Templates overview

In this topic:

Introduction

This page explains the basics of using templates in eZ CMS.

> For more in-depth topics related to templating, see:
>
> Content view | View provider configuration | Default view templates | Twig functions | How to use a custom controller to display a content item or location

To apply a template to any part of your webpage, you need three (optionally four) elements:

1. An entry in the configuration that defines which template should be used in what situation
2. The template file itself
3. Assets used by the template (for example, CSS or JS files, images, etc.)
4. (optional) A custom controller used when the template is read which allows you more detailed control over the page.

Configuration

Each template must be mentioned in a configuration file together with a definition of the situation in which it is used. You can use the `ezplatform.yml` file located in the `app/config/` folder, or create your own separate configuration file in that folder that will list all your templates.

> If you decide to create a new configuration file, you will need to import it by including an import statement in `ezplatform.yml`. Add the following code at the beginning of `ezplatform.yml`:
>
> ```
> imports:
>     - { resource: <your_file_name>.yml }
> ```

> If you are using the recommended .yml files for configuration, here are the basic rules for this format:
>
> The configuration is based on pairs of a key and its value, separated by a colon, presented in the following form: key: value. The value of the key may contain further keys, with their values containing further keys, and so on. This hierarchy is marked using indentation – each level lower in the hierarchy must be indented in comparison with its parent.

A short configuration file can look like this:

Sample configuration file

```
ezpublish:
    system:
        default:
            user:
                layout:
pagelayout.html.twig
            content_view:
                full:
                    article:
                        template:
full\article.html.twig
                        match:

Identifier\ContentType: [article]
                    blog_post:
                        controller:
app.controller.blog:showBlogPostAction
                        template:
full\blog_post.html.twig
                        match:

Identifier\ContentType: [blog_post]
                line:
                    article:
                        template:
line\article.html.twig
                        match:

Identifier\ContentType: [article]
```

This is what individual keys in the configuration mean:

- **`ezpublish`** and **`system`** are obligatory at the start of any configuration file which defines views.
- **`default`** defines the siteaccess for which the configuration will be used. "default", as the name suggests, determines what views are used when no other configuration is chosen. You can also have separate keys defining views for other siteaccesses.
- **`user`** and **`layout`** point to the main template file that is used in any situation where no other template is defined. All other templates extend this one. See below for more information.
- **`content_view`** defines the view provider.

> In earlier versions of eZ CMS, `location_view` was used as the view provider. It is now deprecated.

- **`full`** and **`line`** determine the kind of view to be used (see below).
- **`article`** and **`blog_post`** are the keys that start the configuration for one individual case of using a template. You can name these keys any way you want, and you can have as many of them as you need.
- **`template`** names the template to be used in this case, including the folder it is stored in (starting from `app/Resources/views`).
- **`controller`** defines the controller to be used in this case. Optional, if this key is absent, the default controller is used.
- **`match`** defines the situation in which the template will be used. There are different criteria which can be used to "match" a template to a situation, for example a Content Type, a specific Location ID, Section, etc. You can view the full list of matchers here: View provider configuration. You can specify more than one matcher for any template; the matchers will be linked with an AND operator.

In the example above, three different templates are mentioned, two to be used in full view, and one in line view. Notice that two separate templates are defined for the "article" Content Type. They use the same matcher, but will be used in different situations – one when an Article is displayed in full view, and one in line view. Their templates are located in different folders. The line template will also make use of a custom controller, while the remaining cases will employ the default one.

Full, line and other views

Each Content item can be rendered differently, using different templates, depending on the type of view it is displayed in. The default, built-in views are **full** (used when the Content item is displayed by itself, as a full page), **line** (used when it is displayed as an item in the list, for example a listing of contents of a folder), and **embed** (used when one Content item is embedded in another). Other, custom view types can be created, but only these three have built-in controllers in the system.

See View provider configuration for more details.

Template file

Templates in eZ CMS are written in the Twig templating language.

Twig templates in short

At its core, a Twig template is an HTML frame of the page that will be displayed. Inside this frame you define places (and manners) in which different parts of your Content items will be displayed (rendered).

Most of a Twig template file can look like an ordinary HTML file. This is also where you can define places where Content items or their fields will be embedded.

