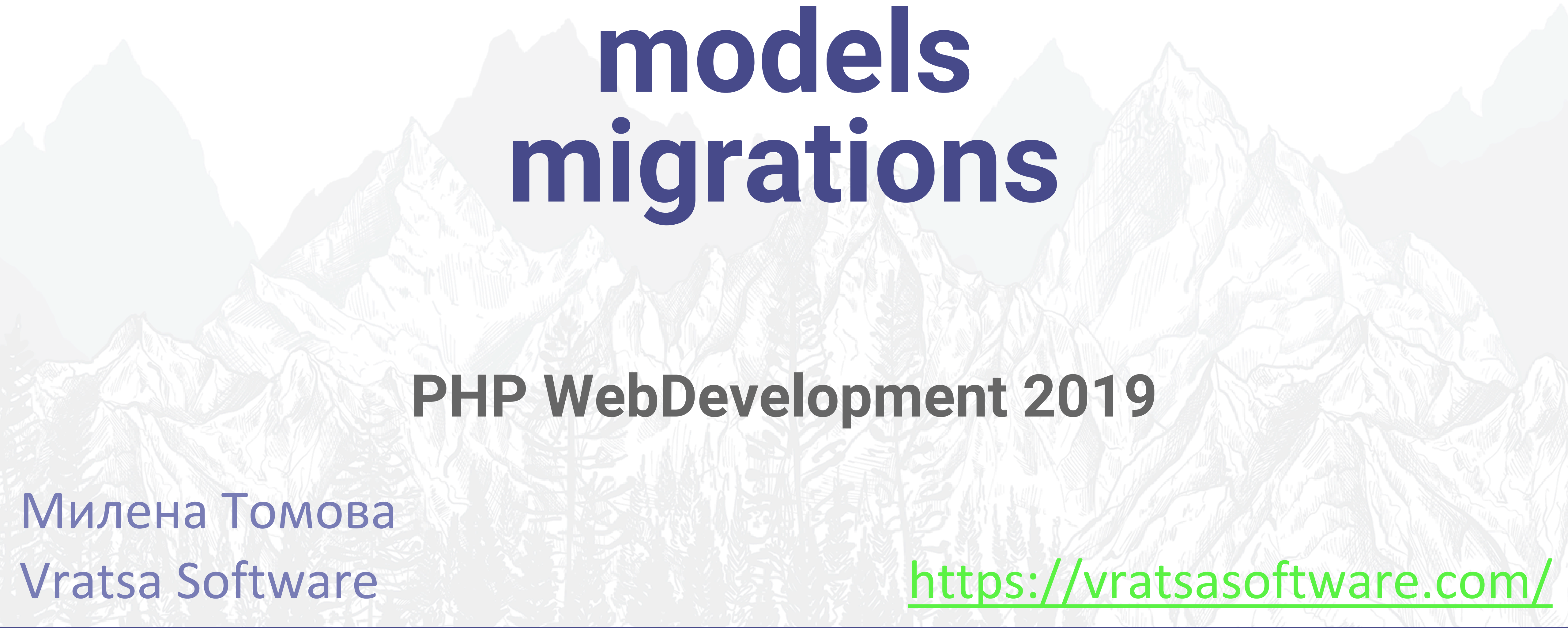


Laravel models migrations

PHP WebDevelopment 2019

Милена Томова
Vratsa Software

https://vratsasoftware.com/

Table of Contents

Task 1

Print all resources -

- Courses
- Levels
- Lectures
- Homeworks

in the resource`s list page.

Use navigation to access related resources.

Model
V
C

Model

All the methods in the Model are designed to process the project data.

- Models works with the database
- Models transform the data in the needed format and quantity
- Models pass the data to the controller on a request -
    - from controller to the database
    - from database to the controller
    - return the transformed data without reaching the database

Model

Every class from the project`s OOP design will have representation in three places in Laravel.

Controllers will hold the functionality of the class.

Models are responsible for the properties of the class and the data the class works with - to retrieve or send it to the database.

Views will display the entry points for the objects of that class to act in the application.

Model

```
php artisan make:model Course
```

This will create a Course model class in root application directory.

The **model name** by convention is always **singular**.

Model

```
protected $fillable = [‘property-to-be-added-to-database’];
```

$fillable is an array holding the class properties that are allowed to be adde, updated or deleted in the database.

