Validation in Lumen

> *ALL VALIDATION MUST GO THROUGH VALIDATION LAYER.*

The document below describes one simple implementation to centralize validation. Validation checks like below is not allowed in controller and should use validation layer.

```
// Not allowed, use validation layer
if (empty($request->input('packageNumber'))) {
            return response(['errors' =>
'MISSING_PARAM_PACKAGE_NUMBER'],
Response::HTTP_BAD_REQUEST);
}
```

. . .

Create a file called `AbstractValidator.php` inside `app/Validators` .

```
<?php

namespace App\Validators;

use App\Exceptions\HttpException;
use Dingo\Api\Exception\ResourceException;
use Illuminate\Contracts\Validation\Factory as Validator;

abstract class AbstractValidator
{
    protected $validator;

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

/**
     * Run validator.
     *
     * @param array $params
     * @param array $rules
```

```
     * @param array $messages
     *
     * @throws ResourceException
     */
    public function validate(array $params, array $rules,
$messages = [])
    {
        $validator = $this->validator->make($params, $rules,
$messages);

if ($validator->fails()) {
            throw new ResourceException(
                'validation_error', // we can make this
dynamic if required
                $validator->messages()
            );
        }
    }
}
```

Now for every model that needs to be validated, create a separate file. For instance, for `User` model we can create `UserValidator.php`. Now, we can add different validation methods to validate data.

```
<?php

namespace App\Validators;

class UserValidator extends AbstractValidator
{
    /**
     * Validate required parameters.
     *
     * @param array $params
     *
     * @return null
     */
    public function validateRequired(array $params)
    {
        $rules = [
            'email' => 'required|email|unique:users',
            'password' => 'required|between:4,10',
            'first_name' => 'required|alpha_spaces',
            'last_name' => 'required|alpha_spaces',
            'is_tc_accepted' => 'in:true,1',
        ];

$this->validate($params, $rules);
    }

    /**
     * Validate uuid.
     *
     * @param string $id
     *
     * @return null
```

```
     */
    public function validateUuid(string $id)
    {
        $this->validate(['id' => $id], ['id' => 'uuid']);
    }
```

```
/**
     * Validate Reset Password.
     *
     * @param array $params
     *
     * @return null
     */
    public function validateResetPassword(array $params)
    {
        $rules = [
            'code' => 'required',
            'password' => 'required|between:4,10'
        ];
```

```
$this->validate($params, $rules);
    }
}
```

For custom validation like `uuid` in above case, create a provider called `ValidationServiceProvider.php` and register custom validation inside `boot` method.

```
$this->app['validator']->extend('uuid', function
($attribute, $value) {
    return (bool) preg_match('/^[a-z0-9]{8}-[a-z0-9]{4}-[a-
z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12}$/', $value);
});
```

Controller

Inject validator in constructor.

```
protected $validator;
```

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

Validate inputs in single line. For eg:

```
public function register(Request $request)
{
    $this->validator->validateRequired($request->all());
}
```