



# Laravel Pipelines



Keeping Data  
Transformations Clean

*Terry Matula*

## Past Jobs



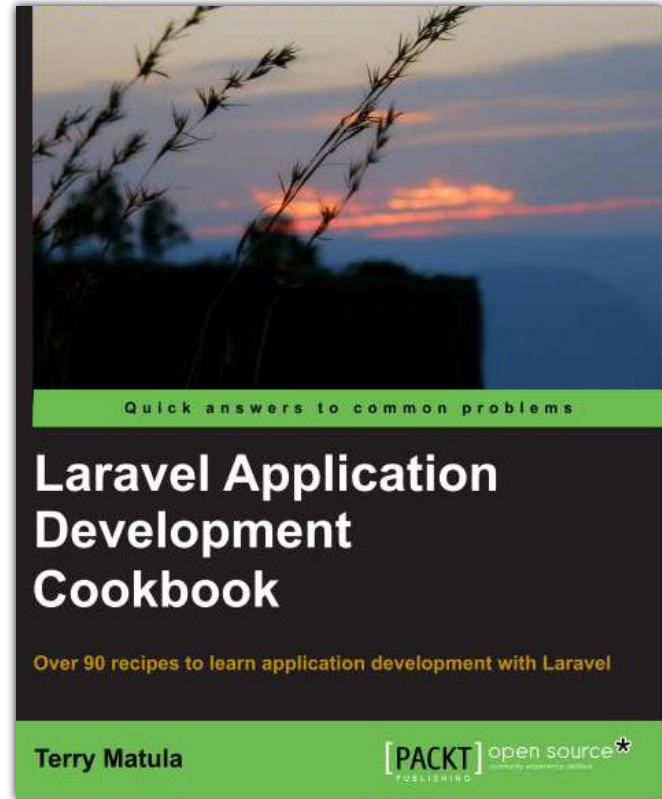
## Past Jobs



# Other things



July 18, 2013



# Business Requirements

# Business Requirements

**CLIENT:** Mario's Plumbing



Wah Hoo!

# Business Requirements

**CLIENT:** Mario's Plumbing

- Customers send message to bot on Slack
- Verify they are a customer
- Convert Slack text format to HTML
- Curse word filter
- Locate on-call plumber
- Send them an email



Letsa Go!

# Slack App

M MarioBot ▼

Settings

Basic Information

Collaborators

Socket Mode

Install App

Manage Distribution

Features

App Home

Org Level Apps

Incoming Webhooks

Interactivity & Shortcuts

Slash Commands

Workflow Steps

OAuth & Permissions

Event Subscriptions

User ID Translation

App Manifest NEW

Beta Features

Submit to App Directory

Review & Submit

## Basic Information

### Building Apps for Slack

Create an app that's just for your workspace (or build one that can be used by any workspace) by following the steps below.

---

### Add features and functionality ▼

Choose and configure the tools you'll need to create your app (or review all [our documentation](#)).

#### Building an internal app locally or behind a firewall?

To receive your app's payloads over a WebSockets connection, enable [Socket Mode](#) for your app.

#### Incoming Webhooks

Post messages from external sources into Slack.

#### Interactive Components

Add components like buttons and select menus to your app's interface, and create an interactive experience for users.

#### Slash Commands

Allow users to perform app actions by typing commands in Slack.

#### Event Subscriptions

Make it easy for your app to respond to activity in Slack.



