

OXID eSales Documentation

Visual CMS Module User Manual

Table of Contents

[Copyright, License, Conventions and Legal Notice](#)

[Introduction](#)

[System Requirements](#)

[Installation](#)

[Funktionale Beschreibung](#)

[Developer Information](#)

[Theme Integration](#)

[Widget Creation](#)

[Custom Grid System](#)

[Widget-Modul Migration from OXID eShop v5 to v6](#)

[Help & Support](#)

Copyright

Copyright © 2018 OXID eSales AG, Germany

Copying of this document or its contents, in particular, using texts or parts of text is subject to the explicit prior permission by OXID eSales AG.

The information provided in this document was prepared according to the current state of the art. OXID eSales AG, however, will assume no liability or warranty for the timeliness, correctness and completeness of the information provided. Since errors – despite all efforts – cannot be ruled out entirely, we always appreciate suggestions.

License

Licensing of the software product depends on the shop edition used.

The software for OXID eShop Community Edition is published under the GNU General Public License v3. You may distribute and/or modify this software according to the licensing terms published by the Free Software Foundation. Legal licensing terms regarding the distribution of software being subject to GNU GPL can be found under www.gnu.org/licenses/gpl.html .

The software for OXID eShop Professional Edition and Enterprise Edition is released under commercial license. OXID eSales AG has the sole rights to the software. Decompiling the source code, unauthorized copying as well as distribution to third parties is not permitted. Infringement will be reported to the authorities and prosecuted without exception.

Conventions

The following typographic conventions are used in this document:

`Monospace font with grey background`

for user inputs, source code and URLs

Italic

for file names and paths

Bold font

for input fields and navigation steps

Bold, dark red font

for warnings and important notes

Legal Notice

OXID eSales AG

Bertoldstraße 48

79098 Freiburg

Germany

Phone: +49 (761) 36889 0

Fax: +49 (761) 36889 29

Executive board: Roland Fesenmayr (CEO), Dr. Oliver Ciupke

Supervisory board: Michael Schlenk (chairman)

Headquarters: Freiburg

Country court Freiburg i. Brg.

Commercial register number: HRB 701648

Visual CMS

Version: 3.2

Vendor: OXID eSales AG

Internet: www.oxid-esales.com

E-Mail: info@oxid-esales.com

Description

With the Visual CMS it becomes possible to create and manage CMS page content easily per drag and drop. The simple user interface allows you to create new CMS pages very quickly.

The different elements on a page are created using so-called widgets. Widgets are content such as blocks of texts, images oder even products. For you to be able to start with the creation of a page very quickly, the Visual CMS comes with a number of pre-defined widgets, which you can place wherever you want onto the page using simple drag and drop.

Because of the responsive grid system, all content that you create is responsive and will be displayed optimized for the device of your end user. The Visual CMS is compatible with the responsive OXID Flow theme and with responsive RoxIVE theme from digidesk media solutions.

With an integrated Live search, you will quickly find the CMS pages you wish to edit. The Visual CMS will help you save a lot of time and will deliver compelling results.

Features

- Many scenarios possible through usage of widgets
- Drag & Drop
- Delivering results fast
- Compatible with OXID Flow and RoxIVE
- Live search of CMS pages
- Responsive grid system

System Requirements

The following system requirements must be met for the module to be used:

- OXID eShop PE/EE 6.x or higher
- PHP 5.6, 7.0

Installation

Visual CMS

1. Visual CMS is installed automatically during an OXID eShop standard installation.
2. Navigate to **Extensions > Modules** in the eShop admin area
3. Activate the module "Visual CMS"
4. Update the VIEWS **Service -> Tools -> Update DB Views now**
5. Empty the Tmp directory of the eShop

