**Оглавление**

[WEB-сервер danwer 2](#_Toc418159231)

[Создание страниц с помошью HTML 5](#_Toc418159232)

[Заголовочная часть документа <HEAD> 8](#_Toc418159233)

[Заголовок документа <TITLE> 9](#_Toc418159234)

[Комментарии 9](#_Toc418159235)

[Свойства CSS 10](#_Toc418159236)

[Создание WEB-страница средствами PHP 12](#_Toc418159237)

[Переменные 14](#_Toc418159238)

[Создание и пересылка HTML-форм 15](#_Toc418159239)

[Работа с БД с помошью PHP 21](#_Toc418159240)

[Установка соединения 22](#_Toc418159241)

[Выбор базы данных 23](#_Toc418159242)

[Получение списка полей таблицы 23](#_Toc418159243)

[Запись данных в базу данных 24](#_Toc418159244)

[Отображение данных, хранящихся в MySQL 25](#_Toc418159245)

[Создание страницы с помошью JavaScript 27](#_Toc418159246)

[Изменение состава DOM 30](#_Toc418159247)

**Методическое указание по курсавому проектированию по дисциплине «Технологии Разработки WEB-приложений».**

# WEB-сервер danwer

Джентльменский набор WEB-разработчика («Д.н.w.р», читается «Денвер») — проект Дмитрия Котерова, набор дистрибутивов (Apache, PHP, MySQL, Perl и т.д.) и программная оболочка, используемые WEB-разработчиками для разработки сайтов на (локальной) Windows-машине без необходимости выхода в Интернет. Главная особенность Денвера — удобство при удаленной работе сразу над нескольктими независимыми проектами и возможность размещения на Flash-накопителе.

В состав Денвера не входит документация к компонентам, т.к. она слишком быстро устаревает. Вместо этого можно воспользоваться адаптированной формой поиска входящий в состав сервера. После нажатия на Enter она сама переадресует запросы на необходимые сайты.

Для установки WEB-сервера danwer необходимо закрыть все браузеры открытые на данный момент на компьютере пользователя.

Для начала установки необходимо запустить программу-инсталлятор.

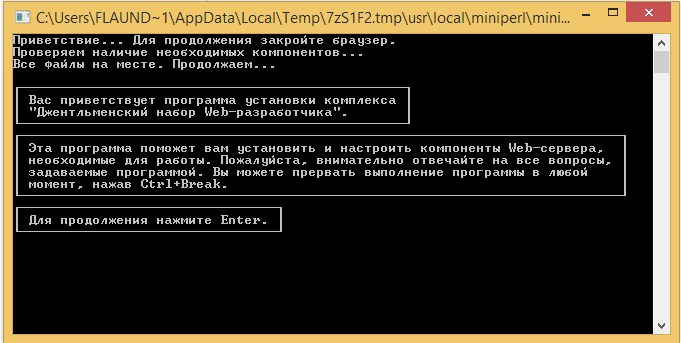


Рисунок 1. Окно инсталлятора WEB-сервера danwer

Для установки WEB-сервера danwer необходимо указать адрес на локальном компьютере для установки программы. WEB-сервер danwer также создает виртуальный жесткий диск, который является виртуальной Unix-like системой. Для виртуальной системы необходимо задать литеру диска.

Также WEB-сервер danwer может быть установлен в 2х режимах. Для простой работы с WEB-сервером достаточно использовать стандартный режим.

Чтобы начать использовать Денвер, после его установки проделайте следующие действия:

1. Запустите Денвер, воспользовавшись ярлыком Start Denwer на Рабочем столе. Если вы не создавали ярлыки, то можно запустить Денвер по команде C:\WEBServers/denwer\Run.exe.
2. Откройте браузер и перейдите по адресу http://localhost.
3. Вы должны увидеть главную страницу Денвера.
4. Если после запуска Денвера http://localhost не открывается, проверьте, не блокируется ли Денвер вашим антивирусом или фаерволом. Например, были замечены проблемы с NOD32 в Windows XP (в нем нужно добавить процесс Z:/usr/local/apache/bin/httpd.exe в список исключений, это можно сделать в окне IMON/Настройка/Разное/Исключение).

Внимание: если вы используете Skype, убедитесь, что он не занимает порты 80 и 443, необходимые для работы Apache в Денвере ("Инструменты - Настройки - Дополнительно - Соединение - Использовать порты 80 и 443 в качестве входящих альтернативных" должно быть отключено).

Если по каким-то причинам Денвер не заработал, свяжитесь, с разработчиками: http://forum.dklab.ru/denwer/bugs/. Прикрепите к сообщению следующую информацию:

1. При каких условиях проявилась ошибка? Что вы сделали перед тем, как его зафиксировали?
2. Точную версию Вашей OS (можно получить по команде winver, запущенной в Командной строке).
3. Файл netstat.txt, получившийся в результате работы команды netstat -nb > C:\netstat.txt.
4. Значимые сообщения из конца файла /usr/local/apache/logs/error.log.

