МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

WEB-ТЕХНОЛОГИИ

Методические рекомендации к лабораторным работам для студентов направления подготовки 01.03.04 «Прикладная математика» дневной формы обучения

Могилев 2023

УДК 004.4
ББК 32.973-018
В26

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «17» октября 2023 г., протокол № 3

Составители: ст. преподаватель Н. В. Выговская;
ст. преподаватель Д. А. Денисевич

Рецензент канд. техн. наук, доц. С. К. Крутолевич

Методические рекомендации предназначены к лабораторным работам для студентов направления подготовки 01.03.04 «Прикладная математика» дневной формы обучения.

Учебное издание

WEB-ТЕХНОЛОГИИ

Ответственный за выпуск А. И. Якимов

Корректор А. А. Подошевко

Компьютерная верстка Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Цель методических рекомендаций к лабораторным работам по дисциплине «WEB-технологии» заключается в закреплении студентами практических навыков создания и сопровождения программных систем современных ЭВМ с доступом через сеть Интернет на серверном языке PHP.

Дисциплина «WEB-технологии» является неотъемлемой частью современных знаний и связана с такими дисциплинами, как «Базы данных» и «Объектно ориентированное программирование».

Выполнение заданий позволит студентам выработать практические навыки разработки сайтов и приложений для сети Интернет. Полученные при изучении дисциплины знания и навыки будут востребованы при практической разработке серверных и клиентских приложений, работающих в сети Интернет и станут востребованными при подготовке выпускной квалификационной работы.

В процессе выполнения лабораторной работы студенты изучают теоретический материал, методы решения задач, выполняют индивидуальное задание и оформляют отчет.

В отчете указывают: название и цель лабораторной работы; структуру и листинг электронных документов; анализ полученных результатов и выводы. В отчете можно привести ответы на наиболее сложные вопросы, приведенные в конце каждой работы.

1 Лабораторная работа № 1. Создание простой программы

Цель работы: получение навыков работы с PHP.

Порядок выполнения работы.
1 Изучить теоретические сведения.
2 Выполнить задание к лабораторной работе в соответствии с вариантом.
3 Оформить отчет.

Содержание отчета:
– цель работы;
– постановка задачи;
– текст программы.

Теоретические сведения

Что нужно, чтобы запустить PHP-скрипт?

Программирование на PHP отличается от программирования на других языках достаточно сложной системой настройки среды. Первым шагом в освоении Web-технологий является воспроизведение рабочей среды на компьютере разработчика. Для этого потребуется запустить и настроить несколько программных компонентов.

Доступность Web-приложений и HTML-файлов в сети Интернет обеспечивается Web-сервером, который по протоколу HTTP выдает их любому клиенту, правильно оформившему запрос. Наиболее популярным сервером, используемым совместно с PHP, долгие годы остается Web-сервер Apache.

Интерпретатор PHP представляет собой либо внешнюю CGI-программу, либо динамическую библиотеку, которую необходимо подключить к Web-серверу, чтобы вместо кода PHP-скриптов клиенту выдавались результаты его выполнения. Ситуация осложняется тем, что к PHP-интерпретатору могут подключаться различные расширения, также оформленные в виде динамических библиотек. Для большинства современных приложений требуется довольно много внешних расширений, поэтому без ручного редактирования конфигурационных файлов довольно трудно обойтись.

Практически ни одна крупная программа не обходится без использования базы данных. Традиционно совместно с Web-сервером Apache и интерпретатором PHP используется СУБД MySQL.

Так как и язык программирования PHP, и Web-сервер Apache, и MySQL-сервер первоначально разработаны для UNIX-подобных операционных систем, их настройка и администрирование сводятся к редактированию конфигурационных файлов и работе в командной строке. Такой подход часто сбивает с толку программистов, не имеющих опыта работы в UNIX.

Можно ли обойтись без утомительной настройки серверов и PHP?

Конфигурирование серверов и PHP, настройка режима их работы являются важными разделами, позволяющими увереннее чувствовать себя Web-разработчикам, быстро осуществлять локализацию ошибок, попыток взлома, настраивать наиболее эффективную работу Web-приложений. Однако даже локальная настройка Web-серверов, а тем более настройка их для работы в сети Интернет, является отдельной областью деятельности, требующей зачастую не меньшей отдачи, чем программирование. Поэтому Web-разработчики зачастую отказываются от изучения конфигурирования серверов и их взаимодействия, оставляя эту область системным администраторам. Чтобы не завязнуть в многочисленных настройках серверов, правилах их связывания друг с другом, Web-разработчики прибегают к готовым пакетам, где вся настройка выполнена профессиональными администраторами. Такой пакет остается только загрузить и установить, после чего можно сразу приступить к работе с языком программирования. Наиболее популярными на сегодняшний день пакетами, объединяющими интерпретатор PHP, Web-сереер Apache и СУБД MySQL, являются Denwer, MAMP и OpenServer. Все эти пакеты находятся в свободном доступе в сети Интернет.

Традиционно тестирование любой программы PHP начинается с фразы «Hello world!».

Рассмотрим чуть более сложный пример.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Простой сценарий на PHP</title>
</head>

