

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Кафедра информационных технологий  
 и вычислительных систем

«УТВЕРЖДАЮ»

**Проректор по образовательной**

**деятельности**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Методические указания к выполнению лабораторных работ

по дисциплине

**Основы Web-технологий**

|  |  |
| --- | --- |
| Уровень высшего образования | бакалавриат |
| Направление подготовки | 09.03.01 «Информатика и вычислительная техника» |
| Направленность (профиль) | Разработка программных комплексов в рамках цифровой трансформации деятельности предприятий |
| Цикл дисциплины и его часть | Блок 1 «Дисциплины (модули)», обязательная часть |
| Форма обучения | очная |

г. Москва 2022 г.

**Раздел 4. Система управления базами данных MySQL.**

## Лабораторная работа №4

## Тема: База данных для сайта. Связь с формой и сервером.

## Цель: Закрепление теоретических знаний по созданию базы данных в СУБД MySQL с помощью программы phpMyAdmin. Приобретение практических навыков работы с данными средствами языка PHP.

## Задание: Создать базу данных товаров или услуг выбранной компании в СУБД MySQL, состоящую из трех таблиц, средствами инструмента phpMyAdmin. Реализовать подключение к базе данных средствами языка PHP. Для этого необходимо создать Web-интерфейс, с помощью которого организовать добавление, сортировку данных по различным критериям, изменение данных в базе данных, считывание данных из таблицы, вывод данных на экран средствами языка PHP.

## Первой таблицей в базе данных будет таблица товаров или услуг, поскольку это – основная единица компании. Для начала создайте в phpMyAdmin новую базу данных, с любым названием и выполните следующий sql-запрос:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | CREATE  TABLE IF NOT EXISTS product (    id int(11) NOT NULL AUTO\_INCREMENT,    manufacturer\_id smallint(6) NOT NULL,    name varchar(255) NOT NULL,    alias varchar(255) NOT NULL,    short\_description text NOT NULL,    description text NOT NULL,    price decimal(20,2) NOT NULL,    image varchar(255) NOT NULL,    available smallint(1) NOT NULL DEFAULT '1',    meta\_keywords varchar(255) NOT NULL,    meta\_description varchar(255) NOT NULL,    meta\_title varchar(255) NOT NULL,    PRIMARY KEY (id),    UNIQUE KEY id (id)    ) ENGINE=MyISAM  DEFAULT CHARSET=cp1251 AUTO\_INCREMENT=1 ; |

## Итак, у нас получилась таблица товаров, содержащая 12 ячеек:

## id –уникальный идентификатор товара в системе

## manufacturer\_id – идентификатор производителя товара

## name – название товара, ограниченное 255 символами

## alias – алиас для системы ЧПУ (если она есть на сайте)

## short\_description – короткое описание товара, пригодиться для списка товаров магазина

## description – полное описание товара, будет выводиться на странице деталей товара

## price – цена товара, может состоять из чисел, с двумя знаками после запятой

## image – основное изображение товара. Это не обязательное поле, но по-началу так будет проще

## available – не обязательное поле, обозначающее доступность товара на складе, по умолчанию принимает значение 1 (товар доступен на складе)

## meta\_keywords, meta\_description  – поля для SEO продвижения сайта, по желанию будут выводится в мета тегах keywords и description

## meta\_title – заголовок (title) страницы товара

## В принципе, такой таблицы хватит для описания одного товара. Обратите внимание, что в таблице нет поля привязки товара к категориям магазина. Это сделано специально, чтобы можно было добавить один товар в несколько категорий. Но, конечно, приведенной таблицы не хватит для более или менее полноценной компании, поскольку у товаров и услуг могут быть дополнительные свойства, такие как размеры, цвета, которые должен выбирать пользователь. Специально для этого мы создадим еще одну таблицу product\_properties:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE  TABLE IF NOT EXISTS product\_properties (    id int(11) NOT NULL AUTO\_INCREMENT,    product\_id int(11) NOT NULL,    property\_name varchar(255) NOT NULL,    property\_value varchar(255) NOT NULL,    property\_price decimal(20,2) NOT NULL,    PRIMARY KEY (id),    UNIQUE KEY id (id)    )  ENGINE=MyISAM DEFAULT CHARSET=cp1251 AUTO\_INCREMENT=1 ; |