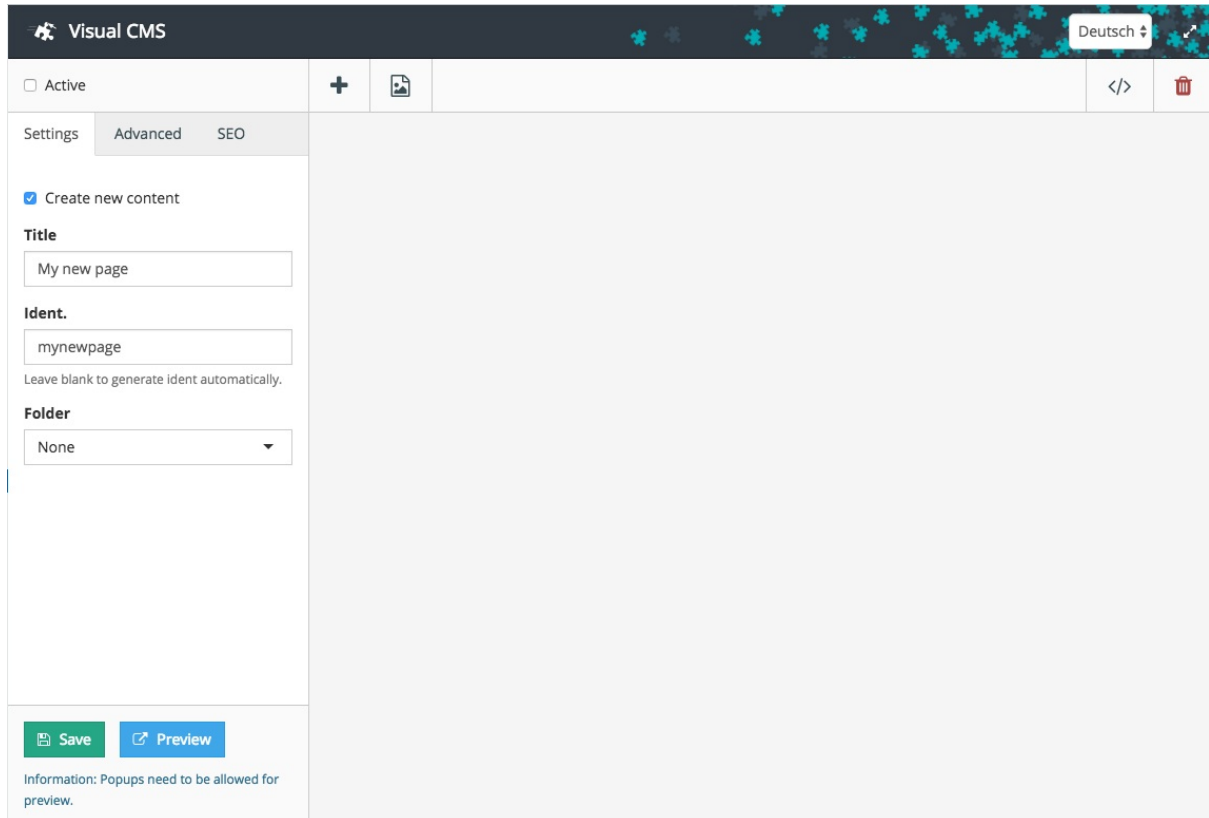
Update

Updates are done automatically with every [OXID eShop Update](#) via Composer.

Introduction

CSM-pages can be changed by adding or editing widgets. The Visual CMS automatically generates a text widget with the page content for CMS pages, which are edited for the first time using the editor. The Visual CMS allows you to edit CMS pages in a fast and comfortable manner. Pages not edited via the Visual CMS still function as before.