На главной странице Денвера в разделе утилиты располагаются ссылки на страницы по созданию и администрированию баз данных MySQL.

Для заведения базы данных необходимо перейти по ссылке «Заведение новых БД и пользователей MySQL».

В предложенной форме обязательны поля «Имя базы данных» и «Логин пользователя». Поле пароль не обязательно для заполнения, если оно не задано, пользователю базы данных не будет создан пароль. Однако в целях безопасности лучше задать пароль пользователю. Поле «Пароль администратора MySQL» не задается, если у пользователя MySQL не задан пароль (по умолчанию его нет).

После создания базы данных, можно приступать к ее редактированию. Для этого нужно перейти в phpMyAdmin. Сделать это можно через главную страницу Денвера в разделе «Утилиты».

При помощи данного инструмента можно создавать, редактировать и удалять таблицы баз данных и данные в них. В phpMyAdmin также есть возможность создавать QSL запросы, что значительно упростит последующее их применение.

# Создание страниц с помошью HTML

HyperText Markup Language (HTML) является стандартным языком, предназначенным для создания гипертекстовых документов в среде WEB. HTML-документы могут просматриваться различными типами WEB-броузеров. Когда документ создан с использованием HTML, WEB-броузер может интерпретировать HTML для выделения различных элементов документа и первичной их обработки. Использование HTML позволяет форматировать документы для их представления с использованием шрифтов, линий и других графических элементов на любой системе, их просматривающей.

Большинство документов имеют стандартные элементы, такие, как заголовок, параграфы или списки. Используя тэги HTML вы можете обозначать данные элементы, обеспечивая WEB-броузеры минимальной информацией для отображения данных элементов, сохраняя вцелом общую структуру и информационную полноту документов. Все что необходимо, чтобы прочитать HTML-документ - это WEB-броузер, который интерпретирует тэги HTML и воспроизводит на экране документ в виде, который ему придает автор.

В большинстве случаев автор документа строго определяет внешний вид документа. В случае HTML читатель (основываясь на возможностях WEB-броузера может, в определенной степени, управлять внешним видом документа (но не его содержимым). HTML позволяет отметить, где в документе должен быть заголовок или абзац при помощи тэга HTML, а затем предоставляет WEB-броузеру интерпретировать эти тэги. Например, один WEB-броузер может распознавать тэг начала абзаца и представлять документ в нужном виде, а другой не имеет такой возможности и представляет документ в одну строку. Пользователи некоторых WEB-броузеров имеют, также, возможность настраивать размер и вид шрифта, цвет и другие параметры, влияющие на отображение документа.

HTML-тэги могут быть условно разделены на две категории:

1. тэги, определяющие, как будет отображаться WEB-броузером тело документа в целом
2. тэги, описывающие общие свойства документа, такие как заголовок или автор документа

Запомните, что основное преимущество HTML заключается в том, что ваш документ может быть просмотрен на WEB-броузерах различных типов и на различных платформах.

Основным понятием CSS является стиль – т. е. набор правил оформления и форматирования, который может быть применен к различным элементам страницы. В стандартном HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходилось каждый раз описывать эти свойства, даже если на одной страничке должны располагаться 10 или 110 таких элементов, ничуть не отличающихся один от другого.

CSS создан для задания свойств элеменьам HTML. Для присвоения какому-либо элементу определенных характеристик необходимо один раз описать этот элемент и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который необходимо оформить соответствующим образом, должен принять свойства стиля.

Более того, можно сохранить описание стиля не в тексте странички, а в отдельном файле – это позволит использовать описание стиля на любом количестве Web­страниц. И еще одно, связанное с этим, преимущество – возможность изменить оформление любого количества страниц, исправив лишь описание стиля в одном (отдельном) файле.

Кроме того, CSS позволяет работать со шрифтовым оформлением страниц на гораздо более высоком уровне, чем стандартный HTML, избегая излишнего утяжеления страниц графикой.

HTML-документы могут быть созданы при помощи любого текстового редактора или специализированных HTML-редакторов и конвертеров. Выбор редактора, который будет использоваться для создания HTML-документов, зависит исключительно от понятия удобства и личных пристрастий каждого автора.

Например, HTML редакторы, такие, как "Netscape Navigator Gold" компании Netscape позволяют создавать документы графически с использованием технологии WYSIWYG (What You See Is What You Get). С другой стороны, большинство традиционных средств для создания документов имеют конвертеры, позволяющие преобразовывать документы к формату HTML.

Все тэги HTML начинаются с "<" (левой угловой скобки) и заканчиваются символом ">" (правой угловой скобки). Как правило, существует стартовый тэг и завершающий тэг. Для примера приведем тэги заголовка, определяющие текст, находящийся внутри стартового и завершающего тэга и описывающий заголовок документа:

<TITLE> Заголовок документа </TITLE>

Завершающий тэг выглядит также, как стартовый, и отличается от него прямым слэшем перед текстом внутри угловых скобок. В данном примере тэг <TITLE> говорит WEB-броузеру об использовании формата заголовка, а тэг </TITLE> - о завершении текста заголовка.