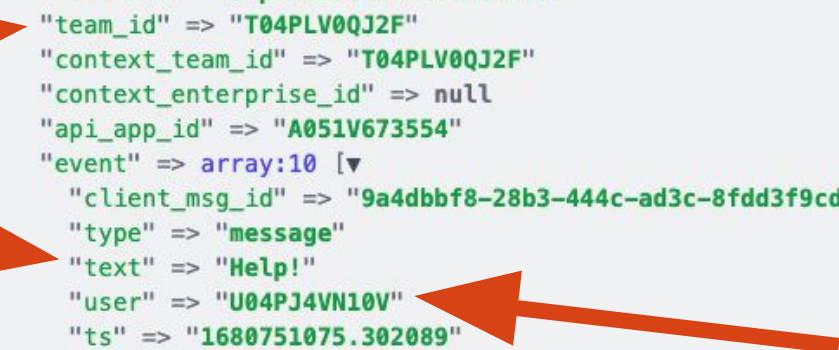
# Roll our own transformation

```
Route::any(uri: 'slack', function (Request $request) {  
    // Slack verification  
    if ($request->has(key: 'challenge')) {  
        return $request->input(key: 'challenge');  
    }  
  
    return 1;  
});
```

# Roll our own transformation

```
array:12 [▼
  "token" => "AJvpXWLn5CCdHNbe0IeC1u35"
  "team_id" => "T04PLV0QJ2F"
  "context_team_id" => "T04PLV0QJ2F"
  "context_enterprise_id" => null
  "api_app_id" => "A051V673554"
  "event" => array:10 [▼
    "client_msg_id" => "9a4dbbf8-28b3-444c-ad3c-8fdd3f9cdb84"
    "type" => "message"
    "text" => "Help!"
    "user" => "U04PJ4VN10V"
    "ts" => "1680751075.302089"
    "blocks" => array:1 [▶]
    "team" => "T04PLV0QJ2F"
    "channel" => "D05249A8892"
    "event_ts" => "1680751075.302089"
    "channel_type" => "im"
  ]
  "type" => "event_callback"
  "event_id" => "Ev051MBAV4KZ"
  "event_time" => 1680751075
  "authorizations" => array:1 [▶]
  "is_ext_shared_channel" => false
  "event_context" => "4-eyJldCI6Im1lc3NhZ2UiLCJ0aWQiOiJUMDRQTfYwU
]
```

# Roll our own transformation



```
array:12 [▼
  "token" => "AJvpXWLm5CCdHNbe0IeC1u35"
  "team_id" => "T04PLV0QJ2F"
  "context_team_id" => "T04PLV0QJ2F"
  "context_enterprise_id" => null
  "api_app_id" => "A051V673554"
  "event" => array:10 [▼
    "client_msg_id" => "9a4dbbf8-28b3-444c-ad3c-8fdd3f9cdb84"
    "type" => "message"
    "text" => "Help!"
    "user" => "U04PJ4VN10V"
    "ts" => "1680751075.302089"
    "blocks" => array:1 [▶]
    "team" => "T04PLV0QJ2F"
    "channel" => "D05249A8892"
    "event_ts" => "1680751075.302089"
    "channel_type" => "im"
  ]
  "type" => "event_callback"
  "event_id" => "Ev051MBAV4KZ"
  "event_time" => 1680751075
  "authorizations" => array:1 [▶]
  "is_ext_shared_channel" => false
  "event_context" => "4-eyJldCI6Im1lc3NhZ2UiLCJ0aWQiOiJUMDRQTfYwU"
]
```

# Roll our own transformation

```
Route::any(uri: 'slack', function (Request $request) {  
    // Slack verification  
>    if ($request->has(key: 'challenge')) {...}  
  
    $slackTeamId = $request->input(key: 'team_id');  
    $slackUserId = $request->input(key: 'event.user');  
    $message = $request->input(key: 'event.text');  
  
    return 1;  
});
```

# Roll our own transformation

```
Route::any(uri: 'slack', function (Request $request) {  
    // Slack verification  
>    if ($request->has(key: 'challenge')) {...}  
  
    $slackTeamId = $request->input(key: 'team_id');  
    $slackUserId = $request->input(key: 'event.user');  
    $message = $request->input(key: 'event.text');  
  
    $client = \App\Models\Client::where(string: 'slack_team_id', $slackTeamId)  
        ->firstOrFail();  
  
    return 1;  
});
```

