**Министерство образования**

**Российской Федерации**

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**№ \_\_\_\_**

**WEB - ПРОГРАММИРОВАНИЕ**

**Методические указания к лабораторным работам**

**для студентов III курса АВТФ**

**направления****230100.62**

**«Информатика и вычислительная техника»**

**дневной формы обучения**

**НОВОСИБИРСК**

**2014**

**Составитель: *А.В. Гунько*, канд. техн. наук, доц.**

**Рецензент *Ю.В. Новицкая*, ст. преп. каф. ВТ**

**Работа подготовлена на кафедре автоматики**

**© Новосибирский государственный технический университет, 2014 г.**

#### ЛАБОРАТОРНАЯ РАБОТА № 1

**MySQL. Проектирование и реализация базы данных.**

1. Цель работы

Изучить команды монитора MySQL, освоить операции создания таблиц, выборки, вставки, изменения и удаления данных.

1. Краткие теоретические сведения

Технология реляционных баз данных предполагает структурирование данных в таблицах, приведенных к нормальной форме. При этом связь между таблицами осуществляется с помощью ключевых полей. Структура таблиц с указанием типов данных в столбцах совместно со связями между таблицами образует структуру базы данных.

Сервер MySQLрасположен на компьютере с адресом 217.71.139.72 под управлением ОС Linux, учетное имя для подключения: userNM, где N – порядковый номер группы (задает преподаватель), M- Ваш номер по журналу, пароль сообщается преподавателем.

Для создания баз данных и таблиц в них, для настройки прав доступа к данным и манипулирования ими в СУБД MySQL применяется программа-монитор mysql. Запустив ее из командной строки, вы подключаетесь к СУБД с именем текущего пользователя операционной системы, если он имеет право на подключение. Для выбора базы данных применяется команда usedbname, где dbname - имя базы данных. Можно указать базу данных, имя пользователя и пароль при вызове монитора:

Mysql dbname -h hostname -u username -p password

Собственно пароль в командной строке лучше не вводить, а оставить ключ -p, тогда пароль будет запрошен дополнительно. Для выхода из монитора применяется команда exit или quit.

Базы данных пользователей и права доступа к ним создаются администратором. Студентам доступна лишь одна (userNM) база, операции создания и удаления таблиц, извлечения, вставки, добавления, удаления записей и изменения структуры таблиц.

Для создания таблиц базы данных применяют командуcreate table tablename (columns), где в скобках перечисляются названия, типы данных и другие атрибуты полей, например:

create table books (id int unsigned not null auto\_increment primary key, author char(30), title char(60), price float(5,2));

Для устранения ошибок при вводе длинных команд рекомендуется записывать их в текстовые файлы, запускаемые из командной строки:

mysql dbname -h hostname -u user -p < create.sql

Для просмотра списка созданных таблиц применяют команду show tables, а для просмотра структуры таблицы - команду describe tablename.

Для ввода, извлечения, изменения и удаления данных из таблиц применяют операторы структурированного языка запросов SQL: insert, select, update и delete. Для отбора данных по условию, уточнения изменяемых и удаляемых записей применяется конструкция where, для группирования - groupby, для упорядочивания - orderby. Для удаления таблиц служит команда drop table tablename.

Монитор mysql помнит в буфере ранее введенные команды. Для их вызова используйте стрелку вверх.

По окончании работы с базой данных выгрузите ее структуру и данные в текстовый файл и сохраните его. Он пригодится для восстановления испорченной или уничтоженной базы данных. Для сохранения в текстовых файлах структуры и содержимого базы данных применяют (по выходу из монитора mysql командой exit) утилиту mysqldump:

mysqldump -u user -p dbname >file.type

1. Методические указания
	1. Для подключения к серверу MySQL применяйте терминальные программы, поддерживающие протокол ssh, различные кодовые страницы и функциональные клавиши, например, telneat или putty.
	2. Базы данных пользователей и права доступа к ним создаются администратором. Вам требуется лишь попросить его задать пароль для доступа к своей базе данных.
	3. При работе с монитором mysql заканчивайте команды точкой с запятой.
	4. Длинные команды (создания таблиц, ввода данных в таблицу) лучше вводить в текстовый файл, передаваемый монитору через командную строку.
	5. Монитор mysql помнит в буфере ранее введенные команды. Для их вызова используйте стрелку вверх.
	6. Для просмотра содержимого таблиц в кодировке, отличающейся от кодировки сервера MySQL, удобно использовать утилиту netadmin.exe.
	7. Как правило, пользователи не имеют прав доступа к базе данных mysql. Для проверки назначенных Вами прав доступа к Вашей базе данных обратитесь к администратору (преподавателю).
	8. По окончании работы с базой данных выгрузите ее структуру и данные в текстовый файл и сохраните его. Он пригодится для восстановления испорченной или уничтоженной базы данных.
2. Порядок выполнения работы
	1. Спроектировать структуру базы данных согласно варианту, обеспечить третью нормальную форму.
	2. Используя выданные преподавателем учетные записи (userNM) и пароли, подключиться по протоколу ssh к серверу 217.71.139.72.
	3. Подключиться к MySQL-серверу, создать и заполнить таблицы, отладить и записать запросы на добавление, извлечение, изменение и удаление записей.
	4. Выгрузить в текстовые файлы структуру и содержимое базы данных с помощью утилиты mysqldump, сохранить на локальной машине и распечатать эти файлы.
3. Варианты заданий

**Вариант 1.** Спроектировать структуру базы данных о студентах для их распределения по местам практики: фамилия, год рождения, пол, группа, факультет, средний балл, место работы, город.

**Вариант 2.** Спроектировать структуру базы данных об автомобилях: номер, год выпуска, марка, цвет, состояние, фамилия владельца, адрес.

**Вариант 3.** Спроектировать структуру базы данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

**Вариант 4.** Спроектировать структуру базы данных о книгах, купленных библиотекой: название, автор, год издания, адрес автора, адрес издательства, цена, книготорговая фирма.

**Вариант 5.** Спроектировать структуру базы данных о сотрудниках, имеющих компьютер: фамилия, номер комнаты, название отдела, данные о компьютерах.

**Вариант 6.** Спроектировать структуру базы данных о заказах, полученных сотрудниками фирмы: фамилия, сумма заказа, наименование товара, название фирмы-клиента, фамилия заказчика.

**Вариант 7.** Спроектировать структуру базы данных об оценках, полученных студентами на экзаменах: фамилия, группа, предмет, номер билета, оценка, преподаватель.

**Вариант 8.** Спроектировать структуру базы данных о преподавателях кафедры: фамилия, должность, степень, номер комнаты, читаемые курсы.

**Вариант 9.** Спроектировать структуру базы данных об авторах web-сайта и их статьях: имя, адрес, учетная запись, пароль, тема, заголовок, текст статьи, иллюстрации.

**Вариант 10.** Спроектировать структуру базы данных о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков, их учетные записи и пароли.

**Вариант 11.** Спроектировать структуру базы данных технологических карт: деталь, вид обработки, длительность.

**Вариант 12.** Спроектировать структуру базы данных о документах, переданных исполнителям: исполнитель, документ, дата передачи, дата возврата.

**Вариант 13.** Спроектировать структуру базы данных об угнанных автомобилях: номер, марка, состояние (угнан/найден), фамилия владельца.

**Вариант 14.** Спроектировать структуру базы данных объявлений о квартирах: вид объявления (сдам/продам/сниму/куплю), адрес, количество комнат, дата, цена.

**Вариант 15.** Спроектировать структуру базы данных о книгах, выданных в библиотеке: название, читатель, дата выдачи, дата возврата.

**Вариант 16.** Спроектировать структуру базы данных о договорах: название фирмы–клиента, вид договора, срок действия.

**Вариант 17.** Спроектировать структуру базы данных о выдаче материальных ценностей сотрудникам: фамилия, номер комнаты, изделие, дата выдачи.

