



SAP Cloud Platform Portal - Neo

Generated on: 2019-04-02

SAP Cloud Platform | Cloud

PUBLIC

Warning

This document has been generated from the SAP Help Portal and is an incomplete version of the official SAP product documentation. The information included in custom documentation may not reflect the arrangement of topics in the SAP Help Portal, and may be missing important aspects and/or correlations to other topics. For this reason, it is not for productive use.

For more information, please visit the [SAP Help Portal](#).

Freestyle Sites

Administrators build freestyle sites to provide end users with a single point of access to all the information they need to carry out their daily tasks.

What is a freestyle site?

Freestyle sites are based on templates. These templates are either provided out-of-the-box by SAP, or your developers can create them in SAP Web IDE. Whatever the case, you as an administrator can use these templates to create your freestyle sites. You can also accelerate your portal implementation by downloading code samples that are available in the SAP Cloud Platform Portal GitHub repository. For more information, see [Using Accelerators to Build Freestyle Sites](#).

Quick links

Here are some quick links to the tasks and information that you will probably want to do when creating your freestyle site:

iNote

Hover over each shape for a description; click for more information.

Developer Story

Create site templates

Create page templates

Create web content

End to end scenario

Admin Story

Build sites

Manage your pages

Manage your apps

End to end scenario

Web Content Editor Story

Access the editing tool


Edit web content




Translate web content

End to end scenario

Who uses the Portal?

Here are the main personas who work with the Portal and what their central tasks are:

<div>Developer</div> <div></div>	<ul style="list-style-type: none">Creates basic templates defining layout of sites.Extends templates with page templates, pages, and widgets.Customizes out-of-the-box templates.
<div>Administrator</div>	<ul style="list-style-type: none">Creates sites using site templates.

	<ul style="list-style-type: none"> • Extends sites by adding pages, apps, and widgets. • Defines site layout and settings. • Publishes the site.
Web Content Editor 	<ul style="list-style-type: none"> • Accesses the Web Content Editor tool. • Edits web content. • Publishes the edited web content.
End User 	Uses the site as a single point of access to apps and pages.

Using Accelerators to Build Freestyle Sites

Administrators and developers can accelerate their portal implementation by downloading code samples that are available in the SAP Cloud Platform Portal GitHub repository.

What are accelerators?

Accelerators are code samples of site and page templates, applications, widgets, plugins, and more. They are intended to accelerate your portal implementation by providing a starting point and acting as a reference for development of custom content when you build your freestyle sites.

How can I find them?

Accelerators are available on [GitHub](#) ➦ .

You can download the content and import it to SAP Web IDE, where you can extend it and then deploy to your SAP Cloud Platform subaccount. Once deployed, the content is available in the Admin Space and Site Designer of your Portal.

Find the latest releases from: [Downloads](#) ➦

Developer Guide

Welcome to the SAP Cloud Platform Portal developer guide.

Hover over each shape for a description, click for more details:



About site templates and components



End to end development flow



Developing web content widgets



Converting apps to widgets



The project structure



Advanced actions



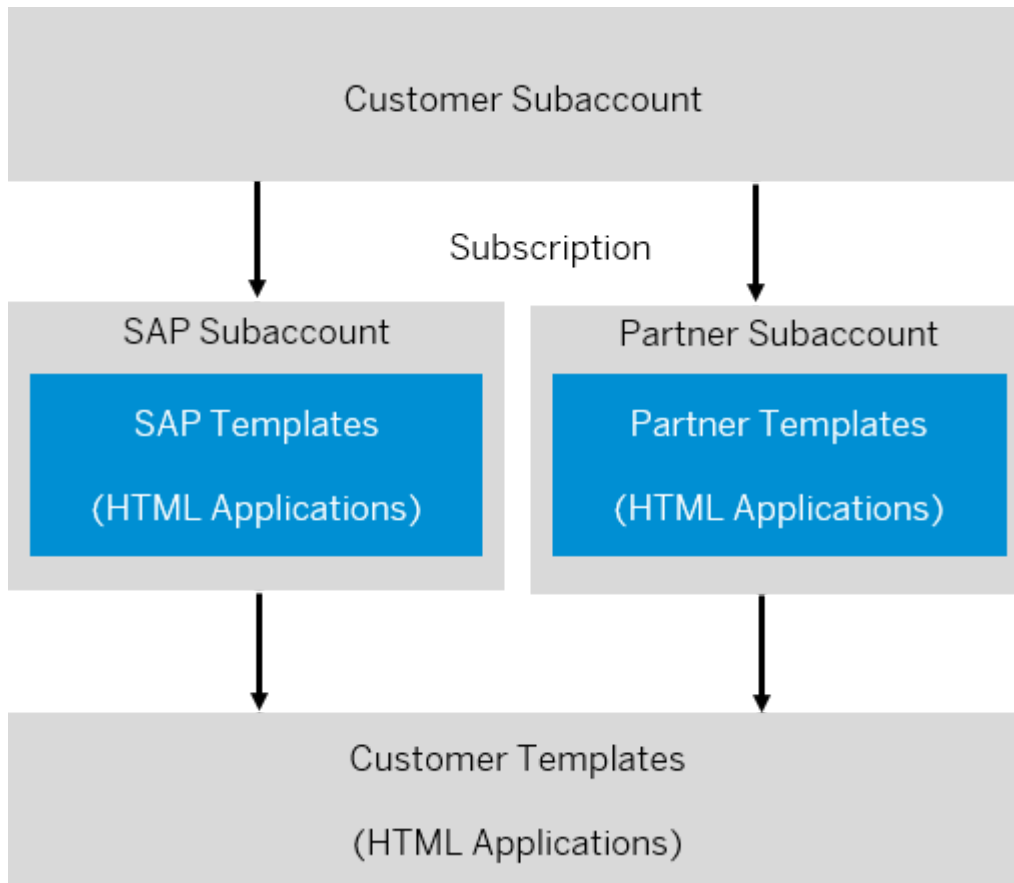
Portal service APIs

About Freestyle Site Templates

Developers create site templates in SAP Web IDE. The administrator then creates sites in the admin tools using these templates.

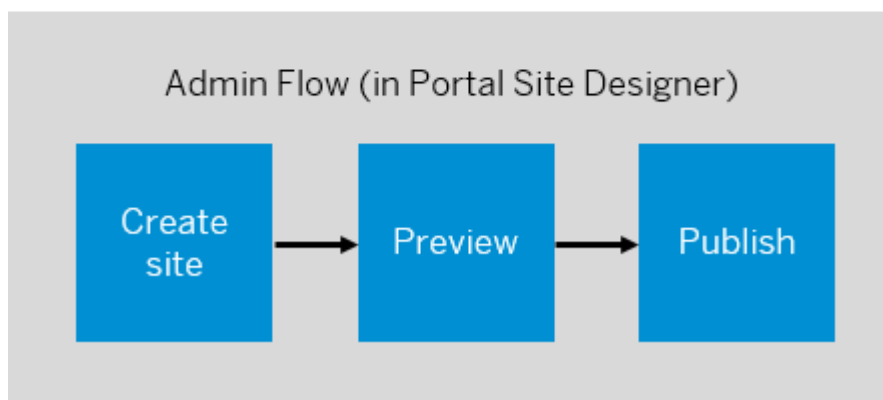
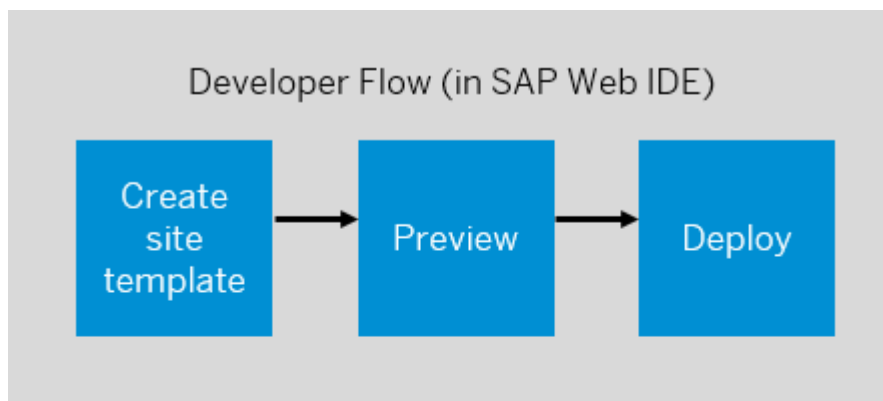
Template Creation Landscape

Freestyle sites are based on templates. As a developer, you can take any of the existing templates provided by SAP or by a different provider, and customize them to create your own templates.



Template Creation Flow

After deploying the templates to the subaccount, they become available for the admin, who then creates site instances from the templates.



End to End Development Flow

A step-by-step guide for creating basic freestyle site components in SAP Web IDE.

iNote

This guide requires a basic working knowledge of SAP Web IDE, and SAP Cloud Platform cockpit. We recommend familiarizing yourself with the basic work flows in these two products before reading this guide. For more information, see [SAP Web IDE documentation](#), [SAP Cloud Platform documentation](#).

SAP Web IDE offers a custom plug-in for developing freestyle site components in a structured manner. The plug-in consists of a set of wizards for creating site templates, page templates, pages, and widgets, as well as an option to convert a Web application to a freestyle site widget. All components are packaged together in one deployable project that can be deployed to your subaccount on the cloud.

Let's get started!

iNote

A typical development scenario is described in the following walkthrough - click each step for more details:

- 1 Enable the Portal plugin in SAP Web IDE
- 2 Create a site template
- 3 Create a page template
- 4 Create pages
- 5 Create a widget template
- 6 Create a web content widget template
- 7 Preview your site template
- 8 Deploy the template to SAP Cloud Platform

Enable Portal Plug-in/Feature on SAP Web IDE

Developers need to enable the Portal plugin/feature in SAP Web IDE before using it.

Prerequisites

- Make sure that the Portal Service is enabled in SAP Cloud Platform cockpit so that it is available in the list of repositories when selecting a plugin or a feature.
- Before creating your first project, make sure the plug-in/feature is enabled.

Context

Depending on the SAP Web IDE version you are using, you can use the Portal plug-in or the Portal feature to create site, page, and widget templates, as well as web content in freestyle sites.

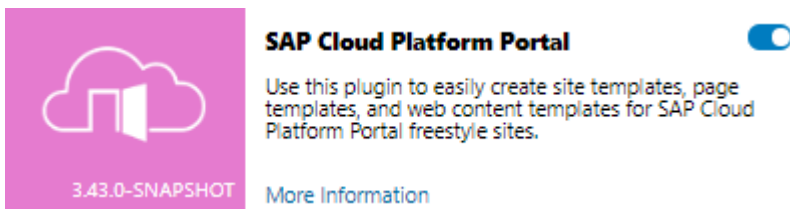
Procedure

1. Log on to your subaccount on SAP Cloud Platform, with a subaccount member user.
2. In the navigation pane, choose **Services**.
3. Search for **SAP Web IDE**.

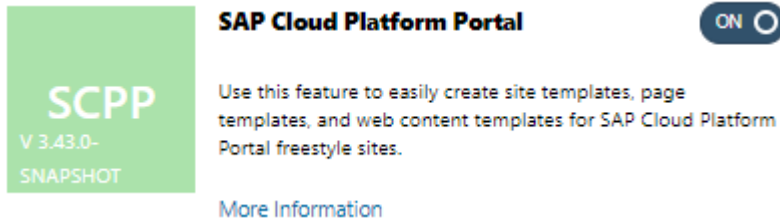
iNote

As part of the onboarding process, the **SAP Web IDE** service is already enabled for your subaccount.

4. Click the **SAP Web IDE Full-Stack** card to open the service screen depending on which version you are using.
5. Click **Go to Service**.
6. If you are using the **SAP Web IDE** classic version:
 - a. In the SAP Web IDE menu, choose the **Features** perspective.
 - b. In the left menu, choose **Plugins**.
 - c. In the **Repository** dropdown list, select **Portal Service**.
 - d. Enable the plug-in by switching it on, and click **Save**.
 - e. Refresh your browser.



7. If you are using the **SAP Web IDE Full-Stack** version:
 - a. In the SAP Web IDE menu, choose the **Preferences** perspective.
 - b. In the left menu, choose **Features**.
 - c. Search for the **Portal** tile.
 - d. Enable the feature by switching it on, and click **Save**.
 - e. Refresh your browser.



Create a Site Template

Creating a site template is the starting point of the Portal developer's flow.

Prerequisites

The Portal Service plug-in in SAP Web IDE, is enabled.

Context

The site template creation flow involves two personas - a developer and a site administrator. You, as a developer, create a site template in SAP Web IDE, and then deploy it to your subaccount on the cloud. You can choose to create a basic site template that defines only the layout of the site, or you can create a rich site template that defines layout and content. Typically, you would choose to define content when the content doesn't change often, such as a header with the company logo, or a page that is designed for a specific flow. After deploying the site template to the subaccount, the site administrator can create site instances based on your deployed template, and edit the site layout and content as needed.

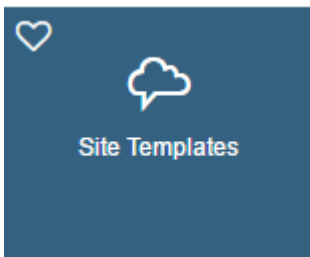
When you create a site template in SAP Web IDE, you always base it on a site template that already exists in your subaccount. Site template sources include:

- Site templates provided by SAP
- Site templates previously created (by you or other developers)
- Site templates provided by subaccounts that you are subscribed to

In this example, you will create a site template that is based on the [Basic Layout](#) site template, provided by SAP.

Procedure

1. In SAP Web IDE, open the [New Project](#) wizard by clicking [File New Project from Template](#).
2. Select the [Portal Service](#) category from the dropdown list and then select the [Site Templates](#) tile. Click [Next](#).



3. Enter a unique name for the new project, for example `MySiteTemplateProj`.

2 Basic Information

Project Name*

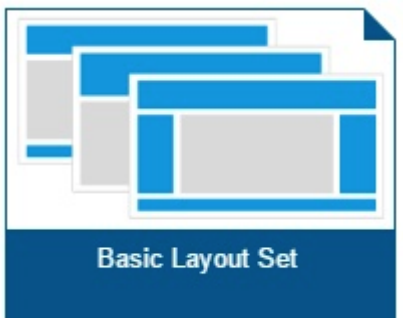
MySiteTemplate

4. If requested, logon to your subaccount.

5. Select the provider subaccount.

This action might take a few seconds to complete. Wait until the provider subaccount is loaded and you see all available site templates for this subaccount, in addition to site templates provided out of the box by SAP.

6. Select a site template from the list. Your new site template will be based on this template. In this example, select **Basic Layout Set** and click **Next**.



7. Enter a unique name and optionally a description for your new site template, for example, MySiteTemplate and click **Next**.

8. In the **Confirmation** screen, review all the details, and click **Finish**.

iNote

It might take a few seconds for the **Finish** button to become enabled.

9. To view your project in the code editor, click the  button on the left toolbar to open the Development perspective.

Results

A new project is created with a defined folder hierarchy. For more information about the project content and structure, see [Project Structure](#).

Create a Page Template

A developer can add page templates to the site template project, for use by the site administrator.

Context

A page template defines the layout of the page, and optionally can also define initial content. Page templates are useful when a specific layout or content is repeated in several pages of the site. Instead of the site admin creating several similar pages, you can create a page template and then the admin can create pages based on the page template.

When you created your site template, based on another site template, your site template might have "inherited" page templates from its source. Check the `pageTemplates` folder in your project structure to see if you have predefined page templates.

Similar to the creation flow of site templates, when you create a new page template, you base it on an existing page template. In this example, you will create a new page template based on the [Header and Footer](#) page template provided by SAP.

Procedure

1. Right-click the site template folder in your project structure, and select **New Page Templates**.
2. Select the provider account.

iNote

This action might take a few seconds to complete. Wait until the provider account is loaded and you see all available page templates for this account, in addition to page templates provided out of the box by SAP.

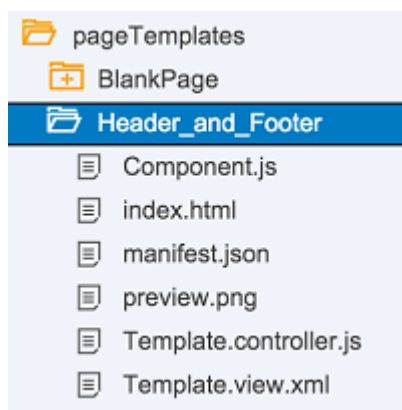
3. Select a page template from the list. Your new page template will be based on this template. In this example, select **Header and Footer** and click **Next**.



4. Enter a unique name and optionally a description for your page template, for example, **Header_and_Footer** and click **Next**.
5. Click **Finish**.

Results

A new folder was added under the **pageTemplates** folder, with your page template name.



Create Pages

Developers can add page instances to their site templates. Administrators can then modify those pages, as required, when they build their sites.

Context

A page is an instance of a page template. Adding pages to your site template is useful when you know the intended content of the page, and you want your site template to already include that page when the admin creates a site based on your site.

template. The admin can still edit the page, whether it was created by a developer or by an admin.

In this example, you will create an instance of a page based on the [Starter Page](#) page template.

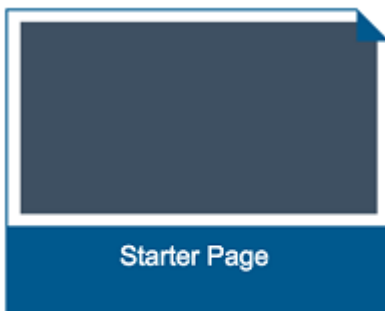
Procedure

1. Right-click the site template, and select [New Pages](#).
2. Select the provider subaccount.

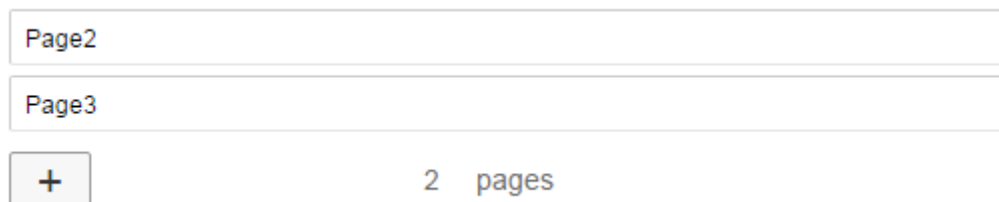
iNote

This action might take a few seconds to complete. Wait until the provider subaccount is loaded and you see all available page templates for this subaccount, in addition to page templates provided out of the box by SAP.

3. Select a page template from the list of available page templates. In this example, select the [Starter Page](#) and click [Next](#).



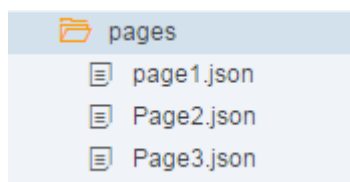
4. Enter a name for your page. You can create several pages at once by clicking on the plus icon.

A screenshot of a user interface for naming pages. It shows two input fields. The first field contains the text "Page2" and the second field contains the text "Page3". Below these fields is a plus icon in a square button. To the right of the plus icon, the text "2 pages" is displayed.

5. Click [Next](#) and then [Finish](#).

Results

Page instances were added under the pages folder.



Create a Widget Template

Developers can add widget templates to their site templates. Those widgets can then be added by the admin to pages and page templates.

Context

You can create a new widget template based on an existing widget template, provided by SAP or by a different provider. When you select the existing widget from the list, the widget with its app content is copied to your subaccount, allowing you to change the implementation.

Procedure

1. Right-click the site template folder, and select **New More Widget Templates**.
2. Select the provider subaccount.

iNote

This action might take a few seconds to complete. Wait until the provider subaccount is loaded and you see all available widget templates for this subaccount, in addition to widget templates provided out of the box by SAP.

3. Select the **Starter** widget template or one of your existing widget templates and click **Next**.
4. Enter a name for your new widget template. For example, **Image Widget**.
5. Enter a name for your widget template folder, for example **imageWidget**.
6. Select an icon from the SAP icon gallery.
7. Click **Next** and then **Finish**.

Results

A new folder is added under the **widgetTemplates** folder for your new widget template. Your new widget template is very basic and your next step would be to customize and enhance the code. For an example how you can enhance your starter widget template, see [Extend the Starter Widget](#).

iNote

Once you deploy the site template to your subaccount, the new widget template will appear in the Site Designer catalog automatically.

iTip

You can create standalone widgets by using the 'Convert App to Widget' flow. After the convert action, the widget is created but it is not part of your site template. After you deploy the new widget template to your subaccount, it appears automatically in the Site Designer widget catalog, and you can add it to your site instance. For more information, see [Convert an Application to a Widget](#).

Related Information

[URL Helper API](#)

Deploy Your Site Template to SAP Cloud Platform

Once the site templates are ready for use, developers can deploy them to their subaccount on the cloud.

Procedure

1. Right click the site template folder and select **Deploy Deploy to** SAP Cloud Platform. If prompted, enter your password.

2. Confirm the details of the deploy screen and click [Deploy](#).

iNote

Changes made to the site template and its components after you have deployed them, are not reflected in the admin tools. Redeploy the updated template and then create a new site instance based on the updated template. The updated template will overwrite the previously deployed version. If the admin already created a site over the original site template, that site instance will not be affected by the changes.

Results

The new site template is added to the list of HTML5 applications deployed on the cloud. You can view it in the cloud cockpit of your subaccount, under the [HTML5 Applications](#) section.

Developing Web Content Widget Templates

Developers can develop a web content widget template in SAP Web IDE and then extend it by adding code snippets and rendering the layout according their user's needs.

In this section of the developer guide, we introduce you to web content and explain some of the terminology that you should get familiar with. We also show you some samples with explanations of how you can not only create web content widget templates, but you can also extend them to suit any customization needs that your users require.

Let's begin by understanding what a web content widget is!

Web content widgets are available for selection in the widget gallery of the Site Designer. The site administrator chooses a web content widget and adds it to a site page. These widgets can then be edited in a dedicated authoring tool called the Web Content Editor; either by the site administrator, or by an authorized web content editor.

In the widget gallery, there are two types of web content widgets:

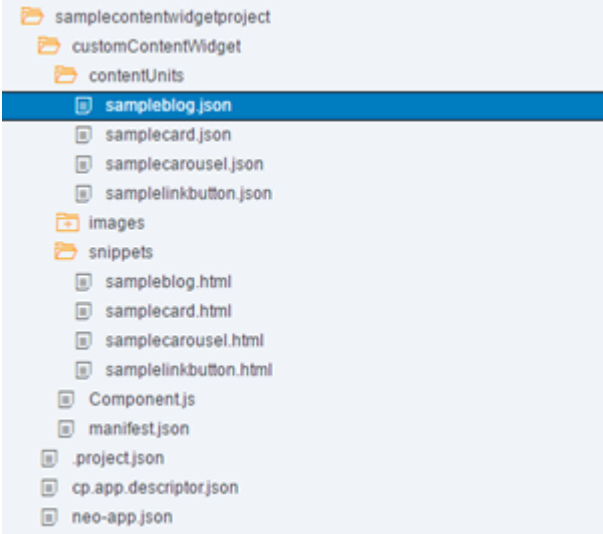
- Out-of-the-box web content widgets provided by SAP.
- Custom web content widgets that you create in SAP Web IDE and deploy to the SAP Cloud Platform subaccount.

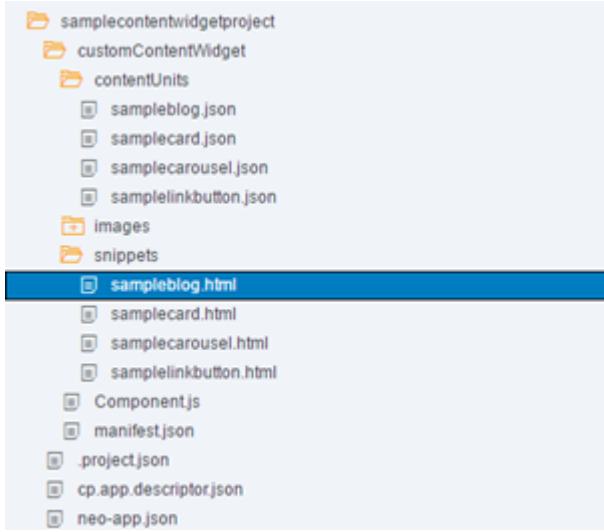
Web content widgets are either single non-repeatable components (such as a title, a link, or an image), or they can include multiple components with handy content snippets (small chunks of predefined content). The site administrator uses these widgets as a starting point for adding content to a page in their site.

For more information about web content, see [About Web Content](#).

Get to know the terminology

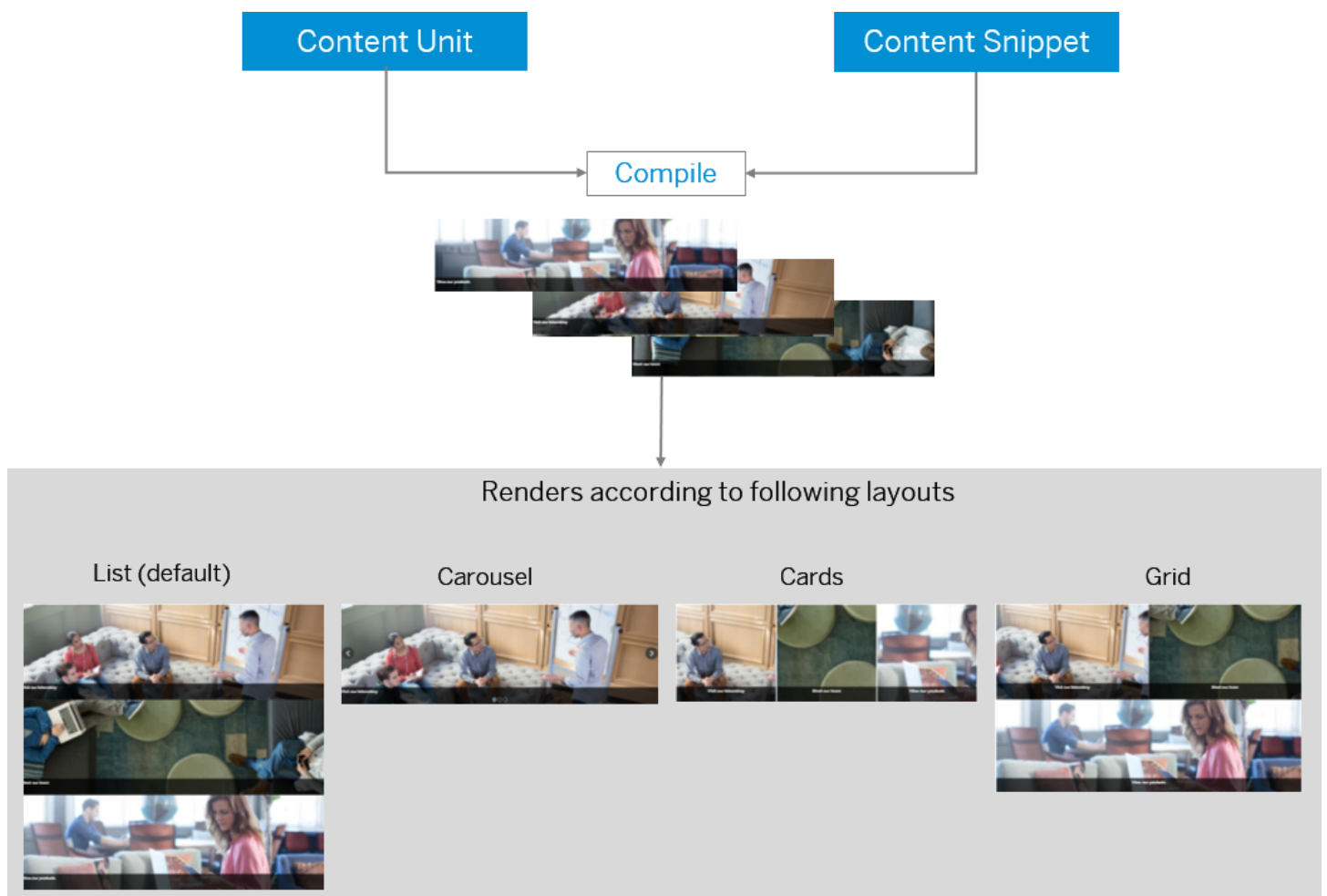
Term	Description
------	-------------

Term	Description
Content Unit	<p>An object that contains the data that is displayed within the web content widget. It contains one or more components and is saved in a JSON file.</p> 
Component	<p>A data set inside the content unit. If the content unit is defined to be repeatable, multiple components can be added to the content unit.</p> <p>≡Sample Code</p> <pre> "components": [{ "text": { "value": "Link Button", "isAsset": false }, "link": { "value": "#", "isAsset": false }, "linkTarget": { "value": "_blank" }, "linkTooltip": { "value": "" } }] </pre>
Element	<p>The component is made up of elements such as a link button, text, image, and so on.</p> <p>≡Sample Code</p> <pre> "text": { "value": "Link Button", "isAsset": false }, </pre>

Term	Description
HTML template (content snippet template)	<p>The HTML template contains the styling (CSS file) and the HTML markup that is used to render the content units in the user interface. Placeholders are integrated into the markup for each element.</p> 
Renderer	Code that defines the layout in which the components are arranged.

How does it work together?

The HTML templates (content snippets) are compiled with the content unit. This creates DOM elements that are arranged by out-of-the-box renderers (layouts) such as grid, carousel, and card, or you can also create your own customized renderers.



Each content unit can be compiled with several HTML templates (content snippets) to create different user interfaces with the same content.

Related Information

[Create a Web Content Widget Template](#)

[Enable Settings for a New Web Content Widget](#)

[The Web Content Widget Project Structure](#)

[Example of How to Extend Your Web Content Widget Template](#)

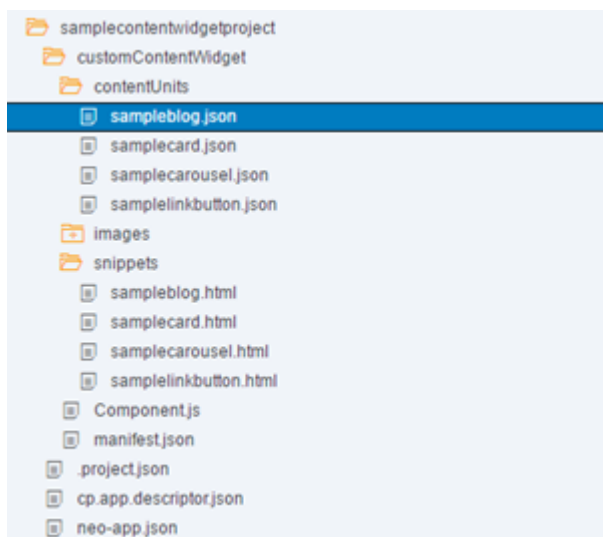
The Web Content Widget Project Structure

In this topic we explain in detail about the different files that make up the structure of the web content widget template.

Files that make up the Web Content Widget Template

1. content unit JSON

The content unit JSON file is located in the **contentUnits** folder of the web content widget template project.



- The texts section of the file contains the data needed for translating the components. It also contains a unique key, called the **translationKey** that links to the relevant component.