# Roll our own transformation

```
Route::any(uri: 'slack', function (Request $request) {  
    // Slack verification  
> if ($request->has(key: 'challenge')) {...}  
  
    $slackTeamId = $request->input(key: 'team_id');  
    $slackUserId = $request->input(key: 'event.user');  
    $message = $request->input(key: 'event.text');  
  
    $client = \App\Models\Client::where(string: 'slack_team_id', $slackTeamId)  
        ->firstOrFail();  
  
    $convertedText = (new \League\CommonMark\CommonMarkConverter())  
        ->convert($message);  
  
    return 1;  
});
```

# Roll our own transformation

```
$convertedText = (new \League\CommonMark\CommonMarkConverter())
    →convert($message);

$filteredWords = [
    '/fork/i' ⇒ 'f~~~',
    '/short/i' ⇒ 's~~~',
    '/heck/i' ⇒ 'h~~~',
];

$filteredText = preg_replace(array_keys($filteredWords), array_values($filteredWords), $convertedText);
```

# Roll our own transformation



plumbers			plumber_schedules	
id	name	email	created_at	
1	Mario	mario@mariosplumbing.pipe	2023-05-11 03:02:51	⬆⬇⬆
2	Luigi	luigi@mariosplumbing.pipe	2023-05-11 03:02:51	⬆⬇⬆
3	Peach	peach@mariosplumbing.pipe	2023-05-11 03:02:51	⬆⬇⬆
4	Toad	toad@mariosplumbing.pipe	2023-05-11 03:02:51	⬆⬇⬆

plumbers					plumber_schedules	
id	plumber_id	start_time		end_time		
1	1 →	08:00:00	⬆⬇⬆	13:59:59	⬆⬇⬆	
2	2 →	14:00:00	⬆⬇⬆	19:59:59	⬆⬇⬆	
3	3 →	20:00:00	⬆⬇⬆	01:59:59	⬆⬇⬆	
4	4 →	02:00:00	⬆⬇⬆	07:59:59	⬆⬇⬆	





# Roll our own transformation

```
$filteredText = preg_replace(array_keys($filteredWords), array_values($filteredWords), $filteredText);

$availablePlumber = \App\Models\Plumber::with('relations: schedule')
    →whereHas('relation: schedule', function ($query) {
        // Current time
        $currentTime = now()→toTimeString();
        // current time is between start_time and end_time
        $query→whereRaw("'{$currentTime}' BETWEEN start_time AND end_time")
            →orWhere(function ($query) use ($currentTime) {
                // for overnight schedules
                $query→where('start_time', '>', 'end_time')
                    →where('start_time', '≤', $currentTime);
            })
            →orWhere(function ($query) use ($currentTime) {
                $query→where('start_time', '>', 'end_time')
                    →where('end_time', '≥', $currentTime);
            });
    })
    →first();
```

# Roll our own transformation

```
    }  
    →first();  
  
    Mail::to($availablePlumber→email)  
    →send(new PlumberAlert($client, $filteredText));
```

# Roll our own transformation



↶ Return 0

**Message date:**  
Sat, 13 May 2023, 11:36  
am  
  
**Size:** 760 B

From Plumber Alert <alert@mariosplumbing.pipe>  
To <luigi@mariosplumbing.pipe>  
Subject **Plumber Alert**  
Tags Add tags...

