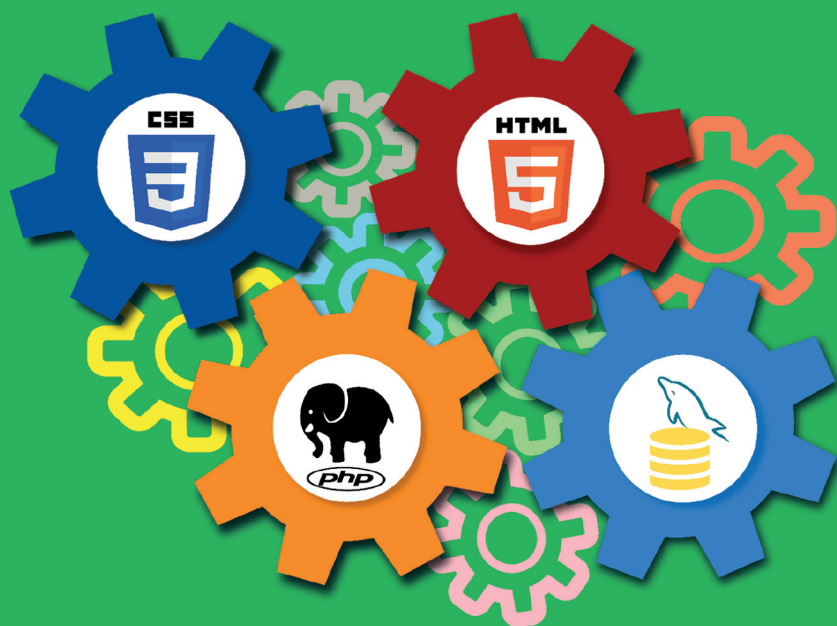


В. А. Ушаков

РАЗРАБОТКА СОВРЕМЕННЫХ ДИНАМИЧЕСКИХ WEB-САЙТОВ СРЕДСТВАМИ ЯЗЫКА PHP



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

В. А. Ушаков

**РАЗРАБОТКА СОВРЕМЕННЫХ
ДИНАМИЧЕСКИХ WEB-САЙТОВ
СРЕДСТВАМИ ЯЗЫКА PHP**
Лабораторный практикум

УДК 004.4
ББК 32.973.202
У93

Рецензенты:

кандидат технических наук, доцент *Н. В. Богословская*;
кандидат технических наук *Е. Л. Турнецкая*

Утверждено
редакционно-издательским советом университета
в качестве лабораторного практикума

Протокол № 2 от 25 марта 2021 г.

Ушаков, В. А.

У93 Разработка современных динамических web-сайтов средствами языка PHP: лабораторный практикум / В. А. Ушаков. – СПб.: ГУАП, 2021. – 73 с.

Приведены краткие теоретические и методические указания к выполнению лабораторных работ по дисциплине «Web-программирование». Лабораторный практикум предназначен для студентов, обучающихся по направлению 09.03.02 «Информационные системы и технологии», а также студентов, изучающих дисциплину «Web-программирование» или разработку современных динамических web-сайтов.

УДК 004.4
ББК 32.973.202

© Санкт-Петербургский государственный
университет аэрокосмического
приборостроения, 2021

ПРЕДИСЛОВИЕ

Дисциплина «Web-программирование» является продолжением дисциплины «Web-технологии» [1-2]. На основании знаний, полученных в ходе освоения дисциплины «Web-программирование», базируются дисциплины «Электронный бизнес» и «Мультимедиа в Web-технологиях». Целью лабораторного практикума является обучение студентов процессу создания современных динамических веб-сайтов [3-5] на скриптовом языке программирования PHP 7 [6-13] и закрепление навыков разработки статических веб-сайтов, приобретенных при изучении дисциплины «Web-технологии» [1-2].

Практикум включает в себя методические указания к пяти лабораторным работам. Первые две лабораторные работы посвящены обработке и проверке данных в HTML-формах. Остальные работы посвящены работе с файлами в PHP, с БД MySQLi [14], а также с сессиями и файлами cookie. Также лабораторный практикум содержит инструкцию по установке локального сервера WAMPserver.

Содержится большое число примеров современных динамических веб-сайтов. Предполагается, что разработанные студентами динамические веб-сайты будут работать не только на локальном сервере, но и будут размещены на хостинге в сети интернет в процессе освоения дисциплины «Электронный бизнес» [15].

Изложение материала строится на процедурном стиле написания кода веб-страниц

Выполнение лабораторных работ, а также приведенные примеры программ рассчитаны на использование локального сервера WAMPserver 3.2 и выше [16], а также скриптового языка PHP версии 7.0 и старше [6-13].

Автор выражает благодарность студенткам Семенихиной Елизавете (гр. 5937) и Мироновой Елизавете (гр. 5038) за помощь в создании обложки.

УСТАНОВКА WAMPSEVER

1. Установка WAMPserver

WAMP-сервер [16] является реализацией классического LAMP-сервера, но для операционной системы (ОС) семейства Windows. В данный дистрибутив входит Apache, PHP, MySQL и автоматический установщик расширений. Рассмотрим основные действия перед установкой Wampserver 3 на ПК [17].

Во-первых, перед началом установки необходимо убедиться, что WAMP-сервер не установлен на Вашем ПК. Если установлен, но работает некорректно, то необходимо его деинсталлировать.

Во-вторых, перед установкой необходимо установить последние версии распространяемых пакетов: VC9 (Visual C ++ 2008), VC10 (Visual C ++ 2010 SP1), VC11 (Visual C ++ 2012, обновление 4), VC13 (Visual C ++ 2013), VC14 (Visual C ++ 2015) и VC15 (Visual C ++ 2017). VC16 (Visual C ++ 2015-2019) обратно совместим с VC14 и VC15. VC14 и VC15 в конечном итоге удаляются, но использование VC14 и VC15 все равно возможно.

Чтобы проверить, правильно ли установлены необходимые пакеты VC, можно использовать программу «Проверка установки пакетов VC++», доступную по ссылке <https://wampserver.aviatechno.net/#tools> или https://wampserver.aviatechno.net/files/tools/check_vccredist.exe.

Все пакеты VC++ можно загрузить из раздела «Распространяемые пакеты Visual C++» (<https://wampserver.aviatechno.net/#vcpackages>).

В-третьих, перед установкой, необходимо отключить или закрыть некоторые приложения:

- закрыть Skype;
- отключить Internet Information Services (IIS);
- закрыть пакеты Visual C++;
- отключить антивирус.

Поясним только те шаги установки Wampserver 3 на ПК [17], которые могут вызывать вопросы. В данном описании выполняется установка на ПК с 64-х битным процессором и ОС Windows, как наиболее распространенные на сегодня.

В начале, запускаем файл установки от имени администратора. Затем выбираем путь установки (рис. 1). Рекомендуется установить Wampserver в папку в корне диска, например «C:\wamp64\» или «D:\wamp64\». Не рекомендуется установка в «C:\Program Files\» или «C:\Program Files (x86)\». Также важно, чтобы на диске было свободно более 2,5 Гб.

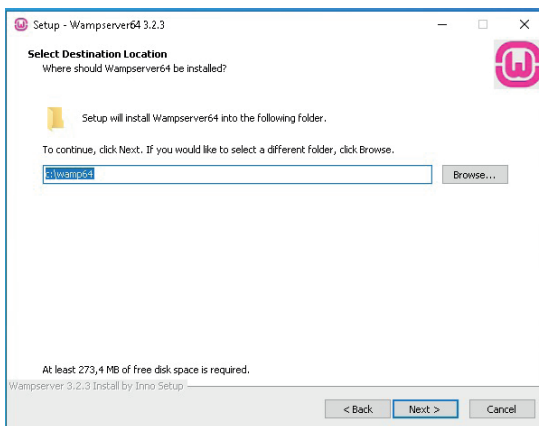


Рис. 1. Путь установки Wampserver 3

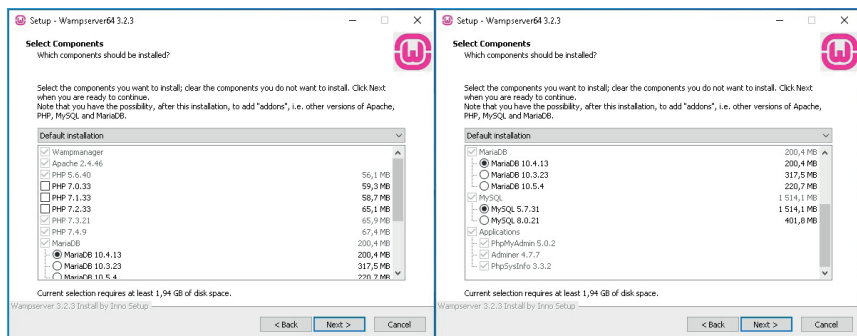


Рис. 2. Выбор компонентов Wampserver 3

Далее необходимо выбрать какие компоненты установить на ПК (для выполнения ЛР по дисциплине «Web-программирование» и «Электронный бизнес» следует установить WAMPmanager, Apache, PHP 7 или новее, MySQL, PhpMyAdmin) (рис. 2).

В процессе установки может появиться вопрос об использовании Internet Explorer в качестве браузера по умолчанию для WAMP-сервера (рис. 3). Если необходимо выбрать другой браузер, то необходимо нажать кнопку «Да». В этом случае будет необходимо указать exe-файл нового браузера.

В качестве примера выберем Google Chrome в качестве браузера по умолчанию для WAMP-сервера (рис. 4). Если потребуется из

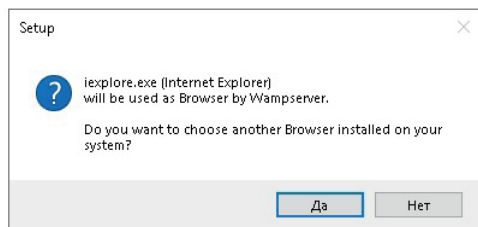


Рис. 3. Выбор браузера по умолчанию для Wampserver 3

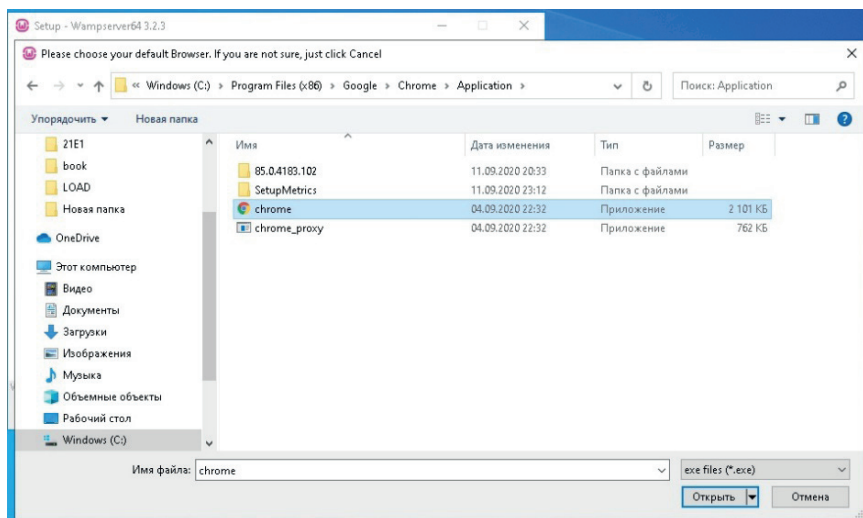


Рис. 4. Выбор Google Chrome в качестве браузера по умолчанию

менять данную настройку в последствии (в процессе использования WAMP-сервера), то необходимо открыть «C:\wamp64\wampmanager.conf». Найти раздел [main], в котором найти строку navigator="C:/Program Files/Internet Explorer/IEXPLORE.EXE" и заменить ее на navigator="C:/Program Files/Google/Chrome/Application/chrome.exe" (для Google Chrome). После этого изменения необходимо перезапустить WAMP-сервер.

В процессе установки может появиться вопрос об использовании Notepad (блокнот) в качестве текстового редактора по умолчанию для WAMP-сервера (рис. 5). Если необходимо выбрать другой текстовый редактор, то необходимо нажать кнопку «Да». В этом случае будет необходимо указать exe-файл нового текстового редактора. Если по-

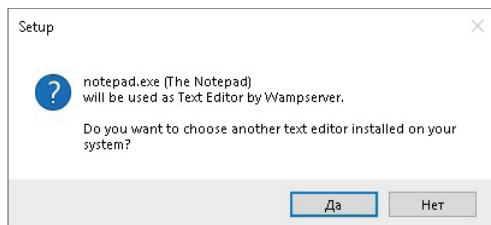


Рис. 5. Выбор текстового редактора по умолчанию для Wampserver 3

требуется изменить данную настройку в последствии (в процессе использования WAMP-сервера), то необходимо открыть «C:\wamp64\wampmanager.conf». Найти раздел [main], в котором найти строку editor="C:/Windows/notepad.exe" и заменить ее на editor="C:/Program Files (x86)/Notepad++/notepad++.exe" (для Notepad++). После этого изменения необходимо перезапустить WAMP-сервер.

Дожидаемся окончания процесса установки, а затем *рекомендуется перезагрузить ПК.*

2. Запуск WAMPserver и обзор основных компонентов

После двойного клика по ярлыку (запуск следует выполнять от имени администратора), сервер WAMP-запустится в системном трее (рис. 6). Следует помнить, что правила Брандмауэра Windows и антивируса должны разрешать подключение к 80 и 443 TCP-портам.

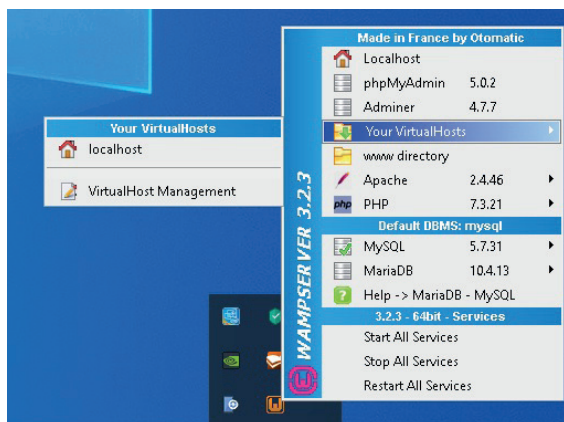


Рис. 6. Меню Wampserver 3

Если все прошло успешно, то значок Wampserver 3 в системном трее через 1-2 минуты станет зеленым.