**Вариант 18.** Спроектировать структуру базы данных о галерее изображений: автор, имя файла, дата размещения.

**Вариант 19.** Спроектировать структуру базы данных о турпутевках: страна, курорт, стоимость, дата выезда.

**Вариант 20.** Спроектировать структуру базы данных о маршрутах транспорта: вид транспорта, номер, маршрут (начальная и конечная остановки), время в пути.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Структура базы данных в графическом виде.
	4. Протокол команд создания таблиц, ввода, отбора, изменения и удаления данных, задания прав доступа.
	5. Структура и содержимое базы данных.
	6. Выводы по работе.
2. Контрольные вопросы
	1. Варианты подключения к базе данных с помощью монитора MySQL.
	2. Таблицы системы привилегий MySQL.
	3. Последовательность контроля доступа к данным в MySQL.
	4. Задание прав доступа к данным в MySQL.
	5. Архитектура языка SQL.
	6. Создание и удаление баз данных и таблиц.
	7. Типы данных в MySQL.
	8. Индексы, их назначение и создание.
	9. Последовательности и автоинкрементирование в MySQL.
	10. Добавление данных в таблицу.
	11. Изменение и удаление данных.
	12. Запросы на извлечение данных.
	13. Объединения данных.
	14. Группирование и упорядочивание данных.
	15. Параметры утилиты mysqldump.

#### ЛАБОРАТОРНАЯ РАБОТА № 2

**PHP. Работа с файлами теневых посылок (cookies) и текстовыми файлами.**

1. Цель работы

Изучить методы получения данных из форм, методы чтения и записи теневых посылок и текстовых файлов.

1. Краткие теоретические сведения

Сценарии работы с теневой посылкой(cookie) предполагают получение данных формы, проверку наличия файла cookie, его считывание и запись новых данных, отображение предыдущего содержимого теневой посылки.

Для передачи данных Web-серверу используется отправка формы. В форме содержатся текстовые поля, переключатели, списки и т.п. Клиент размещает введенные данные в этих полях и пересылает пакет серверу. Процессом передачи формы управляют два атрибута тега <FORM>: METHOD и ACTION. Первый атрибут - METHOD - определяет, каким именно запросом данные пересылаются серверу. Атрибут может иметь два значения: POST и GET. POST диктует программе просмотра, что данные нужно поместить внутрь запроса, а GET пересылает данные как составную часть URL целевой страницы. Второй атрибут - ACTION - задает целевой файл сценария для обработки отправленных данных. Следующий код посылает данные формы сценарию DATA.PHP методом POST:

<FORM METHOD="POST" ACTION="data.php">

<P><INPUT TYPE="TEXT" NAME="Name"></P>

<P><INPUT TYPE="SUBMIT"></P></FORM>

Элемент формы с типом SUBMIT - это кнопка, нажатие которой пользователем заставляет программу просмотра упаковать данные формы и отправить их. Данные имеют вид пар *Поле=Значение*, отсылаемых серверу в формате открытого текста. На сервере эти данные можно вновь разобрать по полям и использовать в любых целях. Для разбора полученных от клиента данных, в зависимости от метода их отправки, в PHP используются глобальные ассоциативные массивы $HTTP\_POST\_VARS[] и $HTTP\_GET\_VARS[], ключами которых являются имена полей форм. Допустима сокращенная форма записи этих массивов: $\_POST и $\_GET соответственно. Например, следующие три строки кода вернут значение поля Name:

$a=$HTTP\_POST\_VARS["Name"];

$a=$\_POST["Name"];

$a=$Name;

Часто в теневых посылках сохраняется текущая дата, она определяется так:$date=date("d.m.Y");

Форматы даты/времени многочисленны и здесь не приводятся.

Зная имя теневой посылки и имена хранящихся в ней переменных, можно проверить наличие значения переменной (value) в указанной теневой посылке (visit):

if (isset($\_COOKIE["visit"])) $value =$visit;

Здесь приведена сокращенная форма обращения к глобальному ассоциативному массиву $HTTP\_COOKIE\_VARS[]. Новое значение переменной и срок действия теневой посылки устанавливается так:

setcookie("visit",$\_POST[value],time()+365\*24\*60\*60);

header("location:index.php");

Здесь не используются два дополнительных параметра теневой посылки, ограничивающих домены и пути к сценариям, которые имеют право чтения теневой посылки, а также параметр, требующий ее передачи по защищенному (протокол https) каналу связи.

Запись теневой посылки всегда производится до начала отображения результатов работы сценария, поэтому применяется перенаправление на файл сценария.

Пример работы сценария см. здесь: http://gun.cs.nstu.ru/web/labs/lab2/lab2\_1/.

При работе с текстовыми файлами, как правило, применяется построчная запись и чтение, поэтому заранее формируется строка с разделителями – символами табуляции:

$str = $date."\t".$\_POST["name"]."\t".$\_POST["textarea"]."\t". join($checkboxes,",")."\t".$\_POST["list"]."\n";

Файл может открываться на добавление ("a") (на запись: "w", на чтение: "r"), при отсутствии - создается:

@ $fp = fopen("1c3.txt", "a");

Символ @ в начале строки служит для подавления вывода в браузер сообщения об ошибке открытия файла (если он не может быть открыт или создан).

Файл открывается в текущем или указанном перед именем файла каталоге, на который сценариям даны соответствующие разрешения.

После открытия файла требуется заблокировать его от записи другими сценариями:

flock($fp, 2);

if (!$fp)

{

 echo "Ваша анкета не может быть обработана сейчас!";

 exit;

}

Существуют блокировки чтения (1), записи (2) и режим снятия блокировки (по окончании работы с файлом - 3):

flock($fp, 3);

В файл, открытый на запись и добавление, можно только записывать, в данном случае целесообразно писать построчно:

fwrite($fp, $str);

По окончании работы с файлом или при необходимости изменить режим работы с ним его необходимо закрыть:

fclose($fp);

При чтении файла необходимо отслеживать его окончание:

while(!feof($fp)){ . . . }

при этом чтение происходит построчно:

$a = fgets($fp, 100);

Другие функции чтения и записи приведены в справочнике по языку.

Пример работы сценария см. здесь: http://gun.cs.nstu.ru/web/labs/lab2/lab2\_2/

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M- номер студента по журналу, пароль сообщается преподавателем.
	3. При настройке клиента FTP необходимо установить пассивный режим.
	4. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	5. По окончании лабораторной работы сохраните документы сайта на сменный носитель.
2. Порядок выполнения работы
	1. Разработать Web-страницу с формой, содержащей указанные в вариантах заданий элементы управления.
	2. Разработать сценарий, проверяющий наличие теневой посылки на компьютере посетителя, при ее наличии выводящий данные из теневой посылки в браузер и отображающий ссылку на страницу с формой, и сценарий, считывающий данные формы и сохраняющий их и текущую дату в теневой посылке.
	3. Разработать сценарий, считывающий данные формы и сохраняющий их и текущую дату в текстовом файле с последующим отображением всех записей из файла в браузере.
	4. Продемонстрировать работу сценариев преподавателю.
	5. Сохранить файлы сценария на локальный компьютер.
3. Варианты заданий
	1. Для теневых посылок:

**Вариант 1.** Текстовое поле для ввода имени пользователя и кнопка отправки данных.

**Вариант 2.** Список для выбора языка отображения страницы и кнопка отправки данных.

**Вариант 3.** Текстовое поле для ввода даты рождения пользователя и кнопка отправки данных.

**Вариант 4.** Список для выбора кодировки страницы и кнопка отправки данных.

**Вариант 5.** Набор радиокнопок для выбора прав доступа (пользователь, модератор, администратор) и кнопка отправки данных.

**Вариант 6.** Набор флажков для выбора допустимых режимов работы (чтение, запись, изменение, удаление) и кнопка отправки данных.

**Вариант 7.** Набор радиокнопок для выбора кодировки страницы и кнопка отправки данных.

**Вариант 8.** Набор флажков для выбора управляемых компонентов сайта (текст, графические данные, стили оформления) и кнопка отправки данных.

**Вариант 9.** Текстовое поле типа password для ввода пароля пользователя и кнопка отправки данных.

**Вариант 10.** Список для выбора текста уведомления и кнопка отправки данных.

**Вариант 11.** Набор радиокнопок для выбора шрифта текста и кнопка отправки данных.

**Вариант 12.** Набор флажков для выбора стиля текста (размер, жирность, курсив, цвет) и кнопка отправки данных.

**Вариант 13.** Текстовое поле для ввода e-mail пользователя и кнопка отправки данных.

**Вариант 14.** Список для выбора интервала уведомления и кнопка отправки данных.

**Вариант 15.** Набор радиокнопок для выбора времени отправки уведомления и кнопка отправки данных.

**Вариант 16.** Набор флажков для выбора способа уведомления (звонок, SMS, e-mail) и кнопка отправки данных.

**Вариант 17.** Текстовое поле для ввода приветствия пользователя и кнопка отправки данных.

**Вариант 18.** Список для выбора места размещения приветствия и кнопка отправки данных.

**Вариант 19.** Набор радиокнопок для выбора места размещения приветствия и кнопка отправки данных.

**Вариант 20.** Набор флажков для выбора степени детализации сообщений об ошибках (замечание, предупреждение, ошибка, фатальная ошибка) и кнопка отправки данных.

