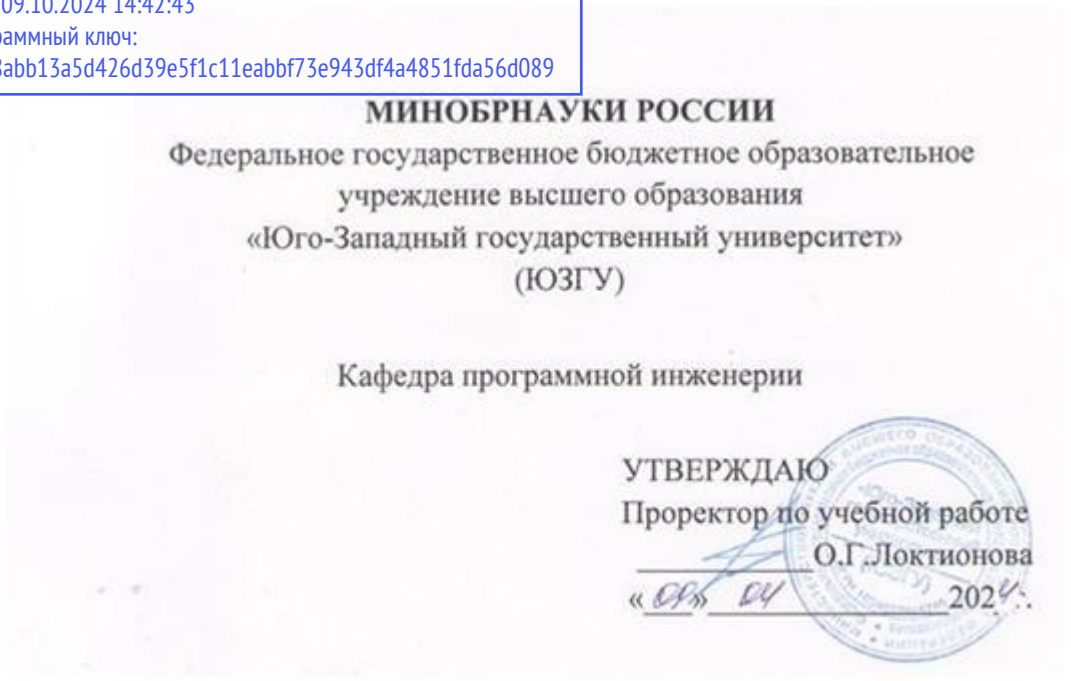


МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«09» 04 2024 г.

Разработка Интернет-приложений

Методические указания по выполнению практических работ для направления 09.04.03 Программная инженерия направленность «Предпринимательство, инновации и технологии будущего в программной инженерии», разработанные по модели элитного обучения

Курск 2024

УДК 681.3(075)
Составитель: Л.А. Лисицин

Рецензент Кандидат технических наук, доцент
Халин Ю.А.

Методические указания по выполнению практических работ для студентов 09.04.03 Программная инженерия направленность «Предпринимательство, инновации и технологии будущего в программной инженерии», разработанные по модели элитного обучения / Юго-Зап. гос. ун-т; сост.: Л.А. Лисицин. Курск, 2024. 68 с.: ил. 20. табл. 1. Библиогр. с. 68.

Содержат сведения по технологиям разработки сайтов. Материал ориентирован на практическую работу студентов в компьютерной среде.

Отражен порядок выполнения практических работ и правила оформления отчетов.

Методические указания разработаны по модели элитного обучения для специальности «Разработка Интернет-приложений».

Текст печатается в авторской редакции

Подписано в печать _____ Формат 60х84 1/16.
Усл.печ. л. __. Уч.-изд. л. ___. Тираж 50 экз. Заказ . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Оглавление

Лабораторная работа 1. Введение в web-серверы. Установка и настройка web-серверов. Сервер apache.

Цель работы – научиться установке и настройке web-серверов: nginx, apache.

Web-приложения можно разделить на несколько типов, в зависимости от разных сочетаний их основных составляющих:

1. Backend (бэкенд или серверная часть приложения). Работает на удаленном компьютере, который может находиться, где угодно. Она может быть написана на разных языках программирования: PHP, Python, Ruby, C# и других. Если создавать приложение, используя только серверную часть, то в результате любых переходов между разделами, отправок форм, обновления данных, сервером будет генерироваться новый HTML-файл и страница в браузере будет перезагружаться.
2. Frontend (фронтенд или клиентская часть приложения). Выполняется в браузере пользователя. Эта часть написана на языке программирования JavaScript. Приложение может состоять только из клиентской части, если не требуется хранить данные пользователя дольше одной сессии. Это могут быть, например, фоторедакторы или простые игрушки.
3. Single page application (SPA или одностраничное приложение). Более интересный вариант, когда используются и бэкенд и фронтенд. С помощью их взаимодействия можно создать приложе ние, которое будет работать совсем без перезагрузок страницы в браузере. Или в упрощенном варианте, когда переходы между разделами вызывают перезагрузки, но любые действия в разделе обходятся без них.

Большинство web-приложений имеют архитектуру клиент- сервер. При этом в качестве клиентской программы используется web-браузер, а обмен с серверной частью происходит с использованием протокола HTTP.

Как показано на рисунке 1, web-сервер (блок, реализующий обмен с клиентом по протоколу HTTP) транслирует HTTP-запрос в некую среду, которая программным путем формирует HTML-ответ.

HTML-страницы, сформированные программным путем, называются динамическими, потому что их содержание меняется во времени и может зависеть как от параметров HTTP-запроса, так и от предыдущих шагов клиента в рамках данной сессии. Среда выполнения может быть различной, более подробно вопрос о способах формирования динамических страниц будет рассмотрен ниже.

Основные причины широкого распространения именно web-приложений обусловлены их достоинствами, а именно:

- доступностью, пользователю не нужно что-либо ставить на компьютер в качестве клиентского программного обеспечения, достаточно просто набрать нужный адрес в браузере;
- отсутствием зависимости от версии приложения, если разработчик изменил код приложения, клиентам не нужно сообщать, доступна ли новая версия программы. Следующий после изменения кода запрос даст пользователю уже новое содержание.

Для разработки и тестирования сайтов или web-приложений обычно используют локальный web-сервер. Начинающие разработчики часто используют готовую сборку web-сервера. *Сборка web- сервера* - это комплект приложений, необходимых для функционирования сайтов. В сборку web-сервера обычно входит как минимум 3 компонента: это сам HTTP-сервер, средство разработки сайтов и система управления базами данных. Для таких сборок используется бесплатное программное обеспечение, поэтому самыми распространенными компонентами являются web-сервер Apache, СУБД MySQL, языки программирования PHP и Perl. У англоязычных разработчиков популярна сборка XAMPP, в то время как у русскогово- рящих пользователей очень распространен джентльменский набор Denwer. Аббревиатура XAMPP означает X (любая из ОС: Linux, Windows, Mac OS, Solaris), Apache, MySQL, PHP, Perl. XAMPP кажется нам более удобным в использовании. Разработчики XAMPP следят за тем, чтобы при появлении новой версии той или иной библиотеки комплект web-сервера обновлялся.

XAMP - акроним, обозначающий набор (комплекс) серверного программного обеспечения, широко используемый во Всемирной паутине. XAMP назван по первым буквам входящих в его состав компонентов:

X - операционная система (Linux, Windows или Mac);

A - web-сервер Apache;

M - СУБД MySQL/MariaDB;

P - язык программирования, используемый для создания web-приложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).

Apache - это web-сервер с открытым исходным кодом, изначально разработанный для систем UNIX. В настоящее время Apache поддерживается на большинстве платформ, включая UNIX, Linux, Windows и Mac, и является одним из наиболее часто используемых серверных приложений. Впервые разработанный в 1995 г. Apache следует подходу с открытым исходным кодом, аналогичным Linux, позволяя пользователям расширять программное обеспечение и вносить вклад в сообщество пользователей. Группа пользователей вокруг Apache разработала The Apache Foundation, которая поддерживает библиотеку решений для web-сервисовНа web-сервере Apache служит компонентом

HTTP, который компилирует результаты из языков сценариев, баз данных и файлов HTML для создания содержимого, которое отправляется пользователю. Apache (или любой web-сервис) будет отслеживать, какие файлы на сервере делать и не принадлежать к сайту, и также контролирует, какие опции доступны для конечного пользователя через свои файлы конфигурации.

Apache и другие HTTP-серверы позволяют нам делиться нашими web-страницами, сценариями и файлами с нашими конечными пользователями. Любой вывод из нашей базы данных и языков сценариев преобразуется в вывод HTML, который браузер клиента отображает в качестве нашей web-страницы. Хотя мы можем просматривать файлы HTML и JavaScript на компьютере, который не является web-сервером, нам нужен http-сервер для просмотра их в качестве места назначения в сети.

Учитывая популярность комбинации XAMP, легко задаться вопросом, почему она не стала просто системой. Однако потребности и предпочтения могут измениться. Возможно, вы находитесь в среде Windows и чувствуете себя более комфортно с операционной системой Windows. Возможно, ваши данные уже доступны в виде простого файла или в формате XML и вам нужна база данных, которая может использовать файлы XML, такие как MongoDB. Или вы можете предпочесть подход и пакеты, доступные в Python, а не те, которые есть в PHP. Каждая система имеет свои сильные и слабые стороны, и ее следует выбирать в зависимости от потребностей проекта.

Хотя это и не включено в аббревиатуры XAMP, еще одним важным элементом, который следует отметить, является наличие сервера протокола передачи файлов (FTP). Поскольку вы, как правило, захотите выполнить действия по разработке на частном сервере перед редактированием вашего живого сервера, вам понадобится механизм, который позволяет перемещать файлы между ними.

FTP предназначен для перемещения файлов между системами, что позволяет синхронизировать элементы, когда вы будете готовы. В дополнение к FTP-серверу вам, скорее всего, понадобится клиентское приложение FTP для компьютеров, на которых находятся файлы, которые вы хотите переместить. Клиент позволяет просматривать файлы в обоих местах и взаимодействовать с ними, чтобы определить, какой файл перемещен и на какой компьютер. Существует несколько бесплатных программ для передачи файлов, некоторые из которых могут быть интегрированы в такие браузеры, как Chrome, с помощью расширений браузера.

Итак, что же такое Apache? Это полнофункциональный, расширяемый веб-сервер, полностью поддерживающий протокол HTTP/1.1 и распространяющийся с открытым исходным кодом. Сервер может работать практически на всемх распространенных платформах. Apache настраивается с помощью текстовых конфигурационных файлов. Основные параметры уже настроены «по умолчанию» и будут работать в большинстве случаев. Самая простая функция, которую может выполнять Apache – стоять на сервере и обслуживать обычный HTML-сайт. При получении запроса на определенную страницу сервер отправляет в ее ответ браузеру. Набираете адрес, открывается страница — все просто. Если на одном сервере с установленной операционной системой семейства Unix и сервером Apache заведено несколько пользователей, то каждому из них можно создать отдельную директорию. Точнее, она будет создаваться автоматически вместе с псевдонимом. Это делается с помощью модуля mod_userdir и директивы UserDir. Полноценный же хостинг обычно предусматривает создание отдельного виртуального сервера для каждого пользователя.

Сервер Apache был одним из первых серверов, которые начали поддерживать виртуальные сервера (хосты). Эта возможность позволяет размещать на одном физическом сервере несколько полноценных сайтов. У каждого из них может быть свой домен, администратор, IP-адрес и так далее. Если вам нужно разместить на вашем сервере домены domain.ru и domain.com, то для начала надо сделать так, чтобы в системе DNS им был сопоставлен ваш IP-адрес. После этого в конфигурационном файле Apache создаете две директивы <VirtualHost>, где описываете каждый виртуальный хост. Таким образом, сервер будет знать, на какую папку «отправлять» пришедший запрос.

В данный момент большинство интернет-страниц являются динамическими. Это значит, что их внешний вид и наполнение формируется с помощью программного скрипта, написанного на одном из «языков» (их нельзя в полной мере назвать языками, определение достаточно условно). В данный момент наиболее сильно распространены технологии CGI и PHP. Разумеется, в Apache существует поддержка и того, и другого, плюс возможность подключать другие языки.

Модуль mod_cgi позволяет вам размещать на сервере CGI-скрипты. Вообще, это всегонавсего исполняемые файлы, написанные на одном из допустимых языков программирования. Они могут содержаться как в откомпилированном виде (например, так делают, если пишут CGI на языке C++), так и в виде исходного текста (если на сервере установлен Perl, то программист может помещать и такие файлы. Иногда они имеют расширение .pl).

Что касается PHP, то возможность интеграции его в Apache предусмотрена разработчиками самого PHP. Apache же выполняет только функции посредника между скриптом и компилятором. Существует два

способа интеграции PHP в Apache. Первый – установка специального модуля, расширяющего возможности сервера, и тогда он сам становится способным «компилировать» скрипты. И второй – установка в конфигурационных файлах связей между php-файлами и самим компилятором (он находится на диске в виде файлов .cgi или .exe).

На основе сервера Apache можно создавать не только простые любительские сайты, но и ресурсы, требующие серьезной криптографической защиты передаваемых данных. Специально для этого был разработан протокол SSL/TLS, а его поддержка была встроена в Apache 2.0. С помощью специального модуля можно осуществлять аутентификацию на основе именных сертификатов, что позволяет практически наверняка гарантировать подлинность пользователя.

Ну и, разумеется, сервер Apache может вести протокол всех действий, совершаемых с ним. Причем администратор может сам выбрать степень подробности протокола. Протоколы ведутся отдельно для ошибок, для успешных операций и для каждого виртуального хоста. Словом, полный набор для тщательного анализа появляющихся ошибок.

Итак, подведем итог. Что нам требуется для функционирования сайта? 1. Веб-сервер(Apache) для обработки http запросов 2. PHP – скриптовый язык программирования 3. MySQL – СУБД

Рассмотрим два варианта: установка компонентов по отдельности и установка комплексом. Установка и конфигурация вебсервера Apache

Классический:

2. Установка и конфигурация скриптового языка программирования PHP 3. Установка и настройка СУБД

4. Отладка совместной работы Apache+MySQL+PHP

Использование комплекса

1. Запуск инсталлятора, содержащего связку Apache+MySQL+PHP.

Таким образом, использование сборок позволяет существенно сократить время и требует меньших знаний о каждом составляющем компоненте. Готовые решения уже можно найти в сети Интернет. Одно из таких решений – Денвер (denwer.ru) Таким образом, характерными особенностями «хороших» комплексов по созданию локального веб-сервера являются: 1. Русификация 2. Виртуальный диск 3. Минимальные знания и требования для установки 4. Автоматическое добавление новых сайтов 5. Графическая оболочка для настроек 6. Небольшой вес дистрибутива 7. Самостоятельное обновление 8. Тестовые и обучающие программы

Практическая часть

1. Установка Денвера Для установки веб-сервера требуется полноценный доступ к изменению файла hosts, расположенному в папке: **C:\WINDOWS\system32\drivers\etc** (разумеется, диск и название папки WINDOWS могут быть другими)

1. Запускаем инсталляционный файл Денвера.

2. Указываем, в какой каталог требуется установить комплекс (по умолчанию, все файлы загрузятся в папку **C:\WebServers**). Все файлы Денвера будут находиться только в этой папке (за исключением трех ярлыков на рабочем столе)

3. Следующий шаг заключается в создании виртуального диска. Придумываем ему имя (например Server) и оставляем умолчание **Z:/**

4. После копирования файлов дистрибутива будет задан вопрос, каким образом запускать и останавливать комплекс. Пользователю предлагается два варианта:

а) создание виртуального диска и запуск Денвера при загрузке компьютера

б) создание виртуального диска и запуск Денвера вручную при щелчке ярлыка запуска (Start Servers) на рабочем столе.

5. Установка комплекса завершена.

Обращаем внимание, что если установка производится под логином «Администратор» (записанном кириллицей), ярлыки на Рабочем столе не создаются. В этом случае они могут быть созданы вручную, используя папку C:\WebServers\etc. При возникновении проблем с установкой системы Денвер, рекомендуется обратиться за дополнительной информацией на сайт разработчика: **http://www.denwer.ru/base.html**

2. Работа с Денвером. Создание нового домена.

Для тестирования работы установленного комплекса используем готовый html-сайт (или отдельную страницу index.html).

1. Запускаем Денвер (если он не запустился сам при загрузке компьютера). Примечание: комплексу требуется полноценный доступ к изменению файла hosts, расположенному в папке: **C:\WINDOWS\system32\drivers\etc.**

2. Для проверки работоспособности запущенного комплекса откройте любой из имеющихся в системе Интернет-браузеров (IE, Opera, Firefox и др.) и в адресной строке наберите **http://localhost**. Если установка была завершена успешно, в окне браузера появится стартовая страница Денвера, показанная ниже.

