Shopware 6 Storefront-Templates

Shopware 6 Storefront-Templates

Shopware 6: Templates

Twig templates in the Shopware 6 storefront

shopware/platform

Shopware 6 is an open source eCommerce platform realised by the ideas and the spirit of its community. – shopware/platform

[expand title="mehr lesen…"]

Templates

The storefront theme is using Boostrap. Therefore the template structure is a derivative of the bootstrap starter template. The templating engine used is Twig. For styling SASS is used as CSS preprocessor. The bundling and transpiling of the javascript is done with Webpack.

The templates can be found in /src/Storefront/Resources/views/storefront/

Template Top Level

<platform/src/Storefront/Resources/views/storefront/>
└── block
└── component
└── element
└── layout
└── page
└── section
└── utilities
└── base.html.twig

block, element, sectionParts of the experience worldscomponentShared content templates form the basis of the pages.layoutLayout templates. Navigation, header and footer content templates are located here.pageThe concrete templates rendered by the page controllers. This directory contains full page templates as well as private local includes and the pagelet ajax response templates if necessary. utilitiesTechnical necessities used across the content and across all domain concepts.base.html.twigBase page layout of the storefront. This file mainly includes header and footer templates from /layout and provides blocks for the /page templates to overwrite.

Page templates

The page directory contains the entry points which are referenced by page controllers and rendered through the Twig engine. The structure is derived from the page controller naming.

```
<platform/src/Storefront/Resources/views/storefront/page>
└── account
└── checkout
└── content
└── error
└── newsletter
└── product-detail
└── search
```

Inside the directories are the actual templates rendered by the storefront. The inner structure depends on the domain context.

Override and extend templates

Due to the plugin and theme system in shopware it is possible that one storefront template gets extended by multiple plugins or themes, but Twig does not allow multi inheritance out of the box. Therefore we created custom twig functions sw_extends and sw_include, that work like twigs native extends or include, except that they allow for multi inheritance. So it is really important to use the sw_extends and sw_include, instead of the native extends and include.

sw_extends

To inherit a template file, you need to use {% sw_extends %}.

Example:

```
{#
YourPlugin/Resources/views/storefront/layout/header/logo.html.
twig #}
```

```
{%                                                  sw_extends
'@Storefront/storefront/layout/header/logo.html.twig' %}

{# Add an <h2> with underneath the logo image block #}
{% block layout_header_logo_image %}
    {{ parent() }}
    <h2>Additional headline</h2>
{% endblock %}
```

sw_include

If you build your own feature and you need e.g. an element to display the price of the current product you can include existing partials with sw_include like this.

Example:

```
{#     MyPlugin/Resources/views/storefront/page/product-
detail/index.html.twig #}

<div class="my-theme an-alternative-product-view">
    ...

    {% block component_product_box_price %}
        {# use sw_include to include template partials #}
                                                {%     sw_include
'@Storefront/storefront/component/product/card/price-
unit.html.twig' %}
    {% endblock %}

    ...
</div>
```

Inheritance order

The order of the inheritance is determined by the order you set in the theme.json of your active theme.

Styles Top Level

The stylesheets are written in SASS. The organization is

inspired by the 7-1 pattern structure.

```
<platform/src/Storefront/Resources/app/storefront/src/scss>
└── abstract
└── base
└── component
└── layout
└── page
└── skin
└── vendor
└── base.scss
```

The base.scss is the global include file which references styles that are written as an extension of the bootstrap base. For further information just take a look at the excellent description at sass-guidelines.es.

Scripts Top Level

The storefront includes a set of JavaScript plugins providing different functionalities to the storefronts templates on the client side. The plugins are written as **ES6 classes** in **vanilla JavaScript**. Additionally since bootstrap is distributed with the jQuery library the storefront also contains this library.

The script root looks like this:

```
<platform/src/Storefront/Resources/app/storefront/src/script>
└── config
└── helper
└── plugin
└── service
└── utility
└── vendor
└── base.js
```

Sanitize content

Filters tags and attributes from a given string.

The filter can be found in /src/Storefront/Framework/Twig/Extension/SwSanitizeTwigFilter.phpExamples:{{ unfilteredHTML|sw_sanitize }}Uses the default config{{ unfilteredHTML|sw_sanitize({'div': ['style', ...]}, true) }}**allow only** div tags + style attribute for div

[/expand]