Некоторые тэги, такие, как <P> (тэг, определяющий абзац), не требуют завершающего тэга, но его использование придает исходному тексту документа улучшенную читаемость и структурируемость.

HTML не реагирует на регистр символов, описывающих тэг, и приведенный ранее пример может выглядеть следующим образом:

<title> Заголовок документа </title>

Внимание! Дополнительные пробелы, символы табуляции и возврата каретки, добавленные в исходный текст HTML-документа для его лучшей читаемости, будут проигнорированы WEB-броузером при интерпретации документа. HTML-документ может включать вышеописанные элементы только если они помещены внутрь тэгов <PRE> и </PRE>.

Когда WEB-броузер получает документ, он определяет, как документ должен быть интерпретирован. Самый первый тэг, который встречается в документе, должен быть тэгом <HTML>. Данный тэг сообщает WEB-броузеру, что ваш документ написан с использованием HTML. Минимальный HTML-документ будет выглядеть так:

<HTML> ...тело документа... </HTML>

## Заголовочная часть документа <HEAD>

Тэг заголовочной части документа должен быть использован сразу после тэга <HTML> и более нигде в теле документа. Данный тэг представляет из себя общее описание документа. Избегайте размещать какой-либо текст внутри тэга <HEAD>. Стартовый тэг <HEAD> помещается непосредственно перед тэгом <TITLE> и другими тэгами, описывающими документ, а завершающий тэг </HEAD> размещается сразу после окончания описания документа. Например:

<HTML>

<HEAD>

<TITLE> Список сотрудников </TITLE>

</HEAD>

...

Внимание! Технически, стартовые и завершающие тэги типа <HTML>, <HEAD> и <BODY> необязательны. Но настоятельно рекомендуется их использовать, поскольку использование данных тэгов позволяет WEB-броузеру уверенно разделить заголовочную часть документа и непосредственно смысловую часть.

## Заголовок документа <TITLE>

Большинство WEB-броузеров отображают содержимое тэга <TITLE> в заголовке окна, содержащего документ и в файле закладок, если он поддерживается WEB-броузером. Заголовок, ограниченный тэгами <TITLE> и </TITLE>, размещается внутри <HEAD>-тэгов, как показано выше на примере. Заголовок документа не появляется при отображении самого документа в окне.

## Комментарии

Как любой язык, HTML позволяет вставлять в тело документа комментарии, которые сохраняются при передаче документа по сети, но не отображаются броузером. Синтаксис комментария:

<!-- Это комментарий -->

Комментарии могут встречаться в документе где угодно и в любом количестве.

## Свойства CSS

Самый простой случай присвоения какому-либо элементу определенного стиля выглядит так:

НАЗВАНИЕ\_ЭЛЕМЕНТА {свойство: значение;},

Где НАЗВАНИЕ\_ЭЛЕМЕНТА – имя HTML­тега (H1, P, TD, A и т. д.), а параметры в фигурных скобках – список свойств элемента и присвоенных им значений.

Пример:

H1 {font-size: 30pt; color: blue;}

В этом примере всем заголовкам на странице, оформленным тегом Н1, присваивается размер шрифта 30 пунктов и синий цвет.

CSS реализует возможность присваивать стили не всем одинаковым элементам страницы, а избирательно – для этого используется параметр CLASS = "имя класса" или идентификатор ID=«имя элемента», присваивающиеся любому элементу страницы. Рассмотрим эти возможности подробнее.

Параметр CLASS применяется в случае, если необходимо создать одинаковый стиль для нескольких, но не всех элементов страницы (одинаковых или разных).

Пример:

.b-с {font-weight: bold; text-align: center}

– описание стиля для класса b-с

Все элементы класса b-с будут отображаться жирным шрифтом с выравниванием по центру страницы (или ячейки таблицы).

<P CLASS="b-с">Текст параграфа</P>

– параграфу присвоен стиль класса b-с. <TD CLASS="b-c">текст</TD>

– ячейке таблицы присвоен стиль класса b-c.

# Создание WEB-страница средствами PHP

PHP (англ. PHP: Hypertext Preprocessor — «PHP: препроцессор гипертекста»; первоначально Personal Home Page Tools — «Инструменты для создания персональных веб-страниц»; произносится пи-эйч-пи) — скриптовый язык программирования общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания динамических веб-сайтов.

Язык и его интерпретатор разрабатываются группой энтузиастов в рамках проекта с открытым кодом. Проект распространяется под собственной лицензией, несовместимой с GNU GPL.

В области веб-программирования, в частности серверная часть, PHP — один из популярных сценарных языков (наряду с JSP, Perl и языками, используемыми в ASP.NET) благодаря своей простоте, скорости выполнения, богатой функциональности, кроссплатформенности и распространению исходных кодов на основе лицензии PHP.

Популярность в области построения веб-сайтов определяется наличием большого набора встроенных средств для разработки веб-приложений. Основные из них:

* автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
* взаимодействие с большим количеством различных систем управления базами данных (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Интерфейс PDO);
* автоматизированная отправка HTTP-заголовков;
* работа с HTTP-авторизацией;
* работа с cookies и сессиями;
* работа с локальными и удалёнными файлами, сокетами;
* обработка файлов, загружаемых на сервер;
* работа с XForms.

