Landing Page Field Type (Enterprise)

Landing Page Field Type represents a page with a layout consisting of multiple zones; each of which can in turn contain blocks.

> Landing Page Field Type is only used in the Landing Page Content Type that is included in eZ Enterprise.
>
> The structure of the Landing Page Content Type should not be modified, as it may cause errors.

Name	Internal name	Expected input
`Landing page`	`ezlandingpage`	`string (JSON)`

In this topic:

Related topics:

Creating Landing Page layouts (Enterprise)

Creating Landing Page blocks (Enterprise)

Layout and zones

Layout is the way in which a Landing Page is divided into zones. Zones are organized structures that are deployed over a layout in a particular position.

The placement of zones is defined in a template which is a part of the layout configuration. You can modify the template in order to define your own system of zones.

For information on how to create and configure new blocks for the Landing Page, see Creating Landing Page layouts (Enterprise).

Blocks

For information on how to create and configure new blocks for the Landing Page, see Creating Landing Page blocks (Enterprise).

Rendering Landing Pages

Landing page rendering takes place while editing or viewing.

When rendering a Landing Page, its zones are passed to the layout as a `zones` array with a `blocks` array each. You can simply access them using twig (e.g. `{{ zones[0].id }}`).

> Each div that's a zone or zone's container should have data attributes:
>
> - `data-studio-zones-container` for a div containing zones
> - `data-studio-zone` with zone ID as a value for a zone container

To render a block inside the layout, use twig `render_esi()` function to call `ez_block:renderBlockAction`.

> `ez_block` is an alias to `EzSystems\LandingPageFieldTypeBundle\Controller\BlockController`

An action has the following parameters:

- `contentId` – ID of content which can be accessed by `contentInfo.id`
- `blockId` – ID of block which you want to render

Example usage:

```
{{ render_esi(controller('ez_block:renderBlockAction', {
  'contentId': contentInfo.id,
     'blockId': block.id
 }))
}}
```

As a whole a sample layout could look as follows:

landing_page_simple_layout.html.twig

```
{# a layer of the required "data-studio-zones-container"
attribute, in which zones will be displayed #}
<div data-studio-zones-container>
  {# a layer of the required attribute for the displayed
zone #}
  <div data-studio-zone="{{ zones[0].id }}">
  {# If a zone with [0] index contains any blocks #}
  {% if zones[0].blocks %}
      {# for each block #}
   {% for block in blocks %}
       {# create a new layer with appropriate id #}
    <div class="landing-page__block block_{{ block.type
}}">
           {# render the block by using the
"ez_block:renderBlockAction" controller #}
                    {# contentInfo.id is the id of the
current content item, block.id is the id of the current
block #}
     {{
render_esi(controller('ez_block:renderBlockAction', {
                'contentId': contentInfo.id,
                'blockId': block.id
            }))
     }}
       </div>
   {% endfor %}
  {% endif %}
    </div>
</div>
```

Viewing template

Your view is populated with data (parameters) retrieved from the `getTemplateParameters()` method which must be implemented in your block's class.

Example:

```
/**
    * @param
\EzSystems\LandingPageFieldTypeBundle\FieldType\LandingP
age\Model\BlockValue $blockValue
    *
    * @return array
    */
   public function getTemplateParameters(BlockValue
$blockValue)
   {
       $attributes = $blockValue->getAttributes();
       $limit = (isset($attributes['limit'])) ?
$attributes['limit'] : 10;
       $offset = (isset($attributes['offset'])) ?
$attributes['offset'] : 0;
       $parameters = [
           'title' => $attributes['title'],
           'limit' => $limit,
           'offset' => $offset,
           'feeds' =>
$this->RssProvider->getFeeds($attributes['url']),
       ];
       return $parameters;
   }
```