Creating New Pages

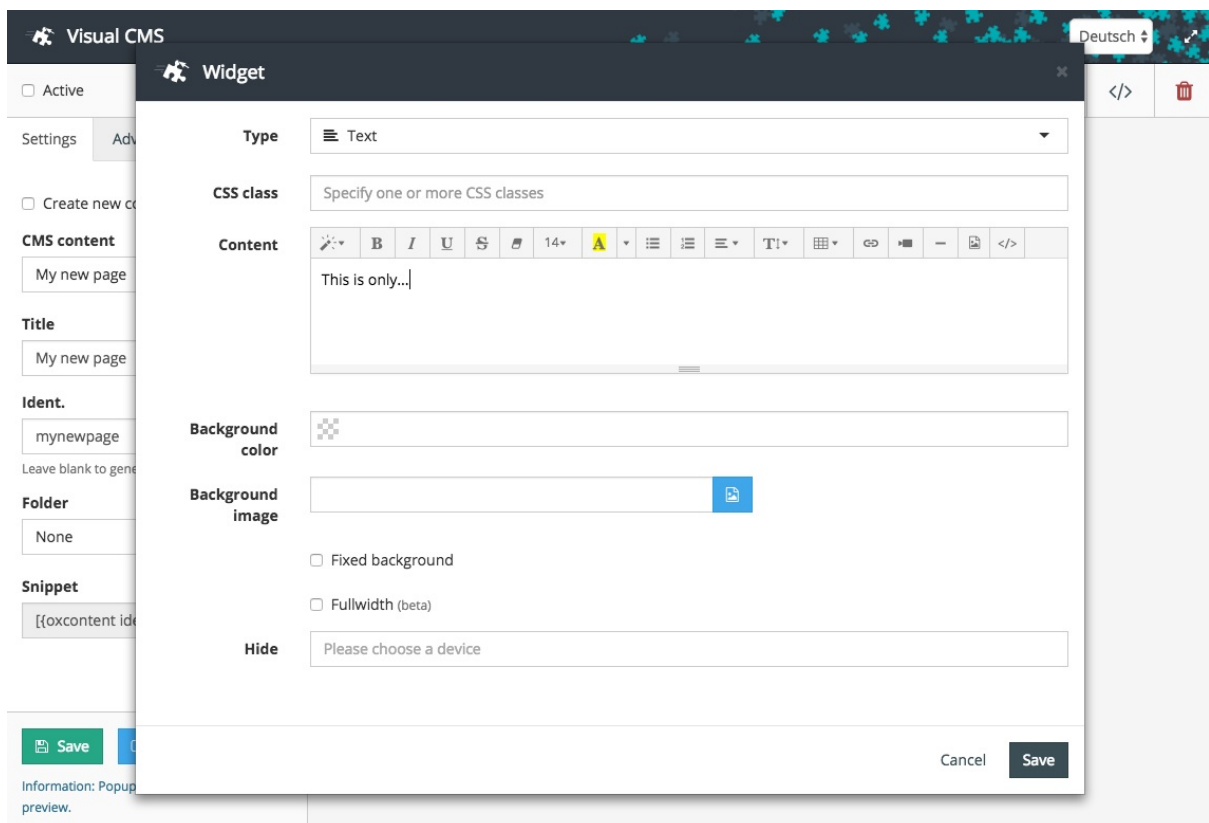
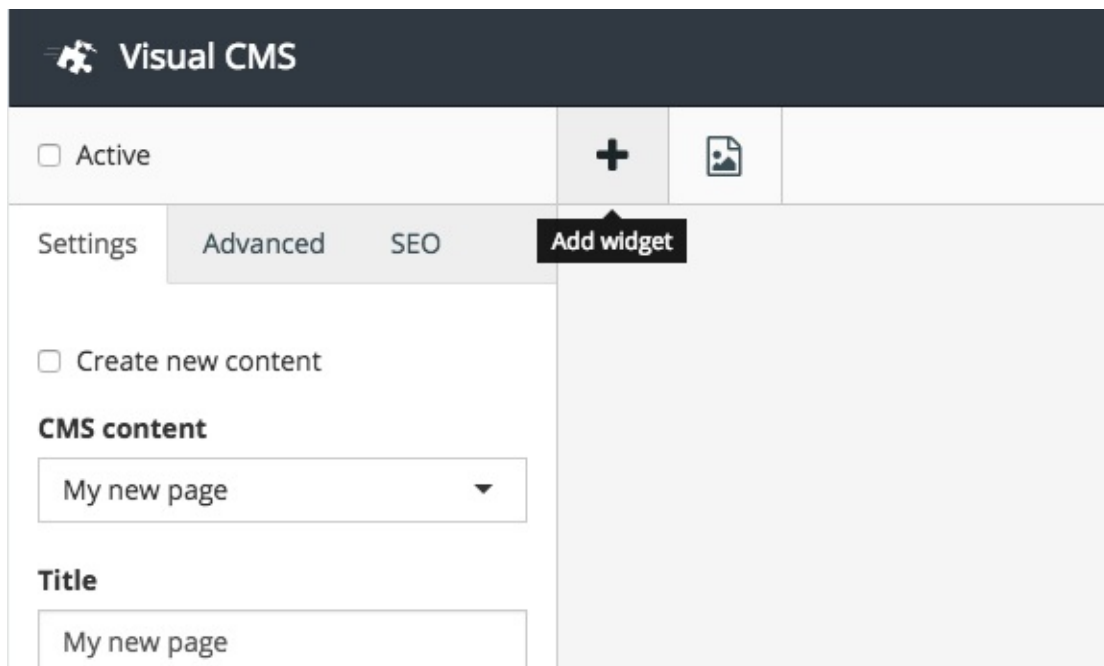


