Sitecore Experience Accelerator 1.0

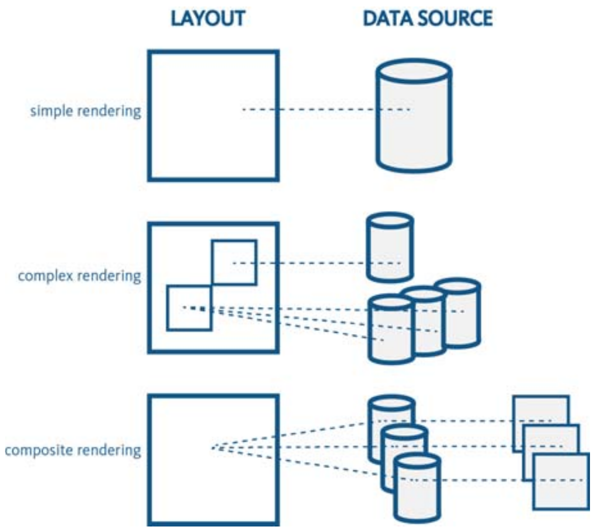
All the official Sitecore documentation.

sitecore®
Own the experience™

Composite renderings

A composite rendering consists of several renderings. Each instance can have its own layout and it can be designed separately in Experience Editor. When a composite rendering is rendered on the page, it queries each item from its data source and pulls data and layout. This facilitates building very complex layouts, but also involves a more advanced setup.

Simple renderings, such as Image, Text, and Video, consist of a layout and use single data source items to provide the content. Complex renderings, such as a container or a splitter, contain multiple simple or complex renderings. These renderings have one layout definition and all added renderings are defined in a renderings page property. Composite renderings, such as Accordion, Carousel, and Toggle, use all data sources with their own layout. In the Renderings field, only references to data source items are stored. Therefore, related layouts are not rendered out-of-the-box.



By default, SXA contains four composite renderings:

- Carousel: The Carousel rendering lets you define a set of rotating slides. You can use a Carousel to display a number of featured pages, link to top stories, show client logos, testimonials, or pictures. The separate items appear one at a time, in a defined order.
- Accordion: The Accordion rendering lets you stack a list of items vertically or horizontally. You can use an Accordion to show FAQs, sets of thumbnails, or divide longer documents in sections. Each item can be expanded to reveal the content associated with that item.
- Flip: The Flip rendering lets you display two slides that are shown alternatively after a visitor either clicks or hovers over it.
- Tabs: The Tabs rendering lets you display content within a tabbed interface, where the content of only one tab is visible at a time. You can, for example, structure the product page to show only one information type (Overview, Specifications, Resources, Availability) at a time.

Send feedback about the documentation to docsite@sitecore.net.

Create a rendering variant

SXA comes with a set of default renderings and rendering variants. Rendering variants are configurable adaptations of the default renderings. To further encourage reusability, designers and front-end developers can also create new rendering variants. This gives authors more options in the way they present their content.

You can create your own variation of a rendering by adding a variant in the Content Editor.

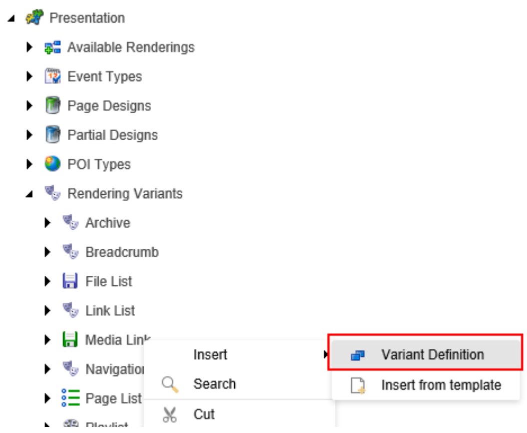
To create a rendering variant:

1. In the Content Editor, click the site and open the *Presentation/Rendering Variants* folder. This folder lists all the renderings that allow variants.

   Note

   To add a rendering to the folder, contact your Administrator.

2. Right-click the rendering that you want to add the variant to, and then click Insert, Variant Definition.

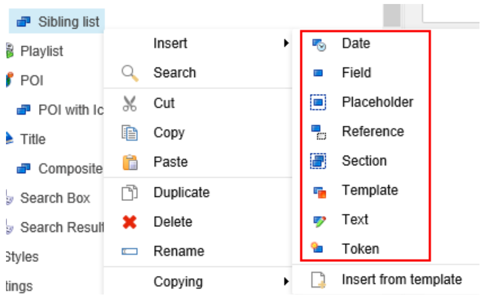


3. Enter a name and click OK.
4. In the Variant Details section, specify information in the following fields:
    - Css Class: specify the class name for the variant in the HTML.
    - Item CSS class: specify the CSS class for list items. For example, for variants of `PageList`, `LinkList`, `FileList`, and `Navigation` renderings.
    - Field used as link target: provide the field name of the target item. This class is added to the list and navigation items markup (`li` HTML element). This link is used to override all links when the Is link, Is prefix link, or Is suffix link check boxes are selected.
    - Allowed in templates: specify the pages that the variant is available for. Click the relevant page template, click the right arrow to move it to the list of selected items, and then click Save. If there are no templates selected, the variant is available for all pages.
5. To add child items to the rendering variant, right-click the variant, click Insert, and then click the relevant item.
    - Date: displays data and time in custom format.
    - Field: displays field name and other field values.
    - Placeholder: allows to render an empty placeholder.
    - Reference: displays field from referenced item.
    - Section: used to create groups. For example, if you want to render two div elements, you can create two section items that each contain one of the two div elements.
    - Template: allows user to define NVelocity template which will be used to render part of the HTML.

      Note

      NVelocity is a .Net-based template engine. It permits developers to use the simple yet powerful template language to reference objects defined in .Net code.

    - Text: displays text.
    - Token: supports tokens `$Size` and `$FileTypeIcon`.

    Depending on the item you add, you can set the following fields:

Field	Variant details
Tag	HTML tag in which the field content will be rendered. For example: `div` If left empty field content will remain unwrapped.
Field name	Name of the field current item.
Prefix	Adds string value as a prefix.
Suffix	Adds string value as a suffix.
Is link	Select to have hyperlinks that wrap the field content.
Is prefix link	Select to wrap prefix in the same link which you use for the field content.
Is suffix link	Select to wrap the suffix with a hyperlink.
Is download link	Select to have a hyperlink with a download attribute.
Css Class	Adds Css class to the tag.
Is editable	Select to edit rendered field.
Date format	Determines the date format.
Render if empty	Select to render empty element when the field is empty.
Pass through field	Defines the name of the field from the nested item.
Text	The text to render.

Template	Define the NVelocity template that renders part of the component variant HTML. You can use the following objects: `$item`: access to the current item (`$item.Name` displays current item name). `$dateTool.Format(date, format)`: formats date and time values. `$numberTool.Format(number, format)`: formats numbers. `$geospatial`: in case of geospatial search (`$geospatial.Distance`) will show distance to certain point of interest).
Token	Use special tokens to format certain field values. Supported tokens are: `$Size`: displays size of a file depending on size unit. `$FileTypeIcon`: displays icon depending on file extension.

Send feedback about the documentation to docsite@sitecore.net.

The SXA renderings and rendering variants

In Sitecore Experience Accelerator (SXA), you can construct your pages by dragging and dropping standard components directly where you need them. These components are called renderings and they are listed in the Toolbox in the Experience Editor. There are renderings available for simple text, images, videos, social media plugins, and so on.

There is a set of out-of-the-box variants that come with renderings that support them. When you place a rendering supporting rendering variants on a page, you can select one of its variants from a dropdown below the Experience Buttons Toolbar. Variants may make a rendering appear differently or may make them show different content. For example, the list rendering can have different variants for: detailed lists, thumbnails list, and a carousel.

You can configure the default renderings and create rendering variants in the Content Editor.

This following tables display an overview of the renderings and rendering variants options available in the Toolbox:

- Composites
- Context
- Events
- Forms
- Maps
- Media
- Navigation
- Page content
- Page structure
- Search
- Security
- Social
- Taxonomy

Composites

Rendering	Variant	Description
Accordion	Embedded renderings can support variants	Displays collapsible panels.
Carousel	Embedded renderings can support variants	Displays set of slides that can contain images, videos, links, and text.
Flip	Embedded renderings can support variants	Displays two sides that both have a title and a content renderings.
Tabs	Embedded renderings can support variants	Adds a tabs to the page.

Context

Rendering	Description
Language Selector	Provides a link to switch between different language versions.
Site Selector	Provides a link to switch between different tenant sites.

Events

Rendering	Description
Event Calendar	Displays events from event lists or a Google calendar.
Event List	Displays lists of events with name, description, place, start/end time, and link.

Forms

Rendering	Description
Form Wrapper	Embeds a form created using the Webforms for Marketers module.

Maps

Rendering	Description
Map	Embeds maps from Google or Bing with locations, routes, and areas that you can mark. The component can also display POI results when associated with a search results source. Points on a map can be formatted with rendering variants.

Media

Rendering	Description
File List	Displays list of files available for download. Supports rendering variants.
Flash	Embeds a SWF object on the page.
Gallery	Displays a gallery of images and/or videos.
Image	Adds an image that you select from the Media library to a page.
Image (Reusable)	Stores image from the Media library to enable reusability.
Media Link	Provides a rich link to an asset in the Media library. Includes description and preview. Supports rendering variants.
Playlist	Enables you to create a playlist for the Video component. Supports rendering variants.
Video	Displays an HTML 5 player (with Flash fallback for legacy browsers) to play videos.

Navigation

Rendering	Description
Archive	Displays blog archives in a tree. Supports rendering variants.
Breadcrumb	Generates a breadcrumb that lists all path items from the root to the current item. Supports rendering variants.
Link	Adds a link to: • sibling or parent page • page from browsing history • arbitrary page or resource
Link List	Displays lists with items that contain a title, link, and text.
Navigation	Generates a navigation menu. You can select:

Page Content

Rendering	Variants	Description
Field Editor	yes	Enables you to edit content fields directly in the Experience Editor.
Page Content	yes	Displays the specific fields on a page from the data source item that you select.
Page List	yes	Displays lists of pages by predefined or composed queries.
Pagination	no	Adds pagination for the Page list rendering.
Plain HTML	no	Embeds HTML code.
Plain HTML (Reusable)	no	Stores HTML code to enable reusability.
Promo	yes	Renders a field from another page and works as a placeholder for other renderings. It consists of an icon or a title and links.
Rich Text	no	Adds formatted text on a page. You can write text using HTML tags.
Rich Text (Reusable)	no	Stores rich text to enable reusability.
Title	yes	Displays the title or subtitle of the current page.

Page Structure

Rendering	Description
Container	Adds additional CSS styling using a wrapper for other renderings.
Divider	Divides a placeholder horizontally.
Edit Mode Panel	Creates a placeholder to embed other renderings that are visible to authors in Edit mode only.
IFrame	Embeds external pages.
Splitter (Columns)	Splits a placeholder horizontally.
Splitter (Rows)	Splits a placeholder vertically.
Toggle	Displays buttons or links that show additional content when toggled.

Search

Rendering	Variants	Description
Filter (Checklist)	no	Filters search results based on selected values.
Filter (Date)	no	Filters the search results on selected date/time.
Filter (Dropdown)	no	Filters search results based on items that are tagged with a selected facet.
Filter (Managed Range)	no	Specifies the result with a minimum and maximum.

Filter (Radius)	no	Filters the search results based on distance from current user location or location provided in the Location Filter rendering.
Filter (Range Slider)	no	Filters the search results based on a facet to be less, more, or equal to the value selected by a visitor.
Filter (Slider)	no	Filters search results based on a specific facet within the selected range.
Load More	no	Loads search results progressively.
Location Finder	no	Specifies user location.
Page Selector	no	Provides paging functionality for search results. This rendering is mutually exclusive with the Load More rendering.
Page Size	no	Enables visitors to switch the number of results displayed once.
Results Count	no	Enables you to indicate the number of results available to the visitor.
Results Variant Selector	no	Enables visitors to change the rendering variant that is used dynamically by the Search Results rendering
Search Box	yes	Filters the search results based on text provided by a visitor.
Search Results	yes	Displays the results of a search query.
Sort Results	no	Enables users to switch the sorting criteria for search results.

Security

Rendering	Description
Login	Displays a login form.
Logout	Displays a logout button.
Social Login Wrapper	Embeds a Social Connected module login form on the page.

Social

Rendering	Description
Feed	Displays RSS and ATOM feeds.
Social Media share	Displays social network links (Facebook, Twitter, Google+).

Taxonomy

Rendering	Description
Tag Cloud	Displays the aggregated tags for the search results, visually weighted based on their occurrence frequency.
Tag List	Displays taxonomy entries that a page is tagged with.

Send feedback about the documentation to docsite@sitecore.net.

The HTML structure of pages and renderings

SXA separates structure (HTML) from design (CSS) to make it easier to change the design of websites. To make this possible, SXA provides a stable well-structured HTML code that is the same for every page. Users can apply different styling without changing the underlying code.

This topic describes the basic structure for:

- Page HTML
- Rendering HTML
- JavaScript
- CSS

Page HTML

All SXA pages use the following layout structure:

```
<html>

<head>

  <!-- meta renderings placeholder "head" -->

</head>

<body>

  <!-- meta renderings placeholder "body-top" -->

  <div id="wrapper">

    <div id="header" class="main clearfix">

      <!-- header components -->

    </div>

    <div id="content" class="main clearfix">

      <!-- content components -->

    </div>

    <div id="footer" class="main clearfix">

      <!-- footer components -->

    </div>

  </div>

  <!-- meta renderings placeholder "body-bottom" -->

</body>
```

All regular renderings can be placed in the following containers:

- Header
- Content
- Footer

Meta renderings can be placed on Meta Partial Designs in the following containers:

- Head
- Body-top
- Body-bottom

Designers can use page splitters to generate additional columns or rows inside the header, content, or footer containers.

There are two types of splitters:

- Column splitters – generates `divs` with proper grid classes wrapped by the row container. Grid values specify the column widths.

  ```
  <div class='row'>

    <div class='grid-6'></div>

    <div class='grid-6'></div>

  </div>
  ```

- Row splitters – generates empty `row divs` that fill the full width of the available parent container.

  ```
  <div class='row'></div>

  <div class='row'></div>

  <div class='row'></div>
  ```

You can add classes for both columns and rows. You can also mark specific splitter sections and style them differently than the other sections. This can be useful for styling a page part that breaks the grid system. However, you can only override grid behavior for specific rows.

Rendering HTML

All SXA renderings are designed to be easily styled. Because the HTML is very standard, it is easy for theme developers to apply CSS and JavaScript.

The following example uses the Accordion rendering. The HTML structure for renderings is wrapped by `div` with the component class and the component name accordion. You can add CSS class variants for styling purposes.

```
<div class="component accordion {custom classes}" data-properties='{"expandOnHover":false,"expandedByDefault":false,"speed":5000,

"easing":"easeInOutBounce","canOpenMultiple":false,"canToggle":false}'

>
```

```
  <div class="component-content">
    <div>
      <ul class="items">
        <li class="item">
          <div class="toggle-header">
            <div class="label">
              Header content
            </div>
          </div>
          <div class="toggle-content">
              Section content
          </div>
        </li>       </ul>
    </div>
  </div>
</div>
```

Note

Use classes for styling. Do not use IDs. Div IDs are used by Sitecore and cannot be changed.

The following properties are used by JavaScript to control rendering behavior:

- *expandedByDefault* – first accordion tab is visible.
- *expandOnHover* – expand tab on mouse enter and close on mouse leave events (the *canOpenMultiple* property is not used in this case).
- *canOpenMultiple* – open multiple tabs at the same time (the canToggle property is always active and cannot be disabled).
- *speed* and *easing* – can be used everywhere to define transition method and time.

Every SXA rendering contains the same wrapping structure:

```
<div class='component <component-name>'
  <div class='component-content'>
      <!-- component html -->
  </div>
</div>
```

The code inside will be different for each rendering. Often, there will be an additional wrapping div with a unique ID that describes the specific rendering used by Sitecore. CSS/JS scripts should not use these IDs.

Note

The inner rendering structure uses clean markup with dash-separated class convention. There are a few exceptions: some elements, such as forms, use additional Sitecore modules (for example, Web Forms for Marketers) and generate HTML that looks different (camelCase class names, and tables in some cases). You cannot modify this HTML.

Every HTML rendering is part of a platform and you cannot change it for a single project or site, except where the HTML is shaped by rendering variants.

JavaScript

Complex renderings have their own JavaScript. These scripts are located in the JavaScript framework in the main themes folder. The framework provides public methods, such as register and init, to register renderings and additional helpers, such as cookies.

All back-end properties used by JavaScript are placed in the data-properties attribute inside the rendering wrapping div. They are encoded in JSON format.

```
<div class='component accordion' data-properties='{"expandOnHover":false,"expandedByDefault":false,"speed":5000,
"easing":"easeInOutBounce","canOpenMultiple":false,"canToggle":false}'
</div>
```

The following is a clipped version of the JavaScript framework:

```
var ZG = (function ($, document) {
    var api = {},
        modules = {};
    /*
     * Register new module
     * @params name - name of the module
     * @params api - API object of the module
     * @params init - init function for module, if not defined api.init will be used
     */
    api.register = function (name, api, init) {
        modules[name] = {
            name: name,
            api: api,
            init: init || api.init || function () {}
        };
    };
    var initScheduled = false;
    /*
     * Initializes all registered modules
     */
    api.init = function () {
        if (!initScheduled) {
            initScheduled = true;
            ZG.ready(function () {
                try {
                    for (var name in modules) if (modules.hasOwnProperty(name)) {
                        modules[name].init();
                    }
                } finally {
                    initScheduled = false;
                }
            });
        }
    };
    /*
     * Wrapper around $(document).ready - fires given function when (or if) document is ready
     */
    api.ready = function (fn) {
        $(document).ready(fn);
    };
    return api;
})($, document);
```

CSS

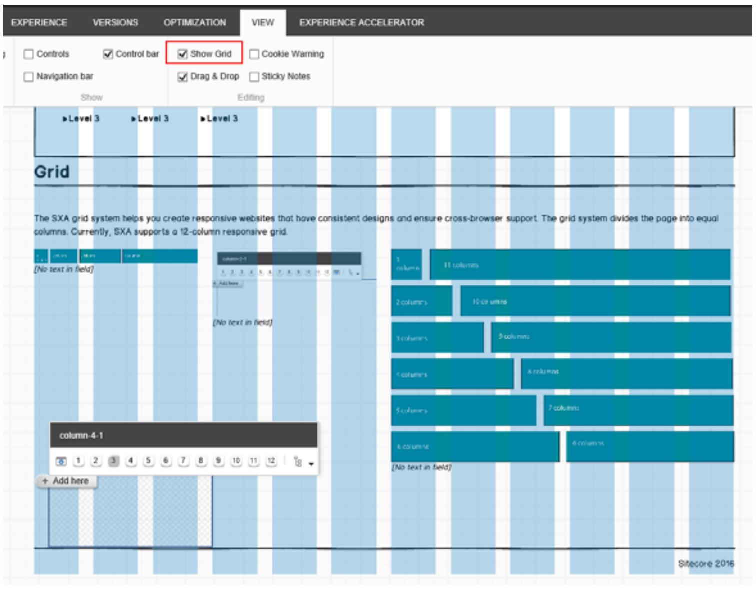
You can replace a single class to change rendering behavior, for example, to change from standard navigation to mobile navigation. You can also create your own rendering variant by adding CSS classes and add styling and JavaScript functionality. You must adhere to naming conventions. Classes for specific renderings always start with the name of rendering. Add words that explain the functionality of the new class using the dash name convention. For example: `.navigation-mobile`, `.navigation-main-horizontal`, `.navigation-sidebar`.

Send feedback about the documentation to docsite@sitecore.net.

The SXA grid layout

The SXA grid system helps you create responsive websites that have consistent designs and ensure cross-browser support. The grid system divides the page into equal columns. Currently, SXA supports a 12-column responsive grid.

