МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам по дисциплине «Управление данными» для студентов направления 09.03.02 «Информационные системы и технологии»

Пятигорск, 2022

Методические указания к лабораторным работам по дисциплине «Управление данными» для студентов направления 09.03.02 «Информационные системы и технологии» подготовлены в соответствии рабочей программой с учетом квалификационных требований ФГОС ВО.

СОДЕРЖАНИЕ

Лабораторная работа №1. Составление технического задания на разработку сайта

Цель работы: научиться разрабатывать техническое задание на разработку сайта.

Теоретическое обоснование

Теоретическое обоснование

На техническое задание существует стандарт ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению». В соответствии с этим стандартом техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки.

При необходимости допускается в техническое задание включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе «Введение» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

2.2. В разделе «Основания для разработки» должны быть указаны: документ (документы), на основании которых ведется разработка; организация, утвердившая этот документ, и дата его утверждения; наименование и (или) условное обозначение темы разработки.

2.3. В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.4. Раздел «Требования к программе или программному изделию» должен содержать следующие подразделы:

требования к функциональным характеристикам; требования к надежности; условия эксплуатации; требования к составу и параметрам технических средств; требования к информационной и программной совместимости; требования к маркировке и упаковке; требования к транспортированию и хранению; специальные требования.

2.4.1. В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

2.4.2. В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечения устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т.п.).

2.4.3. В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т.п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

2.4.4. В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их основных технических характеристик.

2.4.5. В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования и программным средствам, используемым программой. При необходимости должна обеспечиваться защита информации и программ.

2.4.6. В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

2.4.7. В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

2.5а. В разделе «Требования к программной документации» должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

2.5. В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

2.6. В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.

2.7. В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

2.8. В приложениях к техническому заданию, при необходимости, приводят: перечень этапов пользовательских и других работ, обосновывающих разработку; схемы алгоритмов, таблицы, описания, обоснования, расчеты и

другие документы, которые могут быть использованы при разработке; другие источники разработки.

Аппаратура и материалы. Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 128 Мбайт; свободное дисковое пространство – не менее 800 Мбайт; устройство для чтения компакт-дисков; монитор типа SuperVGA (число цветов – 256) с диагональю не менее 17 . **Программное обеспечение** – операционная система WINDOWS 98 / NT / ME / 2000 / XP/ Vista/ 7, MicrosoftWord.

Указания по технике безопасности. Правила техники безопасности при выполнении лабораторной работы совпадают с общепринятыми для пользователей персональных компьютеров. Студентам запрещается самостоятельно производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера необходимо сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Необходимо соблюдать правила техники безопасности при работе с электрооборудованием, не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

Методика и порядок выполнения работы

Составить ТЗ на разработку сайта в соответствии с ГОСТ 19.104-78.

Для оформления отчета можно использовать любой текстовой редактор. Тему согласовать с преподавателем.

Задание. Составить техническое задание на разработку сайта для информационной системы по варианту, согласованному с преподавателем.

Оформить титульный лист по образцу (на странице 9)

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из: 1) титульного листа (образец приведен в лекции 9);

2) технического задания.

Вопросы для защиты работы

1. Назовите, какой раздел технического задания можно считать основным и почему.
2. Какую информацию должны содержать остальные разделы?
3. В чем основная сложность разработки технического задания?

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Зав кафедрой СУиИТ,
_____.
«___» _____201_ г.

Техническое задание на разработку базы данных

Руководитель _____
Исполнитель _____

ФИО, группа

201

Лабораторная работа №2. Архитектура распределенных Internet приложений. Установка пакета Denwer.

Цель работы: научиться устанавливать пакет для web-разработчика Denwer.

Теоретическое обоснование

Установка базового пакета виртуального сервера Denver. Базовый пакет содержит большинство необходимых программ и утилит:

Инсталлятор (поддерживается также инсталляция на flash-накопитель). Apache, SSL, SSI, mod_rewrite, mod_php.

PHP5 с поддержкой GD, MySQL, sqLite.

MySQL5 с поддержкой транзакций.

Система управления виртуальными хостами, основанная на шаблонах. Чтобы создать новый хост, нужно лишь добавить директорию в каталог

/home, править конфигурационные файлы не требуется. По умолчанию уже поддерживаются схемы именования директорий многих популярных хостеров; новые можно без труда добавить.

Система управления запуском и завершением всех компонентов Денвера.

phpMyAdmin — система управления MySQL через Web-интерфейс.

Эмулятор sendmail и SMTP-сервера (отладочная «заглушка» на localhost:25, складывающая приходящие письма в /tmp в формате.eml); поддерживается работа совместно с PHP, Perl, Parser и т.д.

Подготовка к работе с сетью

Многие ассоциируют слово «сеть» с Интернетом, локальной сетью или хотя бы модемом. И совершенно напрасно. Фраза «настроим сеть» может иметь смысл даже в том случае, когда ни одного из перечисленных устройств у компьютера нет! Здесь имеется ввиду лишь установка драйверов и сетевых протоколов, которые позволят Apache запуститься и работать на локальной машине.

Итак, самый простой тест: откройте Пуск — Выполнить и введите там команду (рисунок 2.1):

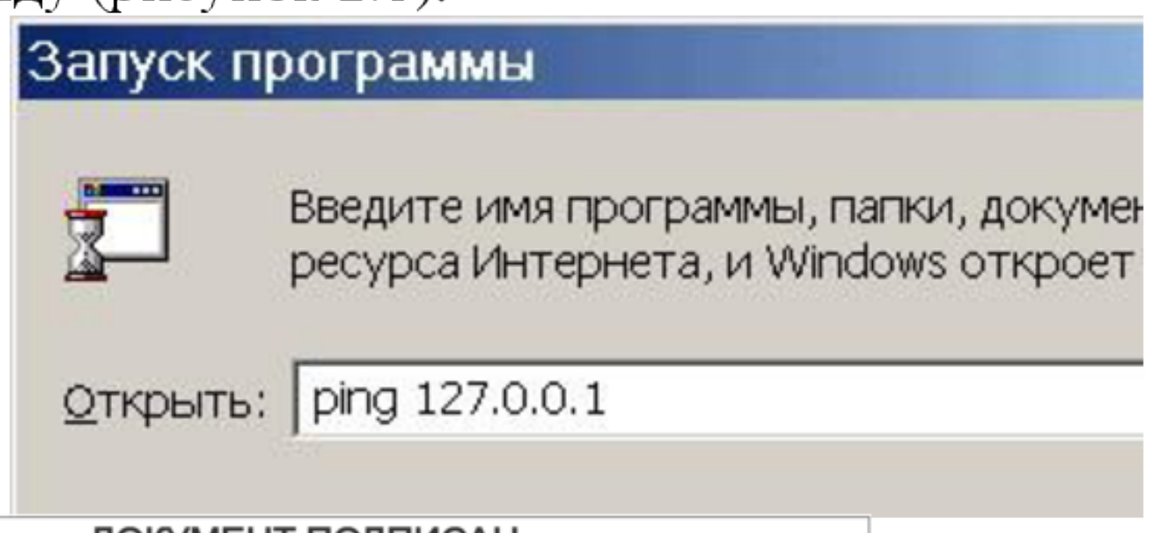

Рисунок 2.1 – Пинг локального сервера

Вы должны увидеть примерно следующую картину (рисунок 2.2):

```
Обмен пакетами с 127.0.0.1 по 32 байт:

Ответ от 127.0.0.1: число байт=32 время<10мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<10мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<10мс TTL=128
Ответ от 127.0.0.1: число байт=32 время<10мс TTL=128
```

Рисунок 2.2 – Результаты команды «пинг» локального сервера

Процесс продолжается несколько секунд. Если вы это видите, то все в порядке, и вы можете приступать к инсталляции дистрибутива. Если же, например, окно лишь «мигнет» (откроется и тут же закроется), либо же будут выведены какие-нибудь сообщения об ошибках, значит, сетевые протоколы не установлены.

Установка дистрибутива

Запустите инсталлятор Денвера. Вы увидите перед собой следующее (рисунок 2.3):

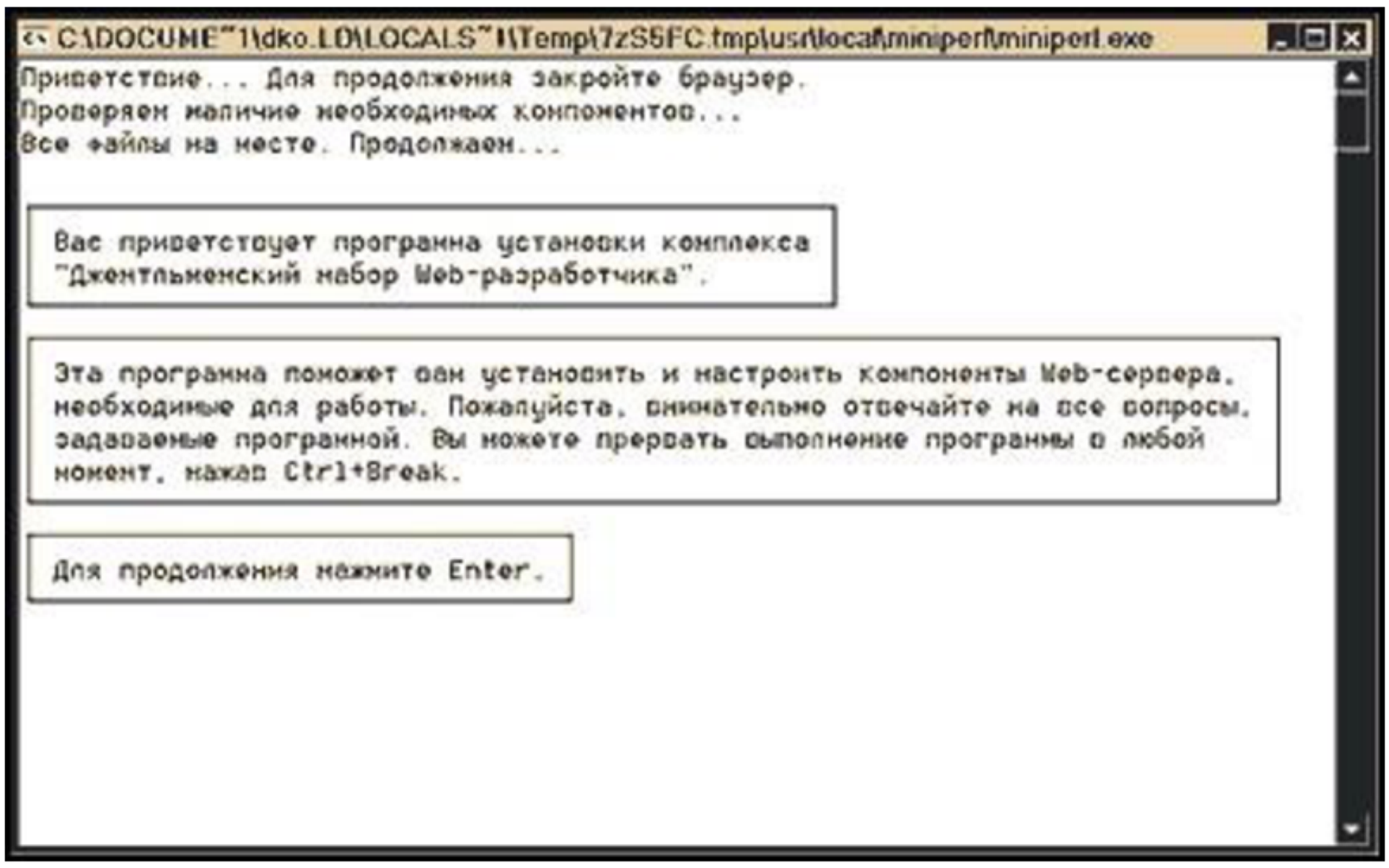

Рисунок 2.3 – Установка Денвера

Вас спросят о том, в какой каталог вы хотели бы установить комплекс (по умолчанию используется C:\WebServers, вам нужно лишь нажать Enter, чтобы согласиться с этим выбором). В указанном каталоге будут расположены абсолютно все компоненты системы, и вне его никакие файлы в дальнейшем не создаются (исключая ярлыки на Рабочем столе).

Настоятельно рекомендуется устанавливать комплекс в каталог первого уровня — то есть, C:\WebServers, а не, например, C:\My\WebServers.

Дело в том, что инсталляторы пакетов расширений ищут базовый комплект именно на первом уровне по всем дискам. И, если не находят, заставляют ввести имя директории вручную.

Далее вам предложат ввести имя виртуального диска, который будет связан с вышеуказанной директорией. Рекомендуется согласиться со значением по умолчанию (Z:). Важно, что диска с этим именем еще не должно содержаться в системе — чаще всего так и происходит с диском Z:.

После этого начнется копирование файлов дистрибутива, а под конец вам будет задан вопрос, как именно вы собираетесь запускать и останавливать комплекс. У вас есть две альтернативы:

Создавать виртуальный диск при загрузке машины (естественно, инсталлятор позаботится, чтобы это происходило автоматически), а при остановке серверов его (диск) не отключать.

Создавать виртуальный диск только по явной команде старта комплекса (при щелчке по ярлыку запуска на Рабочем столе). И, соответственно, отключать диск от системы — при остановке серверов. Вы должны воспользоваться именно этим способом.

Первый запуск Денвера

После установки щелкайте по созданному инсталлятором ярлыку Start Denwer на Рабочем столе, а затем, дождавшись, когда все консольные окна

исчезнут, открывайте браузер и набирайте в нем адрес: http://localhost/denwer/. Выходить из Интернета при этом не обязательно

(рисунок 2.4).

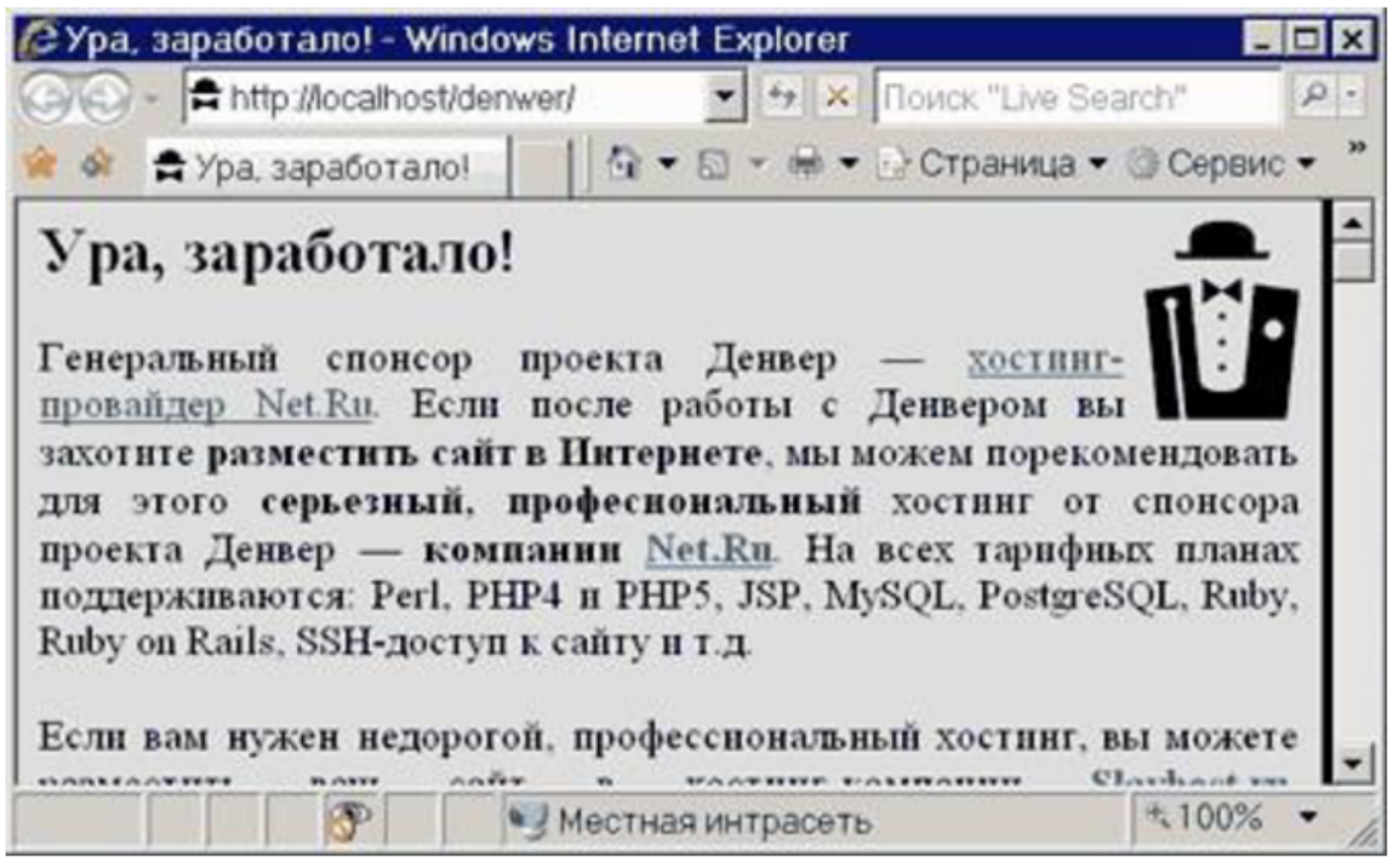

Рисунок 2.4 – Стартовая страница Денвера

Если тестовая страница все же не загрузится, проверьте:

Отключен ли у вас прокси-сервер в настройках браузера?

