10. Конструктор запросов. SELECT

Получение всех записей таблицы

Для создания запроса необходимо использовать метод **table()** фасада **DB**. Фасад DB должен быть подключен:

```
use Illuminate\Support\Facades\DB;
```

Метод table() возвращает экземпляр конструктора запросов для данной таблицы, позволяя вам "прицепить" к запросу ***дополнительные условия*** и в итоге ***получить результат методом*** **get()**:

```
public function index(){
    // получаем все записи таблицы "users"
    $users = DB::table('users')->get();
    // отдаем результат в представление
    return view('index')->(['users' => $users]);
}
```

Метод get() возвращает ***объект*** Illuminate\Support**Collection** (для более ранних версий — массив) с результатом, в котором ***каждый элемент коллекции*** — это экземпляр ***PHP-объекта*** **StdClass**.

Получить доступ к элементу коллекции можно используя квадратные скобки:

```
$users = DB::table('users')->get();
dump($users[0]);
```

Получить значение каждого столбца в коллекции можно обращаясь к столбцу как к свойству объекта:

```
$users = DB::table('users')->get();

dump($users[0]->name);
```

Файл **ex_1/MainController.php**. Метод get()

```
// обработчик маршрута http://host/music/teams/get
public function acTeamsGet() {
     // метод get - извлекаем коллекцию данных о группах
     $teams = DB::table("team")->get();
     // где $teams:
     // объект Collection (коллекция), в которой
     // каждый результат — экземпляр объекта StdClass
     dump($teams);
     // к коллекции можно обратиться как к элементам массива
     dump($teams[0]);
     // каждый элемент коллекции - объект, обращаемся к объекту:
     dump($teams[0]->name);
}
```

Условия WHERE

Для добавления в запрос условий используйте метод **where()** на экземпляре конструктора запросов.

Самый простой вызов where() требует ***три аргумента***.

1. ***имя столбца***,
2. ***оператор*** (любой из поддерживаемых базой данных),
3. ***значение для сравнения*** со столбцом.

Например, запрос, проверяющий равенство значения столбца **like** и значения для сравнения **100**:

```
$users = DB::table('articles')->where('like', '=', 100)->get();
```

> Если необходимо проверить ***равенство значения столбца с значением для сравнения***, можно передать значение вторым аргументом метода where().

Такой запрос ***аналогичен*** предыдущему:

```
$users = DB::table('articles')->where('like', 100)->get();
```

Разумеется, можно использовать различные ***другие операторы*** при написании условия where:

```
$users = DB::table('articles')
     ->where('like', '>=', 100)
     ->get();
$users = DB::table('articles')
     ->where('visible', '<>', 1)
     ->get();
```

В функцию where() можно передать ***массив условий***:

```
$users = DB::table('users')->where([
     ['status', '<>', '1'],
     ['role', '=', 'admin'],
     ])->get();
```

Вы можете сцепить ***несколько условий*** **where**, через условный оператор **or**. Метод **orWhere()** принимает те же аргументы, что и метод where():

```
$users = DB::table('articles')
     ->where('like', '>', 100)
     ->orWhere('views', '>=', 5000)
```

```
    ->get();
```

Получение одной записи из таблицы

Если необходимо получить только ***одну строку*** из таблицы БД, используйте метод **first()**.

Независимо от количества записей в таблице "users" ***метод*** **first** вернёт ***один объект*** **StdClass**:

```
$user = DB::table('users')->first();

echo $user->name;
```

Чаще всего метод first применяют ***совместно с условием*** **where**:

```
$user = DB::table('users')->where('id', '=', '12')->first();

echo $user->name;
```

Файл **ex_1/MainController.php**. Метод first()

```
// обработчик маршрута http://host/music/team/first/{id}
public function acTeamFirst($id) {
    // метод first - извлекаем  объект с данными о группе
    // нет условия выборки, first вернет первый объект
    $team = DB::table("team")->first();
    // first вернул объект StdClass
    dump($team);
    // обращаемся как к свойству объекта
    dump($team->name);
    // есть условие выборки, метод first вернет один объект
    // согласно условия where ("id_team" = $id)
    $team = DB::table("team")->where("id_team", "=", $id)-
    >first();
```

```
    // объект StdClass
    dump($team);
    // обращаемся как к свойству объекта
    dump($team->name);
}
```

Указание столбцов для выборки

Не всегда необходимо выбирать все столбцы из таблицы БД. Используя метод select() вы можете указать необходимые столбцы для запроса:

```
$users = DB::table('team')->select('name', 'alias')->get();
```

При выборке столбцов допускается указание ***алиасных имен***:

```
$users = DB::table('team')->select('name', 'alias as als')->get();
```

Если у вас уже есть ***экземпляр конструктора запросов*** и вы хотите добавить столбец к существующему набору для выборки, используйте метод **addSelect()**:

```
$teams = DB::table("team")->select("name", "alias as als");
$teams->addSelect("country")->get();
dump($teams);
```

Файлы раздаточного материала

ex_1/ - Получение данных методом **first** и **get**.

ex_2/ - Маршруты для работы с таблицей **team**.

ex_3/ - Маршруты для работы с таблицей **album**.

ex_4/ - Маршруты для работы с таблицей **track**.