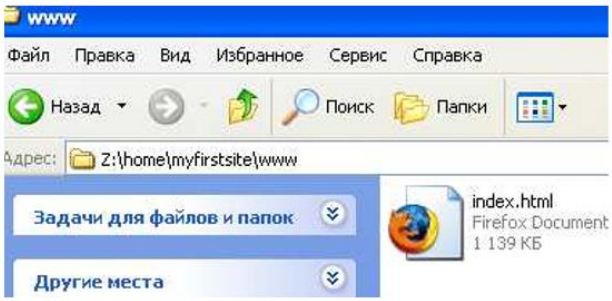


Рис.1 - стартовая страница Денвера

3. На диске Z: (виртуальный диск, создаваемый Денвером при запуске) найдите папку Z:\home, в которой создайте новую директорию (например, myfirstsite). После этого зайдите в эту папку и создайте в ней еще одну папку www. Поместите готовый сайт (или страницу с именем index.html), разработанные ранее, в этот каталог (см. рис.1).

4. После того, как все файлы скопированы, перезапустите веб-сервер, кликнув по ярлыку «Restart Servers».

5. В адресной строчке веб-браузера наберите в http://myfirstsite. Если все этапы установки комплекса и локального размещения сайта выполнены правильно, в окне браузера отобразится содержимое файла index.html.

Работа с Денвером. Создание поддоменов.

1. В директории домена создайте новую папку (subdomain). Разместите в ней файл index.html.

2. Перезапустите Денвер. Проверьте работу поддомена, набрав в адресной строчке браузера адрес вида: **http://subdoman.domain**

Контрольные вопросы

1. Дайте определение веб-сервера.
2. Какие дополнительные функции может обеспечивать веб-сервер?
3. Опишите клиент-серверную архитектуру.
4. По каким параметрам оценивается веб-сервер?
5. Перечислите наиболее распространенные веб-сервера. Охарактеризуйте кратко каждый.
6. Опишите основные функциональные возможности IIS.
7. Опишите основные функциональные возможности Apache.
8. В каких случаях организация локального веб-сервера может быть необходимой?
9. Дайте сравнительную характеристику между классическим способом установки веб сервера и использованием комплекса.
10. Опишите функциональные возможности Денвера.

```
    <img class="thumbnail" src="yourPictureHere.jpg" >
</div>
```

Содержимое до и после плавающего элемента будет пытаться разместиться вокруг него с обтеканием. Если мы не хотим, чтобы это произошло, мы можем добавить правило к стилю этого элемента, чтобы очистить плавающий эффект. Для этого надо добавить

```
    clear: left; // для левой стороны
clear: right // для правой стороны
clear: both // для обеих сторон
```

в зависимости от того, с какими сторонами мы имеем дело.

Лабораторная работа 2. Введение в язык программирования PHP. Основные функции.

Цель работы – научиться создавать программы на PHP.

Если вы хотите использовать PHP в файле, даже если он уже имел расширение имени .htm или .html, вам нужно будет установить расширение имени файла ".php".

Для создания программ на PHP нам потребуется текстовый ре-дактор. Наиболее популярным редактором на сегодняшний день является программа Notepad++.

Перейдем к каталогу C:\localhost, который создается при установке XAMPP. Здесь будут храниться все документы сайта. Создадим текстовый файл и назовем его index.html. Откроем его в текстовом редакторе и добавим в него следующий код:

```
<!DOCTYPE html>
<html>
<head>
 <Первый сайт на PHP</title>
<meta charset="utf-8">
</head>
<body>
<h2>Введите свои данные:<^2>
<form action="display.php" method="POST">
<p>Введите имя: <input type="text" name="firstname" /> </p>
<p>Введите фамилию: <input type="text" name="lastname" /> </p>
<input type="submit" value="Отправить">
</form>
</body>
</html>
```

Код HTML содержит форму с двумя текстовыми полями. При нажатии на кнопку данные этой формы отсылаются скрипту dis- play.php, который указан в атрибуте action.

Теперь создадим скрипт display.php, который будет обрабатывать данные. Добавим в папку C:\localhost новый текстовый файл. Переименуем его в display.php. По умолчанию файлы программ на php имеют расширение имени .php. Добавим в файл display.php следующий код:

```
<!DOCTYPE html>
<html>
<head>
^^Первый сайт на PHP</title>
<meta charset="utf-8">
</head>
<body>
```

```
<?php
$name = $_POST["firstname"];
$surname = $_POST["lastname"];
echo "Ваше имя: <b>".$name . " " . $surname . "</b>";
?>
</body>
</html>
```

Здесь в разметке html есть вкрапления кода PHP. Для добавления выражений PHP на страницу используются теги <? php

?>, между которыми идут инструкции на языке PHP. В коде php мы получаем данные формы и выводим их на страницу.

Каждое отдельное выражение PHP должно завершаться точкой с запятой. В данном случае у нас три выражения. Два из них получают переданные данные формы, например,

$name = $_POST["firstname"];.

$name - это переменная, которая будет хранить некоторое значение. Все переменные в PHP предваряются знаком $. И так как форма на странице index.html использует для отправки метод POST, то с помощью выражения $_POST ["firstname "] мы можем получить значение, которое было введено в текстовое поле с атрибутом name="firstname". И это значение попадает в переменную $name.

С помощью оператора echo можно вывести на страницу любое значение или текст, которые идут после оператора. В данном случае (echo "Ваше имя: ".$name . " " . $surname . "") с помощью знака точки текст в кавычках соединяется со значениями переменных $name и $surname и выводится на страницу.

Теперь обратимся к форме ввода, перейдя по адресу http://localhost:8080 (рис. 2).

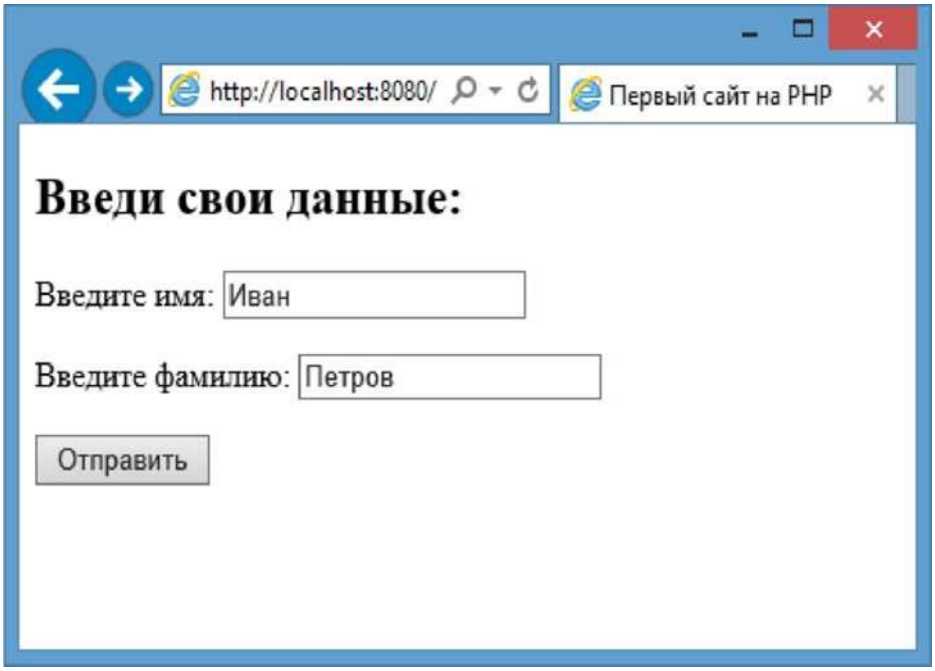


Рис. 2 форма для ввода данных.

Введем какие-нибудь данные и нажмем на кнопку *Отправить.* Скрипт display.php получит отправленные через форму данные и выведет их на страницу (рис. 4).

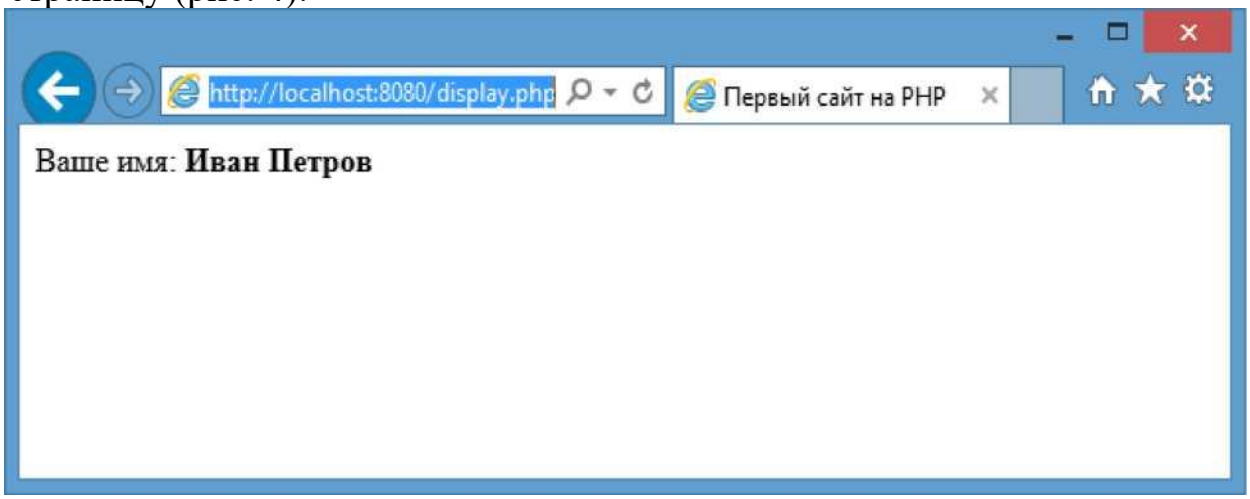


Рис. 3. Результат работы скрипта display.php

Сообщения об ошибках PHP

Понимание сообщений об ошибках до начала работы поможет вам быстро определить, где в вашем коде существуют проблемы (и действительно ли это проблемы).

Мы можем сообщить интерпретатору PHP, о каких типах ошибок мы хотим знать, прежде чем мы даже запустим скрипт. Хотя полный список поддерживаемых уровней сообщений охватывает различные проблемы, есть несколько типов сообщений (уведомления, ошибки и предупреждения), с которыми вы столкнетесь вероятнее всего.

Уведомления ***(Notices)***

Пример уведомления

Notice: Undefined index: message in /home/example.php on line 9

Технически уведомления не являются ошибками. Они будут уведомлять нас о том, без чего PHP может обойтись. Например, использование переменной на странице без ее предварительного объявления приведет к уведомлению. PHP создаст переменную, как Предупреждения (Warnings)

Пример предупреждения:

Warning: main(): Failed opening 'noFileHere.php' for inclusion on l ine 2

Это - предупреждение о попытке использовать include () для несуществующего файла 'noFileHere.php' Предупреждения не пре-пятствуют выполнению сценария, но они указывают на то, что во время выполнения сценария что-то пошло не так.

Ошибки (Errors)

Пример ошибки

PHP Fatal error: Undefined class constant

'MYSQL_ATTR_USE_BUFF-ERED_QUERY' in database.inc on line 43

Неустранимая ошибка (Fatal error) вызывает остановку выполнения приложения. Типичные причины ошибок - ошибки синтаксические, такие как пропущенные точки с запятой, определения функций или классов или другие проблемы, которые движок не знает, как решить. Если бы мы использовали require () для файла вместо include, возникла бы ошибка.

Большинство ошибок являются ошибками этапа синтаксического анализа. Как правило, это проблемы, вызванные отсутствием скобок, точек с запятой или опечатки. Компилятор сообщает, какую проблему он обнаружил и где. Имейте в виду, что нам сообщают, где была обнаружена ошибка, но не обязательно, где существует источник проблемы. Например, пропущенная точка с запятой или скобка могли встретиться за несколько строк до того, как это создало проблемы для компилятора.

Другая категория ошибок - это гораздо более неприятные логические ошибки кода. Логические ошибки обычно обнаруживаются, когда скрипт работает не так, как ожидалось. Источником могут быть, например, ошибки в том, какой код мы выполняем в разных частях оператора if / then, или математическая ошибка в функции, которая дает нам неправильное решение.

Устранение ошибок может быть чем-то вроде искусства. Логические ошибки обычно могут быть устранены путем добавления дополнительных временных выходных данных для отслеживания значения переменной или отслеживания выполнения логических операторов. Этот метод может помочь вам найти, где то, что происходит, отличается от того, что вы ожидаете. Большое значение для предотвращения многих из этих проблем будет иметь модульное тестирование функций и итеративное программирование.

Чтобы определить, какие сообщения мы не хотим видеть в выводе нашего скрипта, мы будем использовать функцию error_report- ing (). Передавая одну или несколько констант, мы контролируем то, что сообщается.

Например, может быть, нам нужна информация о предупреждениях и ошибках, но нам не нужны уведомления. Для этого мы можем вызвать команду error_reporting (E_WARNING | E_ERROR).

Символ (|) работает в этом случае как или. Если мы хотим видеть все, кроме уведомлений, мы можем использовать E_ALL с символом (л), чтобы указать исключение с error_reporting (E_ALL Л E_NOTICE). Рекомендуется устанавливать уровень отчетов об ошибках ближе к началу вашего скрипта, чтобы вы могли легко найти и изменить настройки:

```
<?php
error_reporting(E_WARNING | E_ERROR);
//This next line will trigger a notice that the variable does not exist, but we will not see it echo $test;
?>
<?php error_reporting(E_ALL);
//This time we will see the notice echo $test;
?>
Notice: Undefined variable: test on line 3
```

Основы языка PHP

Перед тем, как приступить к описанию языка PHP, отметим, что в данном пособии будут рассмотрены лишь основы, необходимые для того, чтобы начать программировать на PHP. Все остальное можно изучить самостоятельно по мере практической надобности. Здесь не будут рассматриваться все типы данных и конструкции языка программирования PHP, а также не будет приводиться подробное описание его синтаксиса. Предполагается, что у читателя уже имеется знание какого-либо из языков

Вывод на экран

PHP допускает два метода отправки вывода на экран - print и echo. Хотя они обеспечивают одинаковую функциональность, echo - это команда, а print - это выражение (оно будет вычислено и вернет значение). И print, и echo могут быть использованы в качестве конструкций (вызываемых без круглых скобок после них, как если бы они были командой) или в качестве функций (вызываемых с круглыми скобками как вызов функции). Print возвращает 1 в результате, в то время как echo ничего не возвращает.

Большой объем операторов echo будет работать быстрее, чем большой объем операторов print.

Чтобы отправить вывод на экран, мы можем начать со знаменитого примера Hello, World!

Когда функции нам что-то возвращают, мы обычно сохраняем это значение, а затем выполняем действия для него. Мы также можем отправить вывод непосредственно на экран, если мы знаем, что он отформатирован так, как мы хотим. Например, функция phpinfo () дает нам доступ к информации о сервере. По умолчанию возвращается полная web-страница со всеми подробностями о нашем сервере. Мы можем увидеть это, используя следующую запись:

<?php echo phpinfo(); ?>

Переменные

Переменные создаются и используются при запуске нашего сценария, чтобы создавать и хранить копии фрагментов информации, которые необходимы в течение короткого промежутка времени или которые, как ожидается, изменят значение при запуске сценария. Они содержат ссылку на место в памяти сервера, где хранится значение, а не на фактическое содержимое. Это сообщает парсеру, где искать нужную информацию. PHP является свободно типизированным языком, что означает, что нам не нужно сообщать компьютеру, какой тип данных мы будем хранить в переменной. В строго типизированных языках нам нужно было бы указать, будет ли переменная целочисленной, строковой, с плавающей запятой или другой

опцией, поддерживаемой используемым языком. Попытка сохранить информацию другого типа, чем объявлена в переменной, приведет к ошибке. PHP не выдаст нам ошибку или не проверит ее по умолчанию. Это избавляет от необходимости объявлять переменные, но это может привести к ошибкам в вашем коде. Те из вас, кто имеет опыт работы со строго типизированным языком, таким как C ++,

с радостью заметят, что, хотя это и не обязательно, PHP разрешает объявления, а затем выдает ошибки при их неправильном использовании.

Имена переменных в PHP должны начинаться со знака доллара ($), после которого может идти знак подчеркивания (_) или буква (от a до z, как в верхнем, так и в нижнем регистре). Имена переменных не могут начинаться с цифры, но могут содержать цифры после подчеркивания или хотя бы одну букву; они также не могут содержать пробелы, поскольку пробелы используются для определения того, где начинаются и заканчиваются команды, переменные и другие элементы.

Чтобы создать в PHP переменную с именем ourString со значением Hello World, мы должны ввести следующую команду:

$ourString = 'Hello World';

Команда заканчивается точкой с запятой (;). Точки с запятой используются, чтобы сообщить интерпретатору, где заканчивается утверждение и где начинается следующее. Их легко забыть! Если вы видите синтаксическую ошибку, вам нужно будет посмотреть на строку перед ошибкой, чтобы убедиться, что точка с запятой отсутствует.

PHP поддерживает также некоторые свои собственные имена переменных, которые начинаются с символа подчеркивания. Это так называемые предопределенными переменные. Переменные $_GET, $_POST и $_FILES содержат данные, введенные пользователем или отправленные с использованием форм. Переменные $_COOKIE и $_SESSION используются для хранения информации в течение всей сессии пользователя, а $_ENV содержит информацию о сервере.

Типы данных

PHP является языком с динамической типизацией. Это значит, что тип данных переменной выводится во время выполнения, и в отличие от ряда других языков программирования в PHP не надо указывать перед переменной тип данных.