* 1. Для текстовых файлов:

**Вариант 1.** Текстовое поле для ввода имени пользователя, текстовое поле для ввода даты рождения пользователя и кнопка отправки данных.

**Вариант 2.** Список для выбора темы обсуждения, поле TEXTAREAдля ввода сообщения и кнопка отправки данных.

**Вариант 3.** Текстовое поле для ввода имени пользователя, поле TEXTAREAдля ввода комментария и кнопка отправки данных.

**Вариант 4.** Список для выбора наименования товара, текстовое поле для ввода количества товара и кнопка отправки данных.

**Вариант 5.** Набор радиокнопок для выбора дня недели, список для выбора интервала времени для связи и кнопка отправки данных.

**Вариант 6.** Набор флажков для выбора дня недели, список для выбора интервала времени для связи и кнопка отправки данных.

**Вариант 7.** Набор радиокнопок для выбора кодировки сообщения, поле TEXTAREAдля ввода сообщения и кнопка отправки данных.

**Вариант 8.** Набор флажков для выбора вариантов оформления текста, поле TEXTAREAдля ввода комментария и кнопка отправки данных.

**Вариант 9.** Текстовое поле для ввода имени пользователя, поле TEXTAREAдля ввода приветствия и кнопка отправки данных.

**Вариант 10.** Список для выбора стиля приветствия, текстовое поле для ввода размера шрифта и кнопка отправки данных.

**Вариант 11.** Набор радиокнопок для выбора стиля приветствия, список для выбора размера шрифта и кнопка отправки данных.

**Вариант 12.** Набор флажков для выбора стиля приветствия, список для выбора размера шрифта для связи и кнопка отправки данных.

**Вариант 13.** Текстовое поле для ввода имени пользователя, поле TEXTAREAдля ввода сайтов для управления и кнопка отправки данных.

**Вариант 14.** Список для выбора прав доступа (пользователь, модератор, администратор), текстовое поле типа password для ввода пароля и кнопка отправки данных.

**Вариант 15.** Набор радиокнопок для выбора кодировки страницы, список для выбора шрифта и кнопка отправки данных.

**Вариант 16.** Набор флажков для выбора допустимых режимов работы (чтение, запись, изменение, удаление), список для выбора управляемого сайта и кнопка отправки данных.

**Вариант 17.** Текстовое поле для ввода имени туриста, поле TEXTAREAдля ввода пожеланий по формам отдыха и кнопка отправки данных.

**Вариант 18.** Список для выбора категории гостиницы, текстовое поле для ввода дней проживания и кнопка отправки данных.

**Вариант 19.** Набор радиокнопок для выбора страны посещения, список для выбора курортаи кнопка отправки данных.

**Вариант 20.** Набор флажков для выбора экскурсий, список для выбора времени экскурсий и кнопка отправки данных.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Листинг сценариев.
	4. Выводы по результатам работы.
2. Контрольные вопросы
	1. Атрибуты тега <FORM>.
	2. Элементы управления, которые можно расположить в форме.
	3. Варианты получения данных из формы.
	4. Структура теневой посылки.
	5. Назначение параметров теневой посылки.
	6. Особенности записи теневой посылки.
	7. Режимы открытия текстовых файлов.
	8. Полный перечень параметров функции fopen.
	9. Функции записи в текстовый файл.
	10. Функции чтения из текстового файла.
	11. Назначение и параметры функции flock.
	12. Другие функции открытия файлов, кроме функции fopen.
	13. Функции проверки наличия, определения размера, удаления файлов.
	14. Функции форматного ввода из и вывода в файлы.

#### ЛАБОРАТОРНАЯ РАБОТА № 3

**PHP. Работа с базой данных MySQL.**

1. Цель работы

Изучить синтаксис языка, освоить процедуру подключения к базе данных, реализацию операций выборки, вставки, изменения и удаления записей в базе данных средствами языка сценариев PHP.

1. Краткие теоретические сведения

Для подключения к базе данных используется функция int mysql\_pconnect ([stringhost[:port], [stringuser], [stringpassword]);

Требуется указать имя узла (host), на котором размещен сервер MySQL, имя пользователя (user), чтобы войти в него, и пароль (password):

@ $db = mysql\_pconnect("localhost", "bookorama", "bookorama");

При успехе функция вернет идентификатор связи с базой данных, а при неудаче — значение false. Результат стоит проверить:

if (!$db) {echo "Error: Could not connect to database."; exit; }

Работая с MySQL, следует указать, какая база данных нужна, с помощью функции mysql\_select\_db("books");

При этом будет использоваться последнее открытое соединение. Если открытое соединение не существует, оно открывается по умолчанию, как при вызове mysql\_connect().

Далее необходимо настроить запрос: $query = "select \* from books";

Запрос, отправляемый в MySQL, не требует в конце точки с запятой, в отличие от запроса, который вводится в среде монитора MySQL.

Теперь можно выполнить запрос: $result = mysql\_query ($query);

В функцию можно также передать и соединение с базой данных (в нашем случае $db). Если его не указать, то будет использовано последнее открытое соединение. Если такового нет, функция откроет соединение точно так же, как при выполнении mysql\_connect().

Вместо этого можно воспользоваться функцией: int mysql\_db\_query(string database, string query, [int database\_connection]);

В каком-то смысле это комбинация функций mysql\_select\_db() и mysql\_query(). Обе функции возвращают идентификатор результата в случае успеха и значение false в случае неудачи. Идентификатор результата следует сохранить, это ключ доступа к строкам, возвращенным запросом, которых может быть нуль, одна и более.

Функция mysql\_num\_rows() сообщает количество строк, которые возвращает запрос. В нее следует передать идентификатор результата:

$num\_results = mysql\_num\_rows($result);

Зная их количество, можно организовать цикл:

for ($i=0; $i < num\_results; $i++) { // обработка результатов }

На каждой итерации цикла можно вызвать mysql\_fetch\_array(). Эта функция берет каждую строку из списка результата и возвращает ее в виде ассоциативного массива, с ключом как именем атрибута и значением как соответствующим значением массива.

$row = mysql\_fetch\_array($result);

Имея $row в ассоциативном массиве, можно пройти каждое поле и отобразить его.

Существуют несколько вариантов получения результата из идентификатора результата. Вместо ассоциативного массива можно воспользоваться нумерованным массивом:

$row = mysql\_fetch\_row($result);

Значения атрибутов будут храниться в каждом порядковом значении $row[0], $row[l].

С помощью функции mysql\_fetch\_object() можно выбрать строку внутрь объекта:

$row = mysql\_fetch\_object ($result);

После этого к атрибутам можно получить доступ через $row->title, $row->author.

Каждый из этих вариантов подразумевает выборку строки за раз. Другой вариант — получить доступ, используя mysql\_result(). Для этого потребуется указать номер строки (от 0 до количества строк минус 1) и название поля, например:

$row = mysql\_result($result, $i, "title");

Название поля можно задать в виде строки (либо в форме "title" либо в форме “books.title”) или номером (как в mysql\_fetch\_row()). He стоит смешивать mysql\_result() с другими функциями выборки. Строчно-ориентированные функции выборки намного более эффективны, нежели mysql\_result(), так что лучше использовать одну из них.

Для закрытия непостоянного соединения применяется функция:

mysql\_close(database\_connect±on);

Однако с завершением выполнения сценария соединение закроется автоматически.

Если во время выполнения сценария возникают проблемы, связанные с нехваткой памяти, пригодится функция mysql\_free\_result(). Она вызывается с идентификатором результата:

mysql\_free\_result($result);

и освобождает память, занимаемую результатом. Очевидно, что до обработки результата эта функция вызываться не должна.

