Особенности тестирования web-приложений

Введение

- Web-ориентированные приложения стали широко популярными в конце 1990-х - начале 2000-х годов. Бурное развитие Internet - технологий и сети Internet приводит к тому, что web-приложения становятся более актуальными, распространенными и все более сложными, играя таким образом основную роль в большинстве online проектов

- В классических приложениях вся функциональность заключена в одной программе, которая полностью находится на одном компьютере. Классические приложения оказываются всегда более понятными. Их структура и поведение может меняться, но очень в узких рамках. Методики тестирования уже разработаны и наработаны

Web-приложение

- Web-приложение - это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером web-сервер, что уже является по сути двумя разнопольными программами, которые необходимо тестировать как отдельно, так и в связке. Кроме веб-сервера, приложение может использовать базы данных, другие приложения и удаленные веб-сервисы

Структура приложения

- Любое веб-приложение представляет собой набор статических и динамических веб-страниц.
- Статическая веб-страница – это страница, которая всегда отображается перед пользователем в неизменном виде. Веб-сервер отправляет страницу по запросу веб-браузера без каких-либо изменений
- В противоположность этому, сервер вносит изменения в динамическую веб-страницу перед отправкой ее браузеру. По причине того, что страница меняется, она называется динамической

Статические страницы

- Статический веб-сайт содержит набор соответствующих HTML-страниц и файлов, размещенных на компьютере, на котором установлен веб-сервер
- Веб-сервер — это программное обеспечение, которое предоставляет веб-страницы в ответ на запросы веб-браузеров. Обычно запрос страницы создается при щелчке ссылки на веб-странице, выборе закладки в браузере либо вводе URL-адреса в адресной строке браузера
- Окончательное содержимое статической веб-страницы определяется разработчиком и остается неизменным в процессе запроса страницы

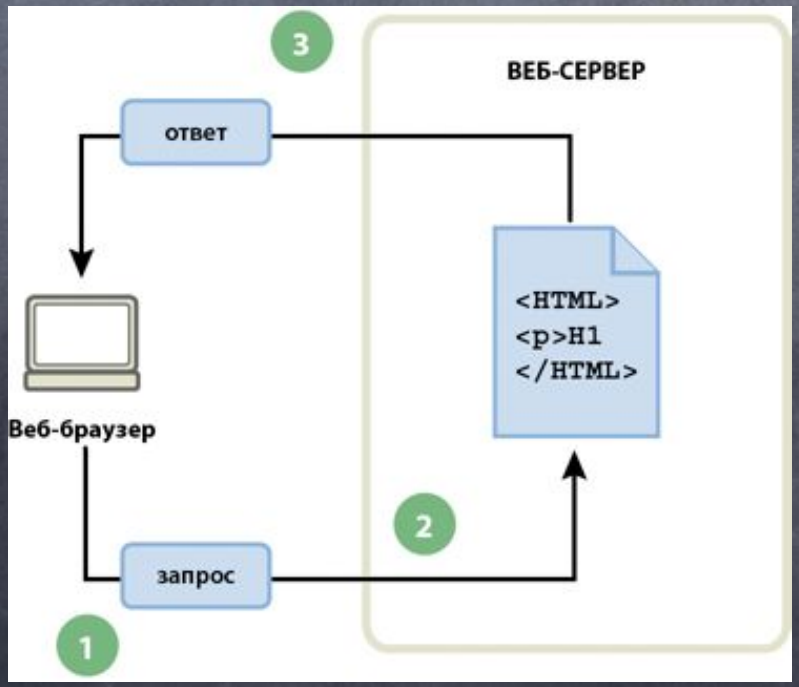
Статические страницы

```
<html>
    <head>
        <title>Trio Motors Information Page</title>
    </head>
    <body>
        <h1>About Trio Motors</h1>
        <p>Trio Motors is a leading automobile manufacturer.</p>
    </body>
</html>
```

Весь HTML-код создается разработчиком до того момента, когда страница будет размещена на сервере. Поскольку HTML-код не меняется после размещения страницы на сервере, данная страница называется статической

Статические страницы

- Когда веб-сервер получает запрос на выдачу статической страницы, то, после анализа запроса, сервер находит нужную страницу и отправляет ее браузеру