Список виртуальных директорий, который будет создан пользователем для WAMP-сервера, будет размещаться в пункте меню «Your VirtualHosts». По умолчанию, в этой директории хранится localhost, который показан на рис. 7. Пункт меню «www directory» (рис. 6) содержит ссылку на директорию «www» в Wampserver 3, которая показана на рис. 8.

После добавления новых каталогов в папку «www» необходимо не забыть выполнить перезапуск WAMP-сервера (рис. 9) через контекстное меню Wampserver 3.

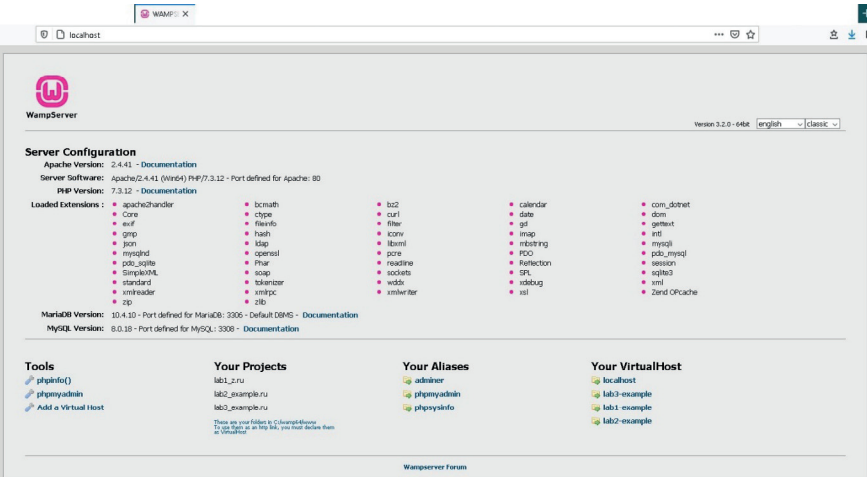


Рис. 7. localhost

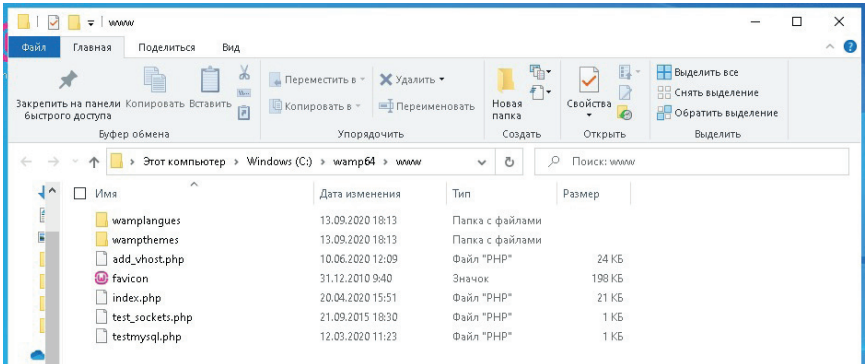


Рис. 8. Директория «www»

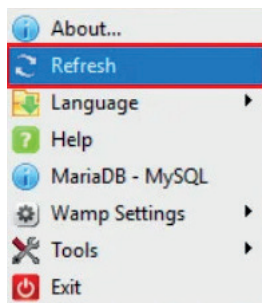


Рис. 9. Контекстное меню Wampserver 3

3. Настройка сервера Apache

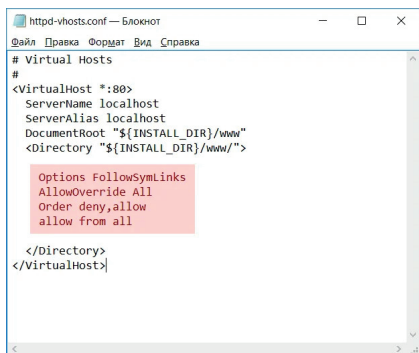
Для корректной работы сервера Apache необходимо разрешить подключение для всех пользователей, а не только для локальных пользователей. Отредактируем файл «C:\wamp64\bin\apache\apache2.4.41\conf\httpd.conf» (рис. 10).

```

httpd.conf — Блокнот
@файл  Правка  Формат  Вид  Справка
DocumentRoot "${INSTALL_DIR}/www"
<Directory "${INSTALL_DIR}/www/">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI Multiviews
#
# Note that "Multiviews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options +Indexes +FollowSymLinks +Multiviews
#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   AllowOverride FileInfo AuthConfig Limit
#
AllowOverride all
#
# Controls who can get stuff from this server.
#
# onlineoffline tag - don't remove
Require all granted
</Directory>

```

Рис. 10. Настройка сервера Apache (файл httpd.conf)



*Рис. 11. Настройка сервера Apache
(файл httpd-vhosts.conf)*

В данном конфигурационном файле необходимо найти секцию: `<Directory "${INSTALL_DIR}/www/">` и заменить значение «Require local» на «Require all granted». Сохраняем изменения и закрываем файл.

Теперь отредактируем файл «C:\wamp64\bin\apache\apache2.4.41\conf\extra\httpd-vhosts.conf» (рис. 11). Заменяем параметры и значения секции Directory на:

```
Options FollowSymLinks
AllowOverride All
Order deny,allow
allow from all
```

Сохраняем и закрываем файл. Перезапускаем WAMP-сервер. Для этого кликаем по значку в трее правой кнопкой мыши и выбираем «Refresh» (рис. 9).

4. Тестовый запуск первого сайта на виртуальном (локальном) хостинге

Для проверки работы www-сервера, создадим новый виртуальный тестовый сайт. Для этого необходимо выполнить следующую последовательность действий:

1. В директории «www» необходимо создать директорию с названием web-проекта (название дается на латыни по аналогии с именем домена), например, «test_website». В директории «test_website» будут храниться все рабочие файлы проекта (с расширением html, php, css и т.д.). По умолчанию при обращении к проекту

запускается файл с названием **index.php** или **index.html**, поэтому главный файл в директории с проектом должен иметь именно такое имя.

2. Создадим html-файл в корневой директории нового виртуального сайта. Для этого открываем блокнот и наполняем его html-кодом (листинг 1). Затем сохраним файл по пути «C:\wamp64\www\test_website\index.html».

3. Добавить директорию «test_website» на виртуальный (локальный) хостинг (рис. 12) через `http://localhost/add_vhost.php`. Заполнить поле «Name of the Virtual Host»: `tester` и «Complete absolute path of the VirtualHost folder»: `c:/wamp64/www/test_website`. Затем нажать кнопку «Start the creation of the VirtualHost».

4. Перезапустить WAMP-сервер (рис. 9), а сообщение об успешном выполнении показано на рис. 13.

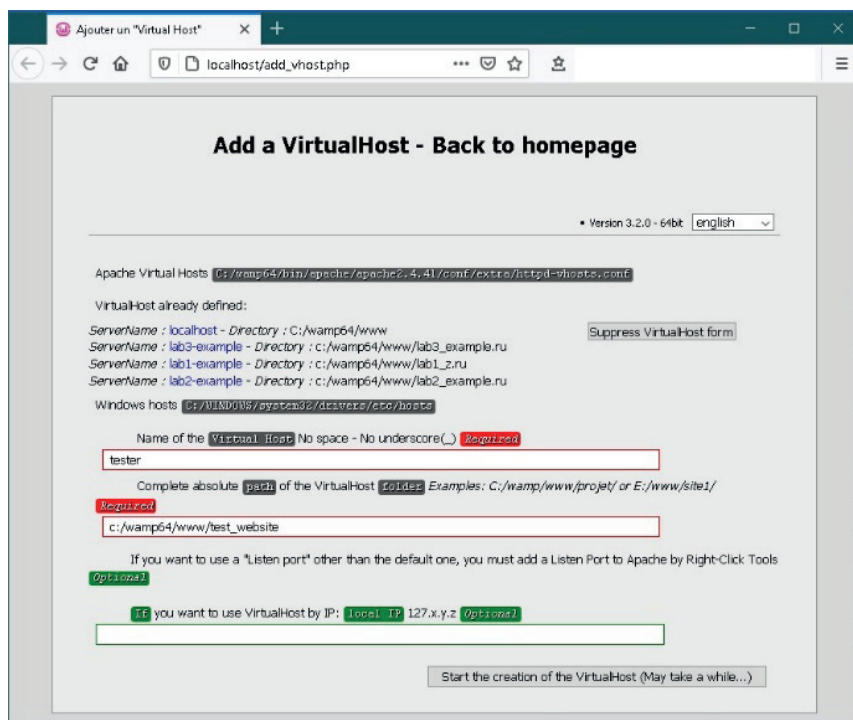


Рис. 12. Добавление директории на виртуальный хостинг

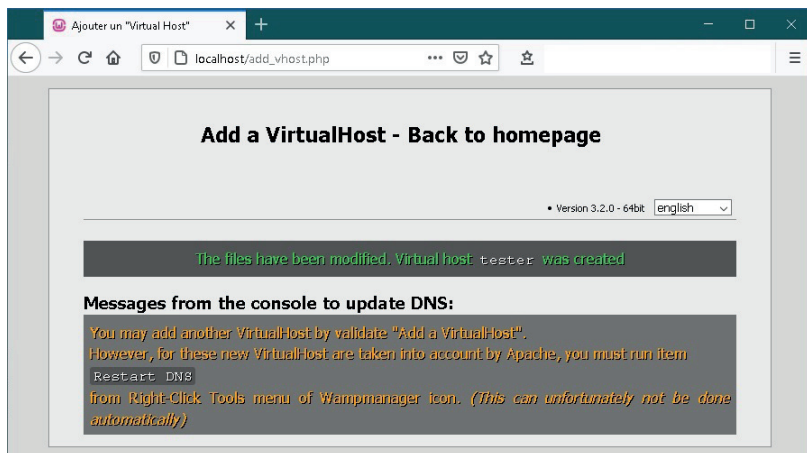


Рис. 13. Результат добавления директории на виртуальный хостинг

5. Открыть веб-браузер и ввести в адресную строку «tester» (рис. 14) или выбрать из списка виртуальных сайтов на localhost.

Листинг 1

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">
    <title>Тестовый сайт</title>
  </head>
  <body>
    <h2>Проверка!!!</h2>
  </body>
</html>
```

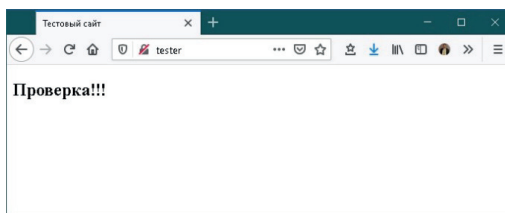


Рис. 14. Виртуальный тестовый сайт

Если на web-странице появилась надпись «Проверка!!!» (рис. 14), то все было сделано верно.

При использовании в ГУАП следует отключить подключение к wi-fi сети «guar-public». Также во время использования можно отключить антивирус.

Лабораторная работа № 1

ОБРАБОТКА ДАННЫХ HTML-ФОРМ В PHP

Цель работы – Получение навыков программирования на языке PHP. Изучение возможностей получения данных из HTML-формы и их обработки средствами PHP.

1.1. Краткие методические сведения

Одним из основных способов передачи данных web-сайту является обработка HTML-форм [18-19]. Формы представляют специальные элементы разметки HTML, которые содержат в себе различные элементы ввода – текстовые поля, кнопки и т.д. И с помощью HTML-форм можно ввести некоторые данные пользователя и отправить их на сервер. А сервер уже обрабатывает эти данные.

Создание HTML-форм состоит из следующих аспектов:

- создание элемента `<form>` в разметке HTML;
- добавление в этот элемент одного или нескольких полей ввода;
- установка метода передачи данных: GET или POST;
- установка адреса, на который будут отправляться введенные данные.

Методы передачи данных GET или POST

GET – это данные, которые передаются в адресной строке браузера, например, когда пользователь нажимает на ссылку.

POST – это данные, которые передаются после нажатия пользователем кнопки в HTML-форме.

Чтобы передать данные методом **GET** не надо создавать в HTML-странице HTML-форму – достаточно ссылки на документ с добавлением строки запроса, которая может выглядеть как пара «переменная=значение». Пары объединяются с помощью амперсанта «&», а к URL страницы строка присоединяется с помощью вопросительного знака «?». Преимущество передачи параметров таким способом заключается в том, что клиенты, которые не могут использовать метод **POST** (например, поисковые машины), все же смогут, просто пройдя по ссылке, передать параметры скрипту и получить содержимое. Поэтому метод **GET** следует использовать:

- для доступа к общедоступным страницам с передачей параметров (повышение функциональности);
- для передачи информации, не влияющей на уровень безопасности.

Недостаток же этого метода заключается в том, что просто изменив параметры в адресной строке, пользователь может изменить ход сценария непредсказуемым образом, что в свою очередь создает «дыру» в безопасности. Поэтому метод **GET не следует использовать**:

- для доступа к защищенным страницам с передачей параметров;
- для передачи информации, влияющей на уровень безопасности;
- для передачи информации, не подлежащей модифицированию пользователем.

Передать данные методом **POST** можно только с помощью HTML-формы на web-странице. Основное отличие **POST** от **GET** в том, что данные передаются не в заголовке запроса, а в теле запроса, следовательно, пользователь их не видит. Модифицировать данные возможно только изменив саму форму. **Преимущество** передачи данных методом **POST** заключается в большей безопасности данных и функциональности запросов с помощью HTML-форм в сравнении с методом **GET**. Поэтому метод **POST следует использовать**:

- для передачи большого объема информации (текст, файлы, и т.д.);
- для передачи важной информации;
- для ограничения доступа.

Какой способ следует применять?

• Если HTML-форма служит для запроса информации, например – при поиске, то ее следует отправлять методом **GET**. Чтобы можно было обновлять страницу, можно было поставить закладку и/или послать ссылку другому пользователю.

• Если в результате отправки HTML-формы данные записываются или изменяются на сервере, то следует их отправлять методом **POST**. Так же, **POST** может понадобиться, если на сервер надо передать большой объем данных (у **GET** объем передаваемых данных сильно ограничен), а также, если не следует отображать передаваемые данные в адресной строке (при вводе логина и пароля, например).

Подготовка к выполнению лабораторной работы.

1. Установить WAMPserver по инструкции, описанной в разделе «Установка WAMPserver».

2. Проверить корректность работы WAMPserver, для этого необходимо запустить браузер и в адресной строке набрать «localhost». Если искомая страница отобразилась, то – все хорошо.

3. В папке «www» создадим папку с названием web-проекта (название дается на латыни по аналогии с именем домена), например, «lab1-example.ru». В директории «lab1-example.ru» будут храниться все рабочие файлы проекта (*.html, *.css, *.php и т.д.). По умолчанию, при обращении к проекту запускается файл с названием

index.php или **index.html**. В рамках выполнения лабораторной работы создать при помощи текстового редактора файл **index.html**.

1.2. Пример

Для примера напомним web-страницу (файл **index.html**), содержащую html-форму для прохождения теста из 5 вопросов (листинг 2-3).

Листинг 2

index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">
    <title>Обработка html-формы</title>
  </head>
  <body>
    <div id="style1">
      <h1>Пример 1. Обработка html-формы</h1>
    </div>
    <div id="style2">
      <form action="test.php" method="POST" id="data">
        <p>Укажите перевод следующих слов
с английского языка на русский:</p>
        <ol class="rounded">
          <li>Слово "Apple" переводится
как</li>
          <div class="radio"><input
type="radio" name="question1" value="a1" form="data" />яблоко<br></div>
          <div class="radio"><input
type="radio" name="question1" value="a2" form="data"
/>дорога</div>
          <li>Введите перевод слова
"beautiful":</li>
          <input type="text"
name="question2" form="data" class="text" /><br>
          <li>Выберите правильный вариант
перевода слова "Engineer"</li>
          <div class="check"><input
```

```

type="checkbox" name="question3" value="a1" form="data"
/>электромеханик<br></div>
<div class="check"><input
type="checkbox" name="question3" value="a2" form="data"
/>инженер<br></div>
<div class="check"><input
type="checkbox" name="question3" value="a3" form="data"
/>садовод<br></div>
<li>Введите перевод слова
"Understand":</li>
<input type="text"
name="question4" form="data" class="text" /><br>
<li>"Sun" в переводе означает</li>
<div class="select"><select
id="s1" name="question5">
<option value="a1">
</option>
<option
value="a2">сон</option>
<option
value="a3">веселый</option>
<option
value="a4">солнце</option>
</select></div>
</ol>
<input type="submit" value="Проверить" />
<input type="reset" value="Очистить" />
</form>
</div>
</body>
</html>

```

В атрибуте action тега <form> прописываем имя скрипта, который будет выполнен при нажатии на активный элемент формы – кнопку. В примере скрипт назван «test.php». А атрибут method="POST" указывает, что в качестве метода передачи данных будет применяться метод POST.

