# Instalando

```
curl -s http://getcomposer.org/installer | php
php composer.phar create-project zendframework/zend-ex
pressive-skeleton restbeer
```

Escolher:
[3]

- Zend ServiceManager
- FastRoute
- Zend View
- Woops

# Testando

```
cd restbeer
php -S localhost:8080 -t public
```

# Modelos

Copiar o beers.db de http://cl.ly/2e473b2M2k1Z e salvar no diretório do projeto

```
cd restbeer
php ../composer.phar require zendframework/zend-db
```

Escolher:
[1]
[y]

Criar o src/App/src/Model/Beer.php

```php
<?php
namespace App\Model;

use Zend\InputFilter\InputFilter;

class Beer
{
    public $id;
    public $name;
    public $style;
    public $img;

    /**
     * Configura os filtros dos campos da classe
     *
     * @return Zend\InputFilter\InputFilter
     */
    public function getInputFilter()
    {
        $inputFilter = new InputFilter();

        $inputFilter->add([
            'name'     => 'id',
            'required' => false,
            'filters'  => [
                ['name' => 'Int'],
            ],
        ]);
```

```
$inputFilter->add([
    'name'     => 'name',
    'required' => true,
    'filters'  => [
        ['name' => 'StripTags'],
        ['name' => 'StringTrim'],
    ],
    'validators' => [
        [
            'name'    => 'StringLength',
            'options' => [
                'encoding' => 'UTF-8',
                'min'      => 1,
                'max'      => 100,
            ],
        ],
    ],
]);

$inputFilter->add([
    'name'     => 'style',
    'required' => true,
    'filters'  => [
        ['name' => 'StripTags'],
        ['name' => 'StringTrim'],
    ],
    'validators' => [
        [
            'name'    => 'StringLength',
            'options' => [
                'encoding' => 'UTF-8',
                'min'      => 1,
                'max'      => 100,
```

```
                    ],
                ],
            ],
        ]);

        $inputFilter->add([
            'name'     => 'img',
            'required' => false,
            'filters'  => [
                ['name' => 'StripTags'],
                ['name' => 'StringTrim'],
            ],
        ]);

        return $inputFilter;
    }
}
```

# Configurando

Alterar o config/autoload/zend-expressive.global.php e incluir:

```
//https://zendframework.github.io/zend-db/adapter/#creating-an-adapter-using-configuration
    'db' => [
        'driver' => 'Pdo_Sqlite',
        'database' => 'beers.db',
    ],
```

Alterar o config/autoload/dependencies.global.php e incluir:

```
// Use 'factories' for services provided by callbacks/
factory classes.
        'factories' => [
            Application::class => ApplicationFactory::
class,
            Helper\UrlHelper::class => Helper\UrlHelpe
rFactory::class,
                        //adicionar
            App\Factory\Db\Adapter\Adapter::class => A
pp\Factory\Db\Adapter\Adapter::class
        ],
```

Criar o src/App/src/Factory/Db/Adapter/Adapter.php com:

```
<?php

namespace App\Factory\Db\Adapter;

use Interop\Container\ContainerInterface;
use Zend\Db\Adapter\Adapter as ZendAdapter;

class Adapter
{
    public function __invoke(ContainerInterface $conta
iner)
    {
        $config = $container->get('config');
        return new ZendAdapter($config['db']);
    }
}
```

# Crud de cervejas

## Configurar as rotas

No config/routes.php:

```
<?php

$app->get('/', App\Action\HomePageAction::class, 'home
');
$app->get('/api/ping', App\Action\PingAction::class, '
api.ping');
$app->get('/beer', App\Action\Beer\Index::class, 'beer
.index');
$app->post('/beer', App\Action\Beer\Create::class, 'be
er.create');
$app->put('/beer/{id}', App\Action\Beer\Update::class,
 'beer.update');
$app->delete('/beer/{id}', App\Action\Beer\Delete::cla
ss, 'beer.delete');
```

Adicionar as classes em config/autoload/dependencies.global.php, dentro do array de factories:

```
App\Action\HomePageAction::class => App\Action\HomePag
eFactory::class,
App\Action\Beer\Index::class => App\Factory\Beer\Index
::class,
App\Action\Beer\Update::class => App\Factory\Beer\Upda
te::class,
```

```
App\Action\Beer\Create::class => App\Factory\Beer\Crea
te::class,
App\Action\Beer\Delete::class => App\Factory\Beer\Dele
te::class,
```

