Nim Form Bundle

Release

February 01, 2014

Contents

1 Installation **1**
- 1.1 Adding required bundles to the kernel . . . 1
- 1.2 Twig configuration . . . 1
- 1.3 http://phpspec.net Specifications . . . 2

2 Form type **3**
- 2.1 Gender type . . . 3
- 2.2 Politeness type . . . 3
- 2.3 Colorpicker type . . . 3
- 2.4 Delete type . . . 3

3 Form extension **5**
- 3.1 Form . . . 5
- 3.2 Date extension . . . 5
- 3.3 Text extension . . . 5
- 3.4 Collection extension . . . 5

4 Form extensions **9**
- 4.1 Eternicode Datepicker Extension . . . 9
- 4.2 Select2 Extension . . . 10
- 4.3 Colorpicker Extension . . . 11
- 4.4 Factory options plugins . . . 12

5 Bootstrap templates **13**
- 5.1 Field macros . . . 13
- 5.2 Notification macros . . . 14
- 5.3 Pagination macros . . . 14

CHAPTER 1

Installation

We assume you're familiar with Composer, a dependency manager for PHP. Use following command to add the bundle to your *composer.json* and download package.

If you have Composer installed globally.

```
$ composer require nim/form-bundle:0.1.0@dev
```

Otherwise you have to download .phar file.

```
$ curl -sS https://getcomposer.org/installer | php
$ php composer.phar require nim/form-bundle:0.1.0@dev
```

1.1 Adding required bundles to the kernel

You need to enable the bundle inside the symfony kernel.

```
<?php

// app/AppKernel.php

public function registerBundles()
{
    $bundles = array(
        new NIM\FormBundle\NIMFormBundle(),
    );
}
```

1.2 Twig configuration

```
twig:
    form:
        resources:
            - 'NIMFormBundle:Bootstrap3:form.html.twig'
```

1.3 http://phpspec.net Specifications

```
$ cd path/bundle/
$ composer install --dev
$ bin/phpspec run -fpretty
```

CHAPTER 2

Form type

2.1 Gender type

Create a choice type with the gender as choices (male, female).

Parent Choice

Name gender

```
$form->add('fieldName', 'gender')
```

2.2 Politeness type

Create a choice type with the politeness as choices (madam, miss, mister).

Parent Choice

Name politeness

```
$form->add('fieldName', 'politeness')
```

2.3 Colorpicker type

Parent Text

Name colorpicker

```
$form->add('fieldName', 'colorpicker')
```

2.4 Delete type

Parent Button

Name delete

```
$form->add('fieldName', 'delete')
```

CHAPTER 3

Form extension

3.1 Form

Options are added to the form type:

- **legend** (string) : Legend of the form type.

3.2 Date extension

Options are added to the form type:

- **placeholder** (string) : Placeholder, the default value is the format option.

3.3 Text extension

Options are added to the form type:

- **placeholder** (string) : Placeholder.

3.4 Collection extension

Two options are added to the basic collection form type:

- **button_add_label** (default : 'form.collection.add') : Label of the addition button
- **item_by_line** (default : 1) : integer assigned to the template to calculate the number of items by line.

Example with bootstrap : **boxWidth = 12 / item_by_line**

3.4.1 HTML markup

The container element needs to have *data-form-type="collection"* as html attribute. It is used to enable the form collection plugin. All data attributes beginning by data-form-collection are used by the plugin and should not be removed.

```
<div data-form-type="collection" data-prototype='...'>
    <div data-form-collection="list" class="row collection-list">

        <input type="hidden" data-form-prototype="prototypeName" value="..." />

        <div data-form-collection="item"
             data-form-collection-index="XX"
             class="col-md-{{ boxWidth }} collection-item">

            <!-- Put here your sub form -->

            <a href="#" data-form-collection="delete">
                Delete
            </a>
         </div>
    </div>

    <a href="#" data-form-collection="add">
        Add
    </a>
</div>
```

List of HTML attributes:

- **data-prototype** : form prototype
- **data-form-collection="list"** : container of the list of the collection item
- **data-form-collection="item"** : container of the collection item
- **data-form-collection-index="XX"** : index of the current collection item
- **data-form-collection="delete"** : HTML element used to remove collection item
- **data-form-collection="add"** : HTML element used to add a new collection item using the form prototype.
- **data-form-collection="update"** : HTML element used to update the collection item when on change event is fired. Element has to belong to the current collection item.
- **data-form-prototype="update"** : HTML element used to update the form prototype when change event is fired.