Запущен ли Денвер? Если да, нет ли ошибок при щелчке на пиктограмме пера (справа внизу)?

Не запущен ли у вас какой-то другой Web-сервер, который мешает Денверу (часто бывает в Windows XP)? Например, Microsoft IIS? Если да, отключите его.

Денвер прошел тестирование в следующих ОС:

Windows 95/98/ME; Windows NT/2000/XP/2003; Windows Vista.

Windows 7

Работа с виртуальными хостами

Внимание пользователей Windows NT, 2000 или XP (и старше). Прежде, чем продолжить, убедитесь, что у вас запущена служба «DNSклиент». Это

можно сделать, открыв Панель управления — Администрирование — Службы. В противном случае, виртуальные хосты работать не будут.

Добавить новый виртуальный хост в Денвере чрезвычайно просто.

Пусть это будет test1.ru. Вам нужно проделать следующее (рисунок 2.5):

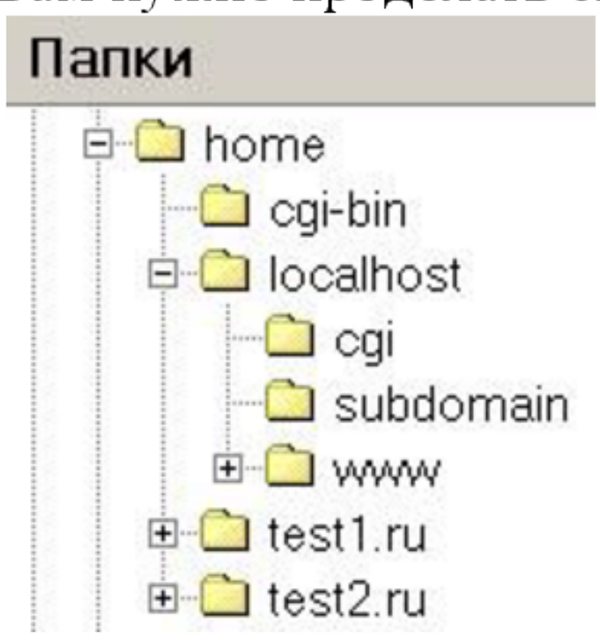

Рисунок 2.5 – Структура папок Девера

- Создать в папке /home директорию с именем, совпадающим с именем виртуального хоста (в нашем случае test1.ru). Имя директории содержит точку. Эта директория будет хранить директории документов доменов третьего уровня для test1.ru. Например, имя abc.test1.ru связывается сервером с директорией /home/test1.ru/abc/, а имя abc.def.test1.ru — с /home/test1.ru/abc.def/. Ну и, конечно, поддиректория www соответствует адресам www.test1.ru и просто test1.ru. На рисунке показано, как может выглядеть директория /home. Не забудьте создать папку www в директории виртуального хоста, ведь именно в ней будут храниться его страницы и скрипты.
- Перезапустить сервер, воспользовавшись, например, ярлыком Restart Denwer на Рабочем столе.

Как только вы начнете создавать виртуальные хосты, Контроллер удаленного доступа на некоторых системах может предлагать вам альтернативу (рисунок 2.6 или 2.7):

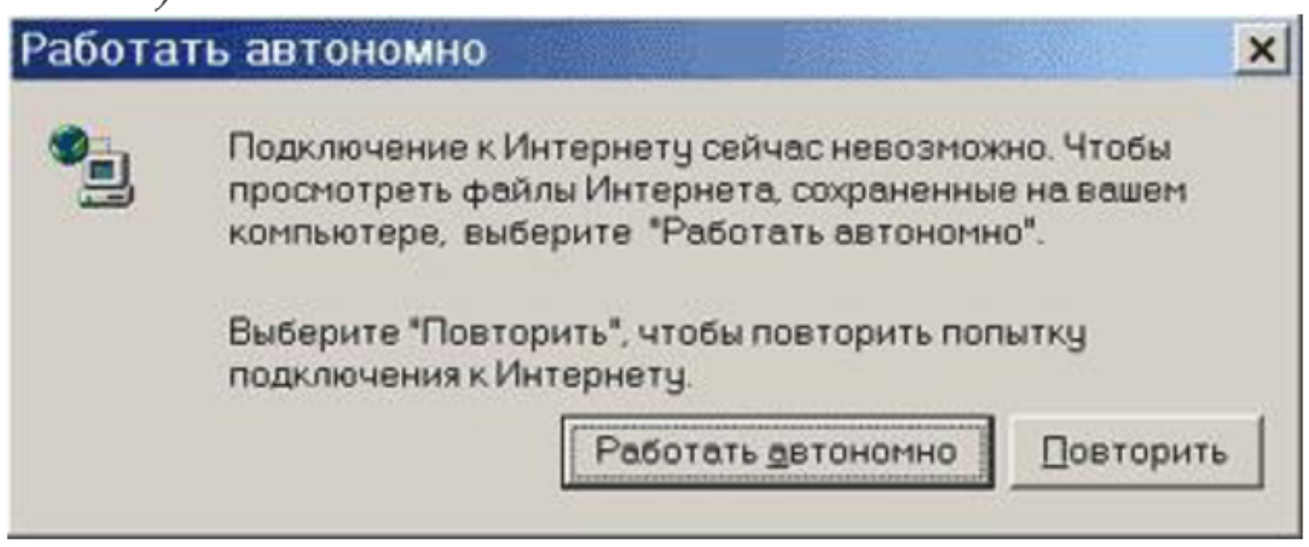

Рисунок 2.6 – Пинг локального сервера

или так:

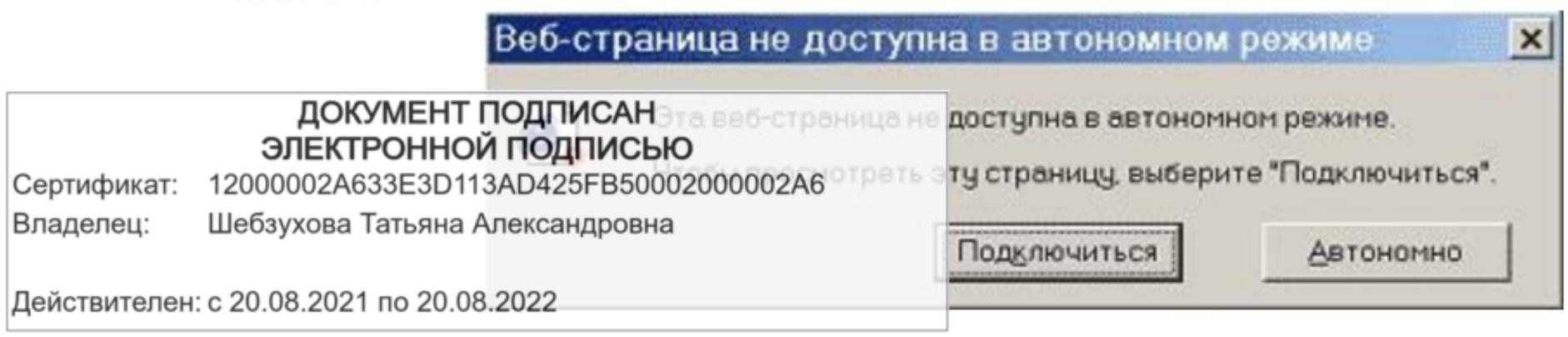

Рисунок 2.7 – Пинг локального сервера

Всегда выбирайтеПодключиться или Повторить.

Ни в коем случае не давайте ему ответ Автономно.

Но если ваш Контроллер удаленного доступа не реагирует и на ответПодключиться начинает набирать номер на модеме, откройте Сервис — Свойства обозревателя — Подключение и в разделе Настройка удаленного доступа поставьте флажок Не использовать (или Never Dial a connection).

Это рекомендации для пользователей Windows 2000. На всех остальных системах пункты меню и кнопки могут называться немного подругому, но смысл остается тот же.

Многие версии Windows поставляются со включенным по умолчанию прокси-сервером. Это может вызвать кое-какие проблемы при работе с Денвером (впрочем, легко разрешимые).

- Если после запуска Денвера страница http://localhost не работает, вероятнее всего, вам нужно отключить прокси-сервер в настройках браузера. Для "простых" хостов (вроде localhost, test, dklab и т.д.) обычно достаточно флажка «Не использовать прокси-сервер для локальных адресов» на вкладке

Свойства обозревателя — Подключение — Настройка сети — Дополнительно.

- Если localhost работает, а test1.ru (и вообще хосты, имя которых состоит из нескольких частей) — нет, то, вероятно, ваш браузер не может распознать последний хост как локальный. Вам необходимо либо полностью отключить прокси-сервер, либо же перечислить хосты в списке Подключение — Настройка сети — Дополнительно — Исключения.

Методика и порядок выполнения работы

Изучить процесс установки, описанный в разделе «Теоретическое обоснование». Установить пакет Денвер.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) описание установки пакета Денвер.

Вопросы для защиты работы:

1. Какие утилиты входят в базовый пакет Денвер?

Лабораторная работа №3.Основная информация о MySQL. Создание базы данных

Цель работы: научиться создавать и администрировать базу данных MySQL.

Теоретическое обоснование

MySQL - это популярная система управления базами данных (СУБД), очень часто применяемая в сочетании с PHP.

База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми - от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети. Для записи, выборки и обработки данных, хранящихся в компьютерной базе данных, необходима система управления базой данных, каковой и является ПО MySQL. Поскольку компьютеры замечательно справляются с обработкой больших объемов данных, управление базами данных играет центральную роль в вычислениях. Реализовано такое управление может быть по-разному - как в виде отдельных утилит, так и в виде кода, входящего в состав других приложений.

MySQL - это система управления реляционными базами данных. В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

MySQL - это ПО с открытым кодом. Применять его и модифицировать может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями.

Использование программного обеспечения MySQL регламентируется лицензией GPL (GNU General Public License), http://www.gnu.org/licenses/, в которой указано, что можно и чего нельзя делать с этим программным обеспечением в различных ситуациях.

Почему веб-программисты отдают предпочтение СУБД MySQL? MySQL является очень быстрым, надежным и легким в использовании. Если вам требуются именно эти качества, попробуйте поработать с данным сервером. MySQL обладает также рядом удобных возможностей,

разработанных в тесном контакте с пользователями. Первоначально сервер MySQL разрабатывался для управления большими базами данных с целью обеспечить более высокую скорость работы по сравнению с существующими на тот момент аналогами. И вот уже в течение нескольких лет данный сервер успешно используется в условиях промышленной эксплуатации

с высокими требованиями. Несмотря на то что MySQL постоянно совершенствуется, он уже сегодня обеспечивает широкий спектр полезных функций. Благодаря своей доступности, скорости и безопасности MySQL очень хорошо подходит для доступа к базам данных по Internet.

Технические возможности СУБД MySQL MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер,

обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API). Мы также поставляем сервер MySQL в виде многопоточной библиотеки, которую можно подключить к пользовательскому приложению и получить компактный, более быстрый и легкий в управлении продукт. Доступно также большое количество программного обеспечения для MySQL, в большей части - бесплатного.

MySQL состоит из двух частей: серверной и клиентской. Сервер MySQL постоянно работает на компьютере. Клиентские программы (например, скрипты PHP) посылают серверу MySQL SQL-запросы через механизм сокетов (то есть при помощи сетевых средств), сервер их обрабатывает и запоминает результат. То есть скрипт (клиент) указывает, какую информацио он хочет получить от сервера баз данных. Затем сервер баз данных посылает ответ (результат) клиенту (скрипту).

Почему всегда передается не весь результат? Очень просто: дело в том, что размер результирующего набора данных может быть слишком большим, и на его передачу по сети уйдет чересчур много времени. Да и редко когда бывает нужно получать сразу весь вывод запроса (то есть все записи, удовлетворяющие выражению запроса). Например, нам может потребоваться лишь подсчитать, сколько записей удовлетворяет тому или иному условию, или же выбрать из данных только первые 10 записей. Механизм использования сокетов подразумевает технологию клиент-сервер, а это означает, что в системе должна быть запущена специальная программа — MySQL-сервер, которая принимает и обрабатывает запросы от программ. Так как вся работа происходит в действительности на одной машине, накладные расходы по работе с сетевыми средствами незначительны (установка и поддержание соединения с MySQL-сервером обходится довольно дешево).

Структура MySQL трехуровневая: базы данных — таблицы — записи. Базы данных и таблицы MySQL физически представляются файлами с расширениями frm, MYD, MYI. Логически - таблица представляет собой совокупность записей. А записи - это совокупность полей разного типа. Имя базы данных MySQL уникально в пределах системы, а таблицы - в пределах базы данных, поля - в пределах таблицы. Один сервер MySQL может поддерживать сразу несколько баз данных, доступ к которым может разграничиваться логинами и паролем. Зная эти логин и пароль, можно работать с конкретной базой данных. Например, можно создать или удалить в ней таблицу, добавить записи и т. д. Обычно имя-идентификатор и пароль

назначаются хостинг провайдерами, которые и обеспечивают поддержку MySQL для своих пользователей.

Методика и порядок выполнения работы

1. Запустить Denver
2. В браузере перейти по ссылке http://localhost/Tools/phpMyAdmin/

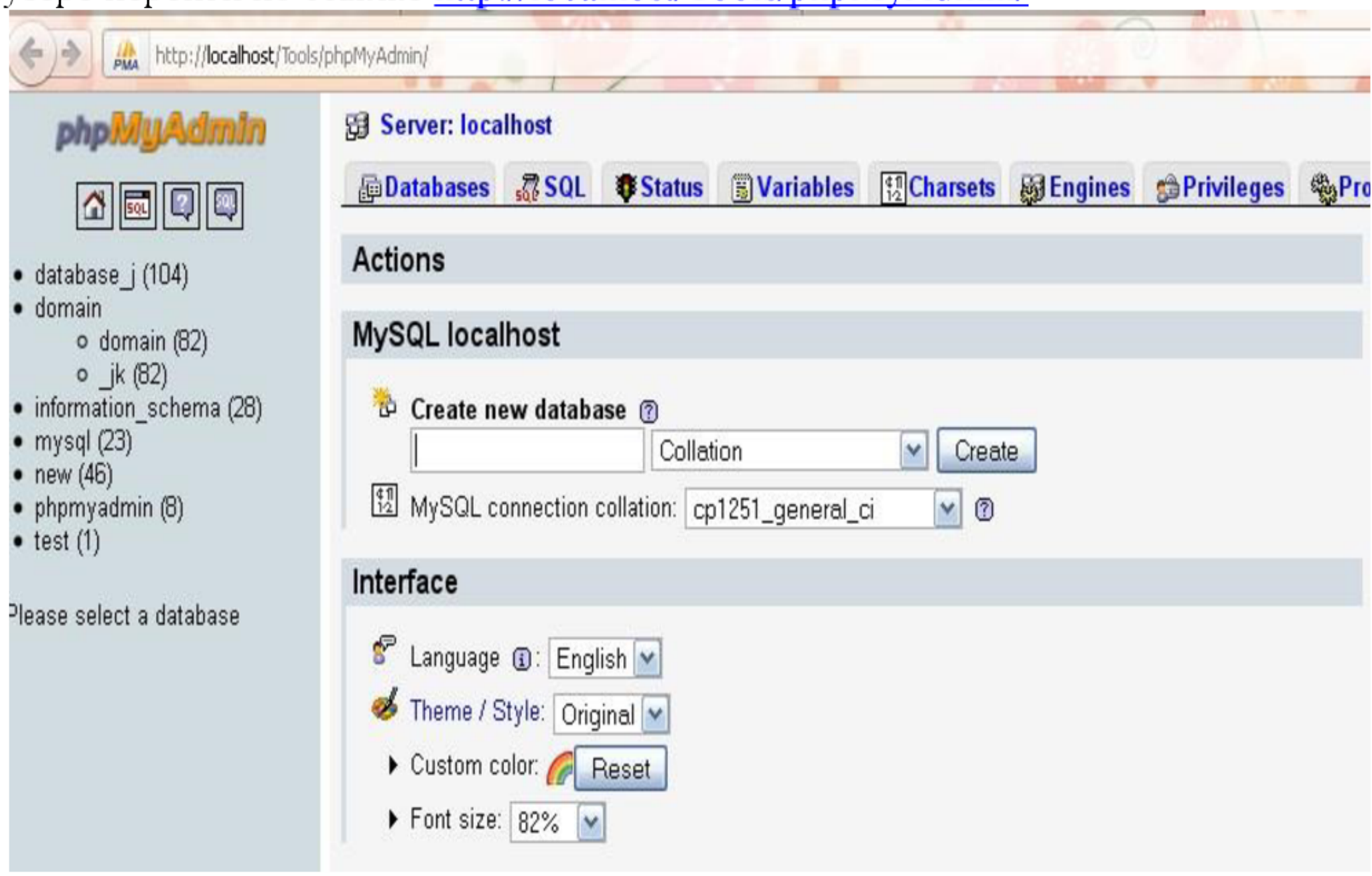

Рисунок 3.1- Окно браузера

В поле Createnewdatabase (рисунок 3.1) ввести имя создаваемой базы данных, в списке MySqlconnection collation выбираем utf8_general_ci, нажимает кнопку create (рисунок 3.2).

