Zubair Mohsin
Posted on Feb 20, 2020 • Updated on Jan 27, 2021

💖 25 🦄 6

Understanding Mass Assignment in Laravel Eloquent ORM

#php #laravel #webdev #eloquent

Laravel Eloquent ORM provides a simple API to work with database. It is an implementation of Active Record pattern. Each database table is mapped to a **Model class** which is used for interacting with that table.

Mass Assignment

Let's break down this word.

- **Mass** means *large number*
- **Assignment** means the *assignment* operator in programming

In order to make developers life easier, Eloquent ORM offers Mass Assignment functionality which helps them **assign**(insert) **large number** of input to database.

Life, without Mass Assignment 😩

Let's consider that there is a form with 10 fields of user information.

```
<form method="POST" action="/signup">
    <input type="text" name="name" />
    <input type="text" name="user_name" />
    <input type="text" name="password" />
    <input type="text" name="address" />
    <input type="text" name="city" />
    ...
    ...
    ...
    <button type="submit">Signup</button>
</form>
```

and our Controller's store method looks like:

```
public function store(Request $request)
{
    //perform validation

    $user = new User;

    $user->name = $request->get('name');
    $user->user_name = $request->get('user_name');
    $user->password = bcrypt($request->get('password'));
    $user->address = $request->get('address');
    $user->city = $request->get('city');

    //....

    $user->save();
}
```

We are assigning each input manually to our model and then saving to database.

> What if there was a way to insert all the coming input without the manual assignment?

Life, with Mass Assignment 🥳

Laravel Eloquent ORM provides a **create** method which helps you save all the input with single line.

```
public function store(UserFormRequest $request)
{
    //validation performed in the UserFormRequest

    $user = User::create($request->validated());
}
```

How cool is that, right? 🤓 With single line we are saving all input to database. In future, if we add more input fields in our HTML form, we don't need to worry about our saving to database part.

> Input fields **name** attribute must match the column name in our database.

Potential Vulnerability

For the sake of understanding, let's consider that we have an **is_admin** column on **users** table with *true / false* value.

A malicious user can inject their own HTML, a hidden input as below:

```
<form method="POST" action="/signup">

    <input type="hidden" name="is_admin" value="1" />

    <input type="text" name="name" />
    <input type="text" name="user_name" />
    <input type="text" name="password" />
    <input type="text" name="address" />
    <input type="text" name="city" />
    ...
    ...
    ...
    <button type="submit">Signup</button>
</form>
```

With Mass Assignment, **is_admin** will be assigned **true** and the user will have Admin rights on website, which we don't want 😡

Avoiding the vulnerability

There are two ways to handle this.

- We can specify (whitelist) which columns **can be** mass assigned.

Laravel Eloquent provides an easy way to achieve this. In your model class, add `$fillable` property and specify names of columns in the array like below:

```
class User extends Model
{
    protected $fillable = [
        'name',
        'user_name',
        'password',
        'address',
        'city'
        ...
    ];
}
```

Any input field other than these passed to **create()** method will throw **MassAssignmentException**.

- We can specify (blacklist) which columns **cannot be** mass assigned.

You can achieve this by adding `$guarded` property in model class:

```
class User extends Model
{
    protected $guarded = ['is_admin'];
}
```

All columns other than *is_admin* will now be mass assignable.

> You can either choose `$fillable` or `$guarded` but not both.

So there you have it. Happy Mass Assigning with Laravel Eloquent 🥳 👨🏽‍💻

Top comments (7)

Mazen Touati • Feb 22 '20 • Edited

Great breakdown I would like to add these extra tips:

- If all of your table fields are fillable you should consider using `protected $guarded = []`
- You can use `$request->only(array $fields)` to pick only certain columns.
- Assuming you're using Laravel form request (learn more here) you can simply use `$request->validated()` to get the validated data.

PS: for tip 2 & 3 you probably should opt for `protected $guarded = []`

Zubair Mohsin • Feb 22 '20

Hey thank you so much for sharing these tips. There are multiple ways to achieve the same objective. To get better understanding, I kept the options minimal.

Neo • Feb 24 '21

Great job , great explanation.

Victor Allen • Jan 25 '21 • Edited

Very well put...you got me sorted! Thanks for sharing 👍

Roelof Jan Elsinga • Feb 20 '20

I've used Laravel professionally for 5 years and even though I assumed this is how it worked, I learned something new here! Great post, thank you for this!

Zubair Mohsin • Feb 20 '20

Awesome :) Glad you liked it.

ypaatil • Oct 1 '21

Great job. You are cleared my concept.

Code of Conduct • Report abuse

Zubair Mohsin

Full Stack Laravel Developer

LOCATION
Lahore, Pakistan

EDUCATION
BS(Computer Sciences)

WORK
Full Stack Developer at Addition

JOINED
Mar 4, 2019

More from Zubair Mohsin

Why Laravel requires `ramsey/uuid` package?
#laravel #php

Why Laravel requires `psr/container` package?
#laravel #php

How closures are serialized in Laravel using `opis/closure` package

#laravel #php