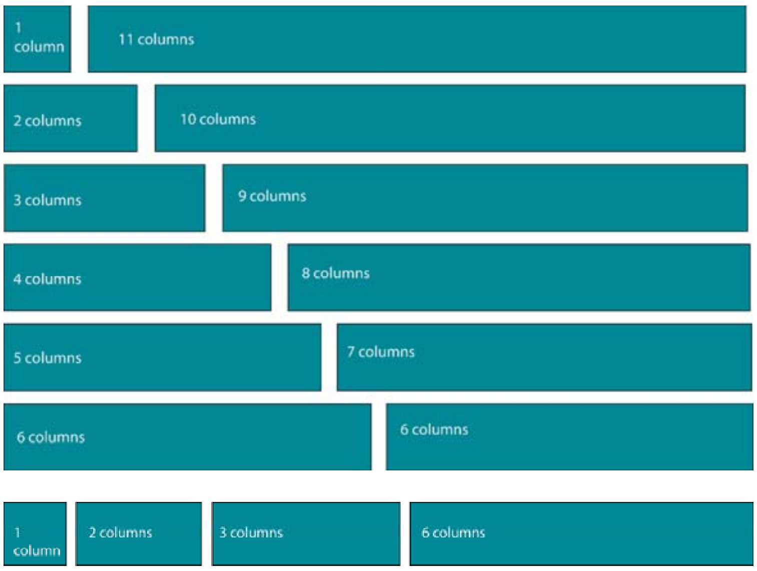
To make the grid columns visible on a page, on the ribbon, on the View tab, in the Show section, select Show Grid.



The SXA grid system helps you create responsive websites that have consistent designs and ensure cross-browser support. The grid system divides the page into equal columns. Currently, SXA supports a 12-column responsive grid.

The 12-column grid is 960 pixels wide. You can choose how you want to divide your columns on the page. By using the SXA grid system, you can achieve visual consistency because the elements are always vertically aligned to one of the columns. On every SXA page, you can use row and column splitters to divide the available space both horizontally and vertically and create the page structure you need.

For example:

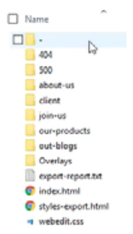


Send feedback about the documentation to docsite@sitecore.net.

Change a site design using Creative Exchange

When a site is being developed, Creative Exchange enables teams to work together on a site simultaneously. A ZIP file exported using Creative Exchange contains the content, structure, assets, and theme of the site.

You can download this ZIP file and work on the design of the site from your local folder before exporting it back to the team. This lets content authors work on content while the designers make changes to the look and feel of the site.



When you have unzipped the site from Creative Exchange, you can make the following changes:

- Add and modify images and files within the *Media Library* folder.
- Add classes on nodes that have: `<!—- add your css classes here -->">`

- Change images in renderings.

Note

Changes to the HTML structure, deleting existing classes, changing text content, and changing base theme folders will not be imported back to the system and can damage your site.

This topic describes how to:

- Change an image
- Change the layout
- Change the styling of text renderings

Change an image

You can change the images used in Image and Image Variant renderings by linking to a new image.

To change the image of an Image rendering:

- In the `index.html` file, change the asset in the `<img>` tag. For example:

```
<div class="component image logo">

    <div class="component-content">

        <a href="index.html"><img src="-/media/Themes/SXA/Images/coglogo.png" alt="coglogo" width="206" height="81" /></a>

    </div>

</div>
```

Make sure that the image either exists in the CMS or provide it in the ZIP file (in the *Media Library* folder) that is imported back into the system. For rendering variants, images are listed as data attributes `data-ceitem` and `data-cefield`. For example:

```
<img src="-/media/showcase-group/wireframe-mvc/int/003.png" alt="my third image" data-ceitem="{590D99E0-9098-47FF-8CBB-6F7FCFD5CB12}"
```

If the image that you want to change is used on multiple pages, you must change all of them in the code. Otherwise the engine will not import the image. You can also add #important to the URL to force the change for the image on all pages:

```
<img src="-/media/showcase-group/wireframe-mvc/int/003.png#important" alt="my third image" data-ceitem="{590D99E0-9098-47FF-8CBB-6F7FC
```

Change the layout

You can assign classes to style the layout or a rendering in the JS or CSS file under `/media/themes/`.

To style a rendering:

- Find the rendering that you want to change and add your class:

```
<div class="component component-name  {source-item-guid} {unique-rendering-instance-guid} {Styles} add-your-css-classes-here <!–-

  <div class="component-content">

      <!-- component specific mark-up -->

  </div>

</div>
```

Note

For clean CSS class structure, use lowercase, and a dash to join multiple words. Additional classes for renderings should start with the name of the rendering followed by a name that describes the functionality. For example:

```
<div class="component carousel {guid} {guid} {Styles} carousel-homepage">
```

or:

```
<div class="component navigation {guid} {guid} {Styles} navigation-main-horizontal">.
```

SXA pages are divided into rows and columns with splitters. Splitters can have their own list of classes.

To style the layout:

- Find the row splitter and add your class.

```
<div class="row {guid} {guid} {Styles1} <!–- ADD YOUR CSS CLASSES HERE -->">

    <!-- components mark-up -->

</div>

<div class="row {guid} {guid} {Styles2} <!–- ADD YOUR CSS CLASSES HERE -->">

    <!-- components mark-up -->

</div>
```

Note

Additional classes for row and column splitters should start with the `column` or `row` prefix, followed by a name that describes the functionality. For example: `<div class="row row-logo">`

- Find the column splitter and add your class.

```
<div class="alpha grid-3 {guid} {guid} {Styles1} <!–- ADD YOUR CSS CLASSES HERE -->">

    <!-- components mark-up -->
</div>
<div class="omega grid-5 {guid} {guid} {Styles2} <!—- ADD YOUR CSS CLASSES HERE -->">
    <!-- components mark-up -->
</div>
```

Note

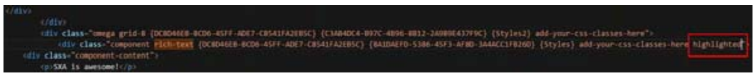
Classes that you added and imported back into SXA are available for use on other instances.

Change the styling of text renderings

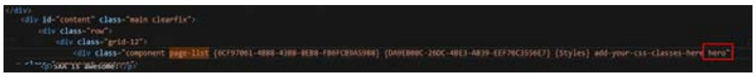
You can change the styling of the text renderings by adding a class.

For example, if you want to change the styling of a Rich Text rendering and the Page List rendering:

1. Open the `index.html` file and navigate to the instance of the rendering.
2. Add the class. For example, add the classes `highlighted` and `hero`:

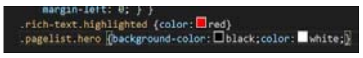


```
</div>
    </div>
    <div class="omega grid-8 {DC8D46EB-8CD6-45FF-ADE7-CB541FA2EB5C} {C3AB4DC4-B97C-4B96-8B12-2A9B0E437F9C} {Styles2} add-your-css-classes-here">
        <div class="component rich-text {DC8D46EB-8CD6-45FF-ADE7-CB541FA2EB5C} {8A1DAEFD-5386-45F3-AF8D-3A4ACC1FB260} {Styles} add-your-css-classes-here highlighted">
  <div class="component-content">
    <p>SXA is awesome!</p>
```



```
</div>
  <div id="content" class="main clearfix">
    <div class="row">
      <div class="grid-12">
        <div class="component page-list {6CF97061-4888-4388-8EB8-FB6FCB9A5988} {DA9EB00C-260C-40E3-A839-EEF76C3556E7} {Styles} add-your-css-classes-here hero">
    <p>SXA is awesome!</p>
```

3. Add the new CSS class and the styling to the main `.css` file:



```
    margin-left: 0; } }
.rich-text.highlighted {color: 🟥red}
.pagelist.hero {background-color: ⬛black;color: ⬜white;}
```

Open the `index.html` file to preview your changes on the local instance of the site.

Send feedback about the documentation to docsite@sitecore.net.

Import and export a web design with Creative Exchange

Creative Exchange is an SXA feature that enables front-end developers to work on static HTML, CSS, and JS files. You can export an SXA site or even a single page to send to your designers while content editors can continue working on the site. The export process creates a ZIP file of a page or the whole site with its theme and content included. Designers can modify the files to modify the site's look and feel. Once the design is ready, you can use the tool to import it back into the site.

This topic outlines how to:

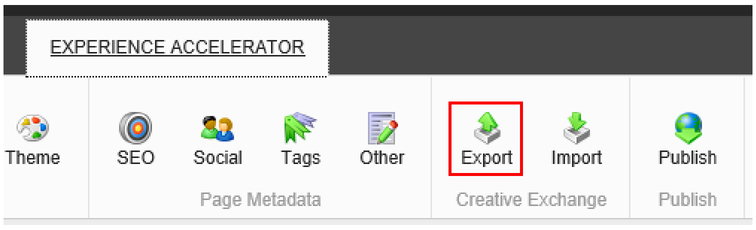
- Export a web design
- Import a web design

Export a web design

To make the site available for designers to work on, you can export the site's wireframes and content using Creative Exchange.

To export a web design using Creative Exchange:

1. In the Experience Editor, on the ribbon, on the Experience Accelerator tab, click Export.



2. In the Export settings dialog, edit the settings:

Field	Description
Device	Specify the device you want to export the site design to. If the device is not available, the layout may not be defined in Sitecore. Ask your Administrator for more information.
Exported Content	Specify what you want to export: *Current page* – exports only the page you are editing. *Current page with sub-pages* – exports the current page with all its descendants. *Whole site* – exports all pages from the current site.

	Note For large sites, you should export the pages of the site separately.
Site context	You can use the same page in different sites. Select the site for which you want to export the page.
Buckets	Specify that you want to export items from item buckets. Note Because item buckets can contain large number of pages, do not select this option unless it is explicitly required.
Mode	Select how you want to export the content: *Agency drop (preview mode, importable)* – exports the pages almost identically to those viewed by visitors, but with some additions to facilitate importing them back into the system. *Author mode (edit mode, importable)* – exports the pages from the Experience Editor containing additional markup that enables on-page editing. *End-user site (normal mode, non-importable)* –exports the pages identically to those viewed by visitors but cannot be imported back into your site.
Destination	Specify the destination for your exported content: *Folder on the server* – saves all the exported files directly to the server file system, where they can be managed by your revision control system. This can be convenient if you are working with external agencies and need to access the files through SVN or Git. *Zip package to download* – creates a package of compressed files. *Delete old packages before exporting* – deletes all previously created ZIP files from the server file system.
Maximum size of a single file in exported package	Enter a number to limit the size of zipped export files. This can be convenient when your site contains large assets, such as videos.

3. Click Next to start the export.

Depending on the destination you selected for your ZIP file, the file is available for download or stored somewhere in a folder on your server. You can send this ZIP file to designers to work on.

Note

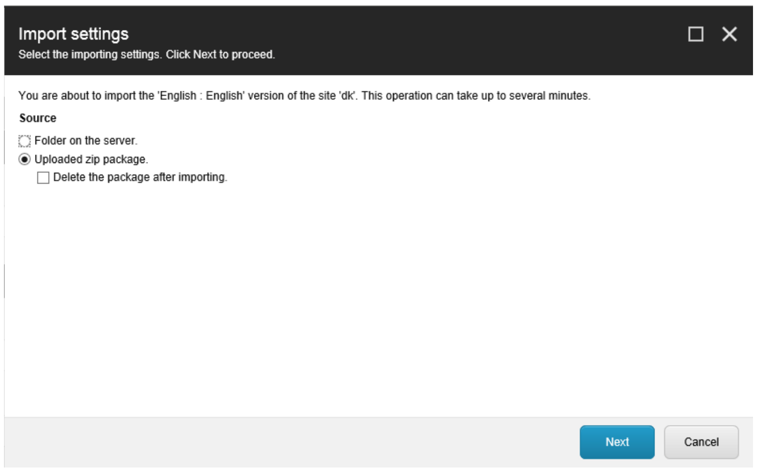
Possible reasons for your export to fail are: pages could not be rendered due to errors, linked assets are missing, or a Creative Exchange timeout.

Import a web design

Once the web design work is done, the folder can be zipped and imported back into the site using Creative Exchange.

To import a web design into the site:

1. In the Experience Editor, on the Experience Accelerator tab, click Import.
2. In the Import settings dialog box, select the source of the designs and click Next.


Import settings
Select the importing settings. Click Next to proceed.

You are about to import the 'English : English' version of the site 'dk'. This operation can take up to several minutes.

Source

Folder on the server. [✓ Uploaded zip package.]

Delete the package after importing.

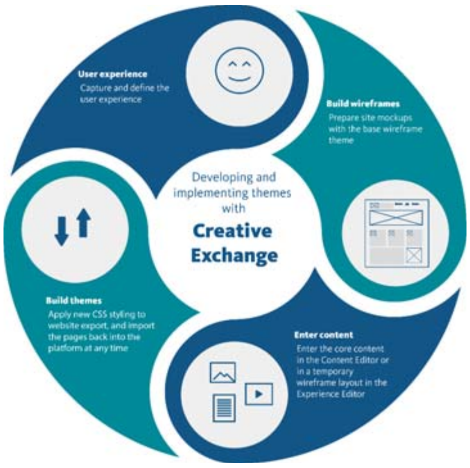
Depending on the source that you selected, select the folder or package, and then click Next. You receive a confirmation message when the import is complete.

Send feedback about the documentation to docsite@sitecore.net.

Working with Creative Exchange

The Creative Exchange process is designed to facilitate several different teams working on a website. For example, the team that is working on the theme of the site can work in parallel with other teams.

With Creative Exchange, you can export your site and produce a ZIP file with all the wireframes and all content. The creative agency or the in-house front-end developer can work with the static HTML offline. When the designs are ready, this ZIP file is sent back to be imported again, and the process can continue.


User experience
Capture and define the user experience
Build wireframes
Prepare site mockups with the base wireframe theme
Developing and implementing themes with
Creative Exchange
Build themes
Apply new CSS styling to website export, and import the pages back into the platform at any time
Enter content
Enter the core content in the Content Editor or in a temporary wireframe layout in the Experience Editor

The SXA base theme comes with core CSS and JavaScript. A creative agency or an in-house front-end developer can modify and extend the HTML classes and add CSS/JavaScriptS/assets using their preferred tools. For example, if you added a style to a particular rendering in the HTML, the import process will identify this new class and apply it to the system.

In SXA, any assets, such as images, fonts, and files that are normally referenced within the stylesheet, are considered to be part of the theme and they are imported with your CSS changes.

Send feedback about the documentation to docsite@sitecore.net.

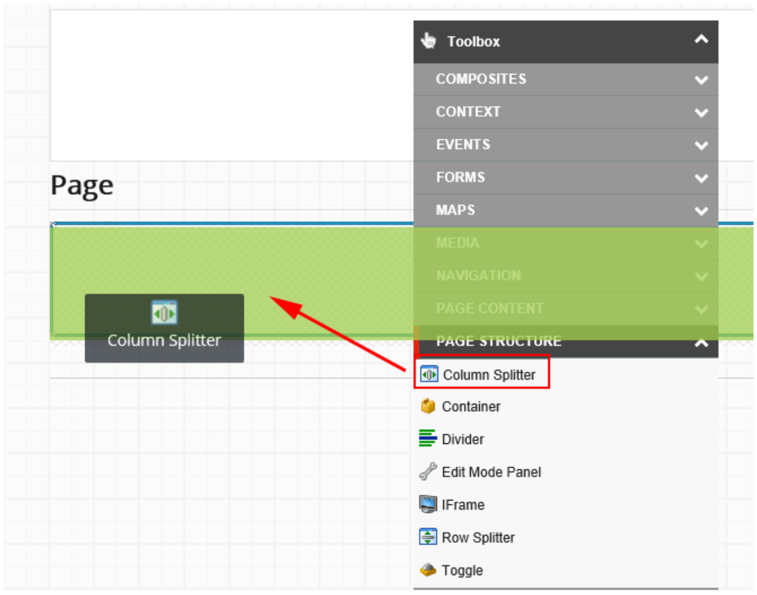
Change the layout of a page

SXA layouts use a responsive grid layout. The default grid divides pages into 12 equally wide columns. By using row and column splitters, you can determine the page structure and divide the available space horizontally and vertically.

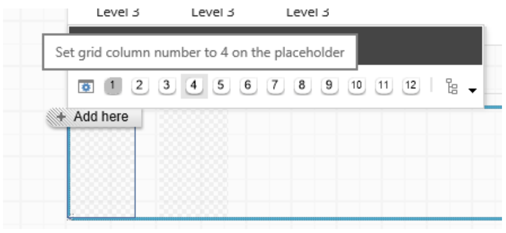
When you create a page in SXA, by default it includes the basic layout that contains three placeholders: header, main, and footer. You can use the splitters to turn this basic layout into the layout that you need.

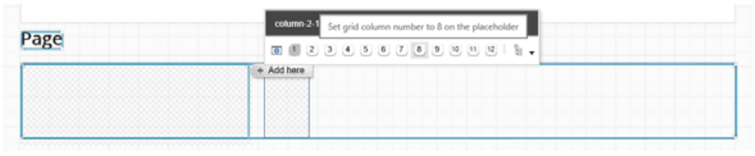
To use the Splitter (Column) to change the layout of the page:

1. On the page that you want to structure, from the Toolbox, drag the Splitter (Column) rendering to the page.

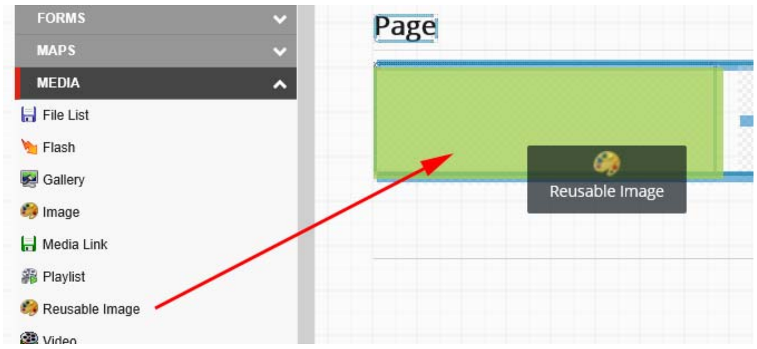


2. Use the floating toolbar to distribute the grid columns. For example, divide the placeholder into two groups: four columns on the left and eight on the right.

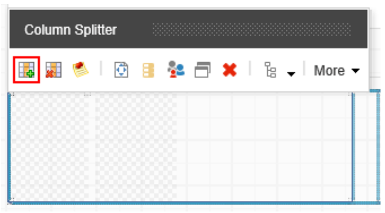




3. Now you can add renderings to the placeholders you created. For example, drag the Image (Reusable) rendering to the left four columns to add a logo.



4. To divide the groups of columns differently, click Add a new column at the end of the splitter. For example, if you need a page that lists three products next to each other, add a placeholder to have three sections of four columns.



Note

To increase or decrease the number of rows on a page, in the Experience Editor, drag the Splitter (Row) rendering to the page and use the floating toolbar. You can also further embed splitters in it to subdivide the rows or just add other renderings directly in the rows.

Send feedback about the documentation to docsite@sitecore.net.

Create and assign a page design in the Content Editor

In SXA, you work with reusable pieces of content and layout. A page design in SXA is a group of partial layouts that make up the design of a webpage.

For example, you may need a reusable page structure for a blog page. By creating a page design that includes the partial designs and renderings that you need for a blog page, you create a basic template that content authors can use to create content without having to worry about the design of the page.

You can work with SXA page designs in both the Content Editor and the Experience Editor.