# Criar as factories

Criar o src/App/src/Factory/Beer/Index.php

```php
<?php

namespace App\Factory\Beer;

use Interop\Container\ContainerInterface;
use Zend\Db\TableGateway\TableGateway;
use App\Action\Beer\Index as Action;

class Index
{
    public function __invoke(ContainerInterface $conta
iner)
    {
        $adapter = $container->get('App\Factory\Db\Ada
pter\Adapter');
        $tableGateway = new TableGateway('beer', $adap
ter);

        return new Action($tableGateway);
    }
}
```

Criar o src/App/src/Factory/Beer/Create.php

```php
<?php

namespace App\Factory\Beer;

use Interop\Container\ContainerInterface;
use Zend\Db\TableGateway\TableGateway;
use App\Action\Beer\Create as Action;

class Create
{
    public function __invoke(ContainerInterface $conta
iner)
    {
        $adapter = $container->get('App\Factory\Db\Ada
pter\Adapter');
        $tableGateway = new TableGateway('beer', $adap
ter);

        return new Action($tableGateway);
    }
}
```

Criar o src/App/src/Factory/Beer/Update.php

```php
<?php

namespace App\Factory\Beer;

use Interop\Container\ContainerInterface;
use Zend\Db\TableGateway\TableGateway;
use App\Action\Beer\Update as Action;
```

```
class Update
{
    public function __invoke(ContainerInterface $conta
iner)
    {
        $adapter = $container->get('App\Factory\Db\Ada
pter\Adapter');
        $tableGateway = new TableGateway('beer', $adap
ter);

        return new Action($tableGateway);
    }
}
```

Criar o src/App/src/Factory/Beer/Delete.php

```
<?php

namespace App\Factory\Beer;

use Interop\Container\ContainerInterface;
use Zend\Db\TableGateway\TableGateway;
use App\Action\Beer\Delete as Action;

class Delete
{
    public function __invoke(ContainerInterface $conta
iner)
    {
        $adapter = $container->get('App\Factory\Db\Ada
pter\Adapter');
```

```
        $tableGateway = new TableGateway('beer', $adap
ter);

        return new Action($tableGateway);
    }
}
```

# Criar as actions

Criar o src/App/src/Action/Beer/Index.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;

class Index implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGateway)
    {
        $this->tableGateway    = $tableGateway;
    }

    public function process(ServerRequestInterface $re
quest, DelegateInterface $delegate)
    {
        $beers = $this->tableGateway->select()->toArra
```

```
y();

        return $delegate->process(
            $request->withParsedBody($beers)
        );
    }
}
```

Criar o src/App/src/Action/Beer/Create.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;

class Create implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGateway)
    {
        $this->tableGateway   = $tableGateway;
    }

    public function process(ServerRequestInterface $re
quest, DelegateInterface $delegate)
    {
        $data = $request->getParsedBody();
```

```
        $this->tableGateway->insert($data);

        return $delegate->process(
            $request->withParsedBody(['id' => $this->t
ableGateway->getLastInsertValue()])
        )->withStatus(201);
    }
}
```

Criar o src/App/src/Action/Beer/Update.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;

class Update implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGateway)
    {
        $this->tableGateway    = $tableGateway;
    }

    public function process(ServerRequestInterface $re
quest, DelegateInterface $delegate)
    {
        $id = $request->getAttribute('id');
```

```
        $beer = $this->tableGateway->select(['id' => $
id]);
        if (count($beer) == 0) {
            return $delegate->process($request)
                                ->withStatus(404);
        }

        parse_str(file_get_contents("php://input"), $d
ata);
        $this->tableGateway->update($data, ['id' => $i
d]);

        return $delegate->process(
            $request->withParsedBody(['id' => id])
        );
    }
}
```

Criar o src/App/src/Action/Beer/Delete.php

```
<?php

namespace App\Action\Beer;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;

class Delete implements MiddlewareInterface
{
    private $tableGateway;

    public function __construct($tableGateway)
```

```
    {
        $this->tableGateway   = $tableGateway;
    }

    public function process(ServerRequestInterface $re
quest, DelegateInterface $delegate)
    {
        $id = $request->getAttribute('id');
        $beer = $this->tableGateway->select(['id' => $
id]);
        if (count($beer) == 0) {
            return $delegate->process($request)
                                ->withStatus(404);
        }

        $this->tableGateway->delete(['id' => $id]);

        return $delegate->process($request)
                        ->withStatus(204);
    }
}
```

# Serialização

Alterar o config/pipeline.php

```
<?php

use Zend\Expressive\Helper\ServerUrlMiddleware;
use Zend\Expressive\Helper\UrlHelperMiddleware;
use Zend\Expressive\Middleware\ImplicitHeadMiddleware;
use Zend\Expressive\Middleware\ImplicitOptionsMiddlewa
```

```
re;
use Zend\Expressive\Middleware\NotFoundHandler;
use Zend\Stratigility\Middleware\ErrorHandler;

