Plugin Development: Form Requests

Form requests let you validate request data before handling it in the controller action.

You've now got a working settings page, but anything can be entered and saved. In most cases, only certain values can be allowed and we can use form requests to apply such validation.

Form requests should be placed in the Requests folder. Each request can have rules defined on what the request data should be like and define custom error messages to show if the data is not valid to those rules. There is also the option to set who is authorised to make the request. Below is an example request for our settings page created in the previous section.

Requests/SettingsRequest.php

```
<?phpnamespace AddonsPluginsHelloWorldRequests;use AppHttp
RequestsFormRequest;use Lang;class SettingsRequest extends
 FormRequest{
    /**     * Determine if the user is authorized to make this request.     *     * @
return bool     */    public function authorize()    {        return true;    }
    /**     * Get the validation rules that apply to the request.     *     * @return
array     */    public function rules()    {        return [
            'setting' => 'required|alpha_num',        ];    }
    /**     * Get the error messages for the defined validation rules.     *     * @r
eturn array     */    public function messages()    {        return [
            'setting.required' => Lang::get(
'Plugins#HelloWorld::validation.setting_required'),        ];    }}
```

By setting the authorize function to always return true, it means any authenticated operator with permission to the plugin can make the request. We may wish to limit this to certain operator groups or those with certain permissions (check previous section). The rules are defined as an array with a list of rules for each request input, separated by the pipe (|) character. In our example, we have defined that the field 'setting' must be required and can only have an alphanumeric value. The messages are also defined as an array, keyed by field name and then by the matching rule, allowing you to define a message for each rule entered for an input. If a custom message is not set, it will use a default validation message.

We must now set the request to run before the action is called, this can be done by setting it as the first parameter on the controller action.

```
public function updateSettings(AddonsPluginsHelloWorld
RequestsSettingsRequest $request){    ...}
```

Now when an operator tries to save the settings with no value or an invalid value for the 'setting' field, it will show them an error and not save, retaining what they attempted to save in the settings view on the redirect.

This is useful in pointing out what is wrong with the data they're trying to submit, but we can further enhance it by giving them hints as they're filling out the form with Javascript validation. We can use the JsValidator facade and feed in the request class for it to automatically generate the Javascript needed.

Controller

```
/** * Get the settings page * * @return IlluminateContractsViewView */public function getSettingsPage(){
    return TemplateView::other('Plugins#HelloWorld::settings')        ->with('jsValidator', JsValidator::formRequest(AddonsPluginsHelloWorldRequestsSettingsRequest::class))        ->with('fields', $this->getSettings());}
```

Finally we just need to get it include the Javascript it generated in the view, which can be done by using the scripts_footer block at the bottom of the view file. We must also add the validate class to the form to indicate that it should attempt to validate on the fly.

View

```
{% block content %}    {{ form_model(fields, {'method': 'POST', 'route': 'plugin.helloworld.settings.update', 'class': 'validate'}) }}    ...{% endblock %}{% block scripts_footer %}    {% if jsValidator|default is not empty %}        {{ jsValidator|raw }}    {% endif %}{% endblock %}
```

Now it'll no longer allow submitting the form with invalid values and tell the user exactly what the problem is with what they've entered right away.

Find out more about [redacted] and view all the available [redacted] at the Laravel website.