MODEL OBSERVER

It seems like you might be referring to an "Observer" pattern in Laravel, but it's not entirely clear from your question. If you're looking for information on Model Observers in Laravel, I can provide you with a comprehensive explanation and tutorial.

In Laravel, Model Observers allow you to attach certain events to model events, such as creating, updating, or deleting records in the database. This allows you to execute code or perform actions when these events occur without having to modify the controller or model directly.

Here's a step-by-step tutorial on how to use Model Observers in Laravel:

Step 1: Create an Observer

First, you need to create an observer. You can generate an observer using the following Artisan command:

```bash
php artisan make:observer MyModelObserver --model=MyModel
```

```

Replace `MyModel` with the name of the model you want to observe. This command will create an observer class in the `App/Observers` directory.

**Step 2: Define the Observer Methods**

Open the generated observer file (`App/Observers/MyModelObserver.php`), and you will see methods like `creating`, `created`, `updating`, `updated`, `deleting`, and `deleted`. You can define the logic you want to execute during these events.

```php
<?php

namespace App\Observers;

use App\Models\MyModel;

class MyModelObserver
{
    public function creating(MyModel $model)
```
```

```
{
    // Logic to execute before the model is created
}

public function created(MyModel $model)
{
    // Logic to execute after the model is created
}

public function updating(MyModel $model)
{
    // Logic to execute before the model is updated
}

public function updated(MyModel $model)
{
    // Logic to execute after the model is updated
}

public function deleting(MyModel $model)
{
    // Logic to execute before the model is deleted
```

```
    }

    public function deleted(MyModel $model)
    {
        // Logic to execute after the model is deleted
    }
}
```

Step 3: Register the Observer

Open the `App/Providers/AppServiceProvider.php` file and add the following code to the `boot` method:

```php
use App\Models\MyModel;
use App\Observers\MyModelObserver;

// ...

public function boot()
{
```

```
    MyModel::observe(MyModelObserver::class);
}
```

Replace `MyModel` and `MyModelObserver` with your actual model and observer names.

Step 4: Run the Application

With the observer in place, your defined logic will be executed when the specified events occur on the model. Remember to run the migrations if you've added new events like `creating` or `updating`:

```bash
php artisan migrate
```

Now, your Observer is set up, and you can add custom logic to respond to various events in your model's lifecycle.