Листинг 3

style.css

```

body {
    font-family: Arial, Verdana, sans-serif;

```

```

        font-size: 12pt;
        background-color: #f0f0f0;
        color: #333333;
    }
    h1 {
        color: #a52a2a;
        font-size: 24pt;
        font-weight: normal;
    }
    p {
        text-align: justify;
        margin-left: 25px;
        margin-right: 10px;
        border-bottom: 1px solid #999999;
        padding-bottom: 10px;
    }

    #style1 {
        position: absolute;
        left: 160px;
        top: 0px;
        width: 550px;
        padding: 5px;
        text-align: left;
    }
    #style2 {
        position: absolute;
        left: 160px;
        top: 75px;
        width: 550px;
        height: 700 px;
        padding: 5px;
        background: #ffff00;
        text-align: left;
        border: 1px solid #cecece;
        box-shadow: inset 0px 20px 20px #ffffff;
        border-radius: 8px;
    }

    .rounded {
        counter-reset: li;

```

```

    list-style: none;
    font: 14px "Trebuchet MS", "Lucida Sans";
    padding: 0;
    text-shadow: 0 1px 0 rgba(255,255,255,.5);
}
.rounded li {
    position: relative;
    display: block;
    padding: .4em .4em .4em 2em;
    margin: .5em 0;
    background: #DAD2CA;
    color: #444;
    text-decoration: none;
    border-radius: .3em;
    transition: .3s ease-out;
}
.rounded li:before {
    content: counter(li);
    counter-increment: li;
    position: absolute;
    left: -1.3em;
    top: 50%;
    margin-top: -1.3em;
    background: #8FD4C1;
    height: 2em;
    width: 2em;
    line-height: 2em;
    border: .3em solid white;
    text-align: center;
    font-weight: bold;
    border-radius: 2em;
    transition: all .3s ease-out;
}
input.text {
    width: 300px;
    font-size: 13px;
    padding: 6px 0 4px 10px;
    border: 1px solid #cecece;
    background: #F6F6f6;
    border-radius: 8px;
}

```

```
.radio, .check, .select {
    display: block;
    height: 25px;
    position: relative;
    left: 30px;
}
```

При запуске страницы в браузере получим HTML-форму, показанную на рис. 1.1.

4. Создать скрипт test.php (листинг 4), проверяющий полноту ввода ответов на вопросы и правильность выбора ответов из предложенных вариантов. Сам файл скрипта создать при помощи текстового редактора. Файл сохранить с расширением php (рекомендуемая кодировка «UTF-8»).

lab1-example/index.html

Пример 1. Обработка html-формы

Укажите перевод следующих слов с ангийского языка на русский:

- Слово "Apple" переводится как
 - ☐ яблоко
 - ☐ дорога
- Введите перевод слова "beautiful":
- Выберите правильный вариант перевода слова "Engineer"
 - ☐ электромеханик
 - ☐ инженер
 - ☐ садовод
- Введите перевод слова "Understand":
- "Sun" в переводе означает

Рис. 1.1. HTML страница с тестом

Листинг 4

test.php

```
<?php
    // получение данных из формы
    if(empty($_POST["question1"])) {
        $q1="";
    } else {
        $q1=$_POST["question1"];
    }
    $q2=$_POST["question2"];
    if(empty($_POST["question3"])) {
        $q3="";
    } else {
        $q3=$_POST["question3"];
    }
    $q4=$_POST["question4"];
    $q5=$_POST["question5"];
    $count=0;    // счетчик правильных ответов
    if ($q1==" or $q2==" or $q3==" or $q4==" or $q5=="
{
    // вывод сообщения, если какие-то из полей формы
не заполнены
    echo "<!DOCTYPE HTML><html><head><title>Форма
не заполнена!</title></head>";
    echo "<body><div style='background: #ff4d00;
width: 200px; height: 150px; text-align: center;'>";
    echo "<p style='text-align: center; font-weight:
600;'>Внимание!<br>";
    echo «<p style='text-align: center;'>Заполните
ответы на вопросы №:";
        if ($q1==" ) echo " 1";
        if ($q2==" ) echo " 2";
        if ($q3==" ) echo " 3";
        if ($q4==" ) echo " 4";
        if ($q5=="a1") echo " 5";
        echo ".<br><br><a href='index.html'>Вернуться
к тесту</a></div></body></html>";
    } else {
        // проверка введенных ответов
        if($q1=="a1") $count++;
        if($q2=="красивый") $count++;
```

```

        if($q3=="a2") $count++;
        if($q4=="понимать") $count++;
        if($q5=="a4") $count++;
        // формирование сообщения с результатами теста
        в зависимости от количества набранных баллов
        if($count<2) $str="Ваш результат - 1 балл.<br>
<br>Вы - хрустальная Сова.<br>Помните, мудр не тот, кто
не совершает ошибок, а тот, кто их осознает. Проверьте,
не ошиблись ли вы с выбором языка для теста?";
        if($count==2) $str="Ваш результат - 2 балла.<br>
<br>Вы - Веселый Роджер.<br>Карамба! После битвы у Лысых Скал
вы остались без переводчика. Вам непросто, но настоящий
пиратский капитан должен сам понимать, что кричат
эти французы, испанцы, англичане на его допросах.";
        if($count==3) $str="Ваш результат - 3 балла.<br>
<br>Вы - Мозговой Штурман.<br>Скорее всего, у вас хорошая
интуиция и сообразительность. Если к этому добавится еще
уверенное знание слов - цены вам не будет.";
        if($count==4) $str="Ваш результат - 4 балла.<br>
<br>Вы - Умная Маша.<br>Она тоже была хорошистой. Для \"
пятерок\" ей не хватало немного удачи и прилежности.
В следующий раз вам должно повезти больше.";
        if($count==5) $str="Ваш результат - 5 баллов.<br>
<br>Вы - Мудрый Каа.<br>Иностранные слова для вас - легкая
добыча. Еще бы, охота на кроликов и мышей сделала вас точным
и сообразительным. Да пребудет с вами сила.";
        // вывод сообщения с результатами теста
        echo «<!DOCTYPE HTML><html><head><title>Результ
ат прохождения теста</title></head><body><div style='position:
absolute; margin-top: 20px; margin-left: 0px;'><img src='img/'.
$count.'.jpg'></div><div style='position: absolute; margin-top:
20px; margin-left: 150px; background: orange; width: 400px;
height: 200px;'>";
        echo "<p style='text-align: left;'>\".$str.</
p>\".<br><a href='index.html'>Пройти еще раз тест</a><br></
div></body></html>";
    }
?>

```

Результат некорректно заполненного теста представлен на рис. 1.2.

Результат успешного прохождения теста по всем 5 вопросам представлен на рис. 1.3.

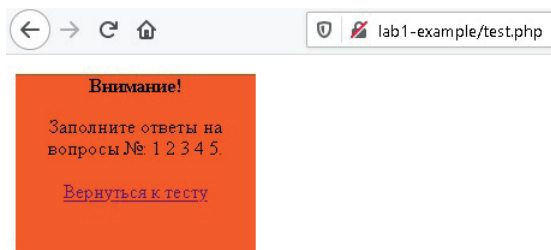


Рис. 1.2. Страница с некорректно заполненным тестом

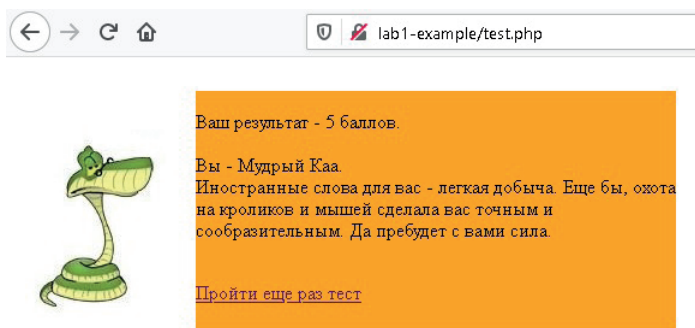


Рис. 1.3. Страница с результатом прохождения теста

Чтобы получить данные формы, используется глобальная переменная `$_POST`. Она представляет ассоциативный массив данных, переданных с помощью метода POST. Используя ключи, мы можем получить отправленные значения. Ключами в этом массиве являются значения атрибутов `name` у полей ввода формы.

Так как атрибут `name` поля ввода логина имеет значение `login` (`<input type="text" name="login" />`), то в массиве `$_POST` значение этого поля будет представлять ключ `«login»: $_POST['login']`

И поскольку возможны ситуации, когда поле ввода будет не установлено, например, при прямом переходе к скрипту: `http://localhost:8080/login.php`. В этом случае желательно перед обработкой данных проверять их наличие с помощью функции `isset()`. И если переменная установлена, то функция `isset()` возвратит значение `true`.

1.3. Задание

1. Создать папку с проектом, содержащую страницу с HTML-формой, каскадной таблицей стилей и php-скриптом.

2. Разработать web-страницу, которая должна предлагать пользователю HTML-форму, включающую различные элементы (текстовые поля, списки, кнопки и т.д.). HTML-форма должна содержать не менее 10 различных элементов. HTML-форма должна иметь дизайн, разработанный средствами CSS. Тематика, для которой создается форма, определяется в соответствии с приложением А. HTML-форма должна иметь обязательные поля (`input type="text"`) в соответствии с разделом «Варианты для лабораторных работ».

РЕКОМЕНДУЕТСЯ использовать форму из Лабораторной работы № 5 «Создание форм в web-документах» по дисциплине «Web-технологии».

3. Разработать PHP-скрипт, который получает данные из HTML-формы методом POST и, в зависимости от полученного содержимого, формирует новую страницу. Итоговая страница должна содержать *текстовые и графические элементы*, связанные с результатом заполненной HTML-формы. **Количество элементов новой страницы должно быть не меньше количества полей в HTML-форме. Для поля типа radio или checkbox предусмотреть возможность изменения минимум одного изображения в HTML-форме в зависимости от выбранного пользователем значения поля.**

4. PHP-скрипт должен проверить полноту введенных пользователем данных. Также полученная с помощью PHP-скрипта страница должна иметь разработанный дизайн средствами CSS.

Примечания к выполнению лабораторной работы, содержанию и требованиям к оформлению отчета приведены в приложении Б.

Лабораторная работа № 2

ПРОВЕРКА ДАННЫХ HTML-ФОРМ В PHP

Цель работы – Изучение возможностей проверки данных из HTML-форм и изучение регулярных выражений в PHP.

2.1. Краткие методические сведения

На любом современном сайте используются всевозможные HTML-формы, которые пользователь должен заполнить и отправить данные на сервер. Это может быть гостевая книга, форум, форма обратной связи, подписка на рассылку и т.д.

Данные, вводимые пользователем в форму и отправляемые в файл-обработчик, необходимо проверять на корректность. Причем, при проверке корректности данных, вводимых пользователем, необходимо уделять большое внимание, поскольку необработанные ошибки, возникающие при неправильном вводе данных, приводят к ошибкам в работе php-скрипта, и зачастую катастрофическим. Предположим, что разработчик создал html-форму для отправки пользователем письма, при этом адрес электронной почты (e-mail) необходимо вводить пользователю. В этом случае, для корректной работы программы необходимо сделать следующее:

- проверить, что поле, в которое заносится адрес электронной почты, не пустое (т.к. пользователь может просто забыть ввести адрес);
- проверить соответствие введенного адреса с помощью регулярного выражения.

E-mail обязательно должен содержать символ «@», быть по длине не менее 8 символов, есть также регулярное выражение, по которому можно проверить данные на предмет соответствия адресу электронной почты.

После обнаружения ошибки необходимо проинформировать пользователя об этом. Один из вариантов решения этой задачи – через сессионные переменные (этот способ пока для информации, т.к. тема «Сессии» будет рассмотрена позже). Пусть есть страница **form.php**, на которой расположена web-форма, и есть скрипт **send.php**, который является обработчиком данных web-формы в файле **form.php**. В обоих файлах должна быть запущена сессия. Другой вариант – с помощью регулярных выражений и каскадных таблиц стилей.

Проверка корректности адреса электронной почты часто осуществляется с помощью регулярных выражений. Как известно, у

адреса две составляющие – имя пользователя и имя домена, которые разделены знаком «@». В имени пользователя могут присутствовать заглавные и прописные буквы, цифры, знаки подчеркивания, минуса и точки. Для проверки разделителя между именем пользователя и именем домена, в выражение требуется добавить «+@».

