# Laravel Lab

## PHP WebDevelopment 2021

Milena Tomova
Borislav Krustev
Vratsa Software

https://vratsasoftware.com/

# Table of Contents

# Learn to Search in Internet

- The course assignments require to search in Internet
  - This is an important part of the learning process
  - Some exercises intentionally have no hints

- Learn to find solutions!
  - Software development includes everyday searching and learning
  - No excuses, just learn to study!
  - Developers learn new technologies, tools, languages every day!

# Laravel Collective

# Laravel Collective - HTML

Laravel Collective

documentation

HTML and form builder

# Laravel Collective - HTML

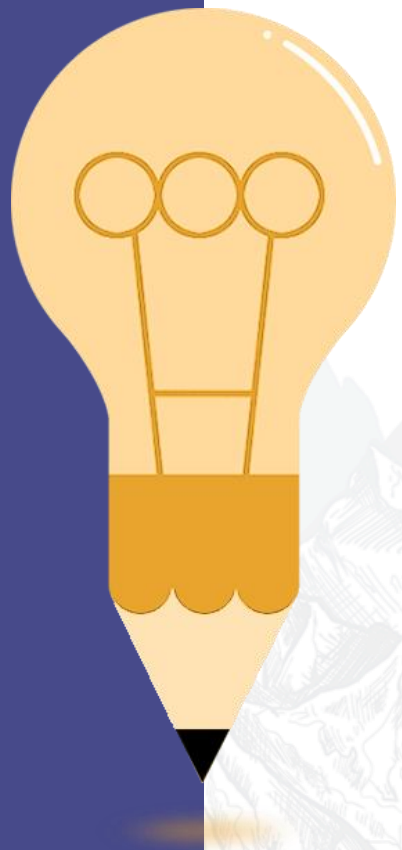
Врача Софтуер Общество

- install by

```
composer require laravelcollective/html
```

*and start using it!*

# Important



Since this package internally uses

**strict comparisons** (=== instead of ==)

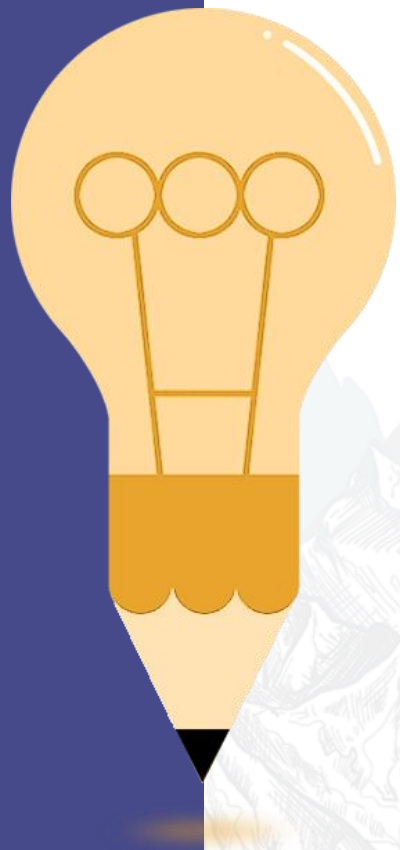
be careful when passing numeric values into your forms.

Values in HTML are submitted as strings and Laravel old values stored in flash session are strings.