This topic outlines how to:

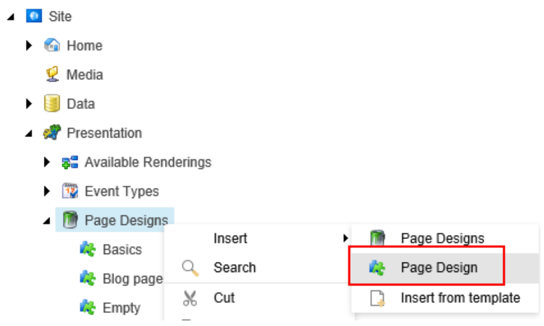
- Create a Page Design
- Assign a Page Design

Create a Page Design

A page design determines the layout of a page and consists of partial designs and renderings. You can add page designs to the presentation folder of your site and select the partial designs that you want to add.

To create a page design:

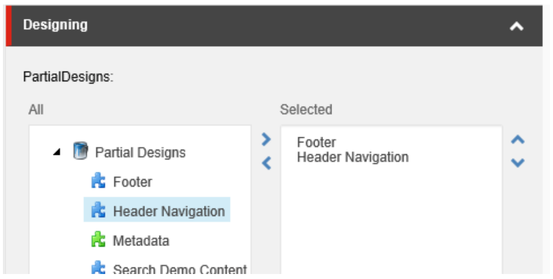
1. In the content tree, on your site, click Presentation, right-click Page Designs, click Insert, Page Design.



Note

If you want to add a group of related page designs, click Page Designs. This can be convenient if you have a complex site that requires a large number of partial designs. You can divide your page into sections of partial designs, for example, blogs, news, products, and careers.

2. Enter a name for the new page design and click OK.
3. In the Designing section, select the partial designs that you want to add, click the right arrow to move them to the list of selected items, and then save it.



4. Right-click the new page design, and then click Experience Editor to view your design.

Note

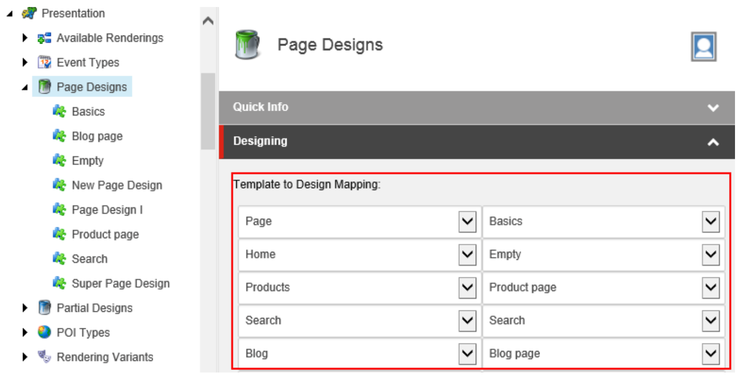
If a page design is not in use, you may want to delete it. To delete a page design in the Content Editor, right-click the page design and click Delete.

Assign a Page Design

You can use Page Designs to map content types to your page layouts. By doing this, you keep the layout of your site consistent. For web pages that you use often, such as landing pages, product pages, and navigation pages, you can keep them consistent by assigning a page design to the template for that page. In this way, you link the content to the design.

To assign a page design to a template:

1. In the content tree, go to your site, click Presentation, Page Designs.
2. In the Designing section, select a template in the left column and, to associate it to a page design, in the right column, select the page design.

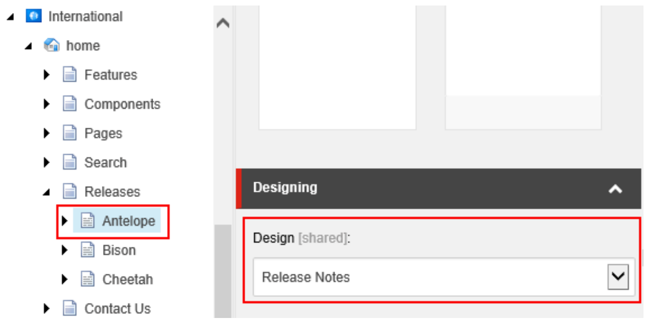


Template to Design Mapping:

Page	Basics
Home	Empty
Products	Product page
Search	Search
Blog	Blog page

Note

You can also assign page designs to specific pages. This may be necessary when you need a page that you normally don't use very often, such as for example a release notes page. To assign a page design to a single page, go to your site and select the page. In the Designing section, select the design from the drop-down list and save the changes.



Right-click the page and click Experience Editor to view the result.

Send feedback about the documentation to docsite@sitecore.net.

Create and assign a page design in the Experience Editor

In SXA, you work with reusable pieces of content and layout. A page design in SXA is a group of partial layouts that make up the design of a webpage.

For example, you may need a reusable page structure for a blog page. By creating a page design for a blog, you can create a basic template that content authors can use to create content without having to worry about the design of the page.

This topic outlines how to:

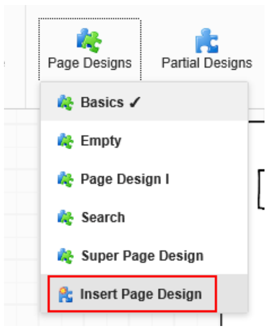
- Create a page design
- Assign a page design

Create a page design

In SXA, you can set up a page design to determine the layout of a page.

To create a page design in the Experience Editor:

1. On the ribbon, on the Experience Accelerator tab, click Page Designs, and then click Insert Page Design.



2. In the Insert Item dialog box, click Page Design, enter a name, and click OK.

   Note

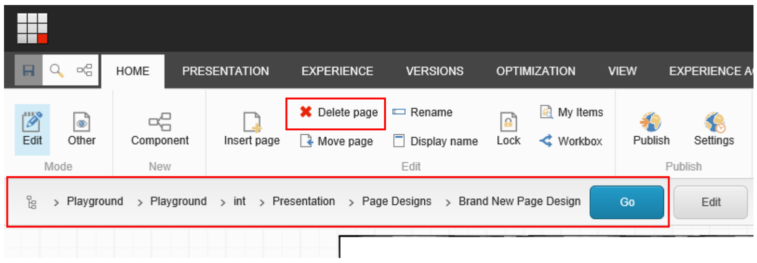
   If you want to add a group of related page designs, click Page designs. This can be convenient if you have a complex site that requires a large number of partial designs. You can divide your page into sections of partial designs, for example, blogs, news, products, and careers.

3. In the Select items dialog, select the Partial Design(s) that you want to add, click the right arrow › to move it to the list of selected items, and click OK.

When you are no longer using a page design, you may want to delete it.

To delete a page design in the Experience Editor:

1. On the ribbon, on the View tab, select Navigation bar, click the page design in the bar that appears, and then click Go.
2. On the Home tab of the page design, click Delete page.

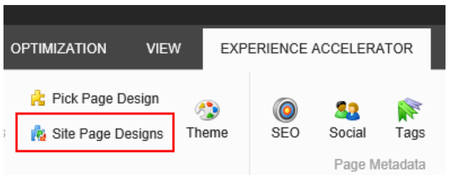


Assign a page design

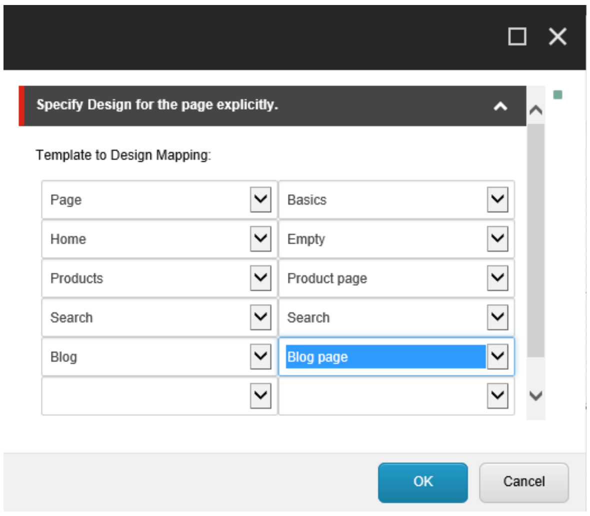
You can use page designs to map content types to your page layouts. By doing this, you keep the layout of your site consistent. For web pages that you use often, such as landing pages, product pages, and navigation pages, you can keep them consistent by assigning a page design to the template for that page. In this way, you link the content to the design.

To assign a page design to a template:

1. On the ribbon, on the Experience Accelerator tab, click Site Page Designs.



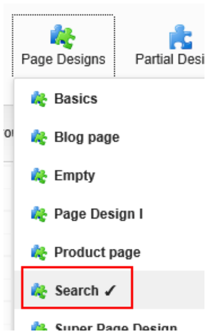
2. In the dialog box that appears, in the left column, select a template and, to associate it to the page design, in the right column, select the page design.


Specify Design for the page explicitly.
Template to Design Mapping:
OK
Cancel

Page	Basics
Home	Empty
Products	Product page
Search	Search
Blog	Blog page

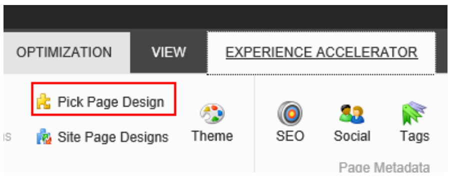
3. Click OK and Save to apply the changes.

To see the page design assigned to a page, click Page Designs. The assigned page design is marked, for example, Search.



Note

If you don't use a particular design very often or you want to override a particular page design, you can assign a page design to a specific page. To do this, click Pick Page Design, select the page, click OK and Save to apply the changes.



Send feedback about the documentation to docsite@sitecore.net.

Create and change a partial design

SXA uses sets of renderings, called partial designs that make up a webpage. You can use the partial designs to create the design elements of your pages quickly for a consistent style. For example, you can create parts of your page once, such as headers and footers, and then use them everywhere on your site.

You can also change a partial design for a specific page, for example, if you need a slightly different header on a particular page. Partial designs can inherit from each other, so you can build increasingly complex designs from a basic set of reusable partial designs.

This topic outlines how to:

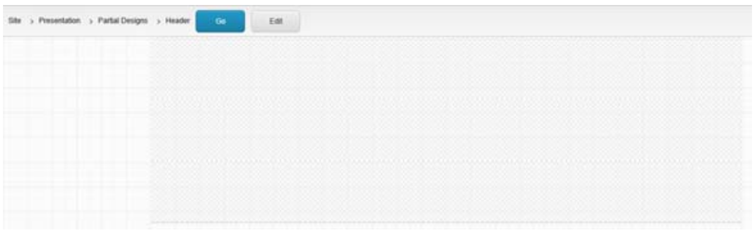
- Create a partial design
- Change a partial design

Create a partial design

If the existing partial designs do not suit your needs, you can create a new partial design to reuse across pages, for example, a partial design for a page header.

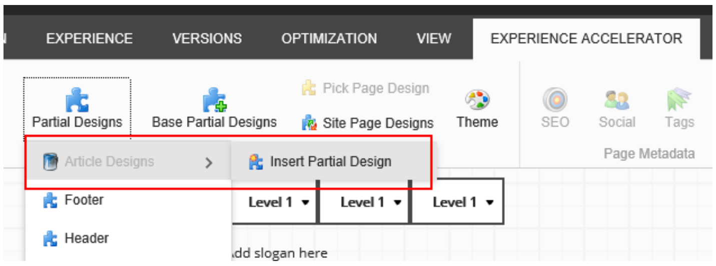
To add a partial design in the Experience Editor:

1. On the ribbon, on the Experience Accelerator tab, click Partial Design, and then click Insert a new Partial Design.
2. In the Insert Item dialog box, select Partial Design, enter a name and click OK. SXA now loads an empty page. If you select the navigation check box, you can see you are in the presentation layer.

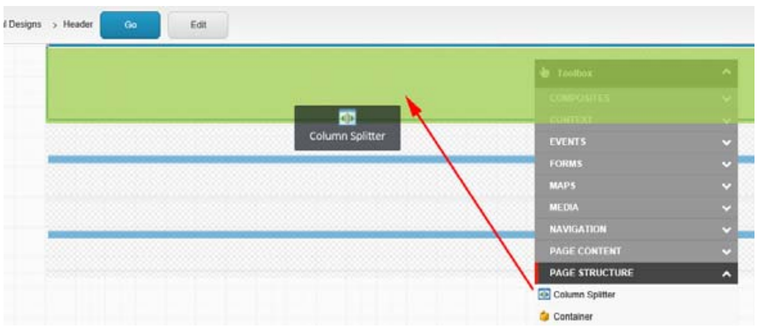


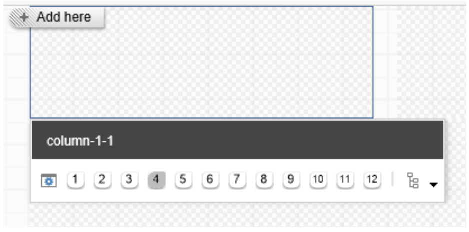
Note

If you want to create a group of related partial designs, select Partial Designs and enter a name and click OK. When you click the new group, you can add partial designs to it. This can be convenient if you have a complex site that requires a large number of partial designs. You can divide your page into sections of partial designs, for example, blogs, articles, and careers.

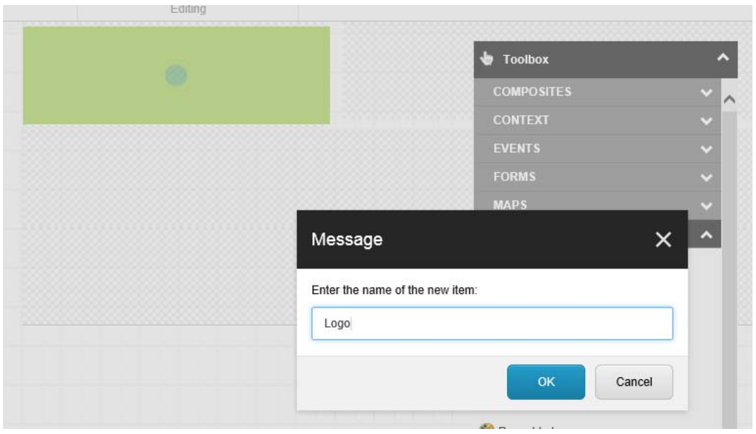


3. Now you can add renderings to your partial design by dragging them from the Toolbox to the page. For example, for a header, you could add some layout elements and divide the header in 4 columns on the left and 8 on the right. Drag the Splitter (Columns) rendering to the page and split the layout.





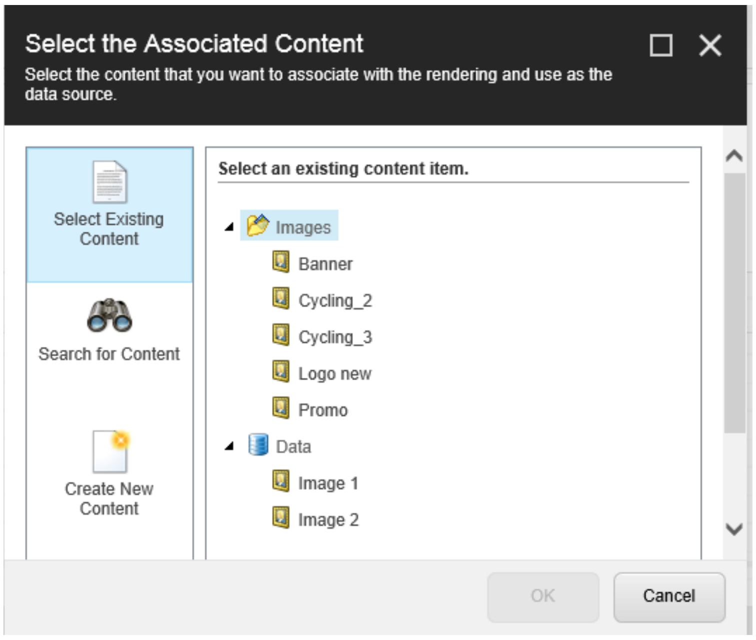
4. Use the Image (Reusable) rendering to add a logo to the left 4 columns. To do that, drag the Image (Reusable) rendering to the left columns, enter a name and click OK.



Enter the name of the new item:

Logo

5. In the Select the Associated Content dialog, select the image and click OK.



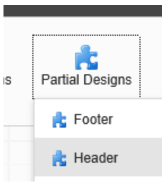
Select the content that you want to associate with the rendering and use as the data source.

Select an existing content item.

Note

If you select a data source under the Page Data node it will be a local data source that is stored as a subitem of the page item. Any changes you make to local data sources will only affect the page you are working on. If you want to be able to reuse the data source and manage it globally, select a different place in the tree.

When your header is ready after for example adding a slogan to the header, and adding navigation, save it to make it available in the Partial Design menu.



Change a partial design

Occasionally, you may want to change a partial design. For example, you might need to update it because the address in the footer has changed or because you want to add a slogan to the header.

Note

When you change a partial design, the changes are updated on every page that uses this partial design. Therefore, you should make sure that you check where it is used before you change it.

To change a partial design in the Experience Editor:

1. On the ribbon, on the Experience Accelerator tab, click Partial Design.
2. Select the partial design that you want to change.

   Note

   When you hover over the partial designs, the renderings that belong to this partial design are highlighted in green on your page.

3. Make the changes by for example changing the text or adding renderings from the Toolbox to the partial design.

4. On the main ribbon, click Save . Now the partial design has changed for every page that uses it.

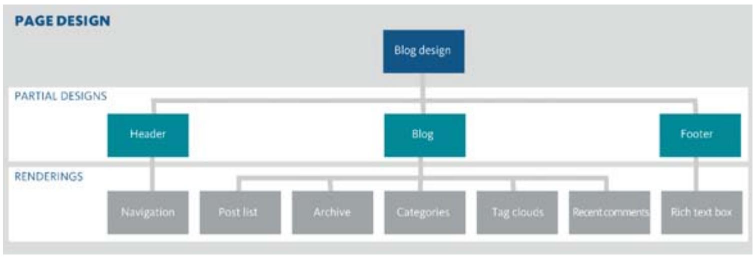
Send feedback about the documentation to docsite@sitecore.net.

Page designs

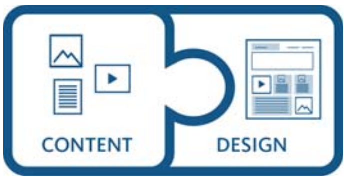
In SXA, you build your website with reusable pieces of content and layout. All these pieces together create the layout for your pages.

A page design in SXA is a selection of partial designs and renderings that help you to structure your pages. You can, for example, make sure the headers and footers are always in the same place. You can also set up a page structure for specific pages, such as a blog page, a landing page, a product page, and so on. Content authors can then place content in these preset layouts.

You assign a page design to a page (or pages of a specific type) to define the elements and renderings that you want to appear. For example, a blog page may need a header with a navigation component, a main placeholder for the content (post list, categories, archive, tag cloud, recent comments), and a footer with company information.



Data templates are the schema for Sitecore content. Any content item in a Sitecore database is based on a data template. In SXA, you can link page designs to data templates. In this way, you can link your content types to your page layouts and keep the layout of your site consistent. It is very convenient to assign a page design to the template for web pages that you know you will be using a lot, so that they look consistent.



Send feedback about the documentation to docsite@sitecore.net.

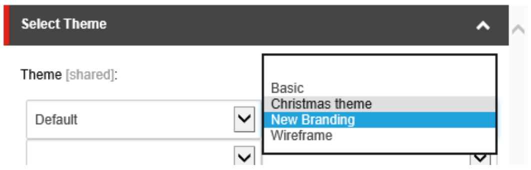
Assign a theme

In SXA, you can style your pages using themes. Themes let you change the style of an existing site, for example, because of company branding changes or if you want a holiday edition, without interfering with the content. You can use Creative Exchange to export and import themes.

You can assign a different theme in the Experience Editor or in the Content Editor.

To assign a theme:

- In the Experience editor, on the ribbon, on the Experience Accelerator tab, click Theme, and in the Select Theme dialog box, select the theme you want to use. Click OK to save your changes.