Также не следует забывать, что электронный почтовый ящик может находиться на поддомене. Поэтому, в регулярном выражении не следует забывать использовать точку (экранированную «\.») для указания того, что часть адреса после «@» может содержать точку, как разделитель доменных имен.

Таким образом, регулярное выражение, проверяющее имя пользователя и наличие разделителя имеет следующий вид:

```
«/^[ -A-Za-z0-9 _ \.]+@[ -A-Za-z0-9^\. ]»
```

Для проверки доменного имени первого уровня учитываем, что его длина уже составляет от 2 до 6 символов. Поэтому добавляем такое выражение: «\.[a-z]{2,6}\$/i»

Объединяя эти шаги, получаем следующее регулярное выражение для проверки адресов электронной почты:

```
«/^[ -A-Za-z0-9 _ \.]+@[ -A-Za-z0-9^\. ]+\.[a-z]{2,6}$/i»
```

Проверка e-mail с помощью регулярных выражений может осуществляться по шаблону с применением функции **preg_match()** [20]:

```
function check_email ($email) { // Проверка корректности
e-mail
    if(!preg_match('|^[ -0-9A-Za-z _ \.]+@[ -0-9A-Za-z^\. ]+\.[a-z]{2,6}$|i',$email)) {
        return false;
    }
    return true;
}
```

Эта пользовательская функция **check_email** возвращает true, если переданное значение переменной **\$email** соответствует шаблону и **false** в противном случае. В итоге проверка на корректность будет выглядеть так:

```
if(!check_email ($_POST["email"])) {
    $_SESSION["error"]="Не верный адрес электронной почты";
    header("Location: form.php");
    exit;
}
```

Соответственно в файле **form.php** перед формой прописываем следующее:

```
if(!empty($_SESSION["error"])) {  
    $error=$_SESSION["error"];  
    unset($_SESSION["error"]);  
} else {  
    $error="";  
}  
echo $error;
```

Поэтому, если ошибка была допущена, то сообщение об ошибке будет напечатано пользователю перед web-формой. Таким же методом можно сохранять данные в заполненных полях формы, чтобы пользователь по несколько раз не вводил одно и то же.

Функция **preg_match**

preg_match – выполняет проверку на соответствие регулярному выражению (ищет в заданном тексте *subject* совпадения с шаблоном *pattern*).

preg_match (string \$pattern, string \$subject [, array &\$matches [, int \$flags = 0 [, int \$offset = 0]]]): int,

где

- *pattern* – искомый шаблон в виде строки;
- *Subject* – входная строка;
- *Matches* – если указан дополнительный параметр *matches*, то он будет заполнен результатами поиска (элемент *\$matches[0]* будет содержать часть строки, соответствующую вхождению всего шаблона, *\$matches[1]* – часть строки, соответствующую первой подмаске и так далее);

- *flags* может быть комбинацией следующих флагов:

- **PREG_OFFSET_CAPTURE** – если этот флаг указан, то для каждой найденной подстроки будет указана ее позиция (в байтах) в исходной строке;

- **PREG_UNMATCHED_AS_NULL** – если этот флаг передан, то несопадающие подмаски будут представлены значениями **NULL**; в противном случае они отображаются в виде пустых строк (**string**);

- *Offset* – по умолчанию поиск осуществляется слева направо, с начала строки, но можно использовать дополнительный параметр *offset* для указания альтернативной начальной позиции для начала поиска (в байтах).

preg_match() возвращает **1**, если параметр *pattern* соответствует переданному параметру *subject*, иначе – **0**.

2.2. Пример

Рассмотрим пример html-формы (листинг 5) с одним обязательным полем и проверкой web-формы в PHP [18,21] на выполнение этого условия в php с помощью функции `preg_match`.

Листинг 5

index.php

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style.css">
        <title>Проверка html-формы</title>
    </head>
    <body>
        <?php
            $nameErr = ""; // сообщение об ошибке
            $name = "";    // данные из поля
web-формы
                                // если данные из web-формы были переданы
методом post
                                if ($_SERVER["REQUEST_METHOD"] =
= "POST") {
                                    if (empty($_POST["name"])) {
                                        // если поле не заполнено,
то формируем сообщение об ошибке
                                        $nameErr = "Имя
обязательно";
                                    } else {
                                        // если поле заполнено, то
получаем данные из web-формы
                                        $name = test_input($_
POST["name"]);
                                    }
                                }
                                // проверка входных данных, полученных из
web-формы
                                function test_input($data) {
                                    $data = trim($data);
                                    $data = stripslashes($data);
                                    $data = htmlspecialchars($data);
```

```

        return $data;
    }
?>
<h2>Пример проверки формы PHP</h2>
<!-- вывод информационного сообщения -->
<span class="error">* обязательное поле
</span><br><br>
<!-- вывод содержимого -->
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>"
    <span>Имя: </span><input type="text"
name="name" />
    <!-- вывод содержимого переменной
об ошибке -->
    <!-- если переменная пуста, то
на web-странице ничего не появится -->
    <!-- если переменная не пуста, то
на web-странице появится сообщение об ошибке -->
    <span class="error"> * <?php echo
$nameErr;?></span><br><br>
    <input type="submit" name="submit"
value="Отправить" />
</form>
<?php
    echo "<h2>Ваш ввод: </h2>". $name. "<br>";
?>
</body>
</html>

```

style.css

```

.error {
    color: #FF0000;
}
body {
    font-family: Arial, Verdana, sans-serif;
    font-size: 12pt;
    background-color: yellow;
}

```

Исходная HTML-форма показана на рис. 2.1. Сообщение об ошибке показано на рис. 2.2. Результат проверки web-формы представлен на рис. 2.3.

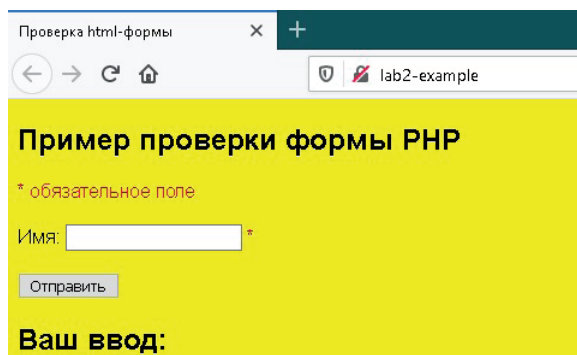


Рис. 2.1. Исходная страница

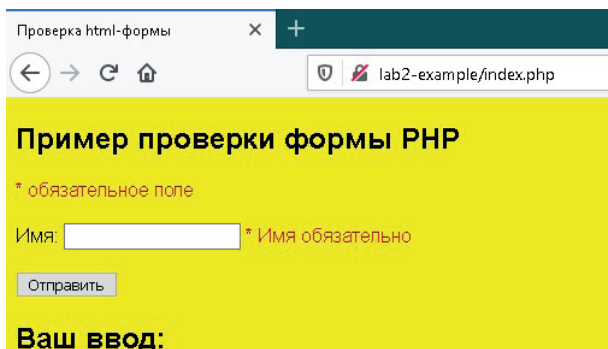


Рис. 2.2. Страница, заполненная с ошибкой

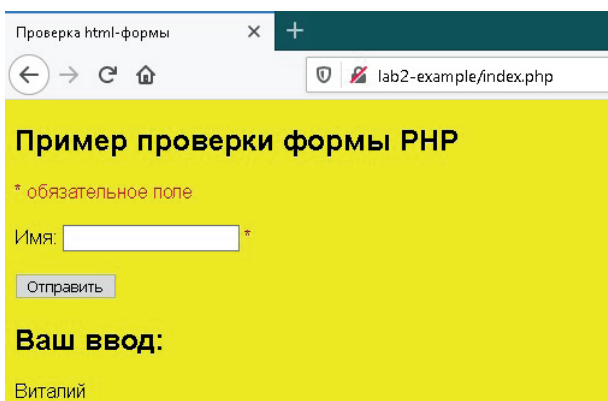


Рис. 2.3. Страница с результатом (заполнена без ошибок)

Безопасный запрос в поле action на проверку данных html-формы:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

где

- функция htmlspecialchars() преобразует специальные символы в HTML сущность. Это означает, что функция заменит HTML символы, такие как "<" и ">" на < и > соответственно, что предотвращает использование кода злоумышленниками путем ввода кода HTML или JavaScript (межсайтовые скриптовые атаки) в формах;

- \$_SERVER["PHP_SELF"] – суперглобальная переменная, которая возвращает имя текущего выполняемого скрипта.

Также сразу поясним назначение 1-й и 2-й функций в функции test_input(\$data):

- trim(\$data) – удаляет ненужные символы (дополнительное пространство, вкладку, новую строку) из пользовательских входных данных);

- stripslashes(\$data) – удаляет обратную косую черту из пользовательских входных данных.

Назначение 3-й функции в функции test_input(\$data) пояснено выше.

В приведенном ниже коде показан простой способ (доработать код выше) проверить, содержит ли поле «Имя:» только буквы и пробелы с помощью функции preg_match(). Если значение поля имени недопустимо, сохраняется сообщение об ошибке:

```
$name = test_input($_POST["name"]);  
if (!preg_match("/^[a-яА-Я ]*$/", $name)) {  
    $nameErr = "Разрешены только буквы и пробелы";  
}
```

Для выполнения этой лабораторной работы ОБЯЗАТЕЛЬНО настроить сервер Apache. Инструкция приведена в разделе «Установка WAMPserver».

2.3. Задание

Используя web-страницу, полученную в ходе выполнения лабораторной работы 1, дополнить ее следующими возможностями.

1. Разработать PHP-скрипт, который с помощью функции preg_match будет проверять правильность заполнения данных в HTML-

Таблица 2.1

Шаблоны заполнения обязательных полей

Поле	Шаблон заполнения обязательного поля
логин	Шаблон заполнения поля: ____ : ____ ^ ____ * ____ . Первый блок (любое количество символов) – только цифры, второй блок (4 символа) – только буквы русского алфавита в диапазонах [А; Р] и [к; э], третий блок (любое количество символов) – любые символы. Последний блок (2 символа) – только английские буквы в диапазоне [e; m].
адрес доставки	Шаблон заполнения поля: ____ , ____ @ ____ ! ____ . Общая длина поля – 10-50 символов. Первый блок (6 символов) – только цифры, второй блок (до 10 символов) – только буквы русского алфавита, третий блок (до 15 символов) – только прописные (заглавные) буквы. Последний блок (любое количество символов) – только английские буквы в диапазонах [А; Х], [с; у] и {-, /}.
e-mail	Шаблон заполнения поля: ____ . ____ @ ____ . ____ . При этом первый блок – только символы русского алфавита, второй блок – любые символы, третий блок – только символы английского алфавита и цифры, длина блока до 10 символов. Последний блок – только буквы, длина блока три символа.
промокод	Шаблон заполнения поля: ____ - ____ ~ ____ \ ____ . При этом первые четыре символа – цифры, следующие пять – только буквы в диапазонах [А; М] и [n; z]. Последние два символа также цифры.
мобильный телефон	Шаблон заполнения номера телефона: 8 - ____ - ____ . ____ . ____ . Заполнение поля должно производиться только с помощью цифр.
номер паспорта	Длина поля – 10 символов. Первый блок (2 символа) – только четные цифры, второй блок (2 символа) – только цифры {3, 5, 8}, третий блок (3 символа) – только цифры [3; 7]. Последний блок (3 символа) – только цифры {2, 4, 6}.
номер банковской карты	Длина поля – 16 символов. Первый блок (4 символа) – только нечетные цифры, второй блок (4 символа) – только цифры [5; 9], третий блок (4 символа) – только четные цифры. Последний блок (4 символа) – только цифры [0; 4].
имя	Поле должно содержать от 7 до 16 символов и только следующие символы [D; S], [r; z] и [3; 6].
фамилия	Поле должно содержать от 3 до 12 символов и только следующие символы [Q; Y], [с; k] и {?, >, %}.
номер бонусной карты	Поле должно содержать до 20 символов и только следующие символы [А; Z], [a; я] и {&, #, \$}.

форме в зависимости от условий из табл. 2.1 для обязательных полей из лабораторной работы 1. В случае верного ввода всех данных должна происходить передача данных на вторую страницу из лабораторной работы 1. В случае обнаружения ошибки с помощью функции `preg_match` у соответствующего поля необходимо вывести подсказку с информацией о правильном формате ввода данных для соответствующего поля.

2. Добавить в web-страницу поле «примечание» или «комментарий» (тэг `textarea`) и посчитать количество символов в поле.

Примечания к выполнению лабораторной работы, содержанию и требованиям к оформлению отчета приведены в приложении Б.

Лабораторная работа № 3

РАБОТА С ФАЙЛАМИ В PHP

Цель работы – Изучить функции языка PHP для работы с файлами (файловые функции).

3.1. Краткие методические сведения

Основные сведения о работе с файлами в PHP рассмотрены в [22-24]. Некоторые правила, которым нужно следовать для работы с файлами в HTML-форме (без этих требований загрузка файла работать не будет):

- убедиться, что HTML-форма использует метод POST (method="post");
- форма должна содержать следующий атрибут enctype="multipart/form-data", который указывает, какой тип контента использовать при отправке формы;
- атрибут type="file" тега <input> показывает поле ввода, как элемент управления выбора файла в браузере.

Основные переменные, используемые в PHP-скрипте при работе с файлами:

- \$target_dir = "uploads/" – указывает каталог, в который будет помещен файл;
- \$target_file – указывает путь к загружаемому файлу;
- \$uploadOk=1 – триггер для проверки наличия ошибок в ходе загрузки файла;
- \$imageFileType – содержит расширение файла (в нижнем регистре).

3.2. Пример

Пример кода web-страницы (листинг 6 и 7) для загрузки файлов показан ниже, а результат на рис. 3.1–3.3.

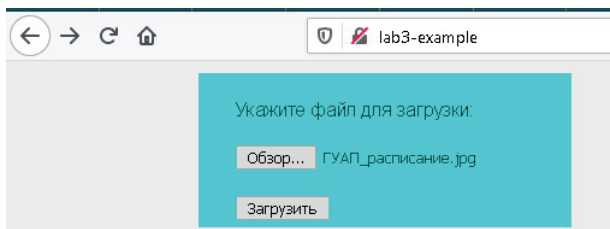


Рис. 3.1. Страница для загрузки файлов

Листинг 6

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style.css">
        <title>Загрузка файла</title>
    </head>
    <body>
        <div id="style1">
            <form action="upload.php" method="post"
name="upload" enctype="multipart/form-data">
                <p>Укажите файл для загрузки:</p>
                <input type="file"
name="fileToUpload"><br>
                <input type="submit"
value="Загрузить" name="button">
            </form>
        </div>
    </body>
</html>
```

Форма на рис. 3.1 отправляет данные в файл с именем "upload.php».