HTML HTML Source Text Headers Raw

## Message from Bowser's Axes

What in the f~~ing s~~s are you doing? h~~~ *me*

# Roll our own transformation

```
Route::any({uri: 'slack', function (Request $request) {
    // Slack verification
    if ($request->has(key: 'challenge')) {
        return $request->input(key: 'challenge');
    }

    $slackTeamId = $request->input(key: 'team_id');
    $slackUserId = $request->input(key: 'event.user');
    $message = $request->input(key: 'event.text');

    $client = \App\Models\Client::where('slack_team_id', $slackTeamId)
        ->firstOrFail();

    $convertedText = (new \League\CommonMark\CommonMarkConverter())
        ->convert($message);

    $filteredWords = [
        '/fork/i' => 'f---',
        '/short/i' => 's---',
        '/hack/i' => 'h---',
    ];

    $filteredText = preg_replace(array_keys($filteredWords), array_values($filteredWords), $convertedText);

    $availablePlumber = \App\Models\Plumber::with('relations'-'schedule')
        ->whereHas('relations'-'schedule', function ($query) {
            // Current time
            $currentTime = now()->toTimeString();
            // current time is between start_time and end_time
            $query->whereRaw("`$currentTime` BETWEEN start_time AND end_time")
                ->orWhere(function ($query) use ($currentTime) {
                    // for overnight schedules
                    $query->where('start_time', '>', 'end_time')
                        ->where('start_time', '≤', $currentTime);
                })
                ->orWhere(function ($query) use ($currentTime) {
                    $query->where('start_time', '>', 'end_time')
                        ->where('end_time', '≥', $currentTime);
                });
            });
        ->first();

    Mail::to($availablePlumber->email)
        ->send(new PlumberAlert($client, $filteredText));

    return 1;
});
```

# Roll our own transformation



**What's Mario's last  
name?**



## Slack challenge to route middleware

```
class SlackChallengeResponse
{
    public function handle(Request $request, Closure $next): Response
    {
        if ($request->has( key: 'challenge')) {
            return response($request->input( key: 'challenge'));
        }

        return $next($request);
    }
}
```

```
})->middleware( middleware: 'slack.challenge');
```

## Object to hold our data

```
class Datasource
{
    1 usage new *
    public function __construct(
        public Request $request,
        public ?string $slackTeamId = null,
        public ?string $slackUserId = null,
        public ?string $message = null,
        public ?\App\Models\Client $client = null,
        public ?string $convertedText = null,
        public ?string $filteredText = null,
        public ?\App\Models\Plumber $availablePlumber = null
    ) {
    }
}
```

```
$datasource = new \App\Pipes\Datasource(
    request: $request
);
```

**Check out:** <https://spatie.be/docs/laravel-data/v3/introduction>



## Set the initial data

```
class SetDatasourceFromRequest
```

```
{
```

```
    new *
```

```
    public function handle(Datasource $datasource): Datasource
```

```
    {
```

```
        $datasource->slackTeamId = $datasource->request->input(key: 'team_id');
```

```
        $datasource->slackUserId = $datasource->request->input(key: 'event.user');
```

```
        $datasource->message = $datasource->request->input(key: 'event.text');
```

```
        return $datasource;
```

```
    }
```

```
}
```

```
$datasource = (new \App\Pipes\SetDatasourceFromRequest())->handle($datasource);
```

## Set the Client

```
class GetClientFromTeamId
{
    new *
    public function handle(Datasource $datasource): Datasource
    {
        $datasource->client = \App\Models\Client::where( string: 'slack_team_id', $datasource->slackTeamId)
            ->firstOrFail();

        return $datasource;
    }
}
```

```
$datasource = (new \App\Pipes\GetClientFromTeamId())->handle($datasource);
```

## Set the Client

```
class GetClientFromTeamId
{
    no usages

    public function __construct(protected ClientRepository $clientRepository)
    {
    }

    public function handle(Datasource $datasource): Datasource
    {
        $datasource->client = $this->clientRepository
            ->getBySlackTeamId($datasource->slackTeamId);

        return $datasource;
    }
}
```

```
$datasource = app(abstract: GetClientFromTeamId::class)->handle($datasource);
```

# Markdown converter

```
class ConvertMarkdownToHtml
{
    new *
    public function handle(Datasource $datasource): Datasource
    {
        $datasource→convertedText = (new \League\CommonMark\CommonMarkConverter())
            →convert($datasource→message);

        return $datasource;
    }
}
```

```
$datasource = (new \App\Pipes\ConvertMarkdownToHtml())→handle($datasource);
```

