Request Life Cycle in laravel

اولا اللي بتبدأ فيه كل الطلبات اللي بتوصل الـ public/index.php بدايتها بيكون في ملف web server (مثل Apache / Nginx). الملف ده مش بيحتوي على كود كتير، لكنه نقطة انطلاق لتحميل بقية الإطار.

الملف بيحمّل ملف autoloader definition اللي بتولده Composer، وبيجيب instance من التطبيق بتاع Laravel من ملف bootstrap/app.php. وأول حاجة بيعملها Laravel هي إنه بينشئ instance من service container. / application

بعد كده، الطلب بيتوجه لـ HTTP kernel أو console kernel، حسب نوع الطلب اللي بيوصل. دول الاتنين بيكونوا النقطة المركزية اللي بتمر من خلالها كل الطلبات. لنركز الآن على HTTP kernel، اللي بيكون في مسار app/Http/Kernel.php.

الـ HTTP kernel بيمتد من الـ Illuminate\Foundation\Http\Kernel class، اللي بتعرف مصفوفة من bootstrappers اللي بتشتغل قبل ما الطلب يتنفذ. الـ bootstrappers دي بتعمل configure لمعالجة الأخطاء، logging، detect للبيئة اللي التطبيق شغال فيها، وتنفيذ مهام تانية مهمة قبل تنفيذ الطلب. في الغالب، الـ bootstrappers دول بتتعامل مع configuration داخلية في Laravel، وملناش دعوة نتفاعل معاها.

الـ HTTP kernel بيعرف كمان مجموعة middleware اللي كل الطلبات بتمر عليها قبل ما تتنفذ من التطبيق. دول الـ middleware بيعملوا تلاعب بالـ HTTP session، وبيعرفوا إذا كان التطبيق في وضع الصيانة، وبيحققوا في الـ CSRF token، وتاني حاجات. هنتكلم أكتر عن الـ middleware دول قريب.

التوقيع اللي في method لـ HTTP kernel بيبدوا بسيط، وبيستقبل Request ويعيد Response. الـ kernel ده بيبقى جيد، نتحدث الآن عن ال Service Providers، وهي أحد أهم الخطوات في عملية بدء تشغيل تطبيق Laravel. حيث تعمل الـ Service Providers على تشغيل جميع المكونات المختلفة للإطار الأساسي، مثل قاعدة البيانات، والطابور، والتحقق من الصحة، ومكوّنات التوجيه. يتم تكوين جميع موفري الخدمات للتطبيق في ملف الإعدادات app.php / config في مصفوفة الموفرين providers.

وبعد ذلك، سيقوم Laravel بتكرار قائمة موفري الخدمة وإنشاء نسخة من كل منها. بعد الإنشاء، سيتم استدعاء الطريقة register على جميع موفري الخدمات. ثم، بعد التسجيل في جميع موفري الخدمات، سيتم استدعاء الطريقة boot على كل موفر خدمة. هذا حتى يتمكن موفري الخدمات من الاعتماد على جميع ربط الحاويات المسجلة والمتاحة بحلول وقت تنفيذ طريقة التمهيد الخاصة بهم.

يتم تشغيل كل ميزة رئيسية تقدمها Laravel عمومًا وتكوينها بواسطة موفر خدمة. حيث تقوم موفرات الخدمة بتمهيد وتكوين العديد من الميزات التي يقدمها الإطار، ولذلك فإن موفرات الخدمة هي الجانب الأهم في عملية تمهيد Laravel بأكمله.

واحد من موفرات الخدمة الأكثر أهمية في تطبيقك هو App\Providers\RouteServiceProvider. يقوم هذا الموفر بتحميل ملفات المسار الموجودة داخل مجلد المسارات routes الخاص بتطبيقك. لذا، تستطيع الاطلاع على رمز RouteServiceProvider ومعرفة كيف يعمل.

The Difference Between a Framework and a Library

Library:

ال Library هي ملف يحتوي على مجموعة من الأكواد والfunction اليي يمكن استخدامها . تعتبر ال Library واحدة من ال tool المهمه اللي يمكن استخدامها في البرمجة وتساعد على توفير الوقت والجهد في كتابة الأكواد وإنشاء البرامج.

على سبيل المثال، إذا كان المبرمج يحتاج إلى قراءة ملف نصي، يمكنه استخدام مكتبة جاهزة للقراءة والكتابة في الملفات، بدلاً ما يكتب الكثير من الأكواد عشان يعمل ده .

وتتوفر ال Library عادة في لغات Java وC++ وPython وغيرها.

Framework:

ال Framework ده عبارة عن برنامج أساسي بيحتوي على مجموعة من ال Library والأدوات اللي يمكن استخدامها في إنشاء التطبيقات بشكل أسرع . يعتبر ال Framework من أهم الأدوات التي يستخدمها المبرمجين في تطوير البرامج.

ويتضمن Framework دايما مكتبات جاهزة عشان تنفيذ مهام معينة، بالإضافة لل (APIs) اللي يمكن استخدامها في التواصل مع مكتبات البرمجة التانيه وخدمات الأخرى.