$app->pipe(ErrorHandler::class);
$app->pipe(ServerUrlMiddleware::class);
$app->pipeRoutingMiddleware();
$app->pipe(ImplicitHeadMiddleware::class);
$app->pipe(ImplicitOptionsMiddleware::class);
$app->pipe(UrlHelperMiddleware::class);
$app->pipeDispatchMiddleware();
$app->pipe(App\Middleware\Format\Json::class);
$app->pipe(App\Middleware\Format\Html::class);
$app->pipe(NotFoundHandler::class);
```

e config/autoload/dependencies.global.php

```
'dependencies' => [
    'invokables' => [
        App\Middleware\Format\Json::class => App\Middl
eware\Format\Json::class,
    ],
    'factories' => [
        App\Middleware\Format\Html::class => App\Facto
ry\Middleware\Format\Html::class
    ],
],
```

Criar o src/App/src/Middleware/Format/Json.php

```
<?php
```

```php
namespace App\Middleware\Format;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;
use Zend\Diactoros\Response\JsonResponse;

class Json implements MiddlewareInterface
{
    public function process(ServerRequestInterface $re
quest, DelegateInterface $delegate)
    {
        $content = $request->getParsedBody();
        $header = $request->getHeader('accept');
        $accept = null;
        if (isset($header[0])) {
            $accept = $header[0];
        }
        if (!$accept || $accept != 'application/json')
 {
                return $delegate->process($request);
        }

        return new JsonResponse($content);
    }
}
```

Criar o src/App/src/Factory/Middleware/Format/Html.php

```php
<?php

namespace App\Factory\Middleware\Format;
```

```
use Interop\Container\ContainerInterface;
use Zend\Expressive\Template\TemplateRendererInterface
;
use App\Middleware\Format\Html as HtmlMiddleware;

class Html
{
    public function __invoke(ContainerInterface $conta
iner)
    {
        return new HtmlMiddleware($container->get(Temp
lateRendererInterface::class));
    }
}
```

Criar o src/App/src/Middleware/Format/Html

```
<?php

namespace App\Middleware\Format;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;
use Zend\Expressive\Template\TemplateRendererInterface
;
use Zend\Diactoros\Response\HtmlResponse;

class Html implements MiddlewareInterface
{
    private $template;
```

```
    public function __construct($template)
    {
        $this->template = $template;
    }

    public function process(ServerRequestInterface $re
quest, DelegateInterface $delegate)
    {
        $content = $request->getParsedBody();

            return new HtmlResponse($this->template-
>render('beer::index', ['content' => $content]));
    }
}
```

Criar src/App/templates/beer/index.phtml

```
<table class="table table-striped">
  <thead>
    <tr>
      <th>#</th>
      <th>Id/th>
      <th>Name</th>
      <th>Style</th>
      <th>Img</th>
    </tr>
  </thead>
  <tbody>
    <?php foreach ($this->content as $beer): ?>
      <tr>
        <th scope="row"><?php echo $beer['id'];?></th>
        <td><?php echo $beer['name'];?></td>
```

```
        <td><?php echo $beer['style'];?></td>
        <td><img src="<?php echo $beer['img'];?>"></td
>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
```

Adicionar os templates em src/App/src/ConfigProvider.php

```
    public function getTemplates()
    {
        return [
            'paths' => [
                'app'    => [__DIR__ . '/../templates/
app'],
                'error'  => [__DIR__ . '/../templates/
error'],
                'layout' => [__DIR__ . '/../templates/
layout'],
                'beer' => [__DIR__ . '/../templates/be
er'],
            ],
        ];
    }
```

# Validando

Instalar o zend-validator e o zend-filter

```
php ../composer.phar require zendframework/zend-validator zendframework/zend-filter zendframework/zend-inputfilter
```

Criar o src/App/src/Middleware/Validate.php

```
<?php

namespace App\Middleware;

use Interop\Http\ServerMiddleware\DelegateInterface;
use Psr\Http\Message\ServerRequestInterface;
use Interop\Http\ServerMiddleware\MiddlewareInterface;
use Zend\Diactoros\Response\JsonResponse;
use App\Model\Beer;

class Validate implements MiddlewareInterface
{
    public function process(ServerRequestInterface $request, DelegateInterface $delegate)
    {
        $beer = new Beer;
        $inputFilter = $beer->getInputFilter();
        $inputFilter->setData($this->parseBody($request));
        if (!$inputFilter->isValid()) {
            $errors = [];
            foreach ($inputFilter->getMessages() as $key => $values) {
                $messages = [];
                foreach ($values as $message) {
                    $messages[] = $message;
                }
```