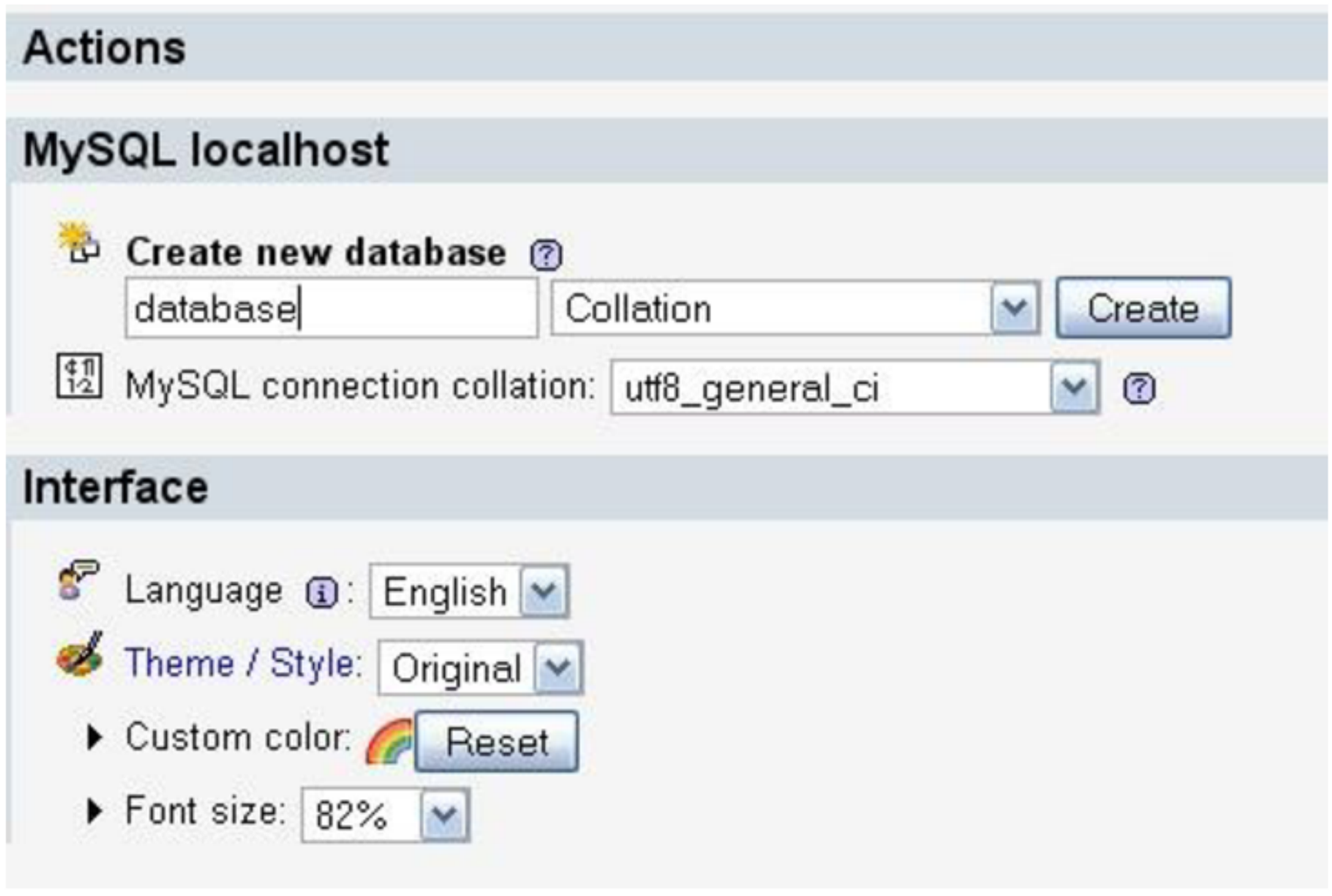

Рисунок 3.2 – Создание БД

Задание

1. Создать нового пользователя базы данных, обладающего всеми привилегиями и установить для него пароль. Воспользуйтесь вкладкой privileges.
2. Создайте пользователя, обладающего ограниченными правами, и задайте ему пароль.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из: 1) названия лабораторной работы; 2) описание создания БД.

Вопросы для защиты работы:

1. Что такое привилегии?
2. Какие привилегии Вы выбрали при создании БД?

Лабораторная работа №4. Поля и их типы в MySQL

Цель работы: научиться использовать различные типы данных базы данных MySQL.

Теоретическое обоснование

База данных с точки зрения MySQL (и некоторых других СУБД) - это обыкновенный каталог, содержащий двоичные файлы определенного формата - таблицы. Таблицы состоят из записей, а записи, в свою очередь, состоят из полей. Поле имеет два атрибута - имя и тип.

Тип поля может быть:

- Целым;
- Вещественным;
- Строковым;

- Бинарным;
- Дата и время;
- Перечисления и множества.

Возможные типы данных, диапазоны и описания представлены в таблицах 4.1- 4.5.

Таблица 4.1- Целочисленные типы данных

Тип	Диапазон
TINYINT	-128...+127
SMALLINT	-32768...+32767
MEDIUMINT	-8 388 608...+8 388 607
INT	-2 147 483 648...+2 147 483 647
BIGINT	-9 223 372 036 854 775 808...+9 223 372 036 854 775 807

Вещественные типы записываются в виде:

ТИП (ДЛИНА, ЗНАКИ) [UNSIGNED]

Длина - это колличество знакомест, в которых будет размещено все число при его передаче, а ЗНАКИ - это колличество знаков после десятичной точки, которые будут учитываться. Если указан модификатор UNSIGNED, знак числа учитываться не будет.

Таблица 4.2- Вещественные числа

Тип	Описание
FLOAT	Небольшая точность
DOUBLE	Двойная точность
REAL	То же, что и DOUBLE
DECIMAL	Дробное число, хранящееся в виде строки
NUMERIC	То же, что и DECIMAL

Любая строка - это массив символов. При поиске с помощью оператора SELECT (мы рассмотрим его далее) не учитывается регистр символов: строки "HELLO" и "Hello" считаются одинаковыми.

Можно настроить MySQL на автоматическое перекодирование символов - в этом случае в базе данных строки будут храниться в одной кодировке, а выводиться - в другой.

В большинстве случаев применяется тип VARCHAR или просто CHAR, позволяющий хранить строки, содержащие до 255 символов. В скобках после типа указывается длина строки:

VARCHAR(48); CHAR(73);

Если 255 символов для вашей задачи недостаточно, можно использовать другие типы данных, например TEXT.

Таблица 4.3 - Строки

Тип	Описание

TINYTEXT	Максимальная длина 255 символов
TEXT	Максимальная длина 65535 символов (64 Кб)
MEDIUMTEXT	Максимальная длина 16 777 215 символов
LONGTEXT	Максимальная длина 4 294 967 295 символов

Бинарные типы данных также можно использовать для хранения текста, но при поиске будет учитываться регистр символов. К тому же, любой текстовый тип можно преобразовать в бинарный, указав модификатор BINARY:

VARCHAR(30) BINARY;

Таблица 4.4- Бинарные типы данных

Тип	Описание
TINYBLOB	Максимум 255 символов
BLOB	Максимум 65535 символов
MEDIUMBLOB	Максимум 16 777 215 символов
LONGBLOB	Максимум 4 294 967 295

Примечание: Бинарные данные не перекодируются "на лету", если установлена перекодировка символов. Таблица 4.5- Дата и время

Тип	Описание
DATE	Дата в формате ГГГ-ММ-ДД
TIME	Время в формате ЧЧ:ММ:СС
TIMESTAMP	Дата и время в формате timestamp, выводится в виде ГГГГММДДЧЧММСС
DATETIME	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС

Другие типы данных MySQL рассматривать бессмысленно, поскольку применение их в PHP нецелесообразно.

Задание

Изучите поля и типы данных MySQL.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;

2) описание типов данных MySQL.

Вопросы для защиты работы:

1. Какие типы данных поддерживает MySQL?

2.Каким образом можно представить дату?

Лабораторная работа №5. Операторы и команды MySQL

Цель работы: научиться использовать операторы и команды базы данных MySQL.

Теоретическое обоснование

Структурированный язык запросов SQL позволяет производить различные операции с базами данных: создавать таблицы, помещать, обновлять и удалять из них данные, производить запросы из таблиц и.т.д.

Далее мы последовательно рассмотрим все эти операторы.

Несмотря на то, что последний стандарт SQL принят в 1992 году, на сегодняшний день нет ни одной СУБД, где бы он полностью выполнялся. Более того, в различных базах данных часть операций осуществляется поразному. Мы будем придерживаться диалекта SQL характерного для СУБД MySQL поэтому не все запросы могут выполняться для других баз данных.

Примечание: Команды SQL не чувствительны к регистру, но традиционно они набираются прописными буквами.

Создание таблиц. Оператор CREATE

Создать таблицу через SQL-запрос позволяет оператор CREATE. Его синтаксис:

```
CREATE TABLE Имя_таблицы
(
Имя_поля1 Тип Модификатор
...
Имя_поляN Тип Модификатор
[первичный ключ]
[внешний ключ]
)
```

Вообще, с помощью оператора CREATE можно создавать и другие объекты, но мы их рассматривать не будем, поскольку их применение весьма ограничено.

В качестве модификаторов можно использовать следующие значения:

- NOT NULL - поле не может содержать неопределенного значения (NULL), то есть поле должно быть явно инициализировано;
- PRIMARY KEY - поле будет первичным ключом (идентификатором записи), по которому можно однозначно идентифицировать запись;
- AUTO_INCREMENT - при вставке новой записи значение этого поля будет автоматически увеличено на единицу, поэтому в таблице не будет двух записей с одинаковым значением этого поля;
- DEFAULT - задает значение, которое будет использовано по

умолчанию, если при вставке записи поле не будет инициализировано явно.

Значение по умолчанию задается следующим образом (таблица 5.1):

Таблица 5.1- Задание значений по умолчанию

Имя_поля	Тип	DEFAULT	Значение, например
NO	INT	DEFAULT	0
NAME	INT	DEFAULT	'Петров'

Теперь создадим таблицы - "Товар", "Клиенты", "Заказы":

```
CREATE TABLE CLIENTS
(
C_NO    int      NOT NULL,
FIO     char(40)  NOT NULL,
ADDR    char(30)  NOT NULL,
CITY    char(15)  NOT NULL,
PHONE   char(11)  NOT NULL
);
```

Таблица CLIENTS содержит поля C_NO (номер клиента), FIO (Фамилия, Имя, Отчество), ADDR (Адрес), CITY (Город) и PHONE

(Телефон). Все эти поля не могут содержать пустого значения (NOT NULL).

```
CREATE TABLE TOOLS
(
T_NO   int         NOT NULL,
DSEC   char(40)    NOT NULL,
PRICE  double(9,2)  NOT NULL,
QTY    double(9,2)  NOTNULL
);
```

Данная таблица будет содержать данные о товарах. Тип double(9,2) означает, что 9 знаков относим под целую чать, и два - под дробную. QTY - этоколличествотоваранаскладе.

```
CREATE TABLE ORDERS
(
O_NO     int        NOT NULL,
DATE     date       NOT NULL,
C_NO     int        NOT NULL,
T_NO     int        NOT NULL,
QUANTITY double(9,2) NOT NULL,
AMOUNT   double(9,2) NOT NULL
);
```


Эта таблица содержит сведения о заказах - номер заказа (O_NO), дату заказа (DATE), номер клиента (C_NO), номер товара (T_NO), колличество

(QUANTITY) и сумму всего заказа AMOUNT (то есть AMOUNT = T_NO * TOOL_PRICE).

Добавление данных в таблицу. Оператор INSERT

Для добавления записей используется оператор INSERT:

INSERT INTO Имя_таблицы [(Список полей)]

VALUES (Список констант);

После выполнения оператора INSERT будет создана новая запись, в качестве значений полей будут использованы соответствующие константы, указанные в списке VALUES.

Теперь добавим данные в наши таблице. Добавить данные можно с помощью оператора INSERT. Рассмотрим пример использование оператора INSERT:

```
INSERT INTO CLIENTS
VALUES (1, 'Иванов И.И.', 'Вокзальная 3', 'Москва', '09599911100');
```

Добавляемые значения должны соответствовать тому порядку, в котором поля перечислены в операторе CREATE. Если вы хотите добавлять информацию в другом порядке, то вы должны указать этот порядок в операторе INSERT, например:

```
INSERT INTO CLIENTS (FIO, ADDR, C_NO, PHONE, CITY)
VALUES ('Петров', 'Мира 29', 2, '-', 'Екатеринбург');
```

С помощью INSERT мы можем добавлять данные и в определенные поля, например, C_NO и FIO:

```
INSERT INTO CLIENTS (C_NO, FIO)
VALUES (1, 'Иванов');
```

Однако, в нашем случае сервер MySQL не выполнит такой запрос, поскольку все остальные поля равны NULL (пустое значение), а наша таблица не принимает пустые значения. Аналогично можно добавить данные в другие таблицы.

В качестве примера, добавим данные в таблицу TOOLS: INSERT INTO TOOLS

```
VALUES (1, 'Клавиатура ABC', 340.98, 5);
```

Обратите внимание, что мы пока не указали первичные ключи таблицы, поэтому нам никто не мешает добавить в таблицу одинаковые записи. Добавить дату в поле DATE можно с помощью функции STR_TO_DATE:

```
INSERT INTO ORDERS VALUES (1,
STR_TO_DATE('01/03/10','%m/%d/%y'),1,1,1,340.98)
```

Данная запись означает, что третьего января 2010 года Иванов И.И. (C_NO=1) заказал одну (QUANTITY=1) клавиатуру ABC (T_NO=1).

Обновление записей. Оператор UPDATE

Синтаксис оператора UPDATE, который используется для обновления записей, выглядит так:

```
UPDATE Имя_таблицы
SET Поле1 = Значение1,..., ПолеN = ЗначениеN
[WHERE Условие];
```

Если не задано условие WHERE, будет модифицирована вся таблица, а это может повлечь за собой непредсказуемые последствия, поскольку для всех записей будут установлены одинаковые значения полей, поэтому всегда указывайте условие WHERE.

Предположим, нам необходимо обновить запись, если, например, клиент Иванов переехал в другой город и нам нужно отметить это событие в базе данных. Сделаемследующее:

```
UPDATECLIENTS
SETCITY = 'Псков' WHEREC_NO = 1;
```

Данный запрос нужно понимать так: найти запись, поле C_NO которой = 1 (это код клиента Иванова), и установить значение CITY равным "Псков".

Удаление записей. Оператор DELETE

Если нам необходимо удалить всех клиентов, номера которых превышают 5, то мы поступим следующим образом:

```
DELETE FROM CLIENTS
WHERE C_NO > 5;
```

С помощью оператора DELETE можно удалить все записи таблицы, указав условие, которое подойдет для всех записей, например:

```
DELETE FROM CLIENTS;
```

Если вторая часть оператора DELETE-WHERE не указана, значит, действие оператора распространяется на все записи сразу.

Выбор записей. Оператор SELECT

Добавление, изменение и удаление записей - это, конечно, очень важные команды, но вы часто будете использовать оператор SELECT, который выбирает данные из таблицы. Синтаксис этого оператора более сложен:

```
SELECT [DISTINCT|ALL] {*| [поле1 AS псевдоним] [,..., полеN AS псевдоним]}
FROM Имя_таблицы1 [,..., Имя_таблицыN]
[WHERE условие]
[GROUP BY список полей] [HAVING условие]
[ORDER BY список полей]
```

Мы полностью не будем рассматривать оператор SELECT, лучше это делать на конкретном примере. Сейчас мы рассмотрим оператор SELECT в общих чертах. Например, для вывода всех записей из таблицы CLIENTS сделайте следующее:

```
SELECT * FROM CLIENTS;
```

В результате вы получите ответ сервера, показанный в таблице 5.2.

Таблица 5.2 – ответ сервера

C_NO	FIO	ADDR	CITY	PHONE
1	Иванов И.И.	Вокзальная 3	Москва	09599911100
2	Иванов И.И.	Вокзальная 3	Москва	09599911100

2	Петров П.П.	Мира 29	Екатеринбург	3438920437

Обратите внимание на первые две записи - они одинаковые. Теоретически, добавление одинаковых записей возможно - ведь мы не указали первичный ключ таблицы. Если вы хотите исключить одинаковые записи из ответа сервера (но не из таблицы), используйте запрос:

```
SELECT DISTINCT *
FROM CLIENTS;
```

Предположим, вы хотите вывести только фамилию и номер телефона клиента, тогда используйте следующий запрос:

```
SELECT DISTINCT FIO, PHONE
FROM CLIENTS;
```

Если вам нужно вывести все товары, цена на которые превышает 800, то воспользуйтесь таким запросом:

```
SELECT *
FROM TOOLS
WHERE PRICE > 800;
```

Вы можете использовать следующие операторы отношений: <, >, =, <>, <=, >=.

Если в вашей таблице присутствуют несколько однофамильцев, то для вывода информации обо всех из них, используйте модификатор LIKE, например:

```
SELECT *
FROM CLIENTS
WHERE FIO LIKE '%Иван%';
```

Приведенный запрос можно причитать так: вывести информацию о клиентах, фамилия которых похожа на 'Иван'.