Внесение новой информации очень похоже на получение существующей. Только в данном случае вместо SELECT будет использоваться INSERT:

$query="insert into books

values('".$id."','".$author."','".$title."')";

$result = mysql\_query($query);

if ($result) echo mysql\_affected\_rows()."row(s) inserted.";

Аналогично, для изменения и удаления записей применяются запросы UPDATE и DELETE. При редактировании изменяться должно лишь содержимое таблицы-связки, содержимое же таблиц справочников рекомендуется отображать в виде списков.

Пример работы сценариев см. здесь: http://gun.cs.nstu.ru/web/labs/lab3/.

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M- номер студента по журналу, пароль сообщается преподавателем.
	3. При настройке клиента FTP необходимо установить пассивный режим.
	4. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	5. При написании сценариев обращайте внимание на кодировку русского текста, правильность написания HTML-тегов, имен переменных и функций PHP, парность открывающих и закрывающих кавычек в операторе echo.
	6. При отладке сценариев обновляйте содержимое окна браузера при нажатой клавише Shift. Для запрета кэширования web-страниц на локальном диске формируйте в сценариях заголовок header(expires='-1d').
	7. По окончании лабораторной работы запишите документы сайта на сменный носитель для сохранения и распечатки.
2. Порядок выполнения работы
	1. Разработать алгоритмы сценариев извлечения, добавления, изменения и удаления записей в базе данных согласно варианту.
	2. Создать файл index.php сценария извлечения записей из базы данных в соответствии со структурой своей базы данных. Из браузера обратиться по адресу http://217.71.139.72/~userNM/index.php, убедиться в наличии прав доступа к базе данных.
	3. На локальном компьютере создать сценарии, реализующие операции извлечения по условию, добавления, изменения и удаления записей, сценарий меню для выбора операции, по протоколу ftp скопировать их на сервер 217.71.139.72.
	4. Отладить сценарии, продемонстрировать их работоспособность преподавателю.
	5. Сохранить на локальной машине и распечатать файлы отлаженных сценариев.
3. Варианты заданий

Задания различаются структурой баз данных, реализованных в соответствии с заданиями к лабораторной работе №1. Обязательна реализация связанных запросов на извлечение данных, запросов на поиск (по параметру), добавления, изменения (редактирования) и удаления данных в/из таблицы-связки.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Структура базы данных в графическом виде.
	4. Листинги сценариев.
	5. Выводы по работе.
2. Контрольные вопросы
	1. Способы внедрения кода PHP в HTML.
	2. Применение конкатенации строк при формировании запросов к базе данных.
	3. Процедура подключения к базе данных из PHP.
	4. Типовые ошибки при подключении к базе данных и способы их перехвата.
	5. Формирование и выполнение запроса к базе данных.
	6. Перемещение по записям, извлеченным из базы данных.
	7. Извлечение данных из полей записи.
	8. Извлечение атрибутов столбцов таблицы базы данных.
	9. Отличие результатов запросов на извлечение и добавление/удаление записей.
	10. Методы смены кодировки данных, извлекаемых из базы данных.
	11. Всегда ли нужно закрывать соединение с базой данных? Когда именно?
	12. Как проще всего после изменения записей отобразить результат?
	13. В каких случаях и какими методами необходимо прерывать работу сценария взаимодействия с базой данных.
	14. Когда и зачем необходимо вызывать функцию mysql\_free\_result()?
	15. Можно ли реализовать все операции с базой данных в одном сценарии? Что для этого потребуется?

#### ЛАБОРАТОРНАЯ РАБОТА № 4

**PHP. Разграничение прав доступа к сайту.**

1. Цель работы

Изучить синтаксис языка, методы аутентификации пользователей и управления сеансами, освоить процедуру разграничения доступа к страницам сайта средствами языка сценариев PHP.

1. Краткие теоретические сведения

Просьба к пользователю доказать свою личность называется аутентификацией. Обычный метод аутентификации в Web — это требование к посетителям предоставить уникальное имя пользователя и пароль. Аутентификация обычно используется для разрешения или запрещения доступа к определенным страницам или ресурсам.

Например, вводятся такие группы доступа, как гости, пользователи, администраторы. Для гостей аутентификация не требуется, а данные доступны для просмотра и поиска. Пользователям и администраторам необходима аутентификация, после которой пользователи получат права добавления и изменения записей в базе данных, а администраторы – и право удаления записей.

Самый простой способ аутентификации - запрос имени пользователя и пароля через форму и сравнение их с данными, хранящимися прямо в сценарии, текстовом файле или базе данных.

Для защиты группы сценариев можно использовать базовую аутентификацию

if ($PHP\_AUTH\_USER != "user" || $PHP\_AUTH\_PW != "pass")

{ header('WWW-Authenticate: Basic realm="Restricted area"');

 header("HTTP/1.0 401 Unauthorized");

echo "<h1>Go Away!</h1>";}

else {echo "<h1>Here it is!</h1>"};

Но наилучший вариант - управление сценариями. Для запуска сеанса в РНР используется уникальный идентификатор сеанса, представляющий собой зашифрованное случайное число. Идентификатор сеанса генерируется РНР и сохраняется на стороне клиента в течение всего времени жизни сеанса. Для хранения идентификатора сеанса используется либо cookie-набор на компьютере пользователя, либо URL.

Прежде чем можно будет воспользоваться функциональными возможностями сеанса, следует запустить сам сеанс. Существует три способа сделать это.

В первом способе сценарий начинается с вызова функции session\_start();

Она проверяет, существует ли идентификатор текущего сеанса. Если нет, она его создает. Если да, то она загружает зарегистрированные переменные сеанса, чтобы они стали доступными для использования.

Второй способ заключается в том, что сеанс запускается при попытке зарегистрировать переменные сеанса. Например, для регистрации переменной $myvar применяется код:

$myvar = 5; session\_register("myvar");

Третий способ запустить сеанс — задать установки РНР, при которых сеанс будет запускаться автоматически. Для этого установите опцию session.auto\_start в файле php.ini.

Получить доступ к переменной сеанса можно через ассоциативный массив: $HTTP\_SESSION\_VARS["myvar"] (сокращенная форма записи: $\_SESSION ["myvar"]). Переменные сеанса не могут быть перезаписаны данными GET или POST. Проверить, является ли переменная зарегистрированной переменной сеанса, можно так: $result = session\_is\_registered("myvar");

После окончания работы с переменной сеанса ее регистрацию можно отменить, воспользовавшись функцией session\_unregister("myvar");

Она за один раз может отменить регистрацию только одной переменной сеанса (в противоположность session\_register()). Для отмены регистрации всех переменных текущего сеанса можно обратиться к session\_unset().

По завершении сеанса сначала потребуется отменить регистрацию всех переменных, а затем для обнуления идентификатора сеанса вызвать session\_destroy();

Пример см. здесь: http://gun.cs.nstu.ru/web/labs/lab4/. Для доступа используются учетные записи администратора tratata с паролем secretpassword и модератора user с паролем test. Ссылка на модуль администрирования представляет собой пример курсовой работы.

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M- номер студента по журналу, пароль сообщается преподавателем.
	3. Лабораторная работа выполняется в отдельной папке на сервере, куда копируются сценарии лабораторной работы №3 для модификации.
	4. При настройке клиента FTP необходимо установить пассивный режим.
	5. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	6. При написании сценариев обращайте внимание на кодировку русского текста, правильность написания HTML-тегов, имен переменных и функций PHP, парность открывающих и закрывающих кавычек в операторе echo.
	7. При отладке сценариев обновляйте содержимое окна браузера при нажатой клавише Shift. Для запрета кэширования web-страниц на локальном диске формируйте в сценариях заголовок header(expires='-1d').
	8. По окончании лабораторной работы запишите документы сайта на сменный носитель для сохранения и распечатки.
2. Порядок выполнения работы
	1. Разработать таблицы пользователей и групп доступа, добавить их в базу данных из лабораторной работы №1, заполнить не менее чем двумя записями.
	2. Модифицировать файл index.php сценария извлечения записей из базы данных так, чтобы неавторизованному пользователю была доступна только функция извлечения данных и поиск по параметру. Разместить в нем ссылку на сценарий авторизации. Из браузера обратиться по адресу http://217.71.139.72/~userNM/index.php, убедиться в наличии минимальных прав доступа к базе данных.
	3. Разработать сценарий авторизации пользователей, выполняющий перенаправление на файл п.4.2 при успешной авторизации, модифицировать файл п.4.2 для отображения учетной записи авторизованного пользователя, ссылки на файл окончания авторизации, отображения ссылок на сценарии добавления, изменения, удаления записей в зависимости от группы доступа авторизованного пользователя. Из браузера обратиться по адресу http://217.71.139.72/~userNM/index.php, выполнить авторизацию, убедиться в наличии прав доступа к базе данных, соответствующих группе доступа авторизованного пользователя.
	4. Модифицировать сценарии, реализующие операции добавления, изменения и удаления записей, для обеспечения доступа к ним только авторизованных пользователей с соответствующей группой доступа.
	5. Разработать сценарий окончания авторизации пользователей, выполняющий перенаправление на файл п.4.2 при успешном окончании авторизации
	6. Отладить сценарии, продемонстрировать их работоспособность преподавателю.
	7. Сохранить на локальной машине и распечатать файлы отлаженных сценариев.
3. Варианты заданий