Sample Code

```
"texts": {
  "title-1": {
    "type": "",
    "value": {
      "": "Enter a title here"
    }
  },
  "text-1": {
    "type": "",
    "value": {
      "": "Replace this default content with your own. Simply click in the box and start"
    }
  }
}
```



```

    },
    "title-2": {
      "type": "",
      "value": {
        "": "Enter a title here"
      }
    },
    "text-2": {
      "type": "",
      "value": {
        "": "Replace this default content with your own. Simply click in the box and start"
      }
    },
    "title-3": {
      "type": "",
      "value": {
        "": "Enter a title here"
      }
    },
    "text-3": {
      "type": "",
      "value": {
        "": "Replace this default content with your own. Simply click in the box and start"
      }
    }
  },
},

```

- For components (an array holding the initial data of each widget instance), only the `translationKey` property is required.

Sample Code

```

"title": {
  "translationKey": "title-1",
  "isAsset": false
},
"text": {
  "translationKey": "text-1",
  "isAsset": false
},

```

- For the default component (the data used when the web content editor adds a new component to the web content widget in the Web Content Editor tool), the `translationKey` property as well as the default value is required.

Sample Code

```

"title": {
  "value": "Enter a title here",
  "translationKey": "title",
  "isAsset": false
},

```

```

"text": {
  "value": "Replace this default content with your own. Simply click in the box and
  "translationKey": "text",
  "isAsset": false
},

```

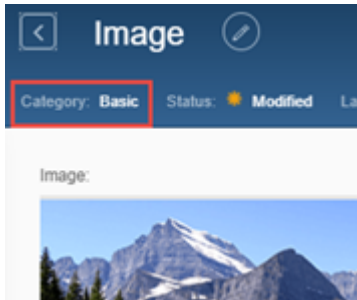
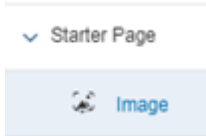
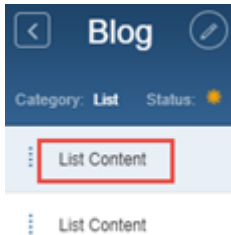
- The payload section of the file consists of the following parts:

Sample Code

```

{
  "identification": {
    "entityType": "contentunit"
  },
  "payload": {
    "type": "Carousel",
    "icon": "sap-icon://header",
    "componentType": "Carousel Content",
    "repeatable": true,
    "components": [...],
    "componentDescriptor": [...]
  }
}

```

Part	More Information
type	<p>The content unit type - a semantic value that you can determine. It is displayed as web content in the V</p> 
icon	<p>The SAPUI5 icon that is displayed in the Web Content Editor tool.</p> 
componentType	<p>The individual components that make up the content unit. A semantic value that you determine in the V</p> 

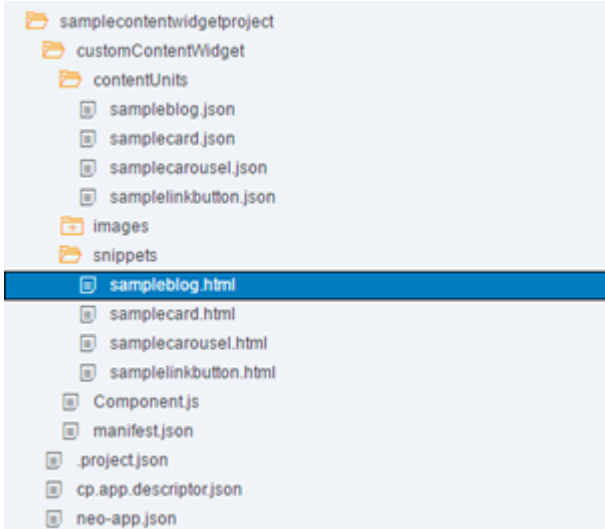
Part	More Information
defaultComponent	<p>The data used when the web content editor adds a new component to the web content widget in the We</p> <p>Relevant only if the content unit is repeatable.</p> <p>≡Sample Code</p> <pre> "defaultComponent": { "title": { "value": "Steam Iron", "translationKey": "title", "isAsset": false }, "content": { "value": "Most powerful steam for the quickest ironing. Detachable wa "translationKey": "content", "isAsset": false }, "imageSrc": { "value": "/images/pic1.jpg", "isAsset": true } }, </pre>
repeatable	Determines whether the web content editor will be able to add additional components to a web content

Part	More Information
components	<p>An array holding the initial data of each widget instance.</p> <p>Let's take a closer look at the elements inside the component as well as the default component:</p> <p>≡Sample Code</p> <pre> "\"title\": { \"value\": \" Enter a title here\", \"isAsset\": \" false\" } </pre> <ul style="list-style-type: none"> o title: The unique key of the field. o translationKey: A unique key that links to the relevant component that is being translated. o value: The data value of the field. o isAsset: Indicates whether the value is an asset. Set to <code>true</code> if the field value contains an image. Specify the relative path to the image as the value. <p>≡Sample Code</p> <pre> "components": [{ "title": { "value": "Steam Iron", "translationKey": "title", "isAsset": false }, "content": { "value": "Most powerful steam for the quickest ironing. Detachable", "translationKey": "content", "isAsset": false }, "imageSrc": { "value": "/images/pic1.jpg", "isAsset": true } }] </pre>

Part	More Information
componentDescriptor	<p>Defines the type, placeholder, and display title of each of the component's elements.</p> <p>Let's take a closer look at the elements in the component descriptor:</p> <p>≡Sample Code</p> <pre> "title": "Title", "type": "SHORT_TEXT", "value": "{/title/value}" </pre> <ul style="list-style-type: none"> ◦ title: The text or label displayed in the Web Content Editor tool. ◦ type: The SAPUI5 input control that is used in the Web Content Editor tool, and will be used fo <p>There are 4 types:</p> <ul style="list-style-type: none"> ▪ SHORT_TEXT - displays a regular input field. ▪ LONG_TEXT - displays a text area. ▪ IMAGE - displays an image upload control. ▪ LINK - displays a regular input field. Note that the links have additional value fields su ◦ value: contains the path to the value in the component data. <p>≡Sample Code</p> <pre> "componentDescriptor": [{ "title": "Image", "type": "IMAGE", "value": "{/imageSrc/value}" }, { "title": "Title", "type": "SHORT_TEXT", "value": "{/title/value}" }, { "title": "Content", "type": "LONG_TEXT", "value": "{/content/value}" }] </pre>

2. HTML Template

The HTML template is located inside the `snippets` folder:



The file has the following structure:

```
Sample Code

<component>
    //style of the snippet
<style>
</style>
//html and embedded placeholders
<div>
    {title}
</div>
</component>
```

Section	More information
Style section	<p>The style section contains a standard CSS file.</p> <p>There are some additional capabilities that can be implemented in the style section:</p> <ul style="list-style-type: none">Integration with the SAP theme<p>To apply the site theme to the HTML templates, you can use theme parameters in your CSS code. For example:</p><pre>Sample Code .title{ color: sapTheme(@sapUiHighlight); }</pre><p>You can find a list of basic theme parameters here: Theme-Dependent CSS-Classes.</p>Responsiveness<p>HTML templates (content snippets) can create responsive HTML components using a component query. Component thinking about responsive web design, where the responsive conditions are applied to the component element on the height of the browser window.</p><p>For example, if a widget is configured to take up one third of the section width, so that the width of the widget is 400 component query of @component-max-width (420).item, even when it is displayed in a desktop environment window is 1980px.</p>


Section	More information
	<p>Here are a few examples of component queries:</p> <ul style="list-style-type: none"> Smaller than 1280px: <pre> ❏ Sample Code @component-max-width (1280) .item { ... }</pre> Wider than 1280px: <pre> ❏ Sample Code @component-min-width (1280) .item { ... }</pre> Between 640px and 1280px: <pre> ❏ Sample Code @component-width (640,1280) .item { ... }</pre> We can also use predefined ranges: <pre> ❏ Sample Code @component-width (small) .item { ... }</pre> <p>Where:</p> <ul style="list-style-type: none"> small is min:0, max: 420 medium is min: 421, max: 1024 large is min: 1025, max: 1920 extra large is min: 1921 <ul style="list-style-type: none"> States <p>To set CSS properties of a content item with a specific state, use the following syntax: <code>@component:stateName</code></p> <pre> ❏ Sample Code <style> @component:first .item a{ display: none; } </style></pre>

Section	More in
	<div class="item"></div>
Markup section	<p>Here is an example of a markup section:</p> <pre>{Sample Code} <div class="imageCarousel_item {\$dir} {\$device}"> <div class="imageCarousel_image_box"> <div class="imageCarousel_image" style="background-image: url('{imageSrc}')"></div> </div> <div class="imageCarousel_text_box"> <div class="imageCarousel_title">{title}</div> <div class="imageCarousel_text">{content}</div> </div> </div></pre> <p>The markup is written in standard HTML and has the following additional capabilities:</p> <ul style="list-style-type: none">{placeholders} <p>Placeholders link the markup to the content unit component data, as it was defined in the <code>contentUnitJSON</code> file.</p> <p>For example:</p> <ul style="list-style-type: none">Component: <pre>{Sample Code} { "title": { "value": "Enter a title here ", "isAsset": false } }</pre> <ul style="list-style-type: none">Markup: <pre>{Sample Code} {title}</pre> <p>This is the runtime result:</p> <p>Enter a title here</p> <ul style="list-style-type: none">{\$dir} <p>To support RTL languages, you can use the <code>{\$dir}</code> placeholder in your HTML markup. The placeholder embeds the text direction in the HTML markup.</p> <ul style="list-style-type: none">Component: <pre>{Sample Code}</pre>

Section	More informati
	<pre>sap.ui.getCore().getConfiguration().getRTL()?'rtl':'ltr'</pre> <ul style="list-style-type: none"> Markup: <pre> <div style=direction: {\$dir} >{data} if the configuration of the core returns the value 'true' for getRTL() <div style='direction: rtl'>שלום </pre> • {\$index} <p>{\$index} is a placeholder that embeds the index of the component data item.</p> <ul style="list-style-type: none"> Component: <pre> components":[{ "title":{ "value":"User1", "isAsset":false } }, { "title":{ "value":"User2", "isAsset":false } }, { "title":{ "value":"User 3 ", "isAsset":false } }] </pre> Markup <pre> <div>{\$index}-{title} </pre> <p>This is the runtime result:</p> <pre> 0-User1 1-User2 2-User3 </pre> • Expressions

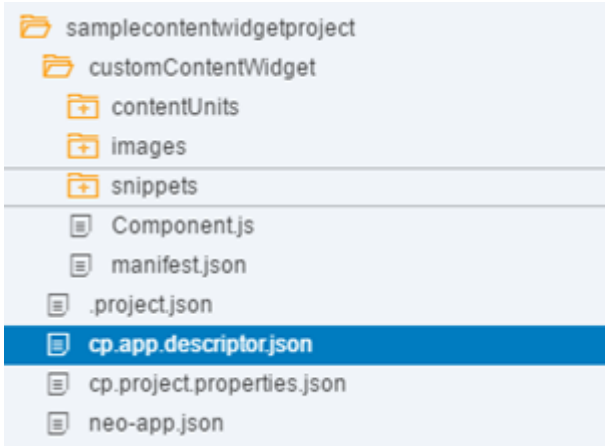
Section	More information
	<p>To use expressions, use [] (square brackets). There are two types of expressions available in HTML markup:</p> <ul style="list-style-type: none">Basic expressions: Sample: [\$index % 2]<ul style="list-style-type: none">Components: <div>Sample Code</div><pre>{ "title":{ "value":"User1", "isAsset":false } }, { "title":{ "value":"User2", "isAsset":false } }, { "title":{ "value":"User 3 ", "isAsset":false } }, { "title":{ "value":"User 4 ", "isAsset":false } }]</pre>Markup: <div>Sample Code</div><pre><div>[\$index%2</pre> <p>This is the runtime result:</p> <pre>0-User1 1-User2 0-User3 1-User3</pre> <ul style="list-style-type: none">Conditional (ternary) operator If you want to use a conditional expression, use this format: ['{status}' === 'success' : 'done' , 'not_doComponent: <div>Sample Code</div>

Section	More informati
	<div><pre>"components": [{ "status":{ "value":"done", "isAsset":false } }, { " status ": { "value": "in process", "isAsset": false } }, { " status ": { "value":"N/A", "isAsset": false } }, { " status ":{ "value":"done ", "isAsset":false } }]</pre></div> <div><ul style="list-style-type: none">Markup:</div> <div><div><div>≡Sample Code</div><pre><component> <style> .done{ color:green; } .not_done{ color:red; } </style> <div class=' [{status}]'=== 'success': 'done', 'not_done'] '>{status}</div> </component></pre></div></div> <div><p>This is the runtime result:</p><div><div>done</div><div>in process</div><div>N/A</div><div>done</div></div></div>

Section	More information
SAP Icons	To embed an SAP icon, use the <code>sap-icon</code> attribute with the name of the SAP icon (see the example below: For example:<sp You can find a list of available SAP UI5 icons here: SAP Icon Explorer .
Accessibility	To write accessible HTML elements, follow the ARIA definitions: Accessible Rich Internet Applications (ARIA)  .

3. `cp.app.descriptor.json`

The `cp.app.descriptor` file contains an array of all the content units in the project.



Each content unit has the following parts:

≡Sample Code

```
{
  "path": "/snippets/samplecard.html",
  "contentUnit": "/contentUnits/samplecard.json",
  "name": "Sample Card",
  "description": "",
  "categories": [
    "custom"
  ],
  "rendererType": "default",
  "icon": "sap-icon://picture",
  "rendererDefaultValues": {
    "default": {
      "height": "19.31rem"
    },
    "grid": {
      "height": "19.31rem"
    },
    "carousel": {
      "height": "19.31rem"
    },
    "card": {
      "width": "20.63rem",
      "height": "19.31rem"
    }
  }
},
```

Part	Description
path	The relative path to the HTML template.

Part	Description
contentUnit	The relative path to the contentUnit JSON file.
name	The display name in the widget gallery.
description	Description of the content unit (optional).
categories	The category in which the widget is displayed in the widget gallery. The default value for widgets generated with the widget template creation wizard in SAP Web IDE is Custom . You can also change the value and create your own category in the widget gallery.
rendererType	<p>The renderer that is used to render the contentUnit with the selected layout. In the Web Content Widget creation wizard in SAP Web IDE, you can choose between the following layouts: Basic, List, Card, Carousel, and Grid. These layouts translate into the following renderers:</p> <ul style="list-style-type: none"> • Basic: "rendererType": "default" • List: "rendererType": "default" • Card: "rendererType": "card" • Carousel: "rendererType": "carousel" • Grid: "rendererType": "grid" <p>iNote</p> <p>If you have enabled the settings button in the Site Designer, an admin can switch the layout of the web content widget. All the renderers that are listed in the <code>cp.app.descriptor</code> file for that widget template are displayed in the Site Designer.</p> <p>For more information, see Enable Settings for a New Web Content Widget.</p> <p>However, if you prefer that some layouts be hidden from the admin (for example you don't want the admin to be able to switch a Card layout to a List or Grid layout), you can simply delete the List and Grid renderers section for that widget from the <code>cp.app.descriptor.json</code> file.</p> <p>If you leave only the Card layout, the entire layout selection option in the widget settings dialog box will be hidden.</p>
icon	The icon that is displayed in the SAP Web IDE wizard when the preview cannot be displayed.
rendererDefaultValues	<p>Width and height of the component at runtime per renderer.</p> <p>iNote</p> <p>To make sure that a widget will always have a specific height, especially in cases where the admin switches the layout of a web content widget in the Site Designer, you can hide the Autofit property in the settings dialog box by adding the property, <code>"allowAutoHeight": false</code>.</p>

Typical Workflow for Developing Web Content Widgets

Typical workflow showing how developers can create a web content widget template, add existing code snippets to it, and then choose how to render the content and make it available in the Site Designer widget gallery.

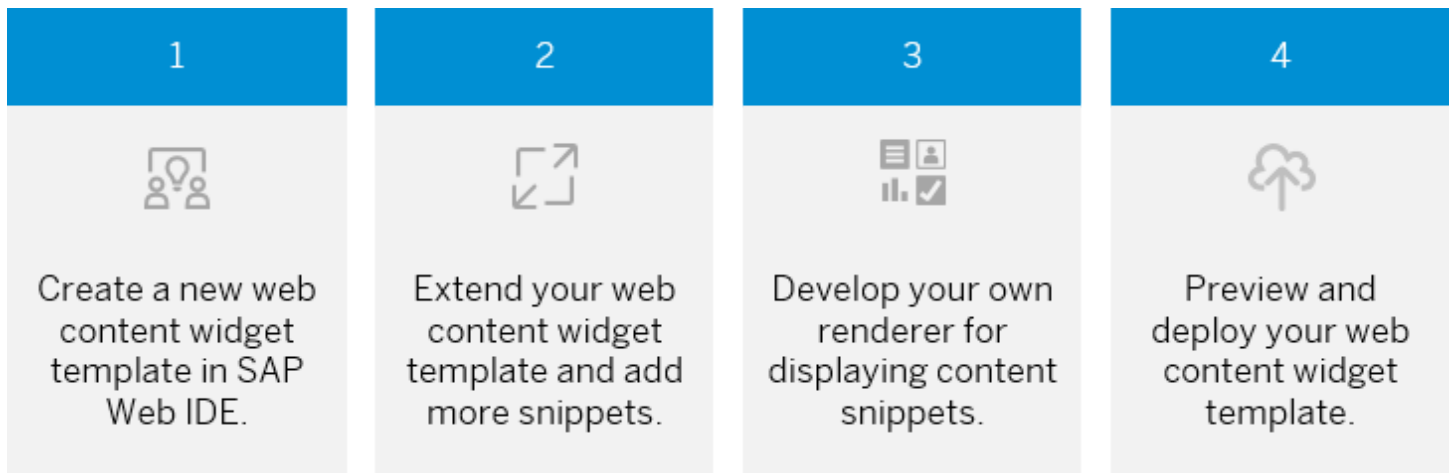
You can develop a web content widget template, add existing code snippets to it, and then choose how to render the content and make it available in the Site Designer widget gallery.

What is the general development flow of a web content widget?

1. Create a custom web content widget template in SAP Web IDE.
2. Add content snippets to your template that are based on existing out-of-the-box content snippets.
3. Select a layout to determine how you want to render and display the snippet in the Site Designer widget gallery. If one of the renderers is not suitable for your scenario, create your own renderer.
4. Extend the web content widget that you created by adding more content snippets to it.
5. Deploy the template you have created to the relevant subaccount on SAP Cloud Platform. By doing this, you are making it available in the Site Designer widget gallery.

For more information, see [Deploy Your Site Template to SAP Cloud Platform](#).

Click each step in the diagram below to learn more about creating a web content widget template.



Create a Web Content Widget Template

Developers can create a web content widget template using SAP Web IDE.

In this first step of the process you are going to create a web content widget project. In the process, you'll create a template, add a content snippet to it, and you'll choose how to render the layout of the content snippet.

Let's get started

1. Log on to SAP Web IDE, right click your workspace, and click **New Project from Template**.
2. From the **Category** dropdown list, select **SAP Cloud Platform Portal**.
3. Click the **Web Content Widget** tile to create a web content widget template.
4. Follow the steps of the wizard as follows:
 - a. Enter a name for your project.
 - b. Select the provider subaccount.
 - c. In the **Content Snippet Creation** step, select one of the available content snippets.

iNote

You are now selecting and adding a content snippet to your web content widget template. The content snippet is an HTML file that shows the different elements of data that the snippet contains (for example, image,

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

caption, text, heading, and so on).

The content snippets that are available for you to choose, are those that exist in your SAP Cloud Platform subaccount.

d. In the next step, select one of the available content snippet layouts.

iNote

You are selecting how you want your content snippet to be rendered and displayed once the site administrator adds it to a page section in the Site Designer. These are your options:

- **Basic** - a single content unit that is non-repeatable. In other words, you cannot add additional components to it.
- **List** - the content units appear one under the other.
- **Card** - the content units are aligned next to each other and when the row is full (depending on how many the admin defined to fit into a row), they wrap to the next line.
- **Carousel** - one content unit is rendered at a time - moving between them is done using pagination or a scroller.
- **Grid** - the content units dynamically resize to take up the entire area of the page section. For example, if you add two components, each component will resize to take up half the area in the page section. If you add three, the components are scaled down to fit into the same area.

As you select a layout, the preview on the right reflects how the content snippet you select is displayed.

e. Enter a name and ID number for your content snippet.

f. Click **Finish** to create the project in your Workspace.

g. In the next step, confirm your project.

Result

A new project is created with a defined folder hierarchy. Once you have deployed your project to the SAP Cloud Platform subaccount, your site administrator can add them as content to pages of their Portal freestyle site.

Add More Content Snippets to Your Project

Developers can extend their web content widget project by adding more content snippets to it.

You can add as many content snippets as you want. Simply do the following:

1. Right click your web content widget template project, and click **New Content Snippet**.
2. The wizard opens at the **Content Snippet Creation** step.
3. Select one of the available content snippets and add it to your project.
4. Select one of the available content snippet layouts.
5. Enter a name and ID number for your content snippet.
6. Click **Finish** and confirm.

You will now see the additional content snippet in your project hierarchy.

Develop Renderers to Display Web Content

Developers can create their own customized renderer to determine how a content snippet in their web content widget project is displayed.

As a developer you can create a web content widget project in SAP Web IDE. During the creation process, you add content snippets to your web content widget template and render them according to the out-of-the-box renderers. You can also create your own customized renderer for your project. In this topic we will show you how to do this.

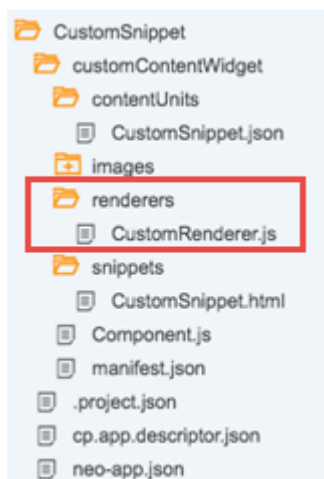
First we will create the renderer and map it to the relevant content snippet that you want to render.

1. Select your web content widget project.
2. Right-click and select **New Renderer**.
3. Follow the wizard steps and click **Finish** to create the new renderer.
4. Now map the content snippet (or content snippets) to the renderer that you have just created as follows:
 - a. Open the `cp.app.descriptor.json` file.
 - b. Replace or add the `rendererType` property with the new renderer namespace.

iNote

The renderer namespace can be taken from the new renderer `<>.js` file under the renderers folder.

For example, if we created a renderer with ID `CustomRenderer` under the `CustomSnippet` project, the path will be `customsnippet.renderers.CustomRenderer`



Now we are going to extend the renderer

1. Open the new renderer `<>.js` file. The file is created with basic code.
2. Now let's add the extended code to it. We will use the following example to walk you through the different API methods:

⌵ Sample Code


```

sap.ui.define([
    "sap/ushell/cpv2/services/control/contentUnit/WebContentRenderer"
], function(WebContentRenderer) {
    "use strict";
    return WebContentRenderer.extend("customsnippet.renderers.CustomRenderer", {

        getDefaultSettings: function() {
            var defaultSettings = {
                height: "10rem"
            };
            return defaultSettings;
        },

        onInit: function() {
            this.itemsMarginUnit = "rem";
        },

        createSettingsEditor: function(propertyEditorManager) {
            var propertyEditor = propertyEditorManager.editor;

            this.addPropertyEditor({
                key: "height",
                category: propertyEditorManager.bundle.getText("WIDGET_CATEGORY_DIMENSIONS"),
                title: propertyEditorManager.bundle.getText("WIDGET_SETTINGS_LABEL_HEIGHT"),
                editor: propertyEditor.Size,
                flags: propertyEditor.Size.AUTO | propertyEditor.Size.UNIT_PX | propertyEditor.Size.UNIT_PERCENT,
            });
        },

        renderer: {
            render: function(oRm, oControl) {
                var settings = oControl.getSettings();
                oRm.write("<div");
                oRm.writeClasses();
                oRm.writeControlData(oControl);
                oRm.write(">");
                oRm.write("<div>");

                var itemMargin;
                if (!oControl.isDesignTime() || !oControl.isHostedInComponent()) {
                    itemMargin = 0 + oControl.itemsMarginUnit;
                } else {
                    itemMargin = 0.5 + oControl.itemsMarginUnit;
                }

                var uiComps = oControl.getItems();
                jQuery.each(uiComps, function() {
                    oRm.write("<div style='position: relative; overflow: hidden; height:" + settings.height + "px; margin-bottom: " + itemMargin + "px; border: 1px solid #ccc; border-radius: 4px; padding: 5px; width: 100%; box-sizing: border-box;';");
                    oRm.renderControl(this);
                    oRm.write("<div style='clear: both'></div>");
                    oRm.write("</div>");
                });
                oRm.write("</div>");
                oRm.write("</div>");
            }
        }
    });
}, true);

```

More details about each API method:

Method	Description
getDefaultSettings	Returns an object with default settings of the renderer. If your renderer has settings, this method must be implemented.
onInit	Called automatically when the renderer instance is created. You can use it for example to initialize data.

Method	Description
createSettingsEditor	<p>Use this method if your renderer has settings to define.</p> <p>If implemented, a widget Settings dialog box is automatically called (in design time) when the administrator clicks the Settings button for a particular widget.</p> <p>To expose the renderer settings in the Settings dialog box, you need to add a property editor for each setting. This editor must include the following properties:</p> <ul style="list-style-type: none"> ◦ key: Links the property editor to the specific setting. ◦ category: Enables the properties to be grouped together into categories in the Settings dialog box like this: <div data-bbox="587 528 1150 860" data-label="Image"> </div> ◦ title: Enables the field labels in the property editor. <div data-bbox="587 936 1096 999" data-label="Image"> </div> ◦ flags: Shows a fixed list of options for the editor. ◦ editor: Choose from one of the following editors: <ul style="list-style-type: none"> ■ Size: Composed of an input field and a dropdown list showing the size units. Available flags: AUTO (use for autofit) as well as fixed units such as UNIT_PT, UNIT_PX, UNIT_REM, UNIT_VH, UNIT_VW <div data-bbox="667 1279 1235 1379" data-label="Image"> </div> ■ Boolean: Available flags: SWITCH_CONTROL: <div data-bbox="667 1547 975 1621" data-label="Image"> </div> YES_NO. <div data-bbox="667 1700 952 1742" data-label="Image"> </div> ■ Alignment: <div data-bbox="667 1816 1326 1883" data-label="Image"> </div> Available flags: None.

Method	Description
renderer	<p>Use this method to define how the snippet will be rendered.</p> <p>Defines the renderer HTML structure that will be added to the DOM tree. This is the extended renderer method used by the SAPUI5 control.</p> <p>iNote</p> <p>You can also use all extended SAPUI5 control methods such as <code>onBeforeRendering</code>, <code>onAfterRendering</code>, and so on.</p>
isHostedInComponent	<p>Use to determine if the widget should run in regular mode (design time and runtime) or preview mode.</p> <p>If the widget runs in regular mode, the method returns a <code>True</code> value; if it runs in preview mode, it returns a <code>False</code> value.</p> <p>You can also use this method to alter the way the renderer is displayed in preview mode (for example change the size).</p>
isDesignTime	<p>Use to determine whether the widget is currently in design time. While in design time, you can change the behavior of the renderer. For example, the carousel renderer doesn't loop in design time.</p> <p>This method returns <code>True</code> if the widget with the custom renderer runs in design time and <code>False</code> if it runs in runtime or preview mode.</p>
getSettings	<p>Returns the current renderer settings.</p> <p>The settings are initially defined in the <code>getDefaultSettings</code> method and can be overridden in the <code>cp.app.descriptor.json</code> file. This can also be changed by the administrator in design time in the Settings dialog box.</p>
getItems	<p>Returns an array of content items already with the correct data taken from the content unit json file.</p> <p>Each content item includes the following methods:</p> <ul style="list-style-type: none"> ◦ <code>getId</code> <p>Returns the content item ID.</p> ◦ <code>setState</code> <p>Adds a state to the content item. Can be helpful in the CSS file since different CSS properties can be set according to a content item state.</p> <p>For more information, see The Web Content Widget Project Structure. Refer to the HTML template - style section in the table.</p> <p>The first and last items always have "first" and "last" states.</p>

iNote

Once you have deployed the renderer, it will be available in the basic flow for creating web content widget templates in the web content wizard in SAP Web IDE.

You can use the customized renderer to define the layout of any content snippet in the same web content widget project.

Preview Your Web Content Widget

Administrators can preview their new Web Content Widget project before deploying to the SAP Cloud Platform subaccount.

To preview your new Web Content Widget, do the following:

1. Go to your project and in the folder hierarchy, either right-click the HTML file inside the `snippets` folder, or the JSON file in the `contentUnits` folder.
2. Select **Preview Web Content Widget**.
3. Enter the **Provider Account** details.

iNote

The default provider account is the value you provided when creating the template.

Result:

You are now previewing the web content widget template as it will appear in the Site Designer widget gallery after you deploy it to the SAP Cloud Platform subaccount. Administrators will add content to their sites based on this template.

Enable Settings for a New Web Content Widget

When developers create a new web content widget, they may need to enable some settings.

During the creation process for a new web content widget in SAP Web IDE, you select the layout of how you want the content snippet of the widget to be rendered and displayed. The admin can then select your custom widget template in the Site Designer and select a different layout by clicking the widget settings button. However this can only happen if you have enabled the settings button to appear in the Site Designer.

This is how you do it:

The `hide.settings` property that hides the settings button is located in the `manifest.json` file of the project and the default is `true`:

≡Sample Code

```
"sap.cloud.portal": {
    "hide.settings": true,
    "settings": {}
}
```

If you want to expose the settings button to the admin, either delete the property, or set it to `false`:

≡Sample Code

```
"sap.cloud.portal": {
    "hide.settings": false,
    "settings": {}
}
```

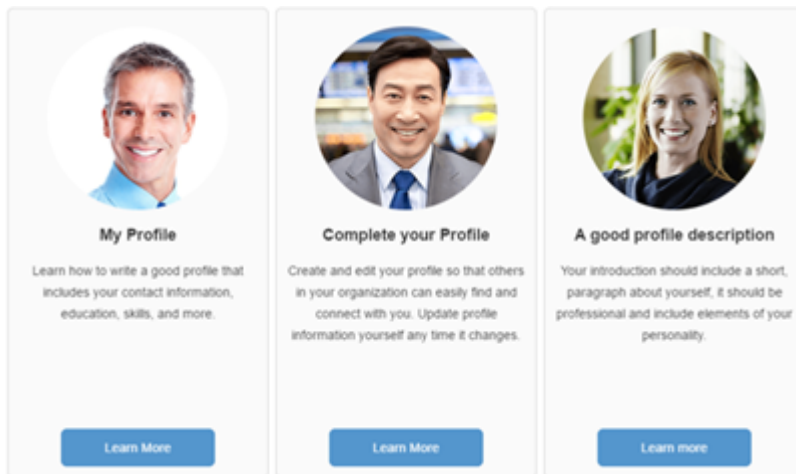
iRemember

Redeploy your project to enable the settings button to appear. You may also have to clear your browser cache.