Если вам необходимо выбрать данные из разных таблиц, то перед именем поля нужно указывать имя таблицы. Вот запрос, который позволяет вывести имена всех клиентов, которые хотя бы один раз покупали товар:

```
SELECT DISTINCT CLIENTS.FIO
FROM CLIENTS, ORDERS
WHERE CLIENTS.C_NO = ORDERS.C_NO;
```

Оператор SELECT позволяет использовать вложенные запросы, однако MySQL их не поддерживает.

Внутренние функции MIN, MAX, AVG, SUM

При работе с оператором SELECT вам доступны несколько очень полезных внутренних функций MySQL, вычисляющих количество элементов (COUNT), сумму элементов (SUM), максимальное и минимальное значения (MAX и MIN), а также среднее значение (AVG).

Следующие операторы выведут, соответственно, колличество записей в таблице CLIENTS, самый дорогой товар и сумму цен всех товаров:

```
SELECT COUNT(*)
```

```
FROM CLIENTS;
SELECT MAX(PRICE)
FROM TOOLS;
SELECTSUM(PRICE)
FROMTOOLS;
```

Группировка записей

Оператор SELECT позволяет группировать возвращаемые значения. Например, клиент Иванов (C_NO=1) несколько раз заказывал какой-то товар. Значит, его номер встречается в таблице ORDERS несколько раз. Другой клиент также мог сделать несколько заказов. Мы можем сгруппировать все записи по полю C_NO (номер клиента), а затем вывести сумму заказа каждого клиента.

```
SELECT CLIENTS.FIO, SUM(ORDERS.AMOUNT) AS TOTALSUM
FROM CLIENTS, ORDERS
WHERE CLIENTS.C_NO = ORDERS.C_NO
GROUP BY ORDERS.C_NO;
```

Группировку выполняет оператор GROUP BY, который является частью оператора SELECT. Оператор GROUP BY можно ограничить с помощью HAVING. Этот оператор используется для отбора строк, возвращаемых GROUP BY. HAVING можно считать аналогом WHERE, но только для GROUP BY:

HAVING <условие>

Например, нас интересуют только клиенты, которые заказали товаров на общую сумму, превышающую 600:

```
SELECT CLIENTS.FIO, SUM(ORDERS.AMOUNT) AS TOTALSUM
FROM CLIENTS, ORDERS
WHERE CLIENTS.C_NO = ORDERS.C_NO
GROUP BY ORDERS.C_NO
HAVING TOTALSUM > 600;
```

ВэтомзапросемыиспользовалипсевдонимстолбцаTOTALSUM.

Сортировка записей

Пока мы не установили первичный ключ, сортировка таблицы не выполняется. Данные будут отображены в порядке их занесения в таблицу. Для сортировки по полю C_NO результата вывода таблицы CLIENTS используется следующий оператор (сама таблица при этом не сортируется):

```
SELECT *
FROM CLIENTS
ORDER BY C_NO;
```

Ключи

Предположим, что кто-то добавил в таблицу CLIENTS запись:

1 Сидоров Свободы 7 Калининград 0113452103

В то же время до этого номер 1 был закреплен за Ивановым. У нас получилось, что один и тот же номер сопоставлен разным клиентам. Чтобы избежать такой путаницы, необходимо использовать первичные ключи:

```
ALTERTABLECLIENTS
```

ADDPRIMARYKEY (C_NO)

После этого запроса поле C_NO может содержать только уникальные значения. В качестве первичного ключа нельзя использовать поле, допускающее значение NULL. Создать первичный ключ можно и проще - при создании таблицы следующим образом:

```
CREATE TABLE CLIENTS
(
C_NO    int      NOT NULL,
FIO     char(50) NOT NULL,
ADDR    char(55) NOT NULL,
CITY    char(20) NOT NULL,
PHONE   char(8)  NOT NULL,
PRIMARYKEY (C_NO);
);
```

Таблица ORDERS содержит сведения о заказах. По полю C_NO этой таблице идентифицируется заказчик. Предположим, что в таблицу ORDERS кто-то ввел значение, которого нет в таблице CLIENTS. Кто заказал товар? Нам нужно не допустить подобной ситуации, поэтому следует использовать подобный запрос:

```
ALTER TABLE ORDERS
ADD FOREIGN KEY(C_NO) REFERENCES CLIENTS;
```

Введенные в таблицу ORDERS номера клиентов C_NO должны существовать в таблице CLIENTS. Аналогично нужно добавить внешний ключ по полю T_NO. Эта возможность называется декларативной целостностью.

Команда ALTER используется не только для добавления ключей. Она предназначена для реорганизации таблицы в целом. Вы хотите добавить еще одно поле? Или установить список допустимых значений для каждого из полей. Все это можно сделать с помощью команды ALTER:

```
ALTER TABLE CLIENTS ADD ZIP char(7) NULL;
```

Этот оператор добавляет в таблицу CLIENTS новое поле ZIP типа char. Обратите внимание, что вы не можете добавить новое поле со значением NOT NULL в таблицу, в которой уже есть данные. Например, если компания работает только с клиентами Москвы и Екатеринбурга, то целесообразно ввести список допустимых значений для таблицы CLIENTS:

```
ALTER TABLE CLIENTS
ADD CONSTRAINT INVALID_STATE CHECK (CITY IN ('Москва',
'Екатеринбург'));
```

Использование внешних ключей

Теперь углубимся в изучение SQL. Вы уже знаете, как добавлять первичный ключ, теперь добавим внешний ключ при создании таблицы. Внешние ключи используются для связи одной таблицы с другой. Например, в таблице CLIENTS есть два клиента - Иванов (C_NO=1) и Петров (C_NO=2). Оператор в магазине при оформлении заказа ошибся и указал несуществующий номер, например, C_NO=3. Как мы потом сможем

идентифицировать клиента? Для решения такой проблемы и существуют внешние ключи:

```
CREATETABLET
(
/* Описаниеполейтаблицы */
FOREING KEY KEY_NAME (LIST)
REFERENCES ANOTHER_TABLE [(LIST2)]
[ON DELETE OPTION]
[ON UPDATE OPTION]
);
```

Здесь:

- KEY_NAME - Имя ключа. Имя не является обязательным, но рекомендуется всегда указывать имя ключа - если вы не укажите имя ключа, вы потом не сможете его удалить;
- LIST - это список полей, входящих во внешний ключ. Список разделяется запятыми;
- ANOTHER_TABLE - это другая таблица, по которой устанавливается не внешний ключ, а необязательный элемент;
- LIST2 - это список полей этой таблицы. Типы полей в списке LIST должны совпадать с типами полей в списке LIST2.

Предположим, что в первой таблице у нас есть поля - NO и NAME - целого и символьного типов соответственно. Во второй таблице у нас есть поля с одинаковыми именами и тапами. Определениевнешнегоключа:

```
FOREIGN KEY KEY_NAME (NO, NAME)
REFERENCES ANOTHER_TABLE (NAME, NO)
```

Это определение некорректно, потому что типы полей NO и NAME не совпадают. Нужно использовать такое определение:

```
FOREIGN KEY KEY_NAME (NO, NAME)
REFERENCES ANOTHER_TABLE (NO, NAME)
[ON DELETE <OPTION>]
[ON UPDATE <OPTION>]
```

Если поля имеют одинаковые имена, как в нашем случае, список LIST2 лучше вообще не указывать.

Необязательные параметры ON DELETE <OPTION> и ON UPDATE <OPTION> определяют действие по обновлению информации в базе данных, при удалении информации из таблицы и при ее обновлении. А действия могут быть следующими:

- CASCADE - удаление или обновление значений везде, где оно встречается. Например, у нас есть таблица клиентов и заказов. Ы хотим удалить запись клиента с номером C_NO=1. Из таблицы заказов будут удалены сведения обо всех заказах, сделанных клиентом;
- NOACTION - вы не сможете удалить информацию из таблицы клиентов до тех пор, пока вы не удалите все заказы, сделанные этим клиентом.

То есть действие NOACTION запрещает удалять запись из основной таблицы, если она используется в дочерней таблице;

- SETNULL - все значения в дочерней таблице будут заменены на NULL (если значения NULL допускаются);
- С помощью параметра SET_DEFAULT вы можете указать значение по умолчанию. Например, если вы укажите SET_DEFAULT 1, то при удалении клиента с любым номером его заказы бубудт приписываться клиенту с номером 1, который есть в таблице CLIENTS.

Удаление полей и таблиц. Оператор DROP

Стандартом SQL не предусмотрено удаление столбцов, однако в MySQL мы это можем сделать:

ALTER TABLE CLIENTS DROP ZIP;

А удалить таблицу еще проще: DROP ORDERS;

Отключение от СУБД

Используя запрос DISCONNECT можно отключиться от используемой базы данных, а затем, используя запрос CONNECT, подключиться к другой базе данных. В некоторых серверах SQL запрос DISCONNECT не работает, а вместо CONNECT применяется запрос USE.

При использовании PHP нет необходимости использовать данные запросы, поскольку для отключения от сервера MySQL используется функция mysql_close(), а для подключения к серверу MySQL используется функция mysql_connect().

Методика и порядок выполнения работы

Создайте базу данных, состоящую из трех таблиц (CLIENTS, ORDERS, TOOLS), заполните ее данными, используя язык запросов SQL(рисунки 5.1 и 5.2). Установите для таблиц ключевые поля, запрет на ввод нулевых (неопределенных) значений. Используя запросы, проверьте целостность базы данных.

Ввести запрос на вкладке SQL и нажать кнопку Go

Рисунок 5.1 – Создание БД

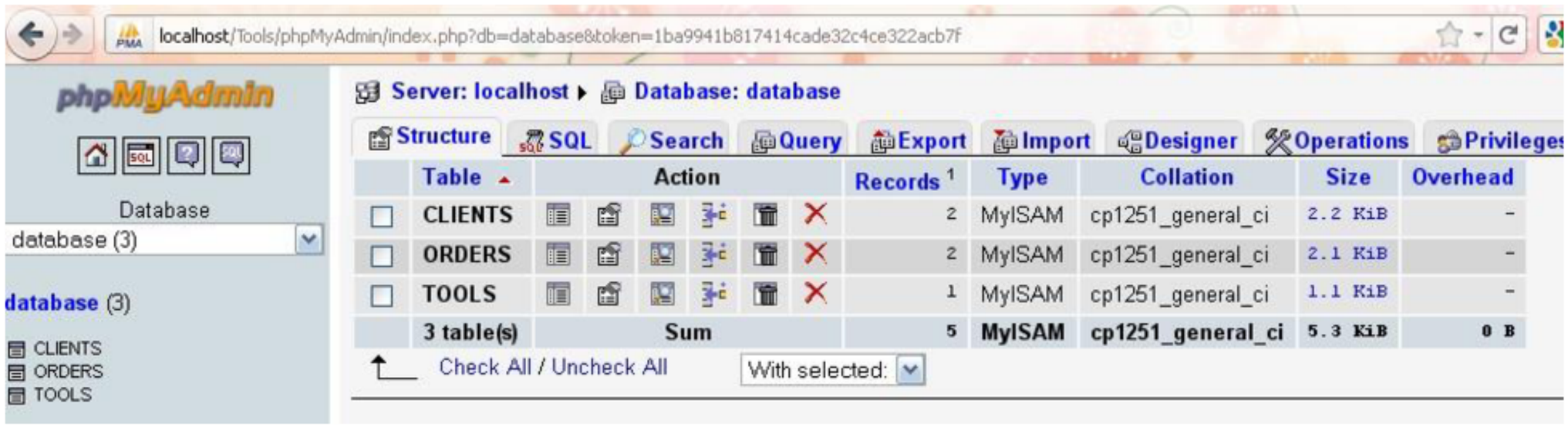

Рисунок 5.2 – Работа с БД

Задание

Создайте БД, состоящую из 3 таблиц.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) созданных таблиц с указанием ключевых полей и указанием запрета неопределенных значений.

Вопросы для защиты работы:

1. Какие атрибуты могут быть первичными ключами?
2. Каким образом можно запретить ввод неопределенных значений?

Лабораторная работа №6. Функции PHP для работы с MySQL

Цель работы: научиться использовать php функции для работы с базой данных MySQL.

Теоретическое обоснование

Рассмотрим основные функции PHP, применяемые для работы с MySQL сервером.

Функции соединения с сервером MySQL

Основной функцией для соединения с сервером MySQL является mysql_connect(), которая подключает скрипт к серверу баз данных MySQL и выполяет авторизацию пользователя базой данных.

Синтаксисуданнойфункциитакой:

mysql_connect ([string $hostname] [, string $user] [, sting $password]);

Как вы наверно заметили, все параметры данной функции являются необязательными, поскольку значения по умолчанию можно прописать в конфигурационном файле php.ini. Если вы хотите указать другие имя MySQL-хоста, пользователя и пароль, вы всегда можете это сделать.

Параметр $hostname может быть указан в виде: хост:порт.

Функция возвращает идентификатор (типа int) соединения, вся дальнейшая работа осуществляется только через этот идентификатор. При следующем вызове функции mysql_connect() с теми же параметрами новое

соединение не будет открыто, а функция возвратит идентификатор существующего соединения.

Для закрытия соединения предназначена функция mysql_close(int $connection_id).

Вообще, соединение можно и не закрывать - оно будет закрыто автоматически при завершении работы PHP скрипта. Если вы используете более одного соединения, при вызове mysql_close() нужно указать идентификатор соединения, которое вы хотите закрыть. Вообще не закрывать соединения - плохой стиль, лучше закрывать соединения с MySQL самостоятельно, а не надеясь на автоматизм PHP, хотя это ваше право.

Если вы будете использовать только одно соединение с базой данных MySQL за все время работы сценария, можно не сохранять его идентификатор и не указывать идентификатор при вызове остальных функций.

Функция mysql_connect() устанавливает обыкновенное соединение с MySQL. Однако, PHP поддерживает постоянные соединения - для этого используйте функцию mysql_pconnect(). Аргументы этой функции такие же, как и у mysql_connect().

В чем разница между постоянным соединением и обыкновенным соединением с MySQL? Постоянное соединение не закрывается после завершения работы скрипта, даже если скрипт вызвал функцию mysql_close(). Соединение привязывается к PID потомка веб сервера Apache (от имени которого он и работает) и закрывается лишь тогда, когда удаляется процесс-владелец (например, при завершении работы или перезагрузке вебсервера Apache).

PHP работает с постоянными соединениями примерно так: при вызове функции mysql_pconnect() PHP проверяет, было ли ранее установлено соединение. Если да, то возвращается его идентификатор, а если нет, то открывается новое соединение и возвращается идентификатор.

Постоянные соединения позволяют значительно снизить нагрузку на сервер, а также повысить скорость работы PHP скриптов, использующих базы данных.

При работе с постоянными соединениями нужно следить, чтобы максимальное число клиентов Apache не превышало максимального числа клиентов MySQL, то есть параметр MaxClient (в конфигурационном файле Apache - httpd.conf) должен быть меньше или равен параметру max_user_connection (параметр MySQL).

Функция выбора базы данных

Функция mysql_select_db (string $db [, int $id]) выбирает базу данных, с которой будет работать PHP скрипт. Если открыто не более одного соединения, можно не указывать параметр $id.

// Попытка установить соединение с MySQL: if (!mysql_connect($server, $user, $password)) { die("Ошибка подключения к серверу MySQL"); exit; } // Соединились, теперь выбираем базу данных: mysql_select_db($db); Функции обработки ошибок

Если произойдет ошибка соединения с MySQL, то вы получите соответствующее сообщение и скрипт завершит свою работу. Это не всегда бывает удобно, прежде всего, при отладке скриптов. Поэтому, в PHP есть следующие две функции:

- mysql_errno(int $id);
- mysql_error(int $id);

Первая функция возвращает номер ошибки, а вторая - сообщение об ошибке. В результате мы можем использовать следующее:

echo "ERROR ".mysql_errno()." ".mysql_error()."\n";

Теперь вы будете знать, из-за чего произошла ошибка - вы увидите соответствующим образом оформленное сообщение.

Функции выполнения запросов к серверу баз данных

Все запросы к текущей базе данных отправляются функцией mysql_query(). Этой функции нужно передать один параметр - текст запроса. Текст запроса модет содержать пробельные символы и символы новой строки (\n). Текст должен быть составлен по правилам синтаксиса SQL.

Пример запроса:

$q = mysql_query("SELECT * FROM mytable");

Приведенный запрос должен вернуть содержимое таблицы mytable. Результат запроса присваивается переменной $q. Результат - это набор данных, который после выполнения запроса нужно обработать определенным образом.

Функции обработки результатов запроса

Если запрос, выполненный с помощью функции mysql_query() успешно выполнился, то в результате клиент получит набор записей, который может быть обработан следующими функциями PHP:

- mysql_result() - получить необходимый элемент из набора записей;
- mysql_fetch_array() - занести запись в массив;
- mysql_fetch_row() - занести запись в массив;
- mysql_fetch_assoc() - занести запись в ассоциативный массив;
- mysql_fetch_object() - занести запись в объект.