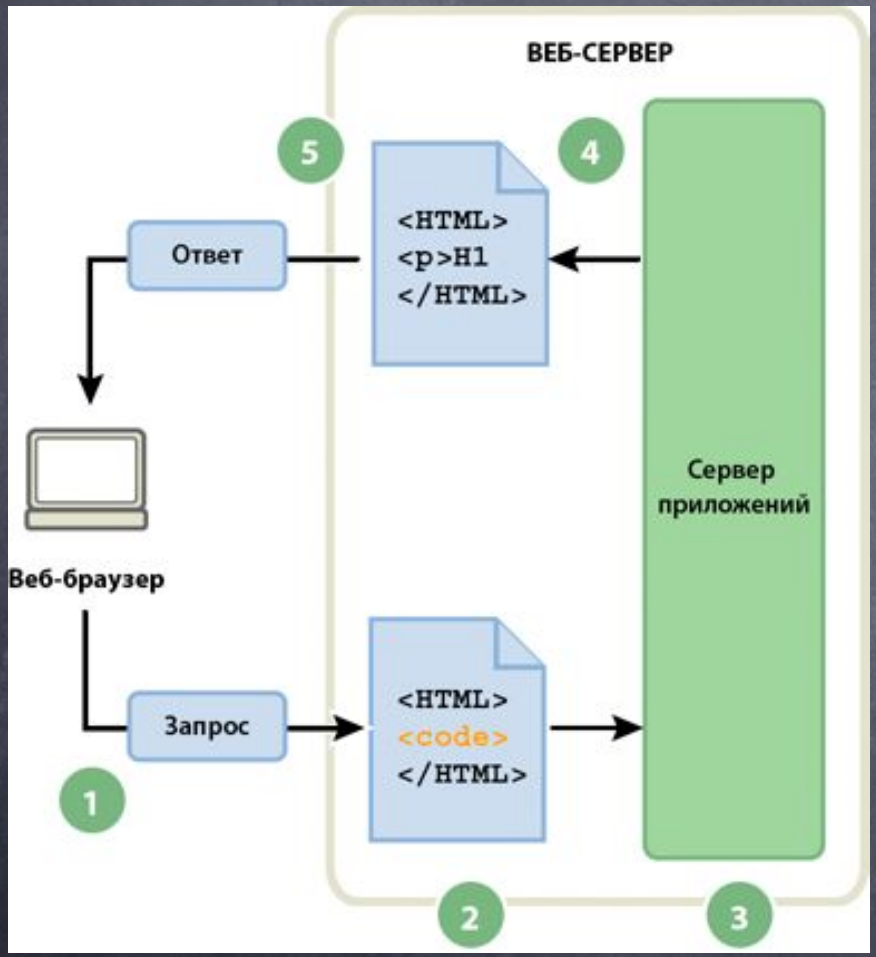


Динамические страницы

- Когда веб-сервер получает запрос на выдачу статической веб-страницы, он отправляет страницу непосредственно браузеру. Однако, когда запрашивается динамическая страница, действия веб-сервера не столь однозначны. Сервер передает страницу специальной программе, которая и формирует окончательную страницу. Такая программа называется сервером приложений

- Сервер приложений выполняет чтение кода, находящегося на странице, формирует окончательную страницу в соответствии с прочитанным кодом, а затем удаляет его из страницы. В результате всех этих операций получается статическая страница, которая передается веб-серверу, который в свою очередь отправляет ее клиентскому браузеру. Все страницы, которые получает браузер, содержат только HTML-код

Динамические страницы



1. Веб-браузер запрашивает динамическую страницу
2. Веб-сервер находит страницу и передает ее серверу приложений
3. Сервер приложений просматривает страницу на наличие инструкций и выполняет ее создание
4. Сервер приложений возвращает подготовленную страницу на веб-сервер
5. Веб-сервер отправляет подготовленную страницу запросившему ее браузеру

STORMNET

Основы HTTP

- Протокол http реализован на прикладом уровне модели OSI и базируется на архитектуре клиент - сервер. Клиент посылает серверу запрос на который сервер в свою очередь отвечает.
- Простой пример: вы решили зайти на www.google.ru для того чтобы что-нибудь найти. Вы (то есть клиент) применяете для этого свой браузер, который, являясь клиентским приложением, отсылает запрос серверу, в котором просит предоставить ему содержимое главной страницы сайта. Сервер обрабатывает этот запрос и посылает вам ответ, содержащий запрашиваемую страницу. Ваш браузер в свою очередь обрабатывает html код и в результате вы видите то что вы видите.