## Filter text

```
class FilterBadWords
{
    public function handle(Datasource $datasource): Datasource
    {
        $filteredWords = [
            '/fork/i' => 'f~~~',
            '/short/i' => 's~~~',
            '/heck/i' => 'h~~~',
        ];

        $datasource->filteredText = preg_replace(
            array_keys($filteredWords), array_values($filteredWords), $datasource->convertedText
        );

        return $datasource;
    }
}
```

```
$datasource = (new \App\Pipes\FilterBadWords())->handle($datasource);
```

# Get plumber

```
class GetAvailablePlumber
{
    public function handle(Datasource $datasource): Datasource
    {
        $datasource->availablePlumber = \App\Models\Plumber::with('relations: 'schedule')
            ->whereHas('relation: 'schedule', function ($query) {
                // Current time
                $currentTime = now()->toTimeString();
                // current time is between start_time and end_time
                $query->whereRaw("'{$currentTime}' BETWEEN start_time AND end_time")
                    ->orWhere(function ($query) use ($currentTime) {
                        // for overnight schedules
                        $query->where('start_time', '>', 'end_time')
                            ->where('start_time', '≤', $currentTime);
                    })
                    ->orWhere(function ($query) use ($currentTime) {
                        $query->where('start_time', '>', 'end_time')
                            ->where('end_time', '≥', $currentTime);
                    });
            })
            ->firstOrFail();

        return $datasource;
    }
}
```

```
$datasource = (new \App\Pipes\GetAvailablePlumber())->handle($datasource);
```

## Send the email

```
class SendPlumberAlert
{
    public function handle(Datasource $datasource): bool
    {
        Mail::to($datasource→availablePlumber→email)
            →send(new \App\Mail\PlumberAlert($datasource→client, $datasource→filteredText));

        return true;
    }
}

if ((new \App\Pipes\SendPlumberAlert())→handle($datasource)) {
    return 1;
}
```

## Final route

```
Route::any(uri: 'slack', function (Request $request) {  
    $datasource = new Datasource(request: $request);  
    $datasource = (new SetDatasourceFromRequest())→handle($datasource);  
    $datasource = (new GetClientFromTeamId())→handle($datasource);  
    $datasource = (new ConvertMarkdownToHtml())→handle($datasource);  
    $datasource = (new FilterBadWords())→handle($datasource);  
    $datasource = (new GetAvailablePlumber())→handle($datasource);  
  
    if ((new SendPlumberAlert())→handle($datasource)) {  
        return 1;  
    }  
  
    return 0;  
})→middleware(middleware: 'slack.challenge');
```



# Testing

```
test( description: 'forking heck', function () {  
    $filter = new \App\Pipes\FilterBadWords();  
    $datasource = new \App\Pipes\Datasource(  
        request: new \Illuminate\Http\Request(),  
        convertedText: 'What the forking heck is going on here?'  
    );  
  
    $result = $filter→handle($datasource, fn($datasource) ⇒ $datasource);  
  
    expect($result→filteredText)→toBe( expected: 'What the f~~~ing h~~~ is going on here?');  
});
```

# Goal

```
Route::any(uri: 'slack', function (Request $request) {  
    $datasource = new Datasource(request: $request);  
  
    return \Illuminate\Support\Facades\Pipeline::send($datasource)  
        →through([  
            SetDatasourceFromRequest::class,  
            GetClientFromTeamId::class,  
            ConvertMarkdownToHtml::class,  
            FilterBadWords::class,  
            GetAvailablePlumber::class,  
        ])  
        →via(method: 'handle')  
        →then(fn(Datasource $datasource) => (new SendPlumberAlert())→handle($datasource));  
    })→middleware(middleware: 'slack.challenge');
```