Также можно определить количество содержащихся записей и полей в результате запроса. Функция mysql_num_rows() позволяет узнать, сколько записей содержит результат запроса:

$q = mysql_query("SELECT * FROM mytable"); echo "Втаблице mytable ".mysql_num_rows($q)." записей";

Запись состоит из полей (колонок). С помощью функции mysql_num_fields() можно узнать, сколько полей содержит каждая запись результата:

$q = mysql_query("SELECT * FROM mytable"); echo "Втаблице mytable ".mysql_num_fields($q)." полей ";

У нас также есть возможность узнать значение каждого поля. Это можно сделать с помощью следующей функции:

mysql_result (int $result, int $row, mixed $field);

Параметр функции $row задает номер записи, а параметр $field - имя или порядковый номер поля.

Предположим, SQL-запрос вернул следующий набор данных:

Email Name Last_Name ----------

ivanov@mail.ru Ivan Ivanov petrov@mail.ru Petr Petrov

Вывести это в браузер можно следующим образом:

```
$rows = mysql_num_rows($q);
$fields = mysql_num_fields($q);
echo "<pre>";  for ($c=0; $c<$rows; $c++) {  for ($cc=0; $cc<$fields; $cc++) {  echo mysql_result($q, $c, $cc)."\t";  echo "\n";
  }
 }
echo "</pre>";
```

Следует отметить, что функция mysql_result() универсальна: зная количество записей и количество полей, можно "обойти" весь результат, но в тоже время, скорость работы данной функции достаточно низка. Поэтому, для обработки больших наборов записей рекомендуется использовать функции mysql_fecth_row(), mysql_fecth_array(), и.т.д.

Функция mysql_fecth_row(int $res) получает сразу всю строку, соответствующую текущей записи результата $res. Каждый следующий вызов функции перемещает указатель запроса на следующую позицию (как при работе с файлами) и получает следующую запись. Если более нет записей, то функция возвращает FALSE. Пример использования данной функции:

```
$q = mysql_query("SELECT * FROM mytable WHERE month=\"$db_m\"
AND day=\"$db_d\"); for ($c=0; $c<mysql_num_rows($q); $c++)
{
$f = mysql_fetch_row($q); echo $f;
}
```

Использовать функцию mysql_fecth_row() не всегда удобно, так как значения всех полей одной записи находятся все в одной строке. Удобнее использовать функцию mysql_fetch_array(), которая возвращает ассоциативный массив, ключами которого будут имена полей.

Функция mysql_fecth_array(int $res [, int $result_type]) возвращает не ассоциативный массив, а массив, заданный необязательным параметром $result_type, который может принимать следующие значения:

- MYSQL_ASSOC - возвращает ассоциативный массив;
- MYSQL_NUM -возвращает массив с числовыми индексами, как в функции mysql_fecth_row();
- MYSQL_BOTH - возвращает массив с двойными индексами, то есть вы можете работать с ним, как с ассоциативным массивом и как со списком (MYSQL_BOTH - это значение по умолчанию для параметра $result_type.

В PHP есть функция, возвращающая ассоциативный массив с одним индексом:

mysql_fetch_assoc(int $res);

Фактически, данная функция является синонимом для mysql_fetch_array($res, MYSQL_ASSOC);

Пример использования функции mysql_fecth_array():

$q = mysql_query("SELECT * FROM mytable WHERE month=\"$db_m\" AND day=\"$db_d\"); for ($c=0; $c<mysql_num_rows($q); $c++)

{

$f = mysql_fetch_array($q); echo "$f[email] $f[name] $f[month] $f[day]
"; }

Как видно, использовать функцию mysql_fetch_array() намного удобнее, чем mysql_fetch_row().

Функции получения информации о результатах SQL-запросов

PHP предоставляет еще несколько полезных функций, которые позволяют узнать информацию о результатах SQL-запросов.

- Функция mysql_field_name(int $result, int $offset) возвращает имя поля, находящегося в результате $result с номером $offset (нумерация начинается с 0). Другими словами, функция возвращает имя поля с номером $offset.
- Функция mysql_field_type(int $result, int $offset) возвращает тип поля с номером $offset в результате $result (номер задается относительно результата, а не таблицы);
- Функция mysql_field_flags(int $result, int $offset) возвращает пречисленные через пробел флаги (модификаторы), которые имеются у поля с номером $offset. Перечислим все поддерживаемые MySQL флаги (Таблица 6.1).

Таблица 6.1 – Флаги MySQL

Флаг	Описание
not_Null	Поле не может содержать неопределенного значения (NULL), то есть поле должно быть явно инициализировано
Primary_Key	Поле будет первичным ключом - идентификатором записи, по которому можно однозначно идентифицировать запись;
auto_increment	При вставке новой записи значение этого поля будет автоматически увеличено на единицу, потому в таблице никогда не будет двух записей с одинаковым значением этого поля;
Unique_Key	Поле должно содержать уникальное значение;
Multiple_Key	Индекс
Blob	Поле может содержать бинарный блок данных
Unsigned	Поле содержит беззнаковые числа

Zerofill	Вместо пробелов используются символы с кодом \0
Binary	Поле содержит двоичные данные
enum	Поле может содержать один элемент из нескольких возможных (элемент перечисления)
timestamp	В поле автоматически заносится текущая дата и время при его модификации

Функция mysql_field_flags() возвращает флаги в виде строки, в которой флаги разделяются пробелами.

Практический пример использования функций PHP-MySQL

Скрипт вывода содержимого таблицы MySQL в виде HTML:

```
<?php
$host = "localhost";
$user = "user";
$password = "secret_password";

// Производим попытку подключения к серверу MySQL:
if (!mysql_connect($host, $user, $password))
{
echo "<h2>MySQL Error!</h2>"; exit;
}

// Выбираем базу данных:
mysql_select_db($db);

// Выводим заголовок таблицы:
echo "<table border=\"1\" width=\"100%\" bgcolor=\"#FFFFE1\">"; echo
"<tr><td>Email</td><td>Имя</td><td>Месяц</td>"; echo
"<td>Число</td><td>Пол</td></tr>";

// SQL-запрос:
$q = mysql_query ("SELECT * FROM mytable");

// Выводим таблицу:
for ($c=0; $c<mysql_num_rows($q); $c++)
{ echo "<tr>";

$f = mysql_fetch_array($q); echo
"<td>$f[email]</td><td>$f[name]</td><td>$f[month]</td>"; echo
"<td>$f[day]</td><td>$f[s]</td>";
echo "</tr>";
} echo "</table>";
```

```
?>
```

Методика и порядок выполнения работы

Создадим в папке denver-home папку для нашего скрипта. Назовем ее new. В папке new создадим папку www. В папке www создадим файл index.php, в который впишем следующий код, выводящий содержимое таблицы CLIENTS:

```
<?php
$host = "localhost";
$user = "root";//пользователь, созданный по умолчанию в базе данных без
пароля
$password = "";

// Производим попытку подключения к серверу MySQL:
if (!mysql_connect($host, $user, $password))
{
echo "<h2>MySQL Error!</h2>"; exit; }
$db="database";//указываем нашу базу данных
// Выбираем базу данных: mysql_select_db($db);

// Выводимзаголовоктаблицы: echo "<table border=\"1\" width=\"100%\"
bgcolor=\"#FFFFE1\">"; echo
"<tr><td>Номер</td><td>ФИО</td><td>Адрес</td>"; echo
"<td>Город</td><td>Номер</td></tr>";

// SQL-запрос:
$q = mysql_query ("SELECT * FROM ClIENTS");

// Выводимтаблицу: for ($c=0; $c<mysql_num_rows($q); $c++)
{ echo "<tr>";

$f = mysql_fetch_array($q); echo
"<td>$f[C_NO]</td><td>$f[FIO]</td><td>$f[ADDR]</td>"; echo
"<td>$f[CITY]</td><td>$f[PHONE]</td>";

echo "</tr>";
} echo "</table>";
?>
```

В результате получаем таблицу 6.2.

Таблица 6.2 – Таблица CLIENTS

Номер	ФИО	Адрес	Город	Номер
1	Иванов И.И.	Вокзальная 3	Псков	0959991110
2	Петров	Мира 29	Екатеринбург	-
3	Иванов		Псков	

Выведем записи полей всех таблиц

```
        $f = mysql_fetch_array($q); echo
"<td>$f[C_NO]</td><td>$f[FIO]</td><td>$f[ADDR]</td>"; echo
"<td>$f[CITY]</td><td>$f[PHONE]</td>";

        echo "</tr>";
        } echo "</table><br><br><br><br>";

        // ВыводимзаголовоктаблицыЗаказы:
        echo "<table border=\"1\" width=\"100%\" bgcolor=\"#FFFFE1\">"; echo
"<tr><td>Номер</td><td>Дата</td>"; echo
        "<td>Клиент</td><td>Номер</td><td>Точность</td><td>цена</td></tr>";

        // SQL-запрос:
        $q = mysql_query ("SELECT * FROM ORDERS");

        // Выводимтаблицу: for ($c=0; $c<mysql_num_rows($q); $c++)
        { echo "<tr>"; $f = mysql_fetch_array($q); echo
"<td>$f[O_NO]</td><td>$f[DATE]</td><td>$f[C_NO]</td>"; echo
        "<td>$f[T_NO]</td><td>$f[QUANTITY]</td><td>$f[AMOUNT]</td>";

        echo "</tr>";
        } echo "</table><br><br><br><br>";

        // ВыводимзаголовоктаблицыИнструменты: echo "<table border=\"1\"
width=\"100%\" bgcolor=\"#FFFFE1\">"; echo
"<tr><td>Номер</td><td>Название</td>"; echo
"<td>Цена</td><td>Количество</td></tr>";

        // SQL-запрос:
        $q = mysql_query ("SELECT * FROM TOOLS");

        // Выводимтаблицу: for ($c=0; $c<mysql_num_rows($q); $c++)
```

```
        $f = mysql_fetch_array($q); echo
"<td>$f[T_NO]</td><td>$f[DSEC]</td><td>$f[PRICE]</td>"; echo
"<td>$f[QTY]</td>";

        echo "</tr>";
        } echo "</table>";
        ?>
```

Организуем ввод данных в базу данных.

Сначала создадим форму для ввода данных. Добавим в код файла index.php

```
<br><br><br>
<form action="vvod.php" method="post">
<table width="60%" border="0" >
<tr>
<td width="34%">Номер</td>
<td width="66%"><input name="c_no" type="text" size="13"
maxlength="13" /></td>
</tr>
<tr>
<td>ФИО</td>
<td><input name="fio" type="text" size="30" maxlength="30" /></td>
</tr>
<tr>
<td>Адрес</td>
<td><input name="addr" type="text" size="30" maxlength="60" /></td>
</tr>
<tr>
<td>Город</td>
<td><input name="city" type="text" size="8" maxlength="8" /></td>
</tr>
<tr>
<td>Телефон</td>
<td><input name="phone" type="text" size="8" maxlength="8" /></td>
</tr>
<tr>
<td colspan="2"><input name="input" type="submit"
value="Зарегистрировать" /></td>
</tr>
</table>
</form>
</table>
```


Создадим скрипт vvod.php, обрабатывающий введенные в форму данные

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.111.ru">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251" />
<title>Запись в базу данных</title>
</head>

<body>
<h1>Запись в базу данных</h1>
<?php

/*if (!$c_no || !$fio || !$addr || !$city || $phone)
{
echo 'Вы ввели не все необходимые сведения. <br>'
.'Пожалуйста, вернитесь на предыдущую страницу и повторите ввод.';
exit;
}
/*if (!get_magic_quotes_gpc())
{
$isbn = addslashes ($isbn);
$author = addslashes ($author);
$title = addslashes ($title);
$price = doubleval ($price);
}*/
// Производим попытку подключения к серверу MySQL:
$host = "localhost";
$user = "ya"; $password = "1111"; if (!mysql_connect($host, $user,
$password))
{
echo "<h2>MySQL Error!</h2>";
exit;
}
$db="database";
// Выбираем базу данных: mysql_select_db($db);
//создание коротких имен переменных
$c_no = $_POST['c_no'];
$fio = $_POST['fio'];
$addr = $_POST['addr'];
$city = $_POST['city'];
$phone = $_POST['phone'];
$query = "insert into CLIENTS values(", $fio, $addr, $city, $phone,")";
```

```
$result = mysql_query($query) or die("Query failed"); echo $result; if ($result) echo $db -> affected_rows. " добавлено.";
?>
</body>
</html>
```

Не забудьте закрыть соединение с базой данных в скрипте index.php: mysql_close();

Проверяем работоспособность скрипта. Результаты представлены в таблицах 6.3а – 6.3д.

Номер	ФИО	Адрес	Город	Номер
1	Иванов И.И.	Вокзальная 3	Псков	09599911100
2	Петров	Мира 29	Екатеринбург	-

Номер	Дата	Клиент	Номер	Точность	цена
1	2020-01-03	1	1	1.00	340.98
1	2010-01-03	1	1	1.00	340.98

Номер	Название	Цена	Количество
1	Клавиатура ABC	340.98	5.00

Номер 15
ФИО 15
Адрес 15
Город 15
Телефон 15
Зарегистрировать

Запись в базу данных

1 добавлено.

Таблица 6.3 Результаты работы скрипта

б

в

г

д

Проверяем, добавлена ли запись в базу данных (таблица 6.4)

Таблица 6.4 – Добавление записей

			C_NO	FIO	ADDR	CITY	PHONE	ZIP
☐	✎	✕	1	Иванов И.И.	Вокзальная 3	Псков	09599911100	*NULL*
☐	✎	✕	2	Петров	Мира 29	Екатеринбург	-	*NULL*
☐	✎	✕	0	15	15	15	15	

Запись добавлена, но, поскольку мы не вписывали в запросе номер для клиента, то добавилось нулевое значение. Логичнее будет изменить поле C_NO, сделав его автоикрементом (рисунок 6.1).

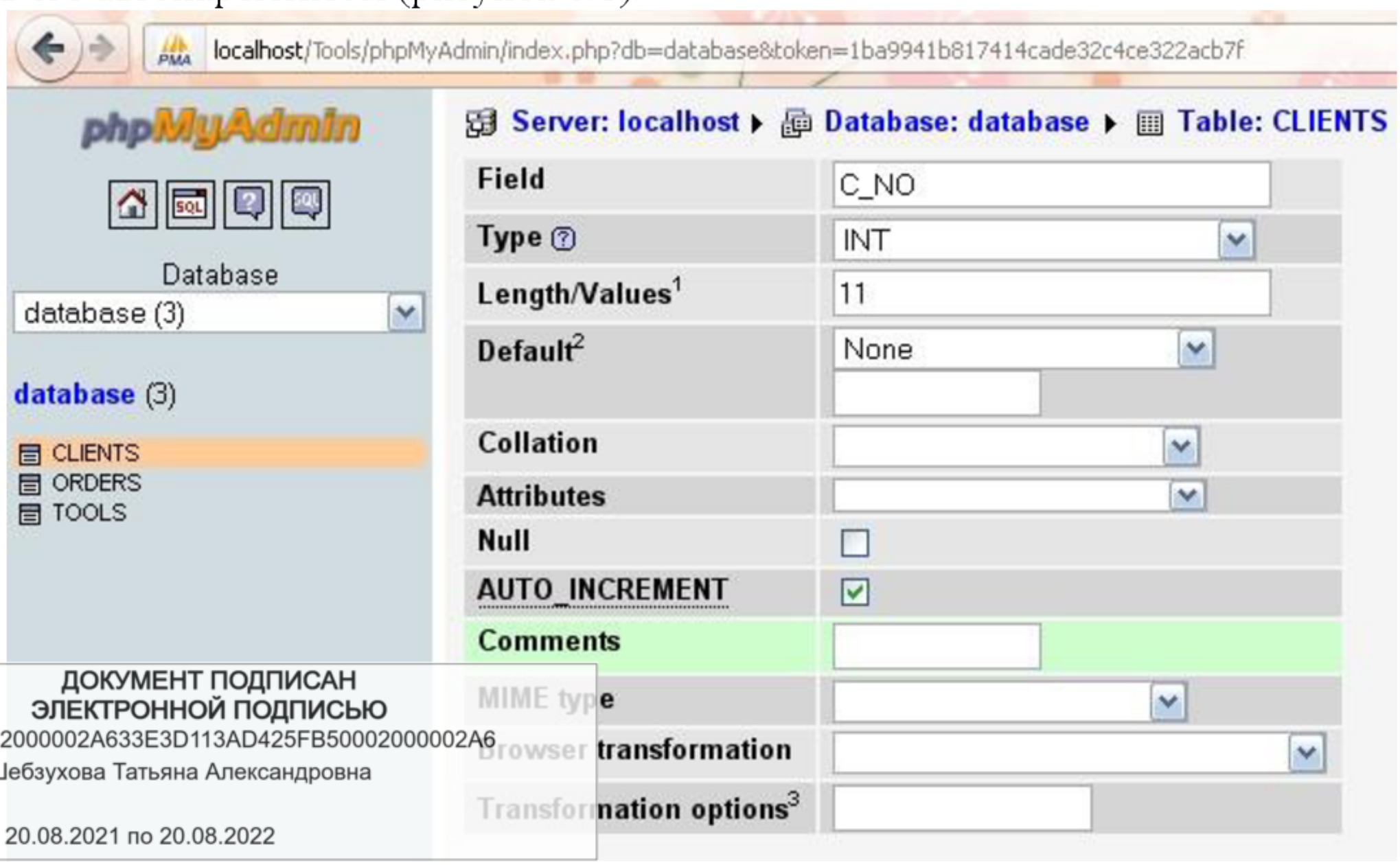

Рисунок 6.1 – Добавление AUTO_INCREMENT

Проверьте, как будет добавляться запись теперь.

Задание

1. Организовать ввод данных для всех таблиц.
2. С помощью web –интерфейса организовать выборку из базы данных по заданному пользователем условию.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) описание добавления строк в таблицы.

Вопросы для защиты работы:

1. Каким образом осуществляется заполнение таблиц?
2. Для чего используется AUTO_INCREMENT?

Лабораторная работа №7. Удаление данных из БД. Обновление данных в БД

Цель работы: научиться удалять и обновлять записи в таблицах базы данных MySQL.

Теоретическое обоснование

В предыдущей работе вы научились вставлять и запрашивать данные из базы данных (БД). В этой работе мы узнаем, как удалять записи из БД, что значительно проще, чем вставка данных.

Удаление данных с помощью SQL

Синтаксис SQL-оператора удаления записей таков:

DELETEFROMTableNameWHERECondition

При удалении записи можно использовать уникальное поле AutoNumber в базе данных. В нашей БД это столбец C_NO таблицы Clients. Использование этого уникального идентификатора гарантирует, что удаляется только одна запись. Удалим клиента с номер 3.