Структура HTTP-запроса

- Запрос в большинстве случаев выглядит так:

<метод> <запрашиваемый_ресурс> HTTP/1.1<\n>
<заголовочное_поле>: <значение><\n>
<заголовочное_поле>: <значение><\n>
...
<заголовочное_поле>: <значение><\n>
<\n>

- Метод – характеризует текущий запрос. Их несколько, например head, post, put, delete, get

Ответ сервера

- Ответ сервера выглядит следующим образом:

HTTP/1.1 <код ответа> <сообщение><\n>
<заголовочное_поле>: <значение><\n>
<заголовочное_поле>: <значение><\n>
...
<заголовочное_поле>: <значение><\n>
<\n>
<тело документа>

- Код ответа нужен для того, чтобы оповестить клиента о происходящем. Например о том, что запрашиваемая страница не найдена или о том, что произошла ошибка

Коды ответов сервера

- 200 OK
- 201 Успешная команда post
- 202 Запрос принят
- 203 Запрос get либо head выполнен
- 204 Запрос выполнен, но нет содержимого
- 300 Ресурс обнаружен в нескольких местах
- 301 Ресурс удален навсегда
- 302 Ресурс временно удален
- 304 Ресурс изменен
- 400 Плохой запрос от клиента
- 401 Неавторизованный запрос
- 402 Необходима оплата за запрос ("зарезервированно для дальнейшего использования", на данный момент не используется)
- 403 Доступ к ресурсу запрещен
- 404 Ресурс не найден
- 405 Метод неприменим для данного ресурса
- 406 Недопустимый тип ресурса
- 410 Ресурс недоступен
- 500 Внутренняя ошибка сервера
- 501 Метод не выполнен
- 502 Перегрузка сервера или неисправный шлюз
- 503 Сервер недоступен или таймаут шлюза

GET-Запрос

- GET-запрос — это самый распространенный вид HTTP-запроса. При помощи него происходит запрос браузером любого файла веб-сервера. HTTP-запрос типа GET состоит только из HTTP-заголовков, тело у него отсутствует
- При помощи GET-запроса можно передать веб-серверу параметры — некоторую информацию. Например, если на сайте предусмотрена авторизация пользователей, то с помощью параметров можно передавать имя пользователя и пароль для проверки. Рассмотрим механизм передачи параметров с помощью GET-запроса