3.4.2 Updating dynamically the form Prototype

When a HTML element which has **data-form-prototype="update"** as HTML attribute fired change event, the plugin will find the new prototype from the server if the **data-form-url="Url"** is specified. The value of the input/select and the position of the collection item will be submitted to the server.

```
<div data-form-collection="item" data-form-collection-index="1">
    <select data-form-url="example.com/update">
        <option value="rule">Rule</option>
        <option value="action">Action</option>
    </select>
</div>
```

In this example, the value of the select and the position will be sent to the server.

Another way exists, hidden inputs can be used too if you don't want to generate the prototype by the server. You need to insert as many hidden inputs as select options in the page. They need to have attribute like **data-form-prototype="prototypeName"**. "prototypeName" needs to match to one of all the select options.

```
<div data-form-collection="item" data-form-collection-index="1">

    <input type="hidden" data-form-prototype="rule" value="..." />
    <input type="hidden" data-form-prototype="action" value="..." />

    <select>
        <option value="rule">Rule</option>
        <option value="action">Action</option>
    </select>
</div>
```

In this example, when you select Rule, the plugin will replace the current form prototype by the value of the hidden input which has data-form-prototype=”rule”.

CHAPTER 4

Form extensions

4.1 Eternicode Datepicker Extension

Extended type date

Rendered as input text field

Inherited options these options inherit from the date date type

Available options, see the javascript plugin doc for more details :

- **plugin_rendered** (string) : Enable or not the jquery plugin, it is enabled by default (available values : plugin or none).
- **leading_zero** (boolean) : Add the leading zero to the date format.
- **autoclose** (boolean) : Plugin option.
- **before_show_day** (string) : Plugin option.
- **calendar_weeks** (boolean) : Plugin option.
- **clear_btn** (boolean) : Plugin option.
- **days_of_week_disabled** (array) : Plugin option.
- **end_date** (string) : Plugin option.
- **force_parse** (boolean) : Plugin option.
- **inputs** (array) : Plugin option.
- **keyboard_navigation** (boolean) : Plugin option.
- **language** (string) : Plugin option.
- **min_view_mode** (string or integer) : Plugin option.
- **orientation** (string) : Plugin option.
- **start_date** (string) : Plugin option.
- **start_view** (string) : Plugin option.
- **today_btn** (boolean) : Plugin option.
- **today_highlight** (boolean) : Plugin option.
- **week_start** (integer) : Plugin option.

The locale and the date form will be automatically calculated.

Enable the extension in your service.xml:

```
<services>
    <service id="nim.form.extension.datepicker"
             class=">NIM\FormBundle\Form\Extension\Plugin\EternicodeDatepickerExtension">
        <tag name="form.type_extension" alias="date" />
    </service>
</services>
```

More informations on eternicode datepicker :

Website http://eternicode.github.io/bootstrap-datepicker

Github https://github.com/eternicode/bootstrap-datepicker

4.2 Select2 Extension

Extended type choice

Rendered as select field

Inherited options these options inherit from the choice type

Available options, see the javascript plugin doc for more details :

- **plugin_rendered** (string) : Enable or not the jquery plugin, it is enabled by default (available values : plugin or none).
- **width** (string) : Plugin option.
- **load_more_padding** (integer) : Plugin option.
- **close_on_select** (bool) : Plugin option.
- **open_on_enter** (bool) : Plugin option.
- **container_css** (string) : Plugin option.
- **dropdown_css** (string) : Plugin option.
- **container_css_class** (string) : Plugin option.
- **dropdown_css_class** (string) : Plugin option.
- **format_result** (string) : Plugin option.
- **format_selection** (string) : Plugin option.
- **sort_results** (string) : Plugin option.
- **format_result_css_class** (string) : Plugin option.
- **format_selection_css_class** (string) : Plugin option.
- **format_no_matches** (string) : Plugin option.
- **format_input_too_short** (string) : Plugin option.
- **format_input_too_long** (string) : Plugin option.
- **format_selection_too_big** (string) : Plugin option.
- **format_load_more** (string) : Plugin option.