```
<html>
<head>
<title>Удаление данных из БД</title>
</head>
<body>
<?php
// Создание связи с сервером БД mysql_connect("localhost", "root", "") or die
(mysql_error());
// Выбор БД mysql_select_db("database") or die(mysql_error()); // SQL-
оператор, удаляющийзапись
```

```
$strSQL = "DELETE FROM CLIENTS WHERE C_NO = 3";
mysql_query($strSQL);
// Закрыть соединение с БД  mysql_close();
?>
<h1>Записьудалена!</h1>
</body>
</html>
```

Помните, что не существует никакой "Recycle Bin" при работе с БД и PHP. Если вы удалили запись, то восстановить её будет невозможно.

Обновление данных с помощью SQL

Синтаксис SQL-оператора обновления полей таблицы:

```
UPDATE TableName SET TableColumn='value' WHERE condition
```

Можно также обновлять несколько ячеек за раз, используя один оператор SQL:

```
UPDATE TableName SET TableColumn1='value1',
TableColumn2='value2' WHERE condition
```

Следующий код обновляет FIO в CLIENTS на «Петров» и меняет телефонный номер на 44444444. Прочая информация не изменяются. Можете попробовать изменить другие данные, создав собственные SQL-операторы.

```
<html>
<head>
<title>Обновление данных в БД</title>
</head>
<body>
<?php
// Соединение с сервером БД  mysql_connect("localhost", "root", "") or die
(mysql_error ());
// ВыборБД  mysql_select_db("database") or die(mysql_error());
// Построение SQL-оператора
$strSQL = "Update CLIENTS set ";
$strSQL = $strSQL. "FIO= 'Петров', ";
$strSQL = $strSQL. "Phone= '44444444' ";
$strSQL = $strSQL. "Where C_NO = 3";
// SQL-оператор выполняется  mysql_query($strSQL);
// Закрыть соединение с БД  mysql_close();  ?>
```

Задание.

1. Организовать обновление базы данных посредством web интерефейса (кнопкой).
2. Организовать удаление записей базы данных посредством web интерефейса, определенных пользователем.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) описание удаления строк из таблицы.

Вопросы для защиты работы:

1. Каким образом осуществляется удаление таблиц?
2. Каким образом осуществляется удаление строк?

Лабораторная работа №8. Введение в Php сессии

Цель работы: научиться удалять и обновлять записи в таблицах базы данных MySQL.

Теоретическое обоснование

При посещении сайта вы выполняете различные действия. Переходите с одной страницы на другую. Возможно, заполняете форму или покупаете что-то. Это очень важно учитывать при создании успешных веб-проектов.

Предположим, например, что вы хотите создать сайт, на котором несколько страниц защищены логином и паролем. Чтобы эта защита действовала эффективно, защищённые паролем страницы должны иметь доступ к информации о том, зашёл ли пользователь ранее в систему. Вы должны, иначе говоря, "помнить", что пользователь делал до этого.

Именно об этом наша лабораторная работа - как использовать сессии в PHP для сохранения и получения информации в процессе визита пользователя на наш сайт.

PHP-сессии дают возможность работать с информацией о пользовательской сессии. Вы можете создавать приложения, которые идентифицируют и собирают информацию о пользователях.

Сессии могут начинаться разными способами. Мы не будем углубляться в технические тонкости, а сконцентрируемся на варианте, когда сессия начинается с сохранения значения. Сессия заканчивается (dies), если пользователь не запрашивает страниц в течение какого-то времени (стандартное значение - 20 минут). Разумеется, вы в любой момент можете закончить сессию в вашем скрипте.

Скажем, 50 пользователей просматривают страницы одного сайта, например, веб-магазина. Информацию о том, что у каждого посетителя в корзине, лучше всего сохранить в сессии. Чтобы идентифицировать пользователей, сервер использует уникальные пользовательские идентификаторы/user ID, которые хранятся в куках. Кука – это небольшой текстовый файл, хранимый на компьютере пользователя. Следовательно, сессии часто требуют поддержки кук в браузерах пользователей.

При запросе страницы сохраним текущее время в сессии. Добавим следующую строку в PHP-скрипт:

```
<?php

session_start();
$_SESSION["Start_Time"] = date("r");
?>
```

Таким образом, сессия началась. Как сказано выше, каждая сессия получает ID от сервера.

Ваша сессия имеет следующий ID: dae18cd921b51edf04b358d76d15e7e7

В любое время можно вызвать "StartTime" из сессии, введя:

```
<?php
session_start();  echo $_SESSION["StartTime"];
?>
```

что покажет, что страница была запрошена в (в соответствии с временем данного вэб-сервера).

Но интересно, что эта информация остаётся в сессии, даже после выхода со страницы. Эта информация будет сопровождать, пока сессия не завершится.

По умолчанию сессия длится, пока пользователь не закроет окно браузера, и тогда она заканчивается автоматически. Но если вы хотите принудительно завершить сессию, её всегда можно закончить таким образом:

```
<?php
session_destroy();
?>
```

Посмотрим другой пример использования сессий: с паролем. Создадим простейшую систему с логином. Первое, что необходимо, это форма, в которой пользователи могу указывать username и password.

Онаможетвыглядетьтак (index.html):

```
<html>
<head>
<title>Login</title>
</head>
<body>
<form method="post" action="login.php">
<p>Username: <input type="text" name="username" /></p>
<p>Password: <input type="text" name="password" /></p>
<p><input type="submit" value="Войти" /></p>
</form>
</body>
</html>
```

Затемсоздадимфайлlogin.php. Вэтомфайлемыпроверяем, введеныликорректныеusernameиpassword. Если это так, мы начинаем сессию, в которой указано, что пользователь вошёл с корректными username и password.

```
<?php
session_start();
?>
<html>
```

```
<head>
<title>Login</title>
</head>
<body>
<?php
// Проверитькорректностьusernameиpassword  if ($_POST["username"] ==
"1" && $_POST["password"] == "1") {
// Если корректны, устанавливаем значение сессии в YES
$_SESSION["Login"] = "YES";  echo "<h1>Вызашли</h1>";  echo
"<p><a href='document.php'>Ссылка на защищённый
файл</a><p/>";  }  else {
// Если некорректны, устанавливаем сессию в NO  session_start();
$_SESSION ["Login"] = "NO";  echo "<h1>Вы зашли НЕкорректно </
h1>";  echo  "<p><ahref='document.php'>Ссылка на защищённый
файл</a><p/>";
}
?>
</body>
</html>
```

При работе с защищёнными файлами мы проверяем, вошёл ли пользователь с корректным логином. Если нет, то пользователь отправляется обратно к логин-форме. Вот как делается эта защита:

```
<?php
// Начать вашу PHP-сессию  session_start();
// Если пользователь не зашёл, отправить его/её к логин-форме  if
($_SESSION["Login"] != "YES") {  header("Location: index.html"); echo "Вы
можете получить к нему доступ, только если вошли в
систему.";
}
?>
<html>
<head>
<title>Логин</title>
</head>  <body>
<h1>Этот документ защищён</h1>
<p></p>
</body>
</html>
```


Задание

1. Организовать сессию в вашей системе без пароля.

2. Организовать сессию вашей системы с паролем и логином.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) описание сессии без пароля;
3) описание сессиис паролем и логином.

Вопросы для защиты работы:

1. Как использовать сессии в PHP для сохранения и получения информации?
2. Каким образом используются пароль и логин?

Лабораторная работа №9. Передача переменных через URL

Цель работы: научиться использовать значения переменных, передаваемых через URL

Теоретическое обоснование

При работе с PHP часто необходимо передать переменные с одной страницы в другую. Возможно, вас удивляло, почему некоторые URL выглядят так:

http://html.net/page.php?id=1254

Почему после имени страницы стоит знак вопроса? Ответ: символы после знака вопроса это строка HTTP-запроса. Строка HTTP-запроса может содержать как имена переменных, так и их значения. В вышеприведённом примере строка HTTP-запроса содержит переменную "id" со значением "1254".

Вот другой пример: http://html.net/page.php?name=Joe

То есть снова переменная ("name") со значением ("Joe"). Как получить переменную с помощью PHP? Предположим, у вас есть PHP-страница people.php. Можно вызвать её с использованием URL:

people.php?name=Joe

В PHP вы можете получить значение переменной 'name' таким образом:

$_GET["name"]

То есть мы используем$_GET для поиска значения именованной переменной. Попробуем на примере:

```
<html>
<head>
<title>Строка запроса</title>
</head>
<body>
<?php// Значение переменной найдено  echo "<h1>Hello ". $ _GET["name"]. "</h1>";
?>
</body>
</html>
```

Попробуйте в этом примере заменить "Joe" вашим собственным в URL и снова вызвать документ. Опишите полученные результаты.

В URL можно передавать и не одну переменную. Разделяя переменные знаком &, можно передавать несколько:

people.php?name=Joe&age=24

Этот URL содержит две переменные: name и age. Как и ранее, можно получит

$_GET["name"]
$_GET["age"]

Добавим в пример ещё одну переменную:

```
<html>
<head>
<title>Строказапроса</title>
</head>
<body>
<?php
// Значение имени переменной name найдено  echo "<h1>Hello ". $_GET["name"]. "</h1>";
// Значение имени переменной age найдено  echo "<h1>You are ". $_GET["age"]. " years old </h1>";
?>
</body>
</html>
```

Методика и порядок выполнения работы Изучить передачу переменных по приведенным примерам.

Задание

1. Организовать передачу переменных по приведенным примерам.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;

2) описание передачи переменныхчерез URL.

Вопросы для защиты работы:

1. Дайте характеристику URL?
2. Каким образом осуществляется передача переменныхчерез URL?

Лабораторная работа №10.Использование внешнего файла для хранения и загрузки внешних данных

Цель работы: научиться организовывать использование внешнего файла для хранения и загрузки внешних данных.

Теоретическое обоснование

Текстовые файлы отлично подходят для хранения разного рода данных. Они не так гибки, как базы данных, но обычно не требую такого количества памяти. Более того, текстовые файлы имеют формат, который читается на большинстве систем.

Для открытия текстового файла используем функцию fopen. Вот её синтаксис:

fopen(filename, mode) где filename – имя открываемого файла.

mode – Mode/Режим может быть "r" (reading/чтение), "w" (writing/запись) или "a" (appending/присоединение). В этом уроке мы будем только читать из файла и, соответственно, используем "r".

Создадим файл unitednations.txt, впишем в него информацию и попробуем открыть его:

```
<?php
// Открыть текстовый файл
$f = fopen("unitednations.txt", "r");
// Закрыть текстовый файл  fclose($f);
?>
```

С помощью функции fgets можно читать строку из текстового файла. Этот метод читает до первого символа переноса строки (но не включая символ переноса строки).

```
<html>
<head>
<title>Чтение из текстовых файлов</title>
</head>
<body>
<?php
$f = fopen("unitednations.txt", "r");
// Читать строку их текстового файла и записать содержимое клиенту
echo fgets($f);   fclose($f);
?>
</body>
</html>
```


Организуем чтение всех строк текстового файла

```
<html>
<head>
<title>Чтение из текстовых файлов</title>
</head>
<body>
<?php
$f = fopen("unitednations.txt", "r");
// Читать построчно до конца файла  while(!feof($f)) {   echo fgets($f).
"<br />";
        }
        fclose($f);
        ?>
        </body>
        </html>
```

В этом коде мы выполняем цикл по всем строкам и используем функцию feof (for end-of-file/до конца файла) для проверки достижения конца файла. Если конец не достигнуть, строка записывается. Вместо цикличного прохода по всем строкам мы можем получить тот же результат функцией fread. При работе с очень большими текстовыми файлами помните, что fread использует больше ресурсов, чем fgets. Для маленьких файлов разница в работе несущественна.

Итак, текстовые файлы могут отлично подойти для хранения данных. Это показано на следующем примере, где создаётся простая директория ссылок из содержимого файла unitednations.txt.

В файле систематизировано запишем: название программы, запятая, домен. Как вы, вероятно, могли предположить, в файле с разделением запятыми можно записать куда больше информации. Для получения информации из каждой строки используем массив.

```
<html>
<head>
<title>Чтение из текстовых файлов</title>
</head>
<body>
<?php
$f = fopen("unitednations.txt", "r");
// Читать построчно до конца файла  while (!feof($f)) {
// Создать массив с запятой-разделителем    $arrM = explode(",",fgets($f));
// Записать ссылки (получить данные из массива)  echo "<li><a
href='http://". $arrM[1]. "'>". $arrM[0]. "</a></li>";
```

```
</body>
</html>
```

Задание

Изучить теоретическое обоснование и приведенные в нем примеры.

Содержание отчета и его форма

Отчет по лабораторной работе должен состоять из:

1) названия лабораторной работы;
2) описание использования текстовых файлов для хранения данных.

.

Вопросы для защиты работы:

1. Как использовать текстовые файлы для хранения данных?
2. Каким образом осуществляется работа с массивами?

Лабораторная работа №11. PHP и поля HTML-форм: текстовые поля, текстовая область, флажки, переключатели, списки

Цель работы: научиться использовать элементы HTML-форм совместно с языком PHP.

Теоретическое обоснование

Использование элемента дизайна форма при разработке Web-сайта является наиболее популярным способом организации интерактивного взаимодействия с его посетителями. С помощью языка HTML можно создавать как простые, так и сложные формы, предполагающие множественный выбор из нескольких вариантов. Формы состоят из одного или несколько полей, в которые пользователь может ввести различную информацию, либо выбрать какую-то опцию. После ввода информации, она передается на сервер, где может обрабатываться различными средствами, в том числе, с помощью языка PHP.

В ниже представленных примерах описывается обработка информации пользователя из форм. К элементам форм относятся однострочное текстовое поле, текстовая область, переключатели, флажки, списки, скрытые поля форм, поля ввода паролей и кнопки. В примерах используются две Webстраницы. Первая страница содержит форму для ввода данных пользователя и имеет расширение .html. Вторая страница обрабатывает введенную информацию средствами языка PHP и имеет расширение .php.

1 Текстовые поля

Текстовые поля являются одними из наиболее известных элементов управления. С их помощью пользователь Web-страницы может ввести любую информацию. Текстовые поля ввода создаются с помощью тега <INPUT>, в котором атрибуту type присваивается значение “text”.

Синтаксис:

<INPUT type="text" name="TextBox" size="n" maxlength="m"> где type - тип поля, name - имя поля как элемента формы, size - размер видимой части текстового поля на экране, maxlength - размер поля.

Ниже представлен пример обработки информации из текстового поля. В данном примере присутствует HTML-форма для ввода имени любимого автора. Программа считывает введенную информацию и осуществляет вывод ее на экран.

HTML-код формы для ввода информации представлен ниже.