Листинг 7

style.css

```
body {
    font-family: Arial, Verdana, sans-serif; /* Семейство
шрифтов */
    font-size: 12pt; /* Размер основного шрифта в пунктах */
    background-color: #f0f0f0; /* Цвет фона веб-страницы */
    color: #333333; /* Цвет основного текста */
}
p {
    text-align: justify;
    margin-left: 25px;
    margin-right: 10px;
    margin-bottom: 0px;
}
```

```
#style1 {
    position: absolute; /* Абсолютное позиционирование */
    left: 160px; /* Положение элемента от левого края */
    top: 10px; /* Положение от верхнего края */
    width: 300px;
    height: 150 px;
    padding: 5px; /* Поля вокруг текста */
    background-color: #00f0f0; /* Цвет фона веб-страницы */
    text-align: left;
}
input {
    text-align: justify;
    margin-left: 25px;
    margin-right: 10px;
    margin-top: 20px;
}
#style2 {
    position: absolute; /* Абсолютное позиционирование */
    left: 0px; /* Положение элемента от левого края */
    top: 10px; /* Положение от верхнего края */
    width: 300px;
    height: 150 px;
    padding: 5px; /* Поля вокруг текста */
    background-color: yellow; /* Цвет фона веб-страницы */
    text-align: left;
}
```

Пример php-скрипта (листинг 8) для загрузки файлов на сервер и выполнения различных проверок перед этим показан ниже.

Листинг 8

upload.php

```
<html>
    <head>
        <meta charset="utf-8">
        <title>Loading...</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <div id="style2">
            <?php
                $target_dir = "uploads/";
```

```

// Проверим директорию для загрузки.
if (!is_dir($target_dir)) {
    mkdir($target_dir, 0777,
true);
}
$target_file = $target_dir.
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType =
strtolower(pathinfo($target_file, PATHINFO_EXTENSION));
// Проверить, существует ли файл
if (file_exists($target_file)) {
    echo "Извините, файл уже
существует.\n";

    $uploadOk = 0;
}
// Проверить размер файла
if ($_FILES["fileToUpload"]
["size"] > 500000) {
    echo "Извините, ваш файл
слишком большой.\n";

    $uploadOk = 0;
}
// Разрешить определенные форматы
файлов
if($imageFileType != "jpg" &&
$imageFileType != "png" && $imageFileType != "jpeg" &&
$imageFileType != "gif" ) {
    echo "Извините, разрешено
только файлы JPG, JPEG, PNG и GIF.\n";
    $uploadOk = 0;
}
// Проверка, имеет ли $uploadOk
значение 0 (ошибка при загрузке)
if ($uploadOk == 0) {
    echo "\nИзвините, ваш файл
не был загружен.";

} else {
    if (move_uploaded_
file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "Файл ".

```

```

basename( $_FILES["fileToUpload"]["name"]). " был загружен.";
        } else {
            echo "К сожалению,
произошла ошибка при загрузке файла.";
        }
    }
    ?>
</div>
</body>
</html>

```

Возможные ошибки при работе с файлами в PHP:

- загрузка файлов может быть отключена в настройках PHP директивой `file_uploads`;
- не загружаются файлы большого размера, причина в ограничениях хостинга. Следует проверить в `phpinfo()` значения директив:
 - `upload_max_filesize` – максимальный размер загружаемого файла;
 - `max_file_uploads` – максимальное количество одновременно загружаемых файлов;
 - `post_max_size` – максимально допустимый размер данных, отправляемых методом POST (его значение должно быть больше `upload_max_filesize` или равно ему);
 - `memory_limit` – значение должно быть больше, чем `post_max_size` или равно ему.

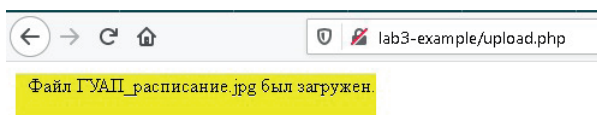


Рис. 3.2. Успешная загрузка файла

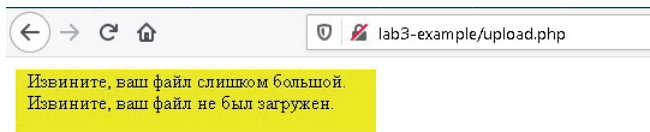


Рис. 3.3. Неудачная загрузка файла

3.3. Задание

Используя web-страницу, полученную в ходе выполнения лабораторной работы 2, дополнить ее следующими возможностями.

1. Написать php-скрипт, позволяющий загружать файл на сервер, который содержит дополнительную информацию по предметной области лабораторной работы 1 (приложение А). Скрипт должен выводить сообщение с результатом загрузки (успех/неуспех и информацией об ошибке). *Необходимо реализовать дополнительным полем в HTML-форме (интегрировать в форму).*

2. Дополнить php-скрипт из пункта 1 возможностью проверки типа и размера загружаемого файла. Тип файла определяется по номеру варианта из табл. 3.1, а размер файла – из табл. 3.2.

3. Добавить в получившуюся в пункте 1 web-страницу кнопку, которая позволяет записать всю текстовую информацию из полей формы с названиями полей формы в текстовый файл (можно *.txt). Для работы кнопки необходимо дополнить php-скрипт из пункта 2. После сохранения информации в файл форма должна предложить просмотреть/скачать получившийся файл. Когда скачивание будет завершено, необходимо выдать сообщение согласно номеру варианта из табл. 3.3. *Текст необходимо структурировать.*

Таблица 3.1

Перечень типов файлов

№ варианта	Тип файла	№ варианта	Тип файла
1	.doc	6	.rtf
2	.mp3	7	.wav
3	.flv	8	.avi
4	.bmp	9	.jpg
5	.7z	10	.zip

Таблица 3.2

Перечень результирующих сообщений

№ варианта	Размер файла
1, 6	50 – 100 кб
2, 7	1 – 5 мб
3, 8	10 – 20 мб
4, 9	100 – 200 кб
5, 10	5 – 10 мб

Таблица 3.3

Перечень результирующих сообщений

№ варианта	Содержание сообщения
1, 6	Размер файла
2, 7	Время последнего изменения файла
3, 8	Тип файла
4, 9	Время последнего доступа к файлу
5, 10	Права доступа к файлу

Примечания к выполнению лабораторной работы, содержанию и требованиям к оформлению отчета приведены в приложении Б.

Лабораторная работа № 4

РАБОТА С БАЗОЙ ДАННЫХ MySQL В PHP

Цель работы – Изучить основные функции языка PHP для работы с базой данных MySQL с помощью расширения MySQLi.

4.1. Краткие методические сведения

Работа с реляционными базами данных (БД) рассмотрена в [25]. Реляционные БД и, в частности, MySQL широко применяются на практике, например, в мобильных устройствах под управлением ОС Android [26].

Расширение MySQLi позволяет получить доступ к функциональности, которую предоставляет MySQL версии 4.1 и выше.

MySQLi (MySQL Improved) [14] – это расширение PHP, которое добавляет в язык полную поддержку БД MySQL. Это расширение поддерживает множество возможностей современных версий MySQL.

Типичный процесс работы с БД в PHP-сценарии состоит из нескольких шагов:

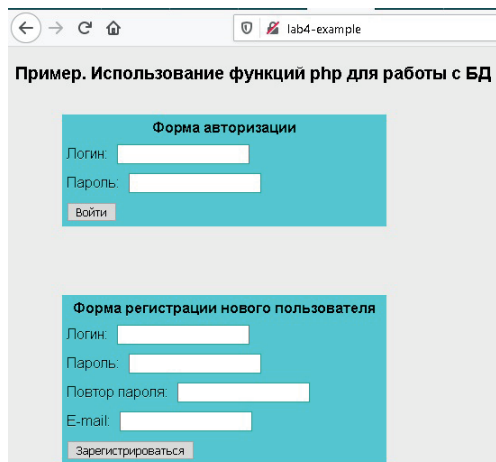
1. Установить подключение к серверу БД, передав необходимые параметры: адрес, логин, пароль, имя БД и порт.
2. Убедиться, что подключение прошло успешно: сервер БД доступен, логин и пароль верные и так далее.
3. Сформировать SQL запрос (например, на чтение данных из таблицы БД).
4. Убедиться, что запрос был выполнен успешно.
5. Получить результат от БД в виде массива из записей.
6. Использовать полученные записи в своем php-скрипте (например, показать их в структурированном виде).

4.2. Пример

Пример работы с БД MySQL с помощью расширения MySQLi представлен ниже, а создание учебного хостинга на web-сервере и подключение к нему БД MySQL показан в [15].

Пример web-страницы с формой авторизации и регистрации пользователя

Страница с примером содержит 2 формы (рис. 4.1): HTML-форму авторизации и форму регистрации пользователя. В атрибутах тегов <form action=""> прописаны имена php-скриптов регистрации (register.php)



*Рис. 4.1. Формы авторизации (сверху)
и регистрации (снизу)*

и авторизации (enter.php) пользователей. Код этой страницы и каскадной таблицы стилей для всех страниц показан в листинге 9.

Листинг 9

index.html

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style44.css">
    <title>Работа с базой данных MySQLi</title>
</head>
<body>
    <h3>Пример. Использование функций php для работы с БД</h3>
    <!-- Форма авторизации -->
    <div class="form_div">
        <form action="enter.php" method="post">
            <p id="style2">Форма авторизации</p>
            <p class="pinput">Логин:</p><input
type="text" name="login" /><br>
            <p class="pinput">Пароль:</p><input
type="password" name="password" /><br>
            <input type="submit" value="Войти">
        </form>
    </div>
    <div class="form_div">
        <h4>Форма регистрации нового пользователя</h4>
        <p>Логин: <input type="text" name="login" /></p>
        <p>Пароль: <input type="password" name="password" /></p>
        <p>Повтор пароля: <input type="password" name="password2" /></p>
        <p>E-mail: <input type="text" name="email" /></p>
        <p><input type="submit" value="Зарегистрироваться" /></p>
    </div>
</body>
</html>
```

```

        </form>
    </div>
    <!-- Форма регистрации нового пользователя -->
    <div class="form_div" id="offset_top">
        <form action="register.php" method="post">
            <p id="style2">Форма регистрации нового
пользователя</p>
            <p class="pinput">Логин:</p><input
type="text" name="login" /><br />
            <p class="pinput">Пароль:</p><input
type="password" name="password" /><br />
            <p class="pinput">Повтор пароля:</
p><input type="password" name="dpassword" /><br />
            <p class="pinput">E-mail:</p><input
type="text" name="email"><br />
            <input type="submit" value=
"Зарегистрироваться" />
        </form>
    </div>
</body>
</html>

```

style44.css

```

body {
    font-family: Arial, Verdana, sans-serif;
    font-size: 12pt;
    background-color: #f0f0f0;
    color: #000000;
}
p,a {
    margin: 0px;
    text-align: center;
}
/* расположение кнопок */
input {
    text-align: left;
    margin-left: auto;
    margin-left: auto;
    margin-top: 10px;
}
/* поля ввода на главной странице */

```