Задания различаются структурой базы данных, регистрационными записями и паролями пользователей, имеющих разные права доступа к ней, реализованными в соответствии с заданиями к лабораторной работе №1, и являются модификацией лабораторной работы №3.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Структура таблицы пользователей и групп доступа.
	4. Листинги сценариев.
2. Контрольные вопросы
	1. Методы аутентификации пользователей, их достоинства и недостатки.
	2. Сеансы, их назначение.
	3. Способы запуска сеанса.
	4. Где могут храниться сеансовые переменные?
	5. Как проверить, является ли переменная сеансовой?
	6. Функции управления сеансами.
	7. Получение идентификатора сеанса и способы его передачи из одного сценария в другой.
	8. Сеансовые директивы в файле php.ini.

#### ЛАБОРАТОРНАЯ РАБОТА № 5

**Работа с графикой в PHP. Построение диаграмм.**

1. Цель работы

Изучить функции работы с изображениями, освоить построение диаграмм средствами языка сценариев PHP.

1. Краткие теоретические сведения

При разработке web-сайтов часто возникает необходимость формировать изображения «на лету» - отображение результатов голосования, динамики запросов, статистики посещений. В PHP нет ограничений на формат вывода. Он может также использоваться для создания изображений и манипуляций с файлами изображений различных форматов, включая gif, png, jpg, wbmp и xpm. Что особенно удобно: PHP может выводить поток изображения непосредственно в браузер. Однако, смешивать форматы вывода невозможно, поскольку формат задается в заголовке ответа HTTP:

header ("Content-type: image/png");

Для отображения динамически сформированного изображения в тексте применяется известный тег HTML:

<img src=graph.php>

Само же формирование изображения начинается вызовом функции

$im=@imagecreate($size\_x, $size\_y) or die ("Cant Initialize GD");

аргументы которой – размер создаваемого изображения в пикселях. Функция возвращает ссылку на созданное пустое изображение, а в случае неудачи создания работа сценария прерывается. Символ @ служит для подавления выдачи стандартного сообщения об ошибке в браузер.

Далее необходимо определить цвета, используемые в изображении:

$magenta\_color=imagecolorallocate ($im,255,0,255);

$line=imagecolorallocate ($im, 64, 16, 0);

$transparent\_color=imagecolortransparent($im,$magenta\_color);

Вызов последней функции обеспечивает прозрачность цвета для его использования в качестве фонового.

В созданном изображении выделенными цветами рисуются графические примитивы – линии, сектора, прямоугольники, эллипсы.

int imageline (im, int x1, int y1, int x2, int y2, int col);

int imagearc (im, int cx, int cy, int w, int h, int s, int e, int col, int style);

int imagerectangle (im, int x1, int y1, int x2, int y2, int col);

int imageellipse (im, int cx, int cy, int w, int h, int col);

Замкнутые фигуры, в том числе все изображение, могутиметь заливку:

int imagefilledrectangle (im, int x1, int y1, int x2, int y2, int col);

bool imagefill (im, int $x , int $y , int $col);

В изображении может присутствовать текст:

bool imagestring (im, int $font, int $x, int $y, string $string, int $color);

array imagettftext (im, float $size, float $angle, int $x, int $y, int $color, string $fontfile, string $text);

Шрифт текста может быть встроенным (для функции imagestring() 1-5 в зависимости от размера), типа TrueType (для функции imagettftext()) или предварительно загруженным специальным типа gdf, сконвертированном из TrueType:

int imageloadfont ( string $file)

Может возникнуть проблема с выводом текста на русском языке – встроенные шрифты (для функции imagestring()) не поддерживают кириллицу, и даже при указании шрифта функции imagettftext() кириллица может выводиться в неверной кодировке.

Есть несколько способов решения этой проблемы. Самый простой – задать кодировку явно:

setlocale(LC\_ALL,"ru\_RU.WIN1251");

или конвертировать строку с символами кириллицы в кодировку UTF-8:

$text = iconv("windows-1251", "UTF-8", "наш текст");

Осталось вывести полученное изображение в нужном графическом формате в браузер или файл:

imagepng ($im[, string $filename ]);

Пример динамически изменяющейся диаграммы, формируемой по записям из базы данных, см. здесь: http://gun.cs.nstu.ru/web/labs/lab5/

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M - номер студента по журналу, пароль сообщается преподавателем.
	3. Лабораторная работа выполняется в отдельной папке на сервере, куда копируются сценарии лабораторной работы №3 для модификации.
	4. При настройке клиента FTP необходимо установить пассивный режим.
	5. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	6. При написании сценариев обращайте внимание на кодировку русского текста, правильность написания HTML-тегов, имен переменных и функций PHP, парность открывающих и закрывающих кавычек в операторе echo.
	7. При отладке сценариев обновляйте содержимое окна браузера при нажатой клавише Shift. Для запрета кэширования web-страниц на локальном диске формируйте в сценариях заголовок header(expires='-1d').
	8. По окончании лабораторной работы запишите документы сайта на сменный носитель для сохранения и распечатки.
2. Порядок выполнения работы
	1. Модифицировать файл index.php сценария извлечения записей из базы данных для отображения результатов работы сценария, формирующего диаграмму.
	2. Сделать копию файла index.php с другим именем, ссылка на который помещена в файл index.php, модифицировать запрос извлечения записей из базы данных для их группирования, сформировать массивы данных и подписей к ним.
	3. Реализовать в этом же файле построение гистограммы с подписями под столбцами и выводом значений столбцов над ними.
	4. Если подписи к столбцам данных хранятся в базе данных на русском языке, организовать загрузку файла подходящего шрифта. Изображения осей абсцисс и ординат и их подписи не обязательны.
	5. Отладить сценарии, убедиться в динамическом перестроении гистограммы при изменении записей в базе данных, продемонстрировать их работоспособность преподавателю.
	6. Сохранить на локальной машине и распечатать файлы отлаженных сценариев.
3. Варианты заданий

Задания различаются структурой базы данных, реализованной в соответствии с заданиями к лабораторной работе №1, и являются модификацией лабораторной работы №3.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. SQL-запрос на извлечение и группирование отображаемых в диаграмме данных.
	4. Листинги сценариев.
	5. Выводы по работе.
2. Контрольные вопросы
	1. Как переключить PHP на вывод графических данных?
	2. Как создать новое пустое изображение, изображение из существующего файла?
	3. Как выделить цвета для изображения, какими они могут быть?
	4. Какие графические примитивы могут быть помещены в изображение?
	5. Какие имеются функции для формирования надписей, в чем их различие?
	6. Как решить проблему отображения надписей на русском языке?
	7. Как отобразить/сохранить полученное изображение?
	8. Как определить реальный тип изображения, созданного из файла?
	9. Какие существуют функции для обработки изображения, созданного из файла?
	10. Какие существуют функции для получения свойств изображений?

#### ЛАБОРАТОРНАЯ РАБОТА № 6

**Подключение компонент сторонних разработчиков на примере CKEditor, KCAPTCHA.**

1. Цель работы

Изучить особенности подключения к сценариям PHP компонент сторонних разработчиков, освоить применение автоматизированных тестов Тьюринга.

1. Краткие теоретические сведения

При разработке web-сайтов часто требуются типовые решения, такие, как online-редактор HTML-кода, автоматизированный тест Тьюринга для защиты от спама в досках объявлений, и другие. Подобные модули несложно разработать самостоятельно, однако существуют готовые решения с открытым исходным кодом. Рассмотрим подключение и применение двух наиболее популярных решений.