Laravel - connect to database

Connect to database

Database configuration file - **config/database.php**

Database credentials are filled in **- .env** file

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=database-name
DB_USERNAME=database-username
DB_PASSWORD=database-user-password
```

Laravel - migrations

Migrations

Migrations are like **version control for your database**, allowing your team to **modify and share** the application's database schema.

Migrations are typically paired with Laravel's **schema builder** to build your application's database schema.

If you use git and migrations and have made changes to the database - added new table or column, your teammates will easily detect and update the project`s database on their local machines.

The Laravel **Schema** facade provides database agnostic support for creating and manipulating tables across all of Laravel's supported database systems.

Migrations

```
php artisan make:migration create_courses_table --create=courses
```

The table name by convention is always **plural.**

To indicate that a new table is created use **--create=table-name** option

Migrations

```
php artisan make:migration add_name_column_to_courses_table --table=courses
```

The table name by convention is always **plural.**

To indicate that **a change is made** to an existing table use **--table=table-name** option

Migrations

Both commands generate migrations file in **database/migrations/** folder.

The filename holds the timestamp and the command string.

```
php artisan migrate
```

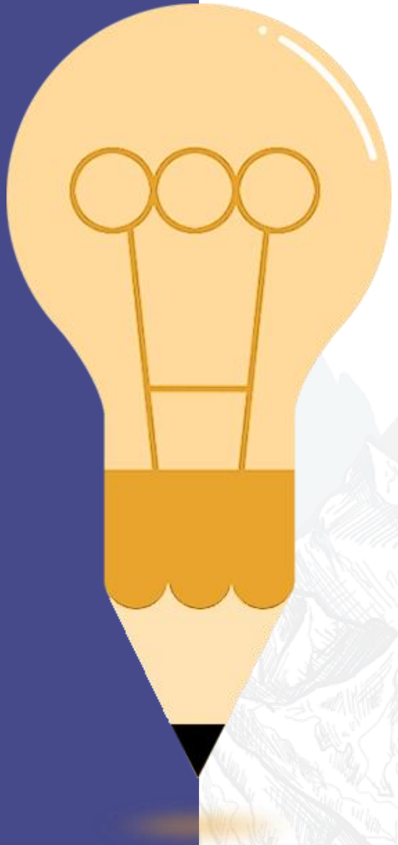
Converts the migration file in a database table or modifies an existing database table.

Before creating the project database

After Laravel 5.4 we must take in consideration the following

Indexes length and MySQL / MariaDB

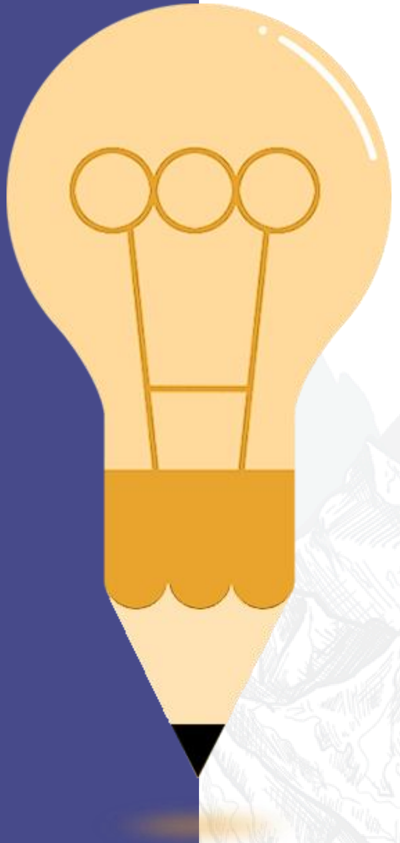
By default Laravel uses utf8mb4, which allows saving "emojis" in the database.
If your MySQL, is older than 5.7.7 or MariaDB is older than 10.2.2, you have to reset the default string length.



Before creating the project database

Врaца Софтуер Общество

Go to app/Providers/AppServiceProvider.php and add to **boot** method -



```
use Illuminate\Support\Facades\Schema;

/**
 * Bootstrap any application services.
 *
 * @return void
 */
public function boot()
{
    Schema::defaultStringLength(191);
}
```

Display records from DB for a Model

```
...
use App\Course;

class CoursesController extends Controller {

        public function index()
    {
        $courses = Course::all();

        return view(‘courses.index’, compact(‘courses’))
    }
}
```

Display records from DB for a Model

Врaца Софтуер Общество

```
routes/web.php
Route::get(‘courses’, ‘CoursesController@index’)->name(‘courses.list’);
```