The configuration described above lets you select one template to be used in a given situation, but this does not mean you are limited to only one file per case. It is possible to include other templates in the main template file. For example, you can have a single template for the footer of a page and include it in many other templates. Such templates do not need to be mentioned in the configuration .yml file.

See Including Templates in Symfony documentation for more information on including templates.

The main template for your webpage (defined per siteaccess) is placed in the `pagelayout.html.twig` file. This template will be used by default for those parts of the website where no other templates are defined.

A `pagelayout.html.twig` file exists already in Demo Bundles, but if you are using a clean installation, you need to create it from scratch. This file is typically located in a bundle, for example using the built-in AppBundle: `src/AppBundle/Resources/views`. The name of the bundle must the added whenever the file is called, like in the example below.

Any further templates will extend and modify this one, so they need to start with a line like this:

```
{% extends "AppBundle::pagelayout.html.twig" %}
```

Although using AppBundle is recommended, you could also place the template files directly in `<installation_folder>/app/Resources/views`. Then the files could be referenced in code without any prefix. See Structuring an eZ project for more information.

Template paths
In short, the `Resources/views` part of the path is automatically added whenever a template file is referenced. What you need to provide is the bundle name, name of any subfolder within `/views/`, and file name, all three separated by colons (:)

To find out more about the way of referencing template files placed in bundles, see Referencing Templates in a Bundle in Symfony documentation.

Templates can be extended using a Twig `block` tag. This tag lets you define a named section in the template that will be filled in by the child template. For example, you can define a "title" block in the main template. Any child template that extends it can also contain a "title" block. In this case the contents of the block from the child template will be placed inside this block in the parent template (and override what was inside this block):

pagelayout.html.twig

```
{# ... #}
    <body>
        {% block title %}
            <h1>Default title</h1>
        {% endblock %}
    </body>
{# ... #}
```

child.html.twig

```
{% extends "AppBundle::pagelayout.html.twig" %}
{% block title %}
    <h1>Specific title</h1>
{% endblock %}
```

In the simplified example above, when the `child.html.twig` template is used, the "title" block from it will be placed in and will override the "title" block from the main template – so "Specific title" will be displayed instead of "Default title."

> Alternatively, you can place templates inside one another using the `include` function.
>
> See http://twig.sensiolabs.org/doc/templates.html# for detailed documentation on how to use Twig.

Embed content in templates

Now that you know how to create a general layout with Twig templates, let's take a look at the ways in which you can render content inside them.

There are several ways of placing Content items or their Fields inside a template. You can do it using one of the Twig functions described in detail here.

As an example, let's look at one of those functions: **ez_render_field** . It renders one selected Field of the Content item. In its simplest form this function can look like this:

```
{{ ez_render_field( content, 'description' ) }}
```

This renders the value of the Field with identifier "description" of the current Content item (signified by "content"). You can additionally choose a special template to be used for this particular Field:

```
{{ ez_render_field(
       content,
       'description',
       { 'template': 'AppBundle:fields:description.html.twig' }
   ) }}
```

> As you can see in the case above, templates can be created not only for whole pages, but also for individual Fields.

Another way of embedding Content items is using the **render_esi** function (which is not an eZ-specific function, but a Symfony standard). This function lets you easily select a different Content item and embed it in the current page. This can be used, for instance, if you want to list the children of a Content item in its parent.

```
{{ render_esi(controller('ez_content:viewAction', {locationId: 33, viewType: 'line'}
)) }}
```

This example renders the Content item with Location ID 33 using the line view. To do this, the function applies the 'ez_content:viewAction' controller. This is the default controller for rendering content, but can be substituted here with any custom controller of your choice.

Assets

Asset files such as CSS stylesheets, JS scripts or image files can be defined in the templates and need to be included in the directory structure in the same way as with any other web project. Assets are placed in the `web/` folder in your installation.

Instead of linking to stylesheets or embedding images like usually, you can use the `asset` function.

Controller

While it is absolutely possible to template a whole website using only Twig, a custom PHP controller gives many more options of customizing the behavior of the pages.

See How to use a custom controller to display a content item or location for more information.