<body>
<h1>Здравствуйте!</h1>
<?php
// Вычисляем текущую дату в формате "день.месяц год"
$dat=date("d.m y");
// Вычисляем текущее время
$tm=date("h:i:s");
# Выводим их
echo "Текущая дата: $dat года<br>\n";
echo "Текущее время: $tm<br>\n";
# Выводим цифры
echo "А вот квадраты и кубы первых 5 натуральных чисел:<br>\n";
for($i=1; $i<=5; $i++)
{ echo "<li>$i в квадрате = ".($i*$i);
echo ", $i в кубе = ".($i*$i*$i)."\n";
}
?>
```

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

Задания

1 Создайте файл 1-1.php, содержащий пять разных переменных, присвойте переменным значения разного типа. Используя gettype(), выведите тип каждой переменной.

2 Создайте файл 1-2.php, содержащий две переменные числового типа. Произведите над переменными произвольное арифметическое действие и выведите его результат.

3 Создайте файл 1-3.php, содержащий две переменные строкового типа. Инициализируйте переменные произвольным текстом. С помощью конкатенации объедините содержимое переменных и выведите результат.

4 Создайте файл 1-4.php, содержащий две переменные с одинаковым типом значений. Используя тернарный оператор сравнения, проведите исследование на возвращаемые результаты.

Контрольные вопросы

1 Что нужно, чтобы запустить PHP-скрипт?

2 Оформление скриптов PHP. Приведите примеры.

3 Что такое оператор echo на языке PHP?

2 Лабораторная работа № 2. Использование основных операторов PHP

Цель работы: изучение основных операторов PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения об операторах и управляющих конструкциях PHP по конспекту.

2 Выполнить задание к лабораторной работе в соответствии с вариантом, выданным преподавателем.

3 Оформить отчет.

Содержание отчета:

– цель работы;

– постановка задачи;

– текст программы.

Теоретические сведения

Теоретические сведения представлены в [1, с. 180–196, гл. 6].

Задания

1 Используя условный переход, выведите сообщение «Счастливчик!» если $age попадает в диапазон между 18 и 35. Если значение иное, выведите «Не повезло». Расширьте предыдущую конструкцию сообщением «Слишком молод», если $age в диапазоне между 1 и 17.

2 Используя циклы, сформируйте массив четных чисел из диапазона от 1 до 100. Выводя массив на экран, исключите из вывода все числа, которые не делятся на 5.

3 Создайте массив с элементами Name, Address, Phone, Mail и заполните его. С помощью цикла foreach осуществите форматированный вывод массива в виде «элемент: значение».

4 while

Выведите последовательно числа от 1990 до 2007, используя цикл while.

5 do-while

Выведите последовательно числа от 1990 до 2007, используя цикл do while.

6 for

Выведите последовательно числа от 1990 до 2007, используя цикл for.

7 break

Выведите последовательно числа от 1990 до 2007, используя цикл while. Прервите вывод на 1995 г.

8 continue

Выведите последовательно числа от 1990 до 2007, используя цикл while. Не выводить года с 1994 по 1997.

9 switch

Задайте значение переменной $name. Произведите проверку переменной $name на имена John, Bill, Sam. Также, в случае отрицательного результата выведите «Приветствую Незнакомец».

Контрольные вопросы

1 Основные управляющие конструкции PHP.

2 Операторы ветвлений PHP. Приведите примеры.

3 Операторы циклов PHP. Приведите примеры.

3 Лабораторная работа № 3. Обработка строковых данных на PHP

Цель работы: изучить функции обработки строковых данных в языке PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения.

2 Выполнить задание к лабораторной работе.

3 Оформить отчет.

Содержание отчета:

– цель работы;

– постановка задачи;

– результаты выполнения, тестирования и разработки программы.

Теоретические сведения

Теоретические сведения представлены в [1, с. 261–269, гл. 13].

Задания

Вариант 1

1 Дана строка «Привет, мир!». Сделайте из нее строку «ПРИВЕТ МИР!»

2 Нарисуйте пирамиду из символов, у которой должно быть столько рядов, чтобы последний элемент пирамидки состоял из одного символа. Первый ряд пирамиды должен храниться в переменной $str (может иметь различное количество символов). Подсказка: воспользуйтесь функциями strlen и substr.

3 Дана строка «Я-учу-PHP!». Замените все дефисы на тег «!».

4 Дана строка «я учу PHP!». С помощью функции explode запишите каждое слово этой строки в отдельный элемент массива.

5 Дана строка «html, php, js». Удалите теги из этой строки.

6 Дана строка «Мама мыла раму». Узнайте количество букв «а» и «м», входящих в эту строку.

7 Проверьте, является ли слово палиндромом (одинаково читается во всех направлениях, примеры таких слов: madam, otto, kayak, nun, level).

Вариант 2

1 Дана строка «PHP». Сделайте из нее строку «php».

2 Дана строка «я учу PHP!». Вырежите из нее слово «учу» и слово «PHP».

3 Дана строка «31.12.2013». Замените все точки на дефисы.

4 В переменной $date лежит дата в формате «31.12.2013». Преобразуйте эту дату в формат «2013-12-31»

5 Дана строка «html, php, js». Выведите ее на экран «как есть»: т. е. браузер не должен преобразовать в жирный.

6 Запишите в переменную $str длинный текст. Подсчитайте количество символов и количество слов в этом тексте.

7 Определите, является ли фраза палиндромом. Примеры: «Never odd or even», «A man, a plan, a canal. Panama». Обратите внимание на то, что при обратном чтении игнорируются пробелы, запятые, дефисы, тире и большие буквы (подсказка: значит сначала нужно привести строку к стандартному виду, т. е. удалить лишние символы, привести все к нижнему регистру).

Вариант 3

1 Дана строка «LONDON». Сделайте из нее строку «London».

2 Дана переменная $str, в которой хранится какой-либо текст. Реализуйте обрезание длинного текста по следующему принципу: если количество символов этого текста больше заданного в переменной $n, то в переменную $result запишем первые $n символов строки $str и добавим в конец троеточие «...». В противном случае в переменную $result запишем содержимое переменной $str.

3 Дана строка $str. Замените смайлики «:)», «:(,» «^-^», которые встречаются в этой строке на соответствующие картинки (<img src="">).

4 В переменной $date лежит дата в формате «2013-12-31». Преобразуйте эту дату в формат «31.12.2013».

5 Дана строка «php». Сделайте из нее три разных строки с помощью функций класса trim: «php», «php», «php».

6 Создайте массив гласных букв. С помощью этого массива подсчитайте количество гласных в строке $str. Результат представьте в виде ассоциативного массива, где ключами будут буквы, а элементами – их количество.

7 Нарисуйте пирамиду из цифр, у которой должно быть девять рядов. Решите задачу с помощью одного цикла и функции str_repeat.

Вариант 4

1 Дана строка «london is the capital of great Britain». Сделайте из нее строку «London Is The Capital Of Great Britain».

2 Дана переменная $password, в которой хранится пароль пользователя. Если количество символов пароля больше пяти и меньше десяти, то выведите пользователю сообщение о том, что пароль подходит, иначе сообщение о том, что нужно придумать другой пароль.

3 Дана переменная $str, в которой хранится строка русского текста. Напишите скрипт, который запишет транслит этого текста в переменную $translit.

4 Дан массив с элементами «html», «css», «php», «js». С помощью функции implode создайте строку из этих элементов, разделенных запятыми.

5 Дана строка «html, php, js». Выведите ее на экран «как есть»: т. е. браузер не должен преобразовать в жирный.

6 Дана строка «1234567890». Разбейте ее на массив с элементами «12», «34», «56», «78», «90».

7 Нарисуйте пирамиду из звёздочек, у которой должно быть восемь рядов. Решите задачу с помощью одного цикла и функции str_repeat.

Контрольные вопросы

1 Какими способами может быть определена строка?
2 Что происходит, если строка определяется в двойных кавычках?
3 Какой синтаксис называется сложным (фигурным) в языке PHP?
4 Для чего используется конкатенация строк и как она выполняется в PHP?

5 Каким оператором рекомендуется пользоваться при сравнении строк в языке PHP?

4 Лабораторная работа № 4. Обработка массивов в PHP

Цель работы: получить навыки в работе с массивами на языке PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения.
2 Выполнить задание к лабораторной работе.
3 Оформить отчет.

Содержание отчета:

– цель работы;
– постановка задачи;
– результаты выполнения, тестирования и разработки программы.

Теоретические сведения

Теоретические сведения представлены в [1, с. 194–200, гл. 10; с. 279–290, гл. 14].

Задания

Создайте массив, содержащий данные в соответствии с вариантом. Произведите необходимые операции над массивом.

Вариант 1

В массиве хранятся следующие данные об учениках: фамилия, имя, отчество, рост, масса. Вычислить средний рост учеников, рост самого высокого и самого низкого ученика. Сколько учеников могут заниматься в баскетбольной секции, если рост баскетболиста должен быть больше 170 см?

Создайте ассоциативный массив, содержащий названия книг, организованных по жанрам («детектив», «женский роман», «классика» и др.), а элементами – названия книг.

Вариант 2

Описать массив «экзаменационная ведомость» (предмет, номер группы, номер зачетной книжки, фамилия, имя, отчество студента, его оценки по итогам текущей сессии). Определить отличников, хорошистов, троечников и двоечников.

Создайте ассоциативный массив, аналогичный телефонному справочнику. Отсортируйте массив по фамилиям абонентов в алфавитном порядке.

Вариант 3

Массив содержит сведения об учителях школы. Распечатайте список тех учителей, которые преподают математику и информатику, укажите стаж их работы и недельную нагрузку.

Создайте ассоциативный массив, содержащий сведения о ваших друзьях. Отсортируйте его по возрасту друзей и выведите всю информацию.

Вариант 4

Опишите массив, содержащий информацию о движении электропоездов из вашего города: направление, время отправления электропоездов, время в пути до конечного пункта, стоимость билетов по зонам. Выведите перечень электропоездов, следующих в заданном направлении.

Создайте ассоциативный многомерный массив, содержащий информацию о пользователях (ФИО, возраст, количество посещений страницы). Выведите всю информацию, начиная с пользователей, у которых количество посещений страницы больше.

Вариант 5

Массив содержит сведения о работниках предприятия. Найдите тех, чья заработная плата за месяц является ниже средней по предприятию, а также распечатайте список тех, кто проработал на предприятии более 10 лет с указанием их фамилии, зарплаты, стажа работы и должности.

Опишите массив служащих, включающий имена, фамилии, отчества служащих, даты рождения, полученное образование, домашние адреса, профессии. Определить имена людей с высшим образованием. Выдайте данные о служащем, который имеет ту или иную профессию.

Пример – Постраничная навигация на PHP.