Текстовый редактор CKeditor обладает большим количеством разнообразных функций и поддерживается большинством современных браузеров. Он оформлен в стиле MS Word, что делает обращение с ним более привычным.

CKeditor можно скачать с сайта разработчика <http://ckeditor.com>. Существует его реализации почти для десятка языков программирования, но рассмотрим только на php. Распакованный архив копируется в папку сайта, после чего необходимо подключить файл:

<?php

include\_once("fckeditor/fckeditor.php") ;

?>

Теперь можно использовать редактор. Для этого просто добавляем следующий код внутри тега <FORM>:

<form action="sampleposteddata.php" method="post" target="\_blank">

<?php

$oFCKeditor = new FCKeditor('FCKeditor1') ;

$oFCKeditor->BasePath = '/fckeditor/' ;

$oFCKeditor->Value = '<p>This is some <strong>sample text</strong>.You are using <a href="http://www.fckeditor.net/">FCKeditor</a>.</p>';

$oFCKeditor->Create();

?>

<br><input type="submit" value="Submit"></form>

Отформатированный в окне редактора текст передается методом POST и может быть сохранен в файле или базе данных.

Если редактор размещается на странице, вход на которую не требует авторизации, то гостевая книга или доска объявлений окажутся быстро переполненными спамом, рассылаемом соответствующими программами. Для того, чтобы разрешить добавление записей только человеком, требуется провести автоматизированный обратный тест Тьюринга, известный также, как CAPTCHA (от [англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Completely Automated Public Turing test to tell Computers and Humans Apart — полностью автоматизированный публичный [тест Тьюринга](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0) для различения компьютеров и людей). Иными словами, это задача, которую легко решает человек, но которую не может выполнить компьютер. Чаще всего CAPTCHA выглядит как тем или иным образом зашумленное случайное число, слово или иная надпись, которую пользователю нужно прочитать и ввести прочитанный результат, хотя существуют и другие алгоритмы.

Проект KCAPTCHA — это готовое решение, написанное на языке PHP, которое вы можете бесплатно скачать и установить на свой сайт для защиты от спама и флуда.

Проект KCAPTCHA ставит перед собой цель предложить программисту решение с одной стороны весьма защищенное, с другой — максимально малотребовательное к ресурсам и конфигурации хостинга. Пример:

<img src="kcaptcha/code.php?<?php echo session\_name()?>=<?php echo session\_id()?>"></p>

Принцип действия: скрипт стартует сессию и записывает в нее под именем $\_SESSION['captcha\_keystring'] случайным образом сгенерированную строку, после чего выдает изображение, содержащее эту самую строку в зашумленном виде. При проверке пользовательского ввода остается только прочитать из сессии кодовую строку и сравнить с тем, что ввел пользователь. Системные требования: PHP версии 4.0.6 и выше с поддержкой GD версии 2.

В комплект входит набор растровых шрифтов, так что скрипт, скорее всего, будет сразу готов к работе, не требуя установки дополнительных компонент.

Можно настраивать цвета и набор символов, применяемые при создании изображения. Все настройки, касающиеся KCAPTCHA, располагаются в файле kcaptcha\_config.php. Здесь можно задать вид отображаемого проверочного кода.

Скачать [KCAPTCHA 2.0](http://www.captcha.ru/kcaptcha.zip) можно здесь: http://www.captcha.ru/kcaptcha.zip

Пример подключения онлайн-редактора и сценария формирования кода подтверждения см. здесь: http://gun.cs.nstu.ru/web/labs/lab6/ /

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M- номер студента по журналу, пароль сообщается преподавателем.
	3. Лабораторная работа выполняется в отдельной папке на сервере, куда копируются сценарии лабораторной работы №2 для модификации.
	4. При настройке клиента FTP необходимо установить пассивный режим.
	5. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	6. При написании сценариев обращайте внимание на кодировку русского текста, правильность написания HTML-тегов, имен переменных и функций PHP, парность открывающих и закрывающих кавычек в операторе echo.
	7. При отладке сценариев обновляйте содержимое окна браузера при нажатой клавише Shift. Для запрета кэширования web-страниц на локальном диске формируйте в сценариях заголовок header(expires='-1d').
	8. По окончании лабораторной работы запишите документы сайта на сменный носитель для сохранения и распечатки.
2. Порядок выполнения работы
	1. Модифицировать файл сценария отображения формы подключением модуля редактора и модуля генерации кода подтверждения.
	2. Модифицировать сценарий записи в файл содержимого формы проверкой сеансовой переменной с кодом подтверждения, при несовпадении сформировать ссылку на файл с формой.
	3. Отладить сценарии, убедиться в корректном отображении окна редактора и изображения с кодом подтверждения, продемонстрировать их работоспособность преподавателю.
	4. Сохранить на локальной машине и распечатать файлы отлаженных сценариев.
3. Варианты заданий

Задания различаются списком полей формы, реализованной в соответствии с заданиями к лабораторной работе №2, и являются модификацией лабораторной работы №2.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Листинги сценариев.
	4. Выводы по работе.
2. Контрольные вопросы
	1. Какие еще подключаемые редакторы существуют? В чем их отличия от использованного?
	2. Какая технология программирования использована при написании CKEditor? Из чего это следует?
	3. Всеми ли браузерами поддерживается CKeditor?
	4. Возможна ли настройка отображаемых в редакторе кнопок? Если да, то каким образом?
	5. Дайте определение классического теста Тьюринга. Чем от него отличается CAPTCHA?
	6. Разберите строку подключения [KCAPTCHA 2.0](http://www.captcha.ru/kcaptcha.zip).
	7. Каков принцип действия пакета [KCAPTCHA 2.0](http://www.captcha.ru/kcaptcha.zip)?
	8. Какие алгоритмы проверки входят в пакет [KCAPTCHA 2.0](http://www.captcha.ru/kcaptcha.zip)? Какие еще алгоритмы Вы можете предложить?
	9. Каковы системные требования для пакета [KCAPTCHA 2.0](http://www.captcha.ru/kcaptcha.zip)?
	10. Как и где настроить вид отображаемого проверочного кода?

#### ЛАБОРАТОРНАЯ РАБОТА № 7

**JavaScript. Объектная модель документа и работа с ней. Применение библиотек семейства JQuery.**

1. Цель работы

Изучить особенности клиентского языка программирования **JavaScript**, объектную модель документа HTML и работу с ней в **JavaScript**, освоить применение библиотек семейства JQuery.

1. Краткие теоретические сведения

Для обработки данных форм перед их передачей в CGI-программу часто применяют язык сценариев JavaScript. Он располагает объектной моделью документа, позволяющей получать значение любого атрибута любого тега и набором функций, позволяющих получить значение атрибута тега и проверить его тип. Событийная модель позволяет организовать реакцию на действия пользователя, а методы, присущие объектам в объектной модели документа, позволяют манипулировать свойствами объектов. Можно перехватить отправку формы

<input type="Submit" value="Послать" onClick="OpenWin()">

и проверить заполнение ее полей в указанной функции с отображением результатов в отдельном окне:

myWin1=window.open("", "TestWindow", "width=500, height=400, status=no, toolbar=no, resizable=yes, scrollbars=no, menubar=no");

name=parent.ss.name.value;

Здесь parent - ссылка на родительский документ, ss - имя формы в документе, name - имя поля формы, value - значение поля. Значение можно проверить на пустоту или соответствие типа и формата ввода, после чего отправить данные:

if (name) {

myWin1.document.write("<br><input type=button value='Отправить' onclick=send();>");

}

или выдать сообщение об ошибке и вернуться к родительскому документу:

else {

if (name=="") alert("Не указано имя!");

myWin1.focus();

myWin1.document.write("<br><input type=button value='Исправить' onclick='self.close();'");

}

Однако при большом количестве и разнообразии полей форм организация подобной обработки требует много усилий. Для решения таких типовых задач существуют готовые свободно распространяемые библиотеки сценариев на JavaScript под обобщенным названием JQuery. Одна из них, JQuery.Validate, позволяет организовать подобную обработку полей форм путем подключения библиотек:

<script type="text/javascript" src="js/jquery-1.2.6.js"></script>

<script type="text/javascript" src="js/jquery.form.js"></script>

<script type="text/javascript" src="js/jquery.validate.js"> </script>

отправки данных при корректности заполнения полей формы:

<scripttype="text/javascript">

$(document).ready(function(){

// готовим опции для метода ajaxSubmit плагина jquery.form.js

var options = {

 target: "#output",

 url: "1c2.php",

 type: "post",

 success: function() {

 alert("Спасибо за комментарий!");

 },

timeout: 3000

 };

проверки корректности заполнения полей форм по соответствующим правилам:

$("#myForm").validate({

 focusInvalid: false,

 focusCleanup: true,

 rules: {

 Name: {

 required: true,

 minlength: 2,

 maxlength: 12

 },

 Email: {

 required: true,

 email: true

 },

 Password: {

 required: true,

 rangelength: [6, 24]

 },

 ConfirmPassword: {

 required: true,

 rangelength: [6, 24],

 equalTo: "#Password"

 },

 Url: {

 url: true,

 required: true

 }

}

и формирования сообщений об ошибках, выводящихся непосредственно рядом с некорректно заполненными полями:

messages: {

Name: {

required: "Укажите свое имя, пожалуйста!",

minlength: "Не менее 2 символов",

maxlength: "Не более 12 символов"

 },

Email: {

required: "Нужно указать email адрес",

email: "Email адрес должен быть корректным"

},

 Password: {

 required: "Укажите пароль!",

 rangelength: "Пароль - от 6 до 24 символов"

 },

ConfirmPassword: {

 required: "Подтвердите пароль!",

 rangelength: "Пароль - от 6 до 24 символов",

 equalTo: "Подтверждение пароля не принято!"

 },

Url: {

 required: "Нужно указать URL",

 url: "Введите корректный URL"

 }

 }

},

 errorPlacement: function(error, element) {

 var er = element.attr("name");

 error.appendTo( element.parent().find("label[@for='" + er + "']").find("em") );

}

});

});

Настройка правил проверки и сообщений при их нарушении достаточно ясны из примера.

Пример проверки заполнения полей форм средствами JavaScript и библиотеки JQuery.Validate см. здесь: http://gun.cs.nstu.ru/web/labs/lab7.

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M - номер студента по журналу, пароль сообщается преподавателем.
	3. Лабораторная работа выполняется в отдельной папке на сервере, куда копируются сценарии лабораторной работы №2 для модификации.
	4. При настройке клиента FTP необходимо установить пассивный режим.
	5. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	6. При написании сценариев обращайте внимание на кодировку русского текста, правильность написания HTML-тегов, имен переменных и функций PHP, парность открывающих и закрывающих кавычек в операторе echo.
	7. При отладке сценариев обновляйте содержимое окна браузера при нажатой клавише Shift. Для запрета кэширования web-страниц на локальном диске формируйте в сценариях заголовок header(expires='-1d').
	8. По окончании лабораторной работы запишите документы сайта на сменный носитель для сохранения и распечатки.
2. Порядок выполнения работы
	1. Модифицировать файл отображения формы лабораторной работы №2 для проверки корректности заполнения полей средствами JavaScript с выдачей окна с замечаниями.
	2. Модифицировать файл отображения формы лабораторной работы №2 для проверки корректности заполнения полей средствами библиотеки JQuery.Validate.
	3. Отладить сценарии, убедиться в их корректной работе, отображении замечаний, продемонстрировать их работоспособность преподавателю.
	4. Сохранить на локальной машине и распечатать файлы отлаженных сценариев.
3. Варианты заданий

Задания различаются списком полей формы, реализованной в соответствии с заданиями к лабораторной работе №2, методами проверки их содержимого в соответствии с содержимым, и являются модификацией лабораторной работы №2.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Листинги сценариев.
	4. Выводы по работе.
2. Контрольные вопросы
	1. Каковы особенности выполнения сценариев на языке JavaScript?
	2. Каковы основные объекты объектной модели документа HTML?
	3. Каков порядок получения значения тега HTML в объектной модели?
	4. Обязательно ли присваивать тегам имена? Что можно использовать вместо них?
	5. Какие методы работы с окном, формой, тегом Вы знаете?
	6. Как и из какого источника можно подключить библиотеки JQuery?
	7. По какому событию и с какими параметрами происходит вызов метода ajaxSubmit плагина JQuery.form.js?
	8. Какие Вы знаете правила проверки полей форм в библиотеке JQuery.validate.js?
	9. Как сформировать сообщение об ошибках при нарушении правил проверки полей форм в библиотеке JQuery.validate.js?
	10. Как указать место вывода ошибок в библиотеке JQuery.validate.js?

#### ЛАБОРАТОРНАЯ РАБОТА № 8

**Компоненты технологии AJAX. Методы обработки событий. Методы передачи данных.**

1. Цель работы

Изучить компоненты технологии AJAX, методы обработки событий, методы передачи данных, освоить применение технологии AJAX.

1. Краткие теоретические сведения

AJAX – это акроним, раскрывающийся как Asynchronous JavaScript And XML и означающий асинхронный JavaScript и XML. AJAX – это технология, включающая сценарии JavaScript в документе HTML, которые по мере необходимости в фоновом режиме выполняют запросы к серверу (серверным сценариям на каком либо языке web-программирования), и получают необходимые данные, обновляя отдельные части документа и тем самым исключая необходимость его повторной загрузки или загрузки другого документа.

Фоновый запрос выполняется объектом XMLHttpRequest (встроенным в браузер), который позволяет из JavaScript организовать асинхронный доступ к серверу, благодаря чему пользователь имеет возможность продолжать работу с документом в ожидании получения ответа. Сценарий на стороне сервера отправляет свой ответ по протоколу HTTP. Ответ должен иметь такой формат, который может быть разобран кодом сценария JavaScript на стороне клиента. Традиционно используется формат XML, но можно использовать и другой формат, даже простой текст. Наиболее популярная альтернатива XML – JavaScript Object Notation (JSON), представление объектов в JavaScript.

Непосредственное применение методов и свойств объекта XMLHttpRequest, равно как и его создание в зависимости от текущего браузера, заметно увеличивает объем кода и затрудняет его разработку. Библиотека jQuery существенно упрощает жизнь разработчику приложений AJAX. Часто для вызова jQuery-методов используется сокращенный вариант, функция $ (знак доллара), а к наиболее популярным методам относятся jQuery.get, jQuery.post, jQuery.getJSON. Рассмотрим пример с применением последнего метода. Вначале подключаем библиотеку jQuery:

<scripttype="text/javascript" src="js/jquery-1.2.6.js"></script>

затем ожидаем окончания загрузки страницы:

<script type="text/javascript">

jQuery(document).ready(function(){

и возникновения нужного события какого-либо тега в ней, например, смены выбранного элемента списка с идентификатором (id) #country:

jQuery('#country').change(function(){

Прочитав значение выбранного элемента

value=$("select[@id=country] option:selected").val();

очищаем элементы связанного (подчиненного) списка с именем (name) city:

$('select[@name=city] option').remove();

Если в первом списке выбор не сделан, то и во втором выводить нечего:

if(!value) {$('select[@name=city]').append('<option value='+0+'> Здесь появятся элементы массива</option>');}

Выполняем запрос к серверу методом GET с ожиданием ответа в массив data:

$.getJSON("select.php?country="+value, function(data){

Полученные данные разбираем и помещаем в список с именем (name) city:

for(i in data) {

$('select[@name=city]').append('<option value='+i+'>'+data[i]+'</option>');};

При запросе данных в формате XML незначительно отличается метод выполнения запроса:

$.get("select.php?country="+value, function(data){

и весьма существенно – разбор полученных данных и добавление их в список:

idarray=data.documentElement.getElementsByTagName('id');

cityarray=data.documentElement.getElementsByTagName('name');

var j=0;

for (var i=0;i<idarray.length;i++) {

 j=idarray.item(i).firstChild.data;

 name=cityarray.item(i).firstChild.data;

 elem[j]=name;

 $('select[@name=city]').append('<option value='+j+'>'+elem[j]+'</option>');

}

Сценарии на стороне сервера тоже имеют свои особенности. Они должны давать ответ только на запросы от объекта XMLHttpRequest:

if($\_SERVER["HTTP\_X\_REQUESTED\_WITH"] == "XMLHttpRequest")