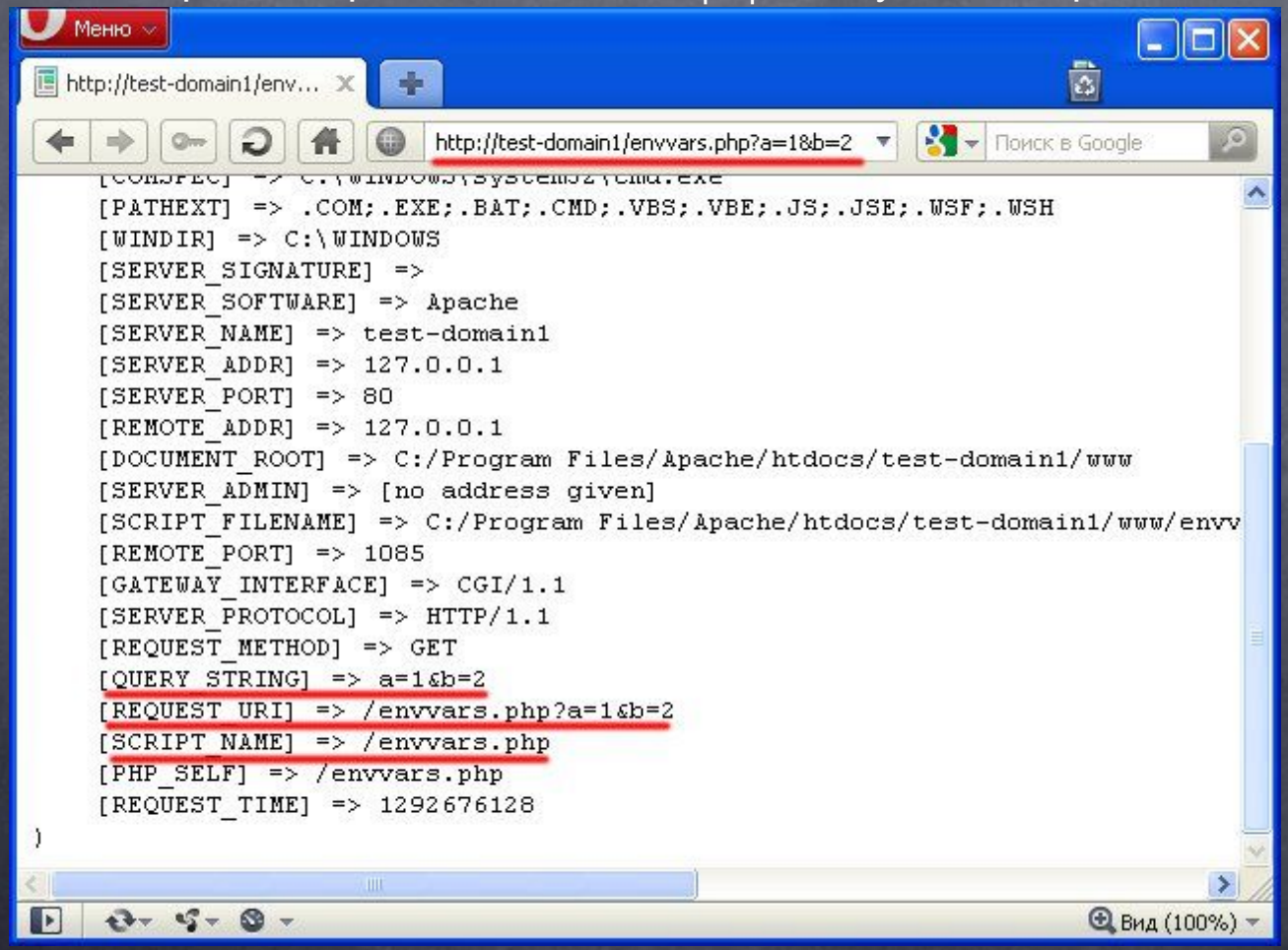
GET-Запрос

- В первой строке запроса после ключевого слова GET помещается путь к запрашиваемому документу. Если в пути встречается знак вопроса, то принято считать, что в этом месте путь заканчивается, а за ним начинаются GET-параметры этого запроса
- *GET /examples/test.html?параметры HTTP/1.1*

- GET-параметр имеет формат имя_параметра=значение_параметра, сами параметры разделяются знаком &. Пример GET-запроса с двумя параметрами:
- *GET /enter?login=admin&password=qwerty HTTP/1.1*

GET-Запрос

- GET-параметры можно вручную дописать в браузере в конце запрашиваемого URL в адресной строке. Обратимся к envvars.php следующим образом:



GET-Запрос

- Например, имеется файл script.php в корневом каталоге test-domain1:

```php
<?php
  echo "Ваше имя: " . $_GET["name"] . "<br />";
  echo "Ваш возраст: " . $_GET["age"] . "<br />";
?>
```

- результатом запроса *http://test-domain1/script.php?name=John&age=33* будет:

```
Ваше имя: John
Ваш возраст: 33
```

GET-Запрос

- Основным преимуществом GET-параметров является их размещение непосредственно в URL, что дает возможность сформировать гиперссылку на документ с определенными параметрами (Например, ссылка на статью на сайте)
- Основным недостатком является ограничение длины URL - 255 байтов/255 символов
- Отдельным недостатком можно выделить явную передачу данных через строку браузера из соображений безопасности (можно визуально запомнить передаваемый логин/пароль или другую важную информацию)
- Так же, через GET-запрос можно передать в скрипт вредоносную информацию в виде MySQL-инъекции или XSS-уязвимости

POST-Запрос

- Если необходимо передать на веб-сервер большой объем данных, например, текст сообщения или файл, используют POST-запрос. В этом типе запроса параметры помещаются в тело HTTP-запроса, а размер передаваемых данных в байтах указывается в заголовке Content-Length:

```
POST /enter HTTP/1.1
<Различные заголовки>
Content-Length: 27
<Различные заголовки>

login=admin&password=qwerty
```

- Таким образом, в URL передаваемые параметры не видны. Простым способом сформировать POST-запрос не получится, они в основном генерируются с помощью HTML-форм