The screenshot shows the Visual CMS interface. At the top, there is a dark header with the 'Visual CMS' logo on the left and a language dropdown set to 'Deutsch' on the right. Below the header is a toolbar with a plus sign, a document icon, and a trash icon. The main content area is divided into a left sidebar and a large central workspace. The sidebar contains a 'Settings' panel with three tabs: 'Settings', 'Advanced', and 'SEO'. The 'Advanced' tab is active, showing a checkbox for 'Create new content' which is checked. Below this are three input fields: 'Title' with the value 'My new page', 'Ident.' with the value 'mynewpage', and 'Folder' with a dropdown menu set to 'None'. At the bottom of the sidebar, there are two buttons: 'Save' (green) and 'Preview' (blue). A small information message at the bottom of the sidebar reads: 'Information: Popups need to be allowed for preview.'

New pages can be easily created using the Visual CMS:

1. Activate the checkbox **Create new content**
2. Enter **Title** und **Ident.**
3. Save

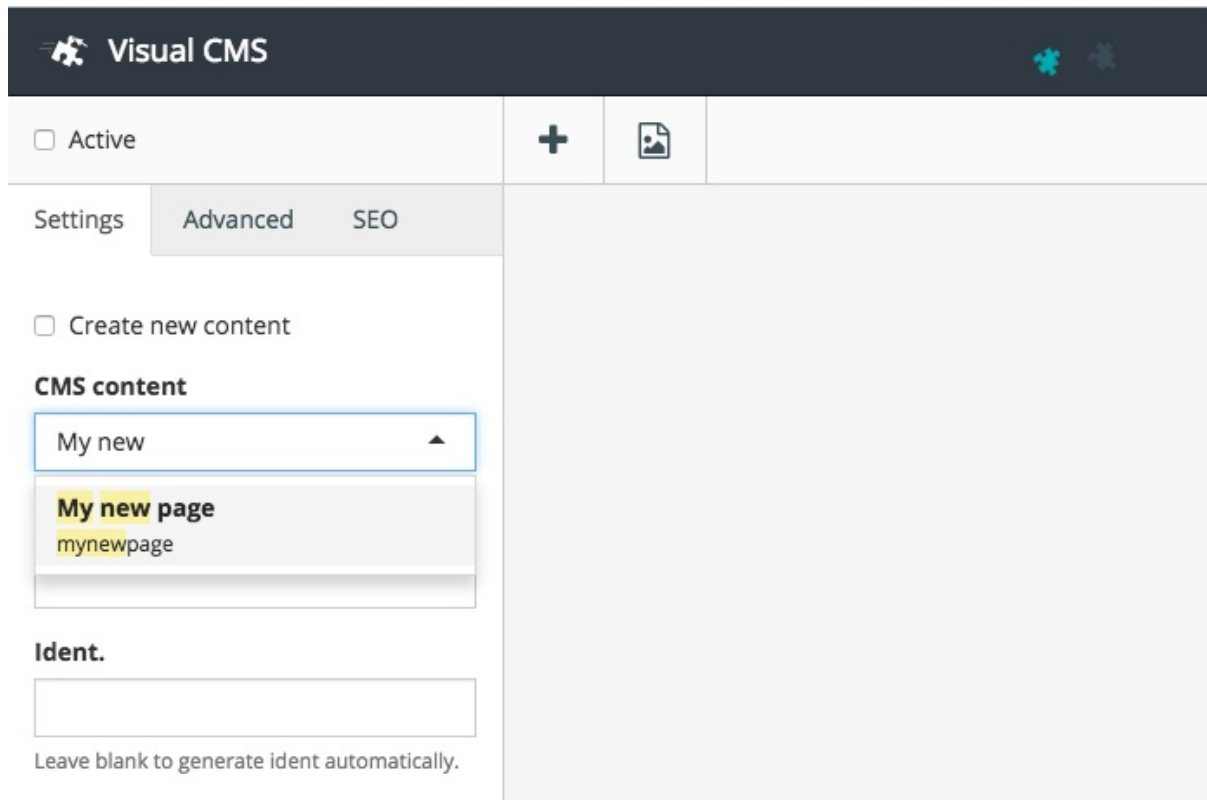
Adding Content



The easiest and fastest way to add content to a CMS-page is to add a text widget:

1. Click on Button **Add widget**
2. Enter your test in the field **Content**
3. Save the widget
4. If necessary, change the size of the widget
5. Click **Save**

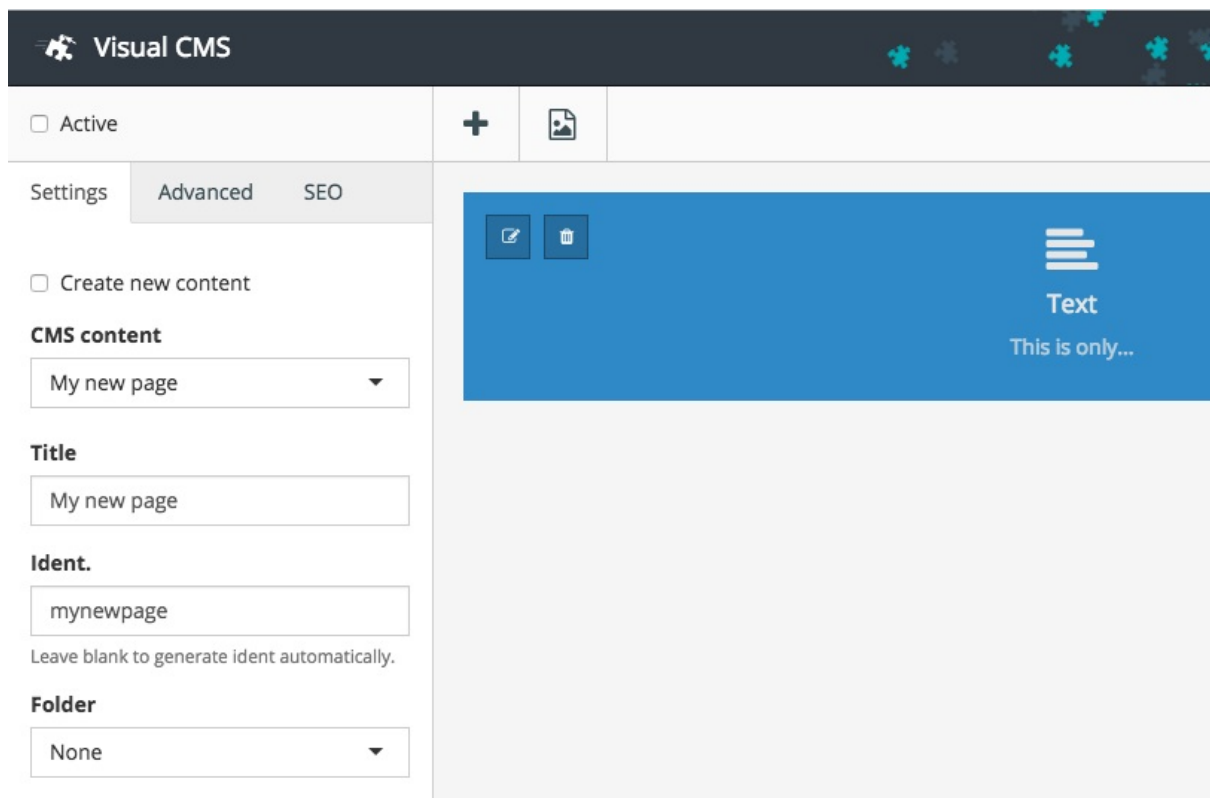
Searching for Content



It is just as easy to search for existing pages:

1. Click into field **CMS-Content**
2. Enter your search term
3. Choose your content from the Live search

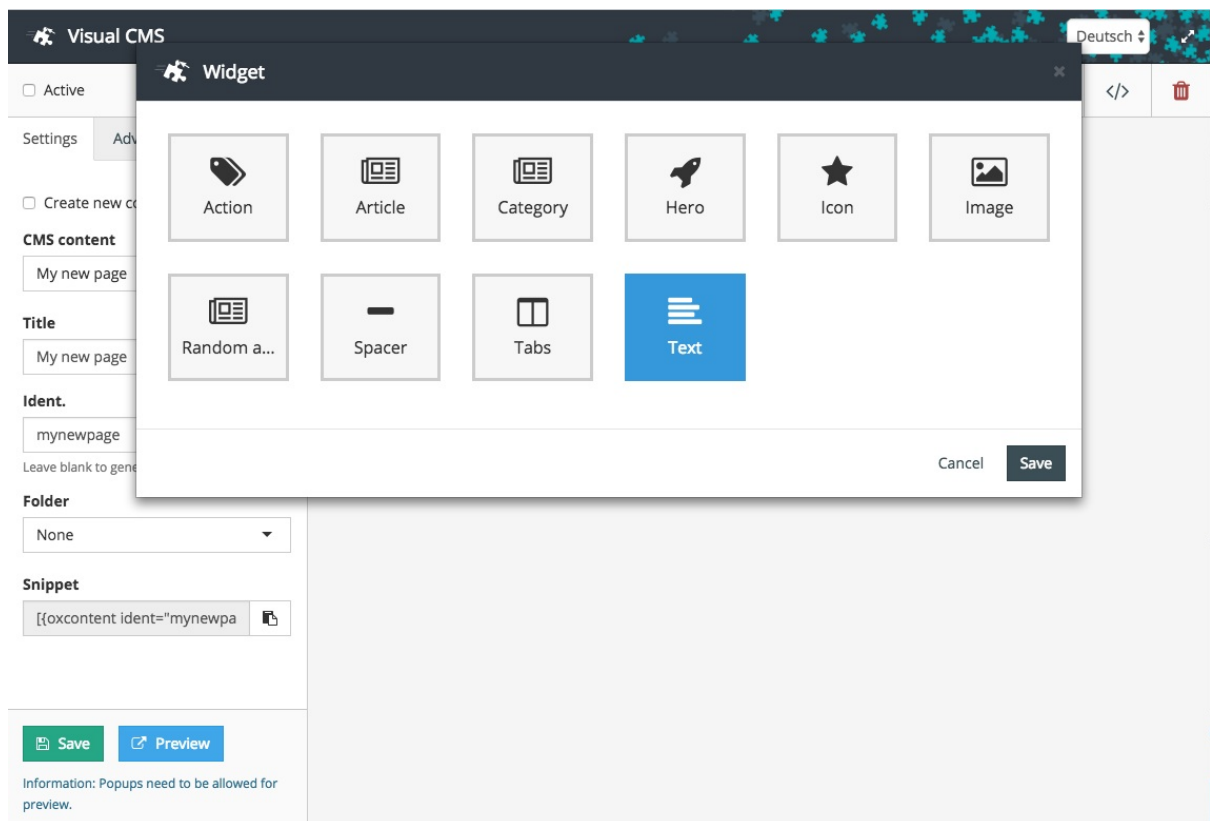
Editing a Widget



Widgets can easily be edited:

1. Choose the page you wish to edit
2. Use your mouse to hover over the widget you wish to edit
3. Click on the edit symbol
4. Edit the widget
5. Save the changes to the widget
6. Save the CMS content

Adding Further Widgets



1. Choose the content you wish to edit
2. Click on Button **Add widget**
3. Choose a widget
4. Edit options and content
5. Save the widget

Developer Information

The Visual CMS can be extended by adding further widgets.

The editor creates short codes from all widgets (similar to BBCode), which it then saves as content. In the front end these short codes are parsed again and the corresponding classes are triggered.

Even the usage of custom grid systems is possible. As default, the module uses a slimmed down version of the Bootstrap grid system.

- [Theme Integration](#)
- [Widget Creation](#)
- [Custom Gridsystem](#)
- [Widget-Modul Migration from OXID eShop v5 to v6](#)

Theme Integration

The module is completely compatible with Azure, Flow and with RoxIVE themes. For other themes, minor adaptations might be necessary.