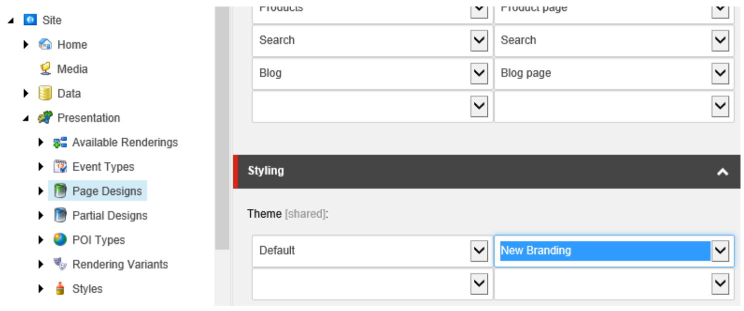
Select Theme

Theme [shared]:

Default

Basic
Christmas theme
New Branding
Wireframe

- In the Content Editor, on your site, click Presentation, Designs, and in the Styling section, select the theme you want to use.



Products | Product page
Search | Search
Blog | Blog page

Styling

Theme [shared]:

Default | New Branding

Note

If you do not select a theme, the default theme is assigned.

Send feedback about the documentation to docsite@sitecore.net.

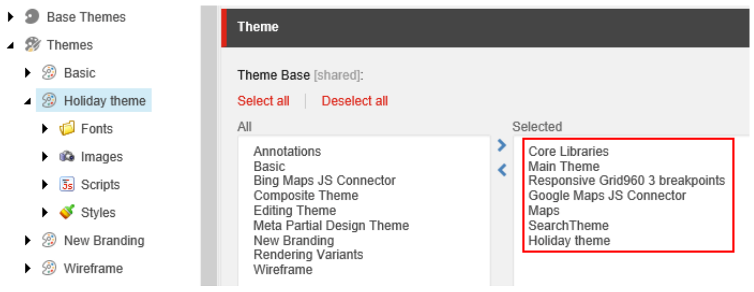
SXA themes

Themes define the look and feel of a site and can be created separately from the site functionality and content. There are two types of themes: base themes and site themes.

SXA comes with the default site theme named *Wireframe* to help you set up your site quickly. A site can be put together using the wireframe theme, while in the meantime the basic theme is sent to a creative agency using Creative Exchange. When imported back, the site can be re-skinned using the new theme. You can create a new theme by copying the Basic theme and adding your own classes, applying a style to a particular rendering on a particular page, and adding assets, such as images, fonts, and files.

SXA themes contains scripts, images, and styles and are located in the Media Library: `/sitecore/Media Library/Themes`

SXA themes support multiple inheritance. This means that you can define multiple parent themes for every single theme. For example, the Holiday theme shown below inherits from several other themes:



All: Annotations, Basic, Bing Maps JS Connector, Composite Theme, Editing Theme, Meta Partial Design Theme, New Branding, Rendering Variants, Wireframe

Selected: Core Libraries, Main Theme, Responsive Grid960 3 breakpoints, Google Maps JS Connector, Maps, SearchTheme, Holiday theme

This topic describes the structure and use of:

- Base themes
- Site themes
- Wireframe theme

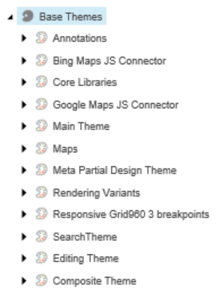
Base themes

Base themes are prototype themes that predetermine the layout of a website. You can have several base themes to support different design philosophies or specific functionality.

Note

Do not change base themes because these are part of the platform. If the base themes do not suit your needs, it is better to create a new base theme to inherit from.

The base themes are saved in the Media Library: `/sitecore/Media Library/Base Themes`.



The *Base Themes* folder contains:

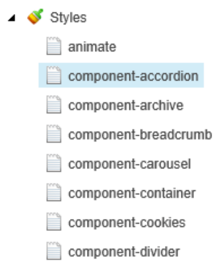
- Core Libraries: the third-party libraries used in projects such as: jQuery, jQuery UI, lo-dash, mediaelement JS, Modernizr, and so on.
- Main Theme: the scripts and styles that are part of the platform (except for the rendering scripts). Main theme has a dependency on Core Libraries so if you are inheriting from it, make sure that you also inherit from Core Libraries first.
- Grid themes: grid CSS generated by a sass grid generator.

Site themes

Site themes are extensions of base themes and can be applied to specific sites. Site themes usually have dependency on base themes and contain scripts and styles for all renderings used in a site.

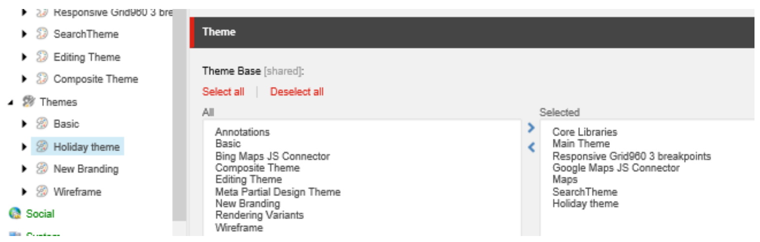
Site themes contain: CSS styles, theme images, and JavaScript libraries used to provide your site branding. Site themes often contain Sass sources, compass config, used to generate CSS styles. In this case, front-end developers should not modify the CSS files directly but rather work within the Sass workflow.

The CSS and JavaScript files in themes are divided into chunks that deal with one rendering at a time.



Styles
- animate
- component-accordion
- component-archive
- component-breadcrumb
- component-carousel
- component-container
- component-cookies
- component-divider

For each site theme, you must select a base theme to define the basic characteristics and properties of the theme. Every theme also needs a grid theme.



Responsive Grid960 3 bre
SearchTheme
Editing Theme
Composite Theme
Themes
Basic
Holiday theme
New Branding
Wireframe
Social

Theme

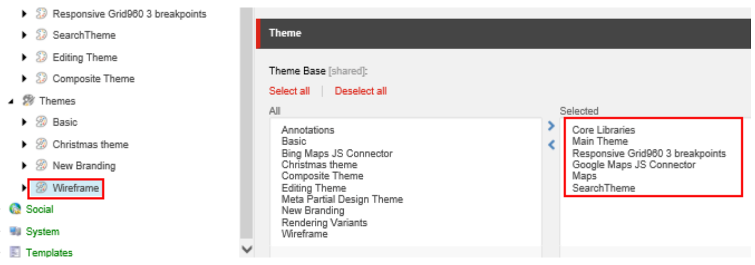
Theme Base [shared]:

Select all | Deselect all

All	Selected
Annotations	Core Libraries
Basic	Main Theme
Bing Maps JS Connector	Responsive Grid960 3 breakpoints
Composite Theme	Google Maps JS Connector
Editing Theme	Maps
Meta Partial Design Theme	SearchTheme
New Branding	Holiday theme
Rendering Variants	
Wireframe	

Wireframe theme

The standard SXA wireframe theme that comes with core CSS and JavaScript is always the starting point for a new site. You can swap the wireframe theme with a site theme once it has been prepared by your front-end developers.



Responsive Grid960 3 breakpoints
SearchTheme
Editing Theme
Composite Theme
Themes
Basic
Christmas theme
New Branding
Wireframe
Social
System
Templates

Theme

Theme Base [shared]:

Select all | Deselect all

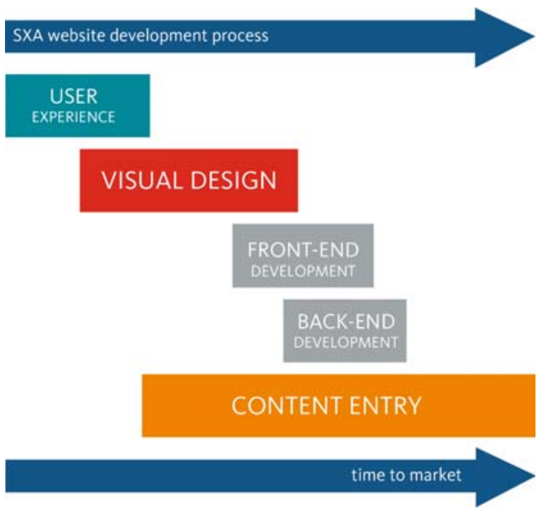
All	Selected
Annotations	Core Libraries
Basic	Main Theme
Bing Maps JS Connector	Responsive Grid960 3 breakpoints
Christmas theme	Google Maps JS Connector
Composite Theme	Maps
Editing Theme	SearchTheme
Meta Partial Design Theme	
New Branding	
Rendering Variants	
Wireframe	

Send feedback about the documentation to docsite@sitecore.net.

Introducing Sitecore Experience Accelerator

Web development teams use Sitecore Experience Accelerator (SXA) to speed up the production of websites and to reuse components, layouts, and templates across a variety of sites.

SXA separates structure from design, so front-end designers, creative designers, content authors, and developers can work in parallel and you can deploy and maintain multiple sites quickly and cost effectively. Once a basic user experience plan is in place, everyone can get started on the platform. For example: the content author can start entering the content in the wireframe environment, while the front-end developer works on the theme, and the developer sets up the data templates.



With SXA you can:

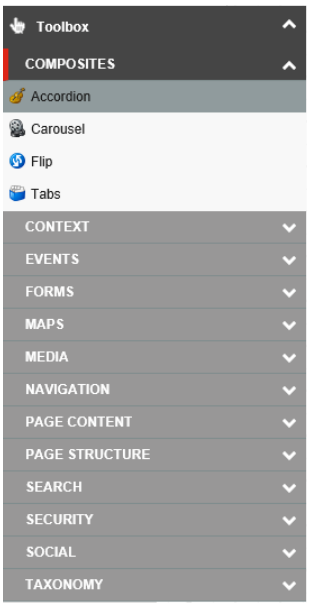
- Accelerate the delivery of sites using standard functionality with minimum-to-no CMS development.
- Enable different work streams to run in parallel.
- Assemble sites using responsive and reusable renderings.
- Use themes to enable brand consistency.

SXA uses the following concepts:

- Toolbox
- Grid and column layout
- Themes
- Page designs and partial designs
- Creative exchange

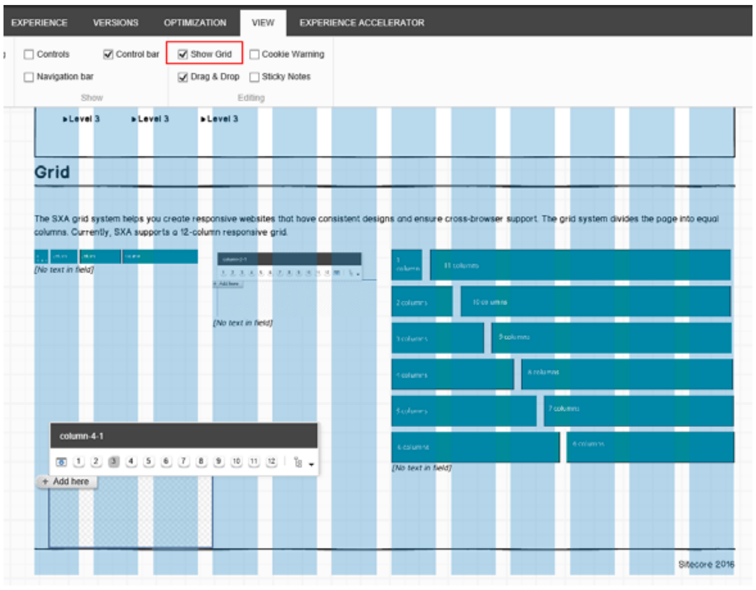
Toolbox

To make it easier to construct your page, SXA comes with a toolbox with reusable renderings that you can drag and drop onto your page. The renderings vary from simple text and images, to videos, and social media plugins.



Grid and column layout

SXA pages use a responsive grid layout. The grid divides pages into equal columns. By using row and column splitters, you can decide how to divide the available columns on your page.



Grid

The SXA grid system helps you create responsive websites that have consistent designs and ensure cross-browser support. The grid system divides the page into equal columns. Currently, SXA supports a 12-column responsive grid.

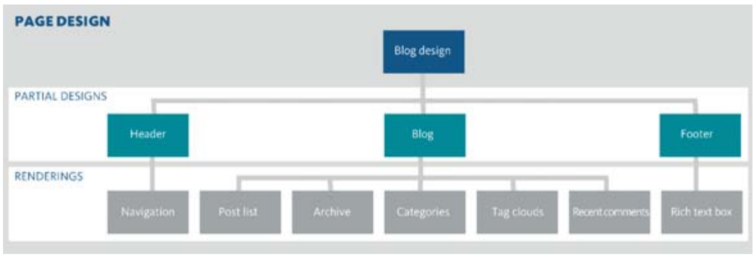
[No text in field]

Pluggable themes

SXA separates structure (HTML) from design (CSS) to make it easier to change the design of websites. To enable easy customization, SXA uses themes. A theme consists of style sheets, script, and images. You can add pluggable themes to SXA to enable changing the styling of a site quickly. Users can begin developing a site using the wireframe theme. When they finish a design, they can use Creative Exchange to import the new theme and re-skin the site.

Page designs and partial designs

Partial designs contain parts of a layout that a site uses in multiple places. A page design is the presentation definition for a page and consists of partial designs and renderings.



Creative Exchange

You can export a static representation of a site and send it to a creative agency to work with. The export file is a .zip that contains the site pages, the site assets, and the site theme. The agency can edit the exported site using their favorite tools and when they have finished, you can import the improved design back into the site.

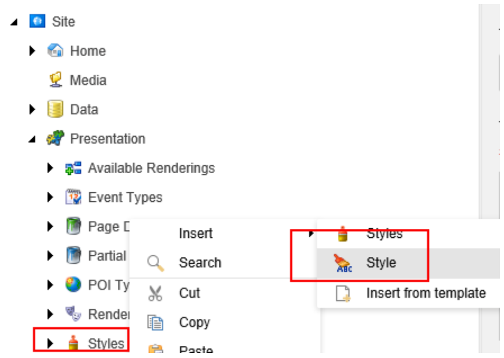
Send feedback about the documentation to docsite@sitecore.net.

Add a style for a rendering

SXA comes with preset styles for renderings. However, sometimes you may want to add your own custom styles. For example, you want some of the images on your site to appear without margins or if you want to be able to add a background color to a rendering.

To add a class item for a rendering in the Content Editor:

1. Go to the site's Presentation folder.
2. Right-click Styles, click Insert, and then click Style to add a new style.

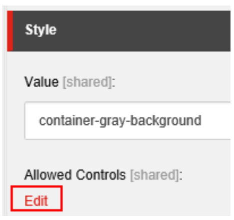


3. Enter the name and click OK.

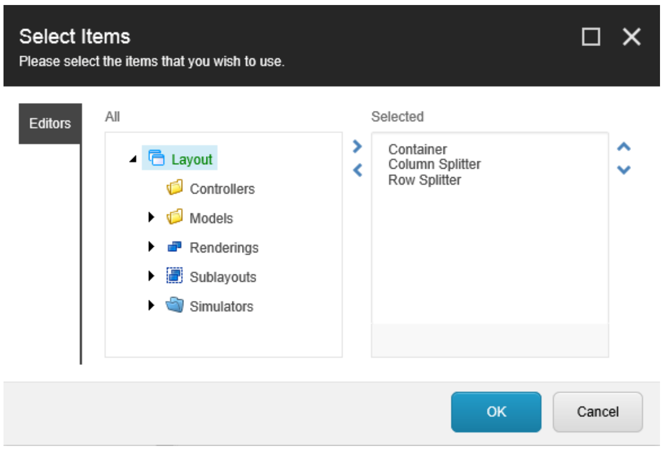
   Note

   Make sure that your new style has a name that helps other users to understand what it does.

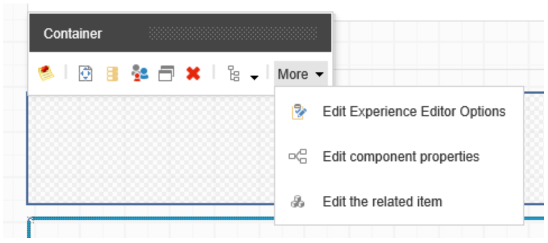
4. To make the new style available for the rendering, in the Style section, click Edit.



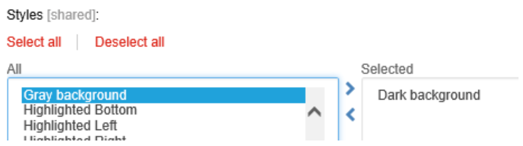
5. In the Select items dialog, click the relevant rendering, click the right arrow › to move it to the list of selected items and click OK. For this example, the grey background is made available for the Container and splitter renderings.



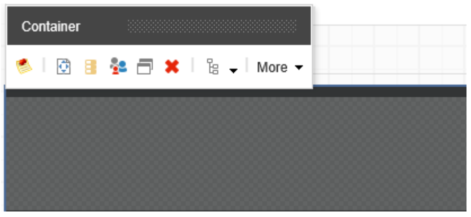
6. To see your new style, in the Experience Editor, open the properties of the rendering.



1. In the Styles section, you can now select the new style.



2. Click OK to apply the new style to the rendering.



Send feedback about the documentation to docsite@sitecore.net.

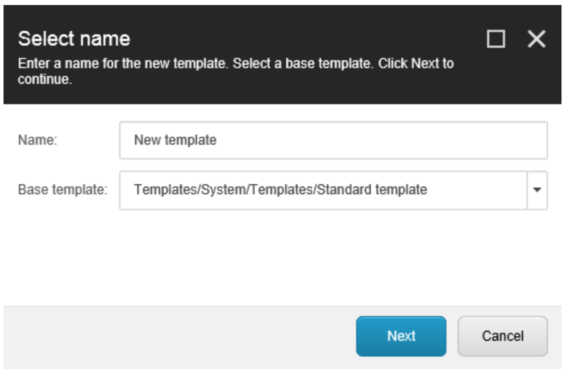
Add an SXA template

Templates are preset schemas for content items that are used to base new items on instead of recreating the specific fields of a new item every time. You may need to add a data template when, for example, a project requires fields that are not defined in existing data templates, or when new items require unique default field values or default settings (for example, a default workflow).

You can add new templates for specific projects. This may be convenient if you are working on a project that needs custom templates. You can add your project templates to: `/sitecore/templates/Project/`

To add a template:

1. Log in to the Content Editor and select the template folder where would you like to add a template. For example, to add a custom template to your new project go to: `/sitecore/templates/Project/projectname`
2. Right-click the item and click Insert, New Template. Alternatively, click New Template in the Options section.
3. Enter a name for the new template, select a base template, and click Next.


Select name

Enter a name for the new template. Select a base template. Click Next to continue.

Name: New template

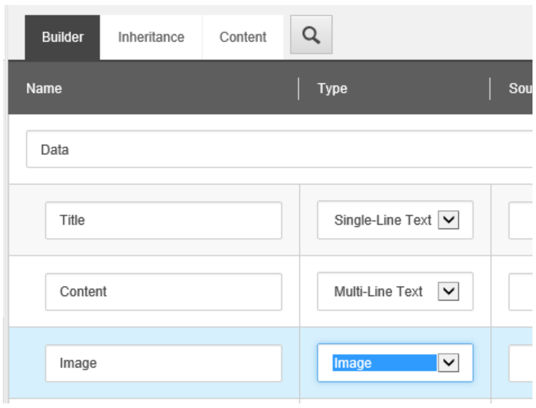
Base template: Templates/System/Templates/Standard template

Note

Templates can consist of many base templates. A template inherits the fields and sections defined in its base templates.

You can add additional base templates to your template when they define standard fields that you want your new template to inherit. To do this, click the template and on the Content tab, in the Data section, double-click the template or use the arrow to add it.

4. Select the location for your new template, and click Next.
5. On the Builder tab, in the Add a new section field, add the relevant data template fields, for example, *Data*.



Name	Type	Sou
Data		
Title	Single-Line Text	
Content	Multi-Line Text	
Image	Image	

Note

To assign default values to fields in your template, on the Options tab, click Standard values. With each new item created from that template, fields will inherit values from the corresponding field in the standard values item.