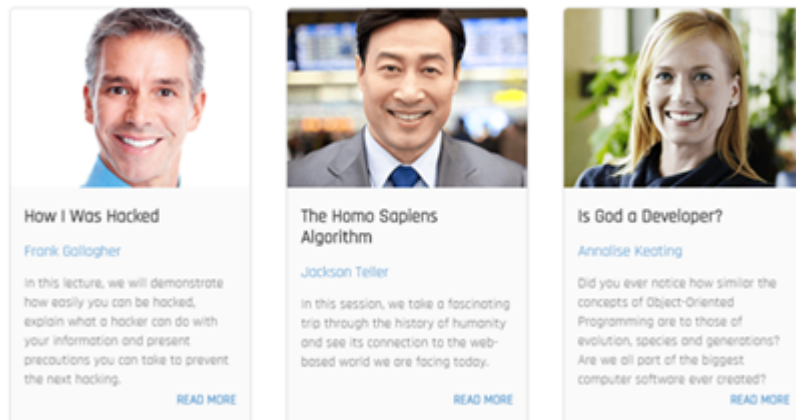
Example of How to Extend Your Web Content Widget Template

In this section, we will build a new web content widget.

Let's take a [List](#) widget and select the following [Profile](#) content snippet as a starting point and adjust it according to what we need.



We want to modify it so that it looks like this:



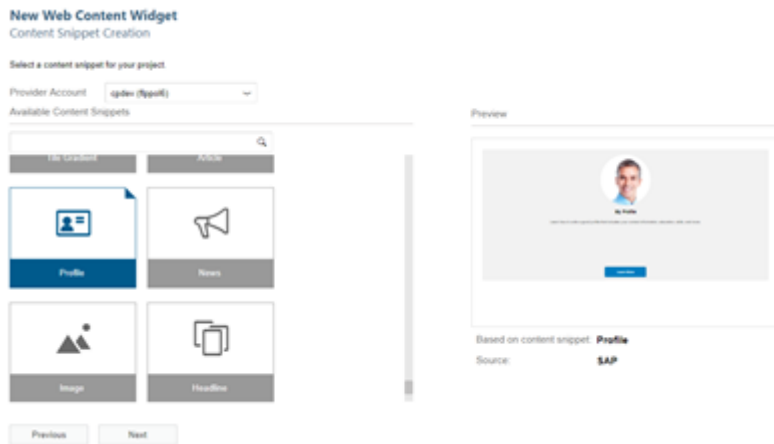
This is what we are going to do:

- Add an additional element.
- Change the element names (labels of the input fields that are visible in the Web Content Editor tool).
- Change the default values (the values of the new component that you add).
- Change the style (font, colors, tile size, and spacing between the tiles).

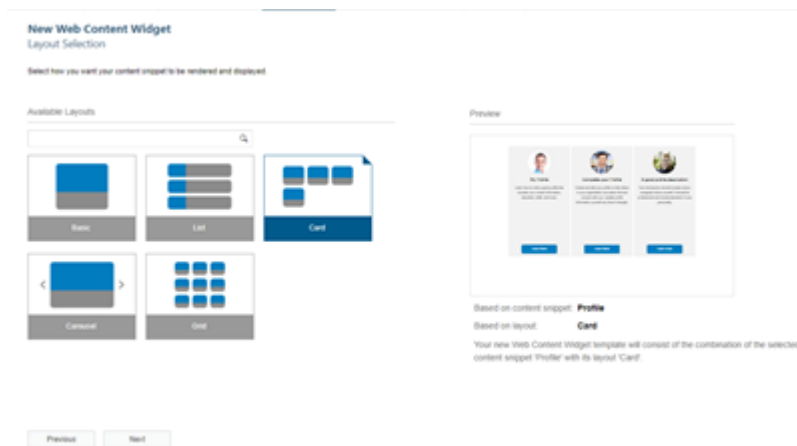
Step 1: Create a new Web Content Widget in SAP WebIDE

1. Log on to SAP Web IDE and click [New Project from Template](#).

2. From the **Category** dropdown list, select **SAP Cloud Platform Portal**.
3. Click the **Web Content Widget** tile to create a web content widget template.
4. Follow the steps of the wizard as follows:
 - a. Enter a name for your project, let's call it **conferencespeakers**.
 - b. Select the provider subaccount.
 - c. In the **Content Snippet Creation** step, select the **Profile** content snippet.

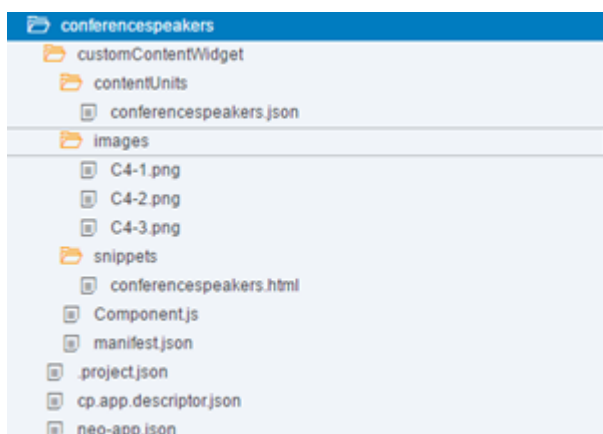


- d. Select the **Card** layout.



- e. Enter the web content widget name and a unique ID. The name is the display name in the widget gallery. The ID is the value given in the HTML and the JSON file. For our example, we will name the template **Conference Speakers** and use **conferencespeakers** as the ID.
- f. Click **Finish** to close the wizard.

Result:



Step 2: Update the ContentUnit JSON file

1. Open the `conferencespeakers.json` file.
2. Update the `componentDescriptor` section as follows:

```
"componentDescriptor": [{
    "title": "Speaker Image",
    "type": "IMAGE",
    "value": "{/imageSrc/value}"
  }, {
    "title": "Lecture Title",
    "type": "LONG_TEXT",
    "value": "{/lecture_title/value}"
  }, {
    "title": "Speaker Name",
    "type": "SHORT_TEXT",
    "value": "{/speaker/value}"
  }, {
    "title": "Lecture Description",
    "type": "LONG_TEXT",
    "value": "{/lecture_description/value}"
  }, {
    "title": "Read More Link",
    "type": "LINK",
    "displayText": "{/linkDisplayText/value}",
    "value": "{/linkURL/value}",
    "linkTarget": "{/linkTarget/value}",
    "linkTooltip": "{/linkTooltip/value}"
  }]
}
```

Note

This is what we did:

- We changed the `title` fields – these are the names of the elements as displayed in the Web Content Editor tool.
- We added an additional element.
- We changed the `value` path that will be used as {placeholders} in the HTML markup.

3. Update the `defaultComponent` section as follows:

```
"defaultComponent": {
    "imageSrc": {
        "value": "/images/C4-1.png",
        "isAsset": true
    },
    "lecture_title": {
        "value": "How I Was Hacked",
        "isAsset": false
    },
    "speaker": {
```

```

        "value": "Frank Gallagher",
        "isAsset": false
    },
    "lecture_description": {
"value": "In this lecture, we will demonstrate how easily you can be hacked, explain what a h
        "isAsset": false
    },

    "linkDisplayText": {
        "value": "READ MORE",
        "isAsset": false
    },
    "linkURL": {
        "value": "about:blank",
        "isAsset": false
    },
    "linkTarget": {
        "value": "_blank"
    },
    "linkTooltip": {
        "value": ""
    }
}

```

iNote

This is what we did:

- We updated the keys of each element with the values from the `componentDescriptor` section.
- We updated the default values for each element.

4. Update the `components` section as follows:

```

"components": [{
    "imageSrc": {
        "value": "/images/C4-1.png",
        "isAsset": true
    },
    "lecture_title": {
        "value": "How I Was Hacked",
        "isAsset": false
    },
    "speaker": {
        "value": "Frank Gallagher",
        "isAsset": false
    },
    "lecture_description": {
"value": "In this lecture, we will demonstrate how easily you can be hacked, explain what a h
        "isAsset": false
    },
    "linkDisplayText": {
        "value": "READ MORE",
        "isAsset": false
    }
}

```



```

    },
    "linkURL": {
      "value": "about:blank",
      "isAsset": false
    },
    "linkTarget": {
      "value": "_blank"
    },
    "linkTooltip": {
      "value": ""
    }
  }, {
    "imageSrc": {
      "value": "/images/C4-2.png",
      "isAsset": true
    },
    "lecture_title": {
      "value": "The Homo Sapiens Algorithm",
      "isAsset": false
    },
    "speaker": {
      "value": "John Snow",
      "isAsset": false
    },
    "lecture_description": {
      "value": "Throughout history, homo sapiens have gone through several revolutions – starting w
      "isAsset": false
    },
    "linkDisplayText": {
      "value": "READ MORE",
      "isAsset": false
    },
    "linkURL": {
      "value": "about:blank",
      "isAsset": false
    },
    "linkTarget": {
      "value": "_blank"
    },
    "linkTooltip": {
      "value": ""
    }
  }, {
    "imageSrc": {
      "value": "/images/C4-3.png",
      "isAsset": true
    },
    "lecture_title": {
      "value": "Function Life(): Is God a Developer?",
      "isAsset": false
    },
    "speaker": {
      "value": "Jackson Teller",
      "isAsset": false
    }
  }
}

```

```

    },
    "lecture_description": {
"value": "Did you ever notice how similar are the concepts of Object-Oriented Programming to",
        "isAsset": false
    },

    "linkDisplayText": {
        "value": "READ MORE",
        "isAsset": false
    },
    "linkURL": {
        "value": "about:blank",
        "isAsset": false
    },
    "linkTarget": {
        "value": "_blank"
    },
    "linkTooltip": {
        "value": ""
    }
}
}

```

iNote

This is what we did:

- We updated the keys of each element with the values from the `componentDescriptor` section.
- We updated the default values for each element.

Step 3: Update the HTML Template file

1. Open the `conferencespeakers.html` file.
2. Update the markup section as follows:

```
<div class="card4_item" {${dir}} {${device}} sapThemeBaseBG">
  <div class="card4_image_box">
    <div class="card4_image" style="background-image: url('{imageSrc}')"></div>
  </div>
  <div class="card4_text_box">
    <div class="card4_title">{lecture_title}</div>
    <div class="card4_speaker">{speaker}</div>
    <div class="card4_description">{lecture_description}</div>
    <div class="card4_link_box">
      <a tabindex="0" class="card4_link" target="['{linkTarget}'?'{linkTarget}':'_blank']" href={l
    </div>
  </div>
</div>
```

iNote

This is what we did:

- o We updated the {placeholders} as they appear in the JSON file.
- o We added markup for the new element (lecture_description)).

3. Update the <style> section as follows:

≡,Sample Code

```
<link href="https://fonts.googleapis.com/css?family=Rajdhani" rel="stylesheet">
<style>
    .card4_item {
        top: 0;
        left:0;
        right:0;
        bottom:0;
        position: absolute;
        text-align: center;
        font-family: 'Rajdhani', sans-serif;
        border-radius: 0.4rem;
        border: 1px solid sapTheme(@sapUiExtraLightBorder);
        overflow: hidden;
        text-rendering: optimizeLegibility;
        -webkit-font-smoothing: antialiased;

    }

    .card4_item .card4_image_box {
        width: 100%;
        height: 12.81rem;
        overflow: hidden;
        display: inline-block;
    }

    @component-max-width (260)    .card4_item .card4_image_box {
        width: 6.81rem;
        height: 6.81rem;
    }

    .card4_item .card4_image_box .card4_image {
        width: 100%;
        height: 100%;
        background-size: cover;
        background-position: 50% 50%;
        background-repeat: no-repeat;
    }

    .card4_item .card4_text_box {
        color: sapTheme(@sapUiBaseText);
        font-size: 0.875rem;
        line-height: 1.5rem;
        width: 100%;
        padding: 1rem;
        box-sizing: border-box;
        text-align:left;
    }

    .card4_item .card4_text_box .card4_title {
        font-size: 1.3rem;
        line-height: 1.5rem;
```

```

        font-weight: 900;
        margin-bottom: 1rem;
        max-height: 3rem;
        overflow: hidden;
    }

    .card4_item .card4_text_box .card4_speaker {
        margin-bottom: 0.9rem;
        max-height: 9rem;
        overflow: hidden;
        color:sapTheme(@sapButton_Emphasized_Background);
        font-size: 1.1rem;
    }

    .card4_item .card4_text_box .card4_description {
        margin-bottom: 0.9rem;
        max-height: 9rem;
        overflow: hidden;
        color: rgba(0,0,0,0.8);
        line-height:1.4rem;
        font-size: 1rem;
    }

    .card4_item .card4_link_box .card4_link {

        color:sapTheme(@sapButton_Emphasized_Background);
        text-decoration: none;
        padding: 1rem;
        display: inline-block;
        box-sizing: border-box;
        max-width: 100%;
        position: absolute;
        bottom: 0;
        right: 0;
        white-space: nowrap;
        overflow: hidden;
        text-overflow: ellipsis;
        max-width: 100%;
        box-sizing: border-box;
        font-weight: bold;
        font-size: 11pt;
    }

    .card4_item .card4_link_box .card4_link:hover {
        text-decoration:underline;
    }

```

iNote

We updated the CSS to fit the target Visual Design.

Note that the colors are parameters that get their value from the sapTheme.

Step 4: Update the cp.app.descriptor file

1. Open the `cp.app.descriptor` file.
2. Change the `rendererDefaultValues` as follows:

≡ Sample Code

```
"card": {
    "width": "19.75rem",
    "height": "32.38rem"
    "margin": "1rem"
}
```

</style>

3. Change the category name:

```
"categories": [
    "Custom Cards"
```

iNote

We changed the name of the category under which the widget appears in the widget gallery.

Step 5: Deploy to your SAP Cloud Platform subaccount

Select the project root folder, right click, and select [New Deploy Deploy to SAP Cloud Platform](#).

iNote

We deployed the new widget to the SAP Cloud Platform subaccount. The site administrator can now select the new widget in the Site Designer widget gallery.

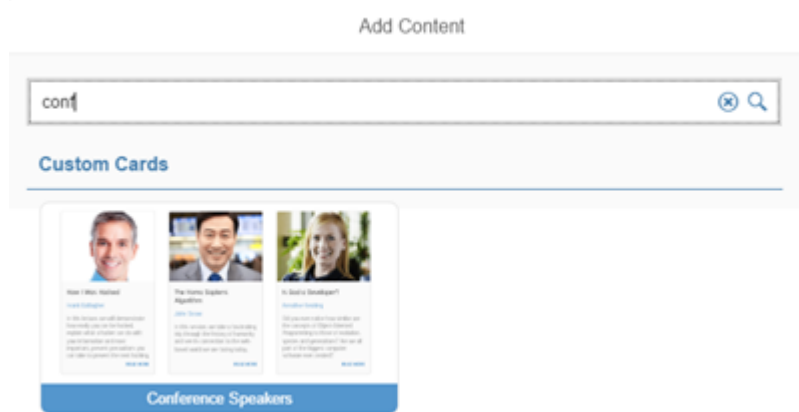
Step 6: Add the widget to the page (administrator)

1. Log on to the Portal as a site administrator.
2. Create a new page.
3. Click **+** in the page section.
4. In the [Add Content](#) dialog box, search for the **conference** widget and preview it.

iNote

The category is the one we defined in the `cp.app.descriptor` file.

Result:

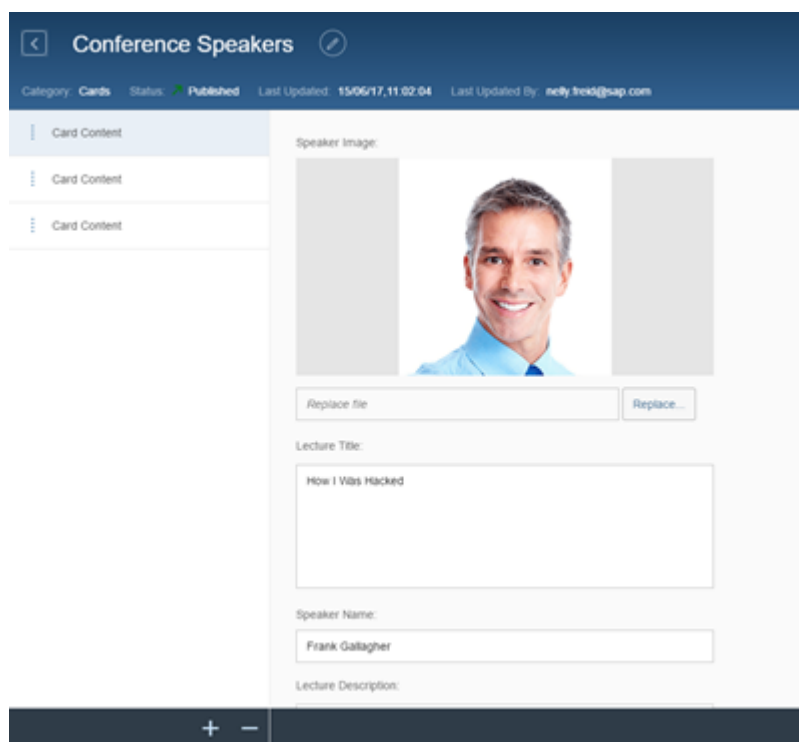


5. Hover and click **+** to add the widget to the page.

Step 7: Edit the widget (administrator or web content editor)

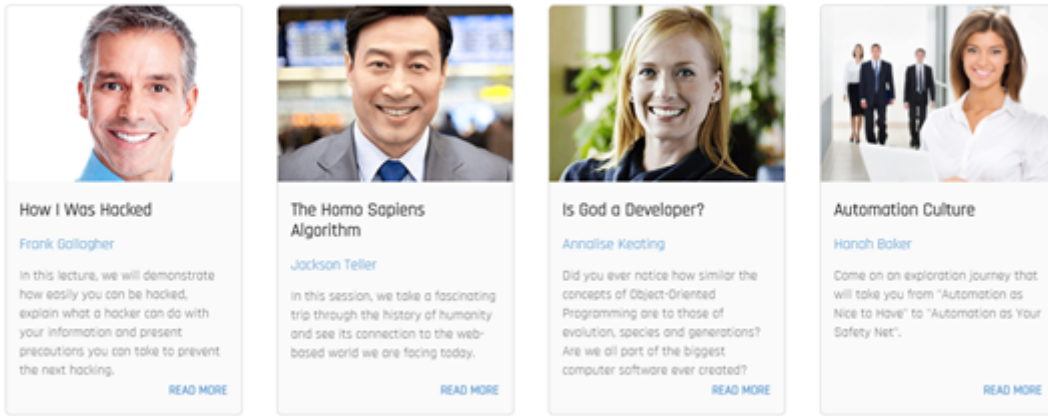
1. As web content editor, access the Web Content Editor tool.
2. Click the web content link under the relevant page to open the editor.

The editor will look like this:



3. Click **+** to add additional components.
4. Edit some of the elements.
5. **Save** and **Publish** your changes.

And what does the user see?



Translating Web Content Widgets

Developers prepare their web content widget texts for translation by adding them to property files. Developers can also add their own hard coded texts to the web content widgets and integrate these strings into the translation process.

How do I handle translation of web content widgets?

Web content widgets that are provided by SAP or those that are custom developed, are translatable. All the strings are listed in property files and are sent to a translation agency.

To see how these strings are prepared for translation in the `content.json` project file, see [The Web Content Widget Project Structure](#). Refer to the section about the `content.json` file in the `text` and `components` sections of the file.

For more information about the translation process, see [Managing Translations](#).

How do I handle translation of hard coded texts in web content widgets?

In some cases, developers may want to insert their own hard coded texts in the web content widgets. In this case, they should provide both the source language and the translations into a resource bundle. This is how it's done:

1. Go to your project in SAP Web IDE and add a new folder for the translation files. Name it `i18n`.
2. Add a new `i18n.properties` translation file to the folder.

iNote

You need to create a separate translation file for each language that you want to translate. Don't forget to add the correct suffix. For example: `i18n_en.properties` for English.

3. Edit the `component.json` file as follows:
 - a. Add the `applicationBundleName` property to the metadata like this:

↵ Sample Code

```
metadata : {
  manifest: "json"
  properties: {
    applicationBundleName: {
      type: "string",
```

```

                                defaultValue: "projectName.i18n.i18n"
                                }
                            }
    },

```

b. Set the `i18nApp` model in the `createContent` method like this:

≡ Sample Code

```

createContent: function() {
    var oI18nAppModel = new sap.ui.model.resource.ResourceModel({
        bundleName: this.getApplicationBundleName()
    });
    this.setModel(oI18nAppModel, "i18nApp");
    return new WebContentControl(this);
}

```

iNote

The model must be called `i18nApp`.

4. Add text to the HTML file. To support the translation of the hard-coded texts, add `{i18n?:TEXT_KEY: fallback text}` and change the KEY to the text key in your translation file. Here is an example:

- o **HTML file:**

≡ Sample Code

```

<div class = "article_item {i18n?:TEXT_KEY: fallback text}">

    <div class = "article_header {i18n?:TEXT_KEY: fallback text}">{header}</div>

    <div class = "article_text" {i18n?:TEXT_KEY: fallback text}> {text}</div>
</div>

```

- o **Translation file:**

≡ Sample Code

```

READ_MORE=Read More

```

iNote

For the web content widget preview feature and the widgets catalog, the fallback text key appears instead of the text itself.

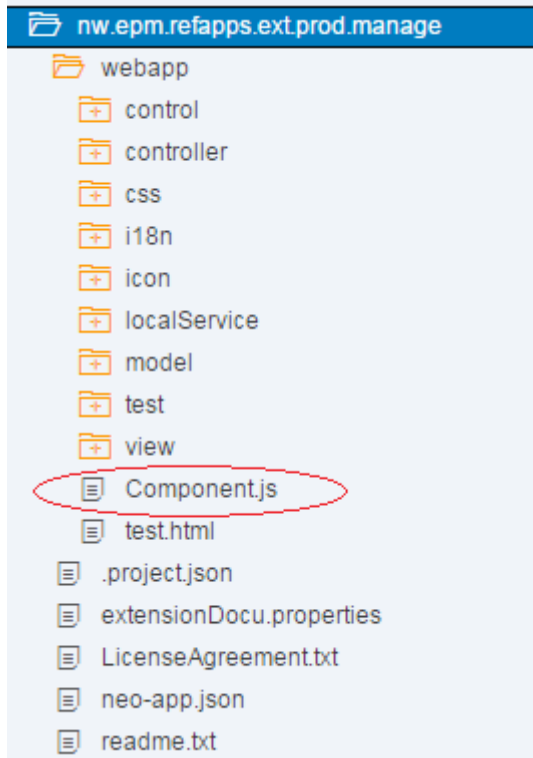
Convert an Application to a Widget

Developers can convert an existing HTML5 application to a Portal widget. The convert action adds two new segments to the app descriptor and the application component.

Prerequisites

You can convert an HTML5 application to a Portal widget only if the following constraints are met:

1. The HTML5 application has a `Component.js` file



2. The HTML5 application does not have a `cp.app.descriptor.json` file. This file is essential for application discovery in the Portal.
3. In the `project.json` file, the `projectType` does not include `sap.site.template`:

```

*.project.json x
1 {
2   "generation": [
3     {
4       "templateId": "sitetemplatePlugin.starterSiteTemplate",
5       "templateVersion": "1.0.0",
6       "dateTimeStamp": "Thu, 26 Nov 2015 14:34:45 GMT"
7     },
8     {
9       "templateId": "sitetemplatePlugin.starterPageTemplate",
10      "templateVersion": "1.0.0",
11      "dateTimeStamp": "Thu, 26 Nov 2015 14:35:14 GMT"
12    }
13  ],
14  "translation": {
15    "translationDomain": "",
16    "supportedLanguages": "en,fr,de",
17    "defaultLanguage": "en",
18    "defaultI18NPropertyFile": "i18n.properties",
19    "resourceModelName": "i18n"
20  },
21  "projectType": [
22    "sap.site.template"
23  ],
24  "hcpdeploy": {
25    "account": "x7750b7b0",
26    "name": "reginatemplate1",
27    "lastVersionWeTriedToDeploy": "1.0.0"
28  }
29 }

```

iNote

The `sap.mPage` container control is the default control when you create a SAPUI5 app. It is suitable for full page applications but when you convert it to a widget and then add it to a page section, it will cause rendering issues. We therefore strongly recommend using a layout control instead, for example, a grid layout.

Here is an example:

Sample Code

```

<mvc:View controllerName="com.testTestApp.controller.View1" xmlns:html="http://www.w3.org/199
  xmlns:layout="sap.ui.layout" xmlns:m="sap.m">
  <layout:Grid defaultSpan="XL4 L4 M6 S12">
    <layout:content>
      <layout:VerticalLayout class="sapUiMediumMargin news-tile">
        <m:Text text="SAP Unveils Powerful New Player Comparison Tool
      </layout:VerticalLayout>
      <layout:VerticalLayout class="sapUiMediumMargin news-tile">
        <m:Text text="SAP Unveils Powerful New Player Comparison Tool
      </layout:VerticalLayout>
      <layout:VerticalLayout class="sapUiMediumMargin news-tile">
        <m:Text text="SAP Unveils Powerful New Player Comparison Tool
      </layout:VerticalLayout>
      <layout:VerticalLayout class="sapUiMediumMargin news-tile">

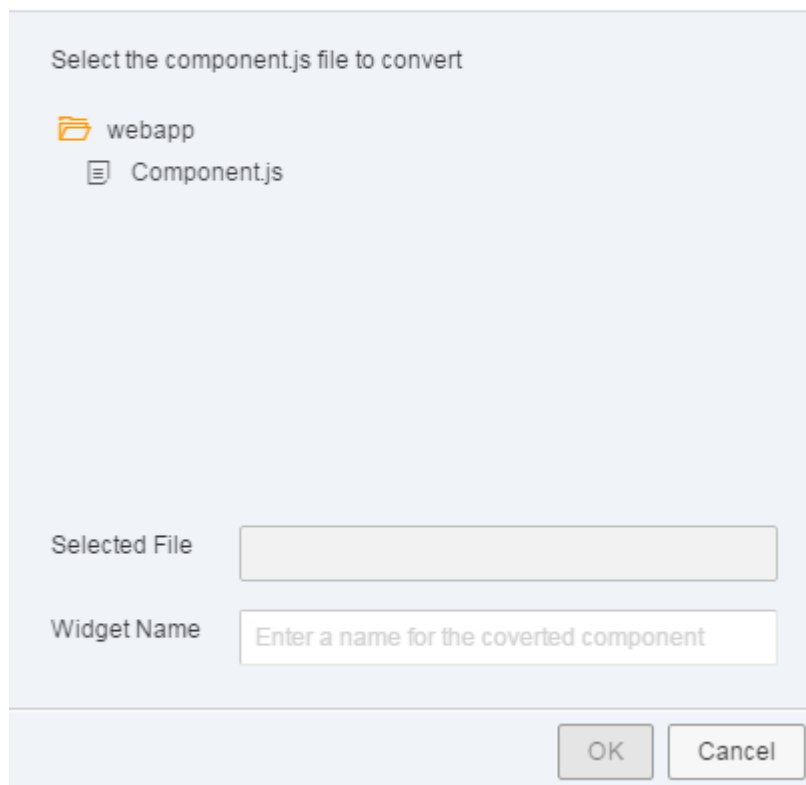
```

[illegible]

Procedure

1. Right click the application, for example, **nw.epm.refapps.ext.shop** **Convert to Portal Service Component**.
2. Select the `Component.js` file that you want to convert and provide a name for the widget.

Convert to Cloud Portal



Select the component.js file to convert

webapp

Component.js

Selected File

Widget Name

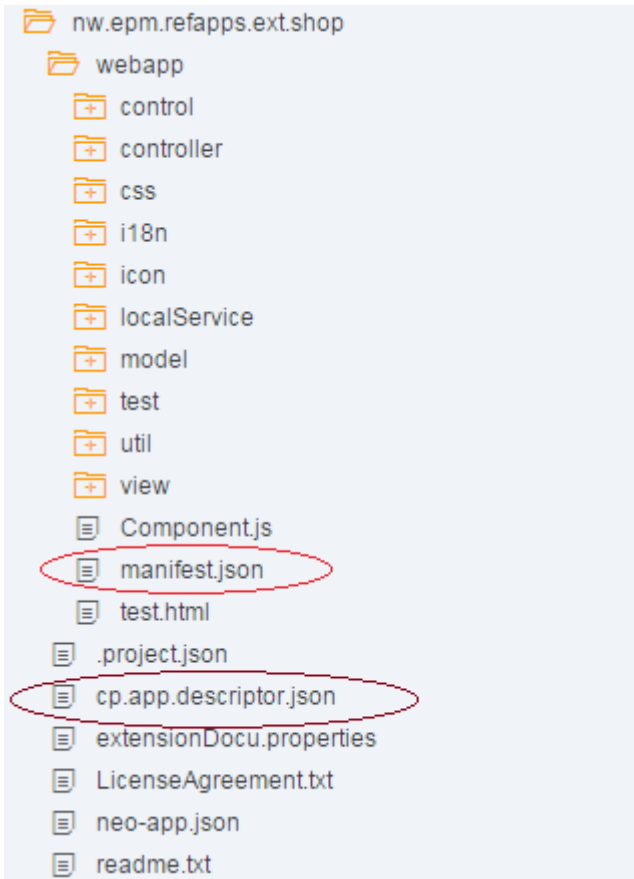
Enter a name for the converted component

OK Cancel

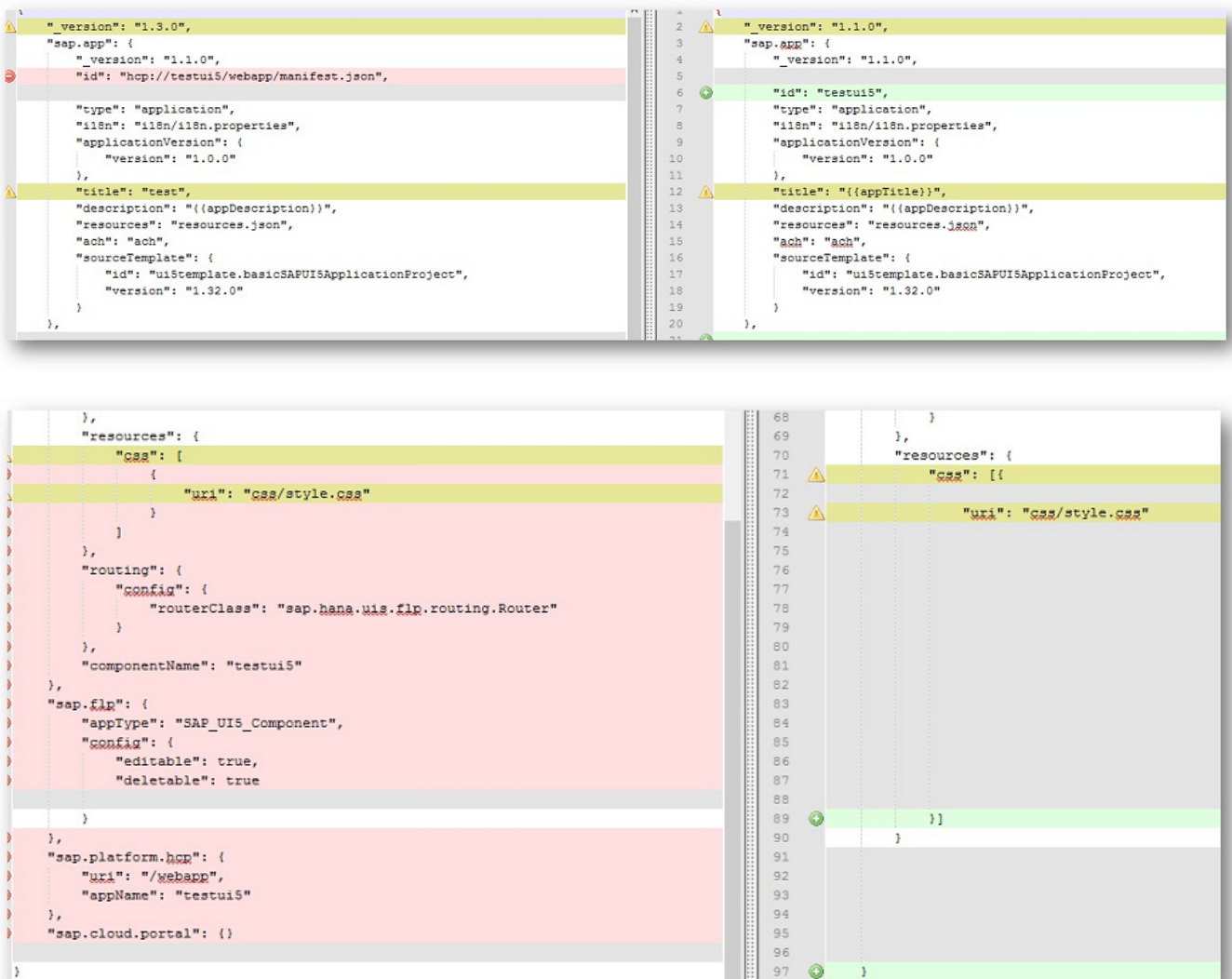
iNote

When converting an app to a widget, verify that the `UIComponent.extend` name is unique.