```

.pinput {
    display: inline;
    text-align: justify;
    margin-right: 10px;
    margin-bottom: 0px;
}
/* блоки */
div {
    position: absolute;
    left: 60px;
    top: 75px;
    width: 350px;
    height: 200 px;
    padding: 5px;
    background-color: #00f0f0;
}
/* блоки на главной странице */
div.form_div {
    text-align: left;
}
/* блоки на страницах, сформированных с помощью php*/
div.php_div {
    text-align: center;
}
/* блок регистрации ниже блока авторизации */
#offset_top {
    margin-top: 200px;
}
#user {
    color: red;
}
#style2 {
    font-weight: bold;
    text-align: center;
    margin-bottom: 0px;
}
}

```

Создание БД для работы web-страницы из примера

При запущенном локальном сервере WAMPserver, переходим по адресу «localhost» и оказываемся на стартовой странице WAMPserver (рис. 4.2).

В разделе «Tools» (рис. 4.2) переходим по ссылке phpMyAdmin – администрирование СУБД MySQLi. По умолчанию, используется «пользователь» – «root», «пароль» – «» и «выбор сервера» – «MySQL» (рис. 4.3). Во вкладке «Базы данных» (рис. 4.4) задаем имя (в латинской транскрипции) новой БД. В примере БД имеет имя «lab».

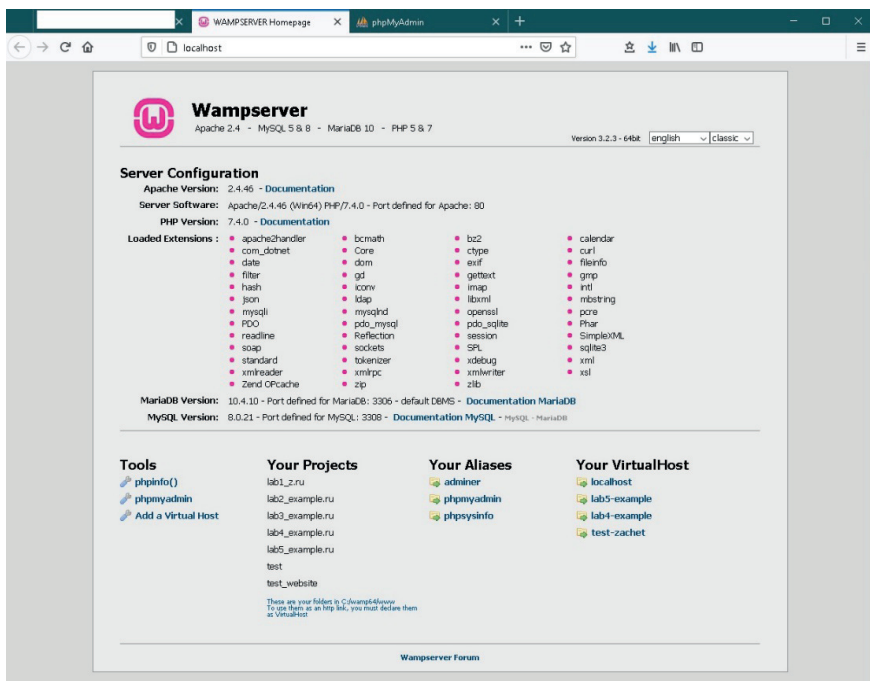


Рис. 4.2. Стартовая страница WAMPserver

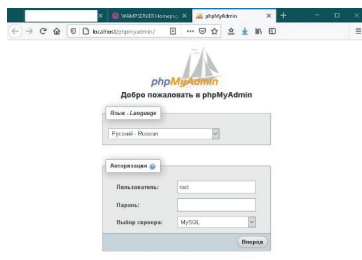


Рис. 4.3. Авторизация в phpMyAdmin

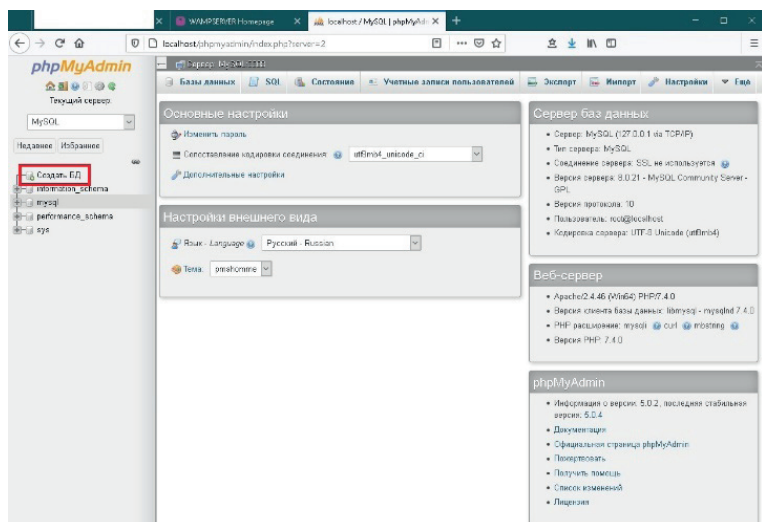


Рис. 4.4. Авторизация в phpMyAdmin

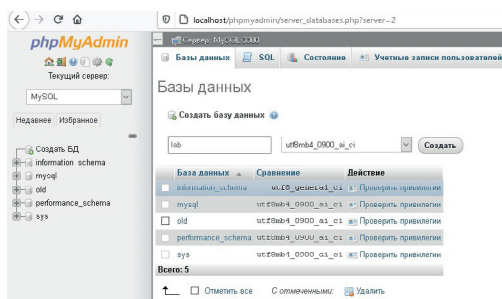


Рис. 4.5. Создание БД

Заходим в созданную БД (рис. 4.5) и задаем имя новой таблицы для хранения данных о пользователях – «users» (рис. 4.6), в которой создаем 5 столбцов.

Нажав кнопку «вперед», задаем названия полей таблицы «users» (рис. 4.7), их тип, длину. Т.к. поле «id» является первичным ключом, то для него также указываем индекс «primary» с атрибутом autoincrement. После чего нажимаем кнопку «сохранить». Результат показан на рис. 4.8.

Добавляем пару записей новых пользователей в таблицу БД. Для этого переходим во вкладку «Вставить» (рис. 4.9). Поле "id" задано

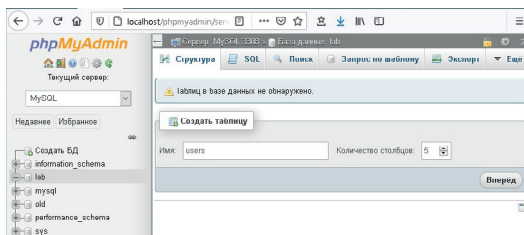


Рис. 4.6. Создание таблицы БД

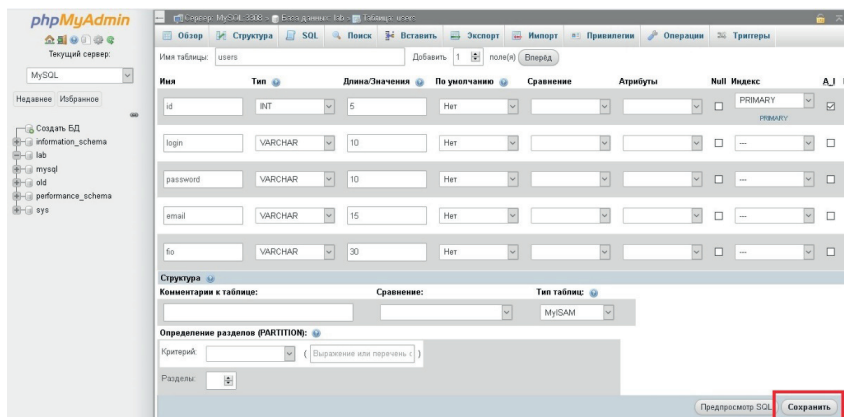


Рис. 4.7. Заполнение полей таблицы БД

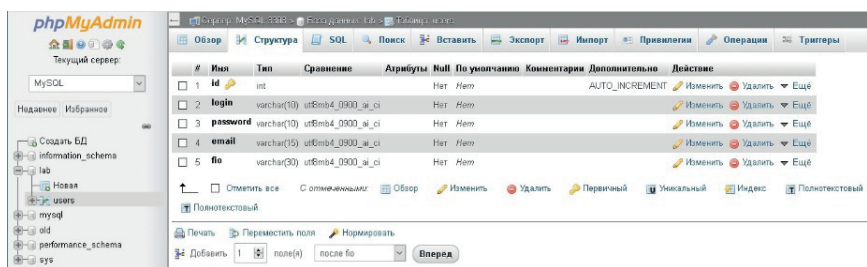


Рис. 4.8. Заполненная структура таблицы БД

с атрибутом autoincrement, поэтому заполнения не требует. Остальные заполняем, например, как показано на рис. 4.9.

В случае удачного выполнения запроса на добавление записей выдается сообщение о времени выполнения запроса и текст выполненного SQL-запроса.

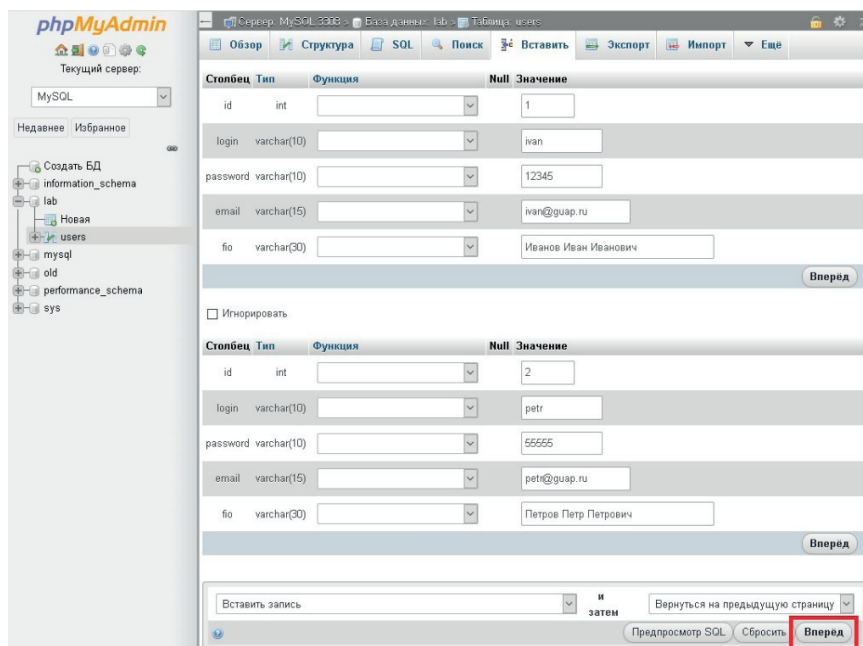


Рис. 4.9. Вкладка «Вставить»

После чего переходим на вкладку «Обзор» (рис. 4.10), где появились 2 записи пользователей.

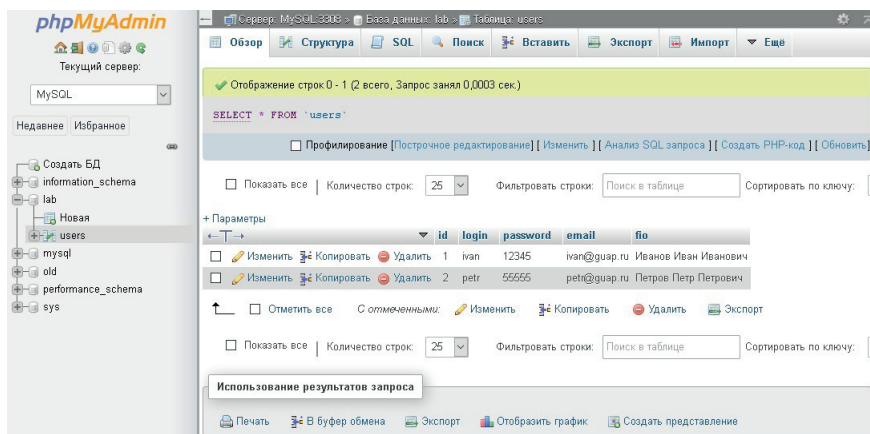


Рис. 4.10. Вкладка «Обзор»

Пример php-скриптов авторизации и регистрации пользователя

В рамках примера лабораторной работы приводятся 2 PHP-скрипта: авторизации (листинг 11) и регистрации нового пользователя (листинг 10). При подключении к MySQLi *обязательно необходимо указывать номер порта.*

Листинг 10

enter.php

```
<?php
    $login = trim($_POST["login"]);
    $password = trim($_POST["password"]);
    /* Проверка заполнения всех полей формы авторизации */
    if(empty($login) or empty($password)) {
        ?>
        <html>
        <head>
            <meta charset="utf8">
            <title>Ошибка</title>
            <link rel='stylesheet' href='style44.css' />
        </head>
        <body>
            <div class="php_div">
                <p>Одно из полей не заполнено</p>
                <a href="/">Назад к заполнению</a>
            </div>
        </body>
        </html>
    <?php
        exit;
    }
    /* Подключение к базе в случае успеха предыдущих
проверок */
    $mysql_login = 'root';
    $mysql_host = 'localhost';
    $mysql_pass = '';
    $mysql_db = 'lab';
    $connect = mysqli_connect($mysql_host, $mysql_login,
    $mysql_pass, $mysql_db, "3308");
```

```

        if (mysqli_connect_errno()) {
            echo "Не могу подключиться к БД MySQL: ".mysqli_
connect_error();
            exit();
        }
        /* Проверка наличия пользователя с таким логином в БД */
        $query_login = mysqli_query($connect, "SELECT * FROM
`users` WHERE `login` = '$login' LIMIT 0, 30") or die(mysqli_
error($connect));
        $answer = mysqli_num_rows($query_login);
        /* Пользователь с таким логином в БД не найден */
        if ($answer == 0) {
            ?>
            <html>
            <head>
                <meta charset="utf8">
                <title>Ошибка</title>
                <link rel='stylesheet' href='style44.css'
/>
                </head>
                <body>
                    <div class="php_div">
                        <p>Пользователь с таким логином
в БД не найден.</p><br />
                        <a href="/register.php">Перейти
к регистрации</a>
                    </div>
                </body>
            </html>
            <?php
                mysqli_close($connect);
                exit;
            }
            /* Проверка корректности пары логин-пароль */
            $answer_pass = mysqli_fetch_array($query_login);
            if($answer_pass['password'] != $password) {
                ?>
                <html>
                <head>
                    <meta charset="utf8">
                    <title>Ошибка</title>

```

```

        <link rel='stylesheet' href='style44.css' />
    </head>
    <body>
        <div class="php_div">
            <p>Пароль введен не верно.

</p><br />

            <a href="/">Вернуться
на главную страницу</a>
        </div>
    </body>
</html>
<?php
    mysqli_close($connect);
    exit;
}
?>
<!-- Вывод личного кабинета пользователя -->
<html>
<head>
    <meta charset="utf8">
    <title>Личный кабинет пользователя</title>
    <link rel='stylesheet' href='style44.css' />
</head>
<body>
    <div class="php_div">
        <p>Личный кабинет пользователя</p>
        <p id="user"><?php echo $answer_
pass['login'];?></p><br />
        <?php
            $users = mysqli_query($connect, "SELECT
* FROM `users`") or die(mysqli_error($connect));
            while($all_users = mysqli_fetch_
array($users)) {
                echo $all_users['id']. ". ".$all_
users['login']. ". ".$all_users['email']. ". ".$all_users['fio'].
"."<br />";
            }
            mysqli_close($connect);
        ?>
        <br /><a href="/">Вернуться на главную страницу
</a>

```

```

        </div>
</body>
</html>

```

Листинг 11

register.php

```

<?php
    $login = trim($_POST["login"]);
    $password = trim($_POST["password"]);
    $dpassword = trim($_POST["dpassword"]);
    $email = trim($_POST["email"]);
    /* Проверка заполнения всех полей формы регистрации */
    if(empty($login) or empty($password) or empty($dpassword)
or empty($email)) {
        ?>
        <html>
        <head>
            <meta charset="utf8">
            <title>Ошибка</title>
            <link rel='stylesheet' href='style44.css' />
        </head>
        <body>
            <div class="php_div">
                <p>Одно из полей не заполнено

                <a href="/">Назад к заполнению</a>
            </div>
        </body>
        </html>
    <?php
        exit;
    }
    /* Проверка равенства ввода паролей */
    if($password != $dpassword) {
        ?>
        <html>
        <head>
            <meta charset="utf8">
            <title>Ошибка</title>
            <link rel='stylesheet' href='style44.css' />
        </head>

```

```

        <body>
            <div class="php_div">
                <p>Введенные пароли не совпали
    </p><br />
                <a href="/">Назад к заполнению</a>
            </div>
        </body>
    </html>
    <?php
        exit;
    }
    /* Подключение к базе в случае успеха предыдущих проверок */
    $mysql_login = 'root';
    $mysql_host = 'localhost';
    $mysql_pass = '';
    $mysql_db = 'lab';
    $connect = mysqli_connect($mysql_host, $mysql_login,
    $mysql_pass, $mysql_db, "3308");
    if (mysqli_connect_errno()) {
        echo "Не могу подключиться к БД MySQL: ".mysqli_
connect_error();
        exit();
    }
    /* Проверка наличия пользователя с таким логином в БД */
    $query_login = mysqli_query($connect, "SELECT * FROM
`users` WHERE `login` = '$login' LIMIT 0, 30") or die(mysqli_
error($connect));
    $answer = mysqli_num_rows($query_login);
    /* Пользователь с таким логином в БД найден */
    if ($answer > 0) {
        ?>
        <html>
        <head>
            <meta charset="utf8">
            <title>Ошибка</title>
            <link rel='stylesheet' href='style44.css' />
        </head>
        <body>
            <div class="php_div">
                <p>Пользователь с таким логином
уже существует</p><br />

```