В настоящее время PHP используется сотнями тысяч разработчиков. Согласно рейтингу корпорации TIOBE, базирующемся на данных поисковых систем, в июне 2013 года PHP находился на 5 месте среди языков программирования. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др.

Тексты PHP-программ (или, как принято говорить об интерпретируемых языках: PHP-скрипта) по-умолчанию хранятся в файлах с раширением \*.php. Существует две идеологии, два способа использования PHP: 1. PHP может являться вкраплением в HTML-страницу, лишь частично изменяя еѐ содержимое. Т.е. основой файла всѐ равно является HTML разметка. 2. PHP может являться полноценной программой (а не добавкой к HTML), которая «на лету» генерирует HTML-разметку, используя, например, шаблоны содержимого, подключает дополнительные файлы и т.д.

Есть несколько способов. Один из них – с помощью открывающего тега <?php и закрывающего тега ?>. Такого вида специальные теги позволяют переключаться между режимами HTML и PHP. Этот синтаксис наиболее предпочтителен, поскольку позволяет задействовать PHP в XML-совместим.

Самый простейший способ общения с пользователем через Web-страницу, это послать ему в странице какой-нибудь текст. Это можно сделать двумя способами:с помощью функции print или echo:

<?php echo "Hello, world."; ?>

<?php print "Hello, world."; ?>ых программах (например, написанных на языке XHTML).

Echo это функция, которая отправляет броузеру текст. Между словом "echo" и символом ";" мы помещаем строку, которая обозначается кавычками. Все, что находится внутри кавычек, будет отправлено браузеру.

В PHP инструкции разделяются так же, как и в Cи или Perl, – каждое выражение заканчивается точкой с запятой. Закрывающий тег «?>» также подразумевает конец инструкции, поэтому перед ним точку с запятой не ставят.

<?php // точка с запятой в конце команды обязательна echo "Hello, world!"; ... ?>

## Переменные

Переменная в PHP обозначается знаком доллара, за которым следует ее имя. Например:

$my\_var

Имя переменной чувствительно к регистру, т.е. переменные $my\_var и $My\_var различны. Имена переменных соответствуют тем же правилам, что и остальные наименования в PHP: правильное имя переменной должно начинаться с буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания. При объявлении переменных в РНР не требуется явно указывать тип переменной, при этом одна и та же переменная может иметь на протяжении программы разные типы. Переменная инициализируется в момент присваивания ей значения и существует до тех пор, пока выполняется программа. Т.е., в случае web-страницы это означает, что до тех пор, пока не завершен запрос.

Для хранения постоянных величин, т.е. таких величин, значение которых не меняется в ходе выполнения скрипта, используются константы. Такими величинами могут быть математические константы, пароли, пути к файлам и т.п. Основное отличие константы от переменной состоит в том, что ей нельзя присвоить значение больше одного раза и ее значение нельзя аннулировать после ее объявления. Кроме того, у константы нет приставки в виде знака доллара и ее нельзя определить простым присваиванием значения. Для этого существует специальная функция define(). Ее синтаксис таков:

define("Имя\_константы", "Значение\_константы", [Нечувствительность\_к\_регистру])

# Создание и пересылка HTML-форм

Для передачи данных от пользователя к серверу в языке HTML есть специальная конструкция – формы. Формы предназначены для того, чтобы получать от пользователя информацию. Например, вам нужно знать логин и пароль пользователя для того, чтобы определить, на какие страницы сайта его можно допускать. Или вам необходимы личные данные пользователя, чтобы была возможность с ним связаться. Формы как раз и применяются для ввода такой информации. В них можно вводить текст или выбирать подходящие варианты из списка. Данные, записанные в форму, отправляются для обработки специальной программе (например, скрипту на PHP) на сервере. В зависимости от введенных пользователем данных эта программа может формировать различные web-страницы, отправлять запросы к базе данных, запускать различные приложения и т.п.

Для создания формы в языке HTML используется тег FORM. Внутри него находится одна или несколько команд INPUT. С помощью атрибутов action и method тега FORM задаются имя программы, которая будет обрабатывать данные формы, и метод запроса, соответственно. Команда INPUT определяет тип и различные характеристики запрашиваемой информации. Отправка данных формы происходит после нажатия кнопки input типа submit.

Метод сообщает о цели запроса клиента. Протокол HTTP поддерживает достаточно много методов, но реально используются только три: POST, GET и HEAD.

При отправке данных формы с помощью метода GET содержимое формы добавляется к URL после знака вопроса в виде пар имя=значения, объединенных с помощью амперсанда &:

action?name1=value1&name2=value2&name3=value3

Здесь action – это URL-адрес программы, которая должна обрабатывать форму (это либо программа, заданная в атрибуте action тега form, либо сама текущая программа, если этот атрибут опущен). Имена name1, name2, name3 соответствуют именам элементов формы, а value1, value2, value3 – значениям этих элементов. Все специальные символы, включая = и &, в именах или значениях этих параметров будут опущены. Поэтому не стоит использовать в названиях или значениях элементов формы эти символы и символы кириллицы в идентификаторах.