3. Click OK.
4. Notice that in the root of your project, an application descriptor with the name `cp.app.descriptor.json` was created. This file is essential for application discovery in the Portal.



The following screenshots illustrate the changes in the converted app code:



Results

The converted app appears in the list of content that the admin can add to a page.

iTip

We recommend that you verify that your app is working properly on the site before deployment. For more information, see [Preview Site Template](#).

iNote

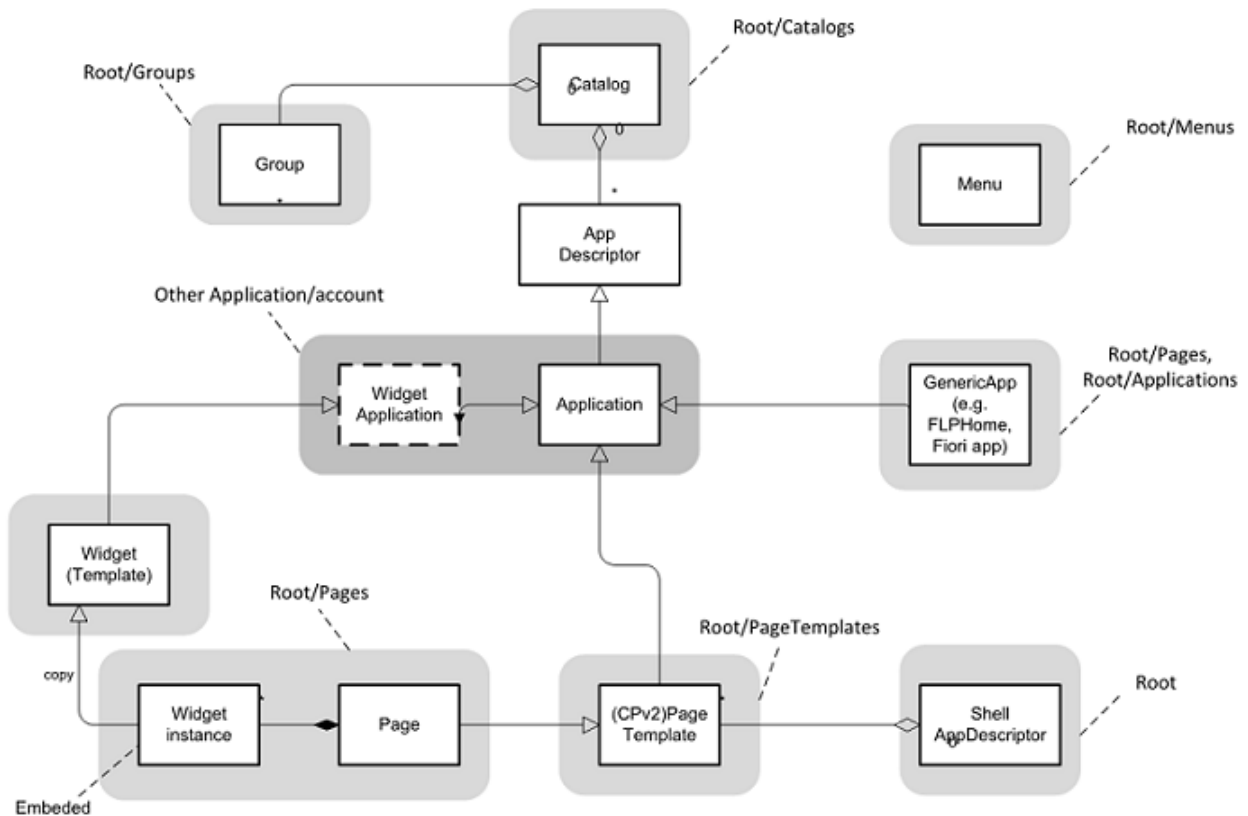
To edit the widget settings, use the Widget Settings API. For more information, see [Widget Settings API](#).

Related Information

[URL Helper API](#)

Project Structure

A site template structure contains page templates, pages, and widget templates. In addition, it includes also permission configuration, menus, groups, and design elements.



Related Information

[Project Components](#)

Project Components

Code samples of different Portal components, for reference purposes.

The application descriptor is inspired by the Web Application Manifest concept introduced by the W3C. The application descriptor provides a central, machine-readable and easy-to-access location for storing metadata associated with an application or application component. The data is stored in json format in the `manifest.json` file with attributes in different namespaces. It contains, for example, the app ID, the version, the data sources used, along with the required SAPUI5 components and SAPUI5 libraries.

The existence of the `manifest.json` file must be declared in the component metadata. The developer creates the `manifest.json` file and it is then delivered as part of the application archive. After delivery, the file is read-only.

In the SAP Cloud Platform Portal, you can find a `manifest.json` file for the following components: page template, widgets, and applications. Any instance of a component (such as a page, which is an instance of a page template) does not have an associated `manifest.json`, and its `json` file includes only the custom metadata.

Related Information

- [The cp.app.descriptor Entity](#)
- [The Site Template Entity](#)
- [The Page Template Entity](#)
- [The Page Entity](#)
- [The Catalog and Role Entities](#)

The cp.app.descriptor Entity

≡Sample Code

```
[{
  "id": "hcp://example/blanksite/site.json",//site.json id
  "name": "Example",//Site Template name
  "description": "",
  "thumbnail": "blanksite/thumbnail.png",//preview image of site template
  "path": "blanksite",
  "_entityType": "PORTAL_SITE_V2_TEMPLATE_SITE"
}]
```

The Site Template Entity

A detailed explanation about the site.json file sections.

Site.json Description

Section	Description	Use Case
payload->sap.cloud.portal->config->icon	Site thumbnail	Appears in authoring screens, the goal is to preview the template
payload->sap.cloud.portal->availablePageTemplates	List of the page templates that are part of this site template	Links between site template and its page templates

Section	Description	Use Case
identification->title	Site title	Appears in authoring screens
identification->description	Site description	Appears in authoring screens
identification->id	ID of the site template	Used by the <code>cp.app.descriptor.json</code> to reference the site template
payload->sap.cloud.portal->config->theme.id	Default theme of the site	Appears in Site Settings Default Theme . Theme ID must reference to a theme deployed on the system.
payload->sap.cloud.portal->config->theme.active	Available themes on the site	Appears in Site Settings Available Themes . Theme ID of all available themes must reference to themes deployed on the system.

Site Template Code Sample

Sample Code

```
{
  "_version": "1.0",
  "identification": {
    "id": "hcp://portalsitetemplates/blanksite/site.json",
    "namespace": "",
    "layer": "vendor",
    "title": "Starter Site",
    "description": "A site template that contains a single page. It is most suitable for the fre",
    "entityType": "site",
    "i18n": ""
  },
  "payload": {
    "homeApp": {
      "semanticObject": "page1",
      "action": "Display"
    },
    "config": {
      "ushellConfig": {
        "modulePaths": {
          "sap.ushell.apps.cpv2MenuPlugin": "/sap.ushell.apps.cpv2MenuPlugin"
        },
        "bootstrapPlugins": {
          "menuplugin": {
            "component": "sap.ushell.apps.cpv2MenuPlugin"
          }
        }
      }
    },
    "sap.cloud.portal": {
      "config": {
        "icon": "portalsitetemplates/blanksite/thumbnail.png",
        "isOnline": "false",
        "theme.id": "sap_bluecrystal",
        "theme.active": "[\"sap_bluecrystal\", \"sap_hcb\"]"
      },
      "availablePageTemplates": [
        {
          "namespace": "",
          "id": "hcp://portalsitetemplates/blanksite/pageTemplates/BlankPage/manifest.json"
        }
      ]
    },
    "_version": "1.2.0"
  }
}
```


The Page Template Entity

A detailed explanation about the page template manifest sections.

Page Template Manifest Description

Section	Property	Description	Use Case
sap.cloud.portal	nodes	Includes an array of sections that contain widgets	<p>A page template is a UI5 component with a view that may include one or many (sap.hana.uis.flp.control.Widget) controls.</p> <p>The Widget control has a property named alias. This property uniquely identifies an application that is rendered inside the Widget.</p> <p>Examples of a widget control that is linked with the page template manifest:</p> <ul style="list-style-type: none">Inside the page template view <pre>Sample Code<cpControls:Section alias="navigation"></cpControl</pre> <ul style="list-style-type: none">Inside the manifest.json <pre>Sample Code"nodes":{"navigation": { "sectionAlias": "page1.n</pre> <p>Every node should be configured with a unique section alias per page template</p>
	nodes>>section	Includes a widget manifest	

Page Template Code Sample

```
Sample Code
{
  "_version": "1.3.0",
  "sap.app": {
    "_version": "1.3.0",
    "id": "hcp://portalsitetemplates/blanksite/pageTemplates/BlankPage/manifest.json",
    "title": "Starter Page",
    "description": "A page layout that contain one predefined section",
    "tags": {
      "keywords": []
    }
  },
  "crossNavigation": {
    "inbounds": {
      "0": {
        "semanticObject": "obj",
        "action": "BlankPage",
        "deviceTypes": {
```

```

        "desktop": true,
        "tablet": true,
        "phone": true
    }
}
}
},
"sap.platform.hcp": {
    "_version": "1.2.0",
    "appName": "portalsitetemplates",
    "uri": "/blanksite/pageTemplates/BlankPage"
},
"sap.ui5": {
    "_version": "1.2.0",
    "componentName": "cp.templates.BlankPage"
},
"sap.cloud.portal": {
    "applicationType": "page",
    "thumbnail": "preview.png",
    "nodes": {
        //definition of widgets
        "section1": {
            //definition widget in section with alias "page1.header"
            "sectionAlias": "page1.header",
            "_version": "1.3.0",
            "sap.app": {
                "_version": "1.3.0",
                "id": "hcp://portalapptemplates/htmlwidget/manifest.json",
                "title": "HTML Widget",
                "description": "A widget template that displays HTML files.",
                "tags": {
                    "keywords": [
                        "keyword"
                    ]
                }
            },
            "crossNavigation": {
                "inbounds": {
                    "0": {
                        "semanticObject": "htmlwidget",
                        "action": "Display"
                    }
                }
            }
        }
    },
    "sap.platform.hcp": {
        "_version": "1.2.0",
        "appName": "portalapptemplates",
        "uri": "/htmlwidget"
    },
    "sap.cloud.portal": {
        "show.portal.settings": "false",
        "settings": {
            "markup": {
                "isRequired": "false",
                "storage_type": "html",
                "value": "<H2>Hello world<H2>",
                "displayName": "HTML",
                "description": "{{resource_url_description}}",
                "id": "markup"
            }
        }
    },
    "sap.ui5": {
        "_version": "1.2.0",
        "componentName": "htmlwidget.local"
    },
    "sap.flp": {
        "appType": "SAP_UI5_Component",
        "config": {
            "editable": true,
            "deletable": true
        }
    },
    "sap.ui": {
        "_version": "1.3.0",

```

```
        "technology": "UI5"
      }
    }
  },
  "sap.flp": {
    "tileType": "",
    "appType": "UI5",
    "config": {
      "editable": true,
      "deletable": true,
      "theme.id": "sap_bluecrystal",
      "theme.active": "[\"sap_bluecrystal\\\", \"sap_hcb\"]"
    }
  },
  "sap.ui": {
    "_version": "1.3.0",
    "technology": "UI5"
  }
}
```

The Page Entity

Pages extend page templates, and as such, the properties that are listed in the page.json file, extend the page template manifest.

Page.json Description

Section	Property	Description	Use Case
sap.app	baseId	Links the page instance with its parent page template	
	Title	Page title	
inbounds	semanticObject & action	Used to define the navigation to the page instance (should be unique for each page)	Enables navigation to the page

Page Code Sample

Sample Code

```
{
  "sap.app": {
    "_version": "1.1.0",
    "id": "hcp://example/blanksite/pages/page1.json",
    "baseId": "hcp://example/blanksite/pageTemplates/BlankPage/manifest.json", //id d
    "title": "Starter Page",
    "description": "A page layout that contain one predefined section",
    "crossNavigation": {
      "context": {},
      "inbounds": {
        "0": {
          "semanticObject": "page1",
          "action": "Display"
        }
      }
    }
  },
  "sap.cloud.portal": {
    "applicationType": "page"
  }
}
```

```

    }
}

```

The Catalog and Role Entities

Catalog Code Sample

In this code sample you can see that all pages in the site template are assigned to the default catalog Everyone.

≡ Sample Code

```

{
  "_version": "1.1",
  "identification": {
    "id": "hcp://portalsitetemplates/basiclayouts/catalogs/defaultCatalog.json",
    "title": "Everyone",
    "description": "",
    "entityType": "catalog"
  },
  "payload": {
    "appDescriptors": [
      {
        "namespace": "",
        "id": "hcp://portalsitetemplates/basiclayouts/pages/page1.json"
      },
      {
        "namespace": "",
        "id": "hcp://portalsitetemplates/basiclayouts/pages/page2.json"
      },
      {
        "namespace": "",
        "id": "hcp://portalsitetemplates/basiclayouts/pages/page3.json"
      },
      {
        "namespace": "",
        "id": "hcp://portalsitetemplates/basiclayouts/pages/page4.json"
      }
    ]
  }
}

```

Role Code Sample

In this code sample you can see that the default catalog is assigned to the role Everyone.

≡ Sample Code

```

{
  "_version": "1.0",
  "identification": {
    "id": "Everyone",
    "namespace": "",
    "entityType": "role"
  },
  "payload": {
    "catalogs": [{
      "namespace": "",
      "id": "hcp://portalsitetemplates/basiclayouts/catalogs/defaultCatalog.json"
    }],
    "groups": []
  }
}

```

Advanced Actions

Developer can perform advanced operations to further customize their templates.

When you create Portal components using the wizards provided in SAP Web IDE, you can choose any of the out of the box templates available in your subaccount as a basis for your own created templates. To add further customizations to the template layout, refer to the following instructions of how to perform advanced actions on your templates:

- [Add a Widget Placeholder in a Page Template](#)
- [Add a Section to a Page Template](#)
- [Add a Widget to a Section or a Widget Placeholder](#)
- [Add Custom Site Properties to Site Template](#)
- [Extend the Starter Widget](#)
- [Edit the Site Template Menu](#)
- [Preview Site Template](#)
- [Define a Site Template Catalog](#)
- [Upload Widget Initial Content to the Asset Repository](#)
- [Upload Content to the Asset Repository - Error Handling](#)

Add a Widget Placeholder in a Page Template

An example of how to add a widget placeholder to a page template.

Context

A widget placeholder is defined in a page template by a widget tag (`cpControl:Widget`). Each widget is referenced by an alias property.

Procedure

1. In the `view.xml` of the page template, add a container (`layout/Vbox/Hbox`).

For more information about possible containers, see the [SAPUI5 - Demo Kit](#).

2. In the container, add a definition for the widget's placeholder using the following format: `<cpControls:Widget alias="page1.widget1"></cpControls:Widget>`. The alias should be unique for the site.

iNote

When a widget placeholder is added to the `view.xml` but is not described in the `manifest.json` of the page template, the placeholder will be empty.

In this example, the page template contains two widget placeholders (`content` and `widget1`).

≡ Sample Code

```

<mvc:View controllerName="cpv2.templates.BlankPage.Template" height="100%" xmlns:cpControl:
  xmlns:l="sap.ui.layout" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m">
  <Page showFooter="false" showHeader="false" enableScrolling="true">
    <content>
      <FlexBox direction="Column" height="100%">
        <items>
          <l:VerticalLayout width="100%">
            <l:layoutData>
              <FlexItemData growFactor="1"/>
            </l:layoutData>
            <cpControls:Widget alias="page1.content"><,
              <VBox>
                <cpControls:Widget alias="page1.widget1"></cpConti
              </VBox>
            </l:VerticalLayout>
          </items>
        </FlexBox>
      </content>
    </Page>
  </mvc:View>

```

iTip

Use the site preview in order to see your changes. Right click the site template folder in your project structure and select **Run Run as Preview Site Template**.

3. Add a widget to each widget placeholder (only one widget is supported per placeholder). For more information, see [Add a Widget to a Section or a Widget Placeholder](#).

Add a Section to a Page Template

An example of how to add a section to a page template.

Context

What is a section?

A section is a custom layout control that aggregates several widgets.

Does your page template support sections?

You can open the `view.xml` of your page template (or any other supported view such as HTML View or JSView). If your page template contains `<cpControls:Section>` then sections are supported.

Procedure

1. In the `view.xml` of your page template, define a `<cpControls:Section>` with a unique alias.

In the following example, the page template contains three sections (header, content, and footer). Each section can contain one or more widgets.

Sample Code

```

<mvc:View controllerName="cpv2.templates.HeaderFooter.Template" height="100%" xmlns:cpConti
  xmlns:l="sap.ui.layout" xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m">
  <Page showFooter="false" showHeader="false" enableScrolling="true">
    <content>
      <cpControls:Section alias="Headerf

```

```

        <cpControls:Section alias="HeaderFooter.content"></cpCor
        <cpControls:Section alias="HeaderFooter.footer"></cpConi
    </content>

</Page>

</mvc:View>

```

2. Add widgets to each section. For more information, see [Add a Widget to a Section or a Widget Placeholder](#)

An example of a `manifest.json` that supports sections in a page template.

Sample Code

```

"sap.cloud.portal": {
  "applicationType": "page",
  "thumbnail": "preview.png",
  //Section details
  "sections": {
    "HeaderFooter.footer": {
      "widgets": {

        //Widget metadata
        "HeaderFooter.footer-rte": {
          "index": 1,

          "alias": "HeaderFooter.footer-rte" // identical to the sectionAlias in

        }
      },
      "layoutData": {

        //possible values for layout are "Column" , "Row" & "Custom"
        "layout": "Column",
        "itemsMargin": "0",
        "maxContentWidth": "1024",
        "minContentHeight": "150",

      }
    }
  }
}

```

To view an example of a section implementation, you can refer to the Support Site template provided in GitHub:

[Support Site](#) ➡

Add a Widget to a Section or a Widget Placeholder

An example of how to add a widget to a page section or to a widget placeholder in a page template.

Context

In the `manifest.json` of a page or page template, you first define a `<sectionAlias>MULTIDRAG` and then you add to it the widget's `manifest.json`.

You can also develop your own widget templates, and then add them to the page/page template.

Procedure

1. In the `manifest.json` of a page or page template, add a node that describes the section or the widget placeholder:

- Page - in the `page.json` of the page, first add a `sap.cloud.portal` block and inside it add a node entry.
- Page template - in the `manifest.json` of the page template, add a node entry inside the `sap.cloud.portal`.

A node for a widget placeholder:

≡ Code Syntax

```
"sap.cloud.portal": {
  "applicationType": "page",
  "thumbnail": "preview.png",
  "nodes": {
    "widget1": { //widget placeholder details
      "sectionAlias": "page1.widget1", //the alias must be identical to the one de
      // add the content of the widget's manifest.json here
    }
  }
}
```

A node for a section:

≡ Sample Code

```
"sap.cloud.portal": {
  "applicationType": "page",
  "thumbnail": "preview.png",
  //Section details
  "sections": {
    "page.section1": // identical to the sectionAlias in the view.xml
    "widgets": { //Widget definitions for the section
      "widget1": {
        "index": 1,
        "alias": "widget1" //the alias must be identical to the sectionAlias of
      }
    },
  },
  "nodes": {
    "widget1": { //widget placeholder details
      "sectionAlias": "widget1",
      // add the content of the widget's manifest.json here
    }
  }
}
```

The widget key node name (in this example, `widget1`) is meaningless but it must be unique in the `nodes` block. The section alias (in this example, `page1.widget1`) must be identical to the alias defined in the `view.xml` of this page template.

2. Copy the content of the widget's `manifest.json` as follows:

- Custom widget template - copy the content of the `manifest.json` immediately after the section alias.
- Widget templates available on a trial account - refer to the following table for code examples of the widgets' `manifest.json`. Also here, copy the content immediately after the section alias.

iNote

When you copy the widget content, make sure that you remove the external brackets `{}` of the `manifest.json` of the widget.

iNote

If the widget's `manifest.json` includes relative paths to resources, make sure that you update them to point to the location of the resources in your project structure.

Widget Type	Location of a Sample manifest.json
Image Widget	<code>https://portalapptemplates-[account name]trial.dispatcher.hanatrial.ondemand.com/imageWidget/manifest.json</code>
HTML Widget	<code>https://portalapptemplates-[account name]trial.dispatcher.hanatrial.ondemand.com/htmlwidget/manifest.json</code>
Rich Text Editor (classic, Tile Grid, List Builder)	<ul style="list-style-type: none"> ◦ Classic: <code>https://portalapptemplates-[account]trial.dispatcher.hanatrial.ondemand.com/rte/widget/widgets/rte/manifest.</code> ◦ Tile Grid: <code>https://portalapptemplates-[account]trial.dispatcher.hanatrial.ondemand.com/rte/widget/widgets/tilegrid/mani</code> ◦ List Builder: <code>https://portalapptemplates-[account]trial.dispatcher.hanatrial.ondemand.com/rte/widget/widgets/listbuilder/m</code>

To view an example of a section implementation, you can refer to the Support Site template provided in GitHub:

[Support Site](#) ➡

An example of a page template `manifest.json` that supports sections:

Sample Code

```
"sap.cloud.portal": {
  "applicationType": "page",
  "thumbnail": "preview.png",
  //Section details
  "sections": {
    "HeaderFooter.footer": {
      "widgets": {

        //Widget metadata
        "HeaderFooter.footer-rte": {
          "index": 1,

          "alias": "HeaderFooter.footer-rte",

          //This part is needed only for "Custom" layout
```

```

        "layoutData": {
            "L": 4,
            "M": 6,
            "S": 12
        }
    },
    "layoutData": {

        //possible values for layout are "Column" , "Row" & "Custom"
        "layout": "Column",
        "itemsMargin": "0",
        "maxContentWidth": "1024",
        "minContentHeight": "150",

        //metadata for Row Layout
        "itemsPerRowDesktop": "2",
        "itemsPerRowTablet": "2",
        "itemsPerRowPhone": "3"
    },
    "appearance": {

        "backgroundColor": "transparent", // possible value 'transparent' or hex for
        "backgroundImageSource": "2",
        "useBackgroundImage": true,
        "backgroundImageState": "1",
        "backgroundImageAlignment": "4",
        "fixedBackgroundImage": false
    },
    "resources": {
        "assets": [
            {
                "value": "/assets/img/Footer/footer_image.jpg"
            }
        ]
    }
}

```

An example of a page template manifest.json with a widget placeholder that contains an image widget.

Sample Code

```

{
    "_version": "1.3.0",
    "sap.app": {
        ...
    },
    "sap.cloud.portal": {
        "applicationType": "page",
        "nodes": {
            "main_image": {
                "sectionAlias": "page1.main_image",

```

```
//Widget's manifest.json starts here
"_version": "1.3.0",
"sap.app": {
  "_version": "1.3.0",
  "id": "hcp://starter/starter/widgetTemplates/widget",
  "title": "image",
  "description": "A widget template that displays images",
},
"sap.platform.hcp": {
  "_version": "1.2.0",
  "appName": "starter",
  "uri": "/starter/widgetTemplates/widget"
},
"sap.cloud.portal": {
  "show.portal.settings": "false",
  "settings": {
    "source": {
      "value": "url"
    },
    "resources": {
      "url": [{
        "value": "/images/default_image.png"
      }]
    },
    "state": {
      "value": "fit"
    },
    "height": {
      "value": 100
    },
    "units": {
      "value": "pt"
    },
    "alignment": {
      "value": "center_middle"
    }
  }
},
"sap.flp": {
  "config": {}
},
"sap.ui": {
  "_version": "1.3.0",
  "technology": "UI5",
  "icons": {
    "icon": "sap-icon://image-viewer"
  }
},
"sap.ui5": {
  "_version": "1.2.0",
  "componentName": "sap.cp.widgets.image",
  "routing": {
    "config": {
      "routerClass": "sap.hana.uis.flp.router"
    }
  }
}
```

You can define only one widget in a given widget placeholder, therefore, to add more widgets, add more node blocks to the page/page template. If your page template supports sections, you can create a section and place several widgets in it. For more information about sections, see [Add a Section to a Page Template](#).

Use the site preview to see your changes. Right click the site template folder in your project structure and select **Run** **Run as** **Preview Site Template**.

Supported Properties

Add Custom Site Properties to Site Template

Developers can add key-value pairs of properties, for customizing site implementations.

Procedure

1. In the `site.json` of the site template, add a `settings` entity inside the `config` entity, under the `sap.cloud.portal`.
2. Add key-value pairs of properties.

Sample Code

```
"sap.cloud.portal": {
  "_version": "1.2.0",
  "config": {
    "isOnline": "false",
    "icon": "portaltemplatesamples/SiteTemplates/mySiteTemplate/thumbnail.png",
    "theme.id": "myTheme", //Optional
  }
}
```

```

    "settings": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    }
  }
}

```

iNote

Once you deploy the site template to your subaccount, the custom properties appear automatically under the **Site Settings Custom Site Properties** of the Site Designer.

Use Custom Site Properties in a Site Template

Procedure

1. Open the `controller.js` of the object (widget/page) where you want to use the custom settings, and add custom implementation.
2. To get the settings, use the `siteService` API.

For more information about the `siteService` API and its methods, see [API Documentation](#).

Sample Code

```

var siteService = sap.ushell.Container.getService('SiteService'),
    siteSettings = siteService.getSiteSettings();

if (siteSettings && siteSettings["key1"] && siteSettings["key1"] === "value1"){
    return "Setting found";
}

return "Settings not found";

```

Extend the Starter Widget

A walkthrough procedure for customizing the Starter widget provided by SAP.

Context

The easiest way to create custom widgets is to use the out of the box **Starter** widget provided by SAP as a basis, and modify its code.

In this example, we will create an image widget with zoom-in functionality.

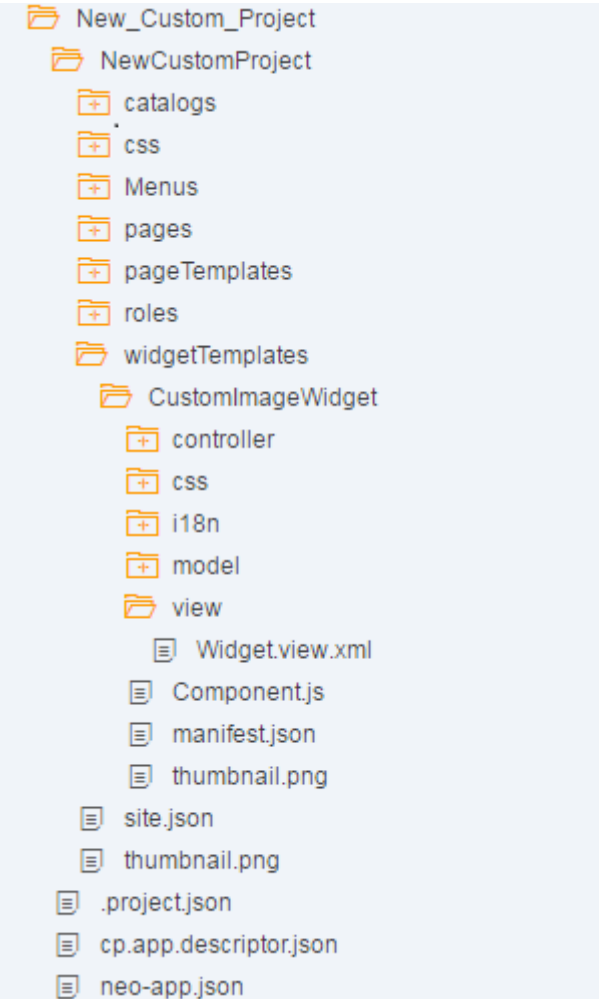
Procedure

1. In SAP Wb IDE, create a new site template as follows:
 - a. Name the project `New_Custom_Project`.
 - b. Select the **Starter** site template from the list of available site templates.

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

- c. Name your new site template `NewCustomProject`.
- 2. Right-click the site template folder, and select **New More Widget Templates** , and create a new widget based on the Starter Widget.
 - a. Name the folder `CustomImageWidget`.
 - b. Name the new widget `Custom Image Widget`.

Your project structure should look similar to this:



The Starter widget includes the following components:

File Name	Description
Component.js	This is the widget main SAPUI5 Component file.
manifest.json	The widget's manifest.json file, containing the widget configuration.
view>>Widget.view.xml	The view of the widget itself, rendered in a page.
controller>>Widget.controller.js	The controller of the widget
css>>style.css	The widget stylesheet file, contains the unique widget style settings.
i18n>> i18nApp.properties	The widget's resource bundle, containing strings for translation.

- 3. Customize the widget.
 - a. Create a folder for images inside the `CustomImageWidget` folder, and add an image to it. In this example it is `colorful_flowers.jpg`.

- b. Open the `manifest.json` of your widget template and in the `settings` section, add a property `url` that points to the image you just added.

≡,Sample Code

```
"sap.cloud.portal": {
  "show.portal.settings": "false",
  "settings": {
    "url": "/images/colorful_flowers.jpg"
  }
}
```

- c. Open the `model.js` file and add the `createWidgetModel()` method.

≡,Sample Code

```
createWidgetModel: function() {
  var oModel = new JSONModel(Device);
  return oModel;
}
```

- d. Open the `Component.js` file and set the widget model inside the `init()`.