```
<?php
   $language[] = "PHP";
   $language[] = "C++";
   $language[] = "Java";
   $language[] = "Ruby";
   $language[] = "Python";
   $language[] = "Perl";
   $language[] = "Visual Basic";
   $language[] = "Fortran";
   $language[] = "Pascal";
   $language[] = "Assembler";
   $language[] = "Lisp";
   $language[] = "Haskell";
   $language[] = "C#";
```

```
// Определяем количество элементов на одной странице
$pnumber = 2;

// Проверяем, передан ли номер текущей страницы
if(isset($_GET['page'])) $page = intval($_GET['page']) ;
else $page = 1;

// Количество элементов в массиве
$total = count($language);
// Вычисляем количество страниц
$number = (int)($total/$pnumber);
if((float)($total/$pnumber)—$number != 0) $number++;

// Начальный индекс массива $language
// для вывода на текущей странице
$start = (($page - l)*$pnumber + 1);
// Конечный индекс массива $language
// для вывода на текущей странице
$end = $page*$pnumber + 1;
if($end > $total) $end = $total;

// Выводим содержимое страниц
for($i = $start; $i < $end; $i++)
{
   echo $language[$i]."<br />";
}

// Постраничная навигация for($i = 1; $i <= $number; $i++)
{
   // Если это произвольная страница
   if($i != $number)
   {
      if($page == $i)
      {
         // Текущую страницу не подсвечиваем ссылкой
         echo " [".( ($i - 1)*$pnumber + 1). "–".$i*$pnumber."] ";
      }
      else
      {
         echo "<a href='index.php?page=$i'>[".
               (($i - 1)*$pnumber + 1). "–".$i*$pnumber."]</a> ";
      }
   }
   // Если это последняя страница, заменяем последнюю цифру
```

```
        // максимальным числом позиций в массиве $temp
        else
        {
           if($page – $i)
           {
              // Текущую страницу не подсвечиваем ссылкой
              echo "[".(($i - 1)*$pnumber + 1). "–".($total - 1)."] ";
           }
           else
           {
              echo "<a href='index.php?page=$i'>[".
                    (($i - 1)*$pnumber + 1). "–".($total - 1)."]</a> ";
           }
           }
        }
     }
?>
```

Контрольные вопросы

1 Дайте определение термину «массив».
2 Как принято называть отдельные переменные в массиве?
3 Какие способы существуют для создания массивов?
4 При помощи какой функции осуществляется вывод структуры массива?

5 Лабораторная работа № 5. Создание функций на PHP

Цель работы: получить навыки создания функций в языке программирования PHP.

Порядок выполнения работы.
1 Изучить теоретические сведения.
2 Выполнить задание к лабораторной работе.
3 Оформить отчет.

Содержание отчета:
– цель работы;
– постановка задачи;
– результаты выполнения, тестирования и разработки программы.

Теоретические сведения

Теоретические сведения представлены в [1, с. 209–221, гл. 11].

Задания

1 Преобразуйте программу для работы с ассоциативным массивом из лабораторной № 4, добавив возможность ввода данных с формы. При организации ввода реализуйте максимально полную проверку корректности. Используйте функции.

2 Реализуйте следующие функции: среди n чисел найти наибольшее и наименьшее простые числа; для заданного числа n постройте треугольник Паскаля; напишите функцию, возвращающую текст приветствия в зависимости от текущего времени.

Пример – Создание примитивной функции на PHP.

```
$a = 10; // Объявление глобальной переменной
// Объявление функции с названием plus
function plus () {
$b = 5; // Объявление локальной переменной
echo $GLOBALS['a'] + $b; // Складываем и выводим на экран
}
// Вызываем функцию plus
plus();
```

Контрольные вопросы

1 При помощи какого ключевого слова производится объявление функции в PHP?

2 Что такое глобальные переменные в PHP и как они объявляются?

3 Какой глобальный массив содержит все переменные сессии текущего пользователя в PHP?

4 Какой глобальный массив содержит все cookie-файлы, которые сервер установил на стороне пользователя в PHP?

6 Лабораторная работа № 6. Создание и обработка INI-файлов

Цель работы: ознакомиться с механизмом создания и обработки INI-файлов на языке PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения.

2 Выполнить задание к лабораторной работе.

3 Оформить отчет.

Содержание отчета:

– цель работы;

– постановка задачи;
– результаты выполнения, тестирования и разработки программы.

Теоретические сведения

Теоретические сведения представлены в [2, с. 326, гл. 16].

Задания

1 Создайте функцию, которая считывает несколько чисел из INI-файла и реализует над ними какую-либо математическую операцию (сложение, деление и т. д.).

2 Создайте несколько языковых версий одностраничной HTML-страницы, добавив возможность загрузки различных настроек для страницы из INI-файла.

Пример – PHP-скрипт, реализующий мультиязычный вывод.

```
<?php
$lang = isset($_GET['lang'])? $_GET['lang']: "en";
$langconst = parse_ini_file("lang_$lang.ini");
echo $langconst['HELLO']." ".$langconst['WORLD']."!";
?>
```

Контрольные вопросы

1 Для чего используются INI-файлы в языке PHP?

2 Какая функция используется для чтения INI-файла в PHP?

3 Что такое GET-параметры ссылки? Приведите пример ссылки с GET-параметром.

4 Приведите пример структуры INI-файла.

7 Лабораторная работа № 7. Применение регулярных выражений

Цель работы: изучить приемы работы с регулярными выражениями в PHP.

Теоретические сведения

Регулярное выражение (***regular expression***, ***regexp***, ***регэксп***) – механизм, позволяющий задать шаблон для строки и осуществить поиск данных, соответствующих этому шаблону в заданном тексте. Кроме того, дополнительные функции по работе с regexp'ами позволяют получить найденные данные в виде массива строк, произвести замену в тексте по шаблону, разбиение строки по шаблону и т. п. Однако главной их функцией, на которой основаны все

остальные, является именно функция поиска в тексте данных, соответствующих шаблону, описанному в синтаксисе регулярных выражений.

Очень часто регулярные выражения используются для того, чтобы проверить, является ли данная строка строкой в необходимом формате. Например, следующий regexp предназначен для проверки того, что строка содержит корректный e-mail адрес:

```
/^\w+([\.\w]+)*\w@\w((\.\w)*\w+)*\.\w{2,3}$/
```

Регулярные выражения пришли к нам из **Unix** и **Perl**. В PHP существует два различных механизма для обработки регулярных выражений: POSIX-совместимые и Perl-совместимые. Их синтаксис во многом похож, однако Perl-совместимые регулярные выражения более мощные и, к тому же, работают намного быстрее (в некоторых случаях до 10 раз быстрее). Поэтому здесь мы будем вести речь только о Perl-совместимых регулярных выражениях.

Кстати, необходимо заметить, что полное описание синтаксиса регулярных выражений, имеющееся в PHP Manual, занимает более 50 Кбайт и, естественно, здесь мы не будем рассматривать весь синтаксис. Нам необходимы только основы, которые помогут вам понять, как именно пишутся регулярные выражения.

Сутью механизма регулярных выражений является то, что они позволяют задать шаблон для **нечеткого** поиска по тексту. Например, если перед вами стоит задача найти в тексте определенное слово, то с этой задачей хорошо справляются и обычные функции работы со строками. Однако если вам нужно найти «то, не знаю что», о чем вы можете сказать только то, как **приблизительно** это должно выглядеть – то здесь без регулярных выражений просто не обойтись. Например, вам необходимо найти в тексте информацию, про которую вам известно только то, что это «*3 или 4 цифры*, *после которых через пробел идет 5 заглавных латинских букв*», то вы сможете сделать это очень просто, воспользовавшись следующим регулярным выражением:

```
/\d{3,4}\s[A-Z]{5}/
```

Синтаксис регулярных выражений.

Регулярные выражения представляют собой строку. Строка всегда начинается с символа разделителя, за которым следует непосредственно регулярное выражение, затем еще один символ разделителя и потом необязательный список модификаторов. В качестве символа разделителя обычно используется слэш («/»). Таким образом, в следующем регулярном выражении: /\d{3}-\d{2}/m, символ «/» является разделителем, строка «\d{3}-\d{2}» – непосредственно регулярным выражением, а символ «m», расположенный после второго разделителя, – это модификатор.

Основой синтаксиса регулярных выражений является тот факт, что некоторые символы, встречающиеся в строке рассматриваются не как обычные

символы, а как имеющие специальное значение (т. н. метасимволы). Именно это решение позволяет работать всему механизму регулярных выражений. Каждый метасимвол имеет свою собственную роль в синтаксисе регулярных выражений. Далее мы рассмотрим все эти метасимволы.

Одним из самых важных метасимволов является символ обратного слэша ('\'). Если в строке встречается этот символ, то парсер рассматривает символ, непосредственно следующий за ним двояко:

– если следующий символ в обычном режиме имеет какое-либо специальное значение, то он теряет это свое специальное значение и рассматривается как обычный символ. Это совершенно необходимо для того, чтобы иметь возможность вставлять в строку специальные символы, как обычные. Например метасимвол «.», в обычном режиме означает «любой единичный символ», а «\.» означает просто точку. Также можно лишить специального значения и сам этот символ: «\\»;

– если следующий символ в обычном режиме не имеет никакого специального значения, то он может получить такое значение будучи соединенным с символом «\». К примеру, символ «d» в обычном режиме воспринимается просто как буква, однако, будучи соединенной с обратным слэшем («\d»), становится метасимволом, означающим «любая цифра».

Существует множество символов, которые образуют метасимволы в паре с обратным слэшем. Как правило, подобные пары используются для того, чтобы показать, что на этом месте в строке должен находиться символ с кодом, который не имеет соответствующего ему изображения или же символ, принадлежащий какой-то определенной группе символов. В таблице 7.1 приведены некоторые наиболее употребительные метасимволы.

Синтаксис регулярных выражений имеет средства для определения собственных подмножеств символов. Например, вам может понадобиться задать условие, что в этом месте строки должна находиться шестнадцатеричная цифра или еще что-то подобное. Для описания таких подмножеств применяются символы квадратных скобок «[]». Квадратные скобки, встреченные внутри регулярного выражения, считаются одним символом, который может принимать значения, перечисленные внутри этих скобок.

Есть небольшая тонкость в том, как работают метасимволы внутри квадратных скобок. Дело в том, что в синтаксисе регулярных выражений существует еще множество метасимволов, но практически все они работают только **вне** секций описаний подмножеств. Единственные метасимволы, которые работают внутри этих секций, это следующие.

Обратный слэш («\»). То есть все метасимволы таблицы 7.1 будут работать.

Минус («-»). Используется для задания набора символов из одного промежутка (например, все цифры могут быть заданы как «0-9»).

Символ «^». Если этот символ стоит **первым** в секции задания подмножества символов (и только в этом случае!) он будет рассматриваться как символ отрицания. Таким образом можно задать все символы, которые **не описаны** в данной секции.

Несколько примеров использования метасимволов приведены в таблице 7.2.

Таблица 7.1 – Метасимволы и их значения с примерами

Метасимвол	Значение
Метасимволы для задания символов, не имеющих изображения	
\n	Символ перевода строки (код 0×0A)
\r	Символ возврата каретки (код 0×0D)
\t	Символ табуляции (код 0×09)
\xhh	Вставка символа с шестнадцатеричным кодом 0xhh, например, \x41 вставит латинскую букву «A»
Метасимволы для задания групп символов	
\d	Цифра (0 – 9)
\D	Не цифра (любой символ, кроме символов 0–9)
\s	Пустой символ (обычно пробел и символ табуляции)
\S	Непустой символ (все, кроме символов, определяемых метасимволом \s)
\w	«Словесный» символ (символ, который используется в словах. Обычно все буквы, все цифры и знак подчеркивания « _ »
\W	Все, кроме символов, определяемых метасимволом \w
Несколько простейших примеров	
Regexp	Комментарии
/\d\d\d/	Любое трехзначное число («123», «719», «001»)
/\w\s\d\d/	Буква, пробел (или табуляция) и двузначное число («A 01»,
«z 45», «S 18»)	

Таблица 7.2 – Использование метасимволов в регулярных выражениях

Regexp	Комментарий
[0-9A-Fa-f]	Цифра в шестнадцатеричной системе счисления
[\dA-Fa-f]	То же, но с использованием метасимвола
[02468]	Четная цифра
[\^d]	Все, кроме цифр (аналог метасимвола \D)
[a^b]	Любой из символов «a», «b», «^». Заметьте, что здесь символ «^» не имеет какого-либо специального значения, потому что стоит не на первой позиции внутри квадратных скобок

Теперь необходимо рассмотреть еще несколько метасимволов. Как было сказано ранее, все они работают только вне секций описаний подмножеств символов (вне квадратных скобок).

Символы «^» и «{$}». Они используются для того, что того, чтобы указать парсеру регулярных выражений на то, чтобы он обратил внимание на положение искомого текста в строке. Символ «^» указывает, что искомый текст должен находиться в начале строки, символ «{$}», наоборот, указывает, что искомый

текст должен находиться в конце строки. Посмотрим, как это работает на примере.

Допустим, у нас есть текст:

12 aaa bbb
aaa 27 ccc
aaa aaa 45

И регулярное выражение для поиска чисел в этом тексте: /\d\d/m (не обращайте пока внимания на модификатор). Поиск по этому регулярному выражению вернет нам три значения: «12», «27», «45». Теперь ограничим поиск, указав, где именно внутри строки должен располагаться текст: /^\d\d/m. Здесь результат будет только один – «12», потому что только это число располагается в начале строки. Аналогично, регулярное выражение /\d\d$/m вернет результат «45».

Символ точки «.». Этот метасимвол указывает, что на данном месте в строке может находиться любой символ (за исключением символа перевода строки). Очень удобно использовать его, если вам нужно «пропустить» какую-нибудь букву в слове при проверке. Например регулярное выражение /.bc/ найдет в тексте и «abc» и «Abc» и «Zbc» и «5bc».

Символ вертикальной черты «|». Используется для задания списка альтернатив. Например, регулярное выражение

/(красное|зеленое) яблоко/

найдет в тексте все словосочетания «красное яблоко» и «зеленое яблоко».

Символы круглых скобок «(« и »)». Эти символы позволяют получить из искомой строки дополнительную информацию. Обычно, если парсер регулярных выражений ищет в тексте информацию по заданному выражению и находит ее – он просто возвращает найденную строку. Однако если он встречает внутри регулярного выражения круглые скобки, то рассматривает содержимое этих скобок как еще одно регулярное выражение, по которому необходимо произвести поиск. Парсер рекурсивно вызывает сам себя для поиска по новому регулярному выражению и использует результаты поиска для дальнейшей обработки основного регулярного выражения. При этом, если поиск хотя бы по одному из внутренних регулярных выражений не увенчался успехом, – поиск по всему регулярному выражению считается безуспешным.

Рассмотрим в качестве примера то, как работает парсер регулярных выражений в случае приведенного выше регулярного выражения о яблоках: /(красное зеленое) яблоко/.

1 Парсер начинает разбор регулярного выражения и встречает выражение в скобках: (красное|зеленое).

2 Парсер вызывает себя для поиска по найденному регулярному выражению.

3 Получив результаты поиска, парсер подставляет по очереди каждый из полученных результатов на место выражения в скобках и смотрит, удовлетворяет ли найденный результат всем условиям основного регулярного выражения (в данном случае смотрит, есть ли после найденного слова слово «яблоко»).

4 Если все в порядке – результаты поиска по каждому из имеющихся регулярных выражений для этого случая возвращаются, если нет – парсер просто переходит к следующему найденному фрагменту. Результат поиска внутреннего регулярного выражения для этого фрагмента при этом теряется.

