Урок 5

Базы данных MySQL и работа с ними на уровне PHP

Краткий обзор БД и СуБД. Знакомство с реализацией общения между PHP и MySQL. Краткий обзор угрозы SQL-инъекций

На предыдущем занятии мы познакомились с принципами и функционалом взаимодействия языка PHP с файлами. Мы научились сохранять и воспроизводить информацию, однако данный подход к хранению данных имеет множество недостатков: неструктурированность, неоптимальное расходование памяти, ресурсоёмкий поиск данных и так далее.
Например, нам необходимо хранить информацию о книгах в библиотеке внутри текстового файла, где каждая строка представляет собой запись об одной книге. Довольно естественной представляется ситуация, в которой нужно найти книгу по её названию. Поскольку PHP не предоставляет встроенных функций по работе с поиском данных в файлах, нам придётся реализовывать некий алгоритм, который будет пробегать все строчки в файле и искать нужные вхождения, – а это не самый лучший вариант. Поэтому в данном уроке мы выясним, как можно избежать всех этих недостатков.

Что такое база данных

База данных – это совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств их моделирования.
Базы данных, как правило, уже имеют готовый функционал для хранения информации, которую мы можем выводить на веб-страницы. Информация в базе данных может быть структурирована в зависимости от задачи – в ней могут храниться обычные статьи, каталоги товаров, списки пользователей и многое другое.

Типы баз данных

В настоящий момент базы данных разделяются на так называемые SQL и NoSQL решения. Поддерживающие запросы на языке SQL (Structured Query Language) БД применяются для тех данных, которые можно представить в виде таблицы, т.е. есть возможность задать их более-менее постоянную структуру. Это новости, товары, расписания. Таблицы можно связывать друг с другом как посредством программного кода, так и средствами самой базы данных.

NoSQL решения используются для данных, которые нельзя структурировать. Иными словами, одна запись будет иметь один набор полей, а другая – совершенно другой. Например, кэшированные данные для разных частей страниц могут смело храниться в одной структурной единице такой БД.

Оба этих решения имеют свои плюсы и минусы, которые должны быть учтены при разработке архитектуры вашей программы.

Само по себе изучение механизмов работы БД и построения SQL-запросов – отдельная большая тема, изучаемая в рамках отдельного курса. Мы лишь познакомиться с самыми простыми запросами и способами их использования.

Не стоит путать базу данных и системы управления базами данных (БД и СУБД). СУБД – это программа, которая предоставляет доступ внешним приложениям к базе данных и обеспечивает её работу. Популярные СУБД – это Oracle, Microsoft SQL Server, MySQL, PostgreSQL. Сайты PHP чаще всего работают в связке с MySQL, поэтому именно данную СУБД мы будем рассматривать в текущем уроке.

БД под управлением MySQL состоит из таблиц и связей между ними. Как известно, любая таблица – это набор столбцов и строк. Столбцов создаётся определённое количество, в то время как строк может быть сколько угодно. Столбец является более важным элементом, чем строка, так как последняя отвечает лишь за конкретную запись в таблице, столбец же – за наличие определённой характеристики у всех записей. Это значит, что каждый ряд является отдельным набором данных.

Команды в БД

Сам по себе сервер базы данных очень похож на веб-сервер. Он занимает определённый порт или сокет, слушает входящие соединения и отвечает на их запросы. Используя специальную консольную утилиту, мы можем получить доступ к нашей базе.

Получение структурированных данных осуществляется с помощью различных команд – запросов. Поскольку начинающему разработчику достаточно сложно ориентироваться в таком представлении данных, лучше использовать специальные утилиты, вроде PHPMyAdmin или HeidiSQL.

Перед тем, как начать работу, мы создадим необходимое окружение. Нам понадобятся:

1. База данных.
2. Пользователи для доступа к ней.

При помощи удобной утилиты выполним запрос, который создаст для нас новую БД.

```
CREATE DATABASE geekbrains
```

Теперь для неё создадим пользователей. Во многих утилитах это делается через очень удобный интерфейс. Не забудьте создавать для всех пользователей пароли!

Представим, что нам нужно хранить информацию о сотрудниках фирмы. При проектировании таблицы сначала нужно просто представить, а какие вообще данные о каждом человеке нам нужны. Например, мы решили, что будем хранить фамилию, имя и отчество сотрудника. Тогда таблица может выглядеть следующим образом:

id_employee	first_name	middle_name	last_name
1	Иван	Иванович	Иванов
2	Петр	Петрович	Петров
3	Павел	Павлович	Павлов
4	Елена	Ивановна	Иванова
5	Елена	Петровна	Петрова

Обратите внимание, что в таблице создаётся дополнительное поле id_employee. Это так называемый первичный ключ – уникальный номер конкретной строки, по которому мы всегда сможем её получить. У каждой таблицы в реляционной базе данных должен быть первичный ключ.

Для того, чтобы создать такую таблицу нужно выполнить следующий запрос:

```
CREATE TABLE employee(
id_employee int (11) NOT NULL AUTO_INCREMENT,
first_name varchar(255) NULL DEFAULT '',
middle_name varchar(255) NULL DEFAULT '',
last_name varchar(255) NULL DEFAULT '',
PRIMARY KEY (`id`)
);
```

Дополнить таблицу можно позже с помощью команды ALTER.

