Template implementation

In order to display data of our FieldType from templates, we need to create and register a template for it. You can find documentation about Field Type templates, as well as on importing settings from a bundle.

In a couple words, such a template must:

- extend `EzPublishCoreBundle::content_fields.html.twig`
- define a dedicated Twig block for the type, named by convention `<TypeIdentifier_field>`. In our case, `eztweet_field`
- be registered in parameters

The template: `Resources/views/fields/eztweet.tpl`

The first thing we will do is create the template. It will basically define the default display of a tweet. Remember that field type templates can be overridden in order to tweak what is displayed and how.

Each FieldType template receives a set of variables that can be used to achieve the desired goal. The variable we care about is `field`, an instance of `eZ\Publish\API\Repository\Values\Content\Field`. In addition to its own metadata (`id`, `fieldDefIdentifier`...), it exposes the Field Value (`Tweet\Value`) through the `value` property.

This would work as a primitive template:

```
{% block eztweet_field %}
{% spaceless %}
    {{ field.value.contents|raw }}
{% endspaceless %}
{% endblock %}
```

`field.value.contents` is piped through the `raw` twig operator, since the variable contains HTML code. Without it, the HTML markup would be visible directly, since twig escapes variables by default. Notice that we nest our code within a `spaceless` tag, so that we can format our template in a readable manner without jeopardizing the display with unwanted spaces.

Using the content fields helpers

Even though the above will work just fine, a couple helpers will help us get something a bit more flexible. The EzPublishCoreBundle::content_fields.html.twig template, where the native FieldType templates are implemented, provides a couple helpers: `simple_block_field`, `simple_inline_field` and `field_attributes`. The first two are used to display a field either as a block, or inline. `field_attributes` makes it easier to use the attr variable, that contains additional (HTML) attributes for the field.

Let's consider that we will display the value as a block element.

First, we need to make our template inherit from `content_fields.html.twig`. Then, we will create a `field_value` variable, that will be used by the helper to print out the content inside the markup. And that's it. The helper will use field_attributes to add the HTML attributes to the generated `div`.

```
{% block eztweet_field %}
{% spaceless %}
    {% set field_value %}
        {{ field.value.contents|raw }}
    {% endset %}
    {{ block( 'simple_block_field' ) }}
{% endspaceless %}
{% endblock %}
```

`fieldValue` is set to the markup we had above, using a `{% set %}` block. We then call the `block` function to process the `simple_block_field` template block.

Registering the template

As explained in the FieldType template documentation, a FieldType template needs to be registered in the eZ Publish semantic configuration. The most basic way to do this would be to do so in `ezpublish/config/ezpublish.yml`:

ezpublish/config/ezpublish.yml

```
ezpublish:
    global:
        field_templates:
            - { template: "EzSystemsTweetFieldTypeBundle:fields:eztweet.html.twig"}
```

However, this is far from ideal. We want this to be part of our bundle, so that no manual configuration is required. For that to happen, we need make our bundle extend the eZ Publish semantic configuration. To do so, we are going to make our bundle's dependency injection extension (`DependencyInjection/EzSystemsTweetFieldTypeExtension.php`) implement `Symfony\Component\DependencyInjection\Extension\PrependExtensionInterface`. This interface will let us prepend bundle configuration:

DependencyInjection/EzSystemsTweetFieldTypeExtension.php

```
use Symfony\Component\DependencyInjection\Extension\PrependExtensionInterface;
use Symfony\Component\Yaml\Yaml;

class EzSystemsTweetFieldTypeExtension extends Extension implements
PrependExtensionInterface
{
    public function prepend( ContainerBuilder $container )
    {
        $config = Yaml::parse( __DIR__ .
'/../Resources/config/ezpublish_field_templates.yml' );
        $container->prependExtensionConfig( 'ezpublish', $config );
    }
}
```

The last thing to do is move the template mapping from `ezpublish/config/ezpublish.yml` to `Resources/config/ezpublish_field_templates.yml`:

```
system:
    default:
        field_templates:
            - {template: "EzSystemsTweetFieldTypeBundle:fields:eztweet.html.twig"}
```

Notice that the `ezpublish` yaml block was deleted. This is because we already import our configuration under the `ezpublish` namespace in the prepend method.

You should now be able to display a content item with this fieldtype from the frontoffice, with a fully functional embed:

Previous: Implementing the Legacy Storage Engine Converter