Legacy template fallback

⚠ The following **does not apply** to siteaccesses configured with **`legacy_mode: true`**

When the standard view provider is not able to select an appropriate template for a given content, **the system fallbacks to the legacy kernel and delegates the view selection to it**. This ensures that a content is always displayed and hence eases project migration from eZ Publish v4.x to v5.x

Pagelayout and main block

When falling back to the legacy kernel, **the old `content/view` module is run** to return the appropriate view for the given content. However, the pagelayout is not rendered as it needs to be still rendered by Twig in the Symfony part, for consistency. In this regard, the system uses the **Decorator design pattern** to include the generated view in your layout.

For this to work, you need to configure 2 things :

1. Which template you want to use as a base template for legacy fallback
2. The name of the block to use in your layout

Base layout for legacy fallback

You can configure this by setting `ez_publish_legacy.system.<siteaccess>.templating.view_layout`:

ezpublish.yml or config.yml (5.4 / 2014.07+)

```
ez_publish_legacy:
    system:
        my_siteaccess:
            templating:
                view_layout: AcmeDemoBundle::my_layout.html.twig
```

⚠ **Version compatibility**
Semantic configuration for this has been added in 5.4 / 2014.07.
For prior versions, use the following configuration:

ezpublish/config/ezpublish.yml

```
parameters:
    # eZ Publish 5.1+ only
    ezpublish_legacy.my_siteaccess.view_default_layout:
AcmeDemoBundle::my_layout.html.twig
```

Block name

Internally when rendering the view coming from the legacy kernel, a Twig template is created on the fly. This template extends the pagelayout you configured and includes the content inside a block. The name of this block is configurable as well (default is **`content`**).

ezpublish/config/ezpublish.yml

```
parameters:
    ezpublish.content_view.content_block_name: content
```

my_layout.html.twig

```
<!DOCTYPE html>
<html>
<head>
    <!-- ... -->
</head>
<body>
    {% block content %}{# Content will be inserted here #}{% endblock %}
</body>
</html>
```

Module layout

Version compatibility
The module layout setting is available from eZ Publish 5.1.

The module layout can also be defined the same way as the base layout, by setting the **`ezpublish_legacy.<scope>.module_default_layout`** config key. This layout is used to handle "non content" related legacy requests.

Displaying legacy module result

This makes your base layout reusable. However, content generated by legacy modules (i.e. `/ezinfo/about`) will not be automatically inserted as your layout will be rendered as a regular template, receiving `module_result` variable coming from the legacy kernel. The solution is to use 2 different templates, one for the global layout, and another for legacy module rendering:

AcmeDemoBundle::layout.html.twig

```
<!DOCTYPE html>
<html>
<head>
    <!-- Insert everything you need here that is common -->
</head>
<body>
    {% block content %}{# Regular content will be inserted here #}{% endblock %}
</body>
</html>
```

AcmeDemoBundle::layout_legacy.html.twig

```
{% extends "AcmeDemoBundle::layout.html.twig" %}

{% block content %}
    {# module_result variable is received from the legacy controller. #}
    {# It holds the legacy module result #}
    {{ module_result.content|raw }}
{% endblock %}
```

ezpublish.yml or config.yml (5.4 / 2014.07+)

```
ez_publish_legacy:
    system:
        my_siteaccess:
            templating:
                view_layout: AcmeDemoBundle::layout.html.twig
                module_layout: AcmeDemoBundle::layout_legacy.html.twig
```

Version compatibility
Semantic configuration for this has been added in 5.4 / 2014.07.
For prior versions, use the following configuration:

ezpublish/config/ezpublish.yml

```
parameters:
    ezpublish_legacy.my_siteaccess.module_default_layout:
AcmeDemoBundle::layout_legacy.html.twig
    ezpublish_legacy.my_siteaccess.view_default_layout:
AcmeDemoBundle::layout.html.twig
```

Persistent variable

The persistent variable is a special variable in legacy templates that you can set in order to pass values from the content template to the pagelayout. This variable, among others, is accessible from the configured Twig pagelayout thanks to the helper **`ezpublish.legacy`**.

Actually, all data contained in `$module_result` in the legacy kernel is exposed.

Access to the persistent variable

```
<!DOCTYPE html>
<html>
<head>
    <!-- ... -->
    {% if ezpublish.legacy.has( 'content_info' ) %}
        {% set persistent_variable = ezpublish.legacy.get( 'content_info'
)['persistent_variable'] %}
    {% endif %}
</head>
<body>
    {% block content %}{# Content will be inserted here #}{% endblock %}
</body>
</html>
```

Assets

Like the persistent variable, it is possible to require css and/or javascripts assets from a content template through the legacy `ezscript_require()` and `ezcss_require()` template operators.

You can easily retrieve those requested assets from the legacy template in the Twig pagelayout with the **`ezpublish.legacy`** helper.

Getting assets requested from a legacy template

```
<!DOCTYPE html>
<html>
<head>
    <!-- ... -->
    {% set requested_css_files = ezpublish.legacy.get( 'css_files' ) %}
    {% set requested_js_files = ezpublish.legacy.get( 'js_files' ) %}

    {# "configured" assets are the ones defined in design.ini
(FrontendCSSFileList/FrontendJavaScriptList) #}
    {% set configured_js_files = ezpublish.legacy.get( 'js_files_configured' ) %}
    {% set configured_css_files = ezpublish.legacy.get( 'css_files_configured' ) %}
</head>
<body>
    {% block content %}{# Content will be inserted here #}{% endblock %}
</body>
</html>
```

⚠ **Assetic incompatibility**
Warning: If you use Assetic to combine and/or compress your assets, please note that `javascripts` and `stylesheets` Twig tags currently don't support variable file lists, so you won't be able to use them with the example above.