Для метода POST содержимое формы кодируется точно так же, как для метода GET, но вместо добавления строки к URL содержимое запроса посылается блоком данных как часть операции POST. Если присутствует атрибут ACTION, то значение URL, которое там находится, определяет, куда посылать этот блок данных. Этот метод, как уже отмечалось, рекомендуется для передачи больших по объему блоков данных.

Основное преимущество POST запросов – это их большая безопасность и функциональность по сравнению с GET-запросами. Поэтому метод POST чаще используют для передачи важной информации, а также информации большого объема. Тем не менее, не стоит целиком полагаться на безопасность этого механизма, поскольку данные POST запроса также можно подделать, например создав html-файл на своей машине и заполнив его нужными данными. Кроме того, не все клиенты могут применять метод POST, что ограничивает варианты его использования.

При отправке данных на сервер любым методом передаются не только сами данные, введенные пользователем, но и ряд переменных, называемых переменными окружения, характеризующих клиента, историю его работы, пути к файлам и т.п. Вот некоторые из переменных окружения:

* REMOTE\_ADDR – IP-адрес хоста (компьютера), отправляющего запрос;
* REMOTE\_HOST – имя хоста, с которого отправлен запрос;
* HTTP\_REFERER – адрес страницы, ссылающейся на текущий скрипт;
* REQUEST\_METHOD – метод, который был использован при отправке запроса;
* QUERY\_STRING – информация, находящаяся в URL после знака вопроса;
* SCRIPT\_NAME – виртуальный путь к программе, которая должна выполняться;
* HTTP\_USER\_AGENT – информация о браузере, который использует клиент

Создадим форму для регистрации участников заочной школы программирования.

Пример:

**<**h2>Форма для регистрации участников</h2>

<form action="1.php" method=POST>

<!--создаем форму-->

<!--данные формы будет обрабатывать файл 1.php, при отправке запроса будет использован метод POST-->

Имя <br>

<input type=text name="first\_name" value="Введите Ваше имя">

<br> Фамилия <br>

<input type=text name="last\_name"><br>

E-mail <br>

<input type=text name="email"><br>

<p> Выберите курс, который вы бы хотели посещать:<br>

<input type=radio name="kurs" value="PHP">PHP<br>

<input type=radio name="kurs" value="Lisp">Lisp<br>

<input type=radio name="kurs" value="Perl">Perl<br>

<input type=radio name="kurs" value="Unix">Unix<br>

<P>Что вы хотите, чтобы мы знали о вас? <BR>

<textarea name="comment" cols=32 rows=5></textarea>

<P><input name="confirm" type=checkbox checked> Подтвердить получение <br>

<input type=submit value="Отправить">

<input type=reset value="Отменить">

</form>

Как мы видим, в форме можно указывать метод передачи данных.

Внутри PHP-скрипта имеется несколько способов получения доступа к данным, переданным клиентом по протоколу HTTP. До версии PHP 4.1.0 доступ к таким данным осуществлялся по именам переданных переменных (напомним, что данные передаются в виде пар «имя переменной, символ «=», значение переменной»). Таким образом, если, например, было передано first\_name=Nina, то внутри скрипта появлялась переменная $first\_name со значением Nina. Если требовалось различать, каким методом были переданы данные, то использовались ассоциативные массивы $HTTP\_POST\_VARS и $HTTP\_GET\_VARS, ключами которых являлись имена переданных переменных, а значениями – соответственно значения этих переменных. Таким образом, если пара first\_name=Nina передана методом GET, то $HTTP\_GET\_VARS["first\_name"]="Nina".

Использовать в программе имена переданных переменных напрямую небезопасно. Поэтому было решено начиная с PHP 4.1.0 задействовать для обращения к переменным, переданным с помощью HTTP-запросов, специальный массив – $\_REQUEST. Этот массив содержит данные, переданные методами POST и GET, а также с помощью HTTP cookies. Это суперглобальный ассоциативный массив, т.е. его значения можно получить в любом месте программы, используя в качестве ключа имя соответствующей переменной (элемента формы).

После введения массива $\_REQUEST массивы $HTTP\_POST\_VARS и $HTTP\_GET\_VARS для однородности были переименованы в $\_POST и $\_GET соответственно, но сами они из обихода не исчезли из соображений совместимости с предыдущими версиями PHP. В отличие от своих предшественников, массивы $\_POST и $\_GET стали суперглобальными, т.е. доступными напрямую и внутри функций и методов.

Для того, чтобы сохранить возможность обработки скриптов более ранних версий, чем PHP 4.1.0, была введена директива register\_globals, разрешающая или запрещающая доступ к переменным непосредственно по их именам. Если в файле настроек PHP параметр register\_globals=On, то к переменным, переданным серверу методами GET и POST, можно обращаться просто по их именам (т.е. можно писать $first\_name). Если же register\_globals=Off, то нужно писать

$\_REQUEST["first\_name"] или

$\_POST["first\_name"],

$\_GET["first\_name"],

$HTTP\_POST\_VARS["first\_name"],

$HTTP\_GET\_VARS["first\_name"].