В качестве примера возьмем строку:

яблоко красное и зеленое яблоко и еще одно красное

яблоко и еще одно яблоко зеленое

Поиск по внутреннему регулярному выражению даст четыре результата (выделены жирным шрифтом):

яблоко **красное** и **зеленое** яблоко и еще одно **красное** яблоко и еще одно яблоко **зеленое**

Однако поиск по всему регулярному выражению даст всего два результата, потому как в остальных случаях условия основного регулярного выражения не выполняются:

яблоко красное и **зеленое яблоко** и еще одно **красное яблоко** и еще одно яблоко, зеленое

Необходимо заметить, что для этих двух случаев будет возвращен не только результат поиска по основному регулярному выражению, но и результат поиска по внутреннему регулярному выражению для каждого из найденных фрагментов. В большинстве случаев это полезно, но иногда наоборот, лучше избавиться от лишних результатов. В этом случае необходимо добавить символы «?:» непосредственно после открывающейся круглой скобки: /(?:красное|зеленое) яблоко/.

Теперь пример, когда получение результатов внутренних регулярных выражений может быть полезным. Допустим, нам необходимо проверить, является ли строка семизначным телефонным номером с указанием кода города и получить из нее код города и номер телефона:

/\((\d{3,5})\)\s+(\d{3}-\d{2}-\d{2})/

Давайте рассмотрим это регулярное выражение подробнее.

Первая круглая скобка здесь теряет свое специальное значение и будет рассматриваться как обычный символ:\(

Далее идет регулярное выражение в скобках (проверка кода города):(\d{3,5}).

После этого идет закрывающая круглая скобка, которая также лишена своего специального значения из-за символа обратного слэша, стоящего перед ней:\).

Затем идет пропуск пустого места:\s+.

И еще одно регулярное выражение в скобках, которое проверяет номер телефона:(\d{3}-\d{2}-\d{2}).

Как видите, здесь есть три регулярных выражения – основное и два внутренних. При этом основное выражение позволяет проверить, имеет ли строка необходимый нам формат, а два внутренних – получить, соответственно, код города и номер телефона.

Посмотрим, как работает это регулярное выражение. Пусть у нас есть строка: «My phone is (095) 123-45-67». Результатами поиска будут три строки: «(095) 123-45-67», «095» и «123-45-67».

Осталось рассмотреть еще одну группу метасимволов, определяющих количественные показатели (т. н. ***quantifiers***). Очень часто бывает необходимо указать, что какой-то символ должен повторяться определенное количество раз. Конечно, можно просто указать его необходимое количество раз непосредственно в строке, но это, естественно не выход. Тем более, что очень часто встречаются ситуации, когда точное количество символов неизвестно. Поэтому синтаксис регулярных выражений содержит набор метасимволов, предназначенных именно для решения подобных задач. Каждый из описанных ниже метасимволов определяет количественную характеристику символа, который находится **непосредственно** перед ним.

Звездочка «*». Указывает, что символ должен быть повторен 0 или более раз (т. е. символ может отсутствовать или присутствовать в любых количествах). Пример: выражение /ab*c/ найдет строки «ac», «abc», «abbc» и так далее.

Плюс «+». Указывает, что символ должен быть повторен один или более раз (т. е. символ обязан присутствовать и может присутствовать в любых количествах). Пример: выражение /ab+c/ найдет строки «abc», «abbc», «abbbc» и т. д., но не найдет строку «ac».

Знак вопроса «?». Указывает, что символ может как присутствовать, так и нет, но при этом не может повторяться более одного раза. Пример: выражение /ab?c/ найдет строки «ac» и «abc», но не найдет строку «abbc».

Фигурные скобки «{» и «}». Определяют количественную характеристику символа. Внутри скобок через запятую перечисляются минимальное и максимальное количество повторений символа. При этом любой из параметров может быть опущен, а кроме того, можно задать точное количество повторений, указав только одно число.

Примеры:

{2,4} – символ должен повториться минимум 2 раза, но не более 4;

{,5} – символ может отсутствовать (т. к. не задано минимальное количество повторений), но если присутствует, то не должен повторяться более 5 раз;

{3,} – символ должен повторяться минимум 3 раза, но может быть и больше;

{4} – символ должен повторяться ровно 4 раза.

Есть еще одна тонкость в использовании метасимвола «?». Посмотрите на такое выражение: /.+a/. Ожидается, что оно вернет нам часть текста до первого вхождения символа «a» в этот текст. На самом деле оно будет работать несколько не так, как ожидается, и результатом поиска будет весь текст до **последнего** вхождения символа «a». Дело в том, что по умолчанию количественные метасимволы «жадничают» и пытаются захватить как можно больший кусок текста. Если это не нужно (как в нашем случае), то необходимо «отучить» их от жадности, указав знак «?» после количественного метасимвола: /.+?a/. После этого выражение будет работать так, как надо.

Модификаторы регулярных выражений.

Механизм регулярных выражений позволяет добавлять модификаторы, влияющие на обработку регулярного выражения. В таблице 7.3 рассмотрены наиболее употребительные.

Таблица 7.3 – Модификаторы регулярных выражений

Модификатор	Значение
i	Включение режима case-insensitive, т. е. большие и маленькие буквы в выражении не различаются
m	Указывает на то, что текст, по которому ведется поиск, должен рассматриваться, как состоящий из нескольких строк. По умолчанию механизм регулярных выражений рассматривает текст как одну строку вне зависимости от того, чем она является на самом деле. Соответственно, метасимволы «^» и «$» указывают на начало и конец всего текста. Если же этот модификатор указан, то они будут указывать, соответственно, на начало и конец каждой строки текста
s	По умолчанию метасимвол «.» не включает в свое определение символ перевода строки. То есть для многострочного текста выражение /.+/ вернет только первую строку, а не весь текст, как ожидается. Указание этого модификатора снимает это ограничение
U	Делает все количественные метасимволы «не жадными» по умолчанию (про «жадность» количественных метасимволов см. выше)

Задания

1 Найдите в документации описание следующих функций работы с регулярными выражениями:

preg_match();
preg_match_all();
preg_replace();
preg_replace_callback();
preg_split();
preg_quote();
preg_grep();
preg_quote().

Реализуйте примеры их использования.

2 Составьте регулярные выражения для маскирования тегов HTML (20–30 на выбор).

3 Составьте регулярное выражение для проверки значения переменной:

Быть целым числом.

Быть вещественным числом.

Быть идентификатором.

Быть правильным телефонным номером (например, 37-81-40).

Быть правильным телефонным номером с кодом города (например, (231) 5-94-00).

Быть не числом.

Не содержать цифр.

Не содержать букв.

4 Постройте регулярное выражение, возвращающее значение параметров тегов html-документа, содержащих URL.

8 Лабораторная работа № 8. Модернизация приложений. Использование ООП на PHP

Цель работы: получить навыки создания классов на PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения.

2 Выполнить задание к лабораторной работе в соответствии с вариантом.

3 Оформить отчет.

Содержание отчета:

– цель работы;

– постановка задачи;

– текст программы.

Теоретические сведения

Конструктор – метод, автоматически вызываемый при создании объекта. Для описания контруктора применяется служебное слово «__construct». Удобен для инициализации, когда для каждого объекта необходимо выполнить ряд действий. В параметрах конструктора можно указать свойства объекта.

Деструктор – метод, автоматически вызываемый перед удалением объекта. Антипод конструктора. Для вызова метода при удалении объекта применяется «__destruct». Круглые скобки в деструкторе указываются только из-за особенностей стнтаксиса php. В деструктор нельзя передать параметр. Кроме того, последовательность удаления объектов четко не определена, вследствие

чего из деструктора *нельзя* обращаться к другим объектам. Вызывается в двух случаях: при завершении выполнения кода (не совсем) и при удалении объекта.

Клон. В PHP, начиная с версии 5.0, знак «=» при создании объекта на самом деле является присваиванием по ссылке. Копирование же объекта осуществляется при помощи служебного слова «clone». В клон нельзя передать параметр. При клонировании конструктор не вызывается!