≡,Sample Code

```
init: function() {
  // calls the base component's init function
  UIComponent.prototype.init.apply(this, arguments);

  // sets the device model
  this.setModel(models.createDeviceModel(), "device");
  this.setModel(models.createWidgetModel(), "widget"); //your cu
}
```

- e. Set the widget model in the `Widget.controller.js` file.

≡,Sample Code

```
onSettingsLoaded: function(settings) {
  var oModel, url,
  viewId = this.getOwnerComponent().getId();
  oModel = this.getView().getModel("widget");
  url = sap.ushell.Container.getService("URLHelper").createComp
  oModel.setData({
    src: url
  });
},
```

- f. Open the `Widget.view.xml` file and add an SAPUI5 image control.

≡,Sample Code

```
<mvc:View controllerName="CustomImageWidget.hcp.portal.service.starter.controller.Wid
  xmlns:mvc="sap.ui.core.mvc" xmlns="sap.m">
  <Image class="imageWidget" src="{widget>/src}"
    mode="Background"
    width="100%"
    height="250px"
    backgroundSize="auto"
```

</mvc:View>

g. Open the `Widget.controller.js` file and add an `onImagePressed` method.

≡ Sample Code

```
onImagePressed: function () {
    var dialog = new sap.m.Dialog({
        draggable: true,
        showHeader: false,
        beginButton: new sap.m.Button({
            text: 'Close',
            press: function () {
                dialog.close();
            }
        }),
        afterClose: function() {
            dialog.destroy();
        }
    });
    var src = this.getView().getModel("widget").getData().src;
    dialog.addContent(new sap.m.Image({
        src: src
    }));
    this.getView().addDependent(dialog);
    dialog.open();
}
```

4. Test the functionality of your widget template.

a. Add the widget template to a section of a page/page template.

For more information see, [Add a Widget to a Section or a Widget Placeholder](#).

b. Preview the site template by right-clicking **Run Run As Preview Site Template**.

c. Click the image in the widget (while in preview mode) and verify that a zoom-in dialog opens.

Edit the Site Template Menu

Developers can customize the site template menu, provided by SAP.

Context

A site template default menu has the following structure:

```
{
  "_version": "1.0",
  "identification": {
    "id": "menu.json",
    "entityType": "menu",
    "i18n": "i18n/menu.properties"
  },
  "payload": {
    "locked": "false",
    "tags": [],
    "entities":[] //definition of nested items
  }
}
```



```

    }
}

```

Procedure

1. Open the `menu.json` file, which is located under the `Menus` folder.
2. Add a new menu item for (page, app, folder, or url) inside the `entities` property, under the `payload` block.

Use the following structure:

```

{
  "title": "", //title
  "entityType": "", //possible values PAGE/URL/FOLDER/APP
  "entities": [], //definition of nested items
  "target": { //navigation target
    "semanticObject": "",
    "action": ""
  }
},

```

If there are no subitems, remove the `entities` property.

3. Define the navigation target as follows, according to the item's type:
 - To define a navigation target to a page, go to the `page.json` of the page you want to use as a target, and copy the semantic object value to the menu:

```

"target": {
  "semanticObject": "page1",
  "action": "Display"
}

```

- To define a navigation target to an app, go to the `app.json` that you want to point to, and copy the semantic object value to the menu:

```

"target": {
  "semanticObject": "ServiceRequests",
  "action": "Display"
}

```

- To define a navigation target to a folder, use:

```

"target": {
  "semanticObject": "",
  "action": ""
}

```

- To define a navigation target to a URL, use:

```

"target": {
  "semanticObject": "",
  "action": "http://www.google.com" //url
}

```

Here is an example of site menu that uses all types of entities:

≡ Sample Code

```
{
  "_version": "1.0",
  "identification": {
    "id": "menu.json",
    "entityType": "menu",
    "i18n": "i18n/menu.properties" //relative path to master language properties
  },
  "payload": {
    "locked": "false",
    "tags": [],
    "entities": [{
      "title": "{{page_title}}",
      "target": {
        "semanticObject": "page1",
        "action": "Display"
      },
      "entityType": "PAGE"
    }, {
      "title": "{{app_title}}",
      "target": {
        "semanticObject": "ServiceRequests",
        "action": "Display"
      },
      "entityType": "APP"
    }, {
      "title": "{{folder_title}}",
      "target": {
        "semanticObject": "",
        "action": ""
      },
      "entityType": "FOLDER"
    }, {
      "title": "{{url_title}}",
      "target": {
        "semanticObject": "",
        "action": "http://www.google.com"
      },
      "entityType": "URL"
    }
  ]
}
```

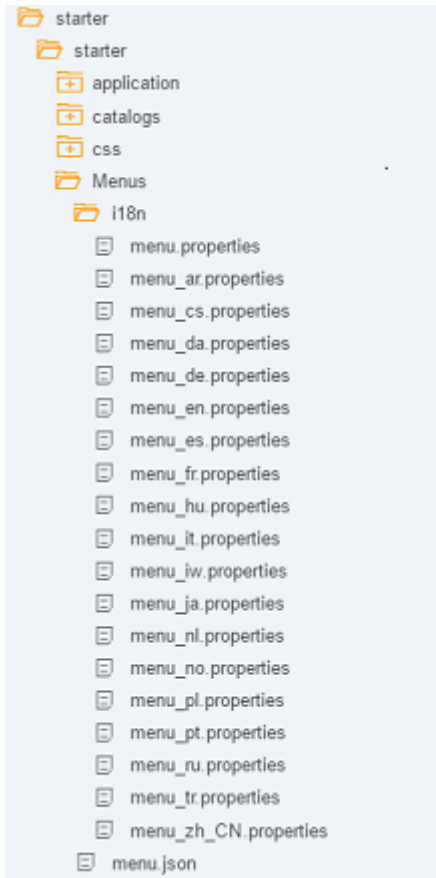
How to Translate Menu Items

Context

To translate the site menu, make the following adjustments.

Procedure

1. Wrap the menu item's title and description with curly brackets `{{<title>}}`.
2. Add the title translations to the `menu.properties` file in the **Menus i18n** folder. Use the same title key as you defined inside the menu item.



How to Remove the Site's Default Menu

Context

You can hide the default menu in your site by removing the `bootstrapPlugins` entity in the `config` block of the `site.json`.

```
{
  "_version": "1.0",
  "identification": { },
  "payload": {
    "homeApp": { },
    "config": {
      "ushellConfig": {
        "modulePaths": {
          "sap.ushell.apps.cpv2MenuPlugin": "/sap.ushell.apps.cpv2MenuPlugin"
        },
        "bootstrapPlugins": {
          "menuplugin": {
            "component": "sap.ushell.apps.cpv2MenuPlugin"
          }
        }
      }
    }
  },
  "sap.cloud.portal": { }
}
```

iTip

Use the site preview to see your changes. Right click the site template folder in your project structure and select **Run Run as Preview Site Template**.

Preview Site Template

Developers can preview their site template in SAP Web IDE before deploying it to the subaccount.

Context

The preview action in SAP Web IDE enables you to preview the site template that you are working on in the development environment, and verifying its look and feel before deploying the final version to the subaccount. This is a very helpful capability that saves you the need to create many draft sites just for verification purposes.

In preview mode, no permissions are taken into account. That means that you will view all pages and apps regardless of their assigned roles. Another important indication is that by default all site pages, except for the home page, are hidden in the site menu. To preview the site template and all its pages, verify that the pages are visible in the site menu.

iNote

SAP Web IDE preview assumes that the site's SAPUI5 version is **Innovation**, which is the latest released version of SAPUI5. If the site template contains controls or a theme that were created based on an older SAPUI5 version, this may result in compatibility issues.

Procedure

1. In SAP Web IDE, expand your site template project.
2. Right click anywhere on project tree and select **Run Run As Preview Site Template**.

iNote

If you edit the layout (XML View, CSS), you can refresh the browser and immediately view the changes. If you edit the site structure (any json file, such as menu, page template, page, and so on), then you need to run the **Run Run As Preview Site Template** again to view the changes.

iNote

The **Run As** menu option uses the default provider subaccount, on which the site template was created. If you want to use a different provider subaccount, you need to use **Run Configurations** before previewing the site template.

iTip

Another very useful aspect of the preview, is the ability to verify the functionality of a converted app. To verify that your app is working properly on the site, you can test it using **Run Run Component.js**. However, this verification method is limited to simple use cases where you are not using any portal services in your converted app. Therefore, it is recommended to create a simple site template (based on the Starter site template), add your converted app to it, and preview the site template with the app.

Define a Site Template Catalog

Developers can define catalogs in their site template. Catalogs include definitions about pages and apps that are assigned to a role entity, which sets the access rights to the content.

Example

Sample Code

```
{
  "_version": "1.0",
  "identification": {
    "id": "hcp://example/blanksite/catalogs/Catalog1.json",
    "description": "",
    "entityType": "catalog"
  },
  "payload": {
    "appDescriptors": [{
      "namespace": "",
      "id": "hcp://example/blanksite/pages/page1.json"
    }]
  }
}

{
  "_version": "1.0",
  "identification": {
    "id": "Anonymous",
    "namespace": "",
    "entityType": "role"
  },
  "payload": {
    "catalogs": [{
      "namespace": "",
      "id": "hcp://example/blanksite/catalogs/Catalog1.json"
    }],
    "groups": []
  }
}
```

Upload Widget Initial Content to the Asset Repository

Developers can define initial content for their widgets. That content will be automatically uploaded to the Asset Repository and can be reused.

Context

You can define initial content for a widget such as a default image, a default HTML, and so on, that will show up when the administrator builds the site. One option is to include this content (images, HTMLs, documents) in the widget code, however this will make the widget code very long and unreadable. A better option is to store the content somewhere else and then to point to it from the widget settings.

If you choose to store the content somewhere else, the location of the initial content in the project structure of SAP Web IDE is meaningful. If you want the content to be available for any site created from this site template, place the initial content at the site level. If you want the content to be available for any widget created from this widget template (even in different sites and accounts), place the initial content at the application level (widget template).

Invoking the content by either creating a site instance or adding a widget to a site instance, will automatically upload the content to the Asset Repository. In addition, you can prepare translation for the content and store it side by side with the original file. As soon as the administrator activates that language, the translated content will become available for end users.

Defining Initial Content at the Site Level

Context

Use this option if you want the initial content to be available for any site created from the site template.

Procedure

1. Create a folder at the project level (same level as Pages, PageTemplates, and so on).

iNote

You can upload multiple assets to a folder. To define more than one asset in a folder, add an array definition.

2. In the section of the page/page template, where you define the widget, add to the widget settings an asset using the following format: "value": "/resourceId".

≡ Sample Code

```
"sap.cloud.portal": {
  "settings": {
    "keyName": {
      "displayName": "My prop 1",
      "value": "starting value"
    },
    "resources": {
      "assets": [
        {
          "value": "/images/myImage.jpg" //the name of the resource as appears in the resource
          "isCloned":true //use this property if you want to duplicate the resource. In this ca
        }
      ]
    }
  }
}
```

Defining Initial Content at the Application Level

Context

Use this option if you want the initial content to be available for any widget created from the widget template.

Procedure

1. Create a folder inside the specific widget template tree (Your widget template >> folder).

iNote

You can upload multiple assets to a folder. To define more than one asset in a folder, add an array definition.

2. In the `manifest.json` of the widget template, under settings, define an asset using the following format: `"value": "/resourceId"`.

Sample Code

```
"sap.cloud.portal": {
  "show.portal.settings": "false",
  "settings": {
    "markup": {
      "isRequired": "false",
      "storage_type": "assets",
      "displayName": "HTML",
      "description": "resource_url_description",
      "id": "markup"
    },
    "resources": {
      "assets": [
        {
          "value": "/htmlResource/Loc
          "isCloned": true
        }
      ]
    }
  }
},
```

Adding Translation to the Initial Content

Context

You can provide translated content and place it at the same location where you stored the original content.

Each translated asset file should have the corresponding locale (language-country) in the file name using the following naming convention: `<asset name>_<Language>[.<extension>]` or `<asset name>_<Language>-<Country>[.<extension>]`. For example: `MyImage.jpg` and `MyImage_de-DE.jpg`.

When the project is deployed to the subaccount, as soon as the site administrator turns on the relevant language, your translated content becomes available for end users.

Note

Translation of initial content at the application level is created only if the relevant locale is already turned on when the widget is added to the site.

Upload Content to the Asset Repository - Error Handling

Widget developers can add error handling when uploading content to the Asset Repository.

Context

When developing widgets, we recommend that you persist static resources in the Asset Repository using the Asset Service API. When retrieving the resources from the Asset Repository, an error might occur. In the site template preview in SAP Web IDE, you get the following indications:

- A popup with an error message indicating that the resources could not be retrieved requests that you check the logs for more information.
- Information about incorrect resources in the log prompts you to open **View - Console**, and check the logs for more information.
- Error message in the wrong section with appropriate error (Could not find the selected resource for the widget. Select a different resource).

Error messages and possible solutions about resource retrieval are available in the Site Designer for administrators.

However, end users do not receive any indication about incorrect or problematic resources. Therefore we recommend that you provide callbacks in the widget itself as follows:

Procedure

1. Open the `controller.js` file of your widget for editing.
2. Add handlers. For example, you can modify the method `onSettingsLoaded` and add `done` and `failed` handlers.
3. Define the behavior when the deferred object is rejected or resolved while retrieving a resource from the Asset Repository using the `AssetService` API.

≡,Sample Code

```
onSettingsLoaded: function () {
    ....

    var assetService= Container.getService('AssetService');
    siteService = Container.getService('SiteService');
    siteId = siteService.getSiteID();

    assetService.getAsset('assetName', siteId)
        .done(function (response) {
            // write your code here

        }.bind(this))
        .fail(function () {
            // provide a proper error handling here

        }.bind(this));
    }
    . . .
}
```

For more information about using the Asset Service API, see [Asset Service API](#)

Consume a Java Application in a Site

Developers can add a Java application to their sites, and view it as a full page application or as a widget.

Context

To be able to add a Java application to your site, complete the following steps:

- Create an HTML5 application.
- Add code that wraps the client-side of the Java application as an SAPUI5 component.
- Deploy the application to the cloud.
- Set a destination that points to the Java application.
- Add the application to the site using the Site Designer tool.

Procedure

1. Create an HTML5 application.

- In the SAP Cloud Platform cockpit, open the [HTML5 Applications](#) tab, and click [New Application](#).
- Provide a name for the application and click [Save](#) to create it.
- Open the application (the application table lists all applications alphabetically) by clicking it.

2. Modify the application code.

- In the application screen, go to the [Versioning](#) tab, under [Source Location](#), click [Edit Online](#). This opens the application for editing in SAP Web IDE.
- If you get a [Clone Repository](#) dialog, enter your user name and password, and click [OK](#).
- Add the following code to your app, according to the app view in the site:

Application View	Required Code Change
Full page application	<p>Add a <code>neo-app.json</code> file to your project, with the following code:</p> <pre> { "routes": [{ "path": "/javasample", "target": { "type": "destination", "name": "javasampledest", "entryPath": "/" } }] }</pre>

Application View	Required Code Change
Widget (the application appears in a page's section)	<p>i. Add a <code>neo-app.json</code> file to your project, with the following code:</p> <pre> { "routes": [{ "path": "/javasample", "target": { "type": "destination", "name": "javasampledest", "entryPath": "/" } }] }</pre> <p>ii. Add a <code>cp.app.descriptor.json</code> file to your project, with the following code:</p> <pre> [{ "id": "hcp://javaapplication/javasample/manifest.json", "_entityType": "PORTAL_SITE_V2_TEMPLATE_APP_CONFIG", "path": "javasample", "name": "Some Widget Name", "thumbnail": "", "preview": "" }]</pre>

3. Deploy your application to your subaccount.

4. Configure a destination to the Java application.

- a. In the SAP Cloud Platform cockpit, open the [Destinations](#) screen.
- b. Add a new destination with the name `javasampledest`, and a URL that points to the folder where the application's `Component.js` file is located.

Sample Code

```
#
#Sun Jan 29 13:32:14 UTC 2017
Name=javasampledest
URL=https://poccpv2javaappcpdev.int.sap.hana.ondemand.com/PocCpv2JavaApp/
ProxyType=Internet
Type=HTTP
CloudConnectorVersion=2
Authentication=NoAuthentication
```

5. Add the application to your site.

- a. To add the application as a full-page app:
 - i. In the Site Designer, **Page and Apps** panel, click the **+** at the bottom of the list on the left, to add a new app.
 - ii. From the **Account Apps** list, select your app and add it. This adds the app under the **Apps** list on the left.

- iii. From the actions menu of the app, select **Add to Menu**.

iNote

Make sure that your app is assigned to a catalog. Only users who have their role assigned to that catalog will be able to access your app.

- b. To add the application as a widget:

- i. In the Site Designer, **Page and Apps** panel, click the page under the **Pages** list, to which you like to add this app.
- ii. Click the page section where you want to place the app as a widget, and then click again the section header to show the section menu.
- iii. In the section menu, click the **+** button to open the list of widgets, and add your app from the list.

6. Set the application type to **SAPUI5**.

- a. In the Site Designer, **Pages and Apps** panel, click the app entry in the list on the left.
- b. In the app editor that opens, click **Edit** and change the **App Type** from URL to SAPUI5.
- c. Add the relevant data under **Component URL** and **SAPUI5 Component**, and click **Save**.

Related Information

[URL Helper API](#)

Access Remote Content from an Application

Fiori application development may target multiple platforms, such as ABAP and SAP Cloud Platform as well as multiple subaccounts in a cloud environment. The Unified Shell services mechanism allows you to develop applications in a platform-independent manner. It is achieved by exposing the same services API with potentially different implementation per platform.

When the application is running on different platforms and systems, the location of static resources and paths to data sources may differ. Even when running on the same platform, such as SAP Cloud Platform, the application may be consumed by different subaccounts, leading to different application URLs according to subscription configuration. As a result, it is strongly recommended to avoid using hardcoded platform dependent URLs in the application code.

A recommended practice is to use dynamic construction of platform dependent URLs. There are several use cases when URL should be constructed:

- When an application needs to access a static resource that is part of that application, and the relative path to that resource is known in advance. Another option is that the application needs to point to its own defined route. Relative URLs in JavaScript are relative to the HTML page, and when applications are loaded from different location, using relative URLs might cause unexpected results.
- When an application needs to access named data source, and the path to it is dynamic and may be calculated in a platform-dependent way.

Fiori technology promotes a declarative way of defining application dependencies, including static resources and data sources. Application configuration is defined by developers in the `manifest.json` file. Providing a `manifest.json` for Web applications is an open Internet standard adopted and extended by SAP.

Ajax Calls to Inner Route of the Application

For Ajax calls to inner route of the application (as defined in the `neo-app.json`):

Sample Code

```
sComponentId = this.getId();

sPath = "/myservice";
// In case running as Fiori Launchpad container
if (sap.ushell.Container) {
// /sap/fiori/app_subscription_name/path/to/component/my
sURI = sap.ushell.Container.getService("URLHelper").crea

} else { // in case running as standalone
sURI = jQuery.sap.getModulePath(sComponentId) + sPath;
}
jQuery.ajax({
url: sURI,
data: {
"some": "data"
},
""
})
```

Data Sources and Models

For data sources and models the same principles applied:

- We recommend to define all data sources used by application in the `manifest.json` file under `"sap.app"` section. For example on SAP Cloud Platform, the following configuration allows you to specify a path to equipment data source that should include doorway mapping information that is not available during development time:

Sample Code

```
"sap.app":

"dataSourcees": {
"equipment": {
"type": "JSON",
"uri": "./equipment"
}
}
}
```

Note the `"/` prefix that notifies SAPUI5 that this is a dynamic URI that should be resolved.

- Most SAPUI5 applications rely on data binding between views and models. There are different ways to create the models. However, the preferable way is to define them in the `manifest.json` file, under the `"sap.ui5"` section and map them to data sources. For example:

Sample Code

```
"sap.ui5": {

"models": {
"equipment": {
```

```
"dataSource": "equipment"
}
}
}
```

This triggers SAPUI5 to automatically create a model called `equipment` during the component initialization phase. When you can't use an SAPUI5 model (when a custom AJAX call should be performed), you can also use the Unified Shell `URLHelper` service to create URLs that are calculated with the same rules as a URI of data source in the `manifest.json` file. For example, in the `Component.js` it is possible to do the following:

Sample Code

```
sComponentId = this.getId();

sDataSourceKey = "equipment";
sPath = "/materials/1/parts";
sURI = sap.ushell.Container
```

The value of `sURI` in this case may be similar to `"/sap/fiori/app_subscription_name/path/to/component/equipment/materials/1/parts?skip=10&limit=10"`.

Related Information

[URL Helper API](#)

Supported Properties

List of supported properties.

Image Widget Properties

Property Name	Supported Values
source	url, asset_repository
state	original, cropped, contained, stretch, fit
alignment	left_top, center_top, right_top, left_middle, center_middle, right_middle, left_bottom, center_bottom, right_bottom

Section Appearance Properties

Property Name	Supported Values
"backgroundColor"	Transparent, color in hexa format
"backgroundImageSource"	url, asset_repository

Property Name	Supported Values
"useBackgroundImage"	Boolean
"backgroundImageState"	original, cropped, contained, stretch, fit
"backgroundImageAlignment"	left_top, center_top, right_top, left_middle, center_middle, right_middle, left_bottom, center_bottom, right_bottom
"fixedBackgroundImage"	Boolean

Section Layout Properties

Property Name	Supported Values
"layout"	Column, Row, Custom
Metadata for all layouts: "itemsMargin", "maxContentWidth" "minContentHeight"	Numbers
Metadata for Row: "itemsPerRowDesktop" "itemsPerRowTablet" "itemsPerRowPhone"	Numbers
Metadata for Custom: "layoutData": {"L", "M", "S"}	Numbers

Portal Routing Service

The process of converting an HTML5 application to a Portal widget, includes also overriding of the router class definition in the manifest.json file.

The Portal router class extends the SAPUI5 router class. The motivation for overriding the SAPUI5 router class is to enable rendering of multiple widgets on the same page that uses the navTo API.

The change in the manifest file is as follows:

≡ Sample Code

```
"sap.ui5": {
  "_version": "1.1.0",
  "routing": {
    "config": {
      "routerClass": "sap.hana.uis.flp.routing.Router"
    }
  }
}
```

Portal Router API

- **navTo** - the same behavior as the SAPUI5 router, with one exception - the new route is not reflected on the browser URL address. For reference see: [SAPUI5 API Documentation](#)
- **back** - navigates to the last inner path, detached from the browser history.

Sample Code

```
sap.ui.controller("MyApp.View", {
    anyEvent: function() {
        sap.ui.core.UIComponent.getRouterFor(this).back();
    }
});
```

Developers that create a full page application (not a widget), can safely use the SAPUI5 **navTo** API. In that case, the **routerClass** definition is not needed (default SAPUI5 router is used instead).

Note

Existing widgets that don't use a `manifest.json` will work with the default router class. Widgets that use the Portal router, the inner routing navigation will not be reflected in the URL.

Intent-Based Navigation in a Nutshell

Developers can implement the intent-based navigation concept in their code.

An intent is a unique combination of a semantic object and an action that is used to define navigation to an application. The intent-based navigation mechanism in Fiori launchpad allows users to launch applications in different views or modes depending on the runtime parameters. At runtime, the application navigation targets are resolved into actual URLs.

Element	Description
Semantic object	Represents a business entity, such as Customer, Sales Order, or Product. Enables you to refer to such entities in an abstract implementation-independent way.
Action	Defines an operation, such as Display or Approve. This operation is intended to be performed on a semantic object.
Parameters	Optional. Parameters that define an instance of the semantic object, for example, employee ID.

Intents have the following syntax: `#<semantic object>-<action>?<semantic object parameter>=<value1>`.

For example, the intent `#SalesOrder-displayFactSheet?SalesOrder=27` specifies a fact sheet for sales order number 27. At runtime, this intent is resolved into the actual URL: `https://<server>:<port>/sap/hana/uis/clients/ushell-app/shells/fiori/FioriLaunchpad.html#SalesOrder-displayFactSheet?SalesOrder=27`

API Documentation

Developers can use the following APIs provided by the SAP Cloud Platform Portal.

APIs available on SAP API Business Hub

To access the Portal's catalog on the SAP API Business Hub, see [Portal Catalog](#).

JavaScript APIs

API Name	Description	JSDOC Link
Asset Service	A service for managing assets, such as documents and images. For detailed instructions for how to use this API, see Asset Service API .	Asset Service API
Site Service	A service that provides methods for custom development related to site menu, site and page settings, app navigation, and site status. For more information, see Site Service API	Site Service API
URL Helper	A service for dynamically constructing platform dependent URLs. For more information, see URL Helper API	URL Helper API
Widget Settings	This API is used for editing the settings of an app that was converted to a portal service widget. For more information, see Widget Settings API	N/A

Asset Service API

The Asset Service API is used for managing assets such as documents and images.

For information about the Javascript implementation, see [Asset Service API](#).

How to Upload an HTML to the Asset Repository

To upload an HTML snippet (`test.html`) to the Asset Repository, save the HTML in a variable named `contentToSave`. Save the file name in a variable named `fileName`.

≡Sample Code

```
var assetService = sap.ushell.Container.getService('AssetService'),
    contentToSave = "<p> Insert your text here <img src='' alt='' /> </p>",

    fileName = "test.html",

    metadata = [];
```

Now call the `updateAsset` method to upload the content. This method creates a new asset in the Asset Repository, if does not exist, or overwrites the asset with the new content, if it already exists.

≡,Sample Code

```
assetService.updateAsset(contentToSave, fileName, siteId)
    .done(function (response) {
        /*test.html is saved in the asset with the contentToSave. The response contains the
    });
```

How to Edit Asset's Metadata

You can also provide metadata to the created asset. Create, if does not exist, or update a file with content and metadata. The metadata enables you to attach tags to the assets. For more details, see [Metadata in the Document Service](#)

≡,Sample Code

```
assetService.updateAssetAndMetadata(contentToSave, metadata, fileName, siteId)
    .done(function(response) {
        /*the response holds the URL to this asset or null otherwise*/
    }) .fail(function() {
        //do some error handling
    })
    .always(function() {
        //some code
    });
;
```

How to Retrieve Content from the Asset Repository

If you already have a file name and you want to retrieve its content from the Asset Repository, use the `getAsset` method.

≡,Sample Code

```
assetService.getAsset(fileName, siteId)
    .done(function (response) {
        /*response === contentToSave*/
    }).fail(function() {
        //do some error handling
    })
    .always(function() {
        //some code
    });
;
```

How to Get the Asset's URL

To get the asset's URL from the Asset Repository, use the `getAssetUrl` method.

≡,Sample Code

```

assetService.getAssetUrl(fileName, siteId)
    .done(function (response) {
        if (response) { //if asset exist
            /*the response hold the URL to this asset*/
        } else {
            /**/
        }
    }).fail(function() {
        //do some error handling
    })
    .always(function() {
        //some code
    });
;

```

How to Generate a URL for an Asset

You can generate a URL for an asset in the Asset Repository without first checking that it exists. This will work faster than when using the `getAssetUrl` method as it does not send a request to the server. However, you should bear in mind that in this case, the asset might not exist at all.

Sample Code

```

7.      assetService.assetUrl(fileName, siteId)
        .done(function (response) {
            /*the response hold the URL to this asset*/
        }).fail(function() {
            //do some error handling
        })
        .always(function() {
            //some code
        });
;

```

How to Verify if an Asset Exists

If you want to verify first if a certain asset exists in the Asset Repository before rendering it, use the `isAsset` method first.

Sample Code

```

assetService.isAsset(fileName, siteId)
    .done(function (isAsset) {
        /*return true if asset, false otherwise*/
    }).fail(function() {
        //do some error handling
    })
    .always(function() {
        //some code
    });
;

```

How to Delete an Asset

To delete an asset from the Asset Repository, use the `deleteAsset` method.

≡Sample Code

```
assetService.deleteAsset(fileName, siteId)
    .done(function (isDeleted) {
        /*return true if succeeded in deletion, false otherwise*/
    }).fail(function() {
        //do some error handling
    })
    .always(function() {
        //some code
    });
;
```

How to Upload Binary Content (For Example, Image)

You can upload binary content such as an image to the Asset Repository, as follows:

≡Sample Code

```
var fileUpload = document.getElementById("fileUpload");
var file = fileUpload.files[0];
var siteId = assetService.siteService.getSiteID();

assetService.updateAssetAndMetadata(file, [], file.name, siteId)
    .done(function(fileUrl) {
        //some code
    })
    .fail(function() {
        //do some error handling
    })
    .always(function() {
        //some code
    });
```

How to Retrieve All Assets of a Site

Retrieve all assets that are tagged for a specific site ID.

≡Sample Code

```
assetService.getAllSiteAssets(sSiteId)
    .done(function(assets) {
        //some code
    })
    .fail(function() {
        //do some error handling
    })
    .always(function() {
```

```

    //some code
  });

```

How to Use the Asset Service in Public Pages

To use the Asset Service API in widgets of public pages, first declare the resource IDs in the widget settings section, as follows:

```

"sap.cloud.portal": {
  "settings": {
    "keyName": {
      "displayName": "My prop 1",
      "value": "starting value"
    },
    "resources": { //you must declare the resource ID when used in a public page
  "assets": [
    {
      "value": "resourceId"
    }
  ]
}
}
}

```

iNote

The Asset Service API is not supported in full-page applications that are used in public pages.

Site Service API

The Site Service API provides different methods related to site settings, site status, site menu, and subscription to navigation events. Developers can use these methods to create their own implementations for their sites.

For information about the Javascript implementation, see [Site Service API](#).

Here are examples of how to implement some of the methods:

- **saveSiteSettings:** Saved in the context of the site. It doesn't matter in which page the code is written.

Sample Code

```

var siteService = sap.ushell.Container.getService("SiteService");
siteService.saveSiteSettings({
  prop1: "val1",
  prop2: "val2"
});

```

- **savePageSettings:** Saved in the context of the page. Saves the setting of the current page.

Sample Code

```

var siteService = sap.ushell.Container.getService("SiteService");
siteService.savePageSettings ({
    prop1: "val1",
    prop2: "val2"
});

```

- `subscribeOnAppNavigation/unsubscribeOnAppNavigation`:

≡ Sample Code

```

function onNavChanged(){
//Place your code here
}

var siteService = sap.ushell.Container.getService('SiteService');

//subscribe
siteService.subscribeOnAppNavigation(onNavChanged);

//unsubscribe
siteService.unsubscribeOnAppNavigation(onNavChanged);

```

URL Helper API

For information about the Javascript implementation, see [URL Helper API](#).

Fiori application development may target multiple platforms, such as ABAP and SAP Cloud Platform as well as multiple subaccounts in a cloud environment. The Unified Shell services mechanism allows to develop applications in a platform-independent manner. It is achieved by exposing the same services API with potentially different implementation per platform.

When the application is running on different platforms and systems, the location of static resources and paths to data sources may differ. Even when running on the same platform, such as SAP Cloud Platform, the application may be consumed by different subaccounts, leading to different application URLs according to subscription configuration. As a result, it is strongly recommended to avoid using hardcoded platform dependent URLs in the application code.