С точки зрения безопасности эту директиву лучше отключать (т.е. register\_globals=Off). При включенной директиве register\_globals перечисленные выше массивы также будут содержать данные, переданные клиентом.

Иногда возникает необходимость узнать значение какой-либо переменной окружения, например метод, использовавшийся при передаче запроса или IP-адрес компьютера, отправившего запрос. Получить такую информацию можно с помощью функции getenv(). Она возвращает значение переменной окружения, имя которой передано ей в качестве параметра.

## Работа с БД с помошью PHP

MySQL – это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySQL – свободно распространяемая система, т.е. платить за ее применение не нужно. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов.

Работать с MySQL можно не только в текстовом режиме, но и в графическом. Существует очень популярный визуальный интерфейс (кстати, написанный на PHP) для работы с этой СУБД. Называется он PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в MySQL.

В дистрибутив PHP входит расширение, содержащее встроенные функции для работы с базой данных MySQL. В этой лекции мы познакомимся с некоторыми основными функциями для работы с MySQL, которые потребуются для решения задач построения web-интерфейсов с целью отображения и наполнения базы данных. Возникает вопрос, зачем строить такие интерфейсы? Для того чтобы вносить информацию в базу данных и просматривать ее содержимое могли люди, не знакомые с языком запросов SQL. При работе с web-интерфейсом для добавления информации в базу данных человеку нужно просто ввести эти данные в html-форму и отправить их на сервер, а наш скрипт сделает все остальное. А для просмотра содержимого таблиц достаточно просто щелкнуть по ссылке и зайти на нужную страницу.

Чтобы построить интерфейс для добавления информации таблицу, нужно ее структуру (т.е. набор ее полей) тобразить в html-форму.

Разобьем эту задачу на следующие подзадачи:

1. установка соединения с БД;
2. выбор рабочей БД;
3. получение списка полей таблицы;
4. отображение полей в html-форму.

После этого данные, введенные в форму, нужно записать в базу данных. Рассмотрим все эти задачи по порядку.

## Установка соединения

Итак, первое, что нужно сделать, - это установить соединение с базой данных. Воспользуемся функцией mysql\_connect.

Синтаксис mysql\_connect

ресурс mysql\_connect ( [строка server

[, строка username [, строка password

[, логическое new\_link

[, целое client\_flags]]]]])

Данная функция устанавливает соединение с сервером MySQL и возвращает указатель на это соединение или FALSE в случае неудачи. Для отсутствующих параметров устанавливаются следующие значения по умолчанию:

server = 'localhost:3306'

username = имя пользователя владельца процесса сервера

password = пустой пароль/

Если функция вызывается дважды с одними и теми же параметрами, то новое соединение не устанавливается, а возвращается ссылка на старое соединение. Чтобы этого избежать, используют параметр new\_link, который заставляет в любом случае открыть еще одно соединение.

Параметр client\_flags - это комбинация следующих констант: MYSQL\_CLIENT\_COMPRESS (использовать протокол сжатия), MYSQL\_CLIENT\_IGNORE\_SPACE (позволяет вставлять пробелы после имен функций), MYSQL\_CLIENT\_INTERACTIVE (ждать interactive\_timeout секунд - вместо wait\_timeout - до закрытия соединения).

Соединение с сервером закрывается при завершении исполнения скрипта, если оно до этого не было закрыто с помощью функции mysql\_close().

## Выбор базы данных

После установки соединения нужно выбрать базу данных, с которой будем работать.

В PHP для этого существует функция mysql\_select\_db.

Синтаксис mysql\_select\_db:

логическое mysql\_select\_db (

строка database\_name

[, ресурс link\_identifier])

Эта функция возвращает TRUE в случае успешного выбора базы данных и FALSE - в противном случае.

## Получение списка полей таблицы

В PHP на этот случай есть команда - mysql\_list\_fields.

Синтаксис mysql\_list\_fields

ресурс mysql\_list\_fields ( строка database\_name, строка table\_name [, ресурс link\_identifier])

Эта функция возвращает список полей в таблице table\_name в базе данных database\_name. Получается, что выбирать базу данных было необязательно, но это пригодится позже. Как можно заметить, результат работы этой функции - переменная типа ресурс. То есть это не совсем то, что мы хотели получить. Это ссылка, которую можно использовать для получения информации о полях таблицы, включая их названия, типы и флаги. Функция mysql\_field\_name возвращает имя поля, полученного в результате выполнения запроса. Функция mysql\_field\_len возвращает длину поля. Функция mysql\_field\_type возвращает тип поля, а функция mysql\_field\_flags возвращает список флагов поля, записанных через пробел. Типы поля могут быть int, real, string, blob и т.д. Флаги могут быть not\_null, primary\_key, unique\_key, blob, auto\_increment и т.д.