в соответствующем формате (обязательно в кодировке UTF-8):

[header](http://www.php.net/header)('Content-Type: application/json; charset=utf-8');

или

header("Content-type: text/xml; charset=UTF-8");

echo "<?xml version='1.0' encoding='UTF-8' standalone='yes'?>";

При формирования ответа в формате JSON можно использовать функцию json\_encode:

echo json\_encode(array("1" => 'Алушта', '2' => 'Сочи'));

при версии PHP >= 5.2.0 или array\_to\_json в противном случае:

echo array\_to\_json(array("1" => 'Алушта', '2' => 'Сочи'));

Сформировать ответ в формате XML можно руками, если структура ответа проста:

if($\_GET['country']==1) {

echo "<response><cities>";

echo "<city><id>1</id><name>Алушта</name></city>";

echo "<city><id>2</id><name>Сочи</name></city>";

echo "</cities></response>";}

Или, если версия PHP >= 5.1.0, с помощью объекта DOMDocument:

$dom=new DOMDocument();

$response=$dom->createElement('response');

$dom->appendChild($response);

$cities=$dom->createElement('cities');

$response->appendChild($cities);

if($\_GET['country']==1) {

$id=$dom->createElement('id');

$idvalue=$dom->createTextNode('1');

$id->appendChild($idvalue);

$name=$dom->createElement('name');

$nametext=$dom->createTextNode('Алушта');

$name->appendChild($nametext);

$city=$dom->createElement('city');

$city->appendChild($id);

$city->appendChild($name);

$cities->appendChild($city);

}

$xmlString=$dom->saveXML();

echo $xmlString;

Применяя AJAX необходимо помнить, что на стороне клиента JavaScript может быть отключен, что сделает приложения AJAX нефункциональными.

Пример применения технологии AJAX для фоновой отправки данных и обновления страниц сайта без перезагрузки см. здесь: http://gun.cs.nstu.ru/web/labs/lab8.

1. Методические указания
	1. Для разработки сценариев применяйте любые доступные текстовые редакторы, например, Jview или Notepad++.
	2. Сценарии располагаются на сервере с адресом 217.71.139.72, подключение к которому производится по протоколу FTP с именем учетной записи userNM, где N – порядковый номер группы, M- номер студента по журналу, пароль сообщается преподавателем.
	3. Лабораторная работа выполняется в отдельной папке на сервере, куда копируются сценарии лабораторной работы №3 для модификации.
	4. При настройке клиента FTP необходимо установить пассивный режим.
	5. При разработке страниц периодически сохраняйте их и проверяйте их отображение в любом из доступных браузеров.
	6. При написании сценариев обращайте внимание на кодировку русского текста, правильность написания HTML-тегов, имен переменных и функций PHP, парность открывающих и закрывающих кавычек в операторе echo.
	7. По окончании лабораторной работы запишите документы сайта на сменный носитель для сохранения и распечатки.
2. Порядок выполнения работы
	1. Модифицировать базу данных из лабораторной работы №1, добавив таблицу-справочник, уточняющий существующий справочник, например, для факультета – список групп, для марок автомобилей – модель и т.д.
	2. Модифицировать сценарии отображения записей из лабораторной работы №3 для отображения записей из дополнительного справочника.
	3. Модифицировать сценарии добавления и изменения записей из лабораторной работы №3 для заполнения списка из дополнительного справочника в зависимости от выбранного элемента списка основного справочника.
	4. Отладить сценарии, убедиться в их корректной работе, отображении замечаний, продемонстрировать их работоспособность преподавателю.
	5. Сохранить на локальной машине и распечатать файлы отлаженных сценариев.
3. Варианты заданий

Задания различаются базами данных, реализованными в соответствии с заданиями к лабораторной работе №1, форматами передачи данных от сервера (нечетные варианты – XML, четные JSON), и являются модификацией лабораторной работы №3.

1. Содержание отчета
	1. Цель работы.
	2. Вариант задания.
	3. Листинги сценариев.
	4. Выводы по работе.
2. Контрольные вопросы
	1. Что входит в состав технологии AJAX?
	2. Каковы наиболее популярные методы отправки данных в библиотекеJQuery?
	3. Какие Вы знаете события, пригодные для вызоваметодов библиотекиJQuery?
	4. Как можно указать тег, чьи события отслеживаются?
	5. Как можно указать формат получаемых из запроса данных?
	6. Каковы особенности сценариев, отвечающих на фоновый запрос?
	7. Как в серверном сценарии указать формат ответа?
	8. Каковы способы формирования ответа в формате JSON?
	9. Каковы способы формирования ответа в формате XML?
	10. Чем отличается разбор данных, пришедших в формате JSON и XML?

#### КУРСОВАЯ РАБОТА

Типовые задания на курсовую работу:

1. Разработка административной панели для управления (добавления, редактирования пароля, прав доступа, удаления) пользователя, авторизуемого на сайте. Пример см. здесь: <http://gun.cs.nstu.ru/web/labs/lab4/>. Для доступа используются учетные записи администратора tratata с паролем secretpassword и модератора user с паролем test. Ссылка на модуль администрирования и представляет собой пример курсовой работы.
2. Модификация лабораторной работы №5 для создания объемных (круговых) диаграмм или графиков.
3. Разработка галереи изображений и административной панели для управления (добавления, редактирования размера, подписи, удаления) изображения, размещаемого на сайте.
4. Разработка модуля учета посещений сайта и графического отображения статистики посетителей (график) и посещенных страниц сайта (диаграмма).
5. Разработка интерфейса к базе данных средствами какого-либо фреймворка.
6. Разработка интерфейса к базе данных средствами технологии AJAX (без перезагрузки страниц).
7. Разработка динамического сайта произвольной тематики соответствующей сложности (по согласованию с преподавателем).

Сценарии, реализованные в курсовой работе, обязательно должны быть размещены в сети интернет, либо на учебном сервере в отдельной папке в каталоге студента, либо на любом доступном студенту сервере.

В пояснительной записке должна быть представлена графическая структура базы данных, логическая и физическая структуры сайта, содержащего курсовую работу, а также дамп базы данных и исходный код сценариев.

Для подтверждения авторства курсовой работы она должна быть защищена путем собеседования с преподавателем либо путем модификации кода сценариев по требованию преподавателя.

#### Литература

1. Яргер Р., Риз Дж., Кинг Т. MySQL и mSQL. Базы данных для небольших предприятий и Интернета. - СПб: Символ-Плюс, 2000 - 560 с.
2. Ратшиллер Т., Геркен Т. PHP4: разработка Web-приложений. - СПб: Питер, 2001. - 384 с.
3. Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySQL. - К.: "ДиаСофт", 2001. - 672 с.
4. Колисниченко Д. Н. PHP 5/6 и MySQL 6. Разработка Web-приложений. - СПб., 2009. - 607 с.
5. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. – СПб: БХВ-Петербург, 2010. - 900 с.
6. Дэнни Гудман, Майкл Моррисон. JavaScript. Библия пользователя. - СПб: Вильямc, 2006. - 1184 с.
7. [Кристиан Дари](http://www.ozon.ru/person/2377756/), [Богдан Бринзаре](http://www.ozon.ru/person/2826438/), [Филип Черчез-Тоза](http://www.ozon.ru/person/2826440/), [Михай Бусика](http://www.ozon.ru/person/2826441/). AJAX и PHP. Разработка динамических веб-приложений. -СПб: [Символ-Плюс](http://www.ozon.ru/brand/856498/), 2009. - 336 с.
8. Томас А. Пауэлл. Ajax. Настольная книга программиста. –М: [Эксмо](http://www.ozon.ru/brand/857671/), 2009. - 720 стр.

**WEB - ПРОГРАММИРОВАНИЕ**

**Методические указания к лабораторным работам**

**Редактор**

**Технический редактор**

**Лицензия № 021040 от 22.02.96. Подписано в печать \_\_\_.\_\_\_.\_\_\_.
Формат 60 х 84 1/16. Бумага оберточная. Тираж 50 экз.
Уч.-изд.л. 1,0. Печ.л. 1. Изд. № \_\_\_\_\_\_. Заказ № \_\_\_\_\_\_ Цена договорная**

 **Отпечатано в типографии**

 **Новосибирского государственного технического университета**

 **630092, г. Новосибирск, пр. К. Маркса, 20**