# Important {{ }} vs. {!! !!}

Врача Софтуер Общество

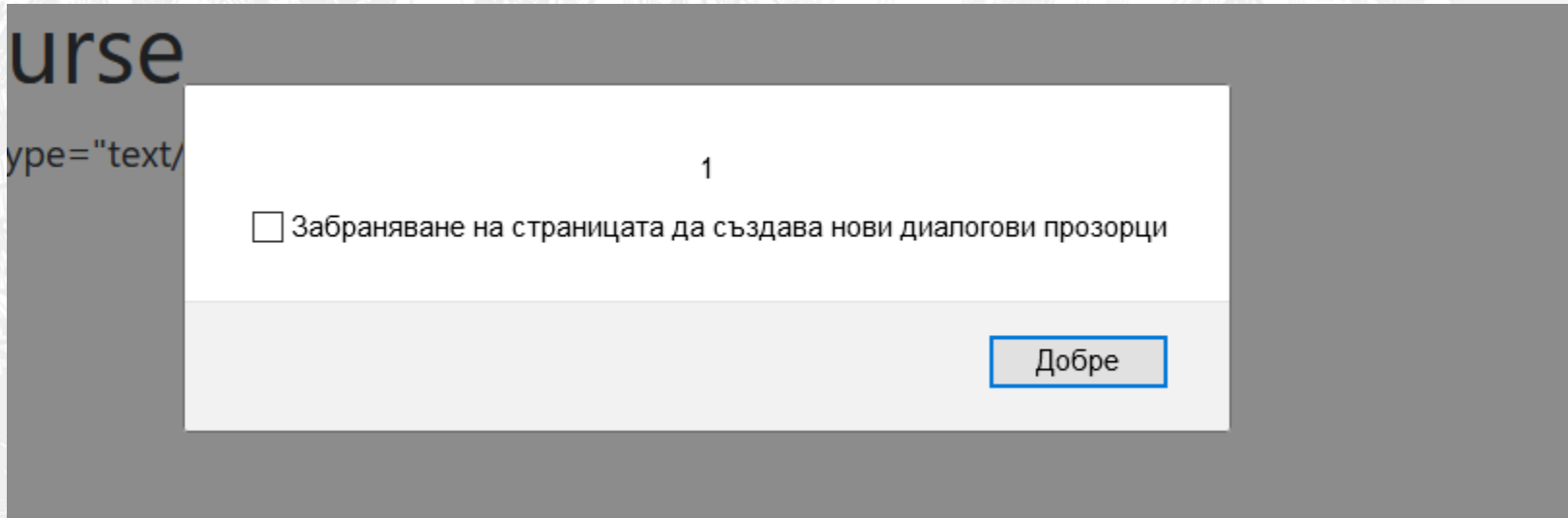
```
@php
    $p = '<script type="text/javascript">alert(1)</script>';
@endphp
```



```
{{ $p }}
```

//<script type="text/javascript">alert(1)</script>

```
{!! $p !!}
```



# Task 1

Transform the forms that

- adds level to course
- creates new users
- updates users
- *** create CustomValidation  for updating user data