A recommended practice is to use dynamic construction of platform dependent URLs. There are several use cases when URL should be constructed:

- When an application needs to access a static resource that is part of that application, and the relative path to that resource is known in advance. Another option is that the application needs to point to its own defined route. Relative URLs in JavaScript are relative to the HTML page, and when applications are loaded from different location, using relative URLs might cause unexpected results.
- When an application needs to access named data source, and the path to it is dynamic and may be calculated in a platform-dependent way.

Fiori technology promotes a declarative way of defining application dependencies, including static resources and data sources. Application configuration is defined by developers in the `manifest.json` file. Providing a `manifest.json` for Web applications is an open Internet standard adopted and extended by SAP.

Loading Static Resources from Locations Relative to the Component.js File

There are several options to load static resources from relative locations:

- For JavaScript UI5 resources (control, view, etc.) it is recommended to load it using UI5 `sap.ui.require(["./path/to/resource"], fnCallback)` mechanism. Note the `"./"` prefix. Alternatively, it is possible to register custom resource root in the `manifest.json` under `"sap.ui5"` section. For example:

Sample Code

```
"sap.ui5": {
  "resourceRoots": {
    "path": "./path"
  }
}
```

Once it is done, it is possible to use `sap.ui.require(["path/to/resource"], fnCallback)` without a prefix.

- For CSS and non-SAPUI5 scripts, it is possible to load them by referring to them in the `manifest.json` file.

Sample Code

```
"sap.ui5": {
  "resources": {
    "css": [{"uri": "css/styles.css"}]
  }
}
```

Resources defined with relative URI always have a URI resolved relatively to the `Component.js` file (`"./"` prefix is not required). SAPUI5 framework includes listed resources automatically, no additional scripting is required.

- If the resource cannot be included automatically (for example images, PDFs), you can generate platform and subaccount dependent URLs using the Unified Shell URLHelper service. (The following example assumes that the code is running inside `Component.js` file).

Sample Code

```
sComponentId = this.getId();
sPath = "/img/logo.jpg";
sURI = sap.ushell.Container.getService("URLHelper").createComponentURI(sComponentId, sPath);
```

The value of `sURI` in this case may be similar to

`"/sap/fiori/app_subscription_name/path/to/component/img/logo.jpg"`.

Related Information

[Access Remote Content from an Application](#)

Widget Settings API

The widget settings API is used for editing the settings of an app that was converted to a Cloud Portal widget.

Use the Default Settings Editor

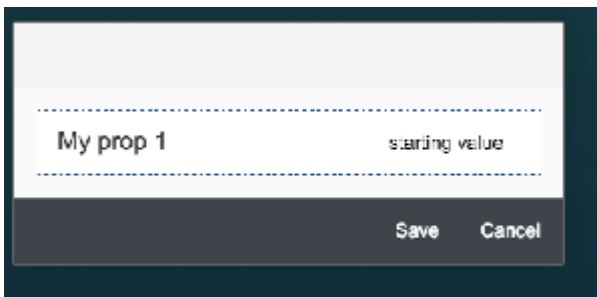
By default, an app that was converted to a Portal widget does not have a **Settings** button. To edit the widget settings, the developer has the following options.

1. Open the widget manifest for editing.
2. Add the following:

Sample Code

```
"sap.cloud.portal": {
  "settings": {
    "keyName": {
      "displayName": "My prop 1",
      "value": "starting value"
    },
    "resources": { //this section is optional, in case you choose to save your asset in
  "assets": [
    {
      "value": "resourceId"
    }
  ]
}
}
```

The result of this example is:



3. To listen to changes in widget properties submitted by the admin, the developer has to implement the `onConfigChange` method. See the following example:

Sample Code

```
onConfigChange : function(){
  Function readPortalSettings(){
    var manifestPortal = this.getMetadata().getManifest()["sap.cloud.portal"];
    if (!manifestPortal || !manifestPortal.settings || manifestPortal["sap.clo
  return;
  }

  Return manifestPortal.settings;
}

  Var settings = readPortalSettings();
}
```

Control the Widget Settings Using the Widget Settings API

1. Open the widget manifest for editing.

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

2. Declare that you are not going to use the default settings editor provided by SAP: `show.portal.settings : "false"`
3. Add the following code to the widget manifest:

≡ Sample Code

```
"sap.cloud.portal":{
  "show.portal.settings" : "false",
  "settings":{
    "keyName":{
      "displayName":"display name",
      "value":"key value"
    }
  }
}
```

4. To read the settings, use the `onConfigChange` method as specified above.
5. The following actions are possible through the API methods:
 - To save a setting, use `component.fireEvent("save.settings", {});`
 - To revert a setting, use `component.fireEvent("revert.settings");`
 - To catch a save setting error, use `component.attachEvent("save.settings.error", function () {});`
6. To use a customized UI, the developer has to subscribe to the following event: `component.attachEvent("open.dialog", function () {})`

Administrator Guide

Welcome to the SAP Cloud Platform Portal Administrator Guide.









As an administrator, you can build freestyle sites using templates either provided out-of-the-box by SAP or custom templates that your developers have created in SAP Web IDE.

There are so many options. Here are some examples of what you can do:

- Create sites from one of the available site templates and use them as is with the preconfigured pages, page templates, and built-in widgets.
- Base your site on one of the templates, edit its predefined content, and extend it with your own page templates, pages, and apps.
- Create a freestyle site with pages and apps and add a launchpad page. In this way, you can enjoy the benefits of SAP Fiori launchpad, and yet have HTML, images, and other types of content in your site.

iNote

The following diagram contains links to more information.


	What you should know
	Typical end to end scenario
	Building a site
	Page management
	Content management
	Site management
	Web content management
	Integration with SAP Fiori launchpad

What You Should Know


Important facts that administrators should know before building sites in the Portal.

The following diagram links you to more information to help you learn more about some aspects of the portal before you start building your freestyle sites.


Hover over the shapes below for details, click them to get more information.




Your working environment




Building blocks for creating a site




Best practices for building sites



More about site templates



More about page templates



More about widgets

Your Working Environment

The SAP Cloud Platform Portal includes features and tools to help administrators build and manage sites.

What are the main areas in the portal?



Admin Space



Site Directory



Site Designer






Web Content Editor

Area	Description
Admin Space	Where you manage your entire subaccount - sites, themes, analytics, and assets.
Site Directory	Where you manage your sites - create, duplicate, import, export, delete, publish.
Site Designer	Where you create sites, define the layout, and add pages and content.
Web Content Editor	Where you or a web content editor can edit the web content in your site.






How do I navigate in the portal?

The menu navigation panel on the left, helps you navigate to the following areas:

Menu Option	Description
Home 	This is the Admin Space - your entry point to the Portal. From here you can access the Site Directory where all your sites are stored. You can also receive product updates, join the community, access online help, and reference the code samples of site and page templates, applications, widgets, and more.

Menu Option	Description
Site Directory 	Access the Site Directory from the side navigation panel in the Admin Space. From here you can manage your existing sites or create new ones.
Services and Tools 	This is where you can access the Theme Manager service to manage your themes for all sites in your SAP Cloud Platform subaccount.

When you open a site for editing in the [Site Directory](#), you reach the Site Designer. The menu panel on the left, helps you navigate to the following areas:






Menu Option	Description
Page Management 	<ul style="list-style-type: none"> • Pages Create and add pages to your site and add content to them. • Page Templates Use the available page templates in your site to create pages or edit them with your own content.
Content Management 	<ul style="list-style-type: none"> • Apps Create and configure apps in the Manage App Configuration editor. • Catalogs Configure catalogs to enable role-based access to apps in the Manage Catalogs editor. • Roles Create and configure roles to manage access for launchpad users.
Menu Editor 	Configure the site menu by adding apps, pages, links, and titles and define their hierarchy in the menu.
Settings 	Change site version, define end user capabilities such as language, themes and search, view site status and publishing information, and set your home pages.
Services and Tools 	Create and manage your site themes, translations, and site's assets (documents, images, videos). Enable another user assigned to the Web Content Editor role to access the Web Content Editor tool.

Building Blocks of a Freestyle Site

Portal sites are based on templates and portal components.

How do I build a freestyle site?

Use the following building blocks to build your site:

	Object	Description
	Site Template	<p>Freestyle sites are based on templates. Developers can take any of the existing templates provided by SAP or by a different provider, and customize them to create their own templates.</p> <p>For more information, see About Site Templates.</p>
	Site	<p>After the developer deploys the site templates to an SAP Cloud Platform subaccount, they become available for the admin, who then creates site instances from the templates.</p> <p>For more information, see Building a Site.</p>
	Page Template	<p>A page template defines the layout of the page, and optionally can also define initial content. Page templates are useful when a specific layout or content is repeated in several pages of the site.</p> <p>For more information, see About Page Templates.</p>
	Page	<p>Developers or administrators can then create pages based on the page templates and populate them with content.</p> <p>For more information, see Adding Pages to a Site.</p>
	Widget	<p>Widgets are the basic unit of content in a freestyle site. They are web-based apps that are added to page sections.</p> <p>They can be any out-of-the-box widgets provided by SAP or any SAPUI5 app developed by your developers or partners.</p> <p>For more information, see Adding Widgets to Pages.</p>

About Site Templates

SAP Cloud Platform Portal is delivered with a set of out-of-the-box site templates that enables administrators to easily build freestyle sites.

Portal freestyle sites are based on site templates. These templates act as a framework for you to create an intuitive and user-friendly site for your end users. By already having a starting point and not having to create a site from scratch, you not only save time, but your design experience is much easier.

Where are site templates created?

A site template is the starting point of the developer flow. Developers create site templates in SAP Web IDE based on available templates. Once they deploy the template project to your SAP Cloud Platform subaccount, you can then create site instances based on these site templates.

What can I do if the template is not exactly what I need for my business scenario?

Don't worry – the templates can very easily be customized. You can change the content on the site pages, add sections, and change the layout and settings according to your users needs.

iNote

Some changes to the site layout and content may require changes to the code by the template developer.

How do I create a site using a site template?

You create sites in the Site Directory or directly from the Admin Space when you log on to the Portal. Simply click **+** and then follow the steps of the wizard. In the **Template Source** dropdown list, you have the following options:

- **My Subaccount** – site templates that a developer created for you and deployed to your SAP Cloud Platform subaccount or those templates that you are subscribed to.
- **Samples** – site templates for creating a site for a specific solution (for example, a support site or human resources site).
- **SAP** – out-of-the-box templates provided by SAP. These include:

Site Template	Description
Starter Site	A very basic site template that acts as a starting point for you to add your own pages and content.
Basic Layout Set	A more comprehensive template containing four pages that are based on the Anchor Navigation , Header and Footer , Starter Page , and Tall Header page templates.
Admin Site	<p>Used to create a dedicated site for administrators. The template contains pages with predefined content that serves as a solid foundation for your new admin site.</p> <p>iNote</p> <p>The Admin Site template includes features such as:</p> <ul style="list-style-type: none"> ◦ Top and bottom menus allowing you to move menu items (pages and apps) between the two menus. ◦ Ability to add icons to the menu items. ◦ Ability to add action buttons (such as Preview and Publish) to the tool bar of your admin site by creating a shell plug-in app. <p>For more information, see Creating and Configuring Shell Plugin Apps.</p>
SAP Fiori Launchpad	A template for creating an SAP Fiori Launchpad site.
Marketing Site	A template that contains default marketing-related content. It is built with web content widgets that enables you to edit the content in the Web Content Editor tool.

About Page Templates

Administrators can use the page templates already created by developers or they can edit the content of existing page templates in a site.

What is a page template?

A page template defines the layout of a page and usually includes predefined content. They can be very handy when a specific layout or content is repeated on several pages, (for example a company logo or contact information that you want to see on each page of your site).

When you create a site based on a certain site template, you inherit any page templates that were created by the developer for that site template. When you build your site, you can choose whether or not to use these page templates as is, or edit the content and add your own.

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

For more information about how to change the content of a page template, see [Changing the Content of a Page Template](#).

About Widgets

Administrators add content to their freestyle sites by selecting out-of-the-box and customized content from a rich gallery of widgets in the Site Designer.

What is a Portal widget?

A widget is the basic unit of content in a Portal freestyle site. It's a simple, web-based application that you can embed in page sections of a site.

Portal widgets are provided out-of the-box by SAP or you can ask your developers to convert any SAPUI5 application into a widget (using SAP Web IDE). The widgets are available in a widget gallery in the Site Designer. You can access them directly from the page section in which you want to insert the widget.

How do I add a widget to my page?

In any page section in your site, click **+** to open the **Add Content** dialog box that displays the widget gallery.

For more information, see [Adding Widgets to Pages](#).

About the widget gallery

The widget gallery includes all SAP's out-of-the-box widgets as well as customized widgets created by your developers. These widgets are organized into two groups:

Group	Description
Web Content Widgets	<p>Web content widgets are editable in a dedicated authoring tool called the Web Content Editor; either by you, the site administrator, or by an authorized web content editor.</p> <p>In the widget gallery, they are organized into the following categories:</p> <ul style="list-style-type: none">• The Basic category that includes widgets that consist of a single content item (such as an Image, Video, or Link).• The List, Cards, and Carousel categories include multiple content items that come with handy content snippets (small chunks of predefined content) that you can use as a starting point for adding content to your page.• Content snippets in the Menu category enable you to easily add predefined menus anywhere in a site page. You can link the menu items to any sections on the page or to other pages, apps, and external links. <p>iNote</p> <p>Any link in a web content widget, can navigate to a page, app, external link, or to a page section. Page sections can be renamed, making it easier to identify where the link is navigating to.</p> <p>For more information about web content widgets, see About Web Content.</p>

Group	Description
Standard Widgets	<p>Standard widgets are only editable in the Site Designer. Therefore as administrator, it is your task to edit this content.</p> <p>They are organized into the following categories:</p> <ul style="list-style-type: none"> • Advanced - widgets with advanced settings that you can change. For example, if you select an Image widget from this category, you can click the ⚙️ (settings) icon and determine the image state, height, and alignment for both desktop and mobile devices. <p>Some other examples of Advanced widgets: HTML, Image, Rich Text Editor, List Builder, Tile Grid, Standard Carousel, Login widget.</p> <p>iNote</p> <p>The Login Widget is very useful for enabling your end users to log in and log out of a public page that doesn't have a SAP Fiori header.</p> <ul style="list-style-type: none"> • Social - these include: <ul style="list-style-type: none"> ◦ Widgets for Facebook, Twitter, and other social networks. ◦ Widgets that you can add for those users who want to connect to the SAP Jam social media platform to collaborate with their colleagues. <p>iRemember</p> <p>To use SAP Jam, you first need to set up integration between the Portal and SAP Jam. For more information, see Collaborating Using SAP Jam.</p> <ul style="list-style-type: none"> • Custom - widgets created by your developers and added to the widget gallery.

iNote

You can build your site pages using a combination of both kinds of widgets.

Best Practices for Improving Performance of a Freestyle Site

Administrators can use these best practices and guidelines to effectively design an easy-to-use, fast loading business site.

Let's face it – if your web site performs well, people will visit your site and use it.

In this topic, we will highlight ways in which you can improve the page speed and load time of your Portal freestyle site.


Why would there be performance issues?

Every file that is included in your site pages (such as CSS, JavaScript, and especially images), needs to be downloaded from the server to the browser. By minimizing the number of round trips that the browser makes to the server, you can substantially improve the performance of your site.

How can I check the size of my site?

In the [Assets Repository Dashboard](#), a sizing tool indicates the total number and size of assets in the site.

To get to the [Assets Repository Dashboard](#), do the following:



1. Click  (Services and Tools).
2. In the **Asset Management** tile, click **Configure**.

You can see the sizing tool above the table of assets together with a link to this document.

How can I optimize the performance of my site?

When you load a site, most of the load time is spent downloading the pages and assets such as images, videos, and HTML files. So, the more assets you have, the longer the site takes to load.

Use the following tips and recommendations to speed up the performance of your site pages significantly and enable your users to enjoy a satisfying browsing experience.

Subject	Recommendations and Best Practices
Streamline content	<ul style="list-style-type: none"> • Widgets: Streamline the number of widgets per page (approximately 5 per page). • Assets: Keep the number of assets to a minimum - assets include the number of images, videos, and HTML files that you add to each page of your site. We recommend that the maximum size of an asset should be 120 KB. • Pages: Limit the number of pages. If you've created a site with pages that aren't being used in the runtime site, consider removing them. • Site: Site size is the sum of all the assets in the site. We recommend that the total size of assets in your site should be less than 45 MB.
Optimize images	<p>Image optimization should be your number one focus. Large images are the main contributors to slowing down your site. The following image optimization best practices will help to reduce the negative impact of images on the performance of your site:</p> <ul style="list-style-type: none"> • Keep the number of images per page to a minimum. • Compress image files to the minimum possible size while still maintaining high quality. <p> Remember</p> <p>Not more than 120 KB</p> <ul style="list-style-type: none"> • If you're using an image as your background, consider keeping it a reasonable size.
Eliminate unnecessary image resources	Delete assets that are no longer being used from the Asset Repository.
Home page content	<p>Avoid putting too many large files on the opening page of your site.</p> <p>This is the first page that loads, and heavy files slow down the loading of your site each time.</p>
OData calls	<p>If your application or widget requires OData calls to retrieve back-end data, make sure that these calls are not blocking the user interface.</p> <p>We recommend that you start loading the UI without back-end values and gradually render the data after it's retrieved.</p> <p>Here is a nice example: SAP Support Site. </p>

Subject	Recommendations and Best Practices
Translation	If widgets and apps in your site need to be translated, define all languages that end users may use. If a language is not defined, the user receives an error message. These messages slow down the loading time of the site page.
Use SAPUI5 Library Preload Mechanism (for developers)	<p>To optimize the loading of resources, use the SAPUI5 dependencies mechanism in your site's manifest file as shown in the following code sample.</p> <p>iTip</p> <p>We highly recommend that you review the list of dependencies and make sure that there are no redundant libraries. Each redundant library affects the overall loading time.</p> <p>Sample Code</p> <pre>"sap.ui5": { "rootView": { "viewName": "testapplication.testapplication.view.View1", "type": "XML" }, "dependencies": { "minUI5Version": "1.60.1", "libs": { "sap.ui.layout": {}, "sap.ui.core": {}, "sap.m": {} } } }</pre> <p>For more information, see Descriptor for Applications, Components, and Libraries. (Table 4: Attributes in the sapui5 namespace).</p>

Basic Workflow of an Administrator

An example of how a site administrator would typically manage and monitor a freestyle site in the Portal Site Designer tool.

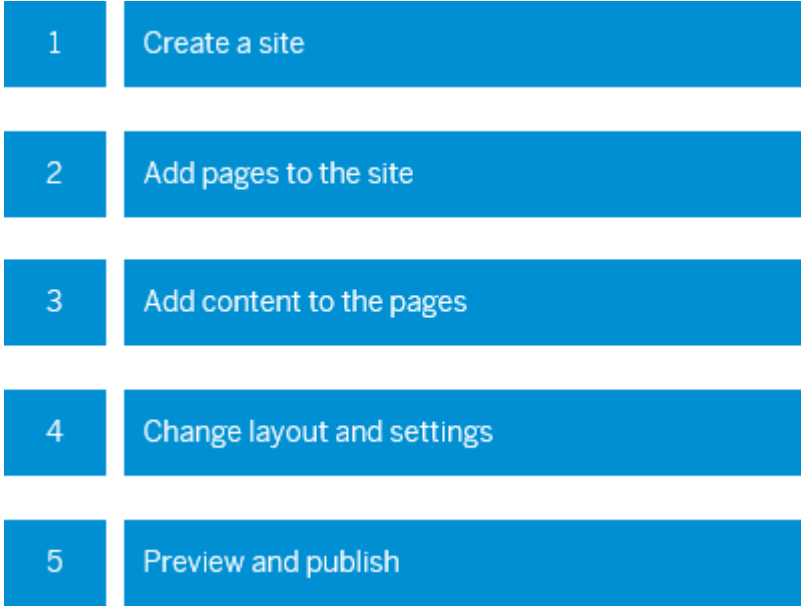
As a site administrator of a company, one of your responsibilities is to manage their portal sites. You have been asked to create a dedicated site for the sales team, enabling them to carry out their tasks efficiently and easily.

To achieve this, you will log on to SAP Cloud Platform Portal and create a site called **Sales Division**. You will then add pages and apps to this site, and decide on the site's layout and other settings. Once you have set it all up, you can then add content or assign another user to create the content for the site. When all is in place you can publish the site so that the sales staff can use the site and carry out their tasks.

Now let's get started!

iNote

The diagram below shows the high-level steps of this workflow – click each step to display more details.



Create a site	Add pages to your site	Add content to the page	Change the layout and settings	Preview and publish
<p>1. Log on to the Portal and click  to open the Site Directory.</p> <p>2. Click + to add a new site. A wizard guides you through the steps to create your new site.</p> <p>iNote</p> <p>You can also create a new site directly from the Admin Space when you log on.</p> <p>3. Name the site Sales Division and select SAP as the Template Source.</p> <p>4. From the available SAP templates, let's select the Basic Layout Set template.</p> <p>5. Click Create. Your new site opens in the Site Designer displaying three predefined pages. It's now up to you. You can edit the content, layout, and settings of these pages or can add your own pages to this site. Let's move on to the next step to see how to do this.</p>	<p>In this step, we will add a new page called My Home Page.</p> <p>1. Click + at the bottom of the Pages menu to add a new page.</p> <p>2. Name it My Home Page and select This Site as your Template Source.</p> <p>3. Let's select the Header and Footer page</p> <p>4. Click Finish to view your new page</p>	<p>Now let's add content.</p> <p>The first thing that you want to do is add a logo to the footer of this template so that it appears on each page based on this template:</p> <p>1. Click  to open the Page Templates list used in this site.</p> <p>2. Select the Header and Footer page template and in the footer section, click + in the section to open the Add Content dialog box. This dialog box includes a gallery of widgets that you can choose from. Add your logo image from the Custom category under Standard Widgets.</p> <p>3. In the Basic category, select the Image widget. A default image is loaded.</p> <p>4. Click the default image and then click  to edit the image settings.</p> <p>5. Replace the image with your logo.</p> <p>Now add content to the home page that you created:</p> <p>1. Click  to go back to the Pages menu and open My Home Page.</p>	<p>In this step, we want to do the following:</p> <ul style="list-style-type: none">• Change the layout of the main section of our home page to accommodate the four widgets we added in the previous step• Change the layout of the Testimonial widget that we selected from the List category. We want to display this widget according to the Grid layout.• Since we added an SAP Jam Content widget to our new home page, we also want to enable SAP Jam integration so that the sales team can communicate with one another. <p>1. Customize the layout of your page as follows:</p>	<p>Click  to view your changes to the site.</p> <p>If you're happy with the results, then click  (publish).</p> <p>Your site is now available to your end users!</p>

in the **Pages** menu of the site.

5. Click ▼ and select **Add to Menu** to make this page visible to your site's users.

6. From the same menu, **Set as home page**.

2. In each section of the page click + to open the **Add Content** dialog box displaying the widget gallery.


iNote

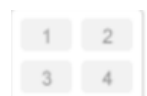
The widgets are divided into two categories:

- o **Web Content Widgets**, which are editable in a dedicated tool called the Web Content Editor (either by you, or a user that has been assigned the web content editor role).
- o **Standard Widgets**, which you edit directly in the Site Designer.


3. From the widget gallery add content to the pages as follows:

- o In the header section of the page, click + and from the **Basic** category, add an **Image** widget.
- o In the middle area, add the following widgets:
 - From the **List** category, add a **Testimonial** widget.
 - From the **Social** category, add the **Social Networks** widget.
 - From the **Advanced** category, add an **Image Carousel** widget.

- a. Open My Home Page.
- b. Click the middle section and then click  to edit section settings.
- c. Select **Row** as your **Layout** type.
- d. In the **No. of widgets in a row** field, select **2**.
- e. **Save** your settings and see how the four widgets that you added to the middle section of the page, become aligned as follows:



2. Change the layout of the Testimonial widget that you added as follows:

- a. Click  (settings) and select the **Grid** layout.
- b. Change the height and margins of each content


- From the **Social** category, add the **SAP Jam Content** widget.


Now to make sure that the widgets are displayed properly on the page, we are going to customize the layout and settings of the pages in your site. Let's move on to the next step to see how to do this.

item in the testimonial.

- c. Determine how many content items you want in a row. For the desktop, select 2 as the value and note how the preview on the right changes to reflect your input.

3. To enable SAP Jam integration:

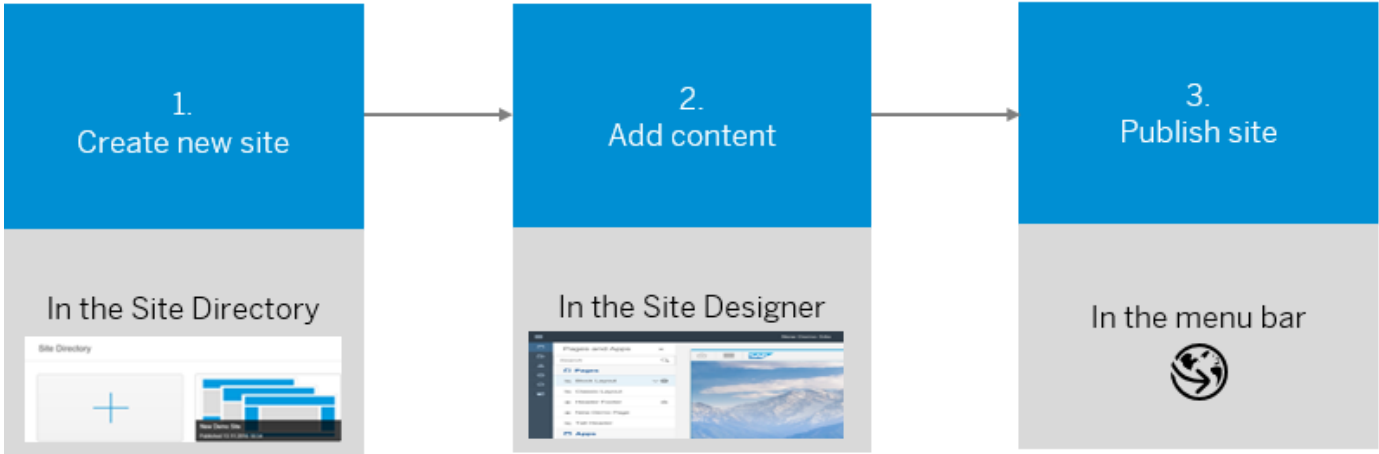
- a. Click  (site settings) in the side panel.
- b. Under **User Settings** enable **SAP Jam Integration**.

Use these tutorials to help you build a Portal freestyle site from scratch: [Get Started with SAP Cloud Platform Portal](#). 

Building a Site

In SAP Cloud Platform Portal, administrators can build intuitive and user-friendly portal freestyle sites in three easy steps.

Build a Site: 1 - 2 - 3



How do we do these steps?

:

Step	Description	More Information
<div>1. Create new site</div>	Create a site	<p>Create a new site from the Site Directory or from a quick link in the Admin Space. When you create a new site, it is added to the Site Directory.</p> <ol style="list-style-type: none">1. From the Site Directory, click + (create site) to open a wizard.2. Base your site on one of SAP's out of the box templates or a template that your developer has created for your site. <p>For more information about site templates, see About Site Templates.</p> <p>iNote</p> <p>You don't always have to create a site from scratch. You can also:</p> <ul style="list-style-type: none">• Duplicate a site. <p>In the Site Directory, hover over your site, click ▼ to open a list of site actions, and select Duplicate. In this case, don't forget to change the site alias. For more information, see Configure Site Settings.</p> <ul style="list-style-type: none">• Build a dedicated site for administrators by using the Admin Site template provided by SAP. For more information, see About Site Templates.

Step	Description	More Information
2 Add content	Add content	<p>The next step is to add content to your site. First add pages and apps to your site and then add content (widgets) to the page sections.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • Adding Pages to a Site • Add Apps to a Site • Adding Widgets to Pages <p>Once your content is in place, you can then change the layout and settings of the widgets on your page to customize the look and feel of your site.</p> <p>For more information, see:</p> <ul style="list-style-type: none"> • Determining Layout of Your Pages • Editing Widgets on Your Pages
3 Publish site	Publish site	<p>Publish your site in one of the following ways:</p> <ul style="list-style-type: none"> • In the Site Directory, hover over your site, click ▼ to open a list of site actions, and select Publish. • In the Site Designer, while editing your site, click 📄 (Publish) at the top right of the screen. <p>iNote</p> <p>When you publish your site, there are two options:</p> <ul style="list-style-type: none"> ◦ Click Publish and Open to publish the site and open it in a new tab. ◦ Click Publish to publish your site without opening it so that you can continue with your work in the Site Designer. <p>Once a site is published, you and your end users can view it online. To access your site, hover over its tile in the Site Directory, and click the URL.</p>

Page Management

In this section, administrators can learn about how to add pages to their freestyle sites and then add content to the sections of each page.

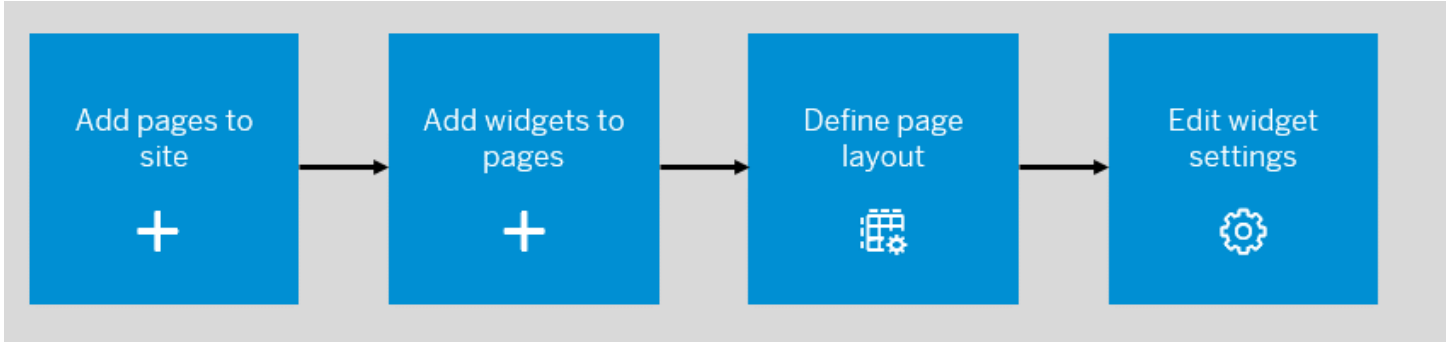
Typical workflow for adding pages and content to your site

You can add as many pages to your site as you want - but don't overdo it as it can affect the performance of your site. In the Portal, each page that you add to your freestyle site is based on a page template. The page is divided up into sections according to the page template that you based it on.

Your next steps would be to:

- Add content to the page sections in the form of widgets.
- Define how you want them arranged in the page sections.
- Change the page layout by determining the respective settings for the widgets and the sections.

This diagram shows the typical workflow of adding pages and content to your site. Click each section for more information.



Adding Pages to a Site

Administrators can add new pages to a freestyle site under **Page Management** in the Site Designer.

