Fast-Route 路由库阅读笔记

介绍

fast-route 是一个基于正则表达式的 php 路由库, 用于

- 基于**正则表达式**的路由配置
- **高效**匹配当前请求对应的路由配置

```
composer require nikic/FastRoute
```

Route

一条路由配置一般包含三个部分

- HTTP Method
- Uri Pattern
- Handler

例如:

- lumen (fast-route)

```
/** @var Laravel\Lumen\Routing\Router $router */
$router->get('1/{routeId:\d+}', ['uses' => 'RegexController@routeId']);
```

- yii2

```
'rules' => [
    'PUT,POST post/<id:\d+>' => 'post/update',
]
```

- laravel (symfony3)

```
use Route;

Route::get('1/{routeId}','RegexController@routeId')->where([
    'routeId'=>'\d+' ,
]);
```

高效

- lumen , symfony 4 , thinkphp 5.1.7 等框架均基于 fast-route 的路由匹配优化原理来封装自己的路由库
- 本机 lumen 和 laravel 对比实测:
    - 360 条相同的正则路由 (http method , uri pattern , handler 均相同)
    - 均只启用一个相同的中间件
    - 不使用 route:cache
    - 不使用 opcache
    - 不使用数据库或缓存等服务

	匹配第一条路由	**匹配最后一条路由**	**无匹配的路由**
lumen / fast-route	50 ms	50 ms	50 ms
laravel / symfony 3	101 ms	130 ms	160 ms
yii2	待测试	待测试	待测试

路由组件在 laravel 和 lumen 框架中实现的区别

先对比 laravel 和 lumen 的框架整体流程

laravel app 整体流程

- 初始化 app (container)
- handle 请求, 获取响应
    - 解析 $_GLOBALS 生成 request 对象
    - bootstrap
        - 收集 routes/web.php 中注册的路由规则 [http method , uri , handler] **(collect)**
        - 注册 middlewares 中间件
    - router 组件进行匹配和分发 (**dispatch**)
        - router 组件遍历路由规则, 与 request 对象进行对比, 得到匹配的路由规则和路由参数, **(match)**
        - 前置中间件处理请求
        - 调用路由对象中的 handler 处理请求
        - 后置中间件处理请求
- 输出响应
- 结束