```
<HTML>
<HEAD>
<TITLE>Пример использования текстового поля</TITLE>
</HEAD>
<BODY>
<FORM method= "GET" action="text.php">
КтоВашлюбимыйавтор?
<INPUT name="Author" type="text">
<BR><BR>
<INPUT type="submit" value= "Отправить">
</FORM>
</BODY>
</HTML>
```

Реализация данного кода представлена на рисунке 11.1

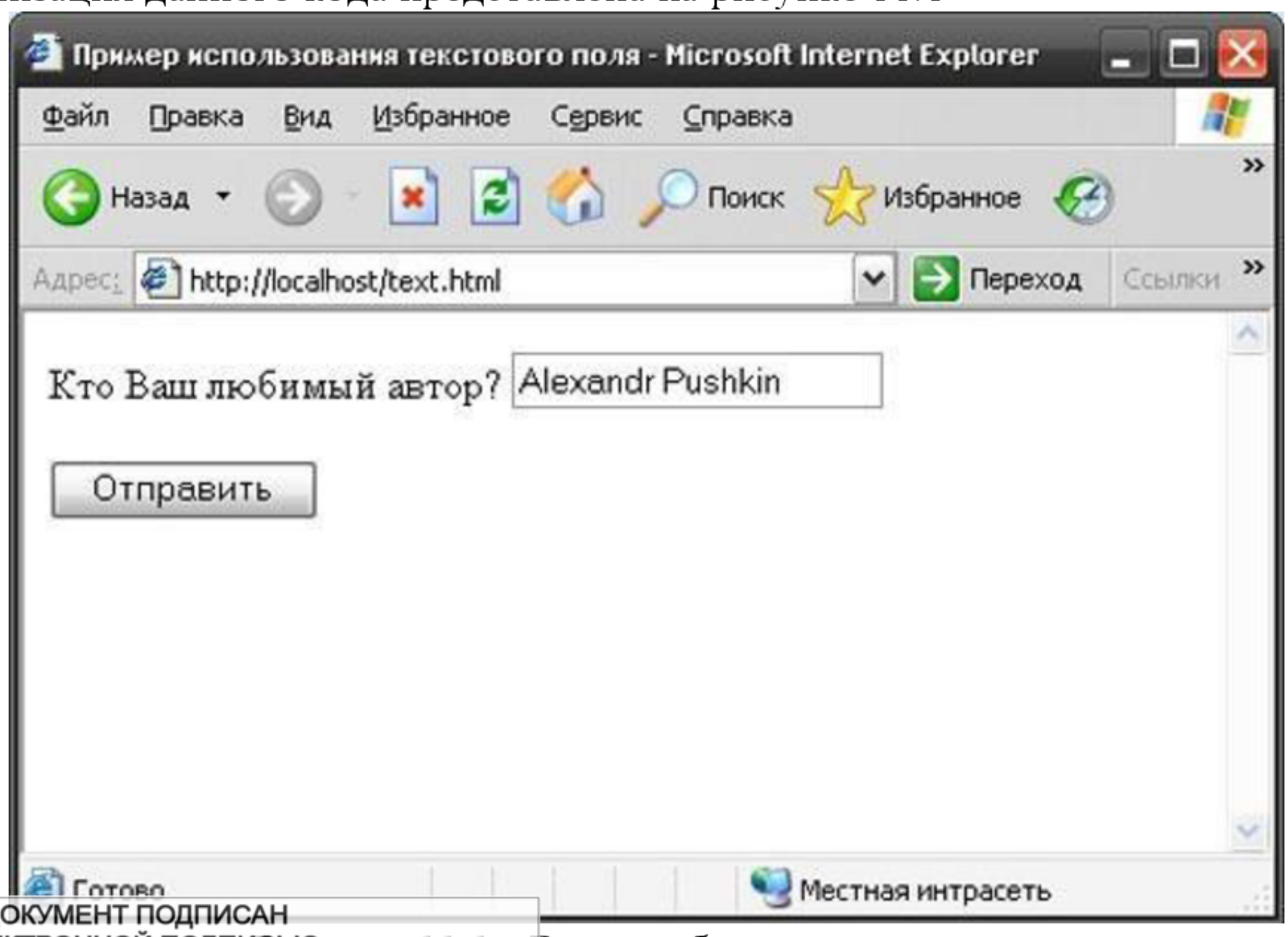

Рисунок 11.1 - Ввод любимого автора

Таким образом, как видно на рисунке 11.1, в данном примере, при загрузке Web-страницы, на экране появляется однострочное текстовое поле, в которое пользователю необходимо ввести информацию и нажать кнопку “Отправить”. После этого подключится обработчик, указанный в атрибуте “action” тега “form”. В данном примере это файл text.php.

Код файла-обработчика представлен ниже.

```
<HTML>
<HEAD>
<TITLE>Пример использования текстового поля</TITLE>
</HEAD>
<BODY>
<B>Ваш любимый автор:<B>
<?php echo $_GET['Author'];
?>
</BODY>
</HTML>
```

Реализация данного кода представлена на рисунке 11.2.

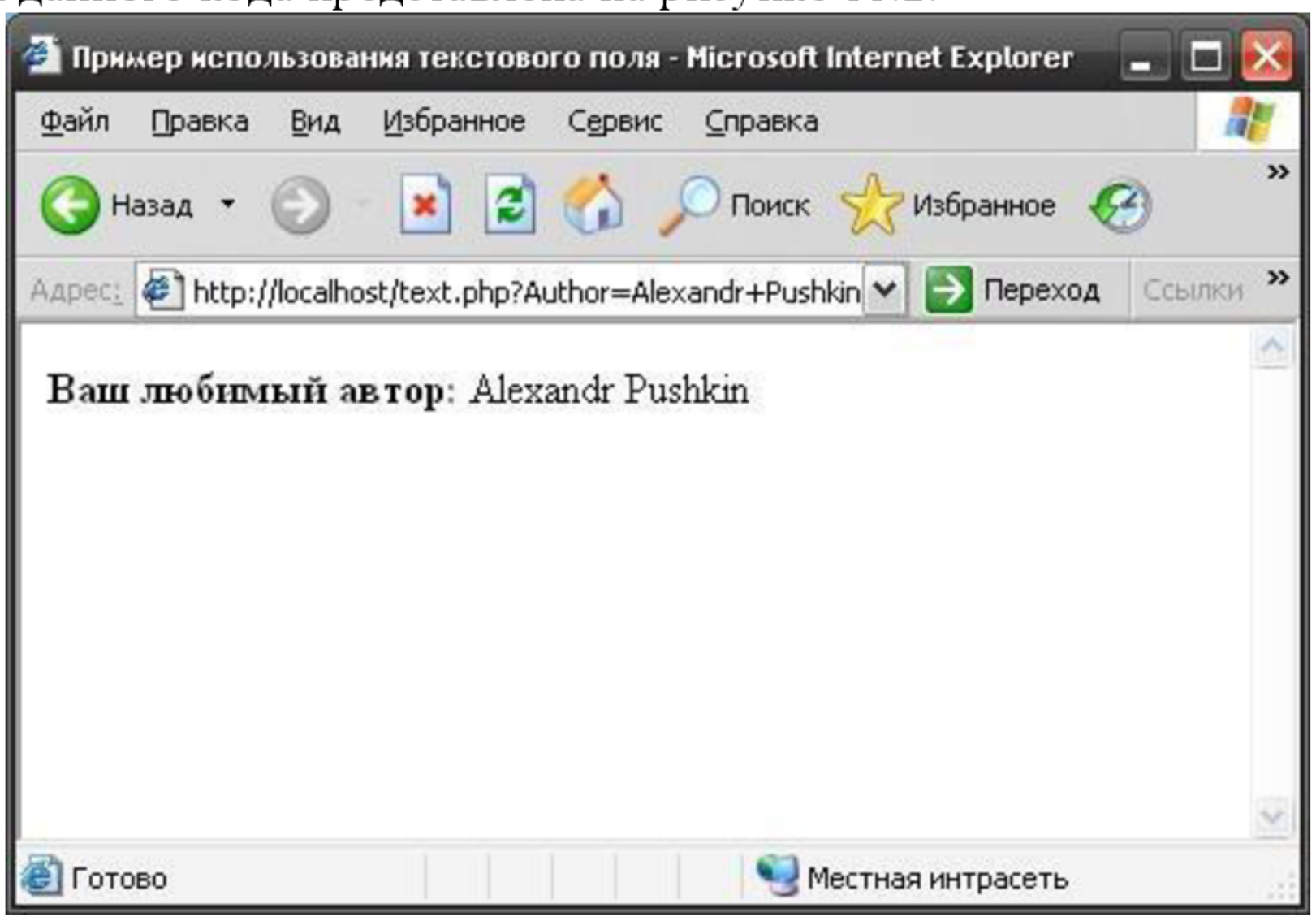

Рисунок 11.2 - Результат обработки введённого автора

Как видно на рисунке 11.2, на экран выводится информация о любимом авторе, то есть “Alexandr Pushkin”. Программа обработки введенных данных text.php, состоит из одной строки PHP-кода: “$_GET ['Author']”, которая отображает содержимое переменной “$Author”. Данная переменная не генерируется в PHP-коде, а автоматически создается, как часть массива “$_GET”, при передаче данных на сервер. При этом “GET” обозначает метод протокола HTTP, с помощью которого передаются данные.

В HTML-файле было создано текстовое поле с именем “Author”. Когда данные из формы передаются Web-серверу, создается массив “$_GET” с элементом Author. Также на рисунке 11.2, в строке окна браузера представлен

результат запроса, в котором на сервер передается пара “переменная”=“значение”. В данном примере переменной “Author” присваивается значение “Alexandr Pushkin”, то есть на сервер передается пара “Author=Alexandr+Pushkin” и эта строка видна пользователю Webстраницы. Эта строка запроса вверху страницы выглядит следующим образом: “?Author=Alexandr+Pushkin”, так как используется метод GET. Использование метода GET вынуждает браузер отправлять информацию из формы в виде строки запроса, а не скрывать ее в теле HTTP-запроса [4].

Также необходимо отметить, что имена переменных в языке PHP чувствительны к регистру символов. Так переменные “Author” и “author” являются двумя разными переменными. Если, в представленном выше примере вместо переменной “$Author” написать “$author”, то содержимое данной переменной на сервер передаваться не будет и на экране не выведется (рисунок 11.3).

Рисунок 11.3 – Неправильное присваивание имени переменой

2 Текстовая область

Текстовые области предназначены для того, чтобы принимать от пользователя предложения и даже целые строки. Для того чтобы создать текстовую область, следует использовать HTML-тег <TEXTAREA>.

Синтаксис:

<TEXTAREA name="textarea" rows="n" cols="m"> где name - имя поля как элемента формы,

rows - высота поля, cols - ширина поля.

Ниже представлен пример использования текстовой области для ввода информации о любимых Web-сайтах пользователя.

HTML-код формы для ввода информации представлен ниже.

<HTML>
<HEAD>

```
<TITLE>
Пример использования текстовой области
</TITLE>
</HEAD>
<BODY>
<FORM method="POST" action="textarea.php">
ПеречислитеВашилюбимые Web-сайты
<TEXTAREA name="Websites" cols="50" rows="5"> http:// http:// http://
http://
</TEXTAREA>
<BR>
<BR>
<BR>
<INPUT type="submit" value="Отправить">
</FORM>
</BODY>
</HTML>
```

Результат работы данного кода представлен на рисунке 11.4.

Рисунок 11.4 – Ввод информации в текстовую область

Таким образом, как видно на рисунке 11.4, в данном примере, при загрузке Web-страницы на экране появляется текстовая область, в которую пользователю необходимо ввести информацию и нажать кнопку “Отправить”. После этого запускается обработчик, указанный в атрибуте “action” тега “form”. В данном примере это файл textarea.php.

Код файла обработчика представлен ниже.

```
<HEAD>
<TITLE>
Пример использования текстовой области
</TITLE>
</HEAD>
<BODY>
Вашилюбимые Web-сайты:
<?php echo $_POST['Websites']; ?>
</BODY>
</HTML>
```

Результат работы данного кода представлен на рисунке 11.5.

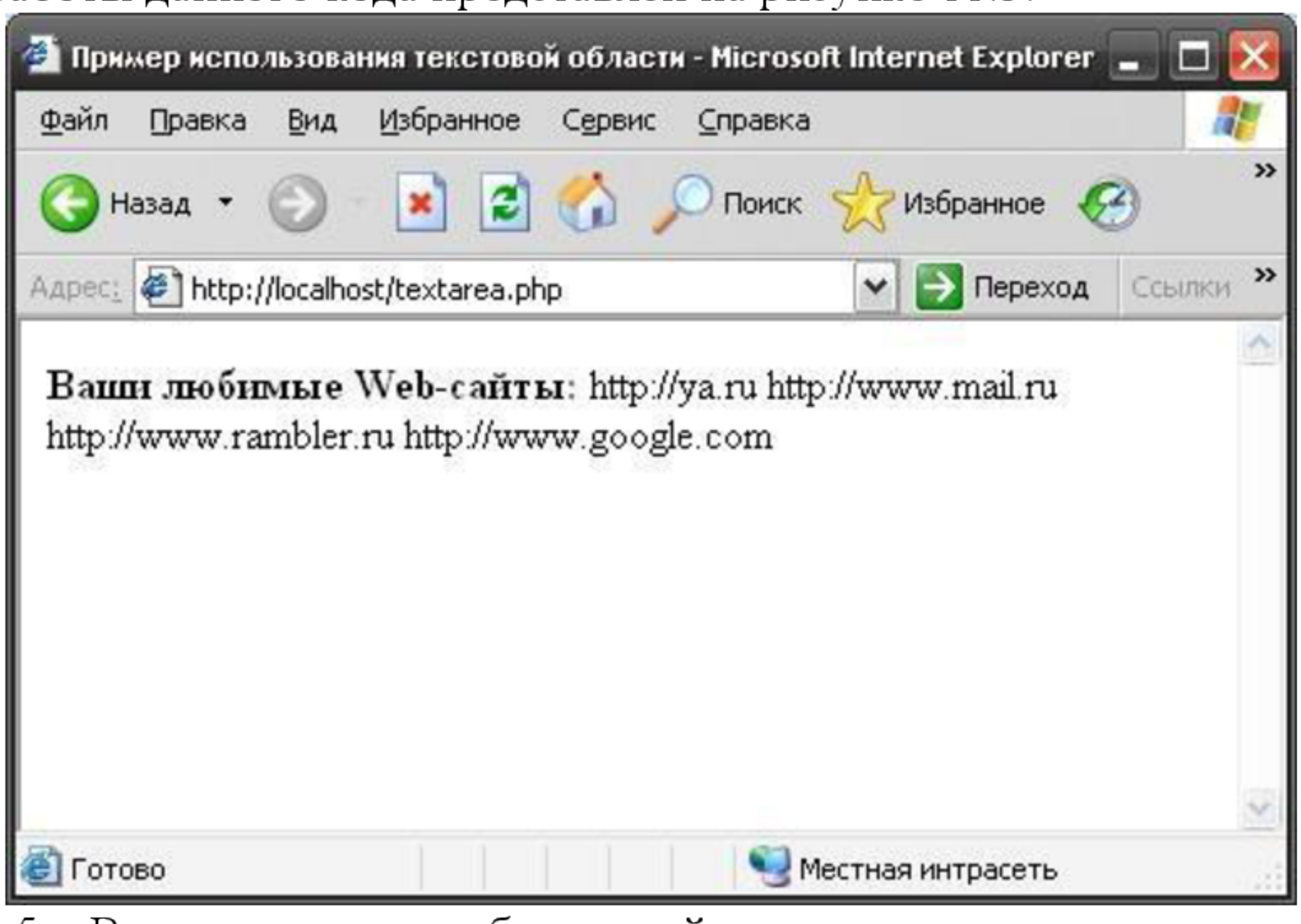

Рисунок 11.5 – Вывод на экран любимых сайтов

Как видно на рисунке 11.5, на сервер передается информация о любимых Web-сайтах пользователя. В данном случае строка запроса не присоединяется, так как в программе был использован метод POST. Использование этого метода необходимо для того, чтобы скрыть информацию из формы при передаче на сервер.

3 Флажки

Флажки являются еще одним элементом управления формами. Они применимы в ситуациях, когда пользователю необходимо ответить на вопрос, требующий строгого однозначного ответа “да” или “нет”, путем установки или снятия “галочки”. Флажки создаются с помощью тега <INPUT>, в котором атрибуту “type” присваивается значение “checkbox”.

Синтаксис:


```
<INPUT name="checkbox" type="checkbox" checked>
```

где type – тип поля, name - имя поля как элемента формы, checked - атрибут, который используется для того, чтобы отметить флажок, как выделенный.

Ниже приведен пример работы с флажками для проверки, является ли пользователь студентом Оренбургского государственного университета.

HTML-код формы для ввода информации представлен ниже.