```
class Pet_db{  // объявление класса Pet_db
public $name;  // объявление свойства будущих объектов
public $type = "type";  // объявление свойства будущих объектов со значением п
о умолчанию

function __construct($t,$n,$w){  // Конструктор
$this->name=$n;
$this->type=$t;
$this->word=$w;
echo "Создан объект класса '".__CLASS__."'. type: {$this->type}; name: {$this-
>n}<br>\n";
echo "Object name {$this->n} sayd: \"{$this->word}\"<br>\n";}

function __clone(){  // Клон
echo "Клонирован объект класса '".__CLASS__."'. type: {$this-
>type}; name: {$this->n}<br>\n";
echo "Object name {$this->n} sayd: \"{$this->word}\"<br>\n";}

function __destruct(){  // Деструктор
echo "Object name {$this->n} sayd \"{$this->word}, and I'll be back\"<br>\n";
echo "Удалён объект класса '".__CLASS__."'. type: {$this->type}; name: {$this-
>n}<br>\n";}
}
$cat = new Pet_db("cat","Cat","I'm a cat");  //создание экземпляров класса Pet_db
$dog = new Pet_db("dog","Dog","I'm a dog");  //создание экземпляров класса Pet_
db

$dog2 = clone $dog;  // клонирование экземпляра класса Pet_db

unset($cat);  // принудительный вызов деструктора с помощью удаления пере-
менной
unset($dog); // принудительный  вызов деструктора с помощью удаления пе
ременной
unset($dog2); // принудительный вызов деструктора с помощью удаления пе
ременной
```

Result:

```
Создан объект класса 'Pet_db'. type: cat; name:
```

Object name sayd: "I’m a cat"
Создан объект класса 'Pet_db'. type: dog; name:
Object name sayd: "I’m a dog"
Клонирован объект класса 'Pet_db'. type: dog; name:
Object name sayd: "I’m a dog"
Object name sayd "I’m a cat, and I’ll be back"
Удалён объект класса 'Pet_db'. type: cat; name:
Object name sayd "I’m a dog, and I’ll be back"
Удалён объект класса 'Pet_db'. type: dog; name:
Object name sayd "I’m a dog, and I’ll be back"
Удалён объект класса 'Pet_db'. type: dog; name:

Порядок выполнения работы.

1 Создайте класс User_dd. В текстовом редакторе откройте файл oop\users.php.

2 В классе создайте свойства name, login и password.

3 В классе создайте и опишите метод showInfo(), который выводит информацию о пользователе в произвольной форме.

4 Создайте три объекта, экземпляры класса «User_dd»: $user1, $user2 и $user3.

5 Задайте произвольные значения свойств name, login и password для каждого из объектов.

6 Вызовите метод showInfo() для каждого объекта.

7 Сохраните файл oop\users.php.

```
class User_dd{
public $n;
public $l;
public $p;

function showInfo(){
echo "name: '{$this->n}'; login: '{$this->l}'; password: '{$this->p}';<br>\n";
}
}

$user01 = new User_dd;
$user01 -> n = "Ivann";
$user01 -> l = "ivann";
$user01 -> p =  "7c2ed4b2b9" ;

$user02 = new User_dd;
$user02 -> n = "Petro";
$user02 -> l = "petro";
$user02 -> p = "81b4fa24b7";
```

```
$user03 = new User_dd;
$user03 -> n = "Fedor";
$user03 -> l = "fedor";
$user03 -> p = "8ac0182ec9";

$content.=$user01 -> showInfo();
$content.=$user02 -> showInfo();
$content.=$user03 -> showInfo();
```

Result:

```
name: 'Ivann'; login: 'ivann'; password: '7c2ed4b2b9';
name: 'Petro'; login: 'petro'; password: '81b4fa24b7';
name: 'Fedor'; login: 'fedor'; password: '8ac0182ec9';
```

8 В текстовом редакторе откройте файл oop\users.php.

9 В классе User_d2 создайте и опишите конструктор, который принимает в качестве аргументов имя, логин и пароль пользователя.

10 Конструктор должен инициализировать свойства name, login и password.

11 Измените код, который инициализирует объекты, передавая нужные параметры в конструктор.

12 Удалите те строки кода, в которых задаются значения свойств объектов;

13 В классе User_d2 создайте и опишите деструктор.

Деструктор должен выводить строку Пользователь [логин_пользователя] удален.

14 Подставьте вместо подстроки [логин_пользователя] значение свойства login.

15 Сохраните файл oop\users.php.

Задания

Создайте класс по варианту, используйте наследование. Создайте экземпляры класса.

Вариант 1

Создайте класс книги, организуйте по жанрам («детектив», «женский роман», «классика» и др.).

Вариант 2

Создайте класс, аналогичный телефонному справочнику.

Вариант 3

Создайте класс, содержащий сведения о ваших друзьях, и выведите всю информацию.

Вариант 4

Создайте класс, содержащий информацию о пользователях (ФИО, возраст, количество посещений страницы). Выведите всю информацию.

Вариант 5

Опишите класс служащих, включающий имена, фамилии, отчества служащих, даты рождения, полученное образование, домашние адреса, профессии. Выдайте данные о служащем, который имеет ту или иную профессию.

9 Лабораторная работа № 9. Создание базы и подключение базы данных MySQL

Цель работы: получить навыки создания базы данных.

Порядок выполнения работы.

1 Изучить теоретические сведения.

2 Выполнить задание к лабораторной работе в соответствии с вариантом.

3 Оформить отчет.

Содержание отчета:

– цель работы;

– постановка задачи;

– текст программы.

Теоретические сведения

1 Приложение phpMyAdmin.

Входим на страницу администрирования базы данных MySQL. Для того чтобы зайти на страницу администрирования баз данных, в командной строке вашего браузера введите следующий адрес: localhost/ phpmyadmin/.

В левой колонке находятся имеющиеся базы данных, в центральной части – основные настройки (здесь вы можете изменить язык, вид, кодировку). Верхние вкладки предназначены для различных задач.

2 Создание базы данных MySQL.

Для того чтобы создать новую базу данных, нажмем на верхнюю вкладку «Базы данных» и перед нами на центральном поле откроется список всех имеющихся баз данных MySQL. Нам же нужно создать новую. Для этого в поле «Создать базу данных» впишем название создаваемой базы и нажмем на кнопку «Создать».

После того, как вы нажмете на кнопку «Создать», база данных добавится в список баз данных в панели слева и на центральном поле. Теперь выберите новую базу данных, кликнув по ее названию. Здесь вам будет предложено создать таблицу. Создадим таблицу базы данных, как показано на рисунке 9.1.

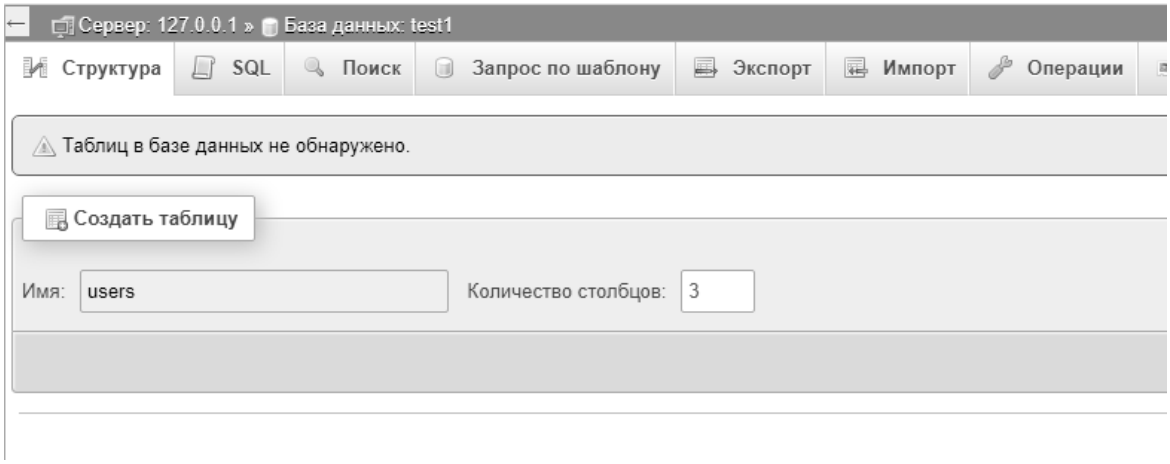

Рисунок 9.1 – Создание таблицы базы данных в СУБД MySQL

В таблице заполните поля «Имя» и «Количество столбцов» и нажмите «Ок». После этого откроется страница для заполнения полей новой таблицы базы данных. Здесь каждому полю нужно присвоить имя, тип хранимых данных, длину (если требуется для данного атрибута) и для такого поля, как идентификатор (id), также требуется указать автоинкремент и первичный ключ.

Для начала надо подключить класс mysqli, при его объявлении он подключает БД.

PHP

