CLOSE ENCOUNTERS with PHP

Level 5

Validation With Respect

Object-oriented Validation

A Closer Look at the Validator Class

validation.php

```php
<?php
use Respect\Validation\Validator;

$v = new Validator;
var_dump($v);

function validate_date($date_string)
{
   if ($time = strtotime($date_string)) {
      return date('F jS Y',$date_string);
   } else {
      return $date_string . ' does not look valid.';
   }
}
```

A Closer Look at the Validator Class

validation.php

new is how we make an object

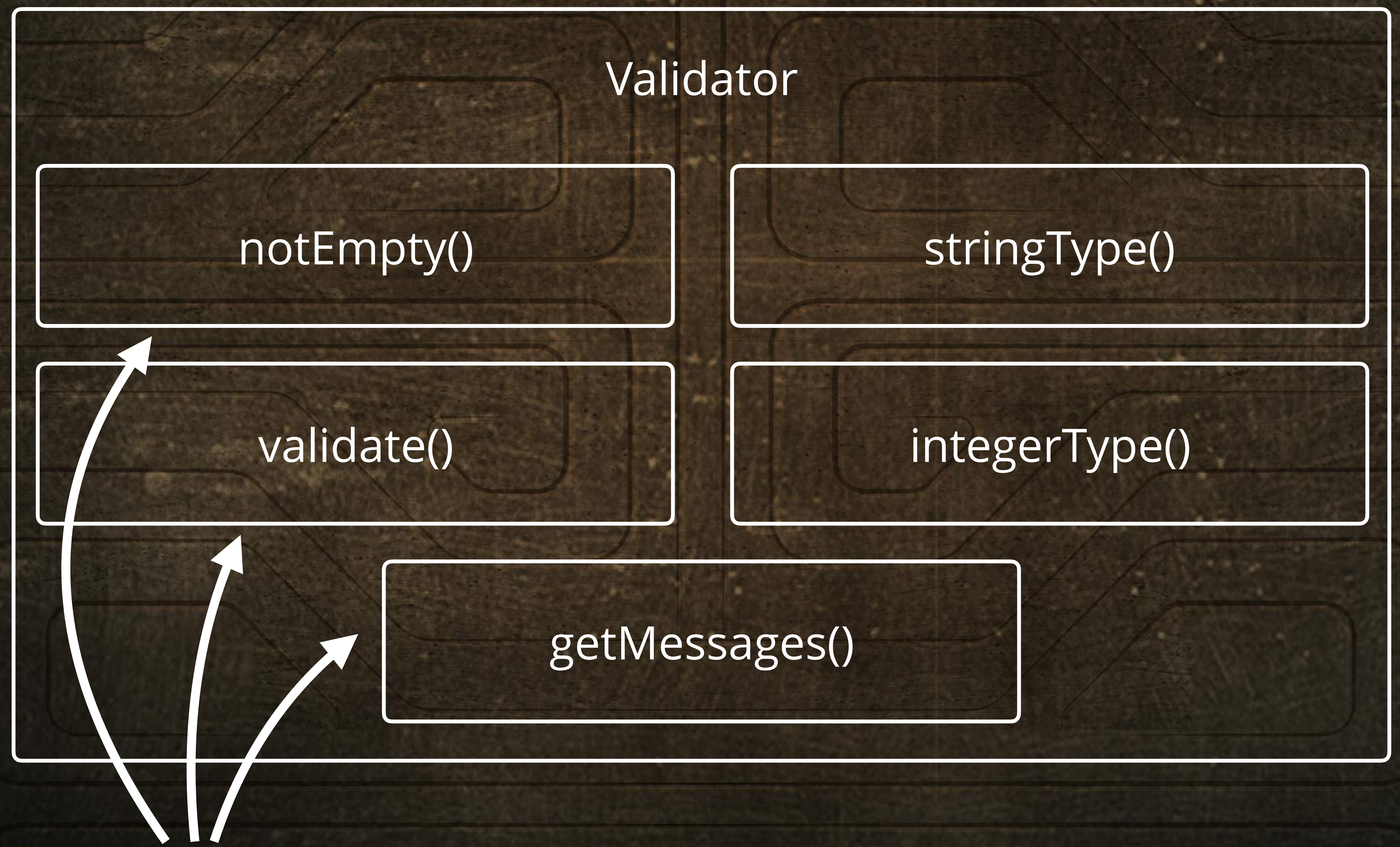
$v is now an object of the Validator type

Validator is the name of a class

```php
$v = new Validator;
var_dump($v);
```

What Is a Class?