Content Pages

When needed, the two themes *content.tpl* and *content_plain.tpl* must be extended by two requests:

content.tpl

```

[[capture append="oxidBlock_content"]]
  [[assign var="oContent" value=$oView->getContent()]]
  [[assign var="tpl" value=$oViewConf->getActTplName()]]
  [[assign var="oxloadid" value=$oViewConf->getActContentLoadId()]]

  [[* Customisation: Check the display of the heading *]]
  [[if !$oContent->oxcontents__ddhidetitle->value]]
    <h1 class="pageHead">[{$oView->getTitle()}]</h1>
  [[/if]]

  <div class="cmsContent">
    [{$oView->getParsedContent()}]
  </div>
[[/capture]]

[[* Customisation: check the display of the side bar *]]
[[if $oContent->oxcontents__ddhidesidebar->value]]
  [[include file="layout/page.tpl"]]
[[else]]
  [[include file="layout/page.tpl" sidebar="Left"]]
[[/if]]

```

content_plain.tpl

```

[[capture append="oxidBlock_content"]]
  [[assign var="oContent" value=$oView->getContent()]]
  [[assign var="tpl" value=$oViewConf->getActTplName()]]
  [[assign var="oxloadid" value=$oViewConf->getActContentLoadId()]]

  [[* Customisation: Check the display of the heading *]]
  [[if !$oContent->oxcontents__ddhidetitle->value]]
    <h1 class="pageHead">[{$oView->getTitle()}]</h1>
  [[/if]]

  [{$oView->getParsedContent()}]
[[/capture]]
[[include file="layout/popup.tpl"]]

```


Widget Creation

Widgets can be created through the creation of short-code classes. The classes must be added to one of the following directories:

- *modules/ddoe/visualcms/Core/shortcodes/*
- *modules/*/visualcms/shortcodes/*

These Modules have to be activated and the name of the file is at the same time the class-prefix.

Following Namespace is always needed:

```
use OxidEsaales\VisualCmsModule\Application\Model\VisualEditorShortcode;
```

In the following Example, these Namespaces are also needed:

```
use OxidEsaales\Eshop\Core\Registry;  
use OxidEsaales\Eshop\Core\DatabaseProvider;  
use OxidEsaales\Eshop\Application\Component\Widget\ArticleBox;  
use OxidEsaales\Eshop\Application\Model\Article;  
use OxidEsaales\Eshop\Application\Model\ArticleList;
```

A short-code class is structured as follows:

```
class article_shortcode extends VisualEditorShortcode  
{
```

- The class name consists of a file name (without a file extension) and the suffix "_shortcode".
- The class should always extend the class VisualEditorShortcode, so that standard attributes and methods are made available.

Attributes

Next, attributes can be defined:

```
protected $_sTitle = 'DD_VISUAL_EDITOR_SHORTCODE_ARTICLE';  
protected $_sBackgroundColor = '#e74c3c';  
protected $_sIcon = 'fa-newspaper-o';
```

- **\$_sTitle:** Title of the widget (Lang-String)
- **\$_sBackgroundColor:** Color of the widget in the back end
- **\$_sIcon:** CSS-class for the icon of the widget in the back end (s.


```
http://fontawesome.io/icons/ )
```

- **\$_sShortCode**: Short-code name
- **\$_aOptions**: contains the widget options

All attributes can also be set using the set-methods (e.g.: setShortCode())

install()-Method

- The **install()** method is called by the back end to initialize the widget.
- The short-code classes extend the OXID class **FrontendController** (**oxUBase** in OXID v5). OXID classes and methods are therefore available.
- In this example, the file name (without file extension) is set as short-code name.
- Subsequently, the widget options and entry fields are set in the back end.

```
public function install()
{
    $this->setShortCode( basename( __FILE__, '.php' ) );

    $oLang = Registry::getLang();

    $this->setOptions(
        array(
            'id' => array(
                // specifies the method used in live search
                'data' => 'searchArticle',
                // possible types: select, text, color, file, multi, textarea, wysiwyg, hidden
                'type' => 'select',
                // Label Description
                'label' => $oLang->translateString( 'DD_VISUAL_EDITOR_WIDGET_ARTICLE' ),
                // placeholder description
                'placeholder' => $oLang->translateString( 'DD_VISUAL_EDITOR_WIDGET_CHOOSE_ARTICLE' ),
                // fields which also be considered in a selection (only type "select")
                'dataFields' => array(
                    // the field "name" returns value "label" to the live search
                    'name' => 'label'
                )
            ),
            'name' => array(
                // hidden field
                'type' => 'hidden',
                // the value is used in previews of widget listings
                'preview' => true
            )
        )
    );
}
```

