Сайт функционирует в двух режимах доступа: для посетителей и для администратора. Пользовательская часть позволяет клиентам перемещаться по сайту и добавлять контент. Материалы группируются по категориям. Для работы с системой необходимо зарегистрироваться, после чего пользователь может перейти в свой профиль, где и осуществляется вся работа.

При вводе логина и пароля администратора активируется панель управления, где используя формы создания и редактирования разделов, категорий, содержимого, пунктов меню, можно вносить изменения в структуру и содержание сайта.

Заключение. Созданный веб-портал позволит существенно сэкономить рабочее время, повысить эффективность рабочих коммуникаций, предоставить актуальную информацию о событиях в компании, назначить задачи и контролировать их выполнение, поддерживать и развивать корпоративную культуру.

Список цитируемых источников

1. Евдокимов Н. В., Бабаев А. Б., Боде М. М. Создание сайтов. СПб. : Питер, 2014. 512 с.
2. Норт Б. Joomla! Практическое руководство. СПб.: Символ-Плюс, 2008. 448 с.

УДК 004.657

А. С. Рогозик

Учреждение образования «Барановичский государственный университет», Барановичи

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ELOQUENT ORM ПРИ РАБОТЕ С БАЗАМИ ДАННЫХ В LARAVEL 5 FRAMEWORK

В работе рассматриваются особенности использования системы объектно-реляционного отображения ORM Eloquent в Laravel 5 Framework: создание модели данных, основные методы для работы с данными, функция массового заполнения, механизм псевдоудаления и механизм работы с отношениями.

This paper considers the features of the use of object-relational mapping ORM Eloquent in Laravel 5 Framework: namely the creation of a data model, the basic techniques for working with data, mass filling, soft deleting mechanism and works with the relationship.

Введение. Система объектно-реляционного отображения ORM Eloquent — красивая и простая реализация шаблона проектирования ActiveRecord в Laravel для работы с различными базами данных. Она позволяет строго определить отношения между объектами базы данных. Каждая таблица имеет соответствующий класс модели, который используется для работы с этой таблицей.

Основная часть. Для начала работы с ORM Eloquent необходимо создать модель вручную или с помощью Artisan-команды: php artisan make:model User. Модели Eloquent должны расширять класс Illuminate\Database\Eloquent\Model. Привязка модели к таблице в базе данных происходит автоматически, для этого используется имя класса в нижнем регистре и во множественном числе, т. е. Eloquent предположит, что модель User хранит свои данные в таблице users. Кроме этого имеется возможность использовать произвольную таблицу, определив свойство table в классе модели:

```
class User extends Model {
 protected $table = 'my_users';
} .
```

ORM Eloquent также предполагает, что каждая таблица имеет первичный ключ с именем id, для изменения этого имени необходимо определить свойство protected $primaryKey.

Основные методы модели Eloquent для работы с данными [1]:

1) получение всех моделей (записей):

```
$users = User::all();
```

2) получение модели (записи) по первичному ключу:

```
$user = User::find(1);
```

3) построение запросов:

```
$users = User::where('votes', '>', 100)->get();
```

4) агрегирующие функции:

```
$count = User::where('votes', '>', 100)->count();
```

5) обработка результата по частям:

```
User::chunk(200, function($users) { … });
```

6) группировка данных:

```
User::where('votes', '>', 100)->groupBy('date')->get(); .
```

ORM Eloquent поддерживает функцию массового заполнения [2]. При создании новой модели конструктору передаётся массив атрибутов. Эти атрибуты затем присваиваются модели через механизм массового заполнения. Использование этого механизма очень удобно, но в то же время представляет серьёзную проблему с безопасностью, если ввод от клиента передаётся в модель без проверок — в этом случае пользователь может изменить любое поле модели Eloquent. По этой причине по умолчанию Eloquent защищает модель от массового заполнения. Для этого необходимо определить в классе модели свойство fillable или guarded. Свойство fillable указывает, какие поля должны быть доступны при массовом заполнении, а свойство guarded содержит список запрещённых к заполнению полей:

```
class User extends Model {
 protected $guarded = ['id', 'password'];
 protected $fillable = ['first_name', 'last_name', 'email'];
} .
```

Также есть возможность запретить все атрибуты для заполнения специальным значением protected $guarded = ['*'].