```
resourses/views/courses/index.blade.php

@if( $courses )
        @foreach( $courses as $course )
                <h2> {{ $course->name }} </h2>
        @endforeach
@endif
```

Task 2

1. Create the project database.
2. Create courses table.
3. Migrate the database.
4. Use existing column types for reference.
5. Display the list of courses.

***

6. Repeat the above steps for halls /have name, capacity/, casting devices /have name/.

Laravel Eloquent

Laravel Eloquent

The Eloquent ORM included with Laravel provides a beautiful, simple ActiveRecord implementation for working with your database.

Each database table has a corresponding "Model" which is used to interact with that table.

Models allow you to query for data in your tables, as well as insert new records into the table.

Laravel Eloquent

```
php artisan make:model Level --migration


php artisan make:model Level -m
```

Creates a model and a migration file for it.

Task 3

1. Create Profile model and a migration file. Build profiles table.
2. Create Level model and a migration file. Build the levels table.
3. Migrate the tables in the database.
4. Define relations **profiles-users** and **courses-levels**. Set the relations in models using Laravel Eloquent.
5. *Display resources -
   - ❑ courses
   - ❑ levels
   - ❑ users

6.*Make every element in a list **a link** for the related resource list /courses, users/

one-to-one relation

Laravel Eloquent

A User model might be associated with one Profile
and /if needed/
a Profile model might be associated with one User /the inverse of the relation/.

Relationships are defined by placing a function in the Eloquent models.

one-to-one relation

Врaца Софтуер Общество

Laravel Eloquent

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

```
class Profile extends Model
{
    public function profile()
    {
        return $this->belongsTo('App\User');


    }
}
```

one-to-one relation

Laravel Eloquent

in controller

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

Use defined relations to retrieved the related data

or in a view

```
@foreach( $users as $user )
          <h2> {{ $user->name }}</h2>
     <p> {{ $user->profile()->bio }}</p>
@endforeach
```

one-to-many relation

Laravel Eloquent

A **one-to-many relationship** is used to define relationships where **a single model** owns **any amount** of **other models**.

For example, a **course** may have an infinite **number of levels**.

one-to-many relation

Laravel Eloquent

```
class Course extends Model
{
    public function levels()
    {
        return $this->hasMany('App\Level');
    }
}
```

A course has many levels

```
class Level extends Model
{
    public function course()
    {
        return $this->belongsTo('App\Course');
    }
}
```

A level belongs to a course

one-to-many relation

Laravel Eloquent

<u>in the controller - get all courses</u>

```
$courses = Course::all();
```

<u>in the view</u>

```
@foreach ($courses as $course)

    @foreach ($course->levels as $level)
                ///
    @endforeach

@endforeach
```

<u>in the controller -</u>
<u>levels for a course</u>

```
$levels = App\Course::find(1)->levels;
```

<u>in the view</u>

```
@foreach ($levels as $level)

@endforeach
```

one-to-many relation

Laravel Eloquent

the inverse relation

in the controller - get all levels

```
$levels = Level::all();
```

in the view

```
@foreach ($levels as $level)
        {{ $level->course->name }}
@endforeach
```

Task 4

1. Create models and migration files for the rest of the project resources described in the project database design.
2. Build resources database table.
3. Migrate tables to the database.
4. Define relations between models using Laravel Eloquent.
5. *Display resources as a list per resource with all the related data available

many-to-many relation

Врaца Софтуер Общество

Laravel Eloquent

To define **many-to-many** relationship, three database tables are needed: users, courses, and course_user.

```
users
        id - integer
        ..
courses
        id - integer
        ..
course_user
     user_id - integer
     course_id - integer
```

many-to-many relation

Laravel Eloquent

```
class User extends Model {

    public function courses()
    {
        return $this->belongsToMany('App\Course');
    }
}
```

A User belongs to many Courses

many-to-many relation

Laravel Eloquent

Access the courses the user belongs to

```
$user = App\User::find(1);
foreach ($user->courses as $course) {
    //
}
```

```
$courses = App\User::find(1)
                ->courses()
                ->orderBy('course_name')
                ->get();
```

many-to-many relation

Врaца Софтуер Общество

Laravel Eloquent

The inverse relation

```
class Course extends Model {

    public function userss()
    {
        return $this->belongsToMany('App\User');
    }
}
```

A Course belongs to many Users

many-to-many relation

Врақа Софтуер Общество

Laravel Eloquent

Access the users that belong to a course

```
$course = App\Course::find(1);
foreach ($course->users as $users) {
    //
}
```