Теперь мы можем добавить данные в нашу таблицу. Воспользуемся командой INSERT. Нужно указать, в какую ячейку мы хотим сохранить тот или иной набор данных.

```
INSERT INTO employee (first_name, middle_name, last_name) VALUES ('testuser', 'test', 'test');
```

Обратите внимание на то, что при наличии у поля id_employee свойства автоинкрементирования, его указывать не обязательно – оно само примет нужное значение.
Команда UPDATE позволяет обновить все значения определённой колонки, либо, используя условие WHERE, обновить только определённые ячейки, под это условие подпадающие. Команда DELETE удаляет либо все ряды, либо ряды, удовлетворяющие условию WHERE.

```
UPDATE employee SET name='testuser1' WHERE id_employee=1;
DELETE FROM employee WHERE id_employee=5;
```

Разумеется, что приведённые примеры – самые простые. В больших проектах часто случается так, что запрос может быть настолько огромным, что не вмещается на целую страницу текста. Задача разработчика сайта состоит в том, чтобы запрос обрабатывался максимально быстро и возвращал корректные данные.

Запрос SELECT позволяет нам выгрузить добавленные данные. Мы должны сообщить, какие именно (или все) ячейки нам нужны для работы, из какой таблицы, и обозначить условия выбора. В эти условия могут входить определённые значения тех или иных ячеек или сравнение с этими значениями (больше–меньше). Мы можем указать группировку данных, порядок их вывода и ограничение на количество элементов в выборке.

```
SELECT * FROM employee WHERE id > 0;
```

Работа с БД средствами PHP

Порядок работы с базой данных в PHP похож на порядок работы с файлами, но имеет определённые особенности:

1. Установить соединение с сервером.
2. Выбрать конкретную базу данных.
3. Выполнить операции.
4. Закрыть соединение.

Рассмотрим по очереди все эти шаги. Для подключения к базе в PHP существует специальная функция mysqli_connect. Чаще всего она вызывается с четырьмя параметрами:

```
$link = mysqli_connect("localhost", "my_user", "my_password", "world");
```

Здесь по порядку указываются адрес сервера БД, имя пользователя, пароль и имя БД. Поскольку мы уже умеем хранить настройки в файле конфигурации, то данные для подключения лучше всего брать оттуда.

В конце работы приложения соединение с БД нужно закрывать через команду:

```
mysqli_close($link);
```

Теперь можно совершать действия, которые мы рассматривали в предыдущем пункте. За выполнение запроса к базе отвечает функция mysqli_query, которая принимает строку с командой на языке SQL и возвращает результат её выполнения, например:

```
$result = mysqli_query("SELECT * FROM employee WHERE id > 0");
```

Для запросов, не подразумевающих получение данных из базы (INSERT, UPDATE, DELETE), функция возвращает true, в случае успешного выполнения операции, и false – в противном случае. Для SELECT запросов функция возвращает дескриптор с результатом выборки, в случае успешного выполнения операции, и false – в противном случае. Таким образом, для дальнейшей работы необходимо преобразовать дескриптор в массив с данными результатов выборки.

```
$epms = array();
while($row = mysqli_fetch_assoc($result))
  $epms[] = $row;
```

Внимательно разберёмся с данным кодом. В переменной $result оказывается результат выборки из базы – список всех сотрудников. Далее мы создаём массив $epms, в который будем добавлять информацию о каждой записи.
Теперь запускаем цикл, который ориентируется на функцию mysqli_fetch_assoc. Она извлекает очередную строку из выборки данных и возвращает её в переменную $row в виде ассоциативного массива. Ключами данного массива являются названия столбцов таблицы, а значениями – данные из конкретной строки. Когда строки закончатся, mysql_fetch_assoc вернёт значение false, и цикл завершится.
На выходе мы получим двумерный массив $epms с информацией о сотрудниках. В нём будет столько же элементов, сколько строк вернула база в результате выборки. Каждый его элемент – ассоциативный массив с информацией об одном сотруднике.
Данная процедура является стандартной. Её достаточно освоить один раз, а затем лишь применять на практике.

Существуют и другие полезные функции при работе с базой, например:

- **mysql_num_rows** – число строк, содержащееся в результате выборки данных;
- **mysql_affected_rows** – число строк, затронутых последним запросом INSERT, UPDATE
- или DELETE;
- **mysql_error** – сообщение о последней ошибке, возникшей в ходе запроса;
- **mysql_insert_id** – id записи, добавленной последним запросом INSERT;
- **mysql_close** – закрывает соединение с сервером MySql.

Применение на проекте

Теперь, когда мы умеем работать с БД, её можно применять на нашем сайте. В качестве примера мы создадим раздел новостей, которые будут храниться в БД. Для реализации этого механизма нам понадобится:

1. Библиотека для работы с БД.

2. Логика вывода новостей в виде ленты.
3. Вывод конкретной новости.

Итоги

Мы ознакомились с самыми основами языка запросов SQL, научились сохранять и получать данные из базы данных.
Используя эти знания, мы можем генерировать страницы из данных из БД, можем создать панель управления сайтом и забыть про наполнение контентом вручную через исходный код.

Домашнее задание

После предыдущего урока вы реализовали галерею изображений с помощью файлов и папок. Теперь необходимо сделать то же самое, но для хранения имен картинок следует использовать базу данных.

1. Создать галерею изображений, состоящую из двух страниц.
    a. Просмотр всех фотографий (уменьшенных копий).
    b. Просмотр конкретной фотографии (изображение оригинального размера) по её ID в базе данных.
2. В базе данных создать таблицу, в которой будет храниться информация о картинках (адрес на сервере, размер, имя).
3. * На странице просмотра фотографии полного размера под картинкой должно быть указано число её просмотров.
4. * На странице просмотра галереи список фотографий должен быть отсортирован по популярности. Популярность – число кликов по фотографии.

Дополнительные материалы

1. https://habrahabr.ru/post/44807/ – MySQL и JOIN-ы.
2. http://devenergy.ru/archives/category/sql – немного о SQL в блоге.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. «Руководство по MySQL» – Сейед Тахагхогхи, Хью Е. Вильямс.