Для создания новой записи в базе данных необходимо создать экземпляр модели и вызвать метод save:

```
$user = new User();
$user->name = 'Jonny';
$user->save(); .
```

Для обновления модели необходимо получить её, изменить атрибут и вызвать также метод save:

```
$user = User::find(1); // get user by id
$user->email = 'jonny@foo.com';
$user->save(); .
```

По умолчанию Eloquent автоматически поддерживает поля created_at и updated_at, записывая в них, соответственно, дату и время (timestamp) создания и обновления строки в базе данных. Если необходимо, чтобы модель не поддерживала этот механизм, добавьте свойство timestamps, равное false, к классу модели:

```
class User extends Model {
 public $timestamps = false;
} .
```

Для удаления модели необходимо вызвать метод delete:

```
$user = User::find(1); // get user by id
$user->delete(); .
```

ORM Eloquent также поддерживает механизм псевдоудаления (Soft Deleting) [3]. Чтобы модель использовала псевдоудаление, необходимо в классе применять конструкцию use SoftDeletes; . При использовании этого механизма записи из базы данных физически не удаляются, а только помечаются как удалённые. Для этого в таблицах употребляется поле deleted_at. При вызове метода delete, поле deleted_at будет установлено в значение текущего времени. При запросе моделей, использующих псевдоудаление, «удалённые» модели не будут включены в результат запроса.

Для восстановления псевдоудалённой модели в активное состояние применяется метод restore:

```
$user->restore(); .
```

При использовании механизма Soft Deletes возникают ситуации, когда запись из базы данных должна быть удалена физически. Для этого необходим метод forceDelete:

```
$user->forceDelete(); .
```

ORM Eloquent имеет встроенный механизм работы и управления отношениями [4]. Laravel поддерживает несколько типов отношений: «один к одному», «один ко многим», «многие ко многим», «связь через третью таблицу» (Has Many Through), «полиморфические связи один к одному», «полиморфические связи многие ко многим».

Связь вида «один к одному» является самой простой, например, модель User может иметь один Phone. Определить такое отношение в Eloquent можно следующим образом:

```
class User extends Model {
 public function phone() {
 return $this->hasOne('App\Phone');
 }
} .
```

Первый параметр, передаваемый hasOne — имя связанной модели. Доступ к связанной модели можно получить через динамические свойства Eloquent:

```
$phone = User::find(1)->phone; .
```

ORM Eloquent считает, что внешнее поле (foreign_key) в связанной таблице называется по имени модели плюс _id, в примере предполагается, что это user_id. Для перекрытия стандартное имя foreign_key необходимо передать в качестве второго параметра методу hasOne:

```
return $this->hasOne('App\Phone', 'foreign_key'); .
```

Если же в модели, для которой строится отношение (в примере User), ключ находится не в столбце id, то необходимо указать его в качестве третьего аргумента:

```
return $this->hasOne('App\Phone', 'foreign_key', 'local_key'); .
```

Для создания обратного отношения в модели Phone следует использовать метод belongsTo («принадлежит к»):

```
class Phone extends Model {
 public function user() {
 return $this->belongsTo('App\User');
 }
} .
```

Заключение. Использование ORM Eloquent в Laravel 5 позволяет разработчику значительно абстрагироваться от написания sql-запросов к базе данных, вести удобную и быструю разработку как модели данных, так и бизнес-логики, использовать преимущества объектно-ориентированного программирования на всех этапах разработки, значительно повышать коэффициент повторного использования кода и создавать лаконичный, понятный код бизнес-логики.

Список цитируемых источников

1. Laravel — The PHP Framework For Web Artisans. URL: http://laravel.com/docs/5.1 (date of access: 25.09.2015).
2. Ibid.
3. The Best Laravel and PHP Screencasts / Laravel 5 Fundamentals. URL: https://laracasts.com/series/laravel-5-fundamentals (date of access: 25.09.2015).
4. Laravel — The PHP Framework For Web Artisans.

УДК 004.75

Д. О. Руднев
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тульский государственный университет», Тула, Российская Федерация

А. А. Сычугов,
кандидат технических наук, доцент
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тульский государственный университет», Тула, Российская Федерация

МЕТОД ПОВЫШЕНИЯ БЕЗОПАСНОСТИ РАБОТЫ АЛГОРИТМОВ ПОИСКА АНОМАЛИЙ В РАСПРЕДЕЛЁННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

В данной работе описан метод безопасного сбора информации об элементах распределённой информационной системы в целях дальнейшего поиска аномалий работы системы. Основой предлагаемого метода является использование беспризнакового распознавания образов. В статье подробно описаны сильные и слабые стороны предлагаемого метода.

This paper describes a method for the safe collection of information about the elements of a distributed information system c to further search of anomalies of the system. The proposed method is to use featureless pattern recognition. The article described in detail the strengths and weaknesses of the proposed method.