- **format_searching** (string) : Plugin option.
- **minimum_results_for_search** (integer) : Plugin option.
- **minimum_input_length** (integer) : Plugin option.
- **maximum_input_length** (integer) : Plugin option.
- **maximum_selection_size** (integer) : Plugin option.
- **id** (string) : Plugin option.
- **matcher** (string) : Plugin option.
- **separator** (string) : Plugin option.
- **token_separators** (array) : Plugin option.
- **tokenizer** (string) : Plugin option.
- **escape_markup** (string) : Plugin option.
- **blur_on_change** (bool) : Plugin option.
- **select_on_blur** (bool) : Plugin option.
- **adapt_container_css_class** (string) : Plugin option.
- **adapt_dropdown_css_class** (string) : Plugin option.
- **next_search_term** (string) : Plugin option.

Enable the extension in your service.xml:

```
<services>
    <service id="nim.form.extension.select2"
             class="NIM\FormBundle\Form\Extension\Plugin\Select2Extension">
        <tag name="form.type_extension" alias="choice" />
    </service>
</services>
```

More informations on select2 :

Website http://ivaynberg.github.io/select2

Github https://github.com/ivaynberg/select2

4.3 Colorpicker Extension

Extended type colorpicker

Rendered as text field

Inherited options these options inherit from the text type

Available options, see the javascript plugin doc for more details :

- **plugin_rendered** (string) : Enable or not the jquery plugin, it is enabled by default (available values : plugin or none).
- **format** (string) : Plugin option.
- **color** (string) : Plugin option.
- **container** (string) : Plugin option.

- **component** (string) : Plugin option.
- **input** (string) : Plugin option.
- **horizontal** (bool) : Plugin option.
- **template** (string) : Plugin option.

Enable the extension in your service.xml:

```
<services>
    <service id="nim.form.extension.colorpicker"
             class="NIM\FormBundle\Form\Extension\Plugin\MjolnicColorpickerExtension">
        <tag name="form.type_extension" alias="colorpicker" />
    </service>
</services>
```

More informations on select2 :

Website http://mjolnic.github.io/bootstrap-colorpicker/

Github https://github.com/mjolnic/bootstrap-colorpicker/

4.4 Factory options plugins

Some plugin options are functions, you can not define them directly in your php form classes. The solution is to use the the plugin options factory, you can add function like that :

```
$.PluginOptionsFactory.add('optionFunctionName', function() {
    return (...)
});
```

You can define them in the form class like that :

```
$builder->add('startedAt', 'date', array(
    'option_name' => 'optionFunctionName',
));
```

CHAPTER 5

Bootstrap templates

5.1 Field macros

field_row : Display the value of a form_row

- label : Label of the field (it can be translation key)
- value : Value of the field
- labelWidth : With of the label (default : 2)

translatableContant : Display the translation of a contact

- entityName : Name of your entity (example : mission)
- fieldName : Name of your field (example : type)
- value : Value (example : option1)

Caution : your translation file need to have the same organisation for each entity :

```
mission:
    field:
        type
            option:
                option1 : Option number 1
                option2 : Option number 2
```

boolean : Display a boolean (with a pretty icon) and wrap it in a link

- bool : Value of the boolean
- successUrl : Href of the link if bool == true
- errorUrl : Href of the link if bool == false

date : Display a date

- date : Value of the date
- default : Default if the date is null
- formatter : IntlFormatter constant : short, medium, long, full (default : long)

date_time : Display a dateTime

- date : Value of the date
- default : Default if the date is null

- dateFormatter : IntlFormatter constant : short, medium, long, full (default : long)
- hourFormatter : IntlFormatter constant : short, medium, long, full (default : long)

address : Display an address

5.2 Notification macros

flashes : Display all the flashes messages

alert : Display a bootstrap alert :

- text : Message to print (string or array)
- type : danger | warning | success | info (Default : info)

Availables shortcut : danger, error, warning, success and info

5.3 Pagination macros

pagination : Rendered the paginator toolbar (resource_bundle.html.twig)

- paginator : Pagerfanta instance
- options