HTTP Requests (درخواست های HTTP)

فهرست محتوا

دسترسی به درخواست HTTP

جهت دسترسی به یک نمونه از درخواست HTTP جاری از طریق dependency injection، می بایست کلاس Illuminate\Http\Request را در تابع constructor یا متد کنترلر خود اعلان نوع (type-hint) نمایید. نمونه ی درخواست جاری به صورت خودکار توسط service container (منبع سرویس های فریم ورک لاراول) تزریق می شود:

```
<?php
 namespace App\Http\Controllers;
 use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
 class UserController extends Controller
{
  /**
   * Store a new user.
   *
   * @param Request $request
   * @return Response
   */
  public function store(Request $request)
  {
    $name = $request->input('name');

    //
  }
}
```

در صورتی که متد کنترلر شما از پارامتر route نیز ورودی دریافت می کند، کافی است آرگومان های route را بلافاصله پس از دیگر dependency ها، لیست نمایید. به عنوان مثال، اگر route را به صورت زیر تعریف کرده باشد:

```
Route::put('user/{id}', 'UserController@update');
```

باز هم می توانید Illuminate\Http\Request را اعلان نوع نموده و با تعریف متد کنترلر به شکل زیر، به id که پارامتر route هست، دسترسی داشته باشید:

```
<?php
namespace App\Http\Controllers;
```

```php
use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
class UserController extends Controller
{
  /**
   * Update the specified user.
   *
   * @param Request $request
   * @param int $id
   * @return Response
   */
  public function update(Request $request, $id)
  {
    //
  }
}
```

اطلاعات پایه ای درخواست

کلاس نمونه ی Illuminate\Http\Request یک مجموعه متد در اختیار ما قرار می دهد که به وسیله ی آن می توانید درخواست HTTP به اپلیکیشن را بررسی و معاینه نمایید. کلاس Illuminate\Http\Request لاراول از کلاس Symfony\Component\HttpFoundation\Request ارث بری دارد. در زیر تعدادی از متدهای این کلاس را با شرح کاربرد ذکر می کنیم:

بازیابی URI یک درخواست

Path یکی از متدهای کلاس نام برده است که URI درخواست مربوطه را بازیابی می کند. بنابراین اگر درخواست به برنامه به این آدرس http://domain.com/foo/bar ارسال شود، متد مزبور foo/bar را برمی گرداند:

```php
$uri = $request->path();
```

متد is به شما این امکان را می دهد بررسی کرده و ببینید آیا URI درخواست ارسالی به برنامه با یک الگوی خاص منطبق هست و همخوانی دارد یا خیر. اگر از متد مزبور استفاده نمایید، در آن صورت اجازه ی استفاده از کاراکتر * به عنوان یک پارامتر wildcard به شما داده می شود:

```php
if ($request->is('admin/*')) {
  //
}
```

برای بدست آوردن آدرس کامل URL (و نه فقط اطلاعات مسیر/path) درخواست فعلی، کافی است متدurl را در نمونه ی request به صورت زیر بکار ببرید:

```php
$url = $request->url();
```

بازیابی متد درخواست

تابع method، فعل یا متد درخواست HTTP را برمی گرداند. همچنین می توانید متد isMethod را برای بررسی اینکه آیا متد HTTP با یک رشته ی مشخص همخوانی و مطابقت دارد یا خیر استفاده نمایید:

```php
$method = $request->method();
if ($request->isMethod('post')) {
  //
}
```

درخواست های PSR-7

استاندارد PSR-7 رابط هایی (interface) را برای پیغام های HTTP از جمله درخواست ها و پاسخ ها طراحی کرده و در اختیار توسعه دهندگان قرار می دهد. برای دریافت نمونه ای از درخواست PSR-7، ابتدا می بایست تعدادی کتابخانه نصب نمایید. لاراول با بهره گیری از کامپوننت Symfony HTTP Message Bridge، درخواست ها و پاسخ های متعارف لاراول را به پیاده سازی های معادل آن در درخواست های PSR-7 (به درخواست های معادل PSR-7 آن ها) تبدیل می کند:

```
composer require symfony/psr-http-message-bridge
composer require zendframework/zend-diactoros
```

پس از نصب کتابخانه های لازم، می توانید با اعلان نوع درخواست (type-hint کردن نوع request) در route یا کنترلر خود، به راحتی یک درخواست PSR-7 دریافت نمایید:

```
use Psr\Http\Message\ServerRequestInterface;
Route::get('/', function (ServerRequestInterface $request) {
  //
});
```

پس از دریافت یک نمونه پاسخ از نوعPSR-7 از route یا کنترلر، آن درخواست به صورت خودکار به پاسخ از نوع متعارف در لاراول تبدیل شده و سپس توسط فریم ورک مورد نظر به نمایش در می آید.