على سبيل المثال، إطار العمل Laravel في لغة PHP يحتوي على العديد من المكتبات المجهزة مثل Eloquent ORM عشان للتفاعل مع Database، ومكتبة Blade Template دي بتدير عرض الصفحات، بالإضافة إلى APIs عشان اتواصل مع خدمات أخرى مثل التحقق من الEmail وغيرها.

بعد كدا، يمكن للمبرمج استخدام هذه ال Library في برنامجه باستخدام function محددة تحمل اسم الدالة وتستخدم عشان تنفيذ وظائف معينة

أما بالنسبة ل Library ، فهو تصميم برمجي يجمع بين مجموعة من ال Library وال tools ، بيقوا مع بعض عشان تسهل عملية التطويروتخليها أسهل وأسرع. على سبيل المثال، إطار العمل Ruby on Rails يجمع بين اللغة Ruby وقاعدة البيانات SQLite والتطوير بنمط Model-View-Controller والعديد من المكتبات والأدوات الأخرى.

بشكل عام كدا، نقدر نقول إن ال Library و Framework هما tools مهمة للمبرمجين لتسهيل وتسريع عملية البرمجة، وتحسين من جودة البرنامج وكفاءته، وتقليل حجم الأكواد اللازمة.

Facade

في Laravel الـ facade بتاعتها بتقدم interface بتعمل زي static methods لل services اللي موجودة جوا الـ container بتاعها. ال facade دي بتعتبر proxy بتمثل الـ service اللي جوا الـ container.

في ناس في المجتمع اللي بيعملوا ب PHP بيتناقشوا كتير عن اسم facade دا، عشان بيقولوا ان الاسم دا مش بيطبق الـ Facade pattern بشكل كامل. بس في النهاية، الاسم دا هو اللي Laravel استخدمته وممكن تستخدم اي اسم تاني بيناسبك لو عايز.

بالنسبة لكيفية تنفيذ الـ facade في Laravel، انت عارف ان كل service جوا الـ container في Laravel بيكون ليه اسم فريد، ولو عايز توصل لل service دا من غير facade، تقدر تستخدم App::make() او app() helper function.

بس في Laravel، بيستخدموا classes اللي بتقدم الـ facade دي عشان يجعلوا الـ services دي متاحة للمطورين بطريقة اسهل واسرع. بالاستخدام facade، بنحتاج نكتب الكود اللي تحت بس عشان نعمل نفس الشغلة:

```
someService::methodName();
```

في Laravel، كل service في الـ container ليها facade class. الـ facade classes دي بترث من الـ Facade class اللي جوا الـ Illuminate/Support package. اللي الـ facade class دي بتحتاج تنفيذه هو getFacadeAccessor method اللي بيرجع اسم الـ service جوا الـ container.

في الكود اللي فوق، someService بيشير لاسم الـ facade class. وmethodName في الواقع بتكون method جوا الـ service اللي في الـ container. لو ننظر للكود دا من غير سياق Laravel، هيكون معناه ان في class اسمها someService بتعرض method بتسمي methodName() ك static method، بس دا مش الطريقة اللي بتستخدمها Laravel. في القسم اللي بعد كده، هنشوف ازاي الـ Facade class بتشتغل ورا الكواليس.

يمكننا استخدام هذا الـ Facade في تطبيق Laravel عن طريق كتابة:

```
use App\Facades\SomeServiceFacade;

... //

SomeServiceFacade::methodName(); // ...
```

وكدا نستطيع الوصول لخدمة معينة من داخل الحاوية (container) باستخدام الـ Facade بشكل أسهل وأكثر قراءة.

Service Provider

الـ Service Providers في تطبيق Laravel هي المكان الرئيسي اللي بتتبنى فيه التطبيق. في المكان ده بنحط خدمات الـ Laravel الأساسية و خدمات التطبيق بتاعنا و كمان بنحط الكلاسات و الأشياء اللي بتعتمد على بعضها البعض في المكان اللي اسمه " Service Container".

يعني بالمصري، الـ Service Provider هو الحاوية اللي بيتم تدوير الأشياء فيها، زي الخلاط بتاعك في المطبخ. بتحط الأشياء اللي تحتاجها في الخلاط و بيتم مزجها في مكان واحد، في الـ Service Container.

بالنسبة لكيفية استخدامها، في Laravel بنلاقي ملف اسمه "app.php" فيه قائمة بـ Service Providers الأساسية اللي بيستخدمها الـ Laravel نفسه.

وميزة الـ Service Provider إنها بتمنع إنشاء عدة نسخ من نفس الكلاس، فلو عندك كلاس معين بيحتاج إلى عدة dependencies، ممكن تكتب كود لإنشاء الـ dependencies دي كل مرة تستخدم فيها الكلاس، بس ده مش منطقي ولا عملي. لذلك، الـ Service Provider بيساعدك في حل المشكلة دي.

وبالنسبة لإنشاء Service Provider في Laravel، بيوفر Laravel أمرًا لإنشاءهم بسهولة، وده بيتم بكتابة الأمر "php artisan make:provider MyServiceProvider" في الـ terminal. وبعد كدا، بتكتب الكود اللي بداخل الـ Service Provider اللي إنت عاوزه.