Синтаксис у всех этих команд одинаков:

строка mysql\_field\_name ( ресурс result, целое field\_offset)

строка mysql\_field\_type ( ресурс result, целое field\_offset)

строка mysql\_field\_flags (ресурс result, целое field\_offset)

строка mysql\_field\_len ( ресурс result, целое field\_offset)

Здесь result - это идентификатор результата запроса (например, запроса, отправленного функциями mysql\_list\_fields или mysql\_query (о ней будет рассказано позднее)), а field\_offset - порядковый номер поля в результате.

## Запись данных в базу данных

Для того чтобы записать данные в таблицу, используется команда INSERT языка SQL.

Для выполнения команд SQL существует функция mysql\_query().

Синтаксис mysql\_query

ресурс mysql\_query ( строка query [, ресурс link\_identifier])

mysql\_query() посылает SQL-запрос активной базе данных MySQL сервера, который определяется с помощью указателя link\_identifier (это ссылка на какое-то соединение с сервером MySQL).

Замечание: строка запроса НЕ должна заканчиваться точкой с запятой.

Только для запросов SELECT, SHOW, EXPLAIN, DESCRIBE, mysql\_query() возвращает указатель на результат запроса, или FALSE, если запрос не был выполнен. В остальных случаях mysql\_query() возвращает TRUE, если запрос выполнен успешно, и FALSE - в случае ошибки. Значение, не равное FALSE, говорит о том, что запрос был выполнен успешно. Оно не говорит о количестве затронутых или возвращенных рядов. Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда. mysql\_query() также считается ошибочным и вернет FALSE, если у пользователя недостаточно прав для работы с указанной в запросе таблицей.

## Отображение данных, хранящихся в MySQL

Чтобы отобразить какие-то данные в браузер с помощью PHP, нужно сначала получить эти данные в виде переменных PHP.

Требуется отправить запрос на выбор всех полей, а результат отобразить в браузере. И здесь результат - это целая таблица значений, а точнее, указатель на эту таблицу. Так что нужны какие-то аналоги функции mysql\_field\_name(), только чтобы они извлекали из результата запроса не имя, а значение поля. Таких функций в PHP несколько. Наиболее популярные - mysql\_result() и mysql\_fetch\_array().

Синтаксис mysql\_result

смешанное mysql\_result (ресурс result, целое row [, смешанное field])

mysql\_result() возвращает значение одной ячейки результата запроса. Аргумент field может быть порядковым номером поля в результате, именем поля или именем поля с именем таблицы через точку tablename.fieldname. Если для имени поля в запросе применялся алиас ('select foo as bar from...'), используйте его вместо реального имени поля.

Вызовы функции mysql\_result() не должны смешиваться с другими функциями, работающими с результатом запроса.

Синтаксис mysql\_fetch\_array

массив mysql\_fetch\_array ( ресурс result [, целое result\_type])

Эта функция обрабатывает ряд результата запроса, возвращая массив (ассоциативный, численный или оба) с обработанным рядом результата запроса, или FALSE, если рядов больше нет.

mysql\_fetch\_array() - это расширенная версия функции mysql\_fetch\_row(). Помимо хранения значений в массиве с численными индексами, функция возвращает значения в массиве с индексами по названию колонок.

Если несколько колонок в результате будут иметь одинаковые названия, будет возвращена последняя колонка. Чтобы получить доступ к первым, следует использовать численные индексы массива или алиасы в запросе.

Важно заметить, что mysql\_fetch\_array() работает НЕ медленнее, чем mysql\_fetch\_row(), и предоставляет более удобный доступ к данным.

Второй опциональный аргумент result\_type в функции mysql\_fetch\_array() является константой и может принимать следующие значения: MYSQL\_ASSOC, MYSQL\_NUM и MYSQL\_BOTH. Эта возможность добавлена в PHP 3.0.7. Значением по умолчанию является: MYSQL\_BOTH.

Используя MYSQL\_BOTH, получим массив, состоящий как из ассоциативных индексов, так и из численных. MYSQL\_ASSOC вернет только ассоциативные соответствия, а MYSQL\_NUM - только численные.

# Создание страницы с помошью JavaScript

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

Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит. На практике это выражается в том, что можно, например, изменить цвет фона страницы или интегрированную в документ картинку, открыть новое окно или выдать предупреждение.

Главный вопрос любого начинающего программиста: "Как оформить программу и выполнить ее?". Попробуем на него ответить как можно проще, но при этом не забывая обо всех способах применения JavaScript-кода.

Исполняет JavaScript-код браузер. В него встроен интерпретатор JavaScript. Следовательно, выполнение программы зависит от того, когда и как этот интерпретатор получает управление. Это, в свою очередь, зависит от функционального применения кода. В общем случае можно выделить четыре способа функционального применения JavaScript:

* гипертекстовая ссылка (схема URL);
* обработчик события (в атрибутах, отвечающих событиям);
* подстановка (entity);
* вставка (контейнер <SCRIPT>).