دریافت و بازیابی ورودی

دریافت مقدار ورودی

به وسیله ی چند متد ساده می توانید به راحتی به ورودی کاربر از طریق نمونه ی Illuminate\Http\Request دسترسی داشته باشید. لازم نیست نگران متد HTTP مورد استفاده ی request باشید زیرا که نحوه ی دسترسی تمامی متدهای HTTP به ورودی یکسان است (ورودی های کاربر همگی از یک طریق و به یک شکل در اختیار فعل HTTP قرار می گیرند):

```
$name = $request->input('name');
```

همچنین می توانید از طریق property های نمونه ی کلاس Illuminate\Http\Request به ورودی کاربر دسترسی داشته باشید. برای مثال اگر یکی از فرم های اپلیکیشن دربردارنده ی فیلد name باشد، در آن صورت می توانید به مقدار فیلد ارسال (post) شده به صورت زیر دسترسی پیدا کنید:

```
$name = $request->name;
```

همچنین می توانید یک مقدار پیش فرض را به عنوان آرگومان دوم به متد input ارسال کنید (تا در صورت ارائه نشدن مقدار توسط کاربر، مقدار پیش فرض را به عنوان خروجی برگرداند). از این طریق چنانچه مقدار ورودی درخواست شده در request موجود نبود، مقدار آرگومان دوم بازیابی می شود:

```
$name = $request->input('name', 'Sally');
```

اگر با فرم هایی که ورودی آن ها از نوع آرایه است کار می کنید، در آن صورت برای دسترسی به آرایه کافی است از عملگر نقطه استفاده نمایید:

```
$input = $request->input('products.0.name');
```

بررسی ارائه شدن/نشدن مقدار ورودی

برای اینکه پی ببریم آیا مقداری در درخواست ارائه شده و موجود می باشد یا خیر، از متد has استفاده می کنیم. در صورتی که مقداری در request موجود باشد و نیز آن مقدار یک رشته ی تهی نباشد، این متد مقدار بولی true را برمی گرداند:

```
if ($request->has('name')) {
  //
}
```

بازیابی تمامی داده های ورودی

این امکان نیز برای شما وجود دارد که تمامی داده های ورودی را با استفاده از متد all به صورت/درقالب یک array بازیابی کنید:

```
$input = $request->all();
```

بازیابی بخش یا زیرمجموعه ای از داده های ورودی

اگر لازم می دانید که باید تنها برخی از داده های ورودی را برگردانید، در آن صورت می توانید از یکی از دو متد only یا except استفاده نمایید. هر دو این متدها یک آرایه یا لیستی از آرگومان ها را که به صورت داینامیک ارائه می شوند به عنوان پارامتر ورودی می پذیرد:

```
$input = $request->only(['username', 'password']);
$input = $request->only('username', 'password');
$input = $request->except(['credit_card']);
$input = $request->except('credit_card');
```

ورودی های قبلی

لاراول به شما این امکان را می دهد تا ورودی های قبلی از یک درخواست را در طول درخواست بعدی نگه دارید. این قابلیت به خصوص در زمان پر کردن مجدد فرم ها با مقادیر پیشین، پس از تشخیص خطای اعتبار سنجی، مفید می باشد. گفتنی است که اگر validation services (سرویس های اعتبار سنجی) لاراول را بکار ببرید، در آن صورت دیگر نیازی به استفاده ی دستی از این متدها نخواهید داشت زیرا که برخی از امکانات درون ساخته ی فریم ورک لاراول آن ها را به صورت اتوماتیک فراخوانی می کنند.