parse()-Method

- The **parse()** method is called when the short-code is parsed in the front end.
- The parameter **\$sContent** is a reserved parameter of the widget option **content**.
- All other values are passed as an array in the second parameter.
- In this example, products are loaded using the product ID and subsequently passed to the Smarty function **oxid_include_widget** so that a product widget is displayed.
- The return value corresponds to the content delivered to the front end.

```
public function parse( $sContent = '', $aParams = array() )
{
    /** @var Article $oArticle */
    $oArticle = oxNew( Article::class );
    $oArticle->load( $aParams[ 'id' ] );

    $sOutput = '<div class="dd-shortcode-' . $this->getShortCode() . ' productData product
Box' . ( $aParams[ 'class' ] ? ' ' . $aParams[ 'class' ] : ' ' ) . '>';
    $sOutput .= '[{oxid_include_widget cl="oxwArticleBox" _parent=$oView->getClassName() _
navurlparams=$oViewConf->getNavUrlParams() anid="' . $aParams[ 'id' ] . '" isVatIncluded=$
oView->isVatIncluded() nocookie=1 sWidgetType=product sListType="listitem_' . $sType . '"
inlist=1 skipESIforUser=1}]';
    $sOutput .= '</div>';

    return $sOutput;
}
```

Alternative: Smarty Template

Alternatively, a Smarty template can be called instead of delivering the content directly:

```
public function parse( $sContent = '', $aParams = array() )
{
    /** @var Article $oArticle */
    $oArticle = oxNew( Article::class );
    $oArticle->load( $aParams[ 'id' ] );

    $oSmarty = Registry::get( 'oxUtilsView' )->getSmarty();
    $oSmarty->assign(
        array(
            'oView' => $this->getConfig()->getTopActiveView(),
            'shortcode' => $this->getShortCode(),
        )
    );
    $sOutput .= $oSmarty->fetch( 'ddoe_widget_article.tpl' );
    return $sOutput;
}
```

The two methods and the attributes mentioned above are all that is needed in order to create a widget or short-code. All else depends on the complexity of the widget.

All provided widgets in the module folder of VisualCMS under *Core/shortcodes* are open source and can be used as examples.

Custom Grid System

In order to use your own custom grid system, only a few changes in the settings are necessary.

1. Please navigate in the admin area of the eShop to **Extensions -> Modules -> Visual CMS -> Settings**
2. Activate the checkbox **Use custom grid**. This checkbox is located under the point **Front end**
3. Enter the prefixes of CSS-classes for your grid system
4. Enter the maximum number of columns of the grid system
5. Save

After that, offsets and column widths can no longer be adjusted by "Layout settings" of the widget, but must be made by dragging and dropping them on the CMS page.

Examples

Foundation

▼ Front end

Use custom grid (deactivates Bootstrap)

columns small-

Column prefix for CSS class (only for custom grid)

small-offset-

Column offset prefix CSS class (only for custom grid)

row

Row CSS class (only for custom grid)

Convert row numbers into words [1 = one] (only for custom grid)

12

Grid size / columns per row (recommended: 12)

960 Grid System

▼ **Front end**

Use custom grid (deactivates Bootstrap)

Column prefix for CSS class (only for custom grid)

Column offset prefix CSS class (only for custom grid)

Row CSS class (only for custom grid)

Convert row numbers into words [1 = one] (only for custom grid)

Grid size / columns per row (recommended: 12)

The grid system of your choice has to have been included into the theme.

Widget-Modul Migration from OXID eShop v5 to v6

- All module files have to be UTF-8 encoded.
- Code must work with PHP 5.6 or higher.
- The following namespace must also be specified in modules from now on:

```
use OxidEsaales\VisualCmsModule\Application\Model\VisualEditorShortcode;
```

Note: Widgets may not be assigned to a namespace!

- So that standard attributes and methods can be adopted, the class is no longer derived from `ddvisualaeditor_shortcode`, but from `VisualEditorShortcode`.

```
class own_shortcode extends VisualEditorShortcode
```

Help & Support

Do you have any questions or do you need any help for the installation?

Please contact our support: <http://www.oxid-esales.com/en/support-services.html>