Основы MySQL

- Сервер приложений предоставляет возможность использовать такие ресурсы сервера, как базы данных. Например, динамическая страница может содержать программные инструкции для сервера приложений, следуя которым серверу необходимо получить определенные данные из базы данных и поместить их в HTML-код страницы
- Хранение содержимого в базе данных позволяет отделить оформление веб-сайта от содержимого, которое будут видеть пользователи. Вместо того чтобы создавать все страницы в виде отдельных HTML-файлов, пишутся только шаблоны страниц для каждого вида представляемой информации. Затем содержимое загружается в базу данных, после чего веб-сайт будет извлекать его при запросах пользователей. Кроме того, можно обновить информацию в одном источнике и продублировать это изменение на всем веб-сайте без редактирования каждой страницы вручную

Основы MySQL

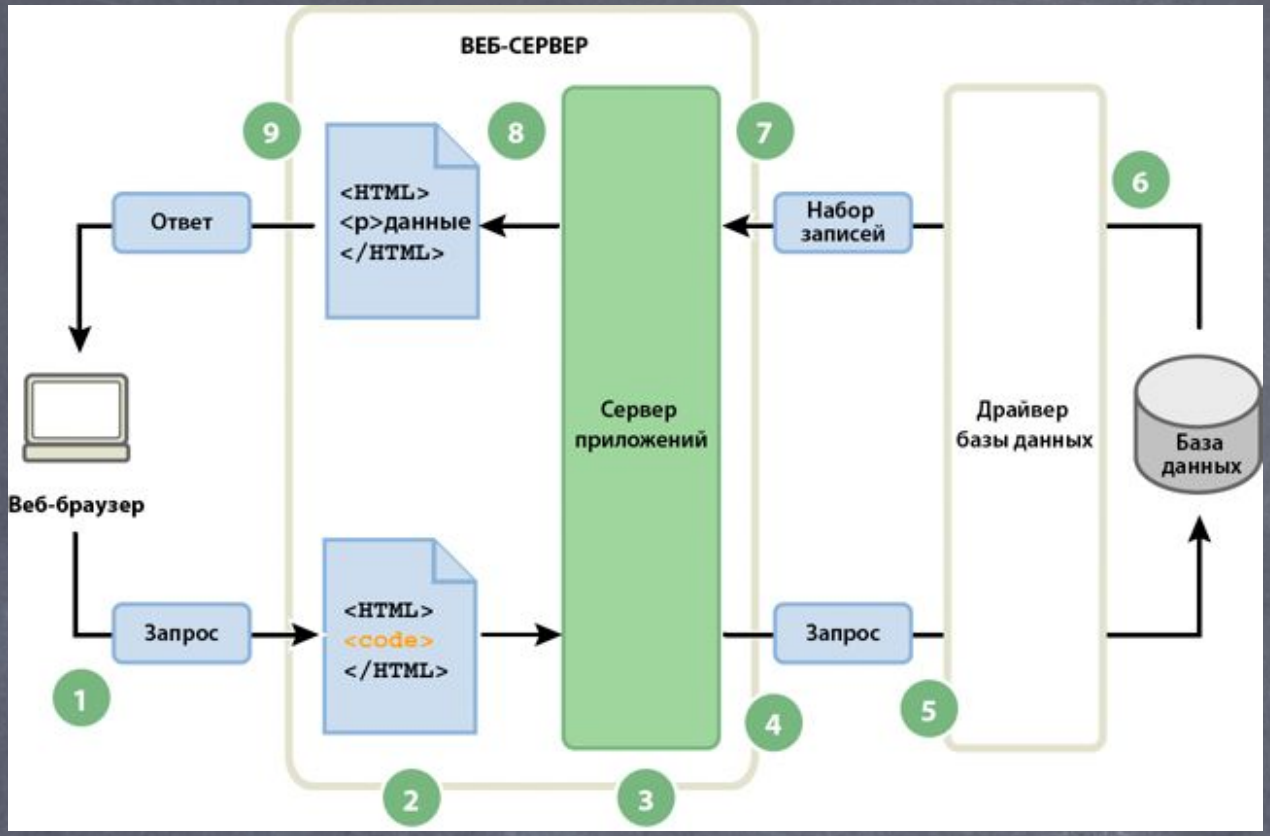
- Программная инструкция, предназначенная для получения данных из базы данных, называется запросом к базе данных
- Запрос состоит из критериев поиска, выраженных с помощью языка баз данных, называемого SQL (язык структурированных запросов). Текст SQL-запроса располагается в сценариях страниц на стороне сервера