```
$users = App\Course::find(1)
                ->users()
                ->orderBy('name')
                ->get();
```

Task 5

1. Connect each user with the course he belongs to -

    make every **user name** /if a student!/ **a link** that leads to a page with **a list of all of the user`s courses**

2. Connect each course from the courses list with the students that belong to this course

    make every **course name a link** that leads to a page with

    a **list of all of the course`s students**

Task 6

1. Create Events resource - each event has name, description, date, place, participants
2. Connect each event with with users that participate in it -

   make every **event name a link** that leads to a page with **a list of all of the participants /users/**

3. Display in every user`s profile a list of the events /if any/ he participates in.

all() vs. get()

use all() when accessing the resource directly

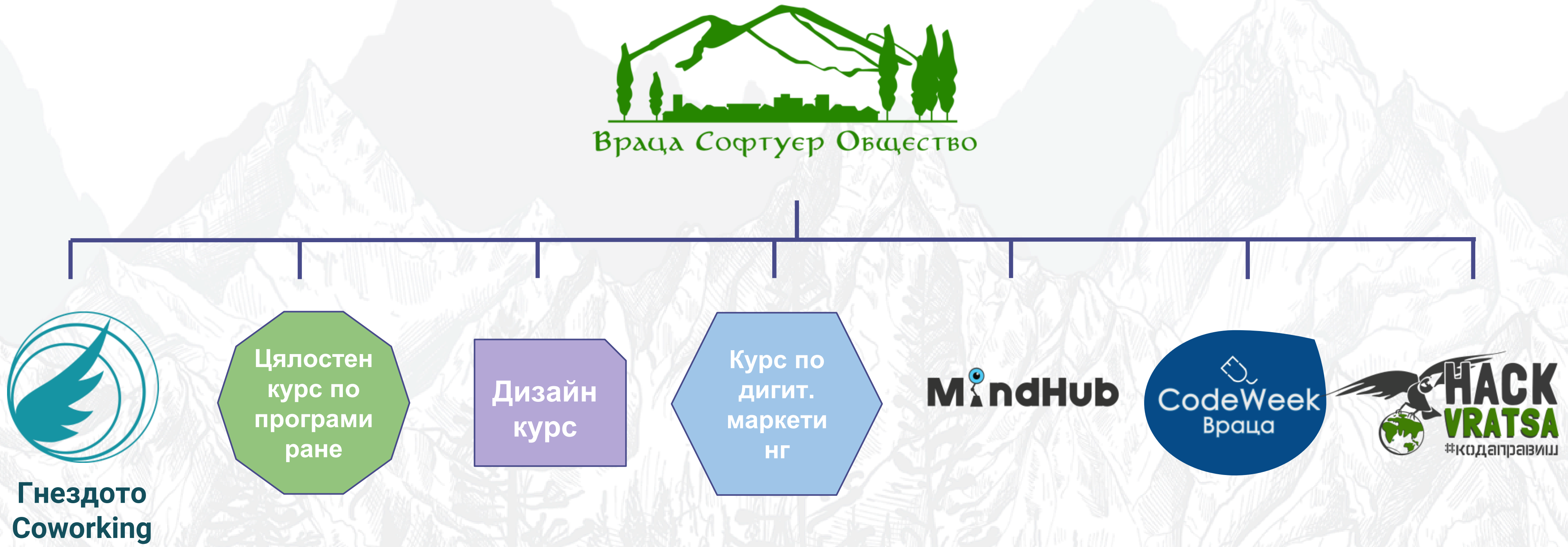
```
$users = User::all();
```

use get() when accessing the resource indirectly

```
$users = App\Course::find(1)
                        ->users()
                ->get();
```

```
$users = App\User::with('profile')
            ->get();
```

Questions?


Враца Софтуер Общество
Гнездото Coworking
Цялостен курс по програми ране
Дизайн курс
Курс по дигит. маркети нг
MindHub
CodeWeek Враца
HACK VRATSA
#кодаправиш

Partners

Враца Софтуер Общество

AMERICA FOR BULGARIA FOUNDATION

ВРАЦА

Telerik Academy

НАЦИОНАЛНА МРЕЖА ЗА ДЕЦАТА

MindHub

ПРОМЯНАТА NOVA

Trainings @ Vratsa Software

- Vratsa Software – High-Quality Education, Profession and Jobs
  - www.vratsasoftware.com
- The Nest Coworking
  - www.nest.bg
- Vratsa Software @ Facebook
  - www.fb.com/VratsaSoftware
- Slack Channel
  - www.vso.slack.com

SoftUni Foundation