## Назначение полей:

## id – уникальный идентификатор свойства (нужен для редактирования или удаления записи)

## product\_id – идентификатор продукта, которому соответствует свойство

## property\_name – название свойства (ширина, высота, цвет и еще что угодно)

## property\_value – значение свойства, принимает строковое значение (для размеров можно записать: 200мм, для цвета: красный)

## property\_price – цена товара или услуги с данным свойством, на тот случай если свойство влияет на цену

## Теперь у нас уже есть таблица для добавления товара или услуги компании, а также таблица для задания свойств товара. Следующей таблицей будет таблица для хранения изображений товара, ведь в большинстве случаев одного изображения не хватит для того, чтобы дать пользователю полное представление о товаре. И так, таблица изображений товара или услуги product\_images:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE  TABLE IF NOT EXISTS product\_images (  id int(11) NOT NULL AUTO\_INCREMENT,  product\_id int(11) NOT NULL,  image varchar(255) NOT NULL,  title varchar(255) NOT NULL,  PRIMARY KEY (id),  UNIQUE KEY id (id)  ) |

Здесь все просто:

**id** – идентификатор товара, необходим для удаления или замены изображения

**product\_id** – идентификатор товара, к которому относится данное изображение

**image** – изображение

**title** – название товара или услуги.

## Есть два метода подключения к базе данных MySQL с помощью PHP: MySQLi и PDO.

## MySQLi расшифровывается как MySQL Improved. Это эксклюзивное расширение MySQL, которое добавляет новые функции в интерфейс базы данных. Функции MySQLi являются как процедурными, так и объектно-ориентированными, причём первую парадигму расширение унаследовало от более ранней версии MySQL.

## Сама MySQL разбивает задачу на линейные, пошаговые процедуры, что затрудняет внесение изменений, поскольку вам приходится редактировать код сверху. Между тем MySQLi рассматривает данные как набор взаимозаменяемых объектов с функциями, позволяя пользователям легко добавлять или удалять данные.

## PDO расшифровывается как PHP Data Object, или объект данных PHP. В отличие от MySQLi, PDO является только объектно-ориентированным методом. Он поддерживает ряд различных типов баз данных, использующих PHP, таких как MySQL, MSSQL, Informix и PostgreSQL.

## Исходные функции mysql\_ устарели. Их лучше не использовать, поскольку они небезопасны и больше не поддерживаются.

## Одна из наиболее важных функций, которую поддерживают оба метода — это подготовленные выражения (prepared statements). Она сокращает время, необходимое MySQL для выполнения повторяемого запроса. Эта функция также используется для предотвращения SQL-инъекций при внесении изменений в базу данных.

## Вам потребуется правильное имя сервера, или имя хоста для конфигурации. Это имя, которое вы будете использовать, если загрузите свой PHP-скрипт на тот же сервер, что и база данных. С другой стороны, если вы подключаетесь к базе данных из удалённого места (например, со своего компьютера), вам придётся использовать IP-адрес MySQL-сервера. Чтобы получить дополнительную информацию, обратитесь к своему хостинг-провайдеру. Он предоставит вам актуальную информацию о том, какое имя использовать в качестве имени хоста.

## PHP-подключение к базе данных MySQL с MySQLi

## Выполните следующие действия, чтобы подключить PHP-скрипт к MySQL посредством MySQLi:

## Перейдите в Файловый менеджер -> public\_html.

## Создайте новый файл, щёлкнув на соответствующую иконку в верхнем меню.

## Сохраните его как databaseconnect.php. Вы можете заменить имя на любое другое, просто убедитесь, что в качестве расширения используется php.