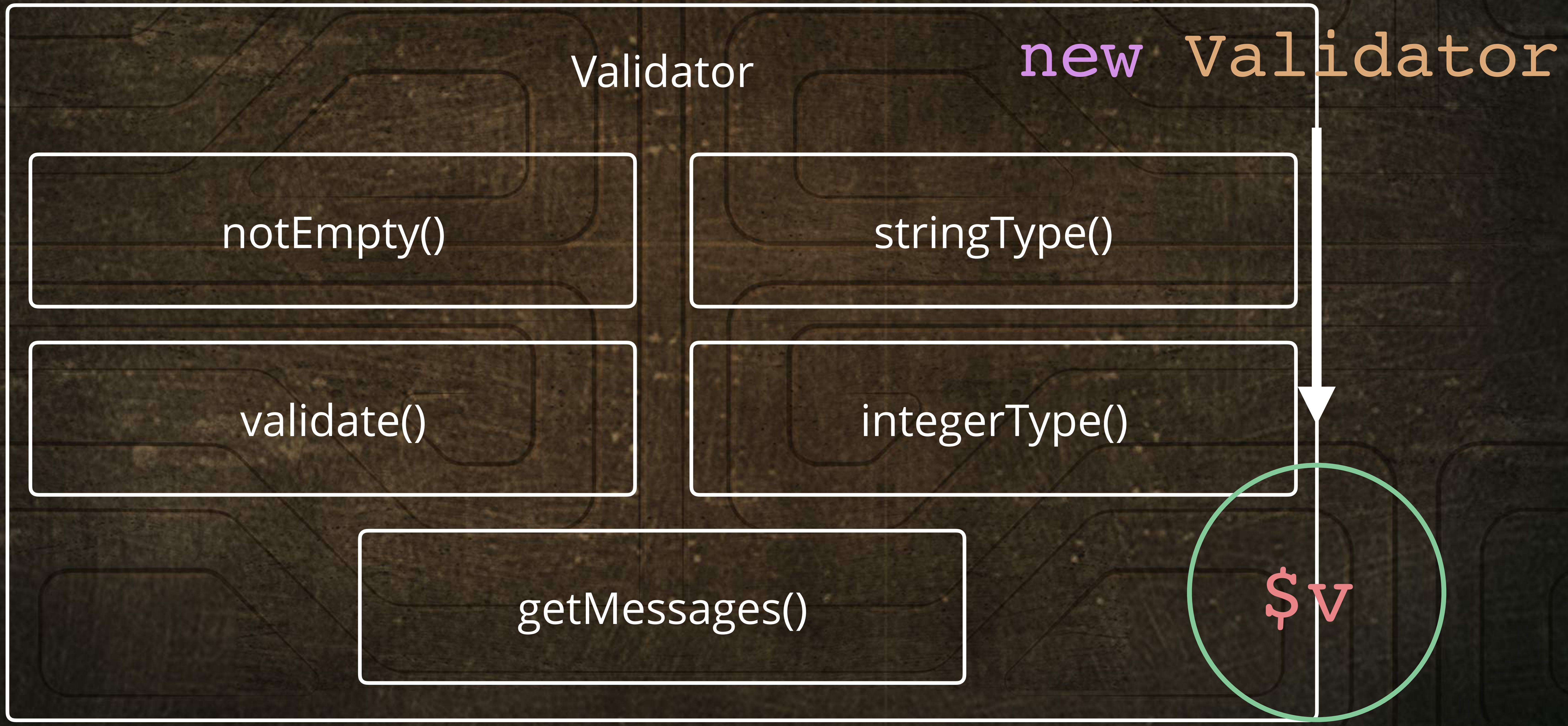
A class is a way to group our code and provide a blueprint for a single-purpose object.