Add a new page made up of a number of sections, or add a page that consists entirely of one app. New pages are only visible to end users once you add them to the site menu.

For more information, see [Create a Site Menu Structure](#).

How do I add pages to my site?

Option	How to do it?
Add a new page	Under Page Management , select Pages and then click + (add) at the bottom of the pages menu.
Duplicate an existing page	Click ✓ next to an existing page name and select Duplicate .
Add an app as a full page	For more information, see Add Apps to a Site .

iTip

If your pages contain repeatable content, such as a logo or contact information, consider adding that content to a page template and then create pages based on the edited page template.

For more information, see [Changing the Content of a Page Template](#)

iNote

To add apps to your portal freestyle site, you must first add a launchpad page to your site.

When adding a new page using the wizard, select **SAP Fiori Launchpad** as your page template.


Adding Widgets to Pages

Administrators can easily add content (widgets) to the sections of a Portal page.


What you need to know before you start adding content

You add content to pages by selecting widgets from the widget gallery in the Site Designer.


There are two options:

- You can add one or more out-of-the-box widgets to each empty section of your page.
- If your section already contains predefined content (inherited from the page template), you can replace this content with your own. To do this, you first have to disconnect the section from the page template. This is how you do it:
 1. Click the section with the predefined content.
 2. Click  (disconnect) to disconnect the section from the page template.
 3. Add your new content.

iNote

If you're not happy with the changes and you want to revert back to the page template content, click  (reconnect) to remove all your changes.

Now let's add the widgets

1. Click the page section.
2. Click  to open the **Add Widget** dialog box that contains the widget gallery.
3. Select one of the **Web Content Widgets** or **Standard Widgets**.

For more information about the different types of widgets, see [About Widgets](#).

iNote

If you have added multiple widgets to a page section, you will probably want to change the layout of the section to arrange the widgets in the page sections. For more information, see [Determining Layout of Your Pages](#).

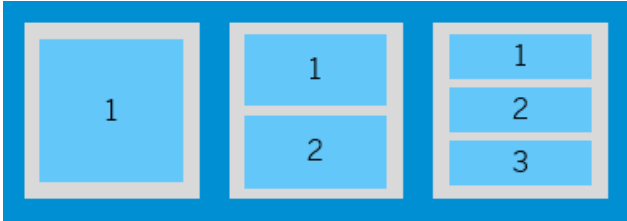
Determining Layout of Your Pages

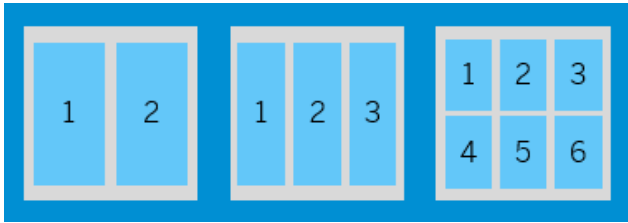
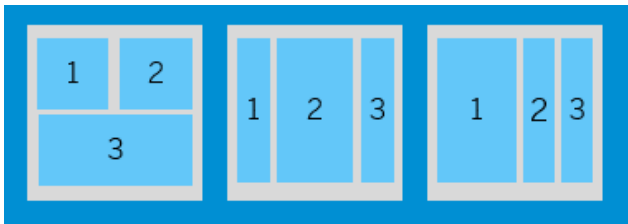
To organize widgets on a page, administrators can define the page section settings.

You've added widgets to your page and now you want to determine how they appear on your page. You can do this by selecting a section layout and defining widget dimensions according to your device (desktop, tablet, or mobile).


First select a layout

You can choose one of the following layouts:

Layout option	How is it displayed?	More information
Column		Widgets are aligned one under the other on any device (mobile, tablet, desktop).

Layout option	How is it displayed?	More information
Row		<p>Widgets are aligned next to each other and then wrap to the next line depending on the value you defined for the No. of widgets in a row.</p> <p>For example: If you have specified that the number of widgets in a row is 3, and you have added 4 widgets, then the fourth widget is wrapped and moves down and starts a new row.</p>
Custom		<p>Widgets are displayed according to how you define them according to your device. In other words for each device you specify how much space each widget takes up on the screen.</p> <p>To see an example of a custom layout, see Example of a Custom Layout.</p>

Then edit the page sections?

Click  (edit section settings) and define the following settings:

Tab	More information
Layout	<p>Select a section layout (column, row, or custom) depending on how many widgets you want to add to a page section and according to your device.</p> <p>For more information about the different layouts, see the section below.</p> <p>At the bottom of the screen, you can see a list of widgets that you have added to the selected section. Reorder them according to how you want them displayed in the section by dragging and dropping them in the table.</p>
Background	<p>This is the background of the entire section. You can add a background color or browse for an image and upload it as your background.</p> <p>If you choose to upload an image, click Show Image and then define the following:</p> <ul style="list-style-type: none">• Image Source: Your image can be uploaded from your computer, from the Asset Repository, or you can provide a URL to an image file.• Image State: Reflects the behavior of the image when the image is not the same size as the section.
Dimensions	<p>Here you can determine:</p> <ul style="list-style-type: none">• Margins: Space between and surrounding the widgets.• Content Width: Amount of space the widgets use in a section.• Minimum Height: Value that determines the minimum height of a section. The height of the section can expand if necessary but cannot be made smaller. For example, if you add an image and it takes up more than the minimum height, the section can expand to incorporate the whole image. If you add content that is less than the minimum height, the height of the section remains at the minimum height.




Example of a Custom Layout

Administrators can define their own customized layout of how widgets are set out in a page section. This topic shows an example of a custom layout and shows how the widgets are displayed at runtime on various devices.




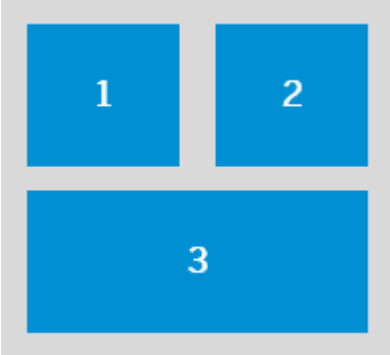

How do I define a custom layout for a page section?

When you select a **Custom** layout, a table opens with a dropdown list of dimensions. Select the dimensions according to how much space you want each widget to take up across the screen.

Let's look at the example in the table below, which shows us adding three widgets to a page section:

Widget	Desktop	Tablet	Mobile
			
1	1/3	1/2	All
2	1/3	1/2	All
3	1/3	All	All

From the configuration of the page section in the example above, the widgets would be displayed on the screen as follows:

Desktop 	 <p>In a desktop device, we have defined that each widget takes up one third of the section space.</p>
Tablet 	 <p>In a tablet device, the first two widgets each take up half the width and the third widget moves down to the next row and takes up the entire width.</p>
Mobile 	



In a mobile device, each widget takes up the entire width of the section and will appear one under the other.

Editing Widgets on Your Pages

Administrators are responsible for maintaining sites in the Portal and ensuring that the overall design and content of the site is accurate and up to date.

Who edits the content?

As administrator, you are responsible for the creation and maintenance of the web site – the structure, the layout, and the overall design. The actual content in the site is either maintained and updated by yourself or you can enable a web content editor to edit the web content.

For more information, see [Access the Web Content Editor Tool](#)

Where is the content edited?

There are two places where you can edit content:

- Site Designer - this is where you maintain the site - you can add and edit content as well as determine the layout and structure of your site. You are able to edit both web content widgets as well as standard widgets.
- Web Content Editor tool - this is a dedicated tool where the web content editor edits web content.



iNote

Web content editors cannot access the Site Designer.

How do I edit the content?

To edit content, you simply click the widget. In the widget controls, you will see one of these icons:

Control	More information
---------	------------------

Control	More information
	<p>Use this control for editing content of widgets.</p> <p>iNote</p> <p>For web content widgets, clicking this control directly opens the web content editor tool where you can edit the content.</p> <p>For more information about editing web content widgets, see Edit Web Content.</p>
	<p>Use this control for defining layouts of widgets, dimensions of the widgets, and other advanced settings. Here are some examples of what settings you can define:</p> <ul style="list-style-type: none"> For web content widgets, select a layout according to how you want your widget displayed. <p>For example if your widget was based on a content snippet from the List category, you can select a Grid layout. Notice in the preview on the right that the layout of your widget changes.</p> <ul style="list-style-type: none"> Define advanced settings for standard and web content widgets. <p>For example, for a Social Networks widget, you can determine links to the various social networks. For a Carousel web content widget, you can define autoplay, pagination, arrows, and other effects.</p>

Using Anchor Navigation

To help end users quickly find what they are looking for, administrators can implement anchor navigation in their portal pages.

Why use it?

Anchor navigation links are handy when users want to quickly find the most relevant part of your site. Instead of having to scroll down a page or navigate through the pages of your site, with one click they can jump directly to the page, app, or section of a page that they want.

How do I use anchor navigation in the Portal?

You can create anchor navigation in your site by using menus or links.


Using Menus

In the Portal, there are two ways of using menus:

- You can base your site on the [Basic Layout Set](#) site template. In this case one of the pages that you inherit with this template is an [Anchor Navigation](#) page template with a predefined anchor navigation menu and 12 sections.
- If your site is not based on this template, you can simply add one of the web content widgets from the Site Designer widget gallery to any page in your site. Look under the [Menu](#) category for various content snippets, each with a different style of menu.

For both options, you simply configure the menu items to link to the relevant target (sections on the same page or to other pages, apps, and external URLs).

Using Links



You can configure any link that exists in one of the out-of-the-box widgets in the Site Designer widget gallery. You simply add a widget to your page and click  (Edit Content) and configure the link to navigate to sections on the same page or to other pages, apps, and external URLs.

iNote

All page sections can be renamed, making it easier to identify where you are linking to.

What is a typical workflow for creating anchor navigation?

Let's use the predefined site template:

1. Create a site based on the **Basics Layout Set** site template. Notice that the site is created with four pages, one of which is the **Anchor Navigation** page with three predefined menu items at the top of the page.
2. Now define your menu items as links like this:
 - a. Select the first menu item and then click  (Edit Content). In the **Edit Content** dialog box, the first link is already selected .
 - b. Enter a display name for the menu item.
 - c. In the **Link Type** field, select whether to link the menu item to a section on the same page, or to another page, app, or even an external link.
 - d. Select the name of the section, page, or app that you are linking to. If you're linking to an external link, enter the URL.
 - e. **Save**.
3. Do the same for each menu item.
4. Use **+** (Add) if you want to add more menu items.
5. Rename your sections. Simply click in the section and replace the default section name with a name that matches the menu item that's linking to it.
6. To customize your menu, you can define some of the following settings. Click a menu item and then click  (Settings):
 - o Select the **Pin to Top** feature to ensure that the menu remains visible as you scroll down the page.
 - o In the logo settings, choose to show the logo that you defined for the site theme and align it in the menu.
 - o Define the height dimensions of your menu.

iNote

As you define the settings, a preview of your menu displays on the right.

7. Publish and preview your site to check that the menu items link to the correct section, page, app, or URL.

Changing the Content of a Page Template

Administrators can use page templates in a site as is or they can change the content to suit their scenario.

How do I change the content of a page template?


1. In the Site Designer, click  to open **Page Management**.

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

2. Select **Page Templates**. Here you can see all the page templates that are available to use in your site.

iNote

Your developer may have created a new page template for this site.

To use this new page template to create page instances for your site, click  (Refresh) to add the new page template to the list.

3. Select a template.
4. Click within a page section and then click **+** to open the **Add Content** dialog box.
5. From here add widgets to your template.

For more information about adding widgets, see [Adding Widgets to Pages](#).

iRemember

The content that you add will appear in all pages that are based on this template.

iNote

Click  next to a page template to open its settings. Under **Page Template Content** you can see the list of widgets that exist in this template.

Content Management

Content Management in the Portal includes creating apps and managing them through the catalogs and roles that are assigned to them.

In this section, we explain the relationship between apps, catalogs, and roles and how to manage these objects in your freestyle portal site.

Where are apps managed in a freestyle site?

Apps are managed in the **Manage App Configuration** editor. From here you can browse for apps in your SAP Cloud Platform subaccount and add them to your site as a full page application. You can also add a launchpad page to your freestyle site and add as many apps to it as you want.

For more information about the different app types, see [About App Types](#).

For more information about how to add apps to your freestyle sites, see [Add Apps to a Site](#).

Where are catalogs managed?

Catalogs are managed in the **Manage Catalogs** editor.

For more information about catalogs, roles, and other authorization mechanisms in the portal, see:

- [About Apps, Roles, and Catalogs](#).
- [Define Access Levels](#).



Add Apps to a Site

Administrators can add apps to their freestyle sites.

There are two ways that you can add apps to your site:

- Add the app as a full paged app
- Add the app to an SAP Fiori launchpad page that you've added to your freestyle site

Let's first create the app

1. Open your freestyle site for editing.
2. Click  in the side navigation panel to open **Content Management** and select **Apps**. You are immediately directed to the **Manage App Configuration** editor.
3. You can either add a new app or you can browse for an app in your subaccount, select it, to copy its configuration into the editor.
 - To add a new app, click **+** at the bottom of your **Apps** menu.
 - Or you can browse for an app in your subaccount by clicking  (browse) in the **App Resource** field.
4. Click **Save**. For both cases, the new app is added to the apps list in the left panel of the editor.
5. Now click **Edit** and go to the **Catalogs** tab and assign your app to a catalog.

iRemember

We are assigning our app to a catalog to enable role based access to the app.

For more information, see [Content Management](#).

6. Make any other changes you want to your app and **Save** your changes.


The app is now configured and you can either add it to your site as a full page app, or you can create a launchpad page and add it there as a tile.

iNote

There are many different app types that you can configure. Simply choose the **App Type** under **App Resource Details** in the **Properties** tab of the editor. The properties will change according to the app type that you choose.

For more information about the different app types, see [About App Types](#).

To add the app as a full page app

1. In the Site Designer side navigation panel, click  to open the **Menu Editor**.
2. Click **+** and select **Add Item**.
3. Select **App** as your **Item Type**.
4. Search for and select your app to add it as a full-page app in your site.

5. **Save** your changes. If you preview your site now, you'll see that the app is part of the site menu and is displayed as a full page app in your site.

To add your app to a launchpad page in your site

1. Since you can't add an app to a launchpad page without first creating a tile, let's create a tile for the app as follows:

- a. Open the **Manage App Configuration** editor and select the app in the **App** menu.
- b. Click **Edit** and go to the **Visualization** tab.
- c. Select a **Tile Type**, fill in the relevant tile properties, and **Save**.

For more information about the tile properties, see [Visualization Properties](#).

2. Now add a launchpad page to your site like this:

- a. Under **Page Management**, select **Pages** and click **+** to add a new page.
- b. Name the page and select an SAP Fiori Launchpad page template.

iNote

An SAP Fiori Launchpad page template is only available if your site is based on a site template that includes an SAP Fiori Launchpad page.

- c. Click **Finish**.

You are directed to the SAP Fiori Launchpad **Manage Groups** editor

3. In the **Manage Groups** editor, add the app that you have just created to a group. This ensures that the end user with a specific role can access the app.

- a. Select an existing group or create a new one using the **+** at the bottom of the list of groups.
- b. Click **Edit**.
- c. Go to the **Apps** tab and in the **Assigned Apps** table, click the **+** to assign the app you created to a group.
- d. Make sure that the applicable roles for the end user are assigned to the group.
- e. **Save**.

If you preview your site now, you'll see that the app you added is represented by the tile you created and appears under the group you assigned it to.

Adding an app as a widget

You can also convert an existing HTML5 application to a Portal widget and add it to a page in your site. This is how you do it:

1. First ask your developer to convert your app to a widget.

For more information, see [Convert an Application to a Widget](#).

2. Once the widget is deployed, it will be available in the Site Designer widget gallery. Simply add it to the page.

About Apps, Roles, and Catalogs

Administrators can apply role-based access to pages and apps of freestyle sites using catalogs.

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

What is a role?

A role is a collection of permissions assigned to specific users giving them the ability to perform certain tasks.

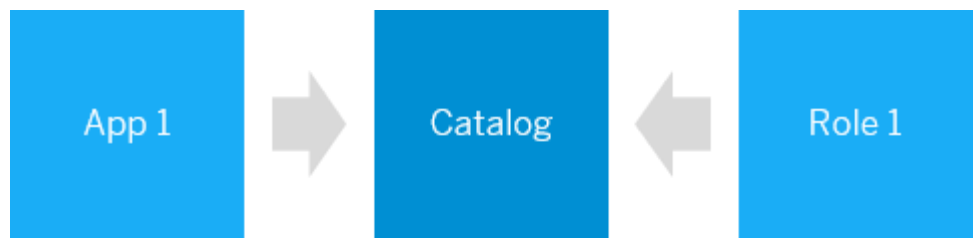
What is a catalog?

A catalog is a collection of pages and apps that are assigned to specific roles.

What is the relationship between apps, roles and catalogs?

Users belonging to a particular role have access to all the pages and apps in the site that are assigned to the same catalog as their role.

Here you can see it more clearly:



App 1 is assigned to the same **Catalog** as **Role 1**. This means that all users assigned to **Role 1** can access **App 1**.

iNote

By default, every new site is assigned to the **Everyone** catalog, which is mapped to the **Everyone** role.

Anyone assigned to this role, has permissions to access all pages and apps in your site.

For more about catalogs, see [Working with Catalogs](#).

More about roles

The existing authorization mechanism in SAP Cloud Platform supports the following role types:

- **Technical Roles**

These roles are internal, predefined roles for subaccount administrators and site creators and are required for accessing and working in design time tools such as SAP Cloud Platform cockpit, Portal Site Designer, and SAP Fiori launchpad configuration cockpit. They include:

Role	Description
SAP Cloud Platform Member	Manages users and their roles within SAP Cloud Platform. These roles support typical tasks performed by users when interacting with the platform. For more information, see Managing Members in the Neo Environment .
TENANT_ADMIN	Enables administration in the Portal (Site Designer and Admin Space) and the SAP Fiori launchpad configuration cockpit. The TENANT_ADMIN role acts as a "super admin" for the portal, and can see the content of all catalogs in the launchpad, can perform all tasks.

- **Custom Roles**

These roles are defined in the SAP Cloud Platform cockpit to restrict access to portal sites, pages, and apps. Custom roles are visible and accessible only within the subaccount where they are created. That's why different subaccounts subscribed to the same application could have different custom roles.

- **Predefined Roles**

These roles are created for users with various job functions in an organization and are imported using an organization's database (identity providers).

Once users are assigned to these roles, you can also restrict access to content (pages and apps) according to different access levels.

For more information, see [Define Access Levels](#).

Define Access Levels

Freestyle sites are a collection of pages and apps. Administrators can manage user access to these pages and apps.

How do I restrict access to the pages and apps in my freestyle site?

- **Pages:** Define access levels for each page and assign the page to a catalog.
- **Apps:** Assign apps to catalogs.

Defining access to your pages

You can either select the access level when you create a page or you can go into **Page Settings** of an existing page and change it there.

iNote

Depending on the access level that you select, system catalogs are automatically generated, which you assign to your page to enable role-based access.

Read more about the following access levels that you can select for each page in your site:

Access Level	Who has access?	What catalogs can I assign to the page?
Public	Authenticated and unauthenticated users	<p>When you create a public page, the following catalogs are automatically available in the Manage Catalogs editor:</p> <ul style="list-style-type: none"> • Everyone (default catalog) • Authenticated Users (system generated) • Guests (system generated)
Guest	Unauthenticated users	<p>When you create a guest page, the following catalogs are automatically available in the Manage Catalogs editor:</p> <ul style="list-style-type: none"> • Everyone (default catalog) • Guests (system generated)

Access Level	Who has access?	What catalogs can I assign to the page?
Role-based	Users with defined roles	When you create a role-based page, you first need to create role-based catalogs in the Manage Catalogs editor and then assign them to your pages.

Defining access to your apps

To define access to your apps, assign them to catalogs as follows:

1. Click the app in the [Pages and Apps](#) panel of the Site Designer. You will immediately access the [Manage App Configuration](#) editor.
2. Click [Edit](#) and in the [Properties](#) tab, under [Assignments](#), assign the app to the relevant catalogs.

iNote

To verify that a page or app is available for public access, make sure that the parameter `authenticationMethod: none` is added to the descriptor file (`neo-app.json`) of the page or app.

Site Management

Administrators can create sites and then manage their sites by defining the site settings, creating a menu structure, determining home pages, and ensuring domain registration.

For more information, click the shapes below:

Create a Site Menu Structure

Administrators can configure a hierarchal menu structure in a freestyle site to help users easily navigate to pages in their portal site.

In a Portal freestyle site, you can create a hierarchal mega menu structure in the [Menu Editor](#) that includes pages, apps, titles, and links.


The mega menu is displayed in a dropdown layout and shows up to three levels at runtime. All menu items are visible at once when you hover over the menu bar. You can also define the appearance of the menu by specifying its dimensions.


iNote

If your site was created before the mega menu was implemented, you have the option to switch your current menu to a mega menu. If you switch to a mega menu, you can't revert back to your current menu therefore we suggest that you duplicate your site and try it out before you make the change.

How do I access the Menu Editor?

There are two ways to access the editor:

- From the side panel of the Site Designer, click 







- If you are working in the **Pages** screen, click  (actions menu) next to any page and then select **Add to Menu** to open the editor.

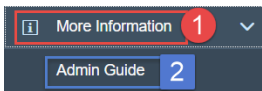


When the **Menu Editor** opens, you will see all menu items (pages, apps, titles, and links) that have been added to the menu.

What can I do in the Menu Editor?

The **Menu Editor** has two tabs.



In the **Structure** tab, you can create the structure of your menu by doing the following:

Action	How to do it
Add pages to the menu structure	<p>There are two ways to add a page to the menu structure:</p> <ul style="list-style-type: none"> • In the Site Designer Pages menu, click  and select Add to Menu. This opens the Menu Editor and you can see your page displayed in the menu structure. Click Save in the Menu Editor. • In the Site Designer side panel, click  (Menu Editor) and then do the following: <ol style="list-style-type: none"> 1. Hover over a menu item row and click  to add the page as an Item or Sub-Item. 2. In the Add Menu Item dialog box, enter a Display Name for the page. 3. In the Select Page list, select the page that you want to add to the menu. 4. Click Add and then Save. <p>iNote</p> <p>When you open the Site Designer of a specific site, in the Pages menu, all pages marked with  are part of the menu structure.</p>
Add apps to the menu structure	<p>In the Site Designer side panel, click  and then do the following:</p> <ol style="list-style-type: none"> 1. Hover over a menu item row and click  to add an app as an Item or Sub-Item. 2. In the Add Menu Item dialog box, change the Item Type to App. 3. Enter a Display Name for the app. 4. In the Select App list, select the app that you want to add to the menu. 5. Click Add and then Save.

Action	How to do it
<p>Add other items to the menu structure.</p> <p>You can also add the following menu items:</p> <ul style="list-style-type: none"> • Title: can be very handy when you want to group a number of links or apps together making navigation easier for the end user. • Link: to link to external sites. <p>For example, we can choose More Information as a title and Admin Guide as a link:</p> <p>This is how your end users will see the menu structure of these items:</p> 	<p>Add a title or link as follows:</p> <ul style="list-style-type: none"> • Click + next to a menu item. • Choose whether you want to add it as an Item or Sub-Item. • In the Add Menu Item dialog box, choose Title or Link as your Item Type. <p>iNote</p> <p>If you select the Link item type, provide the correct URL.</p>
Rearrange the menu items	<p>Drag and drop the menu items up or down to reorder them, or drag them left or right to change their level in the hierarchy.</p> <p>iNote</p> <p>If you drag a menu item to a different location, all sub-items are dragged with it.</p>
Edit a menu item	<p>Hover over a menu item row and click  to open the Edit Menu Item dialog box:</p> <p>From here you can:</p> <ul style="list-style-type: none"> • Change the display name (this is the name that your end users will see). • Add an icon to represent your menu item in the runtime site. • Assign a different menu item (page, app, link, or title) to an existing menu item.
Remove a menu item from the menu structure	<p>Hover over a menu item row and click </p>

In the **Settings** tab, you can do the following:

Action	How to do it
Convert your current menu to a mega menu.	<p>Next to Menu Type, select Mega Menu from the dropdown list.</p> <p>iNote</p> <p>If you have created a new site, the Mega Menu is the only option in the dropdown list.</p>

Action	How to do it
Show or hide the site menu on a desktop or mobile device.	Select the relevant checkbox if you want to show the menu on your desktop or mobile devices.
Determine the space that the menu occupies across the top menu bar. 	From the Menu Width field, specify the width in pixels or select Full to take up the entire width of the screen.
Define the overall height of the menu. 	From the Menu Panel Height field, determine the height of your menu area. Options are narrow, average, or wide.

iRemember
[Save](#) your changes!

Configure Site Settings

In the Portal Site Designer, administrators use the [Site Settings](#) screen to view and edit site settings.

How do you access the Site Settings screen?

Click [Edit](#) on a site to open the Site Designer and then click ⚙️ (Settings) in the navigation menu of the site.

What settings can I edit?

Not all settings are editable. There are some settings that are simply there for you to see. For example, you can preview the layout of the site, you can see when the site was created, and when it was last modified, but you can't change this information.

To edit site settings, first switch to edit mode by clicking [Edit](#) at the bottom right of the screen.

Let's look at the settings in more detail:

General Settings

Setting	More Information
Site Description	Add information about your site.

Setting	More Information
Site Status	<p>Options are:</p> <ul style="list-style-type: none"> • Published - the site is available to end users. • Offline - you have taken the site offline and end users cannot access the site. <p>iNote</p> <p>Administrators can still access the site.</p> <ul style="list-style-type: none"> • Modified - you have edited the site but have not yet published it with the changes. In other words, end users are seeing a previous version of the site. <p>iNote</p> <p>When you edit an already published site, an indicator is displayed next to the Publish button to let you know that the site has been modified. To enable end users to see the changes, republish the site.</p>
Site URL	<p>Opens the runtime version of the site in a new window.</p> <p>The format of the URL is: <custom domain> <site ID></p>
Site Alias	<p>When a site is published, the site URL is automatically generated. By defining a site alias value, you can overwrite the site ID in the URL with a friendlier name.</p> <p>iNote</p> <p>When you duplicate a site, a unique alias is automatically generated consisting of the alias name of the original site as well as a timestamp. We recommend that you change this site alias with a more meaningful name.</p>
Session Timeout Message	<p>Enable or disable the site session timeout message for the site.</p> <p>When enabled (default), after a specific period of time that the user is idle, a message pops up, enabling the user to log back on.</p> <p>You can disable this setting if you want to handle the session timeout in a different way.</p>

System Settings

Setting	More Information
---------	------------------

Setting	More Information
SAPUI5 Version	<p>Select the SAPUI5 version to use for runtime.</p> <ul style="list-style-type: none"> • The Latest Supported Version (or Innovation), is updated according to SAPUI5 releases and contains new features and innovations. • The Maintenance version is a stable version of SAPUI5 that is not updated automatically with new features. <p>iTip</p> <p>The Innovation version is constantly being upgraded and this may have an impact on the functionality of your application. Therefore we recommend that in a productive environment, use the Maintenance version. Use the Innovation version for testing purposes in a development environment.</p> <p>iNote</p> <p>How does this affect your site theme?</p> <ul style="list-style-type: none"> • Selecting the Latest Supported Version version of SAPUI5 changes the theme to SAP Belize. • Selecting the Maintenance 1.38 version of SAPUI5, changes the theme to SAP Blue Crystal. However selecting the Maintenance 1.44 version of SAPUI5, changes the theme to SAP Belize. • When using custom themes, an SAPUI5 upgrade may require that you rebuild the theme in the UI theme designer.
Sign-Out Page	<p>Select a page that you want your end users to open when they sign out of a site.</p> <p>They can be redirected to one of the following pages:</p> <ul style="list-style-type: none"> • Default Sign-Out Page – default sign-out page supplied by SAP. • Site Page – any public page that you created in the Site Designer. The list includes only public pages. <p>iNote</p> <p>Public pages are pages that can be accessed by Public or Guest roles. If you change the access level of a sign-out page and it is no longer a public page, end users are redirected to the default SAP sign-out page.</p> <ul style="list-style-type: none"> • URL – an external web page that you specify. • Log-On Page – redirects to the Portal page that opens when a user logs on. <p>iNote</p> <p>If you delete a page that you've selected as the sign-out page, end users are directed to the default SAP sign-out page</p>
Collect Performance Data	<p>Identify performance issues in your application by tracking UI interaction in Solution Manager.</p>

Setting	More Information
Save App State Data	<p>Some apps are configured to save data using the AppState URL parameter. Set this property to No to stop saving data using the AppState parameter.</p> <p>iNote</p> <p>When set to No, apps that use the AppState functionality might not work as expected and any bookmarks or tiles that were saved using the AppState parameter will no longer load with the saved state.</p> <p>When this property is set to Yes, apps in your site can save potentially personal data. Make sure that your apps comply with local data privacy laws.</p>
Access Type	<p>Select Internal to set up an internal access point landscape. This prevents sensitive OData requests to the corporate network from being routed to the cloud.</p> <p>Select External to set up an external access point landscape. This enables accessing sites from outside the corporate network. This allows you to define access to classic UIs such as SAP GUI and Web Dynpro ABAP.</p>
Access Classic UIs	<p>When accessing the site from outside of the corporate network (External access type), you have the following options for handling classic UIs, which include SAP GUI and Web Dynpro ABAP apps.</p> <ul style="list-style-type: none"> • Yes, and show in launchpad. These apps are displayed in the site, and your end users can access them. • No, but show in launchpad These apps are displayed in the site, but your end users will not be able to access them. • No, and do not show in launchpad. These apps are not displayed in the site.
Launch SAP GUI/WDA apps in place	<p>Display SAP GUI and Web Dynpro ABAP apps in place by default, instead of in a new tab, to improve performance.</p> <p>For more information, see Configure In-Place Navigation for Classic UIs.</p>

Custom Site Properties

Setting	More Information
Key and Value	<p>Optionally add key-value pairs of properties for customizing site implementations.</p> <p>You can also change predefined key-value pairs that were provided as part of the site template.</p>

User Settings

Setting	More Information
Search	Enables search functionality in the site.
SAP Jam Integration	Enables integration between SAP Jam and SAP Fiori launchpad.
Theme Selection	Enables end users to change the theme.

Setting	More Information
Language Selection	<p>Enable this feature to allow an end user to select a language for the site. Each time the same user logs on, the site opens in the language he or she selected.</p> <p>iNote</p> <p>This feature is only available for SAPUI5 1.46 version and above.</p>
SAP Fiori Site Header	<p>Allows you to control whether the site header is visible or not.</p> <p>By default the site header is visible.</p>
Me Area	<p>Enable this feature to allow end users to navigate to the Me Area where they can access many personalization options. If you disable this feature, the end user accesses the personalization options via a dropdown menu on the left side of the home page.</p> <p>iNote</p> <p>This feature is enabled only if you have selected SAPUI5 versions 1.40 and higher.</p>
Recent and Frequent Activity	<p>Specify whether to display in the Me Area of the launchpad the Recent Activity and Frequently Used information.</p>

Home Pages

Setting	More Information
Set multiple home pages	<p>You can enable your site to have more than one home page. This is useful when you want to assign different home pages to different roles.</p> <p>For more information about setting home pages, see Set Home Pages.</p>

Clearing the cache

In the bottom right corner of your screen, you can click [Actions](#) and select [Clear HTML5 Application Cache](#) to manually trigger clearing the cache.