# Summary



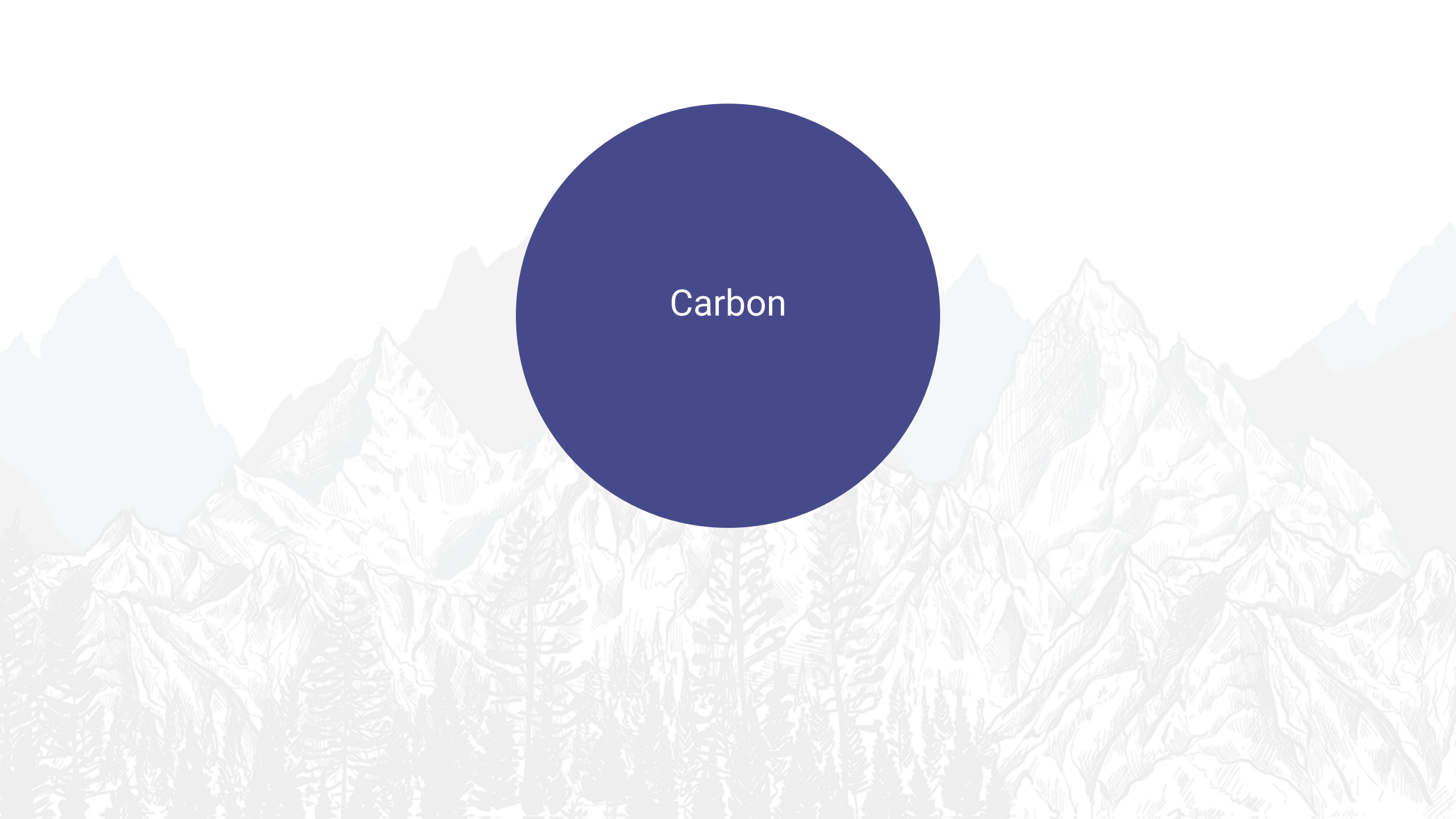
- Since HTML forms only support POST and GET, PUT and DELETE methods will be **spoofed by automatically** adding an element with a method key in the form opening line;
- the model's value matching the field's name will automatically be set as the field value.

    So, for example, for a text input **named email**, the user model's email attribute would be set as the value.
    If there is an item in the Session flash data matching the **input name**, that will take precedence over the model's value.

So, the priority looks like this:
- Session Flash Data (Old Input)
- Explicitly Passed Value
- Model Attribute Data

This allows you to quickly build forms that not only bind to model values,

but easily re-populate if there is a validation error on the server!


Carbon

# Task 2

On every resourse page display columns with

- date and time when the record has been created
- date and time when the record has been updated

in user friendly formats.

For example

- 2 minutes ago
- dd/mm/yy hh:mm:ss

# Carbon

Carbon

Carbon is a powerful extension for working with dates -

- formatting
- comparing
- converting
- finding difference between dates
- telling the day of the week of a date
- etc.

Carbon is an international PHP extension for DateTime. http://carbon.nesbot.com

documentation for using Carbon with  Laravel

# Carbon

- Just check your composer.json file for nesbot/carbon package.
  - In case if you don't have you can install it by typing

```
composer require nesbot/carbon
```

*and start using it!*

# Carbon

- in case for some reason you wanted to use the diffForHumans() function for a column that is neither created_at and updated_at

```
...
use Carbon\Carbon;
use ...
class MyController extends Controller{
        …
  }
```

# Carbon



```
...
use Carbon\Carbon;
use ...
class MyController extends Controller{
        $expire_date_string = '2016-07-27 12:45:32';
       //  Parse date with carbon - convert it to carbon object
       $carbonated_date = Carbon::parse($expire_date_string);
       //  Assuming today was 2016-07-27 12:45:32
       $diff_date = $carbonated_date->diffForHumans(Carbon::now());
       echo $diff_date; // prints "1 month after"
  }
```