Methods are functions inside of a class

Creating an Object

With the new **keyword, we create an object that is a single instance of the** Validator **class.**



How Can We Use Validator Here?

validation.php

```
<?php
use Respect\Validation\Validator;

$v = new Validator;
var_dump($v);

function validate_date($date_string)
{
    if ($time = strtotime($date_string)) {
        return date('F jS Y',$date_string);
    } else {
        return $date_string . ' does not look valid.';
    }
}
```

We need to replace all of the code inside the `validate_date` *function with something from the* `Validator` *class*

Using Respect/Validation for the Date

validation.php

```php
<?php
use Respect\Validation\Validator;

function validate_date($date_string)
{
    if(Validator::notEmpty()->validate($date_string)) {
        return date('F jS Y',$date_string);
    }
}
```

This validate call will return true or false, so it is okay to use it as a conditional

There is a lot going on here! Let's take a closer look.

Inspecting the Validator Class

This line of code means, “If the $date_string is not empty, return true; otherwise, return false.”

validate *is a method that tests our data against the rules — in this case,* notEmpty

```
Validator::notEmpty()->validate($date_string);
```

notEmpty *is a method (function) in the* Validator *class, which validator calls a rule*

Validation of a Date

We have validated that our date string is not empty, but how do we know it is a date?

First, we are evaluating that `$date_string` *is not empty!*

```
Validator::notEmpty()->validate($date_string);
Validator::date()->validate($date_string);
```

`date()` *verifies that the string entered is in a valid date format*

Chaining Validation Methods

With Respect/Validation, we can chain rules together for simplicity and clarity.

Now we're validating that we have some data and that it looks like a date in one line

```
Validator::date()->notEmpty()->validate($date_string);
```

We can chain the validation rules together
by using an object operator ->

This is great, but can we do more?

Creating Custom Validators

Clarifying our code is easy when we create a custom validator variable.

We can assign a Validator class with rules to a variable

```
$date_validator = Validator::date()->notEmpty();

$date_validator->validate($date_string);
```

Now we only have to run the `validate` *method on the* `$date_validator` *variable we created!*

Validation With Our Custom Validator

validation.php

```php
<?php
use Respect\Validation\Validator;

function validate_date($date_string)
{
    $date_validator = Validator::date()->notEmpty();

    if ($date_validator->validate($date_string)) {
        $date_time = strtotime($date_string);
        return date('F jS Y', $date_time);
    }
}
```

Create our validator for the date

Use a conditional to test our validator

Requiring a Date Format

We can set a format inside our date validator call, using any format from the PHP date function.

validation.php

```
<?php
use Respect\Validation\Validator;

function validate_date($date_string)
{
    $date_validator = Validator::date('d-m-Y')->notEmpty();

    if ($date_validator->validate($date_string)) {
        $date_time = strtotime($date_string);
        return date('F jS Y', $date_time);
    }
}
```

The format `d-m-Y` *requires that the date be entered like this: 30-12-2017*

Returning a Message on Error

validation.php

```php
<?php
use Respect\Validation\Validator;

function validate_date($date_string)
{
    $date_validator = Validator::date('d-m-Y')->notEmpty();

    if ($date_validator->validate($date_string)) {
        $date_time = strtotime($date_string);
        return date('F jS Y', $date_time);
    } else {
        return 'The date must be in a DD-MM-YYYY format.';
    }
}
```

If the validator fails, return a message about the format!

Validator Class Review

What we have gone over in this section

- How to create an object
- Using our **Validator** class and methods
- The **validate** method with the **notEmpty** rule
- Chaining rules with the object operator
- Storing custom validators in a variable
- Using custom validators in a conditional

CLOSE ENCOUNTERS with PHP

CLOSE ENCOUNTERS with PHP