For more information, see [Clearing the App Cache](#).


Set Home Pages

Administrators can set multiple pages and apps as home pages for their freestyle sites.

You can enable your site to have more than one home page. This is useful when you want to assign different home pages to different roles.

How do I set a page as a home page?

- To set a single page as your home page, do the following:
 - Under [Page Management](#) in the Site Designer, select [Pages](#) and find the page you want to set as your home page.
 - Click ▼ (actions menu), and select [Set as home page](#).

- To set multiple pages as home pages, do the following:
 1. In the navigation panel of the Site Designer, click  (site settings).
 2. Click **Edit** at the bottom right of the screen.
 3. Under **Home Pages**, select **Set multiple home pages**, and save.
 4. Go to your **Pages** menu in the Site Designer and assign additional home pages.

iNote

- You cannot publish a site that has no home page.
- A role cannot be assigned to more than one home page. You cannot set a page to be a home page if it is assigned to the same role as an existing home page.

How do I set an app as a home page?

You assign an app as a home page in the **Manage App Configuration** editor. This is how you do it:

1. Under **Content Management** in the Site Designer, select **Apps** to open the **Manage App Configuration** editor.
2. In the **Apps** menu, find the app you want to set as your home page.
3. Click **Edit** at the bottom right of the screen.
4. In the **Properties** tab of the editor, change the Intent Navigation properties to:
 - **Semantic Object:** Home
 - **Action:** show
5. Click **Save**.

Now if you go to **Site Settings**, under **Home Pages**, you will see that your app is defined as a home page.

iNote

If the app is already part of the site menu, and you now set it as a home page, first remove it from the site menu and then add it again (due to the intent change).

What happens when I want to set an app as a home page but I still want to keep the original navigation properties?

You can keep the original navigation properties. Simply duplicate the app and in the duplicated version, change the intent to display it as a home page.

iNote

Remember if you do this, the app will appear twice in the editors **Apps** menu - each with a different intent.

Manage Site Assets

Administrators can view and manage their media files such as images, videos, and HTML files in the Portal Asset Repository.

When you upload an image or any other media file into a widget in a page section, it is also automatically uploaded into the asset repository, where it is available for reuse in other pages and sections of the site.

For example, the next time you add an image widget, click the cog icon to open the widget settings, select Asset Repository as the image source, then select the desired image.

In the Asset Repository, click any site to get a list of assets that exist in that site. By clicking the asset, you can download it for translation or for editing.

Web Content Management

To enable web content editors to edit content of a site, administrators must ensure that certain prerequisites are in place.

Some background about web content management

As the site administrator, you are responsible for the design of the site - the structure, the layout, the number of pages, and the general look and feel of the site. However sometimes in an organization you may have a dedicated person who actually edits the content of your site. In the Portal, this is anyone assigned to the WEB_CONTENT_EDITOR role for a specific SAP Cloud Platform subaccount.

For more information, see [Web Content Editor Guide](#).

What management tasks do you have to do?

- Before any web content editor can access the dedicated Web Content Editor tool, you need to provide them access to the tool by sending them the URL to the site that you want them to edit.

At a later stage you will want to access the tool yourself; either to edit web content or to check what the web content editor is changing. For more information see, [Access the Web Content Editor Tool](#).

- Make sure that you have activated the translated languages in the [Translations](#) service to make the languages available to web content editors. In this way if web content editors want to edit the web content of a site in a different language, they can select one of these languages from a dropdown list in the Web Content Editor tool. For more information, see [Translate Web Content](#).


Access the Web Content Editor Tool

Site administrators and web content editors can access the Web Content Editor tool to edit web content.

You can access the Web Content Editor tool from various access points in the Portal.

How can I access the Web Content Editor tool?

There are three ways that you can open the Web Content Editor tool:

1. In the [Site Directory](#), click [Edit Web Content](#) from the dropdown list of actions on the site's tile.
2. From the Site Designer side navigation panel under [Useful Tools](#), click [Web Content Editor](#).
3. Once your site is open, click  (Tools and Services) and then click [Configure](#) on the tile of the [Web Content Editor](#) service.

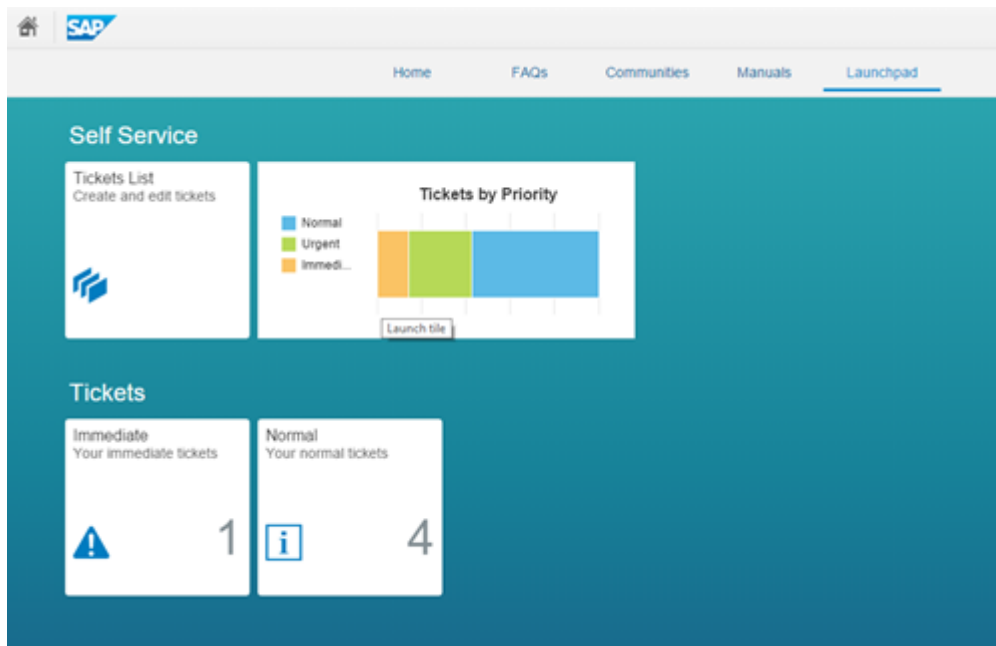
How does a web content editor access the tool?

A web content editor can only access the tool once you provide them with a link to the tool.

Integrating SAP Fiori Launchpad

Administrators can integrate Fiori launchpad capabilities with freestyle sites.

One of the main features of the SAP Cloud Platform Portal service, is its full integration with SAP Fiori launchpad. This integration provides access to launchpad capabilities on cloud while incorporating full capabilities for managing role access permissions via catalogs. You can either base your entire site on SAP Fiori launchpad, or include a launchpad page alongside freestyle pages and third-party content, depending on the best-case scenario for your users.



Related Information

[Create an SAP Fiori Launchpad Site](#)

[Create an SAP Fiori Launchpad Page](#)

Create an SAP Fiori Launchpad Site

Administrators can create SAP Fiori launchpad sites displaying a home page with tiles and links, each representing a business application that an end user can launch.

Context

You can create a site that consists only of an SAP Fiori launchpad. To do this, access the Portal and create a new site in the Site Directory based on the [SAP Fiori Launchpad](#) site template. You can then manage the content of your launchpad from a browser based tool called the SAP Fiori launchpad configuration cockpit.

Let's create an SAP Fiori launchpad site.

Procedure

1. In the Admin Space, select [Site Directory](#).

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

2. Click **+** to open the **Create Site** wizard.
3. Name your site.
4. Select **SAP** as your **Template Source** and from the set of available templates, select **SAP Fiori Launchpad**.
5. Click **Create** and your new SAP Fiori launchpad site is ready for use.

Results

The site is added to the Site Directory and in future when you want to add or maintain the content of your launchpad site, you can look for it there and simply click on **Edit** to open it.

Create an SAP Fiori Launchpad Page

Administrators can add to their freestyle site an SAP Fiori launchpad page, that is parallel to other site pages and is part of the site menu.

Prerequisites

- The freestyle site to which you're adding the page must be based on a site template that contains the SAP Fiori launchpad page template.
- You have configured catalogs for your site.

Adding an SAP Fiori Launchpad Page

Procedure

1. In the Site Directory, hover over the site and click **Edit**.
2. In the Pages panel, click **+** (**Add new page**) to open the wizard.
3. Name your new page.
4. Under **Template Source**, select **This Site** and select the **SAP Fiori Launchpad** page template.
5. Click **Finish** and you will see your new launchpad page in the Pages panel on the left.

iNote

As you click **Finish**, you are automatically directed to the **Manage Groups** editor. Now let's see how to add apps to your new launchpad page.

Adding Apps and Groups to a Launchpad Page

Context

In SAP Fiori launchpad, each app is represented by a tile, and these tiles are displayed in distinct groups. You would typically group apps according to a certain business logic or those most frequently used by users of a given role. Users see only the groups that are assigned to their role.

So you created your launchpad page and were redirected to the **Manage Groups** editor. Let's continue from there.

Procedure

1. At the bottom of the **Groups** panel, click **+** (**New**) to create a new group.
2. In the **Properties** tab, enter a name for the new group.
3. To prevent end-users from customizing this group, select **Locked Group**.
4. Switch to the **Apps** tab and click **+** to open a dialog box where you can select the apps for your group. The apps you selected are now displayed in the **Assigned Roles** table. Each app is added as a tile to the group.

iNote

Use the up or down arrows to order the apps. The order in which the apps are listed is the order in which the tiles are displayed in the launchpad.

5. Now switch to the **Roles** tab and again click the **+** to open a dialog box where you can select the roles. Only users belonging to the roles that you select will be able to access the apps in this group. You can also that the roles you selected, are displayed in the **Assigned Roles** table.
6. Click **Save**.

Frequently Asked Questions (FAQ)

Answers to frequently asked questions regarding the SAP Cloud Platform Portal service.

What units is my site made up of?

[Click here](#) to read about the site building blocks.

How do I edit an existing site?

In the Site Directory, hover over the name of the site and click **Edit**.

What types of content can I add to my site? (Widgets, SAP Jam, SAP Fiori launchpad)?

You can add:

- Widgets provided by your developers or that you create yourself
- Third party applications such as a weather app.

iNote

Third party apps must first be converted into a widget by a developer, using SAP Web IDE.

- An SAP Fiori launchpad page

iNote

To learn more, see [Integrating SAP Fiori Launchpad](#)

For more information on page templates, see [About Page Templates](#)

How can I manage permissions for the pages of my site?

Permissions are handled via catalogs. Catalogs are authorization entities used to provide access to the content of your site. In essence, they are a combination of a list of pages and/or apps, and the roles that have access to these pages. By default, new sites contain the **Everyone** catalog, in which all your pages and apps are assigned to the **Everyone** role - a predefined role that includes all users.

How do I configure the site menu?

If you are using SAP's preconfigured menu, menu configuration is one of the site services, accessed by clicking the wrench icon. After adding a menu item, assign it to a navigation target.


For more information on configuring the site menu, see [Create a Site Menu Structure](#)

My site menu isn't showing.

If you are using SAP's preconfigured menu, click the cog icon to access the site settings, click Edit, and select Show Site Menu. Lastly, click Save.

How do I publish my site?

Publish your site in either of the following ways:

- In the Site Directory, hover over your site, click the down arrow and select **Publish**.
- In the Site Designer, while editing your site, click  (Publish) at the top right of the screen.

Once a site is published, you and your end-users can view it online. To access your site, hover over its card in the Site Directory, and click the URL.

How do I make a page visible to end users?

Pages are made available to end users by connecting them to the site menu. A page that is not available to end-users will display a hidden icon in the Pages and Apps panel.

For more information on configuring the site menu, see [Create a Site Menu Structure](#)

What are page templates?

A page template is a page layout that has a defined structure. It is divided into sections of a given size in fixed positions, and may contain preconfigured content. Depending on the site template you select, your site will contain one or more page templates that determine the general structure of your site's pages.

For more information on page templates, see [About Page Templates](#)

How can I reuse images that have been added to a page?

When you upload an image into an image widget in a page section, it is also automatically uploaded into the asset repository. You can then reuse the image in other pages and sections of the site. The next time you add an image widget, click the cog icon to open the widget settings, select **Asset Repository** as the image source, and select the desired image.

How do I export my site for translation to another language?








Translations are explained in full here: [Translating Texts and Assets of Freestyle Sites](#)

Web Content Editor Guide

Welcome to SAP Cloud Platform Portal's world of web content editing.

iNote

The following image contains links to more information.

	Introduction to web content editing
	About web content
	Accessing the editor
	Editing web content
	Translating web content
	Publishing web content
	Typical end to end scenario

Overview

Web content management capabilities in the Portal allow assigned users to create, edit, and optimize site content.

Let's learn more about web content and how it's managed!

What is Web Content?

Web content is generally managed in a Web Content Management System (CMS) – a software application that enables nontechnical users to easily create, edit, and maintain content of a web site.

Who manages web content in the Portal?

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

Anyone assigned to the WEB_CONTENT_EDITOR role for a specific subaccount, can create and update the content of the sites in that subaccount.

Where is it managed?

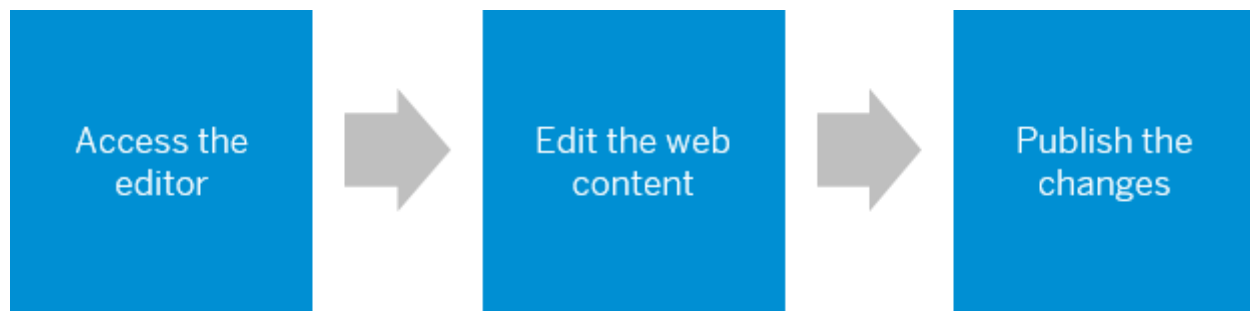
In a dedicated and intuitive tool called the Web Content Editor.

What is the overall workflow?

Basically the process is as follows:

iNote

The following image contains links to more information.



About Web Content

Portal freestyle sites consist of web content that is maintained and edited in a dedicated tool called the Web Content Editor by a person who has been assigned the role of Web Content Editor.

What is web content?

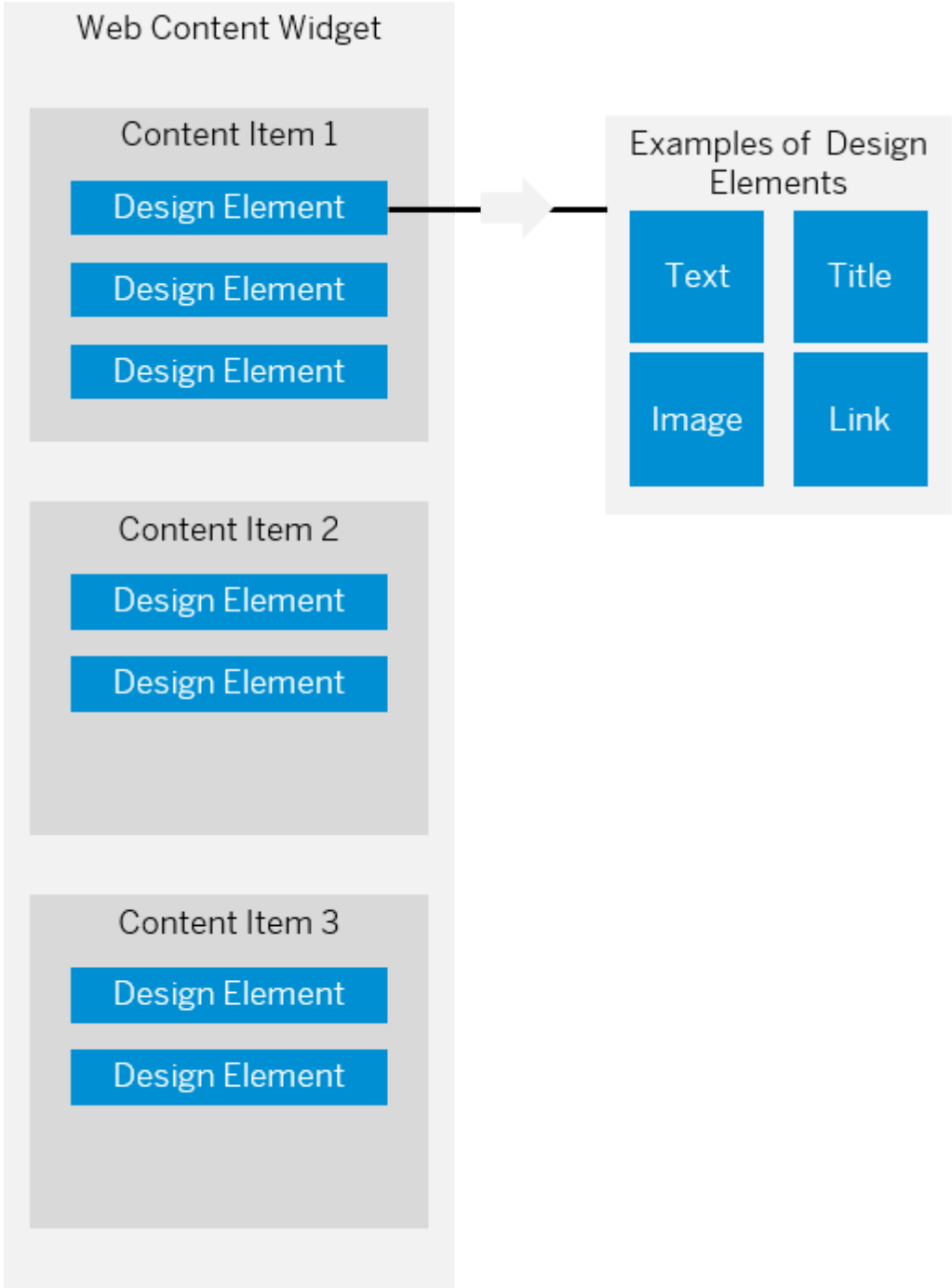
In SAP Cloud Platform Portal, web content is added to a site in the form of widgets that the site administrator selects from a widget gallery in the Site Designer.

What is the structure of this web content?

A web content widget is made up of one or more content items; each with their editable design elements (such as a title, text, images, and links). Web content widgets are divided up into categories:

- A **Basic** web content widget is a single, standalone content item such as a title, link, or image. They are nonrepeatable, meaning that you cannot add additional content items to these widgets.
- More complex web content widgets include multiple content items. These widgets are divided up into the **List**, **Card**, **Carousel**, and **Menu** categories in the Site Designer widget gallery depending on their layout and behavior. They are typically populated with handy content snippets (small chunks of predefined content). The content snippets include repeatable content items each with design elements such as an image, a caption, a link, and so on. Content snippets are repeatable, meaning that you can also add additional content items to these content snippets.

Let's look at this example of a complex web content widget from the **List** category that includes three predefined content items. Each content item includes title, content, image, and link design elements. You can edit any of these elements and you can also add another content item to the widget.



More about the complex web content widgets?

Category	Description
List	Content items are displayed one under the other.
Cards	Content items are displayed next to each other. When the row is full, the content items wrap to the next row depending on the settings determined by the site administrator.
Carousel	Contents items are displayed one by one. The user moves between them using pagination or scrollers.
Menu	The content items are links that enable the user to navigate to any sections on the current page or to other pages, apps, and external links.

Access the Web Content Editor

Web content editors can access the Web Content Editor tool by logging on with a URL provided by the site administrator.

Before you can access the Web Content Editor, make sure of the following:

- You have been assigned to the WEB_CONTENT_EDITOR role. This is done by your subaccount manager in SAP Cloud Platform cockpit.
- Your site administrator has generated and sent you a URL that enables you to access the Web Content Editor tool.

Once these two criteria are in place, you are good to go and you can start editing.

For more information about the Web Content Editor, see [Edit Web Content](#).

Edit Web Content

Web content editors can edit web content of a freestyle site in a dedicated authoring tool called the Web Content Editor.







The Web Content Editor tool consists of two views - a page view and an editor view.




Let's look at each view in more detail.

What can I do in the page view?


As you open the editor, the page view is in focus. On the left you will see a list of pages and page templates with their respective web content. On the right, a preview of the selected page or page template.

Here's what you can do on the page view:

Action	Description
Collapse 	Hide the list of web content under each page or page template name.
	Display the list of web content that is editable under each page or page template name.
Search	Use the search capability if you're editing a site with many pages and they're not all visible at once.
Filter 	Filter web content by type or status. For example, you can filter out all the web content that has a modified status. This can be useful if you want to publish all your changes together.
Close 	Close the list of pages and web content on the left to make more space for previewing the page.
Open 	Open the list of pages and web content.
Edit 	Open the editor view where you can edit the web content items. iNote Note that you can edit the web content by clicking the icon next to its name under the page or directly from the page preview.




Action	Description
Preview a page or page template	Click any page or page template to preview its content on the right. iNote All editable web content in the page is highlighted with a yellow border. If you click a row next to a web content name, only that web content is highlighted.
Check status of web content	A yellow indicator next to the web content name means that you have not yet published the changes you made and its status is modified.
Refresh 	Update the web content of all pages and page templates. iNote Your site administrator may have modified the site. For example, they might have deleted a page with web content that is displayed in the tool or added a new page with web content. Refreshing the editor updates the page view with these changes.
Preview 	Display your latest changes to the web content as it would appear at runtime before you actually publish the changes. You can preview the site according to device and language. If you're happy with the changes, go ahead and publish.
Publish 	Publish the changes you made to the web content to make it visible to end users. For more information about publishing, see Publish Web Content .

What can I do in the editor view?

Once you've clicked  next to a web content name or from the page preview, the editor view opens and a list of the content items are displayed on the left. Click any content item to preview its design elements on the right (such as title, description, links, and so on).

Here's what you can do on the editor view:

Action	Description
Edit	<p>Edit web content as follows:</p> <ul style="list-style-type: none"> • If the web content consists of a single content item such as an image, a video, or a link, the editor opens immediately and you can start editing. • If the web content consists of multiple content items (such as a card widget that includes three content items; each with predefined content), the editor opens displaying each content item. Clicking on each content item displays its design elements on the right, each of which you can edit. <p>iNote</p> <p>If the web content is a menu, the content items displayed on the left are links and you can edit each link in the menu separately to navigate to a specific target.</p> <p>If the web content contains long text content items, you can use a Rich Text Editor tool (with full link capabilities), to style your text.</p> <p>For more information about web content, see About Web Content.</p>

Action	Description
Rename	Click  next to the web content name to give it a more meaningful name. Once you've saved it, the new name appears in the page view under the relative page or page template.
Add	Click  at the bottom of the list of content items to add a new content item.
Delete	Click  at the bottom of the list of content items to remove a content item.
Reorder	Reorder the content items according to how you want them displayed on your page. Simply hover over the grip control next to the content item name and drag it to its new location.
Revert	Remove all the changes you made and revert back to the last published version of the web content.
Publish	<p>Publish the changes you made to the web content.</p> <p>iNote</p> <p>You can only publish your changes to the web content if the site has been published by the site administrator.</p>

For a detailed example of how to edit web content in a freestyle site, see [Typical Workflow of a Web Content Editor](#).

Translate Web Content

Web content editors can translate the web content of a site directly from the Web Content Editor tool by selecting one of the languages activated by the site administrator.

Some background about how the language of your site is determined

Site administrators usually build a site in the language of their browser – this is the default language of the site. They then have the site translated into several other languages using the [Translations](#) service.

To do this, the site administrator accesses the [Translations](#) service from the Site Designer, exports the site content for translation, and then imports the translated content back to the site. The translated languages must then be activated to make them available to end-users as additional language options.

How can I translate web content directly from the Web Content Editor tool?

When you open the Web Content Editor, the tool is displayed in the default language (the language used by the site administrator to build the site).

As a web content editor, you simply select one of the activated languages from the dropdown list of content languages and start editing. In this way, you are assigning the new content to the language you select without having to go through the entire translation process.

iNote

The choice of languages that you can select is determined by the languages enabled by the site administrator in the [Translations](#) service.

Once you have edited the content, save your changes and preview the runtime result before you publish.

iNote

This is custom documentation. For more information, please visit the [SAP Help Portal](#)

A few things to remember:

- If the site administrator has not activated additional languages for the site, the web content of the site will be displayed only in the default language.
- You are only translating the web content texts. You are not translating the editor, so all the fields and buttons will still show up in the default language of the site.
- When you add a new content item to the web content, it is added with the default initial content. You can then replace this content with your own content in the language you selected. Also, note that when you switch between existing content items and the ones you may have just added, you will see that they appear in the language you selected.
- If your web content does not translate once you select a content language, it may mean that the content is from a custom widget that has no translation property. Speak to your developer, who will be able to add the property.

Workflow example for the Site Administrator and the Web Content Editor

You are a site administrator and you have built your site in English. Your site needs to be translated into German and Chinese.

1. Download all the text and asset files, export them to the [Translations](#) service and then import the files back again once they have been translated.
2. Activate the German and Chinese languages in the [Translations](#) service to make them available for selection in the Web Content Editor tool.
3. Send the URL of the site you want edited to the Web Content Editor.

You are the web content editor of the site and you want to edit the web content of the site. You open the Web Content Editor tool with the URL that the site administrator has sent to you. You see that the content is in English (language that the site was built with). However, you prefer switching to German.

1. Go to the [Content Language](#) button at the bottom of the editor.
2. From the dropdown list select German. All web content texts switch to German.
3. Continue editing in German, save your changes, preview the site, and publish.

Publish Web Content

Web content editors can publish their changes to the web content of a Portal freestyle site.

When you click  (Publish) in the page view of the Web Content Editor, there are two possible scenarios:

Publishing changes to web content in a published site

Your changes are published and the end user can see these changes. If at a later stage the site administrator removes the page that you edited from the site, your changes are still visible to the end user until the site administrator republishes the site.

Publishing changes to web content in an offline site

If your site has not been published by the site administrator, you cannot publish the changes you made to the web content of that site.

iNote

When you edit web content, its status changes to modified. This is indicated by a yellow indicator next to the name of the web content in the page view. When you publish your changes, the yellow indicator disappears.

Typical Workflow of a Web Content Editor

This is a real-world example of how a web content editor would edit the content of a Portal freestyle site and interact with the Web Content Editor tool.

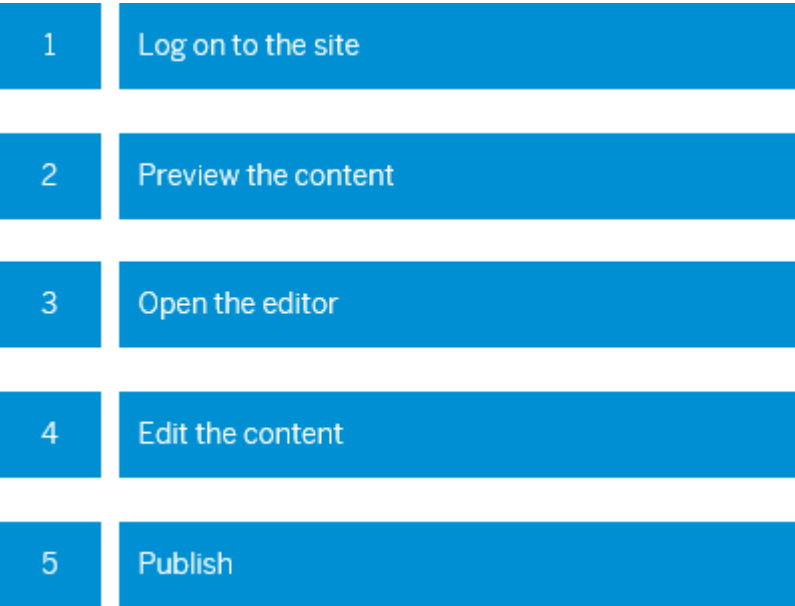
As a Web Content Editor of a company, one of your responsibilities is to edit web content in their sites. You have been asked to edit the content for the Our Products site.


To do this, you access a dedicated tool called the Web Content Editor, where you can easily view the web content and edit it. Once you are done, you can publish the changes you made to the site.

iNote

The diagram below shows the high-level steps of how to edit the web content of the Our Products site. Click each step to display more details.

So let's get started!




	Log on to the site	Preview the content	Open the editor	Edit the content	Publish
Hover over each step for a description. Click the step for more information.	that you are editing The site administrator has already sent you the URL enabling you to access the Web Content Editor for this site.	In the page view, expand each page and the page template to see the editable web content for each. You can preview on two levels:	To edit web content, simply click its name to open the editor. <ul style="list-style-type: none">If the web content consists of a single component such as an	Now let's get started and edit our content. iRemember Don't forget to save your changes after editing each element.	Now go back to the page view and make sure that all your changes appear in the page or page template preview. Click  (publish) to open a dialog box

As you access the editor, a page view opens and you see that the Our Products site includes the following three pages:

- Page: **About Us** that includes a title, an image, and a link.
- Page: **Our Products** that includes a list of three testimonials.
- Page Template: **Footer** that includes links to contact information.

- Click a page to preview its content on the right. Note that in the preview, all editable web content is highlighted with a yellow border.
- Click a row next to the name of the web content to preview it on the right.

iTip

Before you start editing, click  (refresh) to update the web content. Your site administrator may have modified the site (such as deleted a page with web content that still appears in the editor). Refreshing updates the page view with these changes.

For more information about web content, see [About Web Content](#).


image or a link, the editor opens immediately and you can start editing the content.

- If the web content consists of multiple components, the editor opens displaying each component. Clicking on each component displays its elements on the right, each of which you can edit separately.

- On the **About Us** page, we want to replace the image:

Click the **Image** web content and in the editor, click **Replace**, and upload the image of your choice.

- In the **Our Products** page, we want to add another testimonial:

1. Click the **Testimonial** web content in the page view to open the editor. The editor opens showing that this content consists of three components (three testimonials).
2. To add another component, click **+**.
3. Replace the default title, content, and image with your own content.
4. Click  (rename) and change the name to 'What Our Customers Say'.

- In the **Contact Information** page template, we want to change the

with the following options:

- Select a page or page template and publish all its modified content.
- Select the row next to a web content name and publish it.
- Publish all modified web content in your site.

iNote

A yellow indicator next to a web content name indicates that its status is modified and it has not yet been published.

iNote

You can only publish the changes you made, if your site has been published by the site administrator.

display text of the
link button.

Click the Link Button
web content and in
the editor, change
the **Display Text**.

We are now done with our
editing - save your changes
and let's move to the next
step.