PHP поддерживает восемь простых типов данных:

- boolean (логический тип);
- integer (целые числа);
- double (дробные числа);
- string (строки);
- array (массивы);
- object (объекты);
- resource (ресурсы);
- NULL.

Специальный тип NULL указывает, что значение переменной не определено. Использование данного значения полезно в тех случаях, когда мы хотим указать, что переменная не имеет значения. Например, если мы просто определим переменную без ее инициализации и затем попробуем ее использовать, то нам интерпретатор выдаст диагностическое сообщение, что переменная не установлена:

```
<?php
$a;
echo $a;
?>
```

Использование значения NULL поможет избежать данной ситуации. Кроме того, мы сможем проверять наличие значения и в зависимости от результатов проверки производить те или иные действия:

```
<?php
$a=NULL;
if($a)
echo "Переменная a определена"; else
echo "Переменная a не определена";
?>
```

Тип String

String - это тип данных, которые содержат строки слов. Строки бывают двух типов - в двойных кавычках и в одинарных. От типа кавычек зависит обработка строк интерпретатором. Строки, заключенные в одинарные кавычки, интерпретатор об-рабатывает как обычный текст, т. е. на экран будет выводиться текст именно таким, как в кавычках.

Строки в двойных кавычках будут проверяться интерпретатором для поиска всего того, что может быть обработано как команды PHP, специальные символы или тэги HTML. Это означает, что такие элементы, как специальные символы и переменные, перед отправкой вывода на экран будут заменены обработанными значениями.

В следующем примере переменной $result сначала присваивается значение Строки в двойных кавычках, а потом - значение Строки в одинарных кавычках. <?php

```
$a=10;
$b=5;
$result = "$a+$b <br>"; // Строка в двойных кавычках echo $result;
$result = '$a+$b'; // Строка в одинарных кавычках echo $result;
?>
```

В первом случае команда echo $result выводит число 15 и выполняет переход на новую строку экрана, потому что в двойных кавычках записано действие сложения значений переменных $a и $b и тэг
.

Во втором случае команда echo $result выводит без изменения строку $a+$b.

Полные предложения могут храниться в одной строковой переменной или могут быть построены путем объединения других строковых переменных. Управлять строками в PHP можно с помощью ряда функций, которые могут выполнять такие задачи, как поиск и замена слов, разбиение строк на части, использование заглавных букв одного или всех слов и ряд других полезных задач. Читателю данного учебного пособия предлагается изучить функции для работы со строками самостоятельно, потому что они очень похожи на аналогичные функции в языках программирования C и C++.

Строки могут использоваться для создания выходных данных, которые читает пользователь, генерирования части или всего HTML-кода для отображаемой страницы, и даже команд, которые могут передаваться в другие языки для управления их работой.

4.3.5. Экранирующие символы (escape-символы)

Экранирующие символы (escape-символы) - это символы со специальным значением в сочетании с методом экранирования языка. Мы можем указать, что нам нужен переход на новую строку в текстовом файле, используя \ n. В этом примере n - это не просто n, это означает, что нам нужна новая строка, поскольку перед ней стоит обратный слеш, экранирующий символ PHP. В PHP escape- символы обычно используются в строках в

двойных кавычках, поэтому мы можем включать специальные символы (строка в одинарных кавычках будет игнорировать это, как и другие команды PHP, переменные и тому подобное).

Написание

$string = «Я хочу потратить 5,00 $»;

приведет к ошибке имени. Вместо этого мы можем использовать $string = «Я хочу потратить $ 5,00»;

чтобы получить результат, который нам нужен. Если бы мы хотели использовать обратную косую черту, мы могли бы написать расположение папки как

$ address = «c: \\ www \\ ourfolder \\ sometext.txt»;.

В то время как мы могли бы легче сделать это с одинарными кавычками, как

$ address = ‘c:\www\ourfolder\sometext.txt’;

нам нужно добавить любые переменные, на которые мы хотим сослаться, в строку, заключенную в одинарные кавычки.

Константы

Иногда нам нужно хранить часть информации, но мы не хотим, чтобы она изменялась во время работы нашего скрипта. Для этого мы можем создать константу - по сути, переменную, которую нам не разрешено изменять. Создание константы выполняется путем вызова функции define и передачи строки с именем нашей константы и ее содержимым:

define ("OURCONSTANT", "Our constant value");

Вы заметите, что у нас нет знака доллара перед именем константы. Чтобы использовать это значение, мы можем просто повторить имя:

echo OURCONSTANT;

В нашем примере имя константы записано в верхнем регистре. Так часто делают, чтобы различать переменные и константы.

Есть также некоторые предопределенные константы в PHP, такие как PHP_ VERSION и PHP_OS. Первая даст нам номер версии для PHP, а вторая даст нам подробную информацию об операционной системе, на которой работает сервер. Поскольку у констант нет начального знака доллара, мы не можем встраивать их в строку, а вместо этого нужно объединить их, если мы хотим использовать их с другими выходными данными:

echo " This server runs on " . PHP_OS;

Точка здесь означает операцию конкатенации - это соединение нескольких элементов в одну переменную или вывод. Точка используется, чтобы указать, где эти части начинаются и заканчиваются. Если мы хотим добавить что-то к нашему выражению, мы можем продолжать добавлять точки:

echo "This server runs on ". PHP_OS . " and use PHP version ".PHP_VERSION;

4.4. Работа с формами в языке PHP

Одним из основных способов передачи данных web-сайту является обработка форм. Формы - это специальные контейнеры языка разметки HTML, которые содержат в себе различные элементы ввода: текстовые поля, кнопки и т. д. И с помощью данных форм мы можем ввести некоторые данные и отправить их на сервер, а сервер обрабатывает эти данные.

Создание форм состоит из следующих шагов:

- создание контейнера <form> </form> в разметке HTML;
- добавление в этот контейнер одного или нескольких полей ввода;
- установка метода передачи данных (GET или POST);
- установка адреса, на который будут отправляться введенные данные.

Итак, создадим новую форму. Для этого создадим новый файл forrnphp, в который поместим следующее содержимое:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h3>Вход на сайт<Ш>
<form action="login.php" method="POST">
Логин: <input type="text" name="login" /><br><br>
Пароль: <input type="text" name="password" /><br><br>
<input type="submit" value="Войти">
</form>
</body>
</html>
<?php
$login = "Не известно";
$password = "Не известно";
if(isset($_POST['login'])) $login = $_POST['login'];
if (isset($_POST['password'])) $password = $_POST['password'];
echo "Ваш логин: $login <br> Ваш пароль: $password";
?>
```

И поскольку возможны ситуации, когда поле ввода не будет заполнено, например при прямом переходе к скрипту: http://local- host:8080/login.php, то желательно перед обработкой данных проверять их наличие с помощью функции isset(). Если значение переменной введено, то функция isset() возвратит значение true.

Теперь мы можем обратиться к форме (рис. 4).

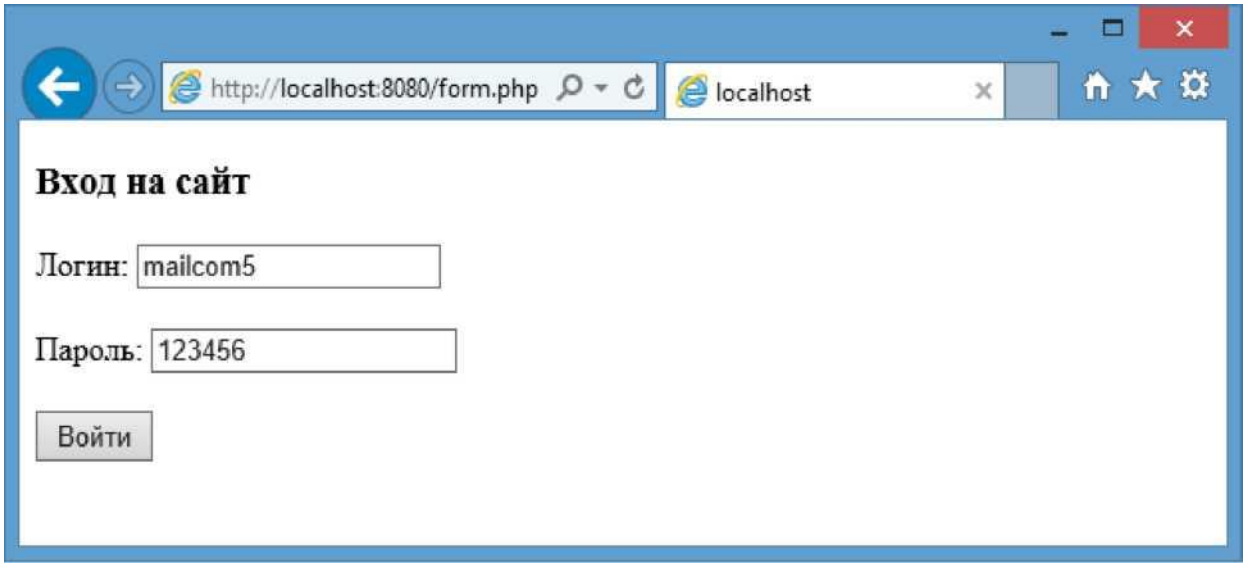


Рис. 4. Форма «Вход на сайт»

И по нажатию кнопки введенные данные методом POST будут отправлены скрипту login.php (рис. 5).

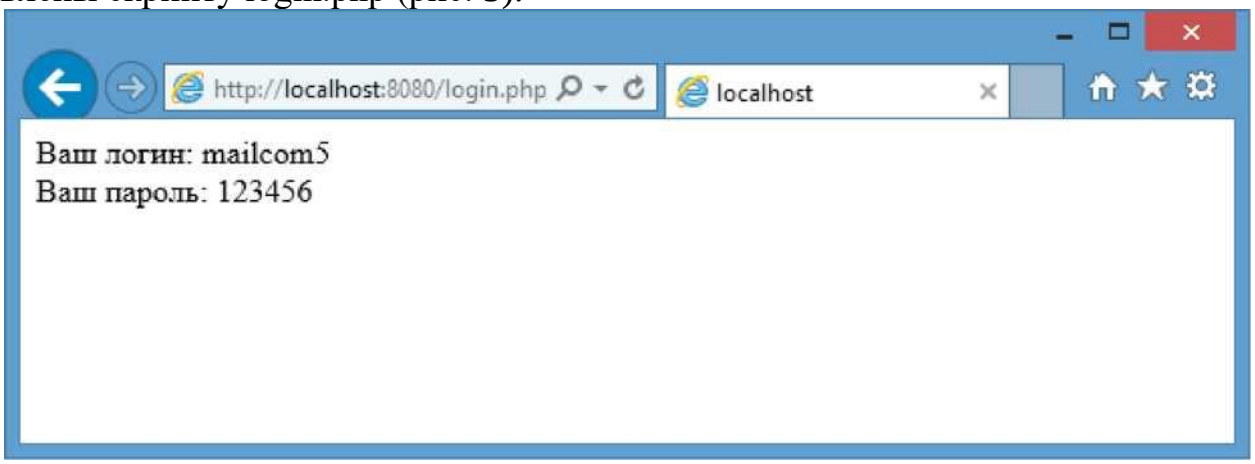


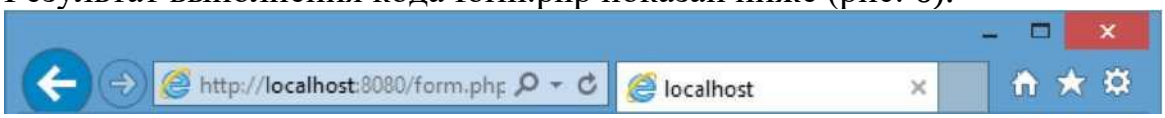
Рис 5. Результат выполнения скрипта login.php

Необязательно отправлять данные формы другому скрипту, можно данные формы обработать в том же файле формы. Для этого изменим файл form.php следующим образом:

```
<html>
<head>
<meta
charset="utf-8">
</head>
<body>
<div>
<?php
if(isset($_POST['login']) && isset($_POST['password'])){
$login=$_POST['login'];
$password = $_POST['password'];
echo "Ваш логин: $login <br> Ваш пароль: $password";
```

```
}
?>
</div>
<h3>Вход на сайт<^3>
<form method="POST">
Логин: <input type="text" name="login" /><br><br>
Пароль: <input type="text" name="password" /><br><br>
<input type="submit" value="Отправить">
</form>
</body>
</html>
```

Результат выполнения кода form.php показан ниже (рис. 6).



Ваш логин: mailcom^ Ваш пароль: 12345

Вход на сайт

Логин:

Пароль:

Отправить

Рис. 6. Результат выполнения кода *form.php*

введем в поле для логина текст *"<script>alert(hi);</script>*", а в поле для пароля введем текст *"<h2>naponb</h2>"* (рис. 7)

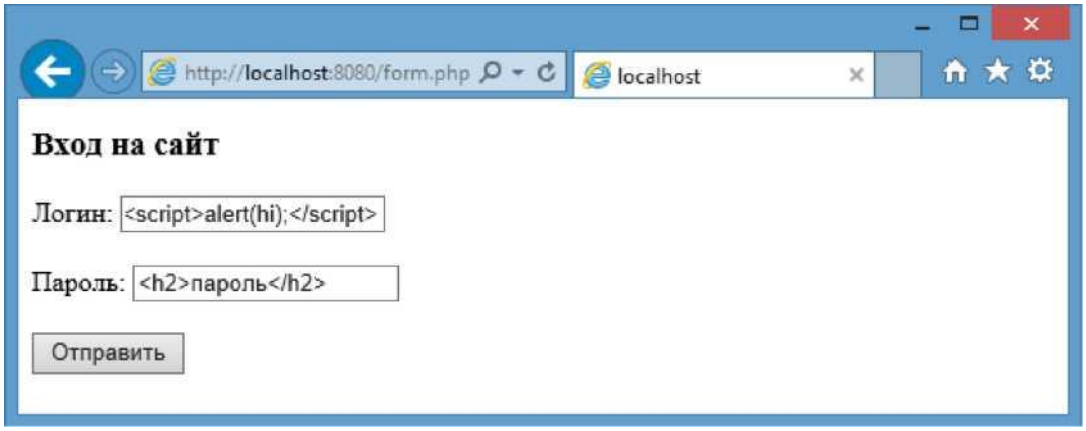


Рис. 7. Ввод в текстовые поля строк программного кода

После отправки данных в html разметку будет внедрен код JavaScript, который выводит окно с сообщением «hi». Чтобы избежать подобных проблем с безопасностью, следует применять функцию htmlentities():