## Updated pipe

```
public function handle(Datasource $datasource, Closure $next)
{
    $datasource→slackTeamId = $datasource→request→input( key: 'team_id');
    $datasource→slackUserId = $datasource→request→input( key: 'event.user');
    $datasource→message = $datasource→request→input( key: 'event.text');
    return $next($datasource);
}
```

**Let's take a closer look**



# Documentation



**Taylor Otwell** 🚀 ⚙️

@taylorotwell



Wrote docs for the new "Pipeline" facade. We've used the Pipeline internally in Laravel for years, but more end users have recently discovered its usefulness. So, had to document it. 📖

[laravel.com/docs/10.x/help...](https://laravel.com/docs/10.x/help...)

## **\Illuminate\Pipeline\Pipeline**

->send(mixed \$data)

->through(array \$pipe)

->then(\$closure)

## **\Illuminate\Pipeline\Pipeline::send()**

Set the object being sent through the pipeline.

Parameters: `mixed $passable`

Returns: `Pipeline`

```
public function send($passable)
{
    $this->passable = $passable;

    return $this;
}
```

## **\Illuminate\Pipeline\Pipeline::through()**

Set the array of pipes.

Parameters: `array|mixed $pipes`

Returns: `Pipeline`

```
public function through($pipes)
{
    $this->pipes = is_array($pipes) ? $pipes : func_get_args();

    return $this;
}
```



## **\Illuminate\Pipeline\Pipeline::via()**

Set the method to call on the pipes.

Parameters: `string $method`

Returns: `Pipeline`

```
public function via($method)
{
    $this->method = $method;

    return $this;
}
```

## **\Illuminate\Pipeline\Pipeline::thenReturn()**

Run the pipeline and return the result.

Returns: `mixed`

```
public function thenReturn()
{
    return $this->then(function ($passable) {
        return $passable;
    });
}
```

## **\Illuminate\Pipeline\Pipeline::then()**

```
public function then(Closure $destination)
{
    $pipeline = array_reduce(
        array_reverse($this->pipes()),
        $this->carry(),
        $this->prepareDestination($destination)
    );

    return $pipeline($this->passable);
}
```

## array\_reduce()

```
// Sum an array of numbers  
$array = [1, 2, 3, 4, 5];  
$reduced = array_reduce($array, fn($carry, $item) => $carry + $item, initial: 0);
```

## array\_reduce()

```
// Sum an array of numbers
$array = [1, 2, 3, 4, 5];
$reduced = array_reduce($array, fn($carry, $item) => $carry + $item, initial: 0);
```

```
$carry = 0;
foreach($array as $item) {
    $carry = $carry + $item;
}
```

## **\Illuminate\Pipeline\Pipeline::then()**

```
public function then(Closure $destination)
{
    $pipeline = array_reduce(
        array_reverse($this->pipes()),
        $this->carry(),
        $this->prepareDestination($destination)
    );

    return $pipeline($this->passable);
}
```

## **\Illuminate\Pipeline\Pipeline::then()**

```
public function then(Closure $destination)
{
    $pipeline = array_reduce(
        array_reverse($this->pipes()),
        $this->carry(),
        $this->prepareDestination($destination)
    );

    return $pipeline($this->passable);
}
```

## Pipeline::carry()

```
protected function carry()
{
    return function ($stack, $pipe) {
        return function ($passable) use ($stack, $pipe) {
            try {
                if (is_callable($pipe)) {
                    return $pipe($passable, $stack);
                } elseif (! is_object($pipe)) {
                    [$name, $parameters] = $this->parsePipeString($pipe);
                    $pipe = $this->getContainer()->make($name);
                    $parameters = array_merge([$passable, $stack], $parameters);
                } else {
                    $parameters = [$passable, $stack];
                }

                $carry = method_exists($pipe, $this->method)
                    ? $pipe->{$this->method}(...$parameters)
                    : $pipe(...$parameters);

                return $this->handleCarry($carry);
            } catch (Throwable $e) {
                return $this->handleException($passable, $e);
            }
        };
    };
}
```



