты вручную, скопировав файлы по FTP протоколу, но лучшим вариантом является использование установочного файла для пакетной загрузки файлов и установки компонента.

Установочный XML файл может содержать разнообразную информацию и инструкции по установке:

* Детали о компоненте и авторе компонента;
* Список файлов, которые должны быть установлены;
* Внешние SQL файлы, которые содержат запросы базы данных, отдельно для установки и деинсталляции компонента.

Листинг установочного файла hello.xml:

*<?xml version="1.0" encoding="utf-8"?>*

*<!DOCTYPE install SYSTEM "http://dev.joomla.org/xml/1.5/component-install.dtd">*

*<install type="component" version="1.5.0">*

 *<name>Hello</name>*

*<!-- Далее идут необязательные элементы-->*

 *<creationDate>20.02.2009</creationDate>*

 *<author>Имя автора</author>*

 *<authorEmail></authorEmail>*

 *<authorUrl>http://www.example.org</authorUrl>*

 *<copyright>GNU/GPL</copyright>*

 *<license>Информация о лицензии</license>*

 *<!-- Номер версии сохраняется как строковое значение в таблице компонентов -->*

 *<version>Версия компонента</version>*

 *<!-- Описание необязательно, и по умолчанию равно имени компонента -->*

 *<description>Описание компонента ...</description>*

 *<!--Копирование файлов в лицевую часть сайта -->*

*<files folder="site">*

 *<filename>index.html</filename>*

 *<filename>hello.php</filename>*

 *<filename>controller.php</filename>*

 *<filename>views/index.html</filename>*

 *<filename>views/hello/index.html</filename>*

 *<filename>views/hello/view.html.php</filename>*

 *<filename>views/hello/tmpl/index.html</filename>*

 *<filename>views/hello/tmpl/default.php</filename>*

 *</files>*

 *<administration>*

 *<!--меню в административном интерфейсе -->*

 *<menu>Hello World!</menu>*

 *<!-- копирование файлов в административную часть сайта -->*

 *<files folder="admin">*

 *<filename>index.html</filename>*

 *<filename>admin.hello.php</filename>*

 *</files>*

 *</administration>*

*</install>*

Как видно из вышеприведенного листинга, в архиве с компонентом должно быть две директории: site и admin.

# Заключение

Внимательно изучив тенденции развития современного явления, более известного как Интернет, можно прийти к выводу, что информационные технологий и всемирная паутина, в частности, не стоят на месте и развиваются с каждым годом все больше и больше.

Остались в прошлом времена, когда для разработки крупного интернет-сайта необходимо было писать вручную сотни или тысячи страниц HTML. В современном мире достаточно выбрать подходящую систему управления содержимым и за несколько минут настроить интернет-портал, на разработку которого во времена начального развития Интернета потребовались бы месяцы.

В первой главе курсовой работе были изучены основные средства, используемые для разработки интернет-сайта, в частности, язык программирования PHP, СУБД MYSQL, веб-сервер компании Microsoft – IIS, а также наибольшее внимание уделено системе управления содержимым Joomla CMS.

Во второй части курсовой работы пошагово описаны и разобраны основы разработки компонентов для этой системы, как наиболее расширяемой части сайта Joomla CMS на примере простейшего компонента «Hello», выводящего приветствие посетителю сайта из базы данных проекта.

# Список использованной литературы

1. Рассохин Д.Е., Лебедев А.Г. World Wide Web – информационная паутина в сети Интернет. – М.:Техиздат, 2002. –115 с.
2. Интернет // Википедия – свободная энциклопедия. Страница «Интернет» - <http://ru.wikipedia.org/wiki/Internet>
3. Общая статистика интернет-сайтов // Сайт компании «Netcraft». Страница «May 2009 Web Server Survey» - <http://news.netcraft.com/archives/2009/05/27/may_2009_web_server_survey.html>
4. Экслер А.Б. Укрощение Интернета, или Самый полный и понятный самоучитель работы в Сети. – М.:НТ Пресс, 2007. –944 с.
5. PHP // Википедия – свободная энциклопедия. Страница «PHP» - <http://ru.wikipedia.org/wiki/PHP>
6. Бенкен Е.С. PHP, MySQL, XML: программирование для Интернета. – С. Пб.:BHV, 2008. –570 с.
7. Харрис Э. PHP/MySQL для начинающих. – С. Пб.:Издательство «КУДИЦ-Образ», 2005. –384 с.
8. Конверс Т.А. PHP 5 и MySQL. Разработка и внедрение. Библия пользователя. – М.: «Вильямс», 2006. –1216 с.
9. MySQL // Википедия – свободная энциклопедия. Страница «MySQL» - <http://ru.wikipedia.org/wiki/MySQL>
10. Введение в MySQL // Русское сообщество разработчиков MySQL. Раздел «Документы». Страница «Введение». - <http://www.mysql.ru/docs/ilyin/>
11. CMS // Википедия – свободная энциклопедия. Страница «Системы управления содержимым» - [http://ru.wikipedia.org/wiki/Системы\_управлния\_содержимым](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%BD%D0%B8%D1%8F_%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%BC%D1%8B%D0%BC)
12. Joomla! // Википедия – свободная энциклопедия. Страница «Joomla!» - <http://ru.wikipedia.org/wiki/Joomla>
13. Хаген Г.Ф. Создание веб-сайтов с помощью Joomla! 1.5 / Пер. с англ. – М.:Вильямс, 2008. -352 с.
14. Рамел Д.А. Joomla! / Пер. с англ. – С. Пб.:BHV, 2008. -572 с.
15. Единый указатель ресурсов // Википедия – свободная энциклопедия. Страница «URL» - <http://ru.wikipedia.org/wiki/URL>
16. Веб-сервер Apache // Википедия – свободная энциклопедия. Страница «Apache» - <http://ru.wikipedia.org/wiki/Apache>
17. Веб-сервер IIS // Википедия – свободная энциклопедия. Страница «Apache» - <http://ru.wikipedia.org/wiki/IIS>
18. Трехуровневая архитектура // Википедия – свободная энциклопедия. Страница «Трехуровневая архитектура» - [http://ru.wikipedia.org/wiki/Трехуровневая\_архитектура](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D1%85%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0%20)
19. Результаты ежеквартального опроса профессиональных участников рынка веб-разработок // Веб-сайт CMS Magazine – аналитический портал рынка веб-разработок. Страница «Исследования» - <http://research.cmsmagazine.ru/>