```
if(isset($_POST['loginJ) && isset($_POST['password'])){
$login=htmlentities($_POST['login']);
$password = htmlentities($_POST['password']); echo "Ваш логин: $login
<br> Ваш пароль: $password";
}
```

И теперь даже после ввода кода HTML или JavaScript все теги будут экранированы, и мы получим на странице вывод, как на рисунке 8.

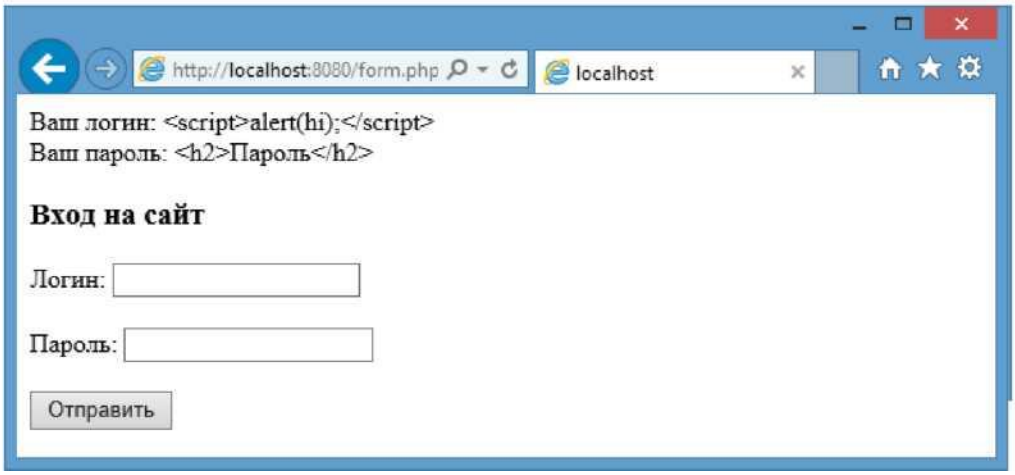


Рис. 8. Результат применения функции *htmlentities*

Результат работы функции *strip_tags()* при том же вводе показан на рисунке 10.

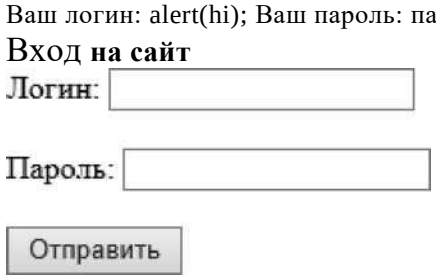


Рис. 9. Результат работы функция *strip_tags()*

Задания по практической работе:
Создать форму на PHP:

1. **Форма регистрации пользователя**: Ввод имени, электронной почты, пароля и подтверждения пароля.
2. **Форма входа в систему**: Ввод электронной почты и пароля для авторизации.
3. **Форма обратной связи**: Ввод имени, электронной почты, темы и сообщения.
4. **Форма добавления книги**: Ввод названия, автора, жанра и описания книги.
5. **Форма заказа товара**: Ввод имени, адреса доставки, номера телефона и информации о заказе.
6. **Форма добавления нового события**: Ввод названия события, даты, времени и места проведения.
7. **Форма редактирования профиля**: Ввод имени, электронной почты и информации о пользователе.
8. **Форма добавления вакансии**: Ввод названия должности, описания, требований и зарплаты.

9. **Форма создания нового поста в блоге**: Ввод заголовка, контента, тегов и изображения.
10. **Форма для оценки фильма**: Ввод имени пользователя, названия фильма и оценки.
11. **Форма поиска**: Ввод ключевого слова для поиска в базе данных.
12. **Форма регистрации на курс**: Ввод имени, электронной почты и выбранного курса.
13. **Форма создания опроса**: Ввод названия опроса, вопросов и вариантов ответов.
14. **Форма добавления нового проекта**: Ввод названия проекта, описания и сроков.
15. **Форма компании для аренды автомобиля**: Ввод данных о модели, годе выпуска, цене аренды и условиях.
16. **Форма добавления нового рецепта**: Ввод названия, ингредиентов и описания приготовления.
17. **Форма обработки жалобы**: Ввод данных о жалобе, пользователе и решении проблемы.
18. **Форма подписки на новости**: Ввод имени и адреса электронной почты.
19. **Форма регистрации на концерт**: Ввод имени, количества билетов и предпочтений по местам.
20. **Форма ввода данных о медицинском пациенте**: Ввод имени, возраста, диагноза и контактной информации.

Лабораторная работа 4. Создание базы данных MySQL. Основные функции.

Цель работы – научиться создавать базы данных MySQL с помощью программы NetBeans.

В лабораторных работах 5–8 приводится пример разработки web-приложения ***Wish list***, после развертывания которого пользователи получают возможность размещения и совместного использования информации в списках желаний, например создания списков подарков к свадьбе, дню рождения или другим праздникам.

Приложение поддерживает регистрацию и авторизацию пользователей.

Любой зарегистрированный пользователь может просмотреть списки пожеланий других пользователей и имеет возможность редактирования собственных списков желаний.

В этом лабораторном практикуме для создания, выполнения и отладки проектов на PHP используются IDE NetBeans для PHP и локальный web-сервер XAMPP.

Загрузку IDE NetBeans для PHP можно осуществить с сайта https://netbeans.apache.org/download/index.html.

XAMPP можно загрузить с электронного адреса: https://www. apachefriends.org/download.html.

Создание базы данных MySQL с помощью программы NetBeans

IDE NetBeans поставляется с включенной поддержкой для MySQL. Для получения доступа к серверу баз данных MySQL в IDE NetBeans необходимо настроить свойства сервера MySQL.

1. Щелкните правой кнопкой мыши узел Databases («Базы данных») в окне Services («Службы») и выберите Register MySQL Server («Зарегистрировать MySQL»). Открывается диалоговое окно свойств сервера MySQL (рис. 10).

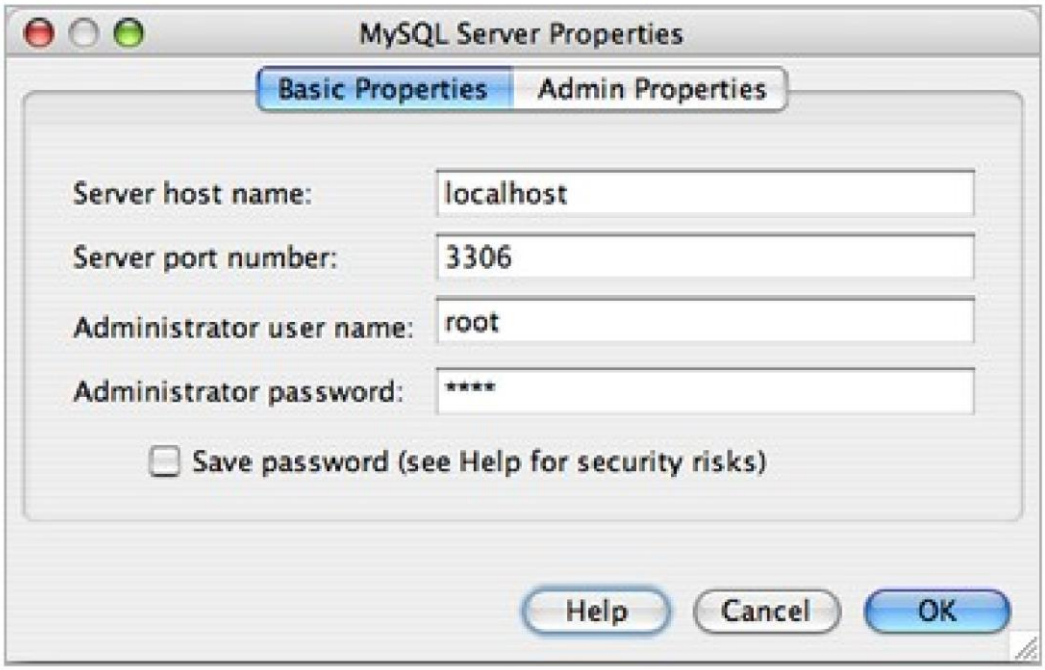


Рис. 10. Диалоговое окно свойств MySQL Server

2. Убедитесь, что имя узла и порт сервера указаны правильно.Обратите внимание, что среда IDE вводит localhost как имя узла сервера по умолчанию и 3306 как номер порта сервера по умолчанию.

3. Введите имя администратора (если оно не отображается). Вам необходим доступ с правами администратора, чтобы иметь возможность создавать и удалять базы данных.

4. Введите пароль администратора. По умолчанию установлено пустое значение (пустой пароль является допустимым).

5. Нажмите вкладку «Свойства администратора» в верхней части диалогового окна.

Отобразится соответствующая вкладка, предоставляющая возможность ввода сведений для управления сервером MySQL (рис. 11).

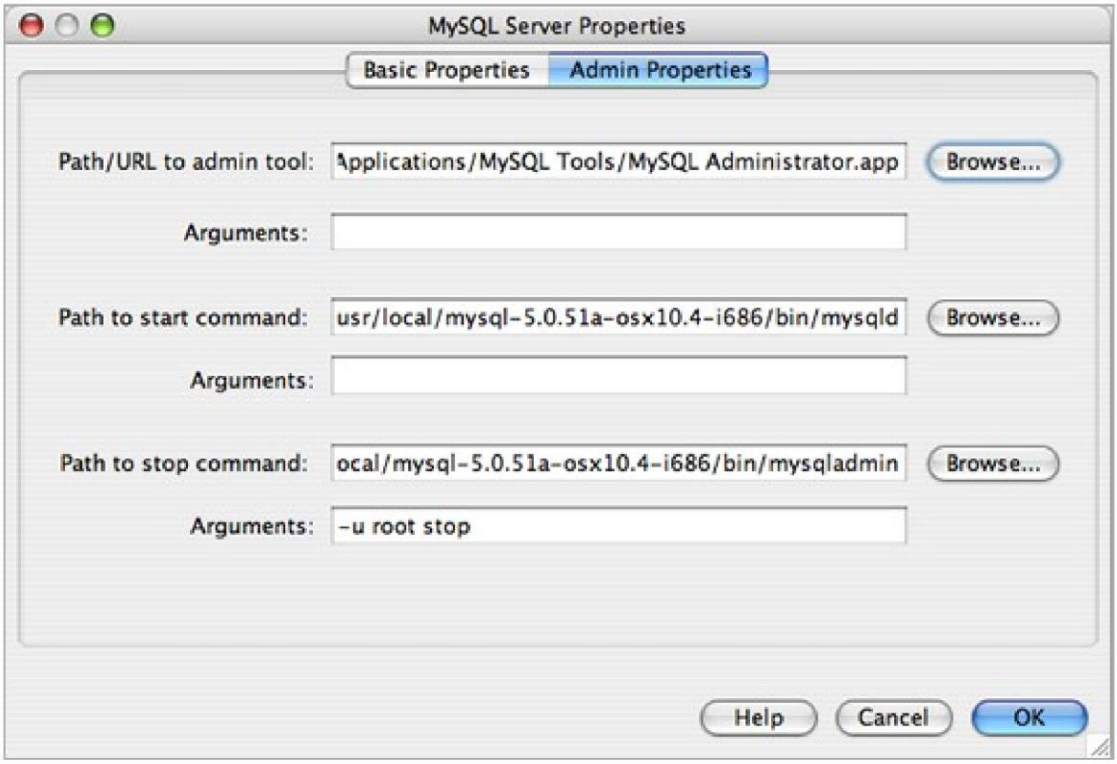


Рис. 11. Внешний вид вкладки «Свойства администратора»

6. В поле «Путь/URL-адрес к средству администрирования» введите путь к средству администрирования MySQL (например, ***MySQL Admin Tool, PhpMyAdmin*** или другому подходящему webсредству) или найдите его при помощи кнопки «Обзор».

7. В поле «Путь к команде запуска» введите соответствующийпуть MySQL или найдите его при помощи кнопки «Обзор». Для получения команды запуска найдите файл ***mysqld*** в папке ***bin*** каталога установки MySQL. Также может потребоваться другая команда запуска при установке MySQL в составе установки XAMPP.

8. В поле «Путь к команде остановки» введите путь к командеостановки MySQL или найдите его при помощи кнопки «Обзор». Обычно требуется ввести путь к файлу ***mysqladmin*** в папке ***bin*** каталога установки MySQL. При использовании команды ***mysqladmin*** введите ***-u root stop*** в поле «Аргументы» для получения прав пользователя ***root*** на остановку сервера.

9. Если настройка выполнена корректно, нажмите кнопку«ОК».

Запуск сервера MySQL

Перед попыткой подключения к серверу базы данных MySQL необходимо убедиться в том, что сервер запущен на компьютере. Если сервер базы данных не подключен, вы увидите ***disconnected*** рядом с именем пользователя в узле MySQL Server в окне «Службы» и не сможете развернуть узел.

Для подключения к серверу баз данных убедитесь, что сервер базы данных MySQL запущен на компьютере, щелкните правой кнопкой мыши ***Базы данных > узел MySQL Server*** в окне «Службы» и выберите «Подключить». Может отобразиться запрос на ввод пароля для подключения к серверу.

После подключения сервера вы сможете развернуть узел MySQL Server и просмотреть все доступные базы данных MySQL.

Технология создания базы данных и подключения к ней

Язык SQL является широко распространенным способом взаимодействия с базами данных. Для этого в IDE NetBeans имеется встроенный редактор SQL. Обычно редактор SQL доступен с помощью параметра ***«Выполнить команду»*** из контекстного меню узла подключения (или дочерних узлов узла подключения). После установления подключения к серверу MySQL можно создать новый экземпляр базы данных в редакторе SQL. Для продолжения работы создайте экземпляр с именем ***MyNewDatabase***:

1. В окне «Службы» среды IDE щелкните правой кнопкой мыши узел сервера MySQL Server и выберите «Создать базу данных». Откроется диалоговое окно «Создание базы данных MySQL».
2. В диалоговом окне «Создание базы данных MySQL» введите имя новой базы данных ***MyNewDatabase***. Не устанавливайте флажок (рис. 12).

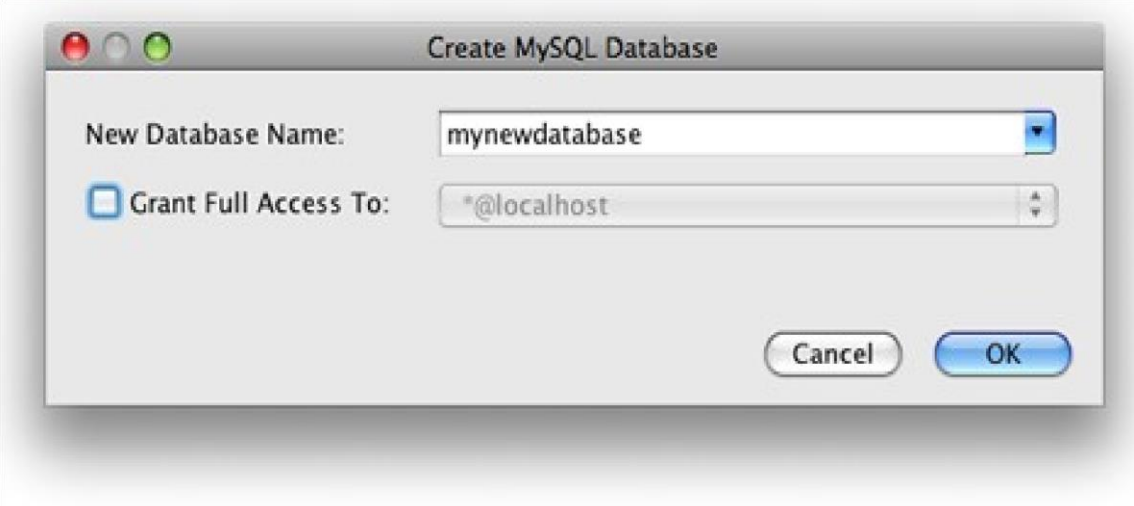


Рис. 12. Диалоговое окно «Создание базы данных MySQL»

Определенному пользователю можно предоставить полный доступ. По умолчанию только администратор обладает правами на выполнение определенных команд. Раскрывающийся список позволяет присваивать эти права определенным пользователям.

3. Нажмите кнопку «ОК». В узле «Сервер MySQL» окна «Службы» будет выведена новая база данных.
4. Щелкните узел новой базы данных правой кнопкой мыши ивыберите «Подключение», чтобы установить соединение с базой данных.

Открытые подключения к базе данных отображаются в узле «Установленные подключения» в окне «Службы».

Создание таблиц базы данных

После установления подключения к базе данных ***MyNewDatabase*** можно начинать изучение принципов создания таблиц, заполнения их данными и изменения данных в таблицах.

База данных ***MyNewDatabase*** в настоящее время пуста. В среде IDE таблицу базы данных можно добавить при помощи диалогового окна «Создание таблицы» или посредством ввода запроса SQL и его запуска напрямую из редактора SQL. Можно использовать оба метода.

Использование редактора SQL

1. В проводнике баз данных разверните узел подключения ***MyNewDatabase***  и обратите внимание, что там содержится три подпапки: «Таблицы», «Представления» и «Процедуры».
2. Щелкните правой кнопкой мыши папку ***Tables*** («Таблицы») и выберите ***Execute Command*** («Выполнить команду»). В главном окне редактора SQL отобразится пустой холст.
3. В редакторе SQL введите следующий запрос. Это определение создаваемой таблицы ***Counselor***.