```
// Здесь мы объявляем сам класс mysqli;
$mysqli = new mysqli($nameServer, $userName, $password, $DBname);

// тут проверяем, удалось соединится с сервером,
// если соединения нет, то останавливаем программу и выводим сообщение;
if ($mysqli -> connect_error) {
  printf("Соединение не удалось: %s\n", $mysqli -> connect_error);
  exit();
};
```

Давайте разберём каждый параметр.
$nameServer – адрес или имя сервера, на котором хранится БД.
$userName – имя пользователя, который может управлять БД.
$password – пароль пользователя.
$DBname – название базы данных, к которой нужно подключиться.

Задания

Вариант 1

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

Запросы:

– выведите информацию о студентах, распределенных на практику в заданный город;

– выведите информацию о студентах, средний балл которых попадает в заданный интервал.

Вариант 2

Спроектируйте структуру базы данных об автомобилях: номер, год выпуска, марка, цвет, состояние, фамилия владельца, адрес.

Запросы:

– выведите информацию об автомобилях заданной марки;

– вывести информацию об автомобилях, год выпуска которых попадает в заданный интервал.

Вариант 3

Спроектируйте структуру базы данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

Запросы:

– выведите информацию о квартирах, расположенных в заданном районе;

– выведите информацию о квартирах, цена которых не превышает указанную.

Контрольные вопросы

1 Как создать базу данных в MySQLи подключить ее в коде PHP?

2 Синтаксис функции mysql_connect().

3 Приведите примеры SQL-запросов для добавления и удаления данных и их использование на PHP.

10 Лабораторная работа № 10. Создание CRUD-приложения на PHP с базой данных

Цель работы: получить навыки создания CRUD-приложения на PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения.

2 Выполнить задание к лабораторной работе в соответствии с вариантом.

3 Оформить отчет.

Содержание отчета:

– цель работы;

– постановка задачи;

– текст программы.

Теоретические сведения

Несмотря на огромное число разнообразных сайтов, практически всю веб-разработку можно свести к CRUD-операциям.

CRUD означает четыре стандартные операции над любой сущностью: создание, чтение, обновление и удаление. Например, в случае с пользователем можно составить такое соответствие:

Create

– Регистрация

Read

– Просмотр профиля пользователями сайта

– Просмотр пользователя в административном интерфейсе

Update

– Обновление личных данных

– Смена емейла

– Смена пароля

Delete

– Удаление

Точно также можно расписать действия над любыми другими ресурсами, фотографиями пользователя, его друзьями, сообщениями.

Создание полного круда включает в себя следующие действия:

– создание сущности в коде (как правило, класса);

– добавление таблицы в базу;

– написание тестов на обработчики;

– добавление обработчиков;

– добавление шаблонов.

Новички тратят на создание такого CRUD-приложения не один день. У опытного разработчика в прокачанном фреймворке этот процесс занимает максимум часы. Придется многое делать руками. В целях обучения это оправданно, но в промышленной разработке, то, что может быть автоматизировано, должно быть автоматизировано.

Ниже рассмотрим процесс создания CRUD-приложения для пользователя за исключением работы с базой данных и тестов (таблица 10.1). Начнем с роутинга. Полный CRUD пользователя включает минимум семь маршрутов. Их может быть больше, т. к. любое из действий может повторяться не один раз.

Такое соглашение изначально появилось в Rails и затем было адаптировано во многих фреймворках на языках, отличных от Ruby.

Разберем первые два маршрута: просмотр списка и конкретного ресурса.

Список (index).

Повторим для закрепления вывод списка. Общий алгоритм действий такого обработчика всегда проходит по одному сценарию и не зависит от языка программирования.

Таблица 10.1 – Описание маршрутов и шаблонов

Метод	Маршрут	Шаблон	Описание
GET	/users	users/index.phtml	Список пользователей
GET	/users/{id}	users/show.phtml	Профиль пользователя
GET	/users/new	users/new.phtml	Форма создания нового пользователя
POST	/users		Создание нового пользователя
GET	/users/{id}/edit	users/edit.phtml	Форма редактирования пользователя
PATCH/PUT	/users/{id}		Обновление пользователя
DELETE	/users/{id}		Удаление пользователя

1 Извлекаем список из хранилища (базы данных). Обычно с учетом пейджинга.
2 Передаем данные в шаблон.
3 Выводим данные в шаблоне, используя цикл.
Результаты вывода данных представлены на рисунке 10.1.

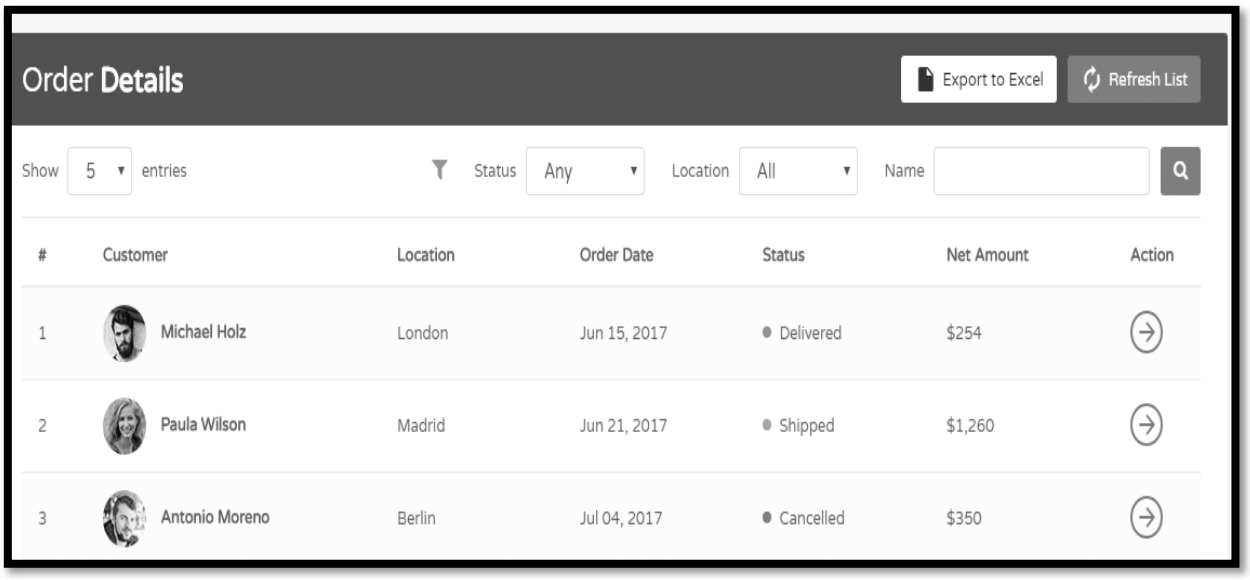

Рисунок 10.1 – Вывод данных списка

Обычно в этот список добавляют различные действия, которые можно выполнять над сущностями, например, редактирование, удаление или просмотр.

Обработчик

```
<?php

$app->get('/schools', function ($request, $response) {
    $repository = new App\SchoolRepository();
    $schools = $repository->all();
    $params = ['schools' => $schools];
```

```
    return $this->get('renderer')->render($response, "schools/index.phtml", $params);
})->setName('schools');
```

Шаблон

```
<table>
  <?php foreach ($schools as $school): ?>
    <tr>
      <td>
        <?= $school['id'] ?>
      </td>
      <td>
        <a href="/schools/<?= $school['id'] ?>"><?= $school['name'] ?></a>
      </td>
    </tr>
  <?php endforeach ?>
</table>
```

Отображение (show).

Страница конкретной сущности. Например, к таким страницам относятся: профиль пользователя, страница курса, страница профессии, страница урока и многие др. Как и в случае со списком, порядок действий для отображения всегда один.

1 Из адреса извлекается идентификатор сущности.

2 Выполняется поиск сущности в хранилище.

3 Она передается в шаблон.

4 В шаблоне рисуется красивый вывод.

Обработчик