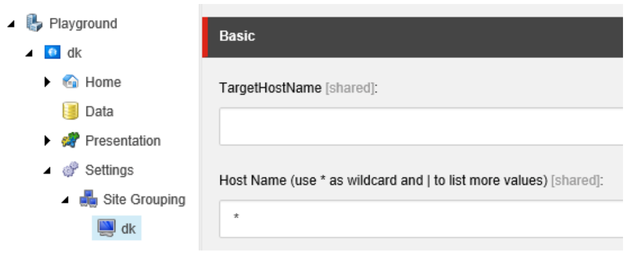
6. Save your changes.

Send feedback about the documentation to docsite@sitecore.net.

Configure a sitemap

Sitemaps help search engine crawlers navigate your site and improve search engine optimization (SEO). With SXA, by default the sitemap is generated for the whole site and stored in cache. An XML sitemap is created specifically for search engines to show details of the available pages in a website, their relative importance, and the frequency of content updates.

By default, the sitemap uses the Host Name defined in the Basic section in the Settings item of your site (`site/Settings`). If both the Host Name and the TargetHostName fields are empty, the sitemap returns a 404 error.



Basic

TargetHostName [shared]:

Host Name (use * as wildcard and | to list more values) [shared]: *

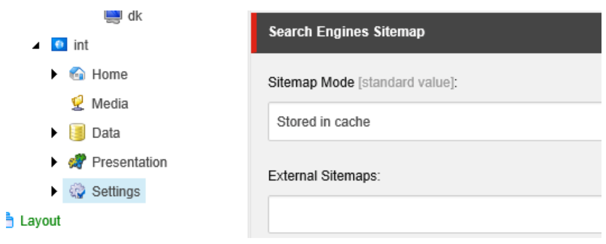
This sitemap file is stored in the root folder of your server, and is usually named `sitemap.xml`.

Every entry in the sitemap contains the following attributes:

- `Loc` – the location of the page.
- `Lastmod` – the date when the page (under `loc`) was last modified.
- `Changefreq` – how often the page changes its content.
- `Priority` – number between 0 and 1 that represents the importance of specific page.

To configure the sitemap for your site:

1. In the Content Editor, navigate to *site/Settings* and in the Search Engines Sitemap section, in the Sitemap Mode field, select the storage option for the sitemap.
   - *Inactive* – turns the sitemap off.
   - *Stored in cache* – stores the sitemap for the whole site in cache. Select this option if your site is hosted on an environment such as Azure and you cannot easily store files on a drive, or if your site is very dynamic and you need to re-generate the sitemap almost every time it is requested. This option is turned on by default.
   - *Stored in file* – stores the sitemap for the whole site in file. Select this option if you have a large site that does not change frequently.

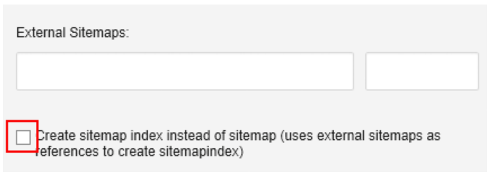
   

   Sitemap Mode [standard value]: Stored in cache

   External Sitemaps:

   Optionally, you can reference external sitemaps in the generated index file by adding the sitemap in `[KEY][VALUE]` format, where `KEY` is name of your choice and `VALUE` is a direct link to an external sitemap file. For example: *http://example.com/sitemap.xml*

2. If you want to reference the external sitemaps in a sitemap index file, select the Create sitemap index instead of sitemap check box. This can be useful if you host external sites such as a WordPress blog on your site.

   External Sitemaps:

   Create sitemap index instead of sitemap (uses external sitemaps as references to create sitemapindex)

3. Save the settings and publish the item. Once the settings are saved, you can find the sitemap here: `TargetHostName/sitemap.xml`. A link to the sitemap is automatically added at the end of the `robots.txt` file.
4. Publish the page to automatically update the sitemap.

Send feedback about the documentation to docsite@sitecore.net.

Create a rendering variant

SXA comes with a set of default renderings and rendering variants. Rendering variants are configurable adaptations of the default renderings. To further encourage reusability, designers and front-end developers can also create new rendering variants. This gives authors more options in the way they present their content.

You can create your own variation of a rendering by adding a variant in the Content Editor.

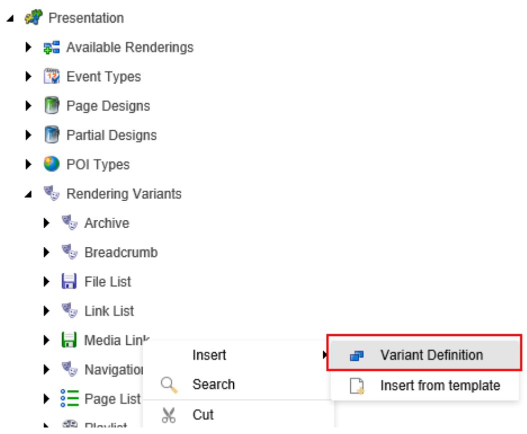
To create a rendering variant:

1. In the Content Editor, click the site and open the *Presentation/Rendering Variants* folder. This folder lists all the renderings that allow variants.

   Note

   To add a rendering to the folder, contact your Administrator.

2. Right-click the rendering that you want to add the variant to, and then click Insert, Variant Definition.

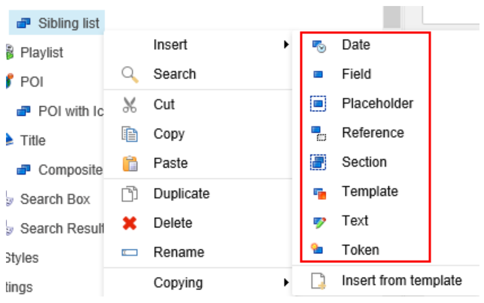
   

3. Enter a name and click OK.
4. In the Variant Details section, specify information in the following fields:
   - Css Class: specify the class name for the variant in the HTML.
   - Item CSS class: specify the CSS class for list items. For example, for variants of `PageList`, `LinkList`, `FileList`, and `Navigation` renderings.
   - Field used as link target: provide the field name of the target item. This class is added to the list and navigation items markup (`li` HTML element). This link is used to override all links when the Is link, Is prefix link, or Is suffix link check boxes are selected.
   - Allowed in templates: specify the pages that the variant is available for. Click the relevant page template, click the right arrow to move it to the list of selected items, and then click Save. If there are no templates selected, the variant is available for all pages.

5. To add child items to the rendering variant, right-click the variant, click Insert, and then click the relevant item.
   - Date: displays data and time in custom format.
   - Field: displays field name and other field values.
   - Placeholder: allows to render an empty placeholder.
   - Reference: displays field from referenced item.
   - Section: used to create groups. For example, if you want to render two div elements, you can create two section items that each contain one of the two div elements.
   - Template: allows user to define NVelocity template which will be used to render part of the HTML.

     Note

     NVelocity is a .Net-based template engine. It permits developers to use the simple yet powerful template language to reference objects defined in .Net code.
   - Text: displays text.
   - Token: supports tokens `$Size` and `$FileTypeIcon`.



Depending on the item you add, you can set the following fields:

Field	Variant details
Tag	HTML tag in which the field content will be rendered. For example: `div` If left empty field content will remain unwrapped.
Field name	Name of the field current item.
Prefix	Adds string value as a prefix.
Suffix	Adds string value as a suffix.
Is link	Select to have hyperlinks that wrap the field content.
Is prefix link	Select to wrap prefix in the same link which you use for the field content.
Is suffix link	Select to wrap the suffix with a hyperlink.
Is download link	Select to have a hyperlink with a download attribute.
Css Class	Adds Css class to the tag.
Is editable	Select to edit rendered field.
Date format	Determines the date format.
Render if empty	Select to render empty element when the field is empty.
Pass through field	Defines the name of the field from the nested item.
Text	The text to render.
Template	Define the NVelocity template that renders part of the component variant HTML. You can use the following objects: `$item`: access to the current item (`$item.Name` displays current item name). `$dateTool.Format(date, format)`: formats date and time values. `$numberTool.Format(number, format)`: formats numbers. `$geospatial`: in case of geospatial search (`$geospatial.Distance`) will show distance to certain point of interest).

Token	Use special tokens to format certain field values. Supported tokens are: `$Size`: displays size of a file depending on size unit. `$FileTypeIcon`: displays icon depending on file extension.

Send feedback about the documentation to docsite@sitecore.net.

Create a tenant and a site

The SXA content architecture includes tenants and sites. SXA supports multitenancy, which means that you can run multiple sites on a single instance of Sitecore. Each tenant can include multiple related sites, for example, to support multiple brands for a single company or multiple languages or locations for a single brand. Organizations can support multiple languages through one-to-one translated versions (native Sitecore language support) or use a model with a separate site for each supported language.

For example, an international clothing company could have different tenants for the different brands of clothing and different sites for the specific countries.

This topic describes how to:

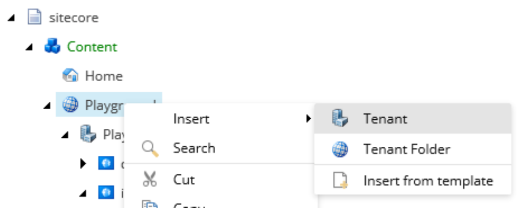
- Create a tenant and a tenant folder
- Create a site and a site folder

Create a tenant and a tenant folder

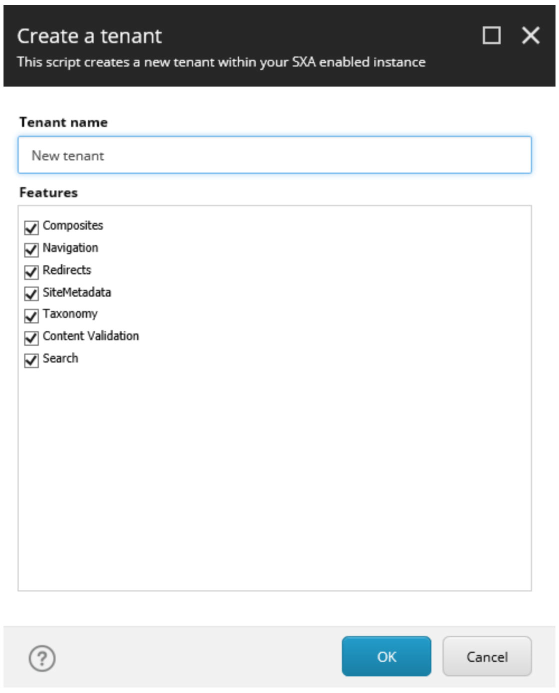
With SXA's multitenant architecture, you can provide each tenant a dedicated share of the Sitecore instance including its data templates, configuration, user management, tenant individual functionality, and non-functional properties.

To create a tenant:

1. In the Content Editor, right-click the item in the content tree, click Insert, Tenant.



2. In the wizard, enter a name for the tenant, select the features, and click OK.



Create a tenant

This script creates a new tenant within your SXA enabled instance

Tenant name

New tenant

Features

- [✓] Composites
- [✓] Navigation
- [✓] Redirects
- [✓] SiteMetadata
- [✓] Taxonomy
- [✓] Content Validation
- [✓] Search

For more complex solutions, you can use groups of tenants. For example, a multinational selling consumer goods could have the following tenant folders and tenants:

Company (Tenant Folder)

- Cosmetics (Tenant Folder)
  - Brand A (Tenant)
  - Brand B (Tenant)
- Laundry detergents (Tenant Folder)

- Brand A (Tenant)
- Brand B (Tenant)
- Brand C (Tenant)

- Hair care (Tenant Folder)
    - Brand A (Tenant)
    - Brand B (Tenant)
    - Brand C (Tenant)
    - Brand D (Tenant)

To create a group of tenants:

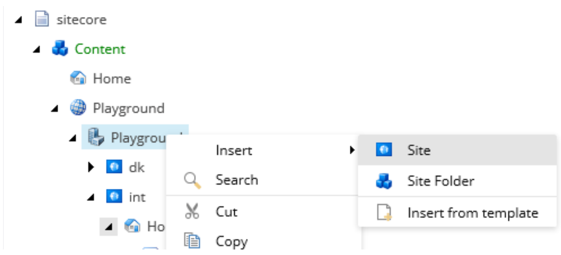
- Right-click the content item in the content tree, click Tenant Folder, enter a name and click OK.

Create a site and a site folder

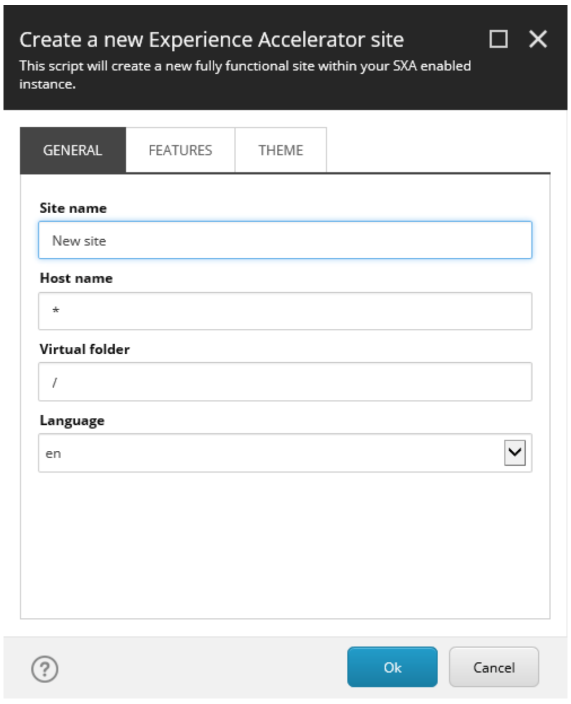
The tenant is a top-level container for the sites underneath. Sites in the same tenant are related, for example, because they share the same set of templates or part of the media library. Sites are the items that represent the website and consist of pages, data, designs, and partial layouts.

To create a site:

1. In the Content Editor, right-click the tenant to which you want to add the site.



2. In the wizard, on the General tab, enter the name for the site.


Create a new Experience Accelerator site

This script will create a new fully functional site within your SXA enabled instance.

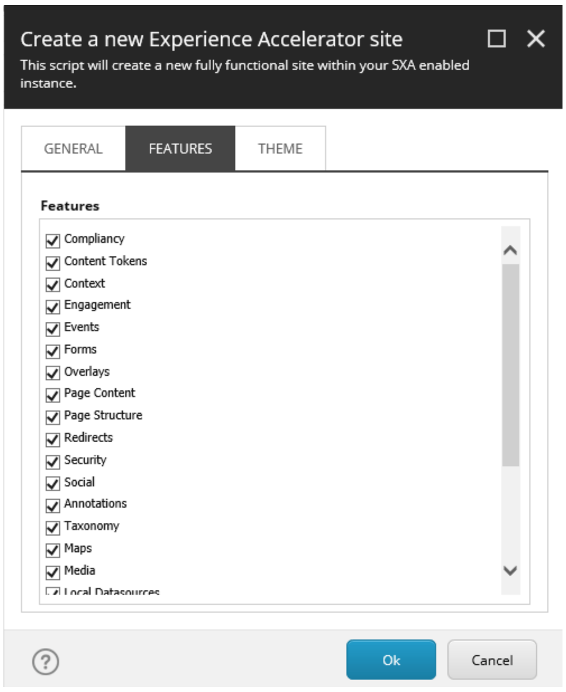
Site name: New site

Host name: *

Virtual folder: /

Language: en

3. On the Features tab, select the features and click OK.

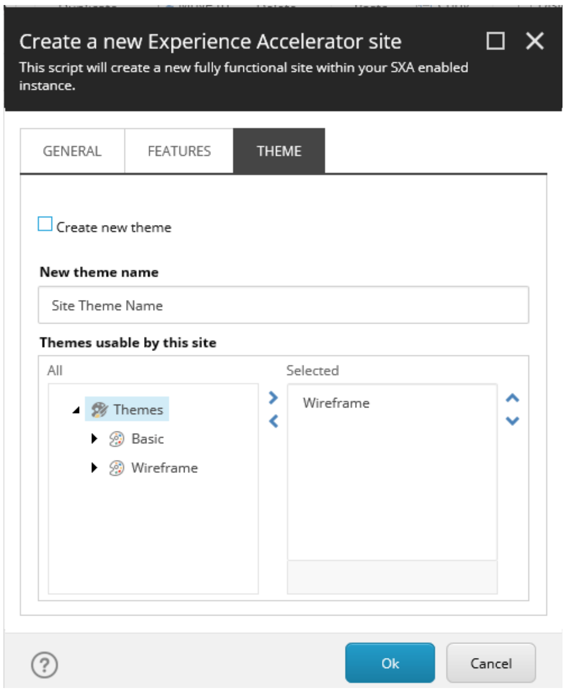

Create a new Experience Accelerator site

This script will create a new fully functional site within your SXA enabled instance.

Features

- ✓ Compliancy
- ✓ Content Tokens
- ✓ Context
- ✓ Engagement
- ✓ Events
- ✓ Forms
- ✓ Overlays
- ✓ Page Content
- ✓ Page Structure
- ✓ Redirects
- ✓ Security
- ✓ Social
- ✓ Annotations
- ✓ Taxonomy
- ✓ Maps
- ✓ Media
- ✓ Local Datasources

4. On the Theme tab, either create a new theme by selecting Create new theme or select one or more existing theme(s) and click OK.


Create a new Experience Accelerator site

This script will create a new fully functional site within your SXA enabled instance.

☐ Create new theme

New theme name

Site Theme Name

Themes usable by this site

Your new site is available immediately.

For governance reasons, you can decide to use groups of sites. For example, for an internalization model where you create different sites for different countries:

- Europe (site folder)
  - Poland (site)
  - Denmark (site)
  - The Netherlands (site)
  - Ukraine (site)
- Asia (site folder)
- Africa (site folder)

To create a group of sites:

- Right-click the tenant in the content tree, click Site Group, enter a name and click OK.

Send feedback about the documentation to docsite@sitecore.net.

Data sources

SXA comes with a library of predefined renderings to ensure modular component based design. Most SXA renderings are designed for reusability and pull data from data source items. This means that the content they display is not bound to the page on which they appear but is stored in data source items. When you add a rendering to a page, in the Associated Content dialog you can select an existing or create a new data source item.

The following fields determine how a rendering relates to its data source item:

- Datasource Location – specify where the user is allowed to look for the data source.
- Datasource Template – specify the types of data sources users can create.
- Data source – specifies a data source item.

This topic describes how SXA renderings, depending on how they use data source items, can be divided into five groups:

- No data source item
- Optional data source item
- Single data source item per platform/tenant/site
- Reusable data source
- Auto-generated data source

Note

When building a new rendering, the Sitecore developer must decide which category a new custom rendering falls into, so the rendering can be properly configured.

No data source item

Renderings without data source items do not store their own data, for example, the Navigation and Breadcrumb renderings. They are static and not editable by content authors.

Field	Value
Datasource Location	Leave empty.
Datasource Template	Leave empty.
Data source	Leave empty.

Optional data source item

By default, renderings with an optional data source item read data from the current page, but can be set to read information from the data source item instead. For example, the Title, Subtitle, and Content renderings.

Field	Value
Datasource Location	Leave empty
Datasource Template	Leave empty
Data source	Leave empty or reference a data source item

Single data source item per platform/tenant/site

Renderings with a single data source item always use one data source that is unique for a specific platform, tenant, or site. For example, the Cookie Warning and Promo renderings.

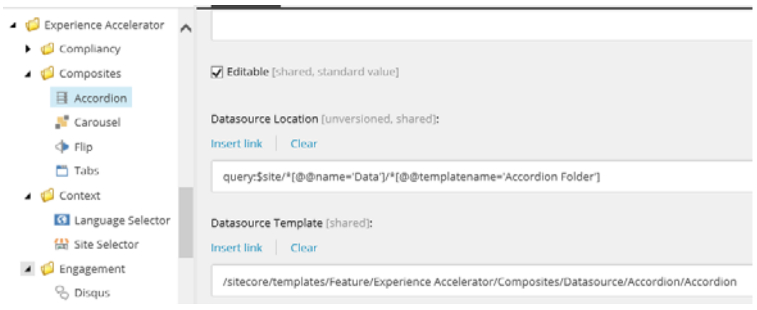
Field	Value
Datasource Location	Leave empty
Datasource Template	Leave empty
Data source	Data source item reference

Reusable data source

Renderings with a reusable data source enable the content editor to create and/or select a data source item. For example, the Accordion, Carousel, and Video renderings.

