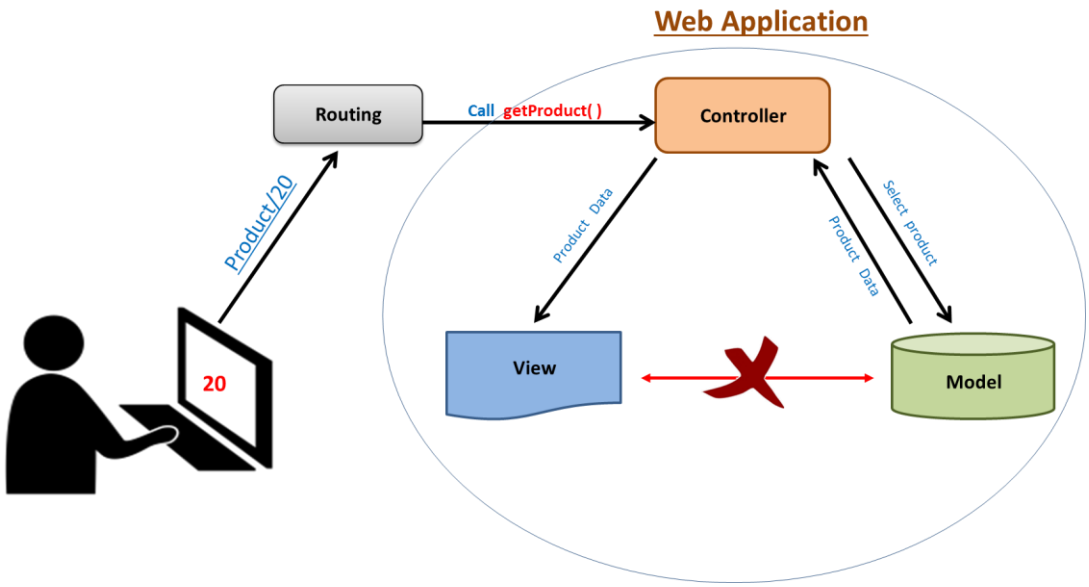
مقدمة حول المتحكمات Controller

ذكرنا مسبقاً أن إطار العمل لارافل يتبع نمط الـMVC ، و هو أسلوب في البرمجة يقوم على تجزئة التطبيق إلى ثلاثة أجزاء (Model-View-Controller) ، و يفيد هذا التقسيم في عزل واجهات المستخدم عن المنطق(الشيفرة البرمجية) ، مما يسهل من عملية الاختبار و التطوير.

الـController هنا يلعب دور بالغ الأهمية حيث يمثل الوسيط بين View و الـModel ، ليتضح لك المقصود بذلك دعنا نستعرض هذا **المثال** :

في تطبيق مبني لارافل و يتبع نمط الـ MVC ، لنفترض أن المستخدم يريد عرض تفاصيل حول المنتج رقم ٢٠ ، سيقوم المستخدم بإدخال رقم السجل الذي يريده و هو هنا ٢٠ ، و الآن و حسب العنوان الذي طلبه المستخدم سيقوم ملف الـRoutes بتوجيه الطلب إلى Controller معين ، الآن دور الكونترولر هو معالجة هذا الطلب ،أي أنه سيعمل على إحضار البيانات المطلوبة من جدول المنتجات ، إذاً سيتواصل مع المودل الذي يمثل جدول المنتجات ، و يطلب بيانات المنتج رقم ٢٠ ، و بعدها سيرسل هذه البيانات إلى الفيو ليقوم الأخير بعرضها للمستخدم بالشكل المناسب .



كما رأيت في المثال السابق فالـ view لن يتخاطب مع الـmodel بشكل مباشر ، وإنما يتم هذا عبر الـcontroller الذي يمثل الوسيط .

غالبية العمليات البرمجية تتم عبر الكونترلورز و لهذا فهي تتضمن معظم الشيفرة البرمجية للتطبيق ، و بالرغم من أنه يمكن كتابة تلك التعليمات البرمجية في ملف الراوت أو حشوها ضمن الـ views إلا أن هذا الأسلوب خاطيء تماماً ، لأنه سيجعل من مهمة صيانة و توسيع التطبيق أعقد بكثير .