Объектно-ориентированный язык программирования предполагает наличие иерархии классов объектов. В JavaScript такая иерархия начинается с класса объектов window, т.е. каждый объект приписан к тому или иному окну. Для обращения к любому объекту или его свойству указывают полное или частичное имя этого объекта или свойства объекта, начиная с имени объекта, старшего в иерархии, в который входит данный объект.

У объектов DOM некоторые свойства обязательно присутствуют, тогда как наличие других зависит от Web-страницы. Например, объект window всегда имеет в качестве своих свойств объекты location и history, т.е. это обязательные свойства. Если HTML-страница содержит контейнер <BODY>, то у объекта window будет присутствовать в качестве свойства объект document. Если HTML-страница содержит контейнер <FRAMESET> со вложенными в него контейнерами <FRAME>, то у объекта window будут присутствовать в качестве свойств имена фреймов, например window.f1.

Коллекция — это структура данных JavaScript, похожая на массив. Отличие коллекции от массивов заключается в том, что массивы программист создает сам в коде программы и заполняет их данными; коллекции же создаются браузером и "населяются" объектами, связанными с элементами Web-страницы. Коллекцию можно рассматривать как другой, зачастую более удобный способ доступа к объектам Web-страницы.

Многие HTML-контейнеры имеют атрибуты. А каждому контейнеру соответствует объект. При этом соответствии атрибутам отвечают свойства объекта. Соответствие между атрибутами HTML-контейнеров и свойствами DOM-объектов не всегда прямое. Обычно каждому атрибуту отвечает некоторое свойство объекта. Но, во-первых, название этого свойства не всегда легко угадать по названию атрибута, а во-вторых, у объекта могут быть свойства, не имеющие аналогов среди атрибутов. Кроме того, как мы знаем, атрибуты являются регистро-независимыми, как и весь язык HTML, тогда как свойства объектов нужно писать в точно определенном регистре символов.

Например, контейнер якоря

<A ...>...</A>

имеет атрибут HREF, который превращает его в гипертекстовую ссылку:

<A HREF="http://its.kpi.ua/">ITS</A>

Данной гиперссылке соответствует объект (класса URL) — document.links[0], если предполагать, что это первая ссылка в нашем документе. Тогда атрибуту HREF будет соответствовать свойство href этого объекта. К свойству объекта можно обращаться с помощью точечной нотации: объект.свойство. Например, чтобы изменить адрес, на который указывает эта ссылка, мы можем написать: document.links[0].href='http://ya.ru/';

В терминологии JavaScript методы объекта определяют функции, с помощью которых выполняются действия с этим объектом, например, изменение его свойств, отображения их на web-странице, отправка данных на сервер, перезагрузка страницы и т.п.

Например, если у нас есть ссылка <A HREF="http://its.kpi.ua/">ITS</A> (будем считать, она первая в нашем документе), то у соответствующего ей объекта document.links[0] есть метод click(). Его вызов в любом месте JavaScript-программы равносилен тому, как если бы пользователь кликнул по ссылке.

Кроме методов и свойств, объекты характеризуются событиями. Собственно, суть программирования на JavaScript заключается в написании обработчиков этих событий. Например, с объектом типа button (контейнер INPUT типа button — "кнопка") может происходить событие Click, т.е. пользователь может нажать на кнопку. Для этого атрибуты контейнера INPUT расширены атрибутом обработки этого события — onClick. В качестве значения этого атрибута указывается программа обработки события, которую должен написать на JavaScript автор HTML-документа:

<INPUT TYPE=button VALUE="Нажать" onClick="alert('Пожалуйста, нажмите еще раз')">

Обработчики событий указываются в специально созданных для этого атрибутах у тех контейнеров, с которыми эти события связаны. Например, контейнер BODY определяет свойства всего документа, поэтому обработчик события "завершена загрузка всего документа" указывается в этом контейнере как значение атрибута onLoad.

## Изменение состава DOM

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. Сейчас разработка jQuery ведется командой jQuery во главе с Джоном Резигом.

append( контент )

Добавляет контент внутрь каждого элемента набора. Добавляемый контент следует за уже существующим.

Данный метод подобен применению appendChild.

Аргументы:

Контент - Строка, Элемент, jQuery

Контент, который необходимо добавить.

Пример:

$("p").append("<strong>Hello</strong>");

prepend( контент )

Добавляет контент внутрь каждого элемента набора. Добавляемый контент следует перед уже существующим.

Аргументы:

Контент - Строка, Элемент, jQuery

Контент, который необходимо добавить.

html( )

Получает содержимое HTML (innerHTML) каждого совпавшего элемента. С документами XML использовать данный метод нельзя, но можно использовать с документами XHTML.

Преобразует HTML в текст при нажатии ЛКМ.

$("p").click(function () {

var htmlStr = $(this).html();

$(this).text(htmlStr);

});

html( значение )

Добавляет код HTML для каждого совпавшего элемента. С документами XML использовать данный метод нельзя, но можно использовать с документами XHTML.

Аргументы:

Значение – Строка в формате HTML

HTML код.