```
<HTML>
<HEAD>
<TITLE>Пример использования флажка</TITLE>
</HEAD>
<BODY>
<FORM method="POST" action="checkbox.php">
ВыстудентОГУ?
<INPUT name="Choice" type="checkbox">
<BR>
<BR>
<INPUT type="submit" value="Отправить">
</FORM>
</BODY>
</HTML>
```

Реализация данного кода представлена на рисунке 11.6

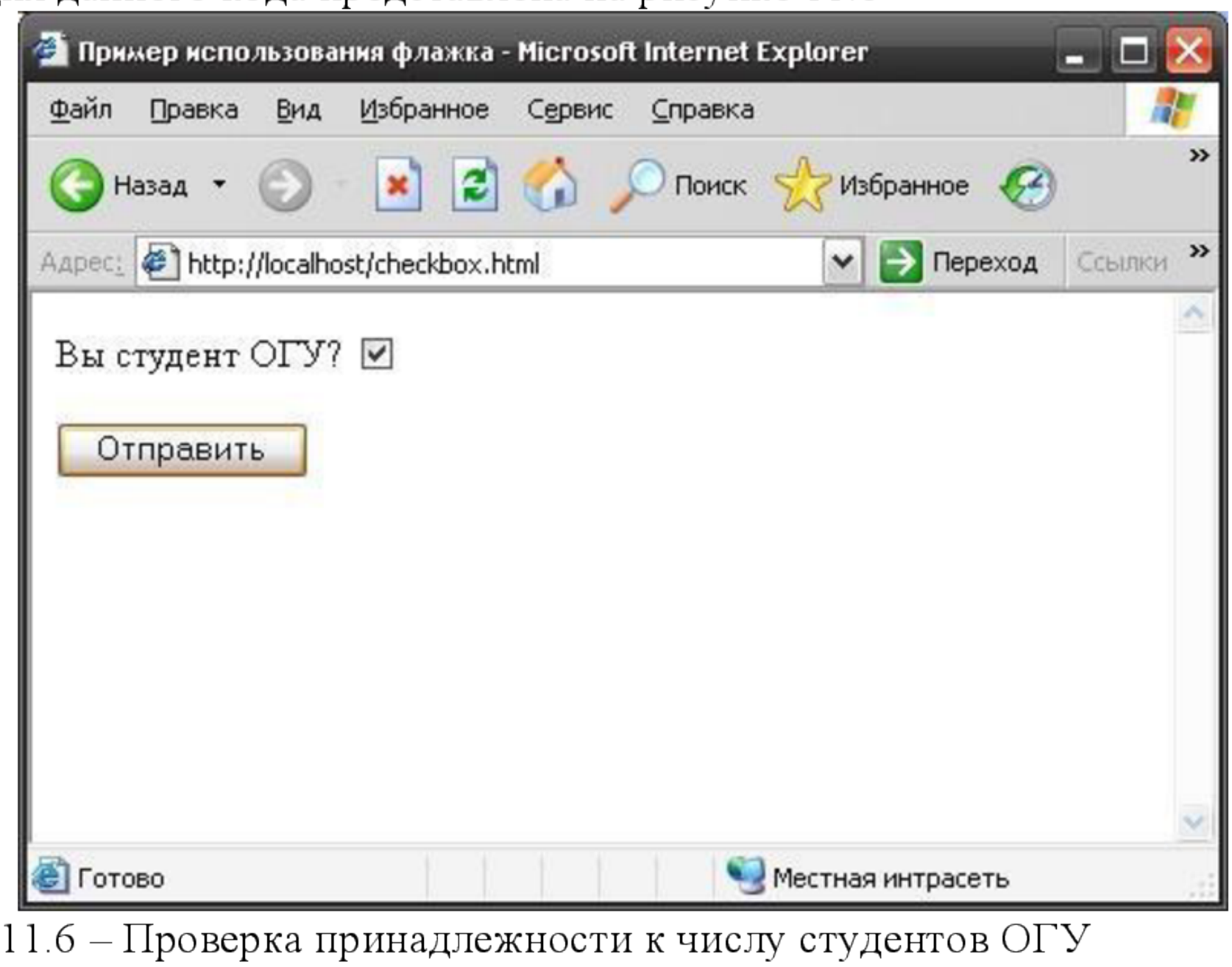

Рисунок 11.6 – Проверка принадлежности к числу студентов ОГУ

Таким образом, как видно на рисунке 11.6, в данном примере, при загрузке Web-страницы на экране появляется флажок, который пользователю необходимо отметить и нажать кнопку “Отправить”. Поле подключится обработчик, указанный в атрибуте “action” тега form. В данном примере это файл checkbox.php. Код файла обработчика представлен ниже.

<HTML>

```
<HEAD>
<TITLE> Пример использования флажка</TITLE>
</HEAD>
<BODY>
<?php echo $_POST['Choice'] ;
?>
</BODY>
</HTML>
```

Реализация данного кода представлена на рисунке 11.7.

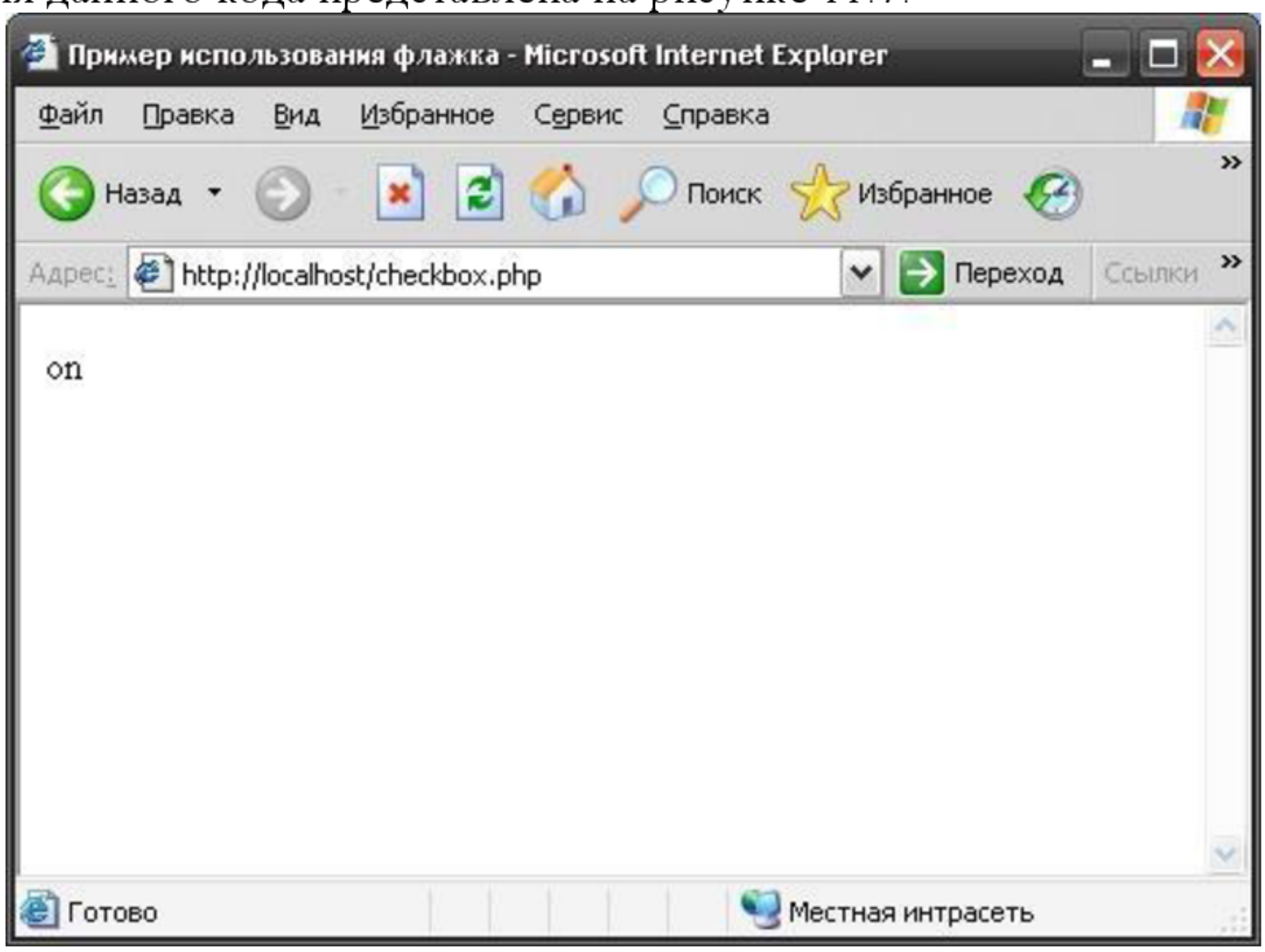

Рисунок 11.7 – Результат проверки

Как видно на рисунке 11.7, из формы на сервер передается не содержимое переменной, а лишь информация о том, был ли выбран данный флажок или нет. Если флажок был установлен пользователем, то значением данной переменной будет “on”. В противном случае, переменная вообще не будет содержать данных. Если же потребуется передать содержимое переменной на сервер, в исходном HTML-файле, в теге <INPUT>, в атрибуте “value” следует указать ее значение [4].

При использовании нескольких флажков, каждый из них является отдельным элементом. Можно выбрать один или несколько флажков, или вообще не выбрать ни одного. При этом если пользователем будут выбраны все флажки, на экран выведется содержимое всех выбранных флажков. Для этого у каждого флажка устанавливается свое значение.

Ниже приведен пример работы с несколькими флажками для выбора языков программирования, которыми владеет пользователь.

HTML код формы для ввода информации представлен ниже.

<HTML>

```
<HEAD>
<TITLE>Пример использования нескольких флажков</TITLE>
</HEAD>
<BODY>
<P>Выберите язык, на котором Вы программируете:<BR><BR>
<FORM method="POST" action="checkboxes.php">
<INPUT name="Choice1" type="checkbox" value="Delphi">
Вы программируете на Delphi?
<br>
<INPUT name="Choice2" type="checkbox" value="C++/C#">
Вы программируете на C++/C#?
<br>
<INPUT name="Choice3" type="checkbox" value="Assembler">
Вы программируете на Assembler?
<BR>
<BR>
<INPUT type="submit" value="Отправить">
</FORM>
</BODY>
</HTML>
```

Результат работы данного кода представлен на рисунке 11.8.

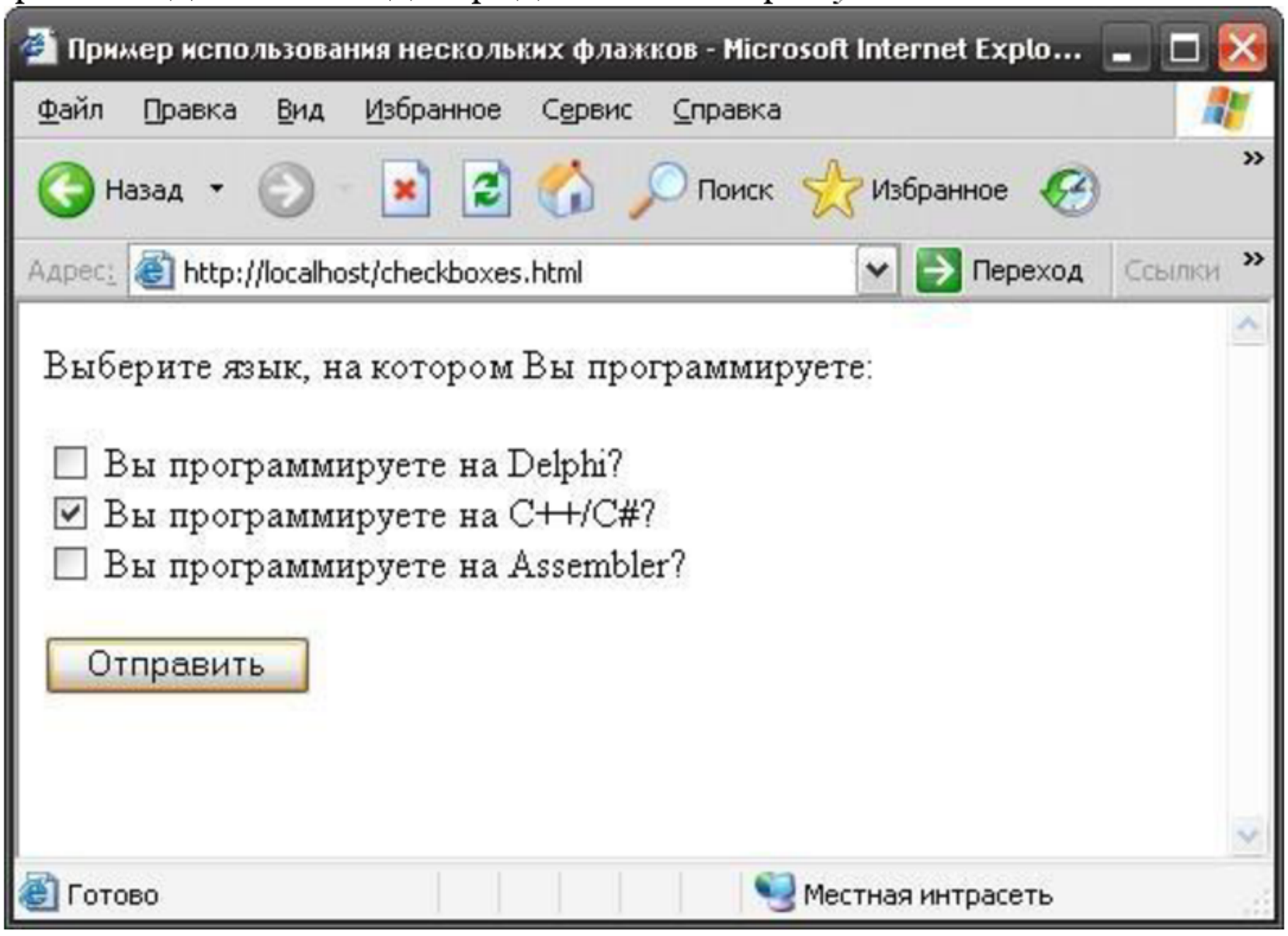

Рисунок 11.8 – Выбор языков программирования

Таким образом, как видно на рисунке 11.8, в данном примере, при загрузке web-страницы появляются флажки, которые пользователю необходимо отметить и нажать кнопку

“Отправить”. После этого подключится обработчик, указанный в атрибуте “action” тега “form”. В данном примере это файл checkboxes.php.

Код файла-обработчика представлен ниже.

```
<HTML>
<HEAD>
<TITLE>
Пример использования нескольких флажков
</TITLE>
</HEAD>
<BODY>
<P>Вывыбралиответ:</P>
<?php echo "$_POST[Choice1]<br>"; echo "$_POST[Choice2]<br>"; echo "$_POST[Choice3]<br>";
?>
</BODY>
</HTML>
```

Результат работы данного примера представлен на рисунке 11.9.

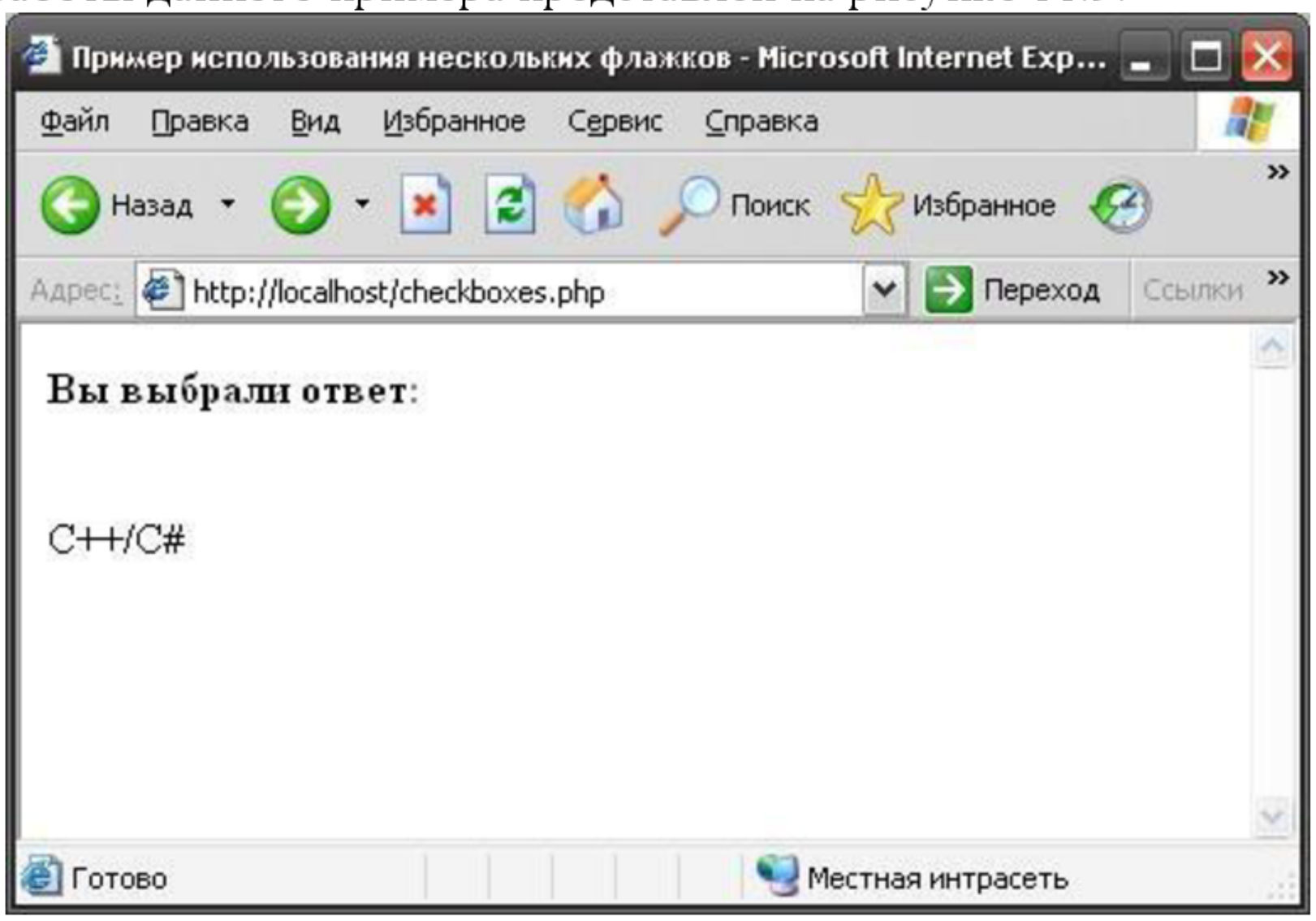

Рисунок 11.9 – Результат выбора языков

Как видно на рисунке 11.9, пользователем был выбран только один флажок и на экран выведено содержимое переменной, равное “C++/C#”. Содержимое остальных переменных выведено не было, так как пользователь не установил другие флажки. Таким образом, в соответствующие PHPпеременные данные передаваться не будут.

4 Переключатели

Переключатели в HTML-формах используются в тех случаях, когда имеется набор возможных ответов, но выбрать можно только один из них. В