# Carbon

Врача Софтуер Общество

- You can add the new column like expired_at to your model:

```
protected $dates = [
    'created_at',
    'updated_at',
    'deleted_at',
    'expired_at'
];
```

And then you can use ->diffForHumans();

# Carbon

- Carbon string formatting

```
echo $user->updated_at;
       //2020-10-24 15:50:49
echo $user->updated_at->toDayDateTimeString();
       //Sat, Oct 24, 2020 3:50 PM
echo $user->updated_at->toDateTimeString();
       //Oct 24, 2020 3:50 PM
```

# Carbon

- Carbon comparison

```
$first = Carbon::create(2012, 9, 5, 23, 26, 11);
$second = Carbon::create(2012, 9, 5, 20, 26, 11, 'America/Vancouver');

var_dump($first->equalTo($second));                    // bool(false)
var_dump($first->lessThanOrEqualTo($second));          // bool(true)
```

# Carbon

Врача Софтуер Общество

- Carbon Addition and Subtraction

```
$dt = Carbon::create(2012, 1, 31, 0);
echo $dt->toDateTimeString();           // 2012-01-31 00:00:00

echo $dt->addYears(5);                  // 2017-01-31 00:00:00
echo $dt->addYear();                    // 2018-01-31 00:00:00
echo $dt->subYear();                    // 2017-01-31 00:00:00
echo $dt->subYears(5);                  // 2012-01-31 00:00:00
```

# Carbon

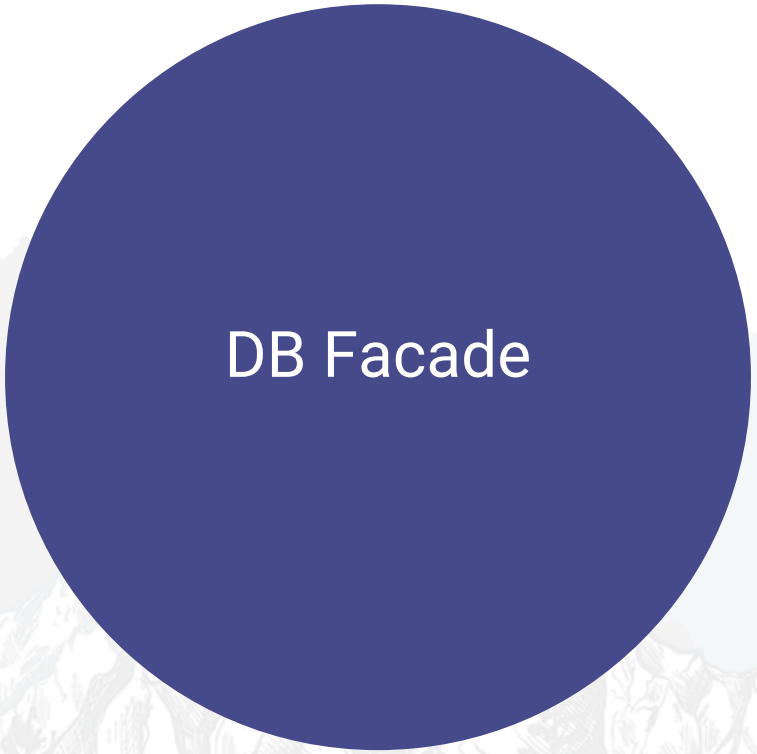
- Carbon localization

```
Carbon::setLocale('bg_Bg');
```

# Carbon

Врача Софтуер Общество

- Carbon Create Differences


DB Facade

# DB Facade

DB Facade

Largely used in **Laravel Database: Query Builder**

documentation

# DB Facade

DB Facade

Laravel's database **query builder** provides a convenient, fluent interface to creating and running **database queries**. It can be used to perform most database operations in your application and works on all supported database systems.