## Дважды щёлкните по файлу, чтобы открыть его. Скопируйте и вставьте в него следующие строки кода. Замените первые четыре значения после <?php учётными данными, которые вы указали ранее.

## <?php

## $servername = "localhost";

## $database = "databasename";

## $username = "username";

## $password = "password";

## // Создаем соединение

## $conn = mysqli\_connect($servername, $username, $password, $database);

## // Проверяем соединение

## if (!$conn) {

## die("Connection failed: " . mysqli\_connect\_error());

## }

## echo "Connected successfully";

## mysqli\_close($conn);

## ?>

## Объяснение Кода MySQLi

## Основным методом, используемым в этом скрипте, является mysqli\_connect (). Это внутренняя функция PHP для установления нового соединения с сервером MySQL.

## В начале нашего кода мы видим несколько объявлений переменных и значений, присвоенных этим переменным. Обычно нам нужно четыре из них, чтобы установить правильное соединение с базой данных: $servername, $database, $username и $password. В коде мы указываем наши данные для доступа к базе данных как значения для этих переменных, чтобы их можно было передать в функцию.

## Если попытка соединения была неудачной, выполняется функция die(). Она убивает наш скрипт и выдаёт сообщение об ошибке подключения, которое мы прописали. По умолчанию в сообщении об ошибке подключения MySQL будет указано «Connection failed», за которым следует точное сообщение об ошибке с описанием проблемы.

## С другой стороны, если MySQL-соединение установлено успешно, мы увидим сообщение «Connected successfully».

## Последняя часть кода, mysqli\_close, позволяет закрыть соединение с базой данных вручную. Если вы ничего не укажете, соединения MySQL закроются автоматически после завершения скрипта.

## PHP-подключение к БД MySQL с PDO

## Другой метод подключения к БД MySQL с использованием PHP-скрипта — через PDO. В целом он похож на предыдущий, но с некоторыми особенностями:

## В public\_html создайте файл с названием pdoconfig.php и вставьте следующий код. Как всегда, не забудьте заменить значения плейсхолдеров информацией из вашей базы данных. Сохраните и закройте его, когда закончите.

## <?php

## $host = 'localhost';

## $dbname = 'databasename';

## $username = 'username';

## $password = 'password';

## Создайте ещё один файл и назовите его databaseconnect.php в том же каталоге, но со следующим кодом. Если вы назвали предыдущий файл по-другому, не забудьте изменить значение после require\_once.

## <?php

## require\_once 'pdoconfig.php';

## try {

## $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);

## echo "Connected to $dbname at $host successfully.";

## } catch (PDOException $pe) {

## die("Could not connect to the database $dbname :" . $pe->getMessage());

## }

## Объяснение Скрипта PDO

## Для подключения к базе данных PDO необходимо создать новый объект PDO с именем источника данных (DSN), именем пользователя и паролем.

## DSN определяет тип базы данных, имя базы данных и любую другую информацию, относящуюся к базе данных, если это необходимо. Это переменные и значения, указанные нами в файле dbconfig.php, на которые один раз ссылается строка require\_once в файле databaseconnect.php.

## В последнем примере вы найдёте код try… catch... Это означает, что скрипт попытается подключиться к MySQL, используя предоставленный код, но в случае возникновения проблемы будет выполнен код в разделе catch. Вы можете использовать блок catch для отображения сообщений об ошибках подключения или запустить альтернативный код в случае сбоя блока try.

## Если соединение установлено успешно, вы увидите сообщение «Connected to $dbname at $host successfully». Однако, если попытка не удалась, код в блоке catch покажет простое сообщение об ошибке и завершит скрипт.

## Проверка Подключения и Устранение Распространённых Ошибок

## Чтобы проверить, успешно ли установлено соединение, войдите в свой домен так: vashdomen/databaseconnect.php. Если вы назвали PHP-файл другим именем, обязательно укажите правильное название.