```
<?php

$app->get('/schools/{id}', function ($request, $response, array $args) {
    $id = $args['id'];
    $repository = new App\SchoolRepository();
    $school = $repository->find($id);

    $params = [
        'school' => $school
    ];

    return $this->get('renderer')->render($response, 'school/show.phtml', $params);
})->setName('school');
```

Шаблон

```
<?php foreach ($school as $key => $value): ?>
  <div>
```

```
        <?= $key ?>: <?= $value ?>
    </div>
<?php endforeach ?>
```

Если сущность была удалена или ее вообще не существовало, то с точки зрения HTTP такой адрес должен вернуть HTTP-код 404. Сделать это можно явно, вернув соответствующий ответ:

```
<?php

$app->get('/schools/{id}', function ($request, $response, array $args) use ($repo) {
    $id =  $args['id'];
    $school = $repo->find($id);

    if (!$school) {
        return $response->write('Page not found')
            ->withStatus(404);
    }
})->setName('school');
```

11 Лабораторная работа № 11. Создание CRUD-приложения на PHP с базой данных с использованием фреймворка

Цель работы: получить навыки создания CRUD-приложения на PHP.

Порядок выполнения работы.

1 Изучить теоретические сведения.

2 Выполнить пример по разработке CRUD-приложения на фреймворке Yii (или другом) и задание к лабораторной работе в соответствии с вариантом из лабораторной работы № 10.

3 Оформить отчет.

Теоретические сведения

Фреймворк Yii основан на принципах MVC и ООП. В Yii URL-адрес выглядит как **http://localhost/yiitest/index.php?r=controllerID/actionID**.

Например, в блоговой системе URL может быть следующим: **http://localhost/yiitest/index.php?r=post/create. post** – это идентификатор контроллера, а **create** – идентификатор действия.

Основываясь на идентификаторах, скрипт входа решает, какой контроллер и метод вызвать.

Контроллер, имеющий идентификатор post, должен быть назван **PostController** (идентификатор получается путем отсечения суффикса Controller от имени класса и изменением первой буквы на строчную).

Идентификатор действия – это идентификатор метода, представленного в контроллере подобным образом; внутри PostController должен быть метод под названием **actionCreate()**.

Может быть несколько представлений, ассоциированных с одним контроллером, поэтому мы храним файлы представлений внутри папок **protected/views/controllerID**.

Можем создать файл представления для нашего контроллера под названием **create.php** в описанном выше каталоге, и затем представить его пользователям, просто написав следующий код в actionCreate():

```
public function actionCreate()
{
   $this->render('create');
}
```

Нужно создать GET/POST – запросы с помощью PHP.

Можно передать дополнительные данные в представление. Это делается следующим образом:

```
$this->render('create', array('data' => $data_item));
```

Внутри файла представления можем получить доступ к данным через переменную $data.

Представление также имеет доступ к переменной **$this**, которая указывает на экземпляр контроллера, воспроизводящего представление.

Более того, если вы хотите иметь удобные для пользователя URL-адреса, то можете раскомментировать следующий участок кода в файле **protected/config/main.php**:

```
'urlManager'=>array(
   'urlFormat'=>'path',
   'rules'=>array(
      '<controller:w+>/<id:d+>'=>'<controller>/view',
      '<controller:w+>/<action:w+>/<id:d+>'=>'<controller>/<action>',
      '<controller:w+>/<action:w+>'=>'<controller>/<action>',
   )
```

Тогда URL-адреса будут выглядеть следующим образом: **http://localhost/yiitest/controllerID/actionID**.

Разработка CRUD-приложения.

Разработаем простую систему, в которой пользователь может выполнять CRUD-операции (создание, извлечение, обновление и удаление) с сообщением в блоге.

Шаг 1.

Создадим базу данных MySQL под названием yiitest и внутри ее создадим таблицу posts. Таблица будет иметь только три столбца: идентификатор (**id**), название (**title**) и контент (**content**).

```
CREATE TABLE posts (
   id INTEGER NOT NULL PRIMARY KEY AUTO_INCREMENT,
   title VARCHAR(100),
   content TEXT
)
```

Откройте файл конфигурации вашего приложения (**protected/config/main.php**) и раскомментируйте следующие строки:

```
'db'=>array(
'connectionString' => 'mysql:host=localhost;dbname=testdrive,
'emulatePrepare' => true,
'username' => 'root',
'password' => '',
'charset' => 'utf8',
)
```

Замените testdrive на имя вашей базы данных, то есть, yiitest. Также вы должны обеспечить полномочия, необходимые Yii для подключения.

Шаг 2.

В фреймворке Yii каждая таблица базы данных должна иметь соответствующий класс модели типа **CActiveRecord**. Польза от этого в том, что нам не нужно иметь дело с таблицами баз данных напрямую. Вместо этого мы можем работать с объектами модели, которые соответствуют различным строкам таблицы.

Например, класс **Post** – это модель для таблицы сообщений. Объект этого класса представляет собой строку из таблицы posts и имеет атрибуты для отражения значений столбцов.

Для того чтобы быстро генерировать модель, будем использовать веб-инструмент Yii под названием **gii**. Этот инструмент может быть использован для генерации моделей, контроллеров и форм для CRUD-операций.

Чтобы использовать **gii** в проекте, найдите следующие строки в файле конфигурации вашего проекта, раскомментируйте их, добавьте пароль.

```
'gii'=>array(
'class'=>'system.gii.GiiModule',
'password'=>your password to access gii,
'ipFilters'=>array('127.0.0.1','::1'),
)
```

Сравнение дат в PHP с помощью функции StrToTime().

Затем обратитесь к gii с помощью следующего адреса: http://localhost/yiitest/index.php?r=gii. Если используете дружественные пользователю URL-адреса, адрес будет такой: http://localhost/yiitest/gii.

Кликнете на **Model Generator.** gii попросит вас ввести имя таблицы; введите posts для имени таблицы, а для имени модели используйте Post. Затем кликнете Generate для создания модели.

Проверьте папку protected/models, и вы найдете там файл Post.php.

Шаг 3.

Теперь кликнете на **CRUD Generator**. Введите в качестве имени модели Post. Идентификатор контроллера автоматически заполнится как post.

Это означает, что новый контроллер будет сгенерирован под именем **PostController.php**.

Кликнете на Generate. Сгенерируется контроллер, а также несколько файлов представления с формами, необходимыми для CRUD-операций.

Теперь у вас есть совершенно новое CRUD-приложение! Кликнете на ссылку **try it now**, чтобы протестировать его. Для управления сообщениями вам нужно будет войти как **admin/admin**.

Для того чтобы создать новое сообщение, вам нужно обратиться по адресу **http://localhost/yiitest/post/create**, а для обновления определенного сообщения просто откройте в браузере ссылку **http://localhost/yiitest/post/update/postID**.

Аналогично вы можете составить список всех сообщений и удалить все или некоторые из них.

Заключение.

Yii – это очень мощный фреймворк для разработке проектов поколения Web 2.0, с помощью которого можно легко создать полностью функционирующую CRUD-систему всего за несколько минут.

С помощью Yii не нужно начинать проект с нуля. Этот фреймворк предоставляет фундамент приложения, и вы можете расширять его по своему усмотрению.

Список литературы

1 **Котеров, Д. В.** PHP 7. В подлиннике / Д. В. Котеров, И. С. Симдянов. – Санкт-Петербург: БХВ-Петербург, 2020. – 1088 с.

2 **Лисьев, Г. А.** Программное обеспечение компьютерных сетей и web-серверов: учебное пособие / Г. А. Лисьев, П. Ю. Романов, Ю. И. Аскерко. – Москва: ИНФРА-М, 2020. – 145 с.

3 **Прохоренок, Н. А.** HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. А. Прохоренок. – Санкт-Петербург: БХВ-Петербург, 2015. – 768 с.