Field	Value
Datasource Location	Reference to the item that groups rendering data source items.

Datasource Template	Reference to the data source item template.
Data source	Leave empty.



Editable [shared, standard value]

Datasource Location [unversioned, shared]:
Insert link | Clear
query:$site/*[@@name='Data']/*[@@templatename='Accordion Folder']

Datasource Template [shared]:
Insert link | Clear
/sitecore/templates/Feature/Experience Accelerator/Composites/Datasource/Accordion/Accordion

Auto-generated data source

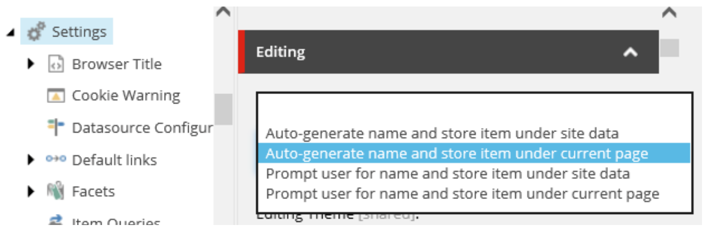
Renderings with an automatically generated data source store data and create the data source item when a content editor places the rendering on the page. For example, the Rich Text, Image, and Plain HTML renderings.

Field	Value
Datasource Location	Reference to the item that groups rendering data source items.
Datasource Template	Reference to the data source item template.
Data source	Leave empty.

Additional setup	SXA automatically generates data items for white-listed folder templates. Therefore, the grouping item (Datasource Location) must be of a named type and this name must be added to the configuration:
	`<experienceAccelerator>`
	`      <!-- List of renderings for which a data source is automatically created after adding that rendering to a page. -->`
	`      <autoDatasourceRenderings>`
	`        <rendering name="Image">{4A0D4E0B-BC20-4F2E-BD84-A4BD35834E86}</rendering>`
	`      </autoDatasourceRenderings>`
	`    </experienceAccelerator>`

Administrators can configure the behavior of non-reusable components in the Editing section of the *Settings* item (`/sitecore/content/TENANT_GROUP/TENANT/SITE/Settings)` by selecting one of the following options:

- Auto-generate name and store item under site data – a new data source item is created with an auto-generated name and is stored in the location shared across the entire site.
- Auto-generate name and store item under current page – a new data source item is created with an auto-generated name and is stored as a page relative data source item under a given page.
- Prompt user for name and store item under site data – the author is prompted to provide a name for the new data source item, which is stored in the location shared across the entire site.
- Prompt user for name and store item under current page – the author is prompted to provide a name for the new data source item, which is stored as a page relative data source item under a given page.



Auto-generate name and store item under site data
Auto-generate name and store item under current page
Prompt user for name and store item under site data
Prompt user for name and store item under current page

Send feedback about the documentation to docsite@sitecore.net.

The SXA pipelines

Understanding how the SXA pipelines and their processors work provides you with an insight into how dependencies are rendered, how tokens for rendering variants are created, how CSS classes are generated, and so on.

A pipeline consists of a sequence of processors. A processor is a .NET class that implements a method. When a pipeline is invoked, the processors are run in order. You can extend and customize the pipelines by adding or replacing processors. Extending a pipeline involves modifying the pipeline definition located in a Sitecore patch file.

This topic describes the following pipelines:

- resolveVariantTokens
- Ioc
- mediaRequestHandler
- resolveTokens
- assetService
- getControlEditability
- getRobotsContent
- refreshHttpRoutes
- getVelocityTemplateRenderers

resolveVariantTokens

The `resolveVariantTokens` pipeline is used to create tokens for rendering variants.

This pipeline includes the following processors:

Processor	Description
ResolveIFileTypeIcon	Renders the span HTML element with the class according to the extension of the file.
ResolveItemId	Specifies the ID of the content item to be resolved.
ResolveItemName	Specifies the name of the content item to be resolved.
ResolveSize	Renders the file size.

ioc

The Inversion of Control (IoC) design principle allows you to change rendering dependencies without changing the rendering itself. The `ioc` pipeline is defined in the `Sitecore.XA.Foundation.IoC.config` file and is used for adding processors in which you can register your custom services in the container.

For example, you can add the `RegisterPageContentServices` processor to register services used in the Page Content feature.

```
<pipelines>
  <ioc>
    <processor type="Sitecore.XA.Feature.PageContent.Pipelines.IoC.RegisterPageContentServices, Sitecore.XA.Feature.PageContent" /
  </ioc>
</pipelines>
```

mediaRequestHandler

The `mediaRequestHandler` pipeline is used in the SXA media requests handler. The `mdeiaRequestHandler` pipeline is defined in the `Sitecore.XA.Foundation.MediaRequestHandler.config` file.

This file extends the standard media request handler by adding a pipeline that implements custom functionalities such as the support of wireframe images or providing optimized assets.

This pipeline includes the following processors:

Processor	Description
ParseMediaRequest	Checks if the HTTP request is a valid media type. If not, the pipeline is aborted.
GetMediaFromUrl	Determines the media item from the URL that is stored in HTTP request.
HandleErrors	Redirects the user to an error page.

resolveTokens

The `resolveTokens` pipeline is used to resolve tokens that can be used in queries. For example, `$site, $tenant, $currenttemplate, $home, and $pageDesigns.` By adding new processors to this pipeline, you can design new tokens.

This pipeline is defined in the `Sitecore.XA.Foundation.TokenResolution.config` file and includes the following processors:

Processor	Description

CurrentTemplateToken	Determines the current tokens used.
EscapeQueryTokens	Used to escape tokens that are used in Sitecore queries.

assetService

The `assetService` pipeline is responsible for assets optimization. This pipeline includes the `AddEditingtheme` processor, which you can use to add a theme when you are in Edit mode.

```
<assetService>
        <processor type="Sitecore.XA.Foundation.Editing.Pipelines.AssetService.AddEditingTheme, Sitecore.XA.Foundation.Editing" />
      </assetService>
```

getRobotsContent

The `getRobotsContent` pipeline is used to extend the response provided to search crawler robots in the `robots.txt` file. The `Robots.txt` file is a simple text file on your site’s root directory that tells search engine robots what to crawl and what not to crawl on your site. The `getRobotsContent` pipeline contains the following processors:

Processor	Description
GetContentFromSettings	Checks if the robots' content field is filled and uses its value.
GetDefaultRobotsContent	Checks the `robots.txt` file for a value when the robots' content field is empty.
AppendSitemapUrl	Adds the path of the `sitemap.xml` file to the robots' content field.

getRenderingCssClasses

The `getRenderingCssClasses` pipeline is used to gather CSS classes that will be applied on rendering (added to the list of CSS classes on rendering).

refreshHttpRoutes

The `refreshHttpRoutes` pipeline is used to refresh HTTP routes after changes in site configuration.

getVelocityTemplateRenderers

The `getVelocityTemplateRenderers` pipeline is used to add a custom renderer that later on can be used in the *NVelocity* template. This pipeline is defined in the `Sitecore.XA.Foundation.RenderingVariants.config` file. It contains the following processors:

Processor	Description
InitializeVelocityContext	Initializes the pipeline argument objects.
AddTemplateRenderers	Provides two custom tools that format data and time values (dateTool) and numbers (numberTool) in a *NVelocity* template.

Send feedback about the documentation to docsite@sitecore.net.

Walkthrough: Adding search functionality to your page

To enable visitors to quickly find what they are looking for, SXA comes with flexible out-of-the-box search functionality. There are different search renderings available from the Toolbox, for example, renderings that add a search box, renderings that sort or filter the search results, and so on. This makes it easy to set up a simple search solution for your site.

For example, you can add a basic search solution to your page that contains a search query box and lists the results with pagination.

This walkthrough describes how to:

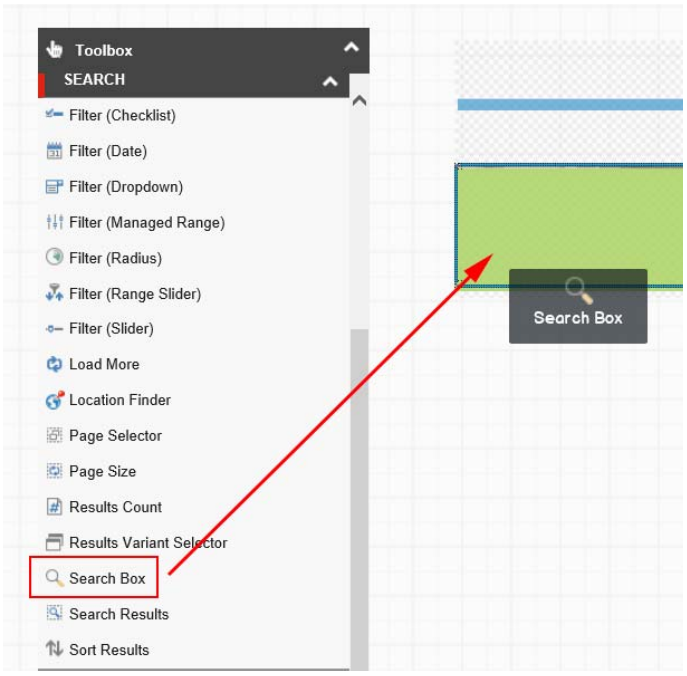
- Add the Search Box rendering
- Add the Search Results rendering
- Add the Page Selector rendering

Add the Search Box rendering

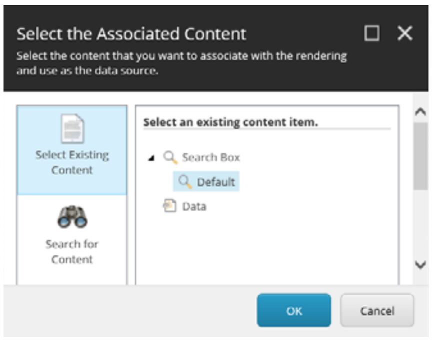
To enable your visitors to search your site, you can add a search box to your pages. By default, the Search Box rendering adds the search text box to the page. You can customize the search box by configuring the item and changing the properties.

To add a search box to your page:

1. From the Toolbox, drag the Search Box rendering to the page.

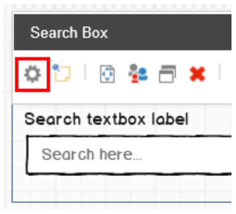


2. In the Select the Associated Content dialog box, for the search box, select the *Default* item.


Select the Associated Content

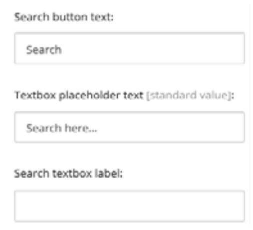
Select the content that you want to associate with the rendering and use as the data source.

3. To change the text of the search box on the page, in the Search Box toolbar, click Edit component configuration item ⚙.



4. In the dialog box, edit the following fields and click OK:
   - Search button text – the text of the rendering's button.
   - Textbox placeholder text – the text in the search box that functions as a hint to describe the expected value.
   - Search textbox label – a title for the search rendering.

   For example, the following fields:

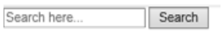


Search button text: Search

Textbox placeholder text [standard value]: Search here...

Search textbox label:

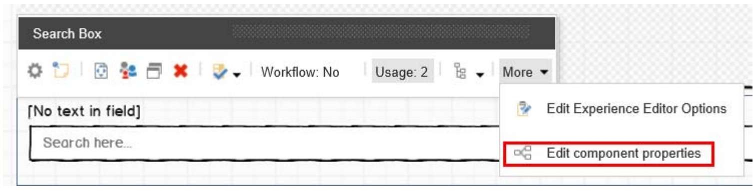
Display the following search box on the page:



Note

If you leave the Search button text field empty, the button does not appear on the page.

5. To change the Search Box rendering properties, for example, to add a search scope or tips for words to search on, in the Search Box, click More, and click Edit component properties:

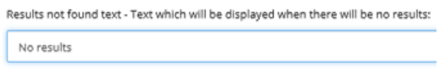


6. In the Control Properties dialog box, for basic search solutions, you can leave the default values. For more complex solutions, you can edit the following fields:
   - Search results signature – enter the unique signature of a specific Search Results rendering to limit the search results. This can be convenient when you have more than one search result rendering on the page.
   - Target signature – if you created a separate search results page, enter the signature of that page here.
   - Search scope – select a scope to limit the search results.
   - Max predictive results count – sets the maximum number of predictive results shown in the drop down. When you leave this field empty, no predictions are shown.
   - Search results page – select the specific search page, if you want to direct to a separate page.
   - Auto complete fields – enter auto complete tips in a comma-separated list.
   - Show search textbox – selected by default. Clear to remove the search text box.

Add the Search Results rendering

For users to view the results of their search, you must add the Search Results rendering on the page. For simple solutions, you can have the search results appear on the same page.

To add the Search Results rendering:

1. To add a search results section to your page, from the Toolbox, drag the Search Results rendering to the page.
2. To change the text of the search box on the page, in the Search Results toolbar, click Edit component configuration item and enter the text that you want to appear when the search returns no results. Click OK.
3. 

Results not found text - Text which will be displayed when there will be no results:

No results

4. To change the Search Results rendering parameters, for example, to determine how many results should be loaded at once, or how the results are sorted, in the Select the Associated Content dialog box, select the *Default* item.
5. In the Control Properties dialog box, edit the following fields to specify how to arrange the search results:
   - Search results signature – enter the unique signature of a specific Search Results rendering to limit the search results. This can be convenient when you have more than one search result rendering on the page and you only want to filter on specific search results. When you leave this field empty, it will filter all Search Results renderings with no signature.
   - Search scope – select a scope to limit the search results.
   - Page size – enter the number of results you want to be loaded by the rendering.
   - Default language filtering – select the language you want to use for the search.
   - Default sort order – select the way the results are sorted on the page. This is only used when the Sort Results rendering is not used.

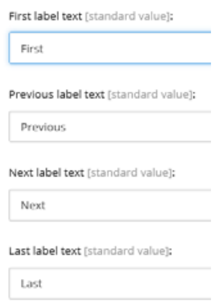
Add the Page Selector rendering

With the Page Selector rendering, you can determine how to display the results pages. The number of pages displayed depends on the default page size configured in the Search Results rendering or, if the Page Size rendering is used, on the page size configured in the Page Size rendering.

To add the Page Selector rendering:

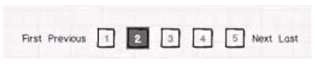
1. From the Toolbox, drag the Page Selector rendering to the page.
2. In the Page Selector toolbar, click Edit component configuration item .
3. In the dialog box, fill in the fields and click OK.

   For example, the following fields:



First label text [standard value]:
First
Previous label text [standard value]:
Previous
Next label text [standard value]:
Next
Last label text [standard value]:
Last

   Appear as:



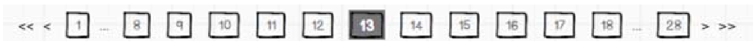
4. To change the properties of the Page Selector rendering, for example, to determine how many pages are displayed in buttons, in the Search Box, click More, and click Edit component properties.
5. In the Control Properties dialog box, edit the following fields and then click OK:

  - Search results signature – if you want to filter on specific Search Results renderings, enter the signature(s) (separated by a coma) of the Search Results rendering(s) that you want to search on.
  - Collapsed mode threshold – if you expect results on many pages, you can replace some of these pages by dots.

For example, if you enter 10 in the Collapsed mode threshold field:

Collapsed mode treshold [shared]:
10

The following is displayed for the search results:



Send feedback about the documentation to docsite@sitecore.net.

Walkthrough: Building a new rendering

The SXA toolbox contains default renderings for simple text, images, videos, social media plugins, and so on. You can also create a new rendering.

This walkthrough describes how to:

- Create a rendering
- Create the controller and action method
- Inject a repository into a controller

Create a rendering

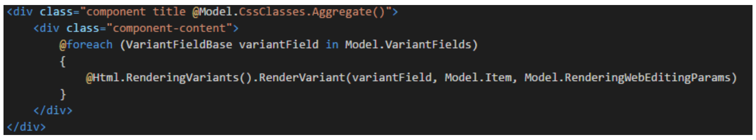
SXA renderings are Sitecore controller renderings and SXA uses the same basic structure for all renderings. The wrappers are the same for every rendering and always contain:

- The `component title` element.
- The `@Model.CssClasses.Aggregate()`,which is necessary for Creative Exchange.
- The content: `<div class="component-content">` – the content can be different for every rendering as long as you use the same wrapper.

To create a rendering:

- In the rendering definition item, specify what action Sitecore takes to render the component.

For example, for the title rendering:


```
<div class="component title @Model.CssClasses.Aggregate()">
    <div class="component-content">
        @foreach (VariantFieldBase variantField in Model.VariantFields)
        {
            @Html.RenderingVariants().RenderVariant(variantField, Model.Item, Model.RenderingWebEditingParams)
        }
    </div>
</div>
```

Create the controller and action method

Every SXA rendering needs a controller. SXA enables you to inherit from a base SXA controller. It provides useful properties and logic to automatically resolve the view name and provide the model (*Rendering*, *PageContext*). It is best practice to use these base classes when you build a new rendering or extend an existing rendering.

To create the controller and the action method:

1. In an ASP.NET MVC Web application project in Visual Studio, create a new controller.

   In SXA, controllers are kept fairly simple and most of the logic is implemented in repositories. To make the controllers use the repositories, you must create controller renderings with a construction injection.

2. Use the `StandardController` base class to implement the standard `Index()` action method that automatically resolves the view name based on the rendering item name.

   For example:

```
public class SimpleComponentController: StandardController

{

    private readonly ISimpleComponentRepository _repository;

    public SimpleComponentController(ISimpleComponentRepository repository)

    {

        _repository = repository;

    }

    protected override object GetModel()

    {

        return _repository.GetModel();

    }

}
```

Inject a repository into a controller

SXA uses repositories to provide the model for the MVC views. Therefore, when you finish the controller, you must inject the repository into the controller.

- To inject a repository into a controller, use the SXA `ServiceLocator` class:

```
public class SimpleComponentController: StandardController
{
    private readonly ISimpleComponentRepository _repository;
    public SimpleComponentController(ISimpleComponentRepository repository)
    {
        _repository = ServiceLocator.Current.Resolve<ISimpleComponentRepository >();
    }
    protected override object GetModel()
    {
        return _repository.GetModel();
    }
}
```

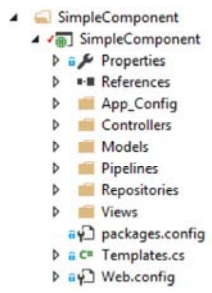
SXA provides base repositories. For example, the `VariantsRepository` and the `ListRepository` provide useful properties and make implementing your own repository a lot easier. Another example is the `ModelRepository` base class that provides useful properties such as:

- PageContext – represents information required to service an individual request for a specific device, containing information about the HTTP response under construction. The `PageContext` contains properties that expose the requested item, device, and the `PageDefinition` object created for the request.
- IsEdit – checks if Edit mode is on.
- IsControlEditable – specifies whether the rendering should be editable.
- CssClasses – lists the CSS classes that are applied on rendering HTML.

All repositories should have their own interface, even when a repository is empty and is just using base interface methods. This is because it is used later on when registering the Dependency Injection container:

```
public interface ISimpleComponentRepository: IModelRepository
{
}
And here is an example of simple repository:
    public class SimpleComponentRepository : ModelRepository, ISimpleComponentRepository
    {
        public override IRenderingModelBase GetModel()
        {
            SimpleComponentModel model = new SimpleComponentModel();
            FillBaseProperties(model);
            model.Title = GetTitle();
            return model;
        }
        private string GetTitle()
        {
            return PageContext.Current[Templates._Title.Fields.Title];
        }
    }
```

When you have finished building your new rendering, for example, for a simple rendering, the project structure could look like this:



And the view:

```
@model Sitecore.SXA.Startup.Feature.SimpleComponent.Models.SimpleComponentModel
<div class="component simple-component @Model.CssClasses.Aggregate()">
    <div class="component-content">
        <h1>You are on @Model.Title page</h1>
    </div>
</div>
```

Note

Most SXA renderings are designed for reusability and pull data from data source items. This means that the content they display is not bound to the page on which they appear but is stored in data source items.