## Если всё работает хорошо, вы увидите «Connected successfully» или другой вариант этого сообщения.

## В случае возникновения проблемы при попытке установить соединение, вы увидите сообщения об ошибке. Они отличаются для MySQLi и PDO.

## Если вы видите сообщение «Access denied» или «Could not connect to database», сопровождаемое “(using password: YES)”, первое, что нужно сделать, это проверить данные для доступа к базе данных. Возможно, вы сделали опечатку или пропустили какую-то часть.

## Если вы видите сообщение «Can’t connect to MySQL server on ‘server’ (110)» в MySQLi, это означает, что скрипт не получил ответа от сервера. Это происходит, когда мы устанавливаем «server» вместо «localhost» в качестве $servername, и имя не распознаётся.

## Сообщение об аналогичной ошибке в PDO будет выглядеть как: «Connection failed: SQLSTATE[Hy000] [2002]». А за ним следует уточнение, что узел MySQL не найден. Причина и решение этой проблемы такое же, как и в предыдущем примере.

## \*Подключение базы данных MongoDB к Node.js

## Для создания сервера с помощью Node.js, будет использоваться библиотека Express. Для её установки в папке проекта необходимо написать:

|  |
| --- |
| npm init -ynpm install express |

## Базовое использование Express:

|  |
| --- |
| const express = require('express');const app = express();app.listen(8000);app.get('/', (req,res)=>{res.send('GET-запрос');});app.post('/', (req,res)=>{res.send('POST-запрос');}); |

## Маршрутизация определяет ответ приложения на запрос клиента к конкретному адресу, и методу запроса HTTP (GET, POST и др.). Каждый маршрут сопоставляется с одной или несколькими функциями маршрута. Вид обработчика маршрута:

## app.method(path, callback)

## app - это экземпляр express

## method - метод запроса

## path - путь на сервере

## callback - функция, выполняемая при обработке маршрута. Принимает аргументами объект запроса и ответа

## Для удобства работы с БД MongoDB будет использоваться графический интерфейс MongoDB Compass:

## 

## 

## В MongoDB данные хранятся в виде документов формата JSON, сгруппированных в коллекции. Пример создания документа продукта:

|  |
| --- |
| {"\_id": {"$oid": "65269e58951312fedaac252b"},"manufacturer\_id": 1,"name": "Пылесос Dyson V15","alias": "DysonV15","short\_description": "Ручной пылесос Dyson V15.","description": "Ручной пылесос Dyson V15 Detect Absolute UK Gray весом 3,08 кг обладает габаритами 126,4х25,2х25 см ","price": 8999.99,"image": "./assets/dysonV15","available": true,"meta\_keywords": ["пылесос", "техника", "беспроводные пылесосы"],"meta\_description": "handstick vacuum cleaner","meta\_title": "Dyson\_V\_15"} |

## Для работы с базой данных используется библиотека mongoose. Для её установки в папке проекта необходимо написать:

|  |
| --- |
| npm install mongoose |

## Для начала работы с базой данных необходимо к ней подключиться и создать схемы для используемых таблиц:

|  |
| --- |
| let mongoose = require("mongoose");mongoose.connect("mongodb://localhost:27017/db\_name",{ useNewUrlParser: true, useUnifiedTopology: true });let Schema = mongoose.Schema;let productSchema = new Schema({manufacturer\_id: Number,name: String,…});let dbProducts = mongoose.model('Products', productSchema); |

## В качестве типа данных можно указывать одно из следующих значений:

|  |  |
| --- | --- |
| StringNumberDateBufferBoolean | MixedObjectidArrayDecimal128Map |

## Пример запроса на поиск документа в БД с отправкой результата поиска:

|  |
| --- |
| dbProducts.find({ name: { $regex: '.\*' + query + '.\*' } }).then(result => {res.send(result)}); |

## Пример запроса на добавление документа в БД с отправкой сообщения об успешной записи:

|  |
| --- |
| dbProducts.create({manufacturer\_id,name,…}).then(result => res.send(`Запись успешно создана`)) |

**Раздел 5. Язык PHP.**

## Лабораторная работа №5

## Тема: Создание формы на сайт с помощью PHP

## Цель: Закрепление теоретических знаний и приобретение практических навыков работы с данными из файлов средствами языка PHP при обработке данных пользователя из форм.