إنشاء المتحكمات Controllers

جميع المتحكمات تكون بداخل المجلد Controllers الموجود على المسار app/HTTP ، حيث يمكن إنشاءها إما بشكل يدوي كفئة ترث الصنف Controller أو عن طريق الـArtisan و هو الخيار المفضل .

جميع أسماء المتحكمات في تطبيق لارافل ينبغي أن تلحقها االكلمة Controller ، على سبيل المثال UserController ، AdminController .

لإنشاء متحكم جديد عن طريق الـartisan سنقوم بكتابة التالي على سطر الأوامر :

```
php artisan make:controller PhotoController
```

حيث PhotoController هو اسم المتحكم ، سيظهر الآن المتحكم مباشرةً بداخل مجلد Controller كما ترى

أما إذا رغبت بإنشاء المتحكم ضمن مجلد فرعي ، فسيكون الأمر بهذا الشكل ، بحيث يذكر اسم المجلد ثم اسم الكونترولر

```
php artisan make:controller FolderName\PhotoController
```

و الآن يمكنك إضافة الوظائف التي سيقوم بها هذا المتحكم ، على سبيل المثال

إضافة صورة `function store(){ }`

عرض صورة `function show(){ }`

أما إذا كان المتحكم سيؤدي مهمة واحدة فقط ، فيمكنك استخدام `function __invoke()` ليتم تنفيذها بشكل افتراضي عند استدعاء الكونترولر .

❖ إنشاء مسار Route إلى متحكم :

تعلمت سابقاً طريقة إنشاء مسار لتنفيذ مهمة معينة أو لإعادة View معين ، و الآن سنرى طريقة إنشاء مسار لمتحكم ، و هي الحالة الأكثر شيوعاً ، و الطريقة تكون كما يلي :

```
Route::get('photo/show', 'PhotoController@show');
```

هذا السطر يعني قم بتنفيذ المهمة show الموجودة بداخل المتحكم PhotoController و ذلك عند طلب العنوان photo/show .

فلنضع الآن أي تعليمات برمجية لتنفيذها عبر الدالة show ، و ليكن على سبيل المثال اعادة الـview المسمى Photo ، إذاً سيكون شكل الدالة show هكذا

```
    protected function show ()
    {
        return view('photo');
    }
```

و الآن قم بطلب العنوان photo/show و شاهد النتيجة

و في حالة كان المتحكم سيؤدي وظيفة واحدة فقط ، فيمكنك استخدام **invoke_** في المتحكم بالشكل التالي :

```
    protected function __invoke()
    {
        return view('photo');
    }
```

و الآن لا داعي لذكر اسم الـfunction ، سنكتفي باسم المتحكم فقط

```
Route::get('photo/show, 'PhotoController');
```

سنطلب العنوان photo/show مرة أخرى ، لاحظ أنه قام بتنفيذ التعليمات الموجودة بداخل invoke مباشرةً.

يمكن أيضاً تمرير بيانات عبر الـRoute إلى المتحكم عبر إضافة اسم البارامتر بين قوسين

```
Route::get('photo/show/{id}, 'PhotoController');
```

يمكنك الآن استخدام هذا الـ id في المتحكم ، على سبيل المثال سأقوم بتمريره إلى الـview في شكل مصفوفة .

```
    protected function __invoke($id)
    {
        return view('photo' , ['id'=>$id]);
    }
```

و للتأكد انه قد مرر فعلاً للـview ، قم بعرض قيمة الـid على الصفحة بالطريقة {{$id}}

متحكّمات الموارد Resource Controllers

تعتبر عمليات إدارة البيانات CRUD (Create , Read , Update , Delete) شائعة جداً في أي تطبيق ويب ، و من المؤكد أنك ستحتاج لتخصيص متحكمات (Controllers) خاصة لتنفيذ هذه العمليات على جداول في قاعدة البيانات ، لارافل توفر لك طريقة سهلة لإنشاء متحكم يشمل مخطط جاهز بجميع الطرق التي تحتاجها ، و لإنشاء متحكم من هذا النوع ، فقط نفذ نفس الأمر الخاص بإنشاء متحكم مع إضافة الكلمة Resource بهذا الشكل .

```
php artisan make:controller CRUDController --resource
```

