Eloquent

- Инкрементация и декрементация
- X или Y методы
- Метод boot() модели
- Свойства модели: timestamps, appends и тд.
- WhereX
- Сортировка по умолчанию

Инкрементация и декрементация

Вместо этого:

```
$article = Article::find($article_id);
$article->read_count++;
$article->save();
```

Вы можете сделать так:

```
$article = Article::find($article_id);
$article->increment('read_count');
```

Так тоже будет работать:

```
Article::find($article_id)->increment('read_count');
Article::find($article_id)->increment('read_count', 10); // +10
Product::find($produce_id)->decrement('stock'); // -1
```

X или Y методы

Eloquent есть несколько функций, которые объединяют два метода, например “пожалуйста, сделай X, иначе сделай Y”.

Пример 1 – `findOrFail()`:

Вместо этого:

```
$user = User::find($id);
if (!$user) { abort (404); }
```

Делаем это:

```
$user = User::findOrFail($id);
```

Пример 2 – `firstOrCreate()`:

Вместо этого:

```
$user = User::where('email', $email)->first();
if (!$user) {
  User::create([
    'email' => $email
  ]);
}
```

Делаем это:

```
$user = User::firstOrCreate(['email' => $email]);
```

Метод boot() модели

В модели Eloquent есть волшебный метод `boot()`, где вы можете переопределить поведение по умолчанию:

```
class User extends Model
{
    public static function boot()
    {
        parent::boot();
        static::updating(function($model)
        {
            // выполнить какую-нибудь логику
            // переопределить какое-нибудь свойство, например $model->something =
transform($something);
        });
    }
}
```

Вероятно, одним из наиболее популярных примеров является установка значения поля на момент создания объекта модели. Предположим, вы хотите сгенерировать поле UUID в этот момент.

```
public static function boot()
{
  parent::boot();
  self::creating(function ($model) {
    $model->uuid = (string)Uuid::generate();
  });
}
```

Свойства модели: timestamps, appends и тд.

Существует несколько «параметров» Eloquent модели в виде свойств класса. Самые популярные из них, вероятно, следующие:

```
class User extends Model {
    protected $table = 'users';
    protected $fillable = ['email', 'password']; // какие поля могут быть заполнены выполняя User::create()
    protected $dates = ['created_at', 'deleted_at']; // какие поля будут типа Carbon
    protected $appends = ['field1', 'field2']; // доп значения возвращаемые в JSON
}
```

Но есть еще:

```
protected $primaryKey = 'uuid'; //  не должно быть "id"
public $incrementing = false; // и не должно быть автоинкрементом
protected $perPage = 25; // Да, вы можете переопределить число записей пагинации (по умолчанию 15)
const CREATED_AT = 'created_at';
const UPDATED_AT = 'updated_at'; // Да, даже эти названия также могут быть переопределены
public $timestamps = false; // или не использоваться совсем
```

И есть еще больше, для более подробной информации ознакомьтесь с кодом по умолчанию abstract Model class и посмотрите все используемые трэйты.

WhereX

Есть элегантный способ превратить это:

```
$users = User::where('approved', 1)->get();
```

В это:

```
$users = User::whereApproved(1)->get();
```

Да, вы можете изменить имя любого поля и добавить его как суффикс в “where”, и оно будет работать как по волшебству.
Также в Eloquent ORM есть предустановленные методы, связанные с датой и временем:

```
User::whereDate('created_at', date('Y-m-d'));
User::whereDay('created_at', date('d'));
User::whereMonth('created_at', date('m'));
User::whereYear('created_at', date('Y'));
```

Сортировка по умолчанию

Что делать, если вы хотите, чтобы `User::all()` всегда сортировался по полю `name`? Вы можете назначить глобальную заготовку (Global Scope). Вернемся к методу `boot()`, о котором мы уже говорили выше.

```
protected static function boot()
{
    parent::boot();

    // Сортировка по полю name в алфавитном порядке
    static::addGlobalScope('order', function (Builder $builder) {
        $builder->orderBy('name', 'asc');
    });
}
```