## Задание: На ранее созданном сайте должно быть представлено текстовое поле, в которое пользователь должен ввести первую букву названия товара или услуги. Программа должна реализовать поиск требуемого и предоставить пользователю всю необходимую информацию о выбранном товаре или услуге. Для этого необходимо создать ассоциированный массив, содержащий названия всех товаров или услуг каталога.

## На сайте необходимо реализовать работу с многомерными массивами для хранения информации о каждом товаре или услуге.

## Для начала приведу код формы для ввода поискового запроса. Это простая разметка, даже без малейших стилевых «изысков».

## <form name="f1" method="post" action="search.php">

## <input type="search" name="search\_q"/></br>

## </br>

## <input type="submit" value="Поиск"/></br>

## </form>

|  |  |
| --- | --- |
|  |  |

## Так эта форма выглядит в браузере:

## https://webformyself.com/wp-content/uploads/2016/571/2.png

## Переходим к скрипту

## Теперь переходим непосредственно к программному коду. Для начала приведу структуру таблицы, которую нужно создать в MySQL. Мы будем искать опубликованные материалы по их заголовку. По этому же принципу можно создать PHP скрипт поиска слов по сайту.

## https://webformyself.com/wp-content/uploads/2016/571/3.png

## Сначала мы перехватываем значение, введенное пользователем в форму. Затем очищаем его от «мусора»: лишних пробелов, тегов и коннектимся к базе. После этого запускаем SQL запрос, в котором сравниваем значение поискового запроса с названием материалов, сохраненных таблице. При совпадении выводим соответствующий контент. В конце закрываем соединение с MySQL и «сбрасываем» запрос. Вот весь код примера:

|  |
| --- |
| <?php$search\_q = $\_POST['search\_q'];$l = mysqli\_connect('localhost', 'root', '', 'site');$search\_q = trim($search\_q);$search\_q = strip\_tags($search\_q);$q= mysqli\_query($l, "SELECT title\_value,content FROM `title` WHERE title\_value LIKE '%$search\_q%'");$itog=mysqli\_fetch\_assoc($q);while ($itog = mysqli\_fetch\_assoc($q)) {printf("%s (%s)\n",$r["title\_value"],$r["content"]);}mysqli\_free\_result($q);mysqli\_close($l);?> |

|  |  |
| --- | --- |
|  |  |

## https://webformyself.com/wp-content/uploads/2016/571/4.png

## Код следует разместить в отдельном файле PHP на стороне сервера, а его название прописать в форме (атрибут action).

## Отображение результатов запроса на странице

## Чтобы отобразить результат запроса к БД, можно использовать Fetch API JavaScript. Пример использования с HTML:

|  |
| --- |
| <html><body><div id = "output"> </div><button onclick = "fetchData()"> Fetch API to get data </button><script>let output = document.getElementById('output');function fetchData() {fetch('localhost/getProducts.php').then(response => response.json()).then(data => {output.innerHTML += "name = " + data.name + "<br/>";output.innerHTML += "description = " + data.description + "<br/>";…})}</script></body></html> |

## Пример использования с React:

|  |
| --- |
| import React, { useState, useEffect } from 'react';function App() {const [data, setData] = useState([]);useEffect(async () => {const result = await fetch('localhost/getProducts.php ',);setData(result);});return (<ul>{data.map(product => (<li key={product.id}><h2>{product.name}</h2><div>{product.description}</div></li>))}</ul>);}export default App; |

## \*Провести валидацию и отправку значения полей формы поиска.

*Лист*

*Лист*