فلنقم الآن بفتح موجه الأوامر و إنشاء resource Controller ، استعرض المتحكم الناتج و ستجده يتضمن سبع طرق

Index لعرض جميع البيانات

Create لعرض النموذج الخاص بإضافة بيانات جديدة

store حفظ بيانات جديدة

Show لعرض بيانات معينة ، مثلاً عرض بيانات السجل رقم ١

Edit عرض بيانات السجل المطلوب تحديثه

Update تحديث البيانات لسجل معين

Destroy حذف بيانات محددة

و الآن ينبغي أن نحدد المسارات لكل طريقة من هذه الطرق ، بحيث يتم تنفيذ كل function هنا عند طلب عنوان معين

ربما سيخطر لك أنه يجب تعريف سبع مسارات بحسب عدد الطرق (functions) الموجودة هنا بهذا الشكل :

```
Route::get('test','CRUDController@index' );
Route::post('test','CRUDController@store');
Route::get('test/create','CRUDController@create);
Route::get('test/edit/{id}','CRUDController@edit');
Route::patch('test/{id}',' 'CRUDController@update' );
```

```
Route::delete('test/{id}', 'CRUDController@destroy' );
Route::get('test/{id}','CRUDController@show' );
```

لحسن الحظ أن لارافل تقدم لك طريقة مختصرة لتعريف جميع هذه المسارات في سطر واحد فقط ، فيكفي أن تضع هذا السطر ليماثل جميع التوجيهات السابقة

```
Route::resource('Photos', 'PhotoController');
```

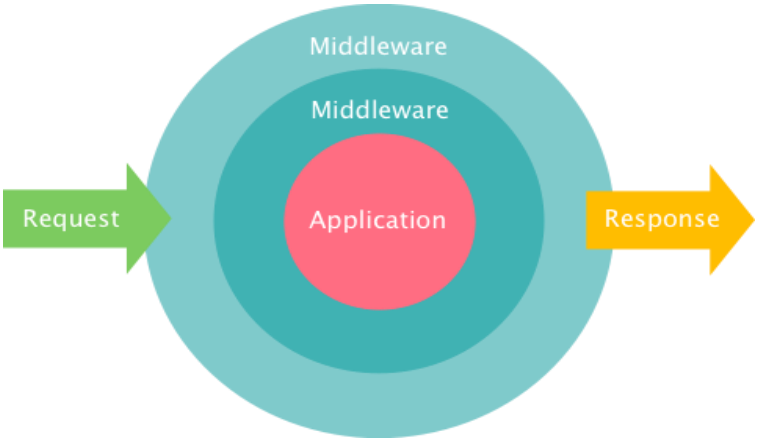
لاحظ أن Route::Resource يأخذ بارامترين أساسيين ، الأول هو عنوان التوجيه ، و الثاني اسم المتحكم (Controller) ، و من خلالهما ستولد جميع المسارات تلقائياً .

و للتأكد من هذا ، قم بفتح موجه الأوامر و اكتب الأمر php artisan route:list ، هذا الأمر مهمته عرض جميع التوجيهات المسجلة في التطبيق ، لاحظ أنه سيعرض التوجيهات لجميع الطرق الموجودة في المتحكم PhotoController ، و التي سجلت في ملف web بالتعليمة Route::Resource .

و الآن فلنقم بطلب العنوان Photo لاحظ انه استدعى التعليمات بداخل index

و هكذا بقية المسارات .

الـ Middleware في لارافل



الـMiddleware هو آلية لترشيح و تصفية طلبات HTTP الداخلة للتطبيق ، أي أنه يعمل كطبقة وسيطة لحماية التطبيق بحيث يسمح فقط بولوج الطلبات التي تحقق شروط معينة .
مثلاً : إضافة Middleware للتحقق من صلاحية المستخدم قبل الولوج للتطبيق ، فإذا كان نوع المستخدم Admin يُسمح له بالدخول ، مالم فإنه يوجَّه إلى صفحة أخرى .

جميع الـMiddlewares تنشئ بداخل المجلد الـMiddleware ، بحيث تشتمل هذه الملفات على التعليمات البرمجية الخاصة بالتعامل مع الطلبات الواردة للتطبيق ، و يتم تسجيلها في الملف Kernal .