## Pipeline::carry()

```
protected function carry()
{
    return function ($stack, $pipe) {
        return function ($passable) use ($stack, $pipe) {
            $carry = method_exists($pipe, $this->method)
                ? $pipe->{$this->method}(...$parameters)
                : $pipe(...$parameters);

            return $carry;
        };
    };
}
```

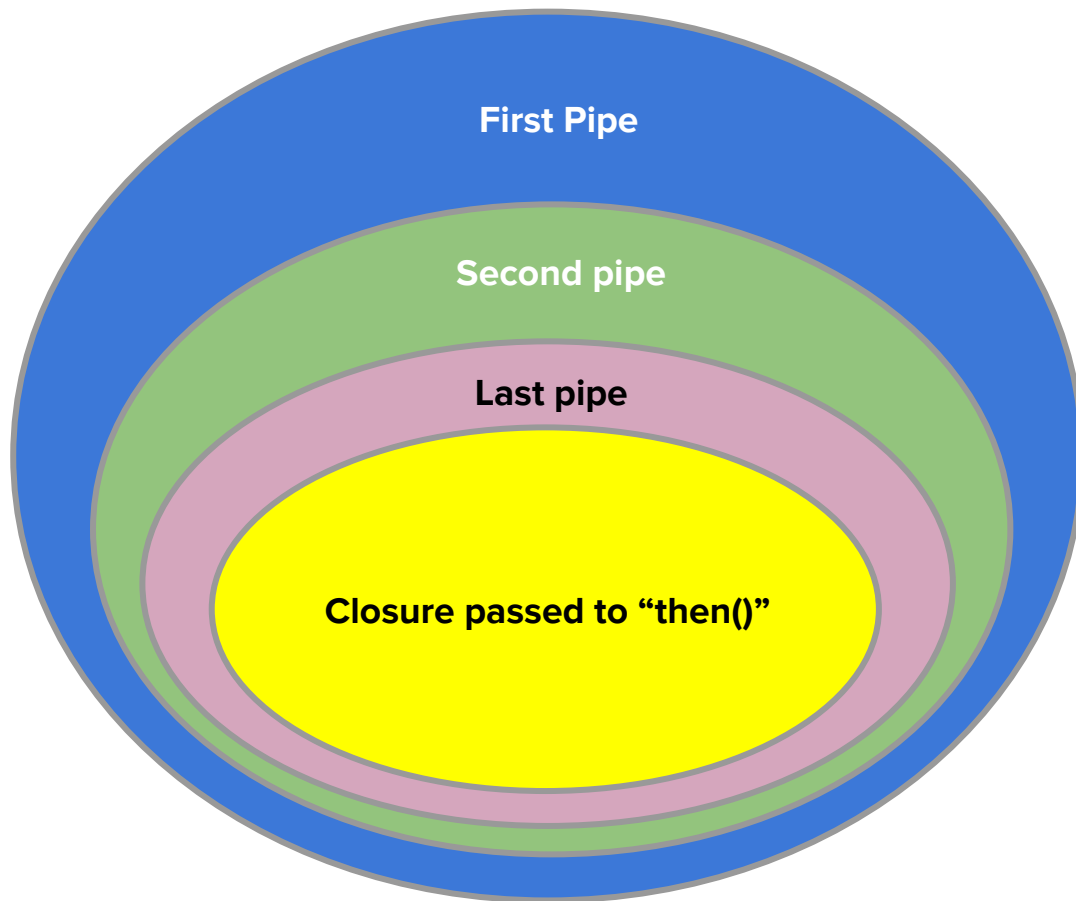


`[$passable, $stack]`

**Super Mario dressed as Shrek.**



# The Callback Onion



## **\Illuminate\Pipeline\Pipeline::then()**

```
public function then(Closure $destination)
{
    $pipeline = array_reduce(
        array_reverse($this->pipes()),
        $this->carry(),
        $this->prepareDestination($destination)
    );

    return $pipeline($this->passable);
}
```

```
// First pipe
$first = function (Datasource $datasource) {
    $datasource->something = '1234';
    // Second pipe
    return function (Datasource $datasource) {
        $datasource->somethingElse = 'abcd';
        // Initial closure
        return function (Datasource $datasource) {
            $datasource->finalthing = 'final';
            return $datasource;
        };
    };
};

$datasource = new Datasource(request: request());
$first($datasource);
```