```
CREATE TABLE Counselor (
  id SMALLINT UNSIGNED NOT NULL
AUTO_INCREMENT,  firstName VARCHAR (50),  nickName
VARCHAR (50),  lastName VARCHAR (50),  telephone
VARCHAR (25),  email VARCHAR (50),
    memberSince DATE DEFAULT '0000-00-00',
    PRIMARY KEY (id)
      );
```

После выполнения запроса в окне «Вывод» будет создан отклик от механизма SQL, указывающий на успешность выполнения или на ошибку.

1. Чтобы выполнить запрос, нажмите кнопку «ВыполнитьSQL» на панели задач в верхней части (Ctrl-Shift-E) или щелкните правой кнопкой мыши в редакторе SQL Editor и выберите ***Выполнить оператор***. В среде IDE будет создана таблица базы данных ***Counselor***, а на экране появится сообщение о выполнении запроса.
2. Для проверки изменений щелкните правой кнопкой мышиузел «Таблицы» в проводнике баз данных и выберите ***Обновить***.

Обратите внимание, что новый узел таблицы ***Counselor***  теперь отображается ниже узла «Таблицы»в проводнике баз данных. Если развернуть

узел таблицы, можно увидеть созданные столбцы (поля), начинающиеся первичным ключом  .

Использование диалогового окна «Создание таблицы»

1. В проводнике баз данных щелкните правой кнопкой мышиузел «Таблицы» и выберите «Создать таблицу». Откроется диалоговое окно «Создание таблицы».
2. Введите ***Subject*** в текстовое поле «Имя таблицы».
3. Нажмите кнопку «Добавить столбец».
4. В поле Name («Имя») столбца введите id. Выберите ***SMALLINT*** в качестве типа данных из раскрывающегося списка ***Type***. Нажмите кнопку «ОК».
5. Установите флажок Primary Key («Первичный ключ») в диалоговом окне Add Column. В этом действии выполняется определение первичного ключа таблицы. Все таблицы, созданные в реляционных базах данных, должны содержать первичный ключ. Обратите внимание, что при выборе флажка «Ключ» выполняется автоматическая установка флажков «Индекс» и «Уникальный», при этом отменяется выбор флажка «Значение отсутствует». Это объясняется тем, что первичные ключи применяются для определения уникальной строки базы данных и по умолчанию используются в индексе таблицы. Поскольку все строки должны иметь уникальный идентификатор, первичные ключи не могут иметь значение Null.
6. Повторите эту процедуру, добавив оставшиеся столбцы, какпоказано в следующей таблице.
7. Выполняется создание таблицы ***Subject***, в которой будут содержатся данные для каждой из следующих записей (рис. 13).

 Имя: тема

- **Описание:** описание темы
- **Идентификатор таблицы Counselor:** идентификатор, соответствующий идентификатору в таблице Counselor.

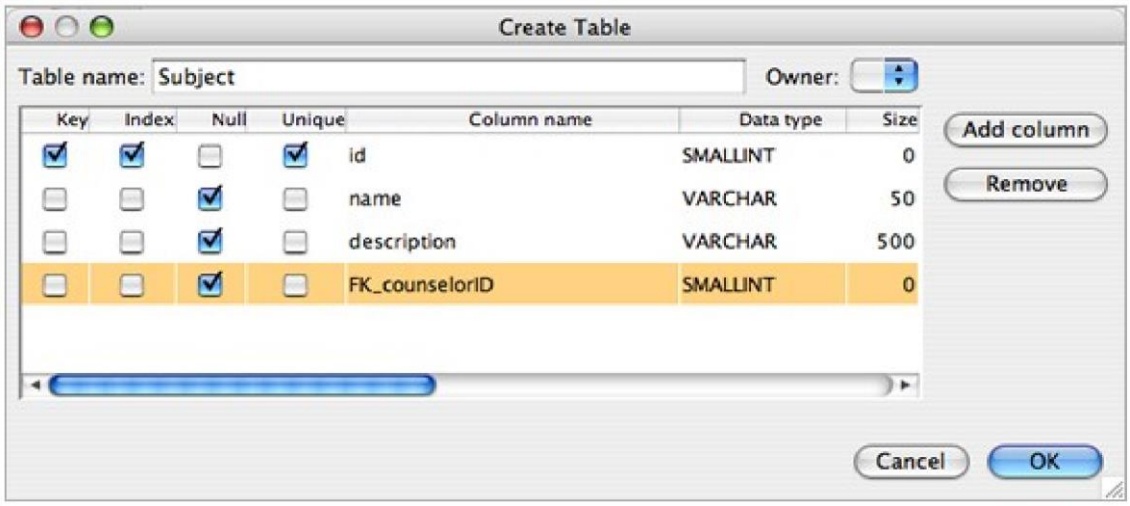


Рис. 13. Структура таблицы ***Subject***

Убедитесь, что все поля в диалоговом окне «Создание таблицы» соответствуют полям в примере выше и нажмите кнопку «ОК». IDE создает таблицу ***Subject*** в базе данных и можно увидеть, что новый узел таблицы Subject  отображается непосредст-венно под пунктом «Таблицы» в проводнике баз данных.

Создание пользователя базы данных

Перед созданием базы данных необходимо создать соответствующего пользователя, которому будет предоставлено право на выполнение любых операций в базе данных. Создание пользователя базы данных включает в себя следующие действия:

- Подключение к серверу *MySQL* пользователя с ролью *root*.
- Подключение к базе данных системы *MySQL* пользователя с ролью *root.* Поскольку выполнение команды *SQL* невозможно без подключения к какой-либо базе данных, это действие позволяет запустить команду *SQL*, необходимую для создания пользователя.
- Выполнение оператора создания пользователя в *MySQL*.

1. Запустите *IDE*, перейдите в окно «Службы» (Ctrl-5) и разверните узел 'Базы данных' (рис. 14).

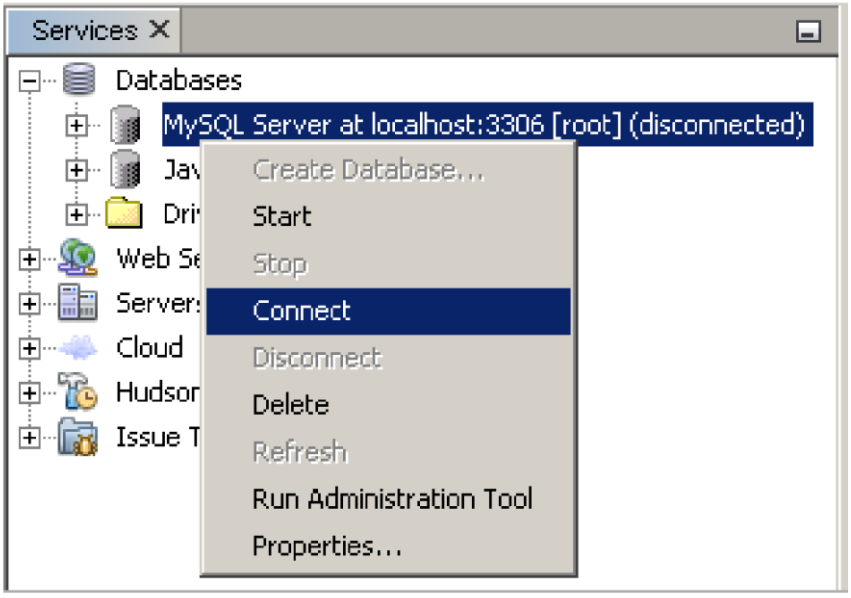


Рис. 14. Контекстное меню «Службы»

2. Для подключения к серверу базы данных *MySQL* перейдите к узлу *MySQL Server* и выберите *Connect* в контекстном меню.

3. *IDE NetBeans* устанавливает соединение с сервером *MySQL*, проверяет наличие доступных баз данных с помощью сервера, обнаруживает системную базу данных *mysql* и добавляет соответствующий новый узел *mysql* к дереву баз данных (рис. 15).

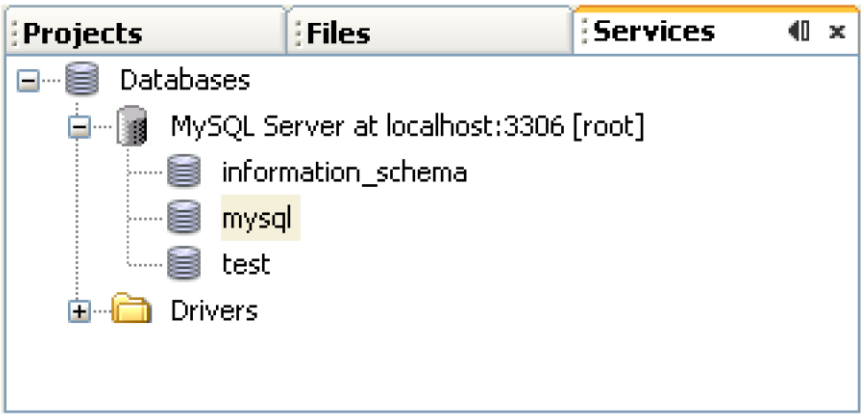


Рис. 15. Добавление нового узла к дереву баз данных

4. Для выполнения команды *SQL* необходимо подключение к базе данных. Поскольку доступна только система *MySQL*, следует подключиться к этой системе. Для подключения к системной базе данных перейдите к узлу *mysql* и выберите *Connect* в контекстном меню. Если подключение на данный момент отсутствует, появится диалоговое окно *New Database Connection*. В поле *User Name* по умолчанию вводится значение ***root.*** В поле *Password* введите пароль пользователя ***root***.

Если вы уже подключались к базе данных *mysql*, это диалоговое окно не отобразится. Вместо этого в дереве просто появится новый узел подключения.

В диалоговом окне *New Database Connection* появится сообщение *Connection established*. Нажмите кнопку *OK*. К дереву баз данных добавлен новый узел с именем jdbc:mysql://localhost:3306/mysql.

5. Перейдите к узлу jdbc:mysql://localhost:3306/mysql и выберите в контекстном меню *Execute Command*.

Откроется окно *SQL Command*. В окне *SQL Command* введите следующие команды:

CREATE USER 'phpuser'@'localhost'
IDENTIFIED BY 'phpuserpw'

Выберите в контекстном меню *Run Statement*. Если команда выполнена успешно, в строке состояния выводится следующее сообщение: *SQL Statement(s) executed successfully*. Если выводится другое сообщение, проверьте синтаксическую правильность введенных команд и выполните советы, относящиеся к данному сообщению.

Создание базы данных Wishlist

Для создания базы данных выполните следующие действия:

1. Перейдите к узлу *MySQL Server at localhost:3306* и выберите из контекстного меню *Create Database*. Появится диалоговое окно *Create MySQL Database*. Заполните поля следующим образом:

- В поле имени *Database Name* введите *wishlist*.
- Установите флажок ***Grant full access to user*** и выберите в раскрывающемся списке ***phpuser@localhost***. Нажмите кнопку *OK*.

Функция *Grant full access to user*, предоставляющая пользователю полные права доступа, срабатывает не всегда. Если она не работает, подключитесь к базе данных как пользователь *root* и отправьте запрос *SQL*.

GRANT ALL ON wishlist.* TO phpuser@localhost.

2. Подключение к базе данных появится в дереве. Однако это подключение создано для пользователя *root*. Вам требуется подключение для пользователя *phpuser*.

Установление подключения к базе данных Wishlist

На предыдущем этапе вы создали базу данных wishlist с подключением для пользователя root. Теперь необходимо создать подключение для пользователя phpuser:

1.В окне «Службы» щелкните правой кнопкой мыши узел «Базы данных» и выберите «Создать подключение». Открывается мастер создания подключений. На панели «Обнаружение драйвера» в мастере создания подключений выберите *MySQL (Connector/ J Driver)*. Нажмите «Далее». Открывается панель «Настройка соединения». В поле «База данных» введите ***wishlist***.

2. В полях *User Name* и *Password* введите соответственно имя и пароль пользователя, указанные в разделе «Создание владельца (пользователя) базы данных» (в нашем примере – это phpuser и phpuserpw). Установите флажок

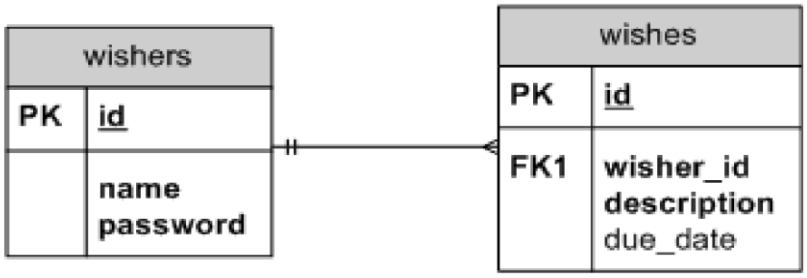
«Запомнить пароль». Нажмите «Проверить подключение». Если соединение установлено успешно, нажмите *OK*. В дереве баз данных будет отображаться соответствующий новый узел подключения. Подключение для пользователя root к базе данных wishlist можно удалить.

 Нажмите на подключение jdbc:mysql://localhost:3306/wishlist и выберите «Удалить».

Проектирование структуры базы данных Wishlist

Для размещения и сохранения всех необходимых данных требуются две таблицы:

Таблица 1



- Таблица *wishers* – для сохранения имен и паролей зарегистрированных пользователей.
- Таблица *wishes* – для содержания описания требований.

Таблица *wishers* содержит три поля:

1) *id* – уникальный идентификатор пользователя. Это поле используется в качестве первичного ключа;
2) *name* – имя;
3) *password* – пароль.

Таблица wishes содержит четыре поля:

1) id – уникальный идентификатор пользователя. Это поле используется в качестве первичного ключа;
2) *wisher_id* – идентификатор пользователя, оставившего пожелание. Это поле используется в качестве внешнего ключа; 3) *description* – описание;
4) *due_date* – требуемая дата исполнения пожелания.

Таблицы связаны посредством идентификатора пользователя. Все поля таблицы *wishes* являются обязательными для заполнения, за исключением *due_date*.

Создание таблиц

1.Для подключения к базе данных щелкните правой кнопкой мыши узел подключения jdbc:mysql://localhost:3306/wishlist и выберите *Connect* в контекстном меню. Если пункт меню недоступен, пользователь уже подключен. Перейдите к действию 2.

2.В том же контекстном меню выберите *Execute Command*. Откроется пустое окно *SQL Command*.

3.Для создания таблицы *wishers*. Введите следующий запрос SQL (отметьте, что как набор символов следует прямо установить UTF-8 для интернационали-зации):

```
CREATE TABLE wishers(
 id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 name CHAR(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL UNIQUE,
 password CHAR(50) CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL
)
```

Можно получить уникальный номер, автоматически создаваемый *MySQL*, задав свойство *AUTO_INCREMENT* для поля. *MySQL* создаст уникальный номер посредством увеличения на единицу последнего номера в таблице и автоматически добавит его к значению поля с этим свойством. В нашем примере автоматически должно увеличиваться значение в поле *ID*.

Щелкните запрос правой кнопкой мыши, затем выберите *Run Statement* в контекстном меню.

Механизмом хранения по умолчанию для *MySQL* является *MyISAM*, не поддерживающий внешние ключи. Если нужна поддержка внешних ключей, используйте в качестве механизма хранения *InnoDB.*

Для создания таблицы *Wishes* введите следующий запрос *SQL*:

```
CREATE TABLE wishes(
 id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 wisher_id INT NOT NULL,
 description CHAR(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL,  due_date DATE,
 FOREIGN KEY (wisher_id) REFERENCES wishers(id)
)
```

Щелкните запрос правой кнопкой мыши, затем выберите *Run Statement* в контекстном меню. Для проверки того, что новые таблицы добавлены к базе данных, перейдите к окну *Services*, а затем к узлу подключения jdbc:mysql://localhost:3306/wishlist. Нажмите правую кнопку мыши и выберите *Refresh*. В дереве появятся узлы *wishers* и *wishes*.

Ввод тестовых данных

Для тестирования приложения необходимо наличие некоторых данных в базе данных. В приведенном ниже примере показано, каким образом можно добавить данные для двух пользователей и четырех желаний.

В узле подключения jdbc:mysql://localhost:3306/wishlist щелкните правой кнопкой мыши и выберите *Execute Command*. Откроется пустое окно *SQL Command*.

Для добавления данных пользователя введите следующие команды:

```
INSERT INTO wishers (name, password)
```

VALUES ('Tom', 'tomcat');

Щелкните запрос правой кнопкой мыши и выберите из контекстного меню *Run Statement*.

Оператор не содержит значения для поля идентификатора. Значения вводятся автоматически, поскольку указан тип поля AUTO_INCREMENT.

Введите данные другого тестового пользователя:

```
INSERT INTO wishers (name, password)
VALUES ('Jerry', 'jerrymouse');
```

Для добавления пожеланий (*wishes*) введите следующие команды:

```
INSERT INTO wishes (wisher_id, description, due_date)
 VALUES (1, 'Sausage', 080401);
INSERT INTO wishes (wisher_id, description)
 VALUES (1, 'Icecream');
INSERT INTO wishes (wisher_id, description, due_date)
 VALUES (2, 'Cheese', 080501);
INSERT INTO wishes (wisher_id, description)
 VALUES (2, 'Candle');
```

Выберите запросы, щелкните каждый правой кнопкой мыши по каждому из них и выберите *Run Selection* в контекстном меню.

Для просмотра тестовых данных щелкните соответствующую таблицу правой кнопкой мыши и выберите из контекстного меню *View Data*.

Разработка web-приложения на языке PHP. Чтение из базы данных

Цель работы – научиться с помощью NetBeans создавать и настраивать проект PHP для чтения из базы данных.

В этой лабораторной работе рассматривается создание и настройка проекта PHP для разработки приложения, создание списка страниц в приложении и определение отношений между ними. Кроме того, основная функциональность разрабатывается и тестируется на данных, введенных в пример базы данных в лабораторной работе

5.

Создаваемый в этой лабораторной работе код PHP выполняет следующие функции.

1. Получает имя лица, введенного пользователем.
2. Проверяет наличие этого лица в базе данных. Если лицо отсутствует в базе данных, выполняется выход с сообщением об ошибке.
3. Отображение таблицы желаний этого лица.

Текущий документ является частью краткого учебного курса «Создание приложения, управляемого базой данных, в IDE NetBeans для PHP».

Создание проекта PHP

Выберите *Файл* > *Создать проект* (Ctrl-Shift-N в Windows).

Создайте новый проект PHP с именем ***wishlist***. После создания проекта PHP по умолчанию он будет содержать файл индекса ***index.php***.

Определение блок-схемы приложения

В рамках приложения возможны следующие варианты использования:

1.Пользователь просматривает список *Wish list* другого пользователя.

2.Пользователь регистрируется в качестве нового пользователя.

3.Пользователь входит в систему и создает собственный список желаний *Wish list*.

4.Пользователь входит в систему и редактирует свой список желаний.

Для реализации этих базовых функциональных возможностей потребуется добавить следующие файлы PHP (рис. 9):

1. Первая страница *index.php* для входа в систему, регистрации и перехода к спискам *Wish list* других пользователей.
2. Страница *wishlist.php* для просмотра списка *Wish list* конкретного пользователя.
3. Страница *createNewWisher.php* для регистрации в качестве автора желаний.
4. Страница *editWishList.php* для редактирования списка его владельцем.
5. Страница *editWish.php* для создания и редактирования желаний.

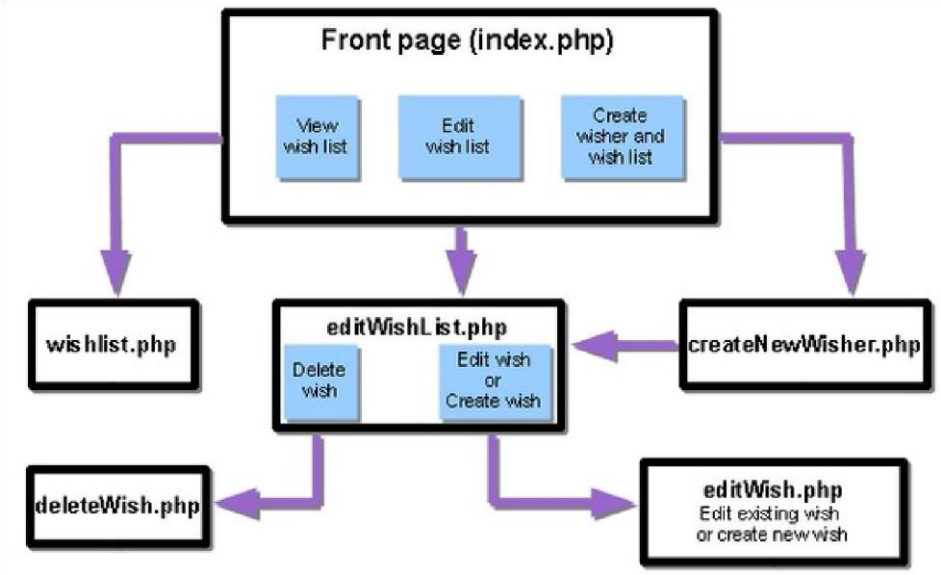


Рис. 16. Блок-схема приложения

После выполнения предварительных действий можно приступить к реализации базовой функциональности приложения. Начать следует с просмотра списка *Wish list* одного из пользователей. Для этой функции не требуется выполнять проверку допустимости, ее можно легко протестировать, поскольку в базу данных уже внесены тестовые данные. Функциональность компонента реализуются на двух страницах – *index.php* и *wishlist.php*.

Добавление формы к index.php

Файл *index.php* не содержит код PHP, таким образом, можно просто удалить следующий блок.

Файл *index.php* используется в двух целях:

1. Отображение страницы с элементами управления для вводаданных.
2. Передача введенных данных в другой файл PHP для обработки данных. Данные передаются в файл с именем *wishlist.php*, который создается в следующем разделе.

Эти действия выполняются с использованием формы HTML.

Каждая форма HTML содержит следующее:

1. Набор полей, соответствующих элементам управления настранице.
2. «Действие», выполняемое после отправки пользователемданных в форме. Действие представлено путем к странице для обработки данных.

Для добавления формы к *index.php* выполните следующее.

- Перейдите к окну «Проекты», разверните узел проекта и узел «Файлы исходного кода», затем дважды щелкните файл index.php. Файл index.php откроется в основной области редактора IDE. Файл содержит шаблон для ввода кодов PHP и HTML.

- Удалите блок PHP. Файл *index.php* не содержит код PHP. Откройте «Палитру» из меню «Окно» или нажав *Ctrl-Shift-8*.
- Из раздела **HTML** палитры (рис. 17) перетащите форму в раздел <body> файла index.php.

Рис. 17. Файл index.php и открытая палитра

Откроется диалоговое окно «Вставить форму» (рис. 18). В поле «Действие» введите путь к файлу, в которой форма будет передавать данные. В данном случае введите ***wishlist.php***. (Этот файл будет создан в том же месте, что и файл ***index.php***.)

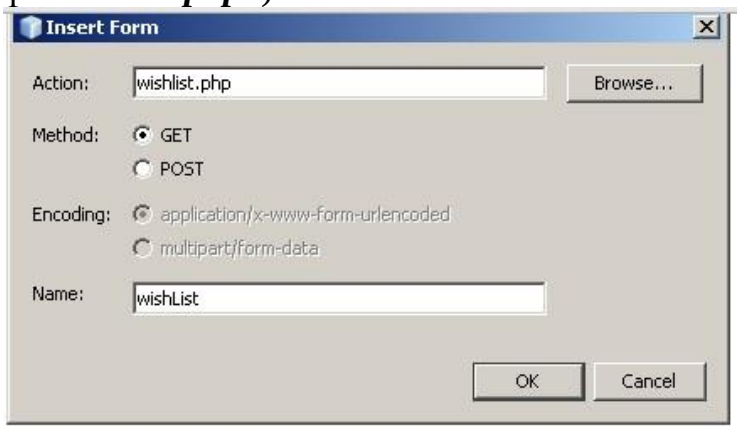


Рис. 18. Диалоговое окно «Вставить форму»

Выберите метод ***GET*** для передачи данных. Присвойте форме произвольное имя, например wishList. Нажмите кнопку *OK* после выполнения действия.

Теперь файл выглядит следующим образом (рис. 19).

Рис. 19. Файл index.php с формой

Между открывающим и закрывающим тегами формы введите текст «Показать список желаний:».

Перетащите компонент «Ввод текста» из раздела **Формы HTML** палитры в пространство после текста «Показать список желаний:». Откроется диалоговое окно «Вставка ввода текста».

Присвойте вводу название ***user***. Выберите тип ввода text. Оставьте все поля пустыми и нажмите кнопку «ОК». Добавьте пустую сроку над тегом </form>. В эту пустую строку перетащите компонент «Кнопка» из раздела **Формы HTML** палитры.

Откроется диалоговое окно «Вставить кнопку». Введите Go в поле «Метка» и нажмите кнопку «ОК».

Теперь форма выглядит так, как показанный ниже код, с одним отличием. В коде ниже атрибут *method* явно указан в теге <form>. *IDE NetBeans* не добавил атрибут метода к используемой форме, поскольку значением по умолчанию этого атрибута является ***GET***. Однако явное указание атрибута ***method*** упрощает понимание кода.

```
<form action="wishlist.php" method="GET" name="wishList">
    Show wish list of: <input type="text" name="user" value=""/>
<input type="submit" value="Go" />
</form>
```

Открывающий тег ***<form>*** содержит атрибут action. Атрибут *action* указывает файл, в который форма передает данные. В данном случае файл имеет имя ***wishlist.php*** и находится в той же папке, что и файл *index.php*. (Этот файл будет создан в разделе *Создание wishlist.php* и *тестирование приложения*.)

Открывающий тег ***<form>*** также содержит метод для применения к переданным данным (GET). PHP использует массив $_GET или $_POST для

значений, переданных этой формой, в зависимости от значения атрибута *method*. В данном случае PHP использует $_GET.

Компонент ввода ***text*** – текстовое поле для ввода имени пользователя, список пожеланий которого необходимо просмотреть. Начальное значение текстового поля – пустая строка. Имя этого поля – ***user***. PHP использует имя поля при создании массива для значений поля. В данном случае массив для значений этого поля – ***htmlentities($_GET["user"])***.

Компонент ввода **submit** со значением *Go*. Тип *submit* означает, что поле ввода отображается на странице как кнопка. Значение *Go* – это метка поля. При нажатии пользователем кнопки данные в компоненте text передаются в файл, указанный в атрибуте action.

Создание страницы wishlist.php и тестирование приложения

В разделе «Добавление формы к index.php» была создана форма, с помощью которой пользователь отправляет имя лица, список пожеланий которого необходимо просмотреть. Имя передается странице *wishlist.php*. Однако этой страницы не существует. Если выполнить *index.php*, при отправке имени возникнет ошибка «404: Файл не найден». В этом разделе будет создан файл wishlist.php, затем будет выполнено тестирование приложения.

Для создания ***wishlist.php*** и тестирования приложения выполните следующие действия.

В созданном проекте ***wishlist*** щелкните правой кнопкой мыши узел «Исходные файлы» и выберите «Создать > Файл PHP» в контекстном меню. Откроется мастер создания web-страниц PHP.

Введите wishlist в поле «Имя файла» и нажмите кнопку «Готово».

Щелкните правой кнопкой мыши узел «Источники» и выберите «Выполнить проект» в контекстном меню или щелкните значок «Выполнить главный проект» на панели инструментов, если проект задан как главный.

В поле *Show wish list* of введите *Tom* и нажмите *Go*. Появится пустая страница со следующим URL-адресом: ***http://localhost:90/Lesson2/wishlist.php?user=tom.***

Наличие этого URL-адреса означает, что главная страница функционирует правильно.

Установка подключения и получение идентификатора автора пожеланий

Дважды щелкните файл *wishlist.php*. Открывшийся шаблон отличается от *index.php*. Файл должен начинаться и заканчиваться тегами <html></html> и <body></body>, поскольку файл будет содержать также код HTML.

```
<!DOCTYPE html>
<html>
  <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title></title>
    </head>
    <body>
        <?php
        // put your code here
        ?>
    </body>
</html>
```

Для отображения заголовка после тега открытия <body> и перед генерируемым тегом <?php введите следующий блок кода:

Wish List of <?php echo htmlentities($_GET["user"])."
";?> Теперь код должен выглядеть следующим образом:

```
<body>Wish List of <?php echo htmlentities($_GET["user"])."<br/>";?>
<?php
    // put your code here
</body>
```

Блок кода PHP выводит на экран данные, которые поступают посредством метода ***GET*** в поле ***user***. Это данные передаются со страницы ***index.php***, где имя владельца списка *Wish list – Tom* было введено в текстовом поле ***user***.

Удалите раздел в шаблоне блока PHP с комментарием. В этом месте введите или вставьте следующий код. Этот код открывает подключение к базе данных.

```
$con = mysqli_connect("localhost", "phpuser", "phpuserpw"); if
(!$con) {
    exit('Connect Error (' . mysqli_connect_errno() . ') '
        . mysqli_connect_error());
}
//set the default client character set mysqli_set_charset($con, 'utf-
8');
```

В соответствии с кодом производится попытка подключения к базе данных и выдается сообщение об ошибке в случае неудачи.

Под фрагментом кода, описывающим подключение к базе данных, в том же блоке PHP укажите следующий код:

```
mysqli_select_db($con, "wishlist");
$user = mysqli_real_escape_string($con, htmlentities($_GET["user"]));
$wisher = mysqli_query($con, "SELECT id FROM wishers WHERE
name='" . $user . "'");
if (mysqli_num_rows($wisher) < 1) {     exit("The person " .
htmlentities($_GET["user"]) . " is not found.
```

```
Please check the spelling and try again");
}
$row = mysqli_fetch_row($wisher);
$wisherID = $row[0];
mysqli_free_result($wisher);
```

Этот код получает идентификатор автора желаний, чей список был запрошен. Если автор пожеланий отсутствует в базе данных, код уничтожает/завершает процесс и отображает сообщение об ошибке.

Осуществляется выбор данных из базы данных ***wishlist*** с помощью подключения ***$con***. Критерием отбора является имя, полученное со страницы ***index.php*** как ***user***.

Синтаксис оператора SQL SELECT может быть кратко описан следующим образом:

1. После оператора SELECT укажите поля, из которых должны быть получены данные. Все поля отмечены звездочкой (*).
2. После блока FROM укажите имя таблицы, из которой требуется извлечь данные.
3. Блок WHERE является необязательным. Укажите в нем условия отбора.

Запрос ***mysqli*** возвращает объект результата. Выполняется выборка строки из результатов выполненного запроса и извлекается значение строки идентификатора, которое сохраняется в переменной ***$wisherID***.

Наконец, освобождается результат ***mysqli***.

Для физического закрытия подключения необходимо освободить все ресурсы, использующие подключение. В противном случае внутренняя система подсчета ссылок PHP сохранит нижележащее подключение к базе данным открытым, даже если ***$con*** неприменимо после вызова ***mysqli_close()***.

Для *MySQL* параметр **htmlentities($_GET["user"])** используется с escape-символом для предотвращения SQL-инъекций. Рекомендуется исключить из участия в запросах *MySQL* такие строки, которые могли бы быть подвержены подобной атаке.

На данный момент блок PHP готов. Файл ***wishlist.php*** теперь выглядит следующим образом:

```
Wish List of <?php echo htmlentities($_GET["user"]) . "<br/>"; ?>
<?php
  $con = mysqli_connect("localhost", "phpuser", "phpuserpw");   if
(!$con) {
     exit('Connect Error (' . mysqli_connect_errno() . ') '
          . mysqli_connect_error());
                                                                }
  //set the default client character set   mysqli_set_charset($con,
'utf-8');
```

```
mysqli_select_db($con, "wishlist");
$user = mysqli_real_escape_string($con, htmlentities($_GET["user"]));
$wisher = mysqli_query($con, "SELECT id FROM wishers WHERE name='" . $user . "'");
if (mysqli_num_rows($wisher) < 1) { exit("The person " . htmlentities($_GET["user"]) . " is not found. Please check the spelling and try again");
}
$row = mysqli_fetch_row($wisher);
$wisherID = $row[0];
mysqli_free_result($wisher);
?>
```

Отображение таблицы желаний

В этом разделе будет добавлен код для отображения таблицы HTML желаний, связанных с автором желаний. Автор желаний определяется идентификатором, полученным в коде предыдущего раздела.

Под блоком PHP введите или вставьте следующий блок кода HTML:

```
<table border="black">
  <tr>
    <th>Item</th>
    <th>Due Date</th>
  </tr>
</table>
```

Этот код открывает таблицу, указывает цвет ее границ (черный) и определяет вид заголовка таблицы, содержащего столбцы ***Item*** и ***Due Date***. Тег ***</table>*** закрывает таблицу.

Введите следующий код блока PHP над закрывающим тегом ***</table>:***

```
<?php
$result = mysqli_query($con, "SELECT description, due_date FROM wishes WHERE wisher_id=" . $wisherID); while ($row = mysqli_fetch_array($result)) {
  echo "<tr><td>" . htmlentities($row["description"]) . "</td>";
echo "<td>" . htmlentities($row["due_date"]) . "</td></tr>\n";
}
mysqli_free_result($result);
mysqli_close($con);
?>
```

Посредством запроса ***SELECT*** желания со сроками их выполнения для указанного пользователя извлекаются в соответствии с идентификатором, который, в свою очередь, был извлечен в действии 4. Кроме того, желания и сроки их выполнения сохраняются в массиве ***$result***.

С помощью цикла отдельные элементы массива ***$result*** выводятся на экран в качестве строк таблиц, пока массив непуст.

Теги ***<tr></tr>*** формируют строки, теги ***<td></td>*** – ячейки внутри строк, а после символа ***\n*** начинается новая строка.

Функция ***htmlentities*** преобразует все символы, имеющие эквивалентные сущности HTML, в сущности HTML. Это помогает предотвратить межсетевые сценарии.

В конце функции освобождают все ресурсы (результаты ***mysqli***) и закрывают подключение к базе данных. Имейте в виду, что для физического закрытия подключения необходимо освободить все ресурсы, использующие подключение к базе данных. В противном случае внутренняя система подсчета ссылок PHP сохранит подключение к базе данным открытым, даже если подключение неприменимо после вызова ***mysqli_close()***.

Убедитесь, что названия полей базы данных введены точно так, как они указаны при создании таблицы базы данных.

Задания по практической работе:

1. Создать базу данных на PHP:

Управление библиотекой книг: Создайте базу данных для учета книг, авторов и читателей. Реализуйте функционал добавления, редактирования и удаления книг.

2. Система управления заказами: Создайте базу данных для магазина, где пользователи могут размещать заказы, просматривать их статус и управлять своими учетными записями.

3. Учебное заведение: Разработайте базу данных для управления студентами, курсами и оценками. Реализуйте возможность добавления и редактирования информации о студентах и курсах.

4. Форум: Создайте базу данных для форума с темами, сообщениями и пользователями. Реализуйте возможность регистрации пользователей и публикации сообщений.

5. Система учета сотрудников: Разработайте базу данных для учета сотрудников компании, включая информацию о должностях, зарплатах и контактной информации.

6. Услуги такси: Создайте базу данных для службы такси, включая информацию о водителях, заказах и клиентах.

7. Система управления ресторанами: Создайте базу данных, позволяющую управлять меню, заказами и отзывами клиентов.

8. Аренда автомобилей: Разработайте базу данных для системы аренды автомобилей с информацией о клиентах, автомобилях и арендах.

9. Электронный магазин: Создайте базу данных для интернет-магазина, включая категории товаров, заказы и платежи.

10. Система регистрации событий: Разработайте базу данных для учета событий с возможностью регистрации участников и просмотра списка мероприятий.

11. Управление проектами: Создайте базу данных для системы управления проектами, включая информацию о задачах, сроках и пользователях.

12. Система управления блогом: Разработайте базу данных для блога, позволяющую добавлять записи, комментарии и метки.

13. Кинотеатр: Создайте базу данных для учета сеансов кино, фильмов и отзывов зрителей.

14. Социальная сеть: Реализуйте простую базу данных для социальной сети с пользователями, постами и подписчиками.

15. Управление пациентами: Создайте базу данных для медицинского учреждения, включая информацию о пациентах, врачах и записях на прием.

16. Система управления финансами: Разработайте базу данных для учета доходов и расходов с возможностью добавления категорий.

17. Образовательная платформа: Создайте базу данных для онлайн-курсов с информацией о пользователях, курсах и заданиях.

18. Система опросов: Разработайте базу данных для создания и управления опросами, результатами и участниками.

19. Учет спортивных соревнований: Создайте базу данных для учета соревнований, участников и результатов.

20. Краудфандинговая платформа: Разработайте базу данных для учета проектов, инвесторов и финансирования.

Лабораторная работа 4. Технологии стороны клиента. Введение в JavaScript. Сценарии обработка событий.

Цель работы – научиться использовать имеющиеся в модели документа события для внесения изменений в страницу.

Наиболее часто в сценариях используется событие ***onclick***. Для того чтобы обратить внимание пользователя на определённый элемент HTML-документа, можно менять свойства этого элемента при наведении на него курсора мыши, а при снятии курсора – восстанавливать прежние значения свойств. Например, можно менять цвет или размер элемента. Попадание курсора мыши на элемент фиксируется событием ***onMouseOver***. Парное для него событие ***onMouseOut*** происходит при снятии курсора мышки с элемента. Эту пару событий удобно применять для изменения свойств элементов или замены элементов на время удержания кнопки мыши нажатой.

Реакция на событие в отдельном элементе

Так как в объектной модели объекты могут быть вложены друг в друга, то событие, происходящее в дочернем объекте, одно-временно происходит и в родительском объекте. JavaScript предос-тавляет различные способы локализации влияния события на иерархию объектов. Простейшей способ локализации (пример 1) заключается в размещении сценария в теге, на который должно воздействовать событие.

Пример 1