في الملف Kernal تسجل جميع مسارات الـMiddlewares ، و ستلاحظ أنها مصنفة إلى ثلاث أنواع :
النوع الأول Global Middleware و هنا تعرف الـMiddleware التي ستعمل مع كل طلب HTTP يتم في التطبيق ، أي أنها تعمل تلقائياً مع جميع الطلبات في التطبيق دون الحاجة إلى استدعائه

النوع الثاني Middleware Groups : و فيه تجمع عدد من مسارات Middlewares في مجموعة واحدة ، بحيث يتم تشغيل المجموعة كاملة عند استدعاء عنوانها فقط ، أي أنه إذا قمنا باستدعاء web فجميع الـ Middlewares المذكورة هنا ستعمل .

النوع الثالث Route Middleware : و هنا يعرف كل Middleware بشكل فردي ، أي أنه عند استدعاء الـAuth فالـMiddleware المسمى Authenticate الموجود على هذا المسار سيعمل مباشرةً.

أعلم بأن الأمر لم يتضح لك حتى الآن ولا يزال هناك الكثير من الغموض

لذا سأقوم بشرح مثال بسيط لنوضح فيه طريقة إنشاء Middleware

لدي هنا الصفحة main .. بعد ادخال العمر في خانة النص و النقر على الزر login فإنه يقوم مباشرة بالانتقال للصفحة التالية و هي home ، لاحظ أنه حالياً يقبل أي رقم أقوم بإدخاله هنا .

سأقوم الآن بعمل قيود على الصفحة home بحيث لا يتم عرضها إلا في حالة كان العمر أقل من ٥٠ مالم فإنه يبقى في نفس الصفحة login ، إذاً سأنشيء Middleware لفحص قيمة الطلب و السماح بالمرور في حالة تطابق الشرط .

الخطوة الأولى إنشاء الـMiddelware

من موجه الأوامر سأكتب الأمر التالي

```
php artisan make:middleware <MiddlewareName>
```

سأسمي هذا الـ الـMiddelware AgeMiddleware . لاحظ أن الملف ظهر مباشرةً بداخل هذا المجلد .

الخطوة الثانية إضافة شروط مرور الطلب على ملف الـMiddelware .

سننتقل لـMiddelware الذي قمنا بإنشاءه مسبقاً و نضيف شرط مرور الطلب بداخل الدالة handle ، لاحظ أن هذه الدالة تستقبل بارامترين ، $request و يحمل عنوان الطلب الوارد للتطبيق و $next المستخدم لتمرير الطلب للتطبيق .

و الآن فلنقم بوضع شرط هنا لمنع تمرير أي طلب في حالة كانت قيمة المدخل أكبر من أو تساوي ٥٠ ، أما في حالة كانت أقل فإنه يتم تمرير الطلب للتطبيق ، أي أنه سيتوجه للصفحة الهدف بنجاح إذا سيكتب الكود بالشكل التالي :

```
public function handle($request, Closure $next)
{
    if ($request->input('age') >= 50) {
      return redirect('main');
   }
   return $next($request);
}
```

، و هنا العبارة $next($request) تعني مرر الطلب للتطبيق

الخطوة الثالثة

تسجيل الـMiddelware في ملف Kernal ، سنقوم بتسجيله هنا بالاسم age .

```
'age' => \App\Http\Middleware\AgeMiddleware::class
```

المفتاح age يختصر لنا هذا السطر بالكامل ، و سيمكننا الآن استخدامه للإشارة لهذا الـMiddleware .

آخر خطوة و هي ربط الMiddleware بالمسار المطلوب .

فلننتقل لملف web ، هنا المسار الخاص بالتوجيه للصفحة home ، حيث أن المتحكم المذكور سيعيد الـview المسمى home . سنقوم بربطه مع الـMiddleware عبر إضافة اسم الـMiddleware بالشكل التالي .

```
Route::post('main', 'MainController')->middleware('age');
```

إذاً لن يتم التوجيه للصفحة home إلا بعد المرور على AgeMiddleware و التأكد من تحقق الشرط المطلوب .

سنجرب الآن إدخال القيمة ٤٠ سيتم الدخول بنجاح

فلنجرب على القيمة ٧٠ لاحظ أنه منع هذا الطلب لأنه لا يحقق الشرط .