## **Extending Pipeline Functionality**

# Preprocessing

```
class CustomPipeline extends Pipeline
{
  1 usage

  public function start(Datasource $datasource, array $pipes, $dispatchFunc = null)
  {
    return $this→send($datasource)→through($this→preprocess($pipes))→then($dispatchFunc);
  }

  1 usage

  public function preprocess($pipes): array
  {
    $newPipes = [];
    foreach ($pipes as $pipe) {
      $newPipes[] = function ($datasource, Closure $next) use ($pipe) {
        $pipe = new $pipe;
        if (!$pipe→skip($datasource)) {
          $pipe→handle($datasource, $next);
        }
        return $next($datasource);
      };
    }

    return $newPipes;
  }
}
```

## Preprocessing - Skip

```
public function preprocess($pipes): array
{
    $newPipes = [];
    foreach ($pipes as $pipe) {
        $newPipes[] = function ($datasource, Closure $next) use ($pipe) {
            $pipe = new $pipe;
            if (!$pipe->skip($datasource)) {
                $pipe->handle($datasource, $next);
            }
            return $next($datasource);
        };
    }

    return $newPipes;
}
```



## Preprocessing - Skip

```
public function skip(Datasource $datasource): bool
{
    return $datasource→skipThisPipe ?? false;
}
}
```

## Preprocessing - Logging

```
foreach ($pipes as $pipe) {  
    $newPipes[] = function ($datasource, Closure $next) use ($pipe) {  
        $pipe = new $pipe;  
        Log::info( message: 'Pipe → ' . get_class($pipe), ['start' ⇒ microtime( as_float: true )]);  
        $pipe→handle($datasource, $next);  
        Log::info( message: 'Pipe → ' . get_class($pipe), ['end' ⇒ microtime( as_float: true )]);  
        return $next($datasource);  
    };  
}
```

## Middleware - \Illuminate\Foundation\Http\Kernel::sendRequestThroughRouter

```
protected function sendRequestThroughRouter($request)
{
    $this->app->instance( abstract: 'request', $request);

    Facade::clearResolvedInstance( name: 'request');

    $this->bootstrap();

    return (new Pipeline($this->app))
        ->send($request)
        ->through( pipes: $this->app->shouldSkipMiddleware() ? [] : $this->middleware)
        ->then($this->dispatchToRouter());
}
```

# Middleware - Illuminate\Foundation\Http\Kernel::sendRequestThroughRouter

Working on pipelines and buses.

+933 -133

 **taylorotwell** committed on Dec 25, 2014

Working on pipelines and buses...

factor of HTTP stack

Middleware" term.

112

114

```
this->app->instance('request', $request);  
  
Acade::clearResolvedInstance('request');  
  
this->bootstrap();  
  
return (new Pipeline($this->app))  
    ->send($request)  
    ->through($this->middleware);
```

# Links

<https://medium.com/swlh/laravel-the-hidden-pipeline-part-1-a4ae91fc55a4>

<https://laravel.com/docs/10.x/helpers#pipeline>

<https://laravel-news.com/livestream-laravel-pipelines>

<https://jeffchoa.me/understanding-laravel-pipelines>

<https://github.com/chefhasteeth/pipeline> - “Super-charged Pipelines”

<https://joind.in/event/phptek-2023/laravel-pipelines-keeping-data-transformations-clean>



# Thank You!