```
<html>
  <body>
    <P align=right ID='alfa'
     onMouseOver="document.all.alfa.align='center'
    "onMouseOut="this.align='left'">
     События onMouseOver и onMouseOut </p>
</body>
</html>
```

Страница в примере 1 состоит из одной строки, заключённой в контейнер ***<P> …</p>*.** В объектной модели страницы событие, происходящее с объектом **P**, происходит также и с родительским объектом ***BODY***. Чтобы локализовать реакцию на событие только пределами строки, т. е. объекта **P**, сценарий реакции на события помещен в тег ***<P >***.

В результате исполнения сценария изменяется положение текста ***d*** строке. Первоначально строка выровнена по правому краю окна. При попадании на неё курсора строка выравнивается по центру, а после снятия курсора выравнивается по левому краю окна. Для обращения к объекту используется коллекция *all*, которая правильно воспринимается браузерами Internet Explorer 6.0 и Mozilla Firefox 2.0. Ключевое слово **this** означает ссылку на текущий объект.

Если при наступлении события нужно произвести много действий, то удобно написать сценарий в виде функции и поместить её отдельно от элемента в специально предназначенный для сценариев контейнер ***<script> …</script>***. В примере 2 каждое из событий **onMouseOver** и **onMouseOut** вызывает два действия – выравнивание и изменение цвета текста в строке.

Пример 2

```
<html>
  <p  align=right ID='alfa' onMouseOver="M_Over()"
      onMouseOut="M_Out()">
Событие onMouseOver</p>
    <script>
     function M_Over()
     {
        document.all.alfa.align='center'
        document.all.alfa.style.color='FF00FF'
      }
     function M_Out()
    {
      document.all.alfa.align='left'
      document.all.alfa.style.color='0000FF'
     }
    </script>
</html>
```

Задание 1

Разработайие HTML-документ, отображающийся в окне браузера в виде следующих четырёх строк:

1. Пять событий с мышкой
2. Щёлкните по мне мышкой
3. На этом тексте нажмите, подержите и отпустите левую кнопку мышки
4. Медленно проведите курсором мышки по этой надписи

Первая строка – заголовок страницы. Вторая строка меняется при щелчке мышкой следующим образом:

– шрифт увеличивается до 48 pt;

– цвет шрифта меняется на белый;– цвет фона меняется на голубой.

Повторный щелчок мышкой возвращает вторую строку к первоначальному виду.

Фон третьей строки меняется, когда курсор мышки находится на ней и нажимается или отпускается левая кнопка мышки. При нажатии фон становится зелёным, а при отпускании – жёлтым.

При попадании курсора мышки на четвёртую строку её фон становится красным, а при снятии – голубым.

Фиксация события в родительском элементе

Если реакцию на какое-либо событие требуют несколько элементов, расположенных на странице, то можно вызвать функцию для обработки этого события только в родительском элементе. В функции определяется, на каком элементе произошло событие, и выполняются соответствующие действия. Удобство такого подхода состоит в том, что весь алгоритм преобразований находится в одном месте, а недостаток – в сложности самой функции. Рассмотрим сценарий (пример 3), в котором для изменения свойств любого из трёх объектов, находящихся в окне браузера, служит одна функция.

Пример 3

```
<html>
   <head>
     <title>Реакция на событие</title>
     <meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
      <style>H1 {color:FF00FF}
        #k1{position:absolute; left:50;top:200;width:300;
height:100;background-color:blue}
        #k2{position:relative; left:50;top:25; width:200;
height:50;background-color:yellow}
      </style>
    </head>
    <BODY ID="B" onclick="rodEl(event)">
      <H1 ID="HH" >ЦВЕТ  </H1>
      <DIV ID="k1" >
      <DIV ID="k2" > </div>
     </div>
    <SCRIPT>
/*    Функция запускается при щелчке мышкой по любой точке
документа */
      flag=0;
      function rodEl(evt)
      {
       var e = evt || window.event;  //e -это объект event
       var elem = e.target || e.srcElement; //elem - элемент (объект),
        // на котором произошло событие
       id1=elem.id
       z1=document.getElementById(id1)
switch(id1)
      {
```

```
                case "k1": //Изменение цвета внешнего прямоугольника
z=z1.style.backgroundColor          if(z!="red")z="red"          else
z="green"          z1.style.backgroundColor=z
                break
                case "k2":   //Изменение цвета внутреннего прямоугольника
z=z1.style.backgroundColor
               if(z!='rgb(0, 255, 255)'){z='rgb(0, 255, 255)'}
                  else{z='rgb(0, 255, 0)'}
z1.style.backgroundColor=z
               break
               case "B": //Изменение цвета заднего плана документа
z=z1.style.backgroundColor
                if(z!='rgb(190, 190, 190)'){z='rgb(190, 190, 190)'}
else {z='rgb(0, 190, 190)'}          z1.style.backgroundColor=z
break
             case "HH": //Изменение цвета слова "Цвет"
if(flag==0)
            {  document.getElementById(id1).style.color='rgb(170,0,170)';
flag=1;

}      else
           {   document.getElementById(id1).style.color='rgb(0,255,170)';
flag=0;
           }
        }
       }
       </SCRIPT>
       </BODY>
       </HTML>
```

В примере 3 родительским по отношению к элементу *H1* и двум элементам **DIV** является элемент **BODY**. Поэтому в теге **<*BODY*>** вызывается функция ***rodEl()***, служащая для обработки события **onclick**.

В момент наступления события вся информация о нём запоминается в объекте ***event***. Этот объект по-разному описывается в стандарте W3C и в браузере *Internet Explorer*. Новые версии *Internet Explorer* поддерживают и стандарт W3C.

В стандарте W3C объект **event** передаётся в функцию в качестве параметра, а для обращения к объекту, на котором произошло событие, служит свойство ***event.target***.

В *Internet Explorer* объект ***window.event*** – глобальный, и поэтому передавать его в функцию в виде параметра не нужно. Для обращения к

объекту, на котором произошло событие, в *Internet Explorer* служит свойство ***window.event.srcElement.***

В примере 3 первые две строчки тела функции ***rodEl()*** служат для кроссплатформенного (в любом браузере) обращения к объекту, на котором произошло событие.

В следующих строках функции ***rodEl()*** сначала определяется **Id** элемента, по которому пользователь щёлкнул мышкой, а затем с помощью оператора **Switch** делается переход к изменению свойств указанного элемента.

Задание 2

Создайте страницу с любым изображением и подписью под ним. При щелчке по подписи надпись должна менять свой цвет. Щелчок по изображению должен вызывать замену изображения и подписи. Функция для обработки события должна вызываться из родительского по отношению к изображению и подписи объекта.

Предотвращение всплывания события. Свойство cancelBubble

В примере 3 рассматривалась простая страница с небольшим количеством элементов, но даже в таком простом случае функция реакции на событие получилась сложной. Проще для каждого элемента написать свою функцию обработки события, а распространение события вверх по дереву иерархической структуры от «детей» к «родителям» (в этом случае говорят о всплывании события) блокировать с помощь специально для этого предназначенного свойства **cancelBubble** объекта event. Изменим пример 3 так, чтобы для реакции на щелчок по каждому из четырёх элементов страницы служила своя функция.

Пример 4

```
<HTML><HEAD><TITLE>Реакция на событие</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<STYLE>H1 {color:#ff00ff}
      #k1{position:absolute; left:50;top:200; width:300; height:100;background-color:blue}
#k2{position:relative; left:50;top:25;  width:200; height:50;background-color:yellow}
</STYLE></HEAD>
<BODY ID="B" bgcolor="AAAAAA" onclick="rodEl()" style="height:600px">
      <H1 ID="HH" onclick="H_l()">ЦВЕТ</H1>
      <DIV ID="k1" onclick="D_l()">
            <DIV ID="k2" style="background-color:yellow"
onclick="D_2(this)"></div>
      </div>
```

```
</BODY>
<SCRIPT>
/*      Функция запускается при щелчке мышкой по
        любой точке документа */
function rodEl()
{   //Изменение цвета заднего плана документа var
z=document.all.B.bgColor
if(z!="#777777"){z="#777777"}        else{z="AAAAAA"}
    document.all.B.bgColor =z
}
function D_l()
 {      //Изменение цвета внешнего прямоугольника
  var  z=document.all.k1.style.backgroundColor
       if(z!="red") z="red"
else z="green"
       document.all.k1.style.backgroundColor=z
}
/*function D_2()
{       //Изменение цвета внутреннего прямоугольника
var z=document.all.k2.style.backgroundColor        if(z=="#ff00ff" ||
z=='rgb(255, 0, 255)'){z="#00ffff"}        else{z="ff00ff"}
       document.all.k2.style.backgroundColor=z
}*/
function D_2(thi)
{       //Изменение цвета внутреннего прямоугольника
var z=thi.style.backgroundColor
       if(z=="#ff00ff" || z=='rgb(255, 0, 255)'){z="#00ffff"}
else{z="ff00ff"}
       thi.style.backgroundColor=z
}
function H_l()
{       //Изменение цвета слова "Цвет"     var
z=document.all.HH.style.color     //alert("1.
z="+document.all.HH.style.color)
    if(z=="#aa00aa" || z=='rgb(170, 0, 170)')
       {       z="#00ffff"
       //alert("2. z="+z)
       }
       else{z="#aa00aa"}
document.all.HH.style.color=z
}
</SCRIPT></HTML>
```

Скопируйте в свой каталог и просмотрите пример 4 в браузере Internet Explorer. Щёлкните по слову *ЦВЕТ*. Изменится не только цвет слова, но и цвет фона документа, так как после щелчка сначала выполнится функция ***H_1()***, а затем событие всплывёт к родительскому элементу ***BODY*** и выполнится функция ***rodE1().*** При щелчке по внутреннему прямоугольнику будут меняться цвета обоих прямоугольников и фона документа. Щелчок по внешнему прямоугольнику изменит цвет этого прямоугольника и цвет фона документа.

Задание 3

Чтобы предотвратить всплывание события в примере 4, вставьте в начало всех функций, кроме первой, оператор window.event.cancelBubble=true

Программирование на JavaScript. Работа с формами

Цель работы – научиться использовать формы для ввода данных пользователем, проверки введенных данных и передачи их на web-сервер.

Форма служит для ввода пользователем через окно браузера данных и передачи их на web-сервер. Форма состоит из контейнера ***<FORM> …</FORM>*** и помещённых в него тегов (элементов) **<INPUT>, <SELECT>** *и* **<TEXTAREA>**.

Проверка данных перед отправкой на сервер

Для уменьшения нагрузки на сеть и web-сервер можно проверять введённые пользователем данные в браузере с помощью скрипта на JavaScript. Если в данных обнаружится ошибки, то пользователю предоставляется возможность их исправить. Введенные правильно данные отправляются на web-сервер. Использование сценария для управления формой демонстрируется в примере 1.

Пример 1

```
<HTML>
<HEAD><TITLE>Первая страница</title></head>
<H2>Представьтесь, пожалуйста</h2>
<FORM name=F1 METHOD="POST" ACTION="">
Имя....
<INPUT TYPE="text" NAME="name"><BR>
Возраст <INPUT TYPE="text" NAME="age">
<P> <INPUT TYPE="submit" VALUE="ВВОД" onclick="Proverka()">
</FORM>
<SCRIPT>
function Proverka()
{ im=document.F1.name.value vozr=document.forms[0].elements[1].value
st=""
if(im == "") st="имя\n" if(vozr == "")
st+="возраст" if(st == "")
document.F1.action="Prim3_1.php" else //отмена
передачи на web-сервер
{   str="Введите:\n "+st
alert(str)
   return=false
}
}
</script>
</html>
<HTML>
<HEAD><TITLE>Вторая страница</title></head>
<BODY>
```

```
<H3>П Р И В Е Т С Т В И Е</h3>
<?php
$imja=$_POST["name"]; //приём параметров из формы
$age=$_POST["age"]; $x="Здравствуйте!
$imja.";
if($age>50) echo "$x Вы включены в старшую группу.";
elseif($age>30)echo "$x Вы включеныв группу среднего возраста."; else
echo"$x Вы относитесь к молодёжной группе.";
?>
<P>
<A href="Prim3_1.html">Возврат </a>
</body>
</html>
```

Пример 1 состоит из двух страниц. Первая страница служит для ввода пользователем данных и их проверки с помощью скрипта, написанного на JavaScript. Если данные введены правильно, то они отправляются на web-сервер. На web-сервере полученные данные обрабатываются скриптом, написанным на языке PHP, формируется и пересылается на браузер пользователя новая страница.

В скрипте, написанном на JavaScript, для доступа к данным в форме используются имена и индексы элементов формы. Для задания адреса (URL) страницы, содержащей PHP-скрипт, используется свойство *action* объекта *Form*.

Получение данных из всплывающего списка

Иногда можно полностью решать задачу ведения диалога с пользователем средствами JavaScript, не обращаясь к web-серверу. В примере 2 пользователь вводит код цвета в модели RGB и выбирает из списка название цвета. После нажатия кнопки ***Ввод*** на экран выводятся окрашенные в выбранные цвета код и название цвета.

Пример 2

```
<html>
<HEAD><TITLE>СКИПТ SELECT </title> </head>
<body>
<!--Пример выбора и выводана экранэлемента списка Select -->
<SCRIPT>
function select_()
{ a=document.all.Kod.value //код цвета
b= document.all.Gor.selectedIndex
//номервыбранного элемента// спискаselect c =
document.all.Gor.options[b].text//текст элементасписка d =
document.all.Gor.options[b].value//передаваемое из формы
//значение<option value= red>
```

```
e="<FONT size= 7 color="+a+">"+a+"</font>"//трансляция и
document.all.alfa.innerHTML= e// вывод на экран HTML-строки e=
"<FONT size = 7 color= "+d+">"+c+"</font>"
document.all.beta.innerHTML= e
}
</script>
<H2>Подбор оттенков цвета</h2>
В первое поле нужно ввести шестнадцатеричный код цвета.<BR>
Например,красный цвет имеет код FF0000.
<BR>Из списка во втором поле выбирается для сравнения
<BR> один из основных цветов(красный, зелёный,синий)
<P>Введитекод цвета
<input TYPE= "text"name="Kod"> <P>Выберите цвет
<SELECT NAME= "Gor">
<option value="red">Красный </option>
<option value="yellow">Жёлтый </option>
<option value="maroon">каштановый</option>
<option value=green">Зеленый</option>
<option value="blue">Синий
</select>
<P> <BUTTON onclick="select_()">Выполнить </button>
<P><B ID="alfa"></b>
<br> <B ID = "beta"></b>
</body>
</html>
```

В примере 2 нет необходимости использовать контейнер ***<FORM> …</form>***, так как на web-сервер ничего не передаётся. Для вывода на экран кода и названия цвета используется свойство *innerHTML*. Строго говоря, *innerHTML* следовало бы называть не свойством, а методом. Рассмотрим применение этого свойства на примере вывода на экран введённого пользователем кода цвета.

Пусть пользователь ввёл код красного цвета *a=FF0000*. В результате выполнения оператора e="<FONT size= 7 color="+a+">"+a+"" сформируется строка e="<FONT size=7 color=FF0000>FF0000"

В результате выполнения оператора document.all.alfa.innerHTML= e в элемент

<B ID="alfa"> вставится

значение переменной **e**:

<B ID="alfa"><FONT size=7 color=FF0000>FF0000

Браузер выполнит преобразованный оператор языка *HTML*, т. е. выведет на экран надпись **FF0000** красного цвета.

Таким образом, с помощью свойства ***innerHTML*** можно на стороне браузера программным путём вносить изменения в HTMLдокумент (страницу сайта).

Задание 1

Создайте сайт из двух страниц.

Первая страница имеет заголовок «*Заказ мебели».*

На странице расположены два поля со списками (теги **<*SELECT*>**), поле ***(<INPUT>)*** и кнопка ***(<SUBMIT>)***. Из первого поля со списком пользователь выбирает изделие (шкаф, стол, сервант и т. д.). Из второго поля со списком пользователь выбирает материал (дуб, орех, бук). В третье поле нужно ввести количество заказываемых изделий.

После ввода данных необходимо проверить, все ли данные введены. Если обнаружена ошибка, то нужно вывести сообщение и предложить её исправить. Правильно введённые данные нужно отправить на web-сервер. Вторая страница содержит написанный на PHP скрипт, с помощью которого формируется следующее сообщение:

Ваш заказ принят

Заказано изделие – *название заказанного изделия*

Материал – *заказанный материал*

Количество – *заказанное количество*

Если в форме нужно заполнить много полей, то пользователю удобнее получать сообщения об ошибках сразу после окончания ввода данных в очередное поле, т. е. после нажатия клавиши **Tab** или клавиши со стрелкой. Для немедленной проверки введённых данных используйте событие ***onchange*:**

<INPUT TYPE ="text" SIZE=6 onchange="arg(this)">

Функция, вызываемая событием **onchange**, имеет примерно такую структуру: function arg(fld)