To add data source logic to the rendering:

if (DataSourceItem == null) { ... }

Send feedback about the documentation to docsite@sitecore.net.

Enable and configure the Asset Optimizer

The Asset Optimizer is a module that optimizes CSS styles and JS scripts. When it is enabled in a production environment, the Asset Optimizer improves overall site performance by reducing the amount of data that needs to be transferred. Sitecore administrators can enable this module either globally for the entire Sitecore instance or locally for selected tenants.

Note

It is best practice to disable the Asset Optimizer in development environments and to enable it in production environments.

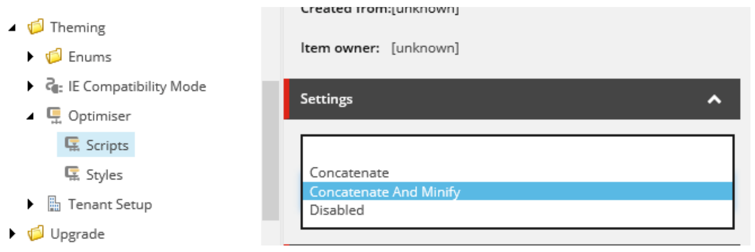
This topic describes how to:

- Enable the Asset Optimizer globally
- Change the optimization settings for a specific site

Enable the Asset Optimizer globally

To enable the optimizer globally:

- In the Content Editor, navigate to `/sitecore/system/Settings/Foundation/Experience Accelerator/Theming/Optimiser` and for both Scripts and Styles select *Concatenate and Minify* to minify all files and concatenate them into one file.



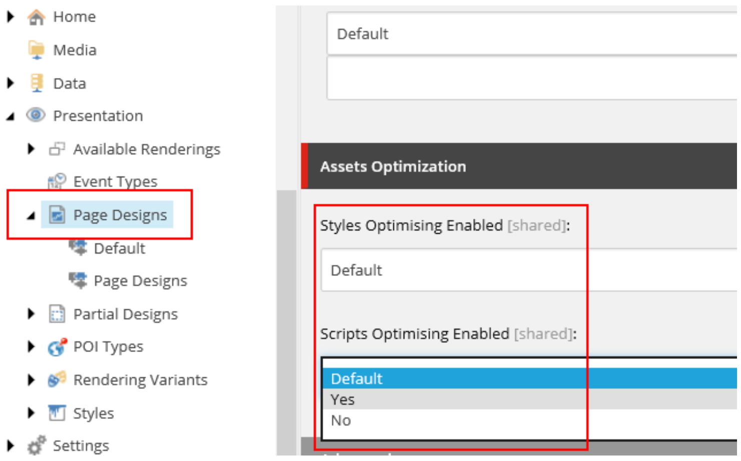
Note

Changes take effect immediately after the item is saved. Anonymous users might not see the changes until you publish.

Change the optimization settings for a specific site

To change the optimization settings for a specific site:

1. In the Content Editor, navigate to `sitecore/content/…/SITE_NAME/Presentation/Page Designs`.
2. In the Asset Optimization section, in the Styles Optimizing Enabled and Scripts Optimizing Enabled fields, to override styles and scripts optimization settings, select:
   - *Default* – to inherit global settings
   - *Yes* – to always enable optimization for this site.
   - *No* – to always disable optimization for this site.

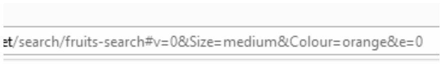


SXA search

To enable visitors to quickly find what they are looking for, SXA comes with flexible out-of-the-box search functionality. When visitors search a site and it returns too many results, they can quickly refine the search by filtering.

You can create a search solution by adding different search renderings to your page. For example, a search box, different filters for refining the search, and the search results. Because SXA uses events to communicate between renderings, when you change a filter for the search, automatically all of the renderings are informed and the search triggers the new search request.

All the filter parameters and values, such as current page, location, and sort order, are stored in the browser URL after the hash sign. For example, a search on a site that lists tropical fruits, with search filters for size and color might look like this:



```
et/search/fruits-search#v=0&Size=medium&Colour=orange&e=0
```

Note

SXA supports both Lucene and Solr search engines. The search engines are used for searching in the content databases, as well as for searching in a number of operational databases that Sitecore uses for collecting analytics data, test data, and so on.

SXA comes with several standard search renderings. You can combine these renderings for a search solution that suits your site. All rendering items have two types of configurations: data source items that are used to store rendering texts (to allow for translations) and other rendering properties, such as target signatures, scopes, and facets.

Note

Developers can create other search renderings.

Facets

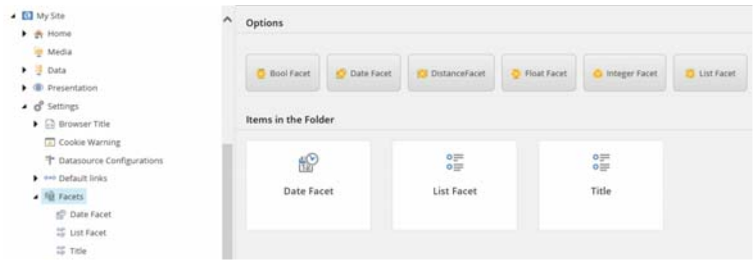
A facet is a way of refining search results by categorizing the items returned by the search. For example, for a blog search page, all blogs contain fields such as: author, date, and language. Based on these fields, you can create facets to allow visitors to use them as filters.

You can add facets to your site in: `sitecore/content/TENANT__GROUP_NAME/TENANT_NAME/SITE_NAME/Settings/Facets/`

Note

The Sitecore platform facets are stored in `/sitecore/system/Settings/Buckets/Facets`

The facet types are: Bool Facet, Date Facet, Distance Facet, Float Facet, Integer Facet, and List Facet.



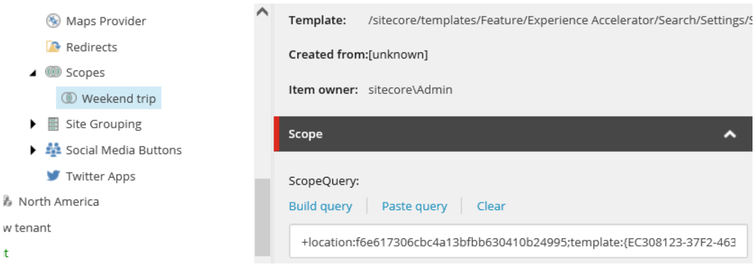
Note

Not all facets apply to all renderings. For example, the Filter (Dropdown) rendering cannot use the Date Facet because it only applies to the Filter (Date) rendering.

Scopes

Search scopes can be used to limit search results based on conditions. Search scopes are stored in: `/sitecore/content/TENANT__GROUP_NAME/TENANT_NAME/SITE_NAME/Settings/Scopes/`

For a new scope, you build a search query that enables you to add several conditions. For example, for a search scope for weekend trips on a travel site, you can combine the location of the trips with the template.



Template: /sitecore/templates/Feature/Experience Accelerator/Search/Settings/S

Created from:[unknown]

Item owner: sitecore\Admin

ScopeQuery:

+location:f6e617306cbc4a13bfbb630410b24995;template:{EC308123-37F2-463

The Asset Optimizer

In a production environment, the SXA Asset Optimizer improves the end user experience by optimizing CSS styles and JavaScript, and reducing the amount of data that needs to be transferred.

In SXA, renderings and other front-end functionality are styled or scripted in files stored in themes that tend to have the front-end capabilities broken down on a component level. This makes front-end development easier, but the number of files that need to be served may not be acceptable in a production environment. Having a large number of small files causes performance to deteriorate and can also cause issues in older browsers. Therefore, we recommend that you enable the Asset Optimizer in production environments. Sitecore administrators can enable this module either globally for an entire Sitecore instance or locally for selected sites.

The Asset Optimizer groups assets together according to the following rules:

- Assets that come from the same folder.
- Assets that have the same extension.

Assets from each group are concatenated into one asset and cached on the server. References to individual assets in the markup are replaced with one reference for each group, so that:

- The reference path starts with the same folder path as the initial assets.
- The reference path ends with the same extension as the initial assets.
- The link tag has the same value of media attribute as the initial assets.

The following code sample is a piece of sample markup for just one theme used on the page where each asset is referenced separately:

```
<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-location-service.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-ajax.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-base-view.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-box.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-facet-data.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-facet-daterange.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-facet-dropdown.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-facet-managed-range.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-facet-range-slider.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-facet-slider.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-load-more.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-location-filter.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-page-selector.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-page-size.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-query.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-radius-filter.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-results.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-results-count.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-results-filter.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-sort.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-url.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-variant-filter.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-service.js"></script>

<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/component-search-router.js"></script>
```

If the Asset Optimizer is enabled, all the links in the previous HTML code sample are gathered in a separate theme file:

```
<script src="/sitecore/shell/-/media/Base-Themes/SearchTheme/Scripts/optimized-min.js?t=20160908T094830Z"></script>
```

Note

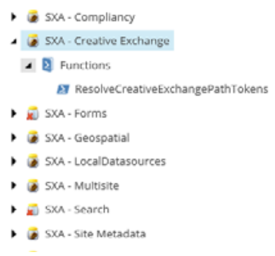
Even with the Asset Optimizer enabled, you can always request a page with the assets broken down into the original files by appending the following parameter to your request URL:

aodisabled=1

The SXA script library

SXA includes a number of PowerShell scripts to automate the most common tasks. The Sitecore PowerShell Extensions (SPE) module provides a command line and a scripting environment and enables you to use PowerShell from within Sitecore. In this way, you can run commands and write scripts according to Windows PowerShell syntax. Every SXA module that uses SPE has its own script library in: `sitecore\System\Modules\PowerShell\Script Library`.

The naming convention for these scripts is: *SXA - Module name*:

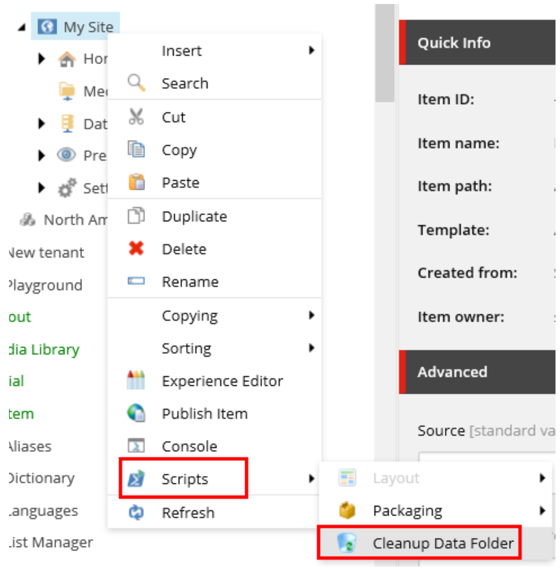


Note

PowerShell scripts can be used to automate tasks that you find yourself doing on a regular basis. To write your own scripts, or view code of existing scripts, use the PowerShell Integrated Scripting Environment (ISE). You can access this tool on the Sitecore Launchpad:

The types of SXA scripts vary from cleanup scripts to cmdlts that add insert options to items:

- Scaffolding – scripts used during scaffolding to automate the process of site/tenant creation.
- Context Menu – scripts that are available for editors using the Content Editor. For example, the cleanup data sources script:



- Cmdlets – a lightweight command that is used in SPE and can be reused by other developers.
- Insert Item – scripts that extend the Insert section in the Content Editor.

The following table describes the available SXA script modules, their functions, descriptions, and type of script:

Module	Function	Description	Type of script
SXA – Compliancy	AddCookieWarning	Inserts the Cookie Warning rendering to a partial design.	Scaffolding
SXA – Creative Exchange	ResolveCreativeExchangePathTokens	Resolves the path tokens in the Creative Exchange `FileStorage` provider definition item to enable unique paths under the *Data* folder. This prevents different sites from overwriting the folder content.	Scaffolding
SXA – Forms	Set forms folder	Sets the WFFM Forms root folder on the site definition item. This enables every site to have its own forms root folder (by default – in WFFM – there is only one global Forms root folder).	Scaffolding
SXA – Geospatial	Assign POI items	Assigns a default rendering variant for POI types. This enables each new site to have the same rendering variants with site-specific IDs.	Scaffolding
SXA – LocalDatasources	Cleanup Data Folder	Searches and cleans unused data sources in the *Page Data* folder.	Context Menu

SXA – Multisite	Add Content fields	Decorates new *Tenant* templates with additional fields. Adds Content and Title fields to a *Page* template.	Scaffolding
	Home base template	Adds a base template to the *Home* template under *Tenant* templates. Adds a *Page* template as a base template.	Scaffolding
	Set home item title field	Sets the home item title after the site is created.	Scaffolding
	Set start item	Sets the start item on the site definition item. By default, the start item is a home item that is the root of all pages.	Scaffolding
SXA – Search	Set default sorting facet	Assigns the Title facet as the default facet to sort data sources.	Scaffolding
SXA – Site Metadata	AddBrowserTitle	Inserts the Browser Title rendering to metadata partial design.	Scaffolding
	AddCustomMetadata	Inserts the CustomMetadata rendering to metadata partial design.	Scaffolding
	AddFavicon	Inserts the Favicon rendering to metadata partial design.	Scaffolding
	AddOpenGraphMetadata	Inserts the OpenGraph rendering to metadata partial design.	Scaffolding
	AddSeoMetadata	Inserts the SeoMetadata rendering to metadata partial design.	Scaffolding
	AddStickyNotes	Inserts the StickyNotes rendering to metadata partial design.	Scaffolding
	AddTwitterMetadata	Inserts the TwitterMetadata rendering to metadata partial design.	Scaffolding
	AddViewport	Inserts the Viewport rendering to metadata partial design.	Scaffolding
SXA – Scaffolding	MultisiteContext/Get-DataItem	The PowerShell wrapper for the `IMultiSiteContext` interface. Resolves the *Data* folder for the current site.	Cmdlets
	MultisiteContext/Get-SettingsItem	The PowerShell wrapper for the `IMultiSiteContext` interface. Resolves the *Settings* folder for current site.	Cmdlets
	MultisiteContext/Get-SiteItem	The PowerShell wrapper for the `IMultiSiteContext` interface. Resolves the *Site* item for the current site.	Cmdlets
	MultisiteContext/Get-SiteMediaItem	The PowerShell wrapper for the `IMultiSiteContext` interface. Resolves the Virtual Media Library for the current site.	Cmdlets
	MultisiteContext/Get-TenantItem	The PowerShell wrapper for the `IMultiSiteContext` interface. Resolves the *Tenant* item for the current site.	Cmdlets
	PresentationContext/Get-PageDesignsItem	The PowerShell wrapper for the `IPresentationContext` interface. Resolves the *Page Designs* folder for the current site.	Cmdlets

PresentationContext/Get-PartialDesignsItem	The PowerShell wrapper for the `IPresentationContext` interface. Resolves the *Partial Designs* folder for the current site.	Cmdlets
Add-BaseTemplate	Cmdlet that adds a base template.	Cmdlets
Add-FolderStructure	Cmdlet that recreates the folder for a given path structure if it does not exist.	Cmdlets
Add-InsertOptionsToItem	Cmdlet that adds `InsertOptions` (Masters) to an item.	Cmdlets
Add-InsertOptionsToTemplate	Cmdlet that adds `InsertOptions (Masters)` to a *Template* item.	Cmdlets
Get-PartialDesign	Returns the first Partial Design in the *Partial Designs* folder.	Cmdlets
Test-ItemIsPartialDesign	Checks whether the item inherits from `PartialDesignTemplate`.	Cmdlets
Test-ItemIsSiteDefinition	Checks whether the item inherits from `SiteDefinitionTemplate`.	Cmdlets
New-Site	Contains all the functions used to create a new site.	Cmdlets
New-Tenant	Contains all the functions used to create a new tenant.	Cmdlets

Site	Adds script to the Insert options in a context menu that lets users create a new site. Appears when the following rules are met:
	
	`Rule 1`
	
	`where the item template is SiteFolder`
	
	`Rule 2`
	
	`where the item template is Tenant`
		Insert item

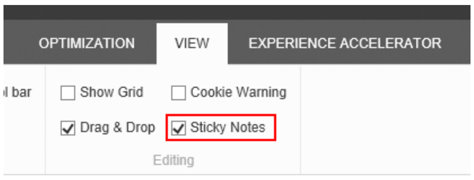
Tenant	Adds script to the Insert options in a context menu that lets users create a new tenant. Appears when the following rules are met:
	
	`Rule 1`
	
	`where the itemID is equal to Content`
	
	`Rule 2`
	
	`where the item template is Tenant Folder`
		Insert item

Send feedback about the documentation to docsite@sitecore.net.

Add a sticky note

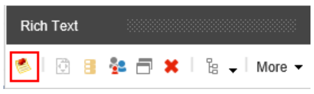
You can add sticky notes to your SXA pages. A sticky note can be convenient when you are working on a website with a group of people so that you can share information on the site. For example, you can remind your co-worker to provide a piece of text or inform everyone that you will change an image later. Sticky notes are not visible when you publish your site.

To view sticky notes, you must select the Sticky Notes check box.



To add a sticky note to a page:

- Click the rendering to which you want to add the sticky note, and on the floating toolbar click Insert a sticky note, and then enter the relevant message.



You can change the color of the sticky note, for example, if you want yellow notes for text and red notes for media.

To see the rendering that a note belongs to, click the magnifier, and the rendering is highlighted in the same color as the note.

To minimize or expand the sticky note, use the - and + icons.

Send feedback about the documentation to docsite@sitecore.net.

Add, edit, and delete a rendering

SXA comes with a library of predefined renderings to make page design easy. You can construct your pages by dragging renderings from the Toolbox to the page.

This topic outlines how to:

- [Add a rendering]()
- [Edit a rendering]()
- [Delete a rendering]()

Add a rendering

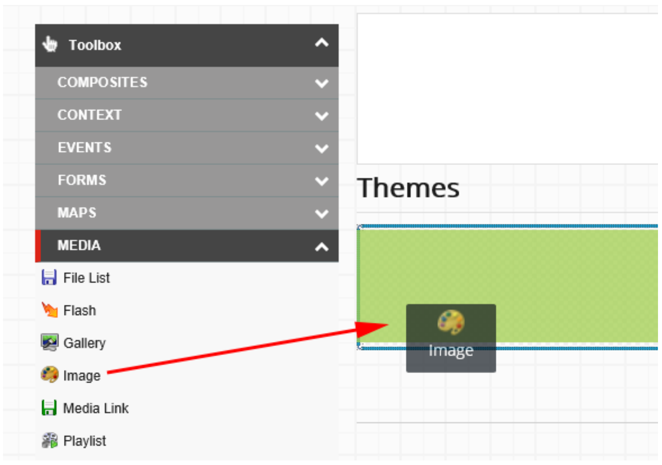
In the Experience Editor, you can add a rendering to the page by dragging it from the Toolbox.

To add a rendering to the page:

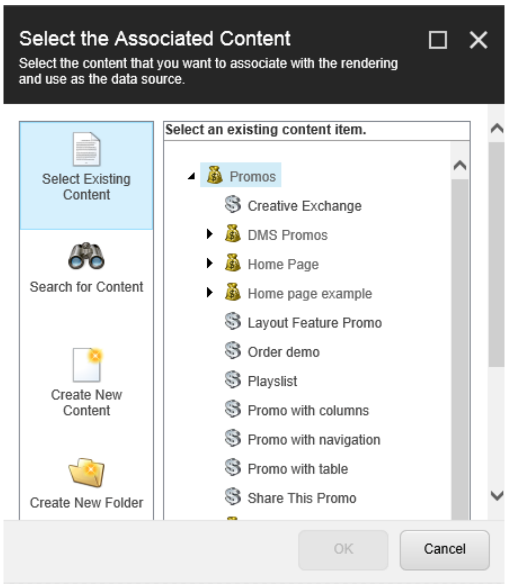
1. Open the Toolbox and find the relevant rendering. When you click and start dragging a rendering, the placeholders where you can drop the rendering light up in blue.