قرار دادن ورودی در session با متد flash

متد flash از نمونه ی Illuminate\Http\Request باعث می شود ورودی جاری در session قرار گیرد (اطلاعات ورودی جاری را داخل session قرار می دهد). با این کار ورودی ها در طول درخواست بعدی کاربر به اپلیکیشن تحت وب برای او قابل مشاهده و دسترسی می باشد:

```
$request->flash();
```

همچنین می توانید با استفاده از متدهای flashOnly و flashExcept تنها برخی از اطلاعات درخواست ها را در session قرار دهید:

```
$request->flashOnly('username', 'email');
$request->flashExcept('password');
```

قرار دادن ورودی در session و سپس بازگشت دادن (redirect)

از آنجایی که قرار دادن ورودی در session را اغلب همراه با فرایند ارجاع کاربر به صفحه ی قبلی انجام می دهید، توصیه می کنیم قرار دادن ورودی در session را با استفاده از متد withInput به redirect متصل نمایید:

```
return redirect('form')->withInput();
return redirect('form')->withInput($request->except('password'));
```

بازیابی داده های قبلی

به منظور بازیابی داده های قرار داده شده در session از درخواست قبلی، کافی است متد old را در نمونه ی Request فراخوانی کنید. متد مذکور یک helper کارآمد ارائه کرده که به راحتی داده های ورودی ذخیره شده در session را واکشی می کند:

```
$username = $request->old('username');
```

لاراول همچنین یک نسخه ی سراسری از تابع کمکی old را در اختیار برنامه نویس قرار می دهد. اگر داده های قدیمی را در یک قالب Blade به نمایش گذاشته اید، در آن صورت بهتر است از تابع کمکی سراسری old برای بازیابی داده های قرار داده شده در sessionاستفاده نمایید:

```
{{ old('username') }}
```

Cookie ها

بازیابی کوکی ها از Request

تمامی کوکی های ایجاد شده توسط لاراول، رمزگذاری و سپس با یک کد احراز هویت امضا و نشانه گذاری می شود. بنابراین چنانچه کوکی توسط سرویس گیرنده یا کلاینت مورد تغییر قرار گیرد، نامعتبر تلقی شده و کاملا غیر قابل استفاده در نظر گرفته می شوند. جهت بازیابی و خواندن مقدار کوکی از یک درخواست، می توانید از متد cookie در نمونه ی Illuminate\Http\Request استفاده کنید:

```
$value = $request->cookie('name');
```

الحاق کوکی جدید به یک پاسخ

لاراول یک تابع کمکی سراسری به نام cookie فراهم می کند که به وسیله ی آن قادر خواهید بود نمونه های جدید از Symfony\Component\HttpFoundation\Cookie ایجاد نمایید (به عنوان یک سازنده برای ایجاد نمونه های

جدید از Symfony\Component\HttpFoundation\Cookie ایفای نقش می کند). کوکی ها را می توانید با استفاده از متد withCookie به نمونه ی Illuminate\Http\Response ضمیمه کنید:

```
$response = new Illuminate\Http\Response('Hello World');
$response->withCookie(cookie('name', 'value', $minutes));
return $response;
```

برای ایجاد کوکی هایی با ماندگاری بالا (که تا ٤ یا ٥ سال باقی می مانند و پاک نمی شوند)، می توانید متد forever را در سازنده ی کوکی صدا بزنید. اما پیش از آن بایستی متد کمکی cookie را بدون آرگومان فراخوانی کنید و بعد متد forever را به سازنده ی کوکی بازگشتی متصل نمایید:

```
$response->withCookie(cookie()->forever('name', 'value'));
```

فایل ها

بازیابی فایل های بارگذاری شده

برای دسترسی به فایل های آپلود شده کافی است متد file را فراخوانی کنید. شی بازگشتی که خروجی این تابع است در واقع نمونه ای از کلاس Symfony\Component\HttpFoundation\File\UploadedFile می باشد. این کلاس خود از کلاس SplFileInfo پی اچ پی ارث بری کرده و توابع متعددی برای کار با فایل در اختیار برنامه نویس قرار می دهد:

```
$file = $request->file('photo');
Verifying File Presence
You may also determine if a file is present on the request using the hasFile method:
if ($request->hasFile('photo')) {
  //
}
```

سنجش اعتبار فایل های بارگذاری شده

علاوه بر بررسی و کسب اطمینان از وجود فایل، بد نیست به بررسی امکان آپلود موفق فایل بر روی سرور بپردازیم. این کار از طریق فراخوانی متد isValid امکان پذیر می باشد:

```
if ($request->file('photo')->isValid()) {
  //
}
```

انتقال فایل های بارگذاری شده

برای انتقال فایل به مکان جدید، از متد move کمک می گیریم. این متد فایل را از مکان قرارگیری موقتی خود (که توسط تنظیمات و config پیش فرض php مشخص می شود) به مقصد نهایی مورد انتخاب شما، انتقال می دهد:

```
$request->file('photo')->move($destinationPath);

$request->file('photo')->move($destinationPath, $fileName);
```