The Laravel query builder uses **PDO parameter binding to protect your application against SQL injection attacks**. There is no need to clean strings being passed as bindings.

# DB Facade

Враца Софтуер Общество

- examples querying database using Laravel Query builder
  - selects

```
$users = DB::table('users')->select('name', 'email as user_email')->get();
```

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

```
$users = DB::table('users')->where('name', 'like', '%test%')->get();
```

# DB Facade

Враца Софтуер Общество

- examples querying database using Laravel Query builder
  - aggregates

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

```
$price = DB::table('orders')->max('price');
```

```
$price = DB::table('orders')->where('finalized', 1)->avg('price');
```

# DB Facade

Враца Софтуер Общество

- building queries dynamically
  - If you already have a query builder instance and you wish to add a column to its existing select clause, use the addSelect method:

```
$query = DB::table('users')->select('name');
```

- on condition change the query:

```
$users = $query->addSelect('age')->get();
```

# DB Facade

Eloquent vs. Query Builder

## Eloquent or Query Builder

- When we'll work on a simple and small records site with simple CRUD and there records are not fact, then use Eloquent there.

- When we'll work on a lot's of records, it is better to use DB Query than Eloquent.

# DB Facade

Eloquent vs. Query Builder

- I'm making an ***University website***. Which may contain maximum 5,000 teachers and 10,000 students and some notices and files. Then it is better to do this with simple Laravel Eloquent which is very much standard and readable.

- Now I'm making a site like ***Stackoverflow***. Which may contains more than 1,000,0000 (1 crore) posts and many more things. I will must choose the conventional DB facades there. It is more faster for searching the posts from so much records.

# DB Facade

Eloquent vs. Query Builder

To insert 1000 rows for a simple table Eloquent takes 1.2 seconds and in that case DB facades take only 800 miliseconds(ms).

- check your app performance with

  Laravel Debugbar

# DB Facade

Eloquent vs.
Query Builder

## Eloquent ORM average response time

```
Joins | Average (ms)
------+-------------
1     | 162,2
3     | 1002,7
4     | 1540,0

Result of select operation average response time for Eloquent ORM
```

## Raw SQL average response time

```
Joins | Average (ms)
------+-------------
1     | 116,4
3     | 130,6
4     | 155,2

Result of select operation average response time for Raw SQL
```

# Raw Queries

# Raw Queries

Raw Queries

Business logic is often complicated.

Or you`ve faced a project with model relations not defined.

Because of this, we often need to write our own SQL queries.

Luckily, Laravel's query builder has the tools we need to safely run such queries.

# Raw Queries

Raw Queries

A **key concern** when writing our own queries is protecting our application from **SQL injection** attacks.

Normally, the **query builder does this for us.**

However, when we write our own SQL, we need to **make sure we don't inadvertently remove this protection.**

# DB Facade

Врача Софтуер Общество

- here is what we want to avoid:

```
$someVariable = Input::get("some_variable");


$results = DB::select( DB::raw("SELECT * FROM some_table WHERE some_col =
'$someVariable'") );
```

- **DB::raw()** is used to make arbitrary SQL commands which aren't parsed any further by the query builder.
- They therefore can create a vector for attack via SQL injection.

# DB Facade

- Change the query in a way that sanitizes the user input:

```
$someVariable = Input::get("some_variable");


$results = DB::select( DB::raw("SELECT * FROM some_table WHERE some_col = :somevariable"),
                                                    ['somevariable' => $someVariable]));
```

# DB Facade

- Use the query in a way that sanitizes the user input:
  - example:

```
$results = DB::select('select * from users where id = ?', [1]);
```

# DB Facade

Врача Софтуер Общество

- Use the query in a way that sanitizes the user input:
  - result is an array of obj or update and delete statements return the number of rows affected by the operation

```
$results = DB::select('select * from users where id = ?', [1]);

DB::insert('insert into users (id, name) values (?, ?)', [1, 'Dayle']);

DB::update('update users set votes = 100 where name = ?', ['John']);

DB::delete('delete from users');
```

# 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

# Trainings @ Vratsa Software

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