```
{ x=fld.value //введённое
значение   if(x. . .) //условия проверки
{ alert("Сообщение об ошибке");
        fld.focus();
fld.select()
    }
}
```

Методы **focus()** и **select()** служат для возвращения курсора мышки в поле ввода и выделения ошибочных данных. Эти методы без использования специальных приёмов правильно работают только в браузере Mozilla.

Задание 2

Создайте web-страницу (рис. 20) для вычисления тригонометрических функций. Вводимые пользователем данные должны проверяться немедленно после ввода и после нажатия кнопки ***Вычислить***.

Рис. 20. Web-страница для задания 2

Не забудьте перевести градусы в радианы.

Название тригонометрической функции можно передавать как параметр тега:

<option value="sin". . . >

Сформируйте текстовую строку вида

"Math." + имя_ф + "(" + знач_аргумета + ")" // имя_ф – sin,cos или tan

Затем воспользуйтесь функцией *eval(строка),* которая выполняет выражение, хранящееся в строке.

Разработка сайта на языке PHP.

Цель работы – научиться создавать на PHP функции для проверки допустимости данных и добавления их в базу данных.

В этой лабораторной работе рассматривается расширение функционала приложения добавлением функции *Create a New Wisher*.

Реализация затрагивает файл *index.php*, при этом будут созданы два новых файла с именами ***createNewWisher.php*** и ***editWishList.php***.

Данный пример состоит из трех действий:

1. Пользователь открывает файл *index.php* титульной страницы и щелкает ссылку для регистрации.
2. Пользователь переходит на страницу *createNewWisher.php* для создания нового автора желания.
3. После создания нового автора желания пользователь переключается на *editWishList.php*, где для него можно создать список желаний.

Добавление ссылки для начала создания нового автора желания

Откройте файл index.php. Добавьте пустую строку под закрывающим тегом ***</form>***. Введите в эту пустую строку следующий блок кода:

```
<br>Still don't have a wish list?!
<a href="createNewWisher.php">Create now</a>
```

Создание новых web-страниц PHP

Создайте в исходных файлах проекта две новые web-страницы PHP: ***createNewWisher.php*** и ***editWishList.php***.

В ***editWishList.php*** добавьте текст *Hello!* к тексту в формате HTML, а в остальном оставьте всё как было. Этот файл необходим в качестве объекта ссылки для ***createNewWisher.php.***

Добавление формы HTML для ввода данных нового автора желания

Введите или вставьте следующий блок *HTML* в строку ***createNewWisher.php*** под блоком *PHP*:

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
   Welcome!<br>
  <form action="createNewWisher.php" method="POST">
Your name: <input type="text" name="user"/><br/>
        Password: <input type="password" name="password"/><br/>
Please confirm your password: <input type="password" name="password2"/>
<br/>      <input type="submit" value="Register"/>
</form>
    </body>
</html>
```

Тип ***password*** – это специальный тип текстового поля, в котором символы заменяются звездочками. Код представляет собой форму HTML, позволяющую ввести имя и пароль нового автора пожелания в текстовые поля. При нажатии кнопки *Register* введенные данные передаются для проверки допустимости на ту же страницу – *createNewWisher.php*. Предупреждения от средства проверки *HTML* можно проигнорировать.

Проверка допустимости данных и добавление их в базу данных

В этом разделе мы добавим код PHP к *createNewWisher.php*. Добавьте этот код к блоку *PHP* на верху файла. Блок *PHP* должен находиться над кодом *HTML* ***all***, пустыми строками или пробелами. Расположение блока кода *PHP* является важным для правильного функционирования оператора переадресации. Внутри блока *PHP* введите или вставьте в указанном порядке блоки кода, описанные ниже в данном разделе.

Необходимо инициализировать переменные. Первая группа переменных осуществляет передачу параметров доступа к базе данных, а другая группа переменных используется в работе кода PHP.

Добавьте следующий код для проверки допустимости данных:

```
3.      /** database connection credentials */
$dbHost="localhost"; //on MySql
4.      $dbXeHost="localhost/XE";
$dbUsername="phpuser";
$dbPassword="phpuserpw";
/** other variables */
$userNameIsUnique = true;
$passwordIsValid = true;
$userIsEmpty = false;
$passwordIsEmpty = false;
$password2IsEmpty = false;
```

Под переменными следует добавить блок ***if***. Параметр блока ***if*** выполняет проверку того, что страница была запрошена из нее самой посредством метода POST. Если это не так, дальнейшие проверки допустимости не выполняются, и на экран выводится страница с пустыми полями, как описано выше.

```
/** Check that the page was requested from itself via the POST method. */ if
($_SERVER["REQUEST_METHOD"] == "POST")
{
}
```

Внутри фигурных скобок блока ***if*** добавьте другой блок ***if*** , позволяющий проверить, ввел ли пользователь имя автора желания в поле. Если текстовое поле ***user*** является пустым, значение ***$userIsEmpty*** меняется на ***true***.

```
/** Check that the page was requested from itself via the POST method. */ if
($_SERVER["REQUEST_METHOD"] == "POST") {
** Check whether the user has filled in the wisher's name in the
text field "user" */    if
($_POST["user"]=="")
 {
$userIsEmpty = true;
}
}
```

Добавьте код, устанавливающий подключение к базе данных. Если установить подключение невозможно, то выводится ошибка MySQL:

```
/** Check that the page was requested from itself via the POST method. */
if ($_SERVER["REQUEST_METHOD"] == "POST") {
/** Check whether the user has filled in the wisher's name in the text field
"user" */       if ($_POST["user"]=="") {
            $userIsEmpty = true;
        }
/** Create database connection */
```

```
$con = mysqli_connect($dbHost, $dbUsername, $dbPassword); if
(!$con)
 {       exit('Connect Error (' . mysqli_connect_errno() . ')
'mysqli_connect_error());
  }
//set the default client character set     mysqli_set_charset($con, 'utf8');
}
```

Добавьте код, позволяющий проверить, существует ли пользователь, имя которого соответствует указанному в поле ***user.*** Эта задача выполняется путем поиска идентификационного номера автора пожелания в соответствии с именем, указанным в поле ***user***. Если такой номер существует, значение ***$userNameIsUnique*** меняется на ***false***.

```
/** Check that the page was requested from itself via the POST
method. */ if ($_SERVER["REQUEST_METHOD"] == "POST") {
/** Check whether the user has filled in the wisher's name in the text field
"user" */
  if ($_POST["user"]=="")
{
    $userIsEmpty = true;
  }
/** Create database connection */
  $con = mysqli_connect($dbHost, $dbUsername, $dbPassword);     if
(!$con)
 { exit('Connect Error (' . mysqli_connect_errno() . ') '                .
mysqli_connect_error());
  }
 /**set the default client character set */     mysqli_set_charset($con,
'utf-8');
 /** Check whether a user whose name matches the "user" field already
exists */
  mysqli_select_db($con, "wishlist");
  $user = mysqli_real_escape_string($con, $_POST["user"]);
  $wisher = mysqli_query($con, "SELECT id FROM wishers WHERE
name='".$user."'");
  $wisherIDnum=mysqli_num_rows($wisher);
  if ($wisherIDnum) {
    $userNameIsUnique = false;
  }
}
```

После кода, проверяющего уникальность пользователя, добавьте серию блоков ***if*** , проверяющих, правильно ли пользователь ввел и подтвердил пароль. Код выполняет проверку того, что поля *Password* (переменная

password) и *Confirm Password* (***переменная password2***) заполнены и идентичны друг другу. В противном случае значения соответствующих логических переменных также изменяются:

```
        if
($_POST["password"]=="") {
$passwordIsEmpty = true;
        }
        if
($_POST["password2"]=="")  {
$password2IsEmpty = true;
        }
        if ($_POST["password"]!=$_POST["password2"]) {    $passwordIsValid =
false;
        }
```

Завершите блок if ($_SERVER['REQUEST_METHOD'] == *POST*), добавив код, вставляющий новую запись в базу данных *Wishers*:

```
        /** Check that the page was requested from itself via the POST method. */
        if ($_SERVER['REQUEST_METHOD'] == "POST") {
           /** Check whether the user has filled in the wisher's name in the text field
"user" */
           if ($_POST['user'] == "") {
              $userIsEmpty = true;
           }
           /** Create database connection */
           $con = mysqli_connect($dbHost, $dbUsername, $dbPassword);     if
(!$con) {
              exit('Connect Error (' . mysqli_connect_errno() . ') '
                    . mysqli_connect_error());
           }
           //set the default client character set     mysqli_set_charset($con,
'utf-8');
           /** Check whether a user whose name matches the "user" field already
exists */
           mysqli_select_db($con, "wishlist");
           $user = mysqli_real_escape_string($con, $_POST['user']);
           $wisher = mysqli_query($con, "SELECT id FROM wishers WHERE
name='".$user."'");
           $wisherIDnum=mysqli_num_rows($wisher);
           if ($wisherIDnum) {
              $userNameIsUnique = false;
           }
           /** Check whether a password was entered and confirmed correctly
```

```
*/
   if ($_POST['password'] == "") {
      $passwordIsEmpty = true;
   }
   if ($_POST['password2'] == "") {
      $password2IsEmpty = true;
   }
   if ($_POST['password'] != $_POST['password2']) {
      $passwordIsValid = false;
   }
   /** Check whether the boolean values show that the input data was
validated successfully.
   *     If the data was validated successfully, add it as a new entry in the
"wishers" database.
   *     After adding the new entry, close the connection and redirect the
application to editWishList.php.
    */
   if (!$userIsEmpty && $userNameIsUnique && !$passwordIsEmpty
   && !$password2IsEmpty && $passwordIsValid) {
      $password = mysqli_real_escape_string($con, $_POST['password']);
      mysqli_select_db($con, "wishlist");
      mysqli_query($con, "INSERT wishers (name, password) VALUES
   ('" . $user . "', '" . $password . "')");
mysqli_free_result($wisher);     mysqli_close($con);
      header('Location: editWishList.php');
exit;
   }
}
```

В соответствии с кодом выполняется проверка того, что указано уникальное имя пользователя и что пароль введен и подтвержден правильно. Если эти условия выполнены, код извлекает значения *user* и *password* из формы HTML и вставляет их соответственно в столбцы ***Name*** и ***Password***, относящиеся к новой строке в базе данных *Wishers*. После создания строки код закрывает подключение к базе данных и переадресует приложение на страницу *editWishList.php*.

Отображение сообщений об ошибках в форме ввода

Перейдем к реализации вывода сообщений об ошибках при неверно введенных данных. Реализация основывается на проверках допустимости и изменении значений логических переменных.

Введите следующий блок кода PHP в форме ввода HTML непосредственно под именем пользователя:

```
Welcome!<br>
```

```
<form action="createNewWisher.php" method="POST">
   Your name: <input type="text" name="user"/><br/>
<?php
   if ($userIsEmpty) { echo ("Enter your
name, please!"); echo ("<br/>");
}
if (!$userNameIsUnique) {
echo ("The person already exists. Please check the spelling and try again");
echo ("<br/>");
}
?>
```

Введите следующий блок кода PHP в форме ввода HTML под кодом для ввода пароля:

```
Password: <input type="password" name="password"/><br/>
<?php
  if ($passwordIsEmpty) { echo ("Enter the
password, please!"); echo ("<br/>");
}
?>
```

Введите следующий блок кода PHP в форме ввода HTML под кодом для подтверждения пароля:

```
Please confirm your password: <input type="password"
name="password2"/><br/>
<?php
if ($password2IsEmpty) {
echo ("Confirm your password, please");
echo ("<br/>");
}
if (!$password2IsEmpty && !$passwordIsValid) { echo  ("The
passwords do not match!");
echo ("<br/>");
}
?>
```

Тестирование функциональных возможностей по созданию нового пользователя Create New Wisher

Запустите приложение. Откроется страница-указатель.

На странице-указателе щелкните ссылку рядом с текстом *Still don't have a wish list?* Откроется форма.

Оставьте поля пустыми и нажмите кнопку *Register* («Зарегистрировать»). На экране появится сообщение об ошибке.

Введите имя зарегистрированного пользователя, например, *Tom* в поле *Your name*, внимательно заполните другие поля и нажмите кнопку *Register*. На экране появится сообщение об ошибке.

Заполните поля *Password* и *Please confirm your password* различными значениями и нажмите кнопку *Register*. На экране появится сообщение об ошибке.

Введите *Bob* в поле *Your name*, укажите в полях пароля один и тот же пароль и нажмите кнопку *Register*. Откроется пустая страница, однако переадресация осуществляется правильно, поскольку URL-адрес заканчивается текстом *editWishList.php*:

Проверьте, что данные сохранены в базе данных, путем перехода к разделу *Wishers* в окне *Services*, расположенном под узлом *wislist1*, и выбора *View Data* в контекстном меню.

Задания по практической работе:

Создать сайт а базой данных на тему:

1. База данных продаж продуктового киоска
2. Управление клиентами торговой организации
3. Маркетинговая база книжного магазина
4. Финансовый контроль универмага
5. Система инвентаризации учебного заведения
6. Аналитика бизнес процессов компьютерного салона
7. Торговая платформа интернет-магазина
8. Управление проектами строительного салона
9. База данных поставщиков магазина стройматериалов
10. Клиентский реестр фирмы «Землемер»
11. Актуальные тренды рынка программного обеспечения
12. Система отчетности частного предпринимателя (электрика)
13. База данных сотрудников школы
14. Система учета продаж газетного киоска
15. ГенерацияЛидовDB
16. Анализ конкурентов строительного бизнеса
17. Отчетность по финансам частного предпринимателя (сантехника)
18. База идеи для бизнеса
19. CRM-Система
20. Прогнозирование продаж частного предпринимателя (одежда)

ЗАКЛЮЧЕНИЕ

Вы закончили изучение основ web-программирования по нашему учебному пособию, разработанному по модели элитного обучения. Надеемся, что не только изучили теоретический материал, но и выполнили цикл практических работ, следовательно, теперь имеете представление об общих принципах построения интернет-приложений. Если решили серьезно заниматься интернет-разработкой, то для вас это только начало пути.

В рамках одного учебного пособия невозможно продемонстрировать весь арсенал современных средств для проектирования и создания интернет-приложений на стороне клиента и на стороне сервера.

Вы должны понимать, что языки и технологии web-программирования постоянно развиваются и меняются, поэтому придется всё время продолжать осваивать их самостоятельно. Существует огромное количество профессиональных сайтов и форумов, на которых размещены справочные материалы по различным языкам web-программирования и технологиям разработки web-приложений, которыми активно пользуются и студенты, и профессиональные разработчики при решении конкретных задач.

Мы желаем Вам успехов в создании собственных интернет-приложений!

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шабашов, В. Я. Организация доступа к данным из PHP-приложений для различных СУБД: учеб. пособие по дисциплине «Webпрограммирование» / В. Я. Шабашов. – М.; Берлин: Директ-Медиа, 2019. – 121 с.

2. Основы работы в Web-среде: лабораторный практикум / авт.-сост. С. В. Говорова; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – Ставрополь: СКФУ, 2017. – 160 с.

3.Аникина Е.И. ПРОФЕССИОНАЛЬНАЯ ЭТИКА IT-СПЕЦИАЛИСТОВ.-Курск.- 2018.- С.176

4. Аникина Е.И. Компьютерная и информационная этика.- Saarbrücken.- 2015.-С. 68.

5. Аникина Е.И. Информационные технологии: этические аспекты.- Saarbrücken.- 2016.-С. 220.

6. Anikina E.I. DEVELOPMENT OF DATABASE DRIVEN WEB-APPLICATION.-Saarbrücken, 2017.-С.152.

7. Аникина Е.И. МОБИЛЬНЫЕ ТЕХНОЛОГИИ BYOD: ТЕНДЕНЦИИ РАЗВИТИЯ И ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ В ВЫСШЕМ ОБРАЗОВАНИИ// Известия Юго-Западного государственного университета. Серия: Лингвистика и педагогика.2019.-Т.9.-№3.- С.132-141.

8. Аникина Е.И., Бабков А.С., Малышев А.В.АВТОМАТИЗАЦИЯ ФУНКЦИЙ ДЕКАНАТА В ЭЛЕКТРОННОЙ ИНФОРМАЦИОННО-ОБРАЗОВАТЕЛЬНОЙ СРЕДЕ ЮЗГУ//

Известия Юго-Западного государственного университета.-2017.-№ 6 (75).-С.44-50.

9. Диков, А. В. Веб-технологии HTML и CSS: учеб. пособие / А. В. Диков. – 2-е изд. – М.: Директ-Медиа, 2012. – 78 с.

Редактор *С. П. Тарасова*