   Alternatively, you can use the touch panel to drag renderings to the page with your finger or you can click the Rendering icon, on the HOME tab.

2. Position the rendering above the desired placeholder, and when the placeholder lights up in green, drop the rendering on the page.



3. Depending on the rendering you choose, you may need to [select a content item](). In the Select The Associated Content dialog box, select the content item you want and then click OK.



Once your rendering is on the page, you can move it to a different placeholder without returning to the toolbox. Click the on the floating toolbar and move the rendering to a different placeholder.

Edit a rendering

There are certain renderings that are editable and others that you cannot edit. If you can edit a rendering, a floating toolbar appears.

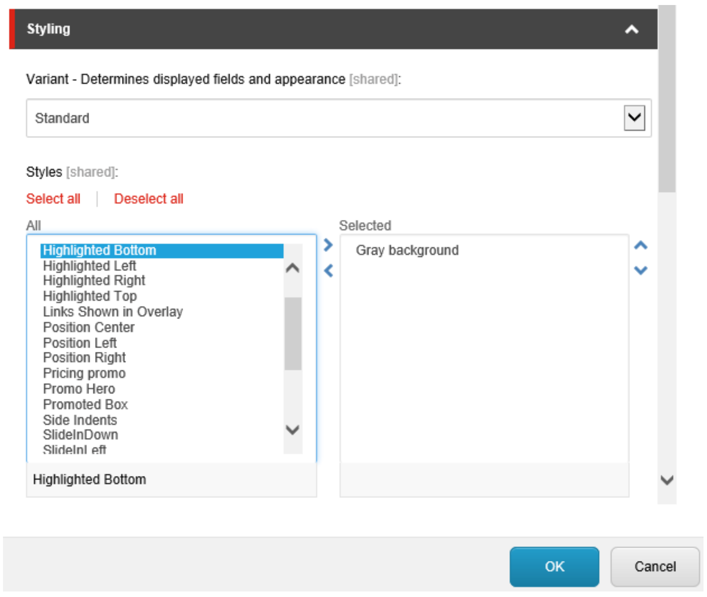
To edit a rendering:

1. Click the rendering that you want to edit and in the floating toolbar, click Edit the Component Properties . If the rendering is a text, you can edit it directly on the page.
2. In the Control properties dialog, specify the rendering behavior and/or styles that you want. The available options depend on the type of rendering. For all renderings, you can change the style settings. For example, you can change the paragraph style of the title or change the dimensions of the preview icon.

   Note

   Do not change the Placeholder and Data Source properties. Changing these properties can cause the rendering to disappear or may lead to other unexpected behavior.

3. To change the style, in the Control properties dialog go to the Styling section, select the style you want, click the right arrow and then click OK.



4. Click Publish to publish the data source assigned to the rendering. It will not publish the site.

Delete a rendering

Occasionally, you might want to remove a rendering from a page. For example, because a promotion offer is no longer valid.

To delete a rendering from a page:

- Click the rendering that you want to delete and in the floating toolbar, click Remove rendering.

  Note

If you have created a complex page layout with lots of column and/or row splitters and you try to delete nested renderings, you may receive a message asking which rendering you want to remove. The section that will be removed after clicking Remove rendering is highlighted. If you click OK, all of the listed components will be removed.

Send feedback about the documentation to docsite@sitecore.net.

Improve page SEO

Everyone wants their site to show up on the first page of a search result. Fortunately, SXA enables you to easily follow the best practices for search engine optimization (SEO) and improve the way your site is ranked in search results.

This topic outlines how to:

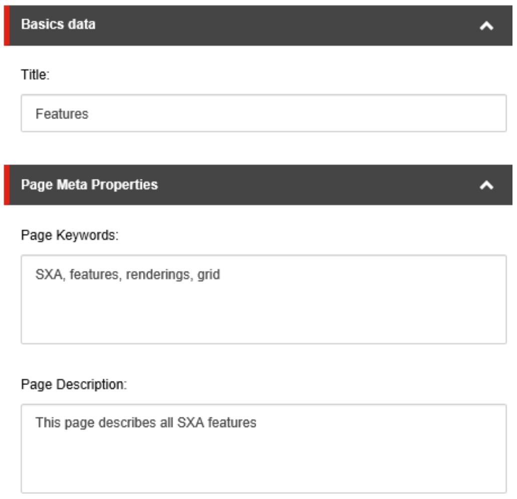
- Edit page SEO information
- Edit a page to improve links for social media
- Prioritize a page in the search engine sitemap

Edit page SEO information

Although search engine ranking is continuously evolving, the secret to a good ranking is to make pages relevant. Search engines use your page titles, image captions, and keywords when ranking pages and therefore it is important that you use meaningful, descriptive, and relevant words for these fields. With SXA, you can edit the metadata for your pages in the Experience Editor.

To edit the SEO information of a page:

1. In the Experience Editor, go to the page you want to improve.
2. On the ribbon, on the Experience Accelerator tab, click SEO.
3. In the dialog box, edit the page title, page keywords, and page description.



Basics data

Title: Features

Page Meta Properties

Page Keywords: SXA, features, renderings, grid

Page Description: This page describes all SXA features

4. Click OK.

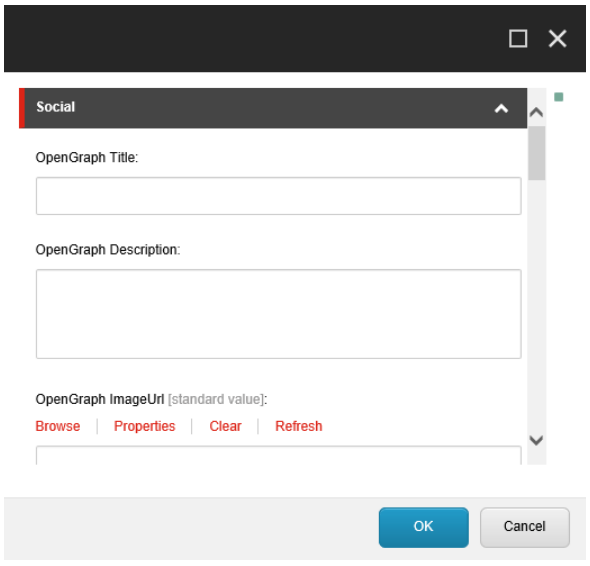
Edit a page to improve links for social media

When social media is the major driver of your website's traffic, it is a good idea to add Open Graph metadata to the pages with social media content. Adding Open Graph tags to your website does not directly affect your on-page SEO, but it improves the performance of your links in social media.

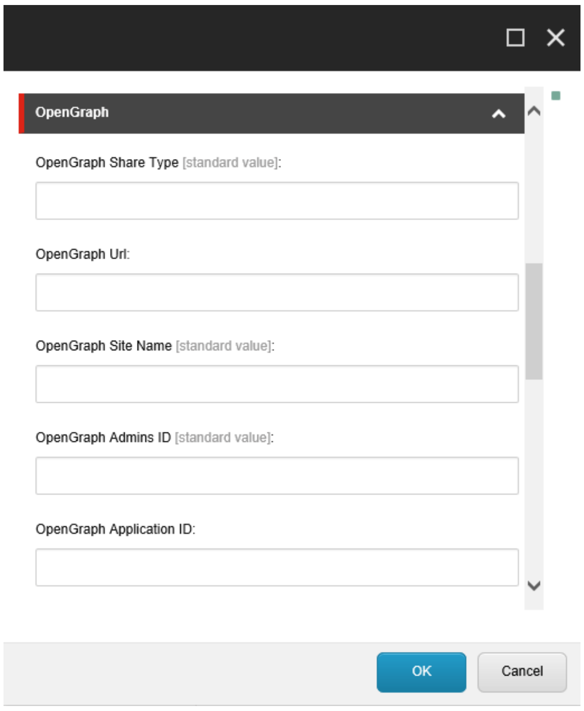
Facebook introduced Open Graph to promote integration between Facebook and other websites. Open Graph is now used by most social media and it allows you to control the way information travels from a third-party website to a social media site when a page is shared. In order to make this possible, information is sent using Open Graph meta tags in the <head> part of the website’s code.

To edit page settings related to social networks:

1. In the Experience Editor, go to the page you want to improve.
2. On the ribbon, on the Experience Accelerator tab, click Social.
3. In the dialog box, in the Social section, edit the following fields that the social network uses when the content/page is shared:
    - OpenGraph Title – enter the title for the content/page.
    - OpenGraph Description – enter a description for the content/page.
    - OpenGraph ImageUrl – insert the image for the content/page.


Social
OpenGraph Title:
OpenGraph Description:
OpenGraph ImageUrl [standard value]:
Browse | Properties | Clear | Refresh
OK
Cancel

4. In the OpenGraph section, to improve the performance of your social media links, edit the following fields:
   - OpenGraph Share Type – enter a description of the content type, for example: video, article, or website.
   - OpenGraph Url – enter the URL of the content.
   - OpenGraph Site Name – enter the name of the site that is shared.
   - OpenGraph Admins ID – enter the user ID, or list of user IDs for Facebook if your page is a Facebook app.
   - OpenGraph Application ID – enter the single app ID for Facebook.

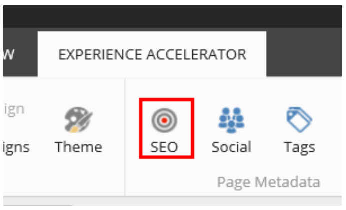

OpenGraph
OpenGraph Share Type [standard value]:
OpenGraph Url:
OpenGraph Site Name [standard value]:
OpenGraph Admins ID [standard value]:
OpenGraph Application ID:
OK
Cancel

Prioritize a page in the search engine sitemap

To help search engines determine how often to check a specific page on your site, you can change the priority and frequency settings for each page individually. You can indicate how often you update the page and how important the page is compared to the rest of your site. For example, a homepage changes daily, a blog post changes weekly, and an archived post changes yearly.

To configure sitemap behavior for a specific page:

1. In the Experience Editor, go to the page that you want to edit.
2. On the ribbon, on the Experience Accelerator tab, click SEO.

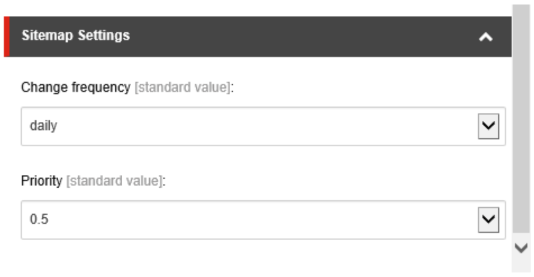


3. In the dialog box, in the Sitemap Settings section, in the Change frequency field, specify how often the page changes its content.

Note

If you want to exclude a page from the sitemap, set the Change Frequency to *do not include*.

4. In the Priority field, specify a number between 0.1 and 1.0 (with 0.1 being the lowest and 1.0 the highest) to indicate the importance of the page.


Sitemap Settings
Change frequency [standard value]:
daily
Priority [standard value]:
0.5

5. Click OK and then publish the page to update the sitemap.

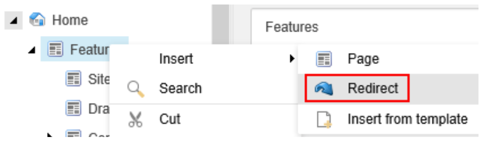
Send feedback about the documentation to docsite@sitecore.net.

Redirect a URL

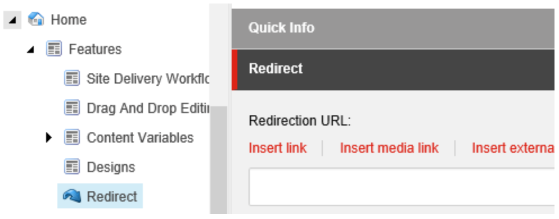
A redirect is a way to send both users and search engines to a different URL from the one they originally requested. For example, to redirect visitors who enter `namewebsite.com/home-a` in their browser to `namewebsite.com/home-b`. This is very useful when you want to redirect a specific page to a new location, change the URL structure of a site so that it appears higher up in the navigation, or even redirect users to another website entirely.

To add a redirect item:

1. In the Content Editor, right-click on the page where you want to specify your redirect, and click Insert, Redirect.



2. Enter a name for the redirect item and click OK.
3. In the Redirect section, enter the URL that you want to direct to and save your changes.



Send feedback about the documentation to docsite@sitecore.net.

Select, modify, and create associated content

Most SXA renderings are designed for reusability and pull data from data source items or associated content. The content they display is not bound to the page on which they appear but is stored in data source items. When you add a rendering to a page, you can select an existing data source or create a new data source item. This gives you full control over the content architecture, naming conventions, and the level of reusability that you want.

This topic outlines how to:

- Select associated content
- Modify associated content
- Create associated content

Select associated content

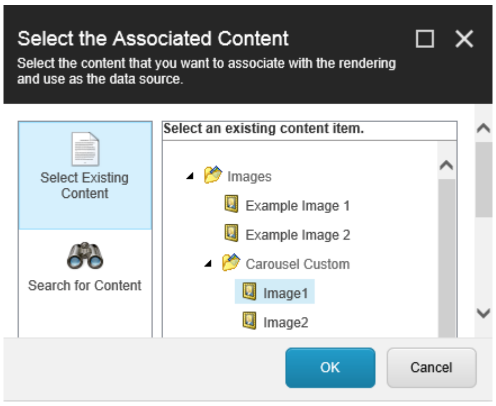
When you add a reusable rendering to the page in the Experience Editor, the items available for selection depend on the rendering you selected, which prevents you from associating data source items that do not match the rendering's requirements.

Note

You cannot select associated content for the following renderings: non-reusable renderings (such as Rich Text), renderings that are shared across the entire site (Login), renderings that display data from the current page (Page Content), and renderings that do not display any content (Divider).

To select associated content:

1. In the Experience Editor, from the toolbox drag a rendering that supports reusable content to the page.
2. In the Select the Associated Content dialog box, click the content item that you need and click OK.

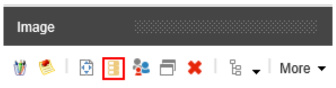

Select the Associated Content
Select the content that you want to associate with the rendering and use as the data source.
Select Existing Content
Search for Content
Select an existing content item.
Images
Example Image 1
Example Image 2
Carousel Custom
Image1
Image2
OK
Cancel

Modify associated content

You might want to change an associated item after it is placed on a page. For example, if you need to replace an image on the page.

To modify associated content:

1. In the Experience Editor, on the rendering's floating toolbar, click Associate a content item.



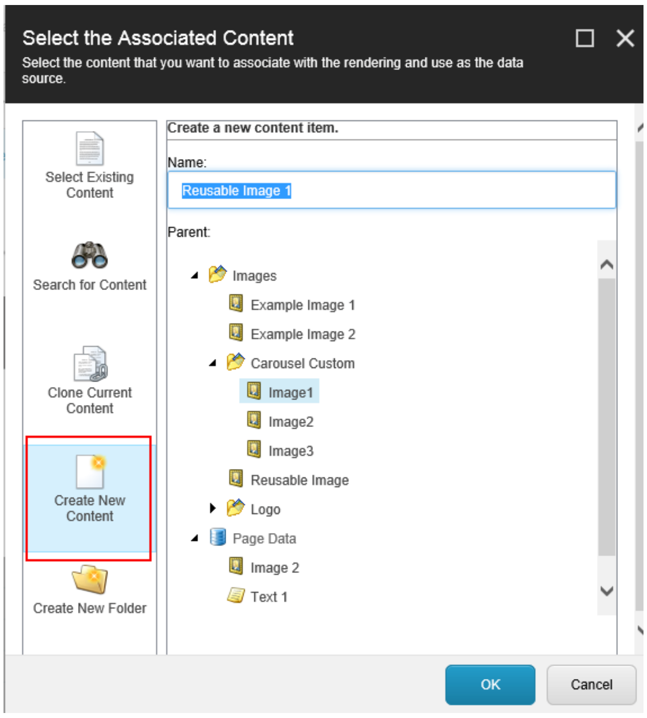
2. In the Select the Associated Content dialog box, select the item that you want to associate with the rendering and click OK.

Add a data source

If you the content item you need is not available, you can add a new data source item.

To add a data source:

1. In the Select the Associated Content dialog box, click Create New Content.



2. Select the place in the tree where you want to create the new data source, enter a name for it and click OK.

   Note

   If you create the data source under the Page Data node it will be a local data source that is stored as a subitem of the page item. Any changes you make to local data sources will only affect the page you are working on. If you want to be able to reuse the data source and manage it globally, select a different place in the tree.

   Now you have created a new data source item that is automatically associated with the rendering.

Send feedback about the documentation to docsite@sitecore.net.

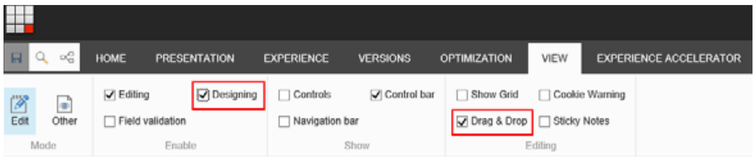
The Toolbox

Use the SXA Toolbox in the Sitecore Experience Editor to quickly build your webpages by [dragging renderings]() directly to where you need them. There are default renderers available for simple text, images, videos, social media plugins, and so on. To make it easier to find the rendering you need, SXA organizes all renderings in categories, such as Page content, Page structure, Navigation, Forms, and so on. Collapse or expand these rendering categories, so you do not have to scroll through a lengthy list of all the available renderings.

Note

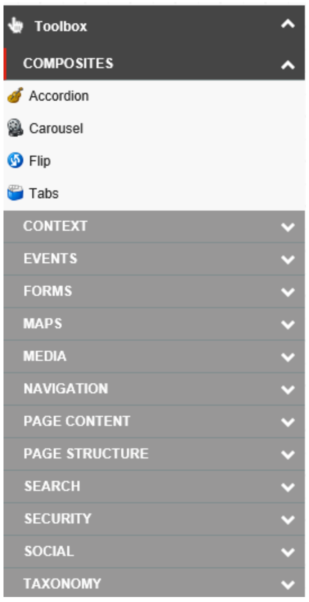
To change the structure of the categories, contact your Administrator.

If you can't find the toolbox, make sure you selected the Designing and Drag & Drop check boxes on the View tab.

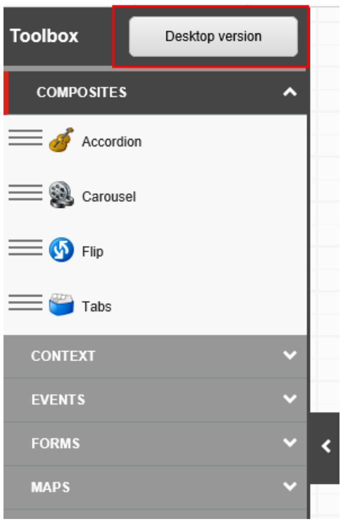


☑ Editing ☑ Designing ☐ Field validation ☐ Controls ☐ Navigation bar ☑ Control bar ☐ Show Grid ☑ Drag & Drop ☐ Cookie Warning ☐ Sticky Notes

To position the toolbox wherever you want on your page, click at the top of the Toolbox and drag it to the desired location.



If you work on a touch device, such as tablet or a touch-enabled laptop, by default the Toolbox opens in touch mode. If you work on a touch-enabled laptop but prefer working in desktop mode, you can switch to desktop mode by clicking Desktop version.


Toolbox
Desktop version
COMPOSITES
Accordion
Carousel
Flip
Tabs
CONTEXT
EVENTS
FORMS
MAPS