Основы MySQL

- Сервер приложений не может непосредственно получить данные из базы, поскольку базы данных используют специфические форматы хранения данных, в результате чего попытка получения таких данных будет напоминать попытку открытия документа Microsoft Word с помощью текстового редактора Notepad или BBEdit. Поэтому для подключения к базе данных сервер приложений использует посредника — драйвер базы данных
- Драйвер базы данных представляет собой программный модуль, с помощью которого устанавливается взаимодействие между сервером приложений и базой данных

Основы MySQL

- После того как драйвер установит соединение, выполняется запрос к базе, в результате чего формируется набор записей. Набор записей представляет собой множество данных, полученных из одной или нескольких таблиц базы данных. Набор записей возвращается серверу приложений, который использует полученные данные для формирования страницы



1. Веб-браузер запрашивает динамическую страницу
2. Веб-сервер находит страницу и передает ее серверу приложений
3. Сервер приложений просматривает страницу на наличие инструкций и выполняет ее подготовку
4. Сервер приложений отправляет запрос драйверу базы данных
5. Драйвер выполняет запрос в базе данных
6. Драйверу возвращается набор записей
7. Драйвер передает набор записей серверу приложений
8. Сервер приложений вставляет данные в страницу и передает страницу веб-серверу
9. Веб-сервер отправляет подготовленную страницу запросившему ее браузеру.

Основы MySQL

- Пример запроса

```
1 | SELECT lastname, firstname, fitpoints
2 | FROM employees
```

- С помощью этой инструкции формируется набор записей из трех столбцов, содержащих фамилию, имя и набранные баллы всех сотрудников, сведения о которых хранятся в базе данных

Основные команды MySQL (Управление базами)

- Создать базу данных на SQL сервере
- mysql> create database [databasename];

- Просмотреть весь список баз данных на SQL сервере
- mysql> show databases;

- Переключить базу данных.
- mysql> use [db name];

- Смотреть все таблицы в базе данных
- mysql> show tables;

- Удалить базу данных
- mysql> drop database [database name];

Основные команды MySQL (Управление таблицами)

- Создать таблицу ‘supportContacts’ с полями: Автоматический счетчик ‘id’, текстовое поле на 20 символов ‘type’ и текстовое поле на 30 символов ‘details’.
- CREATE TABLE supportContacts (id int auto_increment primary key, type varchar(20), details varchar(30));

- Удаление таблицы ‘users’.
- DROP TABLE `users`

Основные команды MySQL (Получение данных)

- Смотреть все данные в таблице
- mysql> SELECT * FROM [table name];

- Показать определенные выбранные строки со значением "whatever".
- mysql> SELECT * FROM [table name] WHERE [field name] = "whatever";

- Посмотреть все записи, содержащие название "Bob" и номер телефона, '3444444 '.
- mysql> SELECT * FROM [table name] WHERE name = "Bob" AND phone_number = '3444444';

Основные команды MySQL (Условия поиска данных)

- Показать все записи, начинающиеся с букв 'Bob' и номер телефона, '3444444 '.
- mysql> SELECT * FROM [table name] WHERE name like "Bob%" AND phone_number = '3444444';

- Показать все записи, начинающиеся с букв 'Bob' и номер телефона, '3444444 ' в пределах от 1 до 5.
- mysql> SELECT * FROM [table name] WHERE name like "Bob%" AND phone_number = '3444444' limit 1,5;

- Показать уникальные записи
- mysql> SELECT DISTINCT [column name] FROM [table name];

Основные команды MySQL (Сортировка данных)

- Показать все записи, не содержащей название "Bob" и номер телефона, '3444444 'сортировка по phone_number области.
- mysql> SELECT * FROM [table name] WHERE name != "Bob" AND phone_number = '3444444' order by phone_number;

- Показать выбранные записи отсортированы по возрастанию (ASC) или по убыванию (DESC).
- mysql> SELECT [col1], [col2] FROM [table name] ORDER BY [col2] DESC;

Основные команды MySQL (Управление данными)

- Для обновления информации, которая уже находится в таблице. Поменять имя Bob на Bill
- mysql> UPDATE [table name] SET name = 'Bill' where name = 'Bob';

- Удалить строку (и) из таблицы.
- mysql> DELETE from [table name] where [field name] = 'whatever';

- Добавить строку в таблицу.
- INSERT INTO [table name] ([field name 1], [field name 2]) VALUES ('[value 1]', '[value 2]');