```

                                <a href="/">Назад к заполнению</a>
                        </div>
</body>
</html>
<?php
    mysqli_close($connect);
    exit;
}
/* Добавление пользователя в БД */
$add_user="INSERT INTO `users` (`id`, `login`, `password`,
`email`, `fio`) VALUES (NULL, '$login', '$password', '$email',
'')";
$insert = mysqli_query($connect, $add_user) or
die(mysqli_error($connect));
if (!$insert) {
    /* Пользователь не добавлен */
    ?>
    <html>
    <head>
        <meta charset="utf8">
        <title>Ошибка</title>
        <link rel='stylesheet' href='style44.css' />
    </head>
    <body>
        <div class="php_div">
            <p>Пользователя не добавить, запрос
не сработал</p><br />
            <a href="/">Назад к заполнению</a>
        </div>
    </body>
    </html>
    <?php
        mysqli_close($connect);
        exit;
    } else {
        /* Пользователь добавлен */
        ?>
        <html>
        <head>
            <meta charset="utf8">
            <title>Успешная регистрация</title>

```

```

        <link rel='stylesheet' href='style44.css' />
    </head>
    <body>
        <div class="php_div">
            <p>Поздравляем с успешной
регистрацией, теперь вы можете авторизоваться</p><br />
            <a href="/">Перейти
к авторизации</a>
        </div>
    </body>
</html>
<?php
    mysqli_close($connect);
    exit;
}
?>

```

В результате успешной авторизации существующего пользователя получаем страницу (рис. 4.11).

В результате успешной регистрации нового пользователя получаем страницу (рис. 4.12). Примеры сообщений об ошибках для php-скриптов (рис. 4.13).

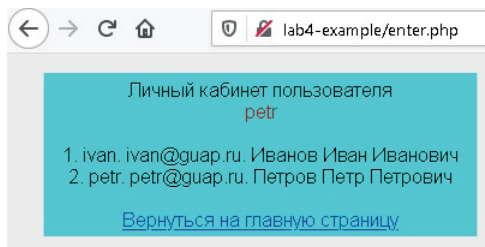


Рис. 4.11. Список учетных записей пользователей

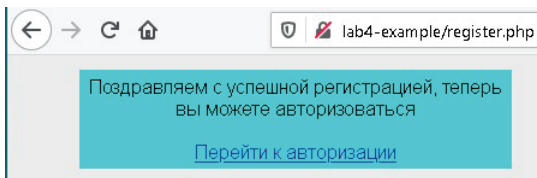


Рис. 4.12. Результат регистрации

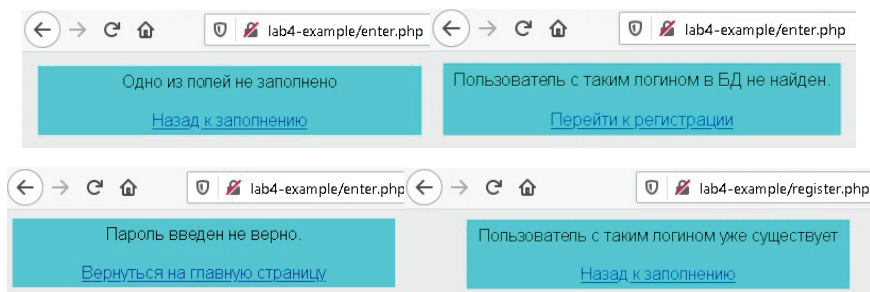


Рис. 4.13. Примеры сообщений об ошибках для php-скриптов

Для работы этой лабораторной работы **ОБЯЗАТЕЛЬНО** настроить сервер Apache. Инструкция приведена в разделе «Установка WAMPserver».

4.3. Задание

1. Создать web-страницу, содержащую HTML-форму, включающую различные элементы (текстовые поля, списки, кнопки и т.д.). В сумме не менее 10 различных элементов.

РЕКОМЕНДУЕТСЯ использовать форму из Лабораторной работы № 3.

2. Зайти в phpmyadmin (<http://localhost/Tools/phpmyadmin/index.php>) и создать базу данных (с латинским именем) с таблицей согласно полям по пункту 1.

3. Написать PHP-скрипт, принимающий данные и записывающий их в БД.

4. Написать PHP-скрипт для вывода списка всех записей из БД и выбранной записи. Разработать дизайн вывода как конкретной записи, так и всех записей.

5. Написать PHP-скрипт для удаления записи из БД. Критерий удаления по заданию преподавателя.

6. Написать PHP-скрипт для обновления записи в БД.

Тематика, для которой создается форма, определяется в соответствии с разделом «Варианты для лабораторных работ».

Примечания к выполнению лабораторной работы, содержанию и требованиям к оформлению отчета приведены в приложении Б

Лабораторная работа № 5

РАБОТА С СЕССИЯМИ И ФАЙЛАМИ COOKIE В PHP

Цель работы – Изучить функции языка PHP для работы с сессиями, файлами cookie, а также работу с несколькими PHP файлами.

5.1. Краткие методические сведения

Когда пользователи видят страницу web-сайта в браузере, PHP скрипт уже давно отработал и «забыл» о пользователе. Поэтому, если пользователь переходит с одной страницы web-сайта на другую страницу – PHP-скрипт не может запомнить данные с предыдущей страницы, например, значения переменных.

Однако, такой механизм очень нужен, хотя бы для того, чтобы запоминать выбор пользователя или то, что пользователь был авторизован.

На помощь разработчику web-приложений приходят такие механизмы как *сессии* [27] и файлы *cookie* [28].

Работа с сессиями в PHP

При работе с web-сайтом пользователь открывает его, вносит некоторые изменения и затем закрывает его. Это похоже на сеанс. ПК знает пользователя. ПК знает, когда пользователь запустит приложение и когда пользователь закончит. Но в интернете есть одна проблема: web-сервер не знает пользователя и что он делает, потому что HTTP-протокол не поддерживает состояние.

Переменные \$_SESSION решают эту проблему, сохраняя информацию о пользователе для использования на нескольких страницах (например: имя пользователя, любимый цвет и т.д.).

Сессия – это механизм PHP, который позволяет хранить данные для конкретного пользователя между запусками скрипта. Т.е. можно записывать какую-либо информацию в сессию и считывать ее оттуда при следующем запуске этого или другого PHP-скрипта web-сайта. С помощью сессии можно реализовать авторизацию пользователей, корзину интернет-магазина, форма для заполнения пользователем, разбитая на несколько страниц сайта (на первой мы спрашиваем имя, на второй e-mail и т.д.) и другое.

Чтобы записать что-то в сессию ее сначала нужно *инициализировать* с помощью функции **session_start()**:

```
<?php
    session _ start();
?>
```

В одном скрипте сессия должна инициализироваться **только один раз**, иначе скрипт выдаст ошибку. После инициализации мы можем записать что-нибудь в сессию.

```
// записываем данные в сессию
$_SESSION['test'] = 'Тест!';
```

Пример извлечения данных из сессии:

```
// выведем данные переменной test из сессии
echo $_SESSION['test'];
```

Переменную сессии можно удалить с помощью функции **unset**:

```
<?php
    //После выполнения этой команды в $_SESSION['var']
    станет null
    unset($_SESSION['var']);
?>
```

Если нужно удалить все переменные сессии для данного пользователя, то вместо **unset** следует воспользоваться функцией **session_destroy()** (ее можно вызывать только тогда, когда сессия запущена через **session_start()**):

```
<?php
    //После выполнения этой команды ВСЕ переменные сессии
    станут null
    session_destroy();
?>
```

Было рассмотрено принудительное завершение сессии, но сессия может завершиться сама в двух следующих случаях: при закрытии браузера пользователем или по истечении определенного времени, во время которого пользователь не совершает никаких действий на странице (по умолчанию, это время около 15-25 минут).

Работа с файлами cookie в PHP

Cookie – это способ долговременного хранения данных в браузере пользователя. В cookie можно сохранить только 4 килобайта информации. Кроме того, есть ограничение на количество cookie для домена.

Стоит помнить, что в cookie нужно писать до любого вывода на экран. Написать что-то в cookie можно с помощью функции **setcookie**, которая первым параметром принимает имя файла cookie, а вторым – значение:

```
<?php
    //Записываем в cookie с именем test значение 'Тест!'
    setcookie('test', 'Тест!');
?>
```

Однако файлы cookie из примера выше долго не живут – только до закрытия браузера. Продлить время жизни cookie можно с помощью третьего параметра, который принимает время (конкретную дату) окончания жизни cookie в формате timestamp. Однако устанавливать конкретную дату «смерти» cookies не очень удобно, так как дата установки этой cookie всегда разная. Поэтому третий параметр принято записывать так: *настоящий момент времени + N секунд*. Настоящий момент времени в формате timestamp можно получить с помощью функции `time()`.

```
<?php
    // Запишем cookie на час
    setcookie("test","Тест!", time() + 3600);

    // Запишем cookie на сутки
    setcookie("test","Тест!", time() + 3600*24);
?>
```

Cookie можно прочитать с помощью глобального массива `$_COOKIE`. Пример чтения ранее установленного файла cookie test:

```
<?php
    echo $_COOKIE['test'];
?>
```

Для удаления cookie необходимо установить дату «смерти» cookie на текущий момент времени или момент в прошлом:

```
<?php
    //Удалим cookie, установив третий параметр в текущий
    момент времени
    setcookie('test', '', time());
?>
```

5.2. Пример

В качестве примера создадим страницу, содержащую форму для ввода данных о пользователе (рис. 5.1) Пример заполнения формы показан на рис 5.3. Код приведен в листинге 12. При нажатии на кнопку «Старт сессии» данные из полей формы сохраняются как элемен-

A web browser window with the address bar showing 'lab5-example'. The page title is 'Функции для работы с сессиями в PHP'. The main content is a light blue box with the heading 'Введите Ваши данные:'. It contains four input fields: 'Фамилия:', 'Имя:', 'Отчество:', and 'Любимое число (от 1 до 5):'. The 'Любимое число' field is a dropdown menu currently showing '1'. At the bottom of the box is a button labeled 'Старт сессии'.

Рис. 5.1. HTML-форма

A light blue message box with the text 'Одно из полей не заполнено' and a blue link 'Назад к заполнению'.

Рис. 5.2. Сообщение о некорректном заполнении HTML-формы

The same web browser window as in Figure 5.1, but the form fields are now filled with data: 'Фамилия:' is 'Иванов', 'Имя:' is 'Иван', 'Отчество:' is 'Иванович', and 'Любимое число (от 1 до 5):' is '4'. The 'Старт сессии' button remains at the bottom.

Рис. 5.3. Заполненная HTML-форма

ты массива сессии, а при некорректном заполнении появляется сообщение об ошибке (рис. 5.2). При успешном заполнении пользователю доступна ссылка перехода на следующую страницу (рис. 5.4). При переходе по ссылке открывается страница приветствия (рис. 5.5), в тексте которой использованы данные, введенные пользователем в первой web-форме и кнопка «Удалить данные сессии». При нажатии на нее сессионные переменные уничтожаются, результат будет продемонстрирован на следующей странице (рис. 5.6).

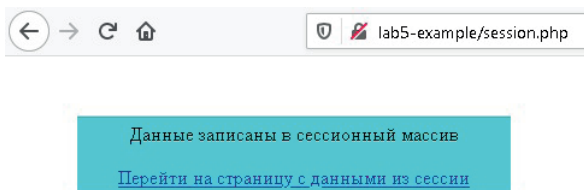


Рис. 5.4. Сообщение о том, что данные записаны в сессионный массив

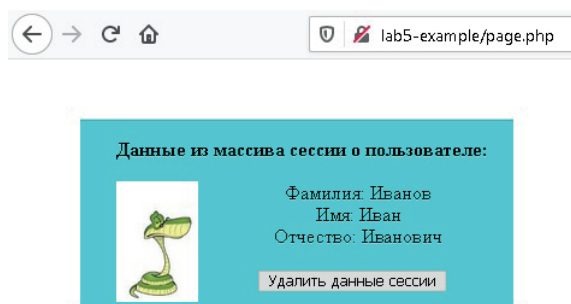


Рис. 5.5. Данные из массива сессии о пользователе

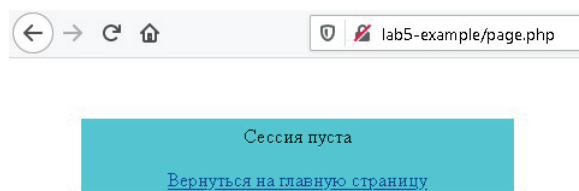


Рис. 5.6. Сообщение о том, что сессия пуста

Листинг 12

index.html

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style5.css">
        <title>Функции для работы с сессиями в PHP
    </title>
    </head>
    <body>
        <h1>Функции для работы с сессиями в PHP</h1>
        <div id="style1">
            <form action="session.php" method="post">
                <p id="style2">Введите Ваши
даные:</p><br>
                    <p class="pinput">Фамилия:
</p><input type="text" name="surname" /><br>
                    <p class="pinput">Имя:</p><input
type="text" name="name" /><br>
                    <p class="pinput">Отчество:
</p><input type="text" name="middlename" /><br>
                    <p class="pinput">Любимое число
(от 1 до 5):</p>
                        <select id="s1" name="num">
                            <option value="1">1</option>
                            <option value="2">2</option>
                            <option value="3">3</option>
                            <option value="4">4</option>
                            <option value="5">5</option>
                        </select><br>
                        <input type="submit" value="Старт
сессии" />
                    </form>
                </div>
            </body>
        </html>
```

style5.css

```
body {
    font-family: Arial, Verdana, sans-serif;
```

```

        font-size: 12pt;
        background-color: #f0f0f0;
        color: #000000;
    }
    h1 {
        text-align: justify;
        margin-left: 60px;
        margin-bottom: 0px;
        color: #333333;
    }
    input, select {
        text-align: left;
        margin-left: 25px;
        margin-top: 10px;
    }
    .pinput {
        display: inline;
        text-align: justify;
        margin-left: 25px;
        margin-right: 10px;
        margin-bottom: 0px;
    }
    #style1 {
        position: absolute;
        left: 60px;
        top: 75px;
        width: 350px;
        height: 150 px;
        padding: 5px;
        background-color: #00f0f0;
        text-align: left;
    }
    #style2 {
        font-weight: bold;
        text-align: left;
        margin-left: 25px;
        margin-top: 10px;
        margin-bottom: 0px;
    }
    #style3 {
        position: absolute;

```



```

        left: 60px;
        top: 50px;
        width: 350px;
        height: 50 px;
        padding: 5px;
        background-color: #00f0f0;
        text-align: center;
    }
    p,a {
margin: 0px;
    }
    .pinput2 {
        padding: 5px;
        display: inline;
        text-align: left;
        margin-left: 5px;
        margin-bottom: 0px;
    }
    img {
        height: 100px;
        float: left;
        margin-left: 25px;
        margin-right: 0px;
    }
    #style4 {
        font-weight: bold;
        text-align: left;
        margin-left: 25px;
        margin-top: 10px;
        margin-bottom: 0px;
    }
}

```

В атрибуте action тега form прописываем имя php-скрипта, который будет сохранять данные из html-формы в массив – session.php (листинг 13).