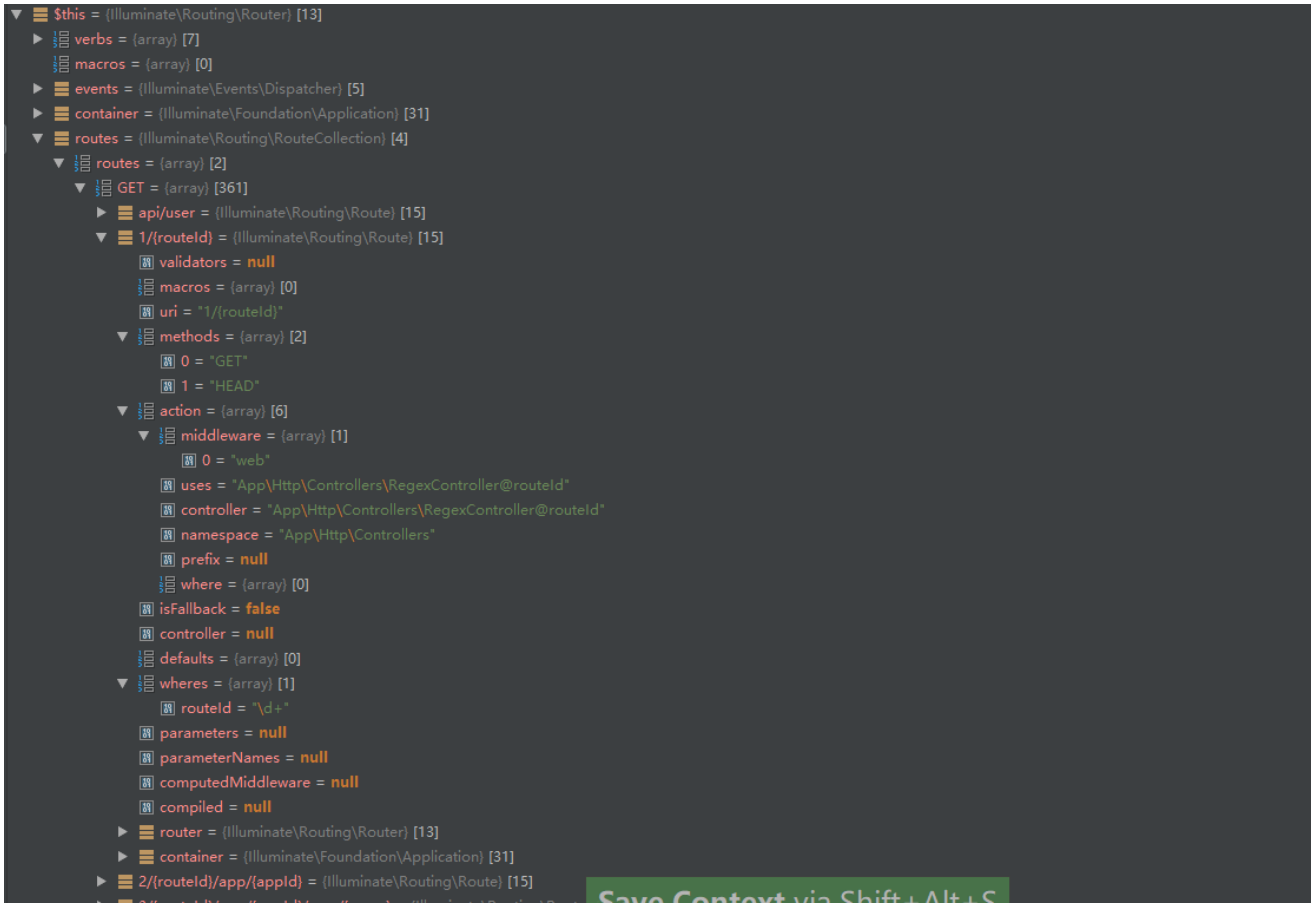
lumen app 的整体流程

- 初始化 app (container)
  - 调用 app 中的 router 对象, 收集 routes/web.php 中注册的路由规则 [http method , uri , handler] **(collect)**
  - 注册 middlewares 中间件
- handle 请求, 获取响应
  - 解析 $_GLOBALS 生成 request 对象
  - RoutesRequests 组件进行匹配和分发 (RoutesRequests 类似于 laravel 中的 HttpKernel) (**dispatch**)
    - 非正则模式尝试匹配请求 **(match)**
    - 上一步失败时,创建 FastRouteDispatcher 对象获取匹配的路由 **(match)**
    - 前置中间件处理请求
    - 调用路由对象中的 handler 处理请求
    - 后置中间件处理请求
- 输出响应
- 结束

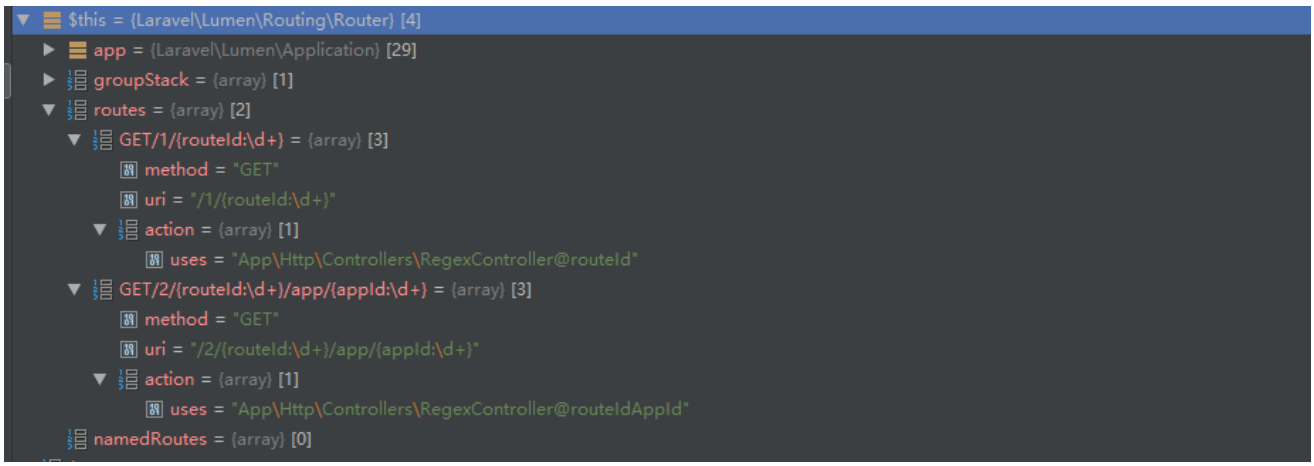
可以看到 laravel 和 lumen 中, 路由部分的流程是相似的, 都经过了 collect -> match -> dispatch 这三步, 那为什么 fast-route 中的路由组件比 laravel 要快呢?

collect 阶段对比

- laravel 收集到的路由规则数组:



- lumen 收集到的路由规则数组



在 collect 阶段, laravel 收集的 Route 格式是对象, lumen 收集的 Route 格式是数组,

但此阶段还**没有**涉及正则解析 , 因此, 这个阶段对耗时的差异影响较小

dispatch 阶段对比

laravel 和 lumen 都是调用 Route 中的 action 相同的 handler , 都经过了相同的中间件, 因此此阶段对耗时的差异影响也较小.

match 阶段:

在lumen 中, 路由相关的组件间的关系如下:

$app

-> $router (Laravel\Lumen\Routing\Router) 用于收集 routes/web.php 中注册的路由规则, 对外提供注册接口

-> $routes 收集到的路由规则数组 , 格式见图一

-> $dispatcher (FastRoute\Dispatcher) 用于处理正则模式的请求

- collector

  收集路由规则, 将路由规则格式化([method , uri , handler]), 并按照 method 分组
- parser
- generator
- dispatcher

fast-route 库的优化原理

其他路由库的优化原理

参考资料