Листинг 13

session.php

```

<?php
    session_start();    //запуск новой сессии, либо
возобновление существующей
    // проверка входных данных, полученных из web-формы

```

```

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
// получение данных из web-формы
$surname=test_input($_POST["surname"]);
$name=test_input($_POST["name"]);
$middlename=test_input($_POST["middlename"]);
$num=test_input($_POST["num"]);
// проверка заполнения всех полей web-формы
if (empty($name) || empty($surname) || empty($middlename)
|| empty($num)) {
    ?>
    <!-- вывод сообщения об ошибке, если есть
незаполненные поля -->
    <!DOCTYPE html>
    <html>
        <head>
            <title>Error</title>
            <link rel="stylesheet" href="style5.
css">
            </head>
            <body>
                <div id="style3">
                    <p>Одно из полей
не заполнено</p><br>
                    <a href="/">Назад
к заполнению</a>
                </div>
            </body>
        </html>
        <?php
            exit; // прекращение выполнения скрипта
    } else {
        // сохранение данных пользователя в массив сессии
        $_SESSION['surname']=$surname;
        $_SESSION['name']=$name;
        $_SESSION['middlename']=$middlename;
        $_SESSION['num']=$num;
    }
}

```

```

?>
<!-- вывод ссылки на следующую страницу -->
<!DOCTYPE html>
<html>
    <head>
        <title>Успешная запись в сессию</title>
        <link rel="stylesheet" href="style5.css">
    </head>
    <body>
        <div id="style3">
            <p>Данные записаны в сессионный массив</p><br>
            <a href="/page.php">Перейти на страницу
с данными из сессии</a>
        </div>
    </body>
</html>

```

PHP-скрипт по окончании работы, если все поля заполнены, то формирует страницу, показанную на рис. 5.4.

При нажатии на ссылку вызывается скрипт `page.php` (листинг 14), работающий только с данными из массива `$_SESSION`.

Листинг 14

page.php

```

<?php
    session_start();    // запуск новой сессии, либо
возобновление существующей
    // нажатие кнопки удаления сессии
    if (isset($_POST["submit_exit"])) {
        // очистка переменной $_SESSION
        session_unset();
        // уничтожение всех данных, связанных с текущей
сессией
        session_destroy();
    }
    // проверка существования сессионного массива
    if (!isset($_SESSION["name"]) && !isset($_SESSION
["surname"]) && !isset($_SESSION['middlename']) && !isset
($_SESSION["num"])) {
        ?>
        <!-- вывод сообщения о том, что сессия пуста -->
        <!DOCTYPE html>

```

```

<html>
<head>
    <title>Session error</title>
    <link rel="stylesheet" href="style5.css">
</head>
<body>
    <div id="style3">
        <p>Сессия пуста</p><br>
        <a href="/">Вернуться на главную
страницу</a>
    </div>
</body>
</html>
<?php
    session_start();    // возобновление
существующей сессии для удаления сессии
    session_unset();    // очистка переменной
$_SESSION
    session_destroy(); // уничтожение всех
данных, связанных с текущей сессией
    exit; // прекращение выполнения скрипта
    }
?>
<!-- вывод данных пользователя и кнопки для очистки сессии -->
<!DOCTYPE html>
<html>
<head>
    <title>Страница с данными пользователя</title>
    <link rel="stylesheet" href="style5.css">
</head>
<body>
    <div id="style3">
        <p id="style4">Данные из массива сессии
о пользователе:</p><br>
        ">
        <p class="pinput2">Фамилия:</p><?php echo
$_SESSION['surname'];?><br>
        <p class="pinput2">Имя:</p><?php echo
$_SESSION['name'];?><br>
        <p class="pinput2">Отчество:</p><?php echo
$_SESSION['middlename'];?><br><br>

```

```

        <form action="page.php" method="POST">
            <input type="submit" name="submit_exit"
value="Удалить данные сессии">
        </form>
    </div>
</body>
</html>

```

В результате работы PHP-скрипта получаем страницу, показанную на рис. 5.5.

При нажатии на кнопку «Удалить данные сессии» выполняется часть скрипта, отвечающая за обработку нажатия кнопки с именем «submit_exit»:

```

if (isset($_POST["submit_exit"])) {
    session_unset();
    session_destroy();
}

```

В результате нажатия на кнопку получаем сообщение (рис. 5.6) о невозможности получения данных из массива \$_SESSION.

5.3. Задание

1. Реализовать ввод данных для HTML-формы из лабораторной работы 3 в 3 шага (этапа), используя технологию сессий. **Каждый шаг (этап) должен находиться на отдельной странице.** Переключение страниц осуществить с помощью метода GET.

2. Написать скрипт, который применяет функции управления сессией для запоминания того, какие страницы уже посещались пользователем. Вывести список ссылок на все посещенные страницы с указанием количества посещений.

3. Реализовать возможность сохранения данных, введенных пользователем в прошлый раз в HTML-форме с помощью сессий, чтобы их повторно не вводить.

4. Модернизировать php-скрипт путем добавления нулевого (стартового) шага (этапа). На нулевом (стартовом) шаге (этапе) предложить пользователю указать свое имя и выбрать цвет фона страниц сайта. Сохранить эту информацию, используя файлы cookie. Далее, используя файлы cookie, на всех шагах (этапах), кроме нулевого, приветствовать пользователя по имени и использовать выбранный пользователем цвет фона. При приветствии запрещается использовать всплывающие окна.

Примечания к выполнению лабораторной работы, содержанию и требованиям к оформлению отчета приведены в приложении Б.

Варианты для лабораторных работ

Таблица 1

Тематика HTML-формы

№ варианта	Тематика HTML-формы и обязательные поля*
1	Дистанционное обучение <ul style="list-style-type: none"> • логин (номер студенческого билета) • номер паспорта
2	Электронный магазин бытовой техники <ul style="list-style-type: none"> • адрес доставки • номер банковской карты
3	Электронный магазин компьютерной техники <ul style="list-style-type: none"> • адрес доставки • имя
4	Фитнес-клуб <ul style="list-style-type: none"> • логин (номер карты клуба) • имя
5	Бронирование мест в гостинице <ul style="list-style-type: none"> • e-mail • номер банковской карты
6	Туристическое агентство <ul style="list-style-type: none"> • промокод • номер бонусной карты
7	Агентство недвижимости (продажа объектов) <ul style="list-style-type: none"> • мобильный телефон • фамилия
8	Агентство недвижимости (аренда объектов) <ul style="list-style-type: none"> • мобильный телефон • номер паспорта
9	Заказ билетов в театр <ul style="list-style-type: none"> • промокод • фамилия
10	Заказ билетов на самолет (поезд) <ul style="list-style-type: none"> • e-mail • номер бонусной карты

* обязательные поля используются в лабораторной работе «Проверка данных HTML-форм в PHP»

Пояснения к выполнению лабораторных работ и оформлению отчетов

Примечания к выполнению лабораторных работ:

В ходе выполнения ЛР следует использовать HTML5, CSS3 (в частности, блочную верстку или технологию FlexBox) и PHP7 или новее.

В ходе выполнения ЛР следует выносить код каскадной таблицы стилей и сценариев в файлы *.css и *.js соответственно. Код скриптов должен располагаться в файлах *.php. *Не следует использовать Javascript, JQuery и другие технологии для тех задач, которые можно решить с помощью PHP.*

Рекомендуется использовать относительные пути при создании *.html-, *.css-, *.php- файлов для обращения к рисункам, файлам css и т.д.

Дополнительные баллы (для лабораторной работы «Обработка данных HTML-форм в PHP» и «Работа с базой данных MySQL в PHP») за демонстрацию хороших остаточных знаний по дисциплине «Web-технологии» (HTML, CSS, JavaScript) [1-2] и разработку хорошего графического дизайна HTML-формы и итоговой web-страницы.

Содержание отчета

1. Титульный лист
 2. Цель работы
 3. Задание к лабораторной работе (согласно варианту ЛР)
 4. Код web-страниц (*.html и *.css) и/или php-скриптов
- Листинг кода должен содержать комментарии
5. Примеры web-страниц (демонстрация работы web-страниц и php-скрипта)
 6. Выводы по лабораторной работе

Требования к оформлению отчета о лабораторной работе

При оформлении отчета о лабораторной работе следует пользоваться ГОСТ 7.32-2017 издания 2017 года. Правила оформления текстовых документов по ГОСТ 7.32-2017, а также титульные листы лабораторных работ представлены на сайте ГУАП (<https://guap.ru/standart/doc>).

Дополнительные интернет ресурсы

1. <https://schoolsw3.com/php/index.php>
2. <http://www.php-s.ru/self-teacher/>
3. <http://www.php.su/lessons/>

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Красильникова О. И.* Web-технологии для разработки клиентской части web-страниц: учеб. пособие в 2 ч. Ч. 1 / О. И. Красильникова, Н. Н. Красильников. – СПб.: ГУАП, 2017. – 59 с.
2. *Красильникова О. И.* Web-технологии для разработки клиентской части web-страниц: в 2 ч.: учеб. пособие. Ч. 2 / О. И. Красильникова, Н. Н. Красильников. – СПб.: ГУАП, 2018. – 43 с.
3. Программирование интерактивных веб-приложений: учеб. пособие / А. В. Аграновский и др. – СПб.: ГУАП, 2019. – 92 с.
4. *Аграновский, А. В.* Основы интернет-программирования: учеб. пособие / А. В. Аграновский, В. С. Павлов, Е. Л. Турнецкая. – СПб.: ГУАП, 2018. – 135 с.
5. *Аграновский, А. В.* Разработка веб-приложений средствами языка PHP: учеб. пособие / А. В. Аграновский, В. А. Ненашев, В. С. Павлов, Е. Л. Турнецкая. – СПб.: ГУАП, 2018. – 121 с.
6. *Котеров Д. В., Симдянов И. В.* PHP 7. – СПб.: БХВ-Петербург, 2016. – 1071 с.
7. *Дронов В., Прохоренко Н. А.* HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. – СПб.: БХВ-Петербург, 2019. – 912 с.
8. *Никсон Р.* Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. – М.: Питер, 2017. – 768 с.
9. *Скляр Д.* Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов. – М.: Вильямс, 2017. – 464 с.
10. Doug Bierer PHP 7 Programming Cookbook. UK.: Packt Publishing Ltd., 2016. – 610 с.
11. PHP и MySQL. Разработка веб-приложений / Д. Н. Колисниченко. – 5-е изд. – СПб.: БХВ-Петербург, 2015. – 592 с.
12. *Прохоренко, Н. А.* HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера: пособие / Прохоренко Н. А. – 4-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2015. – 768 с.
13. <https://www.php.net/manual/ru/>
14. <https://www.php.net/manual/ru/book.mysql.php#book.mysql>
15. *Богословская Н. В.* Электронный бизнес (бизнес-портал): учеб.-метод. пособие / Н. В. Богословская, А. В. Бржезовский. – СПб.: ГУАП, 2020. – 91 с.
16. <https://www.wampserver.com/>
17. <https://serverspace.by/support/help/ustanovka-i-nastroika-wamp-servera/>
18. http://coderhs.com/archive/php_send_form

19. https://schoolsw3.com/php/php_forms.php
20. <https://www.php.net/manual/ru/function.preg-match.php>
21. https://schoolsw3.com/php/php_form_validation.php
22. <https://snipp.ru/php/uploads-files>
23. <https://prowebmastering.ru/php-upload-file.html>
24. https://schoolsw3.com/php/php_file_upload.php
25. Семененко Т. В. Создание баз данных в среде MS Access: метод. указ. к вып. лаб. работ. СПб.: ГУАП, 2014. 94 с.
26. Ушаков В. А. Мультимедиа в мобильных системах: практикум / В. А. Ушаков. – СПб.: ГУАП, 2020 – 67 с.
27. <http://old.code.mu/books/php/auth/rabota-s-sessiyami-php.html>
28. <http://old.code.mu/books/php/auth/rabota-s-cookie-na-php.html>

СОДЕРЖАНИЕ

Предисловие.....	3
Установка WAMPserver	4
1. Установка WAMPserver	4
2. Запуск WAMPserver и обзор основных компонентов	7
3. Настройка сервера Apache.....	9
4. Тестовый запуск первого сайта на виртуальном (локальном) хостинге	9
Лабораторная работа № 1. Обработка данных HTML-форм в PHP	14
1.1. Краткие методические сведения	14
1.2. Пример	16
1.3. Задание.....	24
Лабораторная работа № 2. Проверка данных HTML-форм в PHP	25
2.1. Краткие методические сведения.....	25
2.2. Пример	28
2.3. Задание.....	31
Лабораторная работа № 3. Работа с файлами в PHP	34
3.1. Краткие методические сведения.....	34
3.2. Пример	34
3.3. Задание.....	39
Лабораторная работа № 4. Работа с базой данных MySQL в PHP	41
4.1. Краткие методические сведения.....	41
4.2. Пример	41
Пример web-страницы с формой авторизации и регистрации пользователя	41
Создание БД для работы web-страницы из примера	44
Пример php-скриптов авторизации и регистрации пользователя	49
4.3. Задание.....	56
Лабораторная работа № 5. Работа с сессиями и файлами cookie в PHP...	57
5.1. Краткие методические сведения.....	57
Работа с сессиями в PHP	57
Работа с файлами cookie в PHP.....	58
5.2. Пример	59
5.3. Задание.....	68
Приложение А. Варианты для лабораторных работ	69
Приложение Б. Пояснения к выполнению лабораторных работ и оформлению отчетов	70
Библиографический список	71

Учебное издание

Ушаков Виталий Анатольевич

**РАЗРАБОТКА СОВРЕМЕННЫХ
ДИНАМИЧЕСКИХ WEB-САЙТОВ
СРЕДСТВАМИ ЯЗЫКА PHP**

Лабораторный практикум

Публикуется в авторской редакции
Компьютерная верстка *С. Б. Мацапуры*

Подписано к печати 10.06.21.
Формат 60×84 1/16. Усл. печ. л. 4,3. Уч.-изд. л. 4,6.
Тираж 50 экз. Заказ № 180.

Редакционно-издательский центр ГУАП
190000, Санкт-Петербург, Б. Морская ул., 67