SAP NetWeaver
How-To Guide

How To... Create New Web Forms

Web Page Composer for SAP NetWeaver 7.3

Applicable Releases:

Portal for SAP NetWeaver 7.3

Version 1.0

April 2011

HOW-TO GUIDE

THE BEST-RUN BUSINESSES RUN SAP™

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Icons

Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1. Business Scenario

With the Web Page Composer for SAP NetWeaver 7.3, you use Web forms to enter content for areas. If you want to use your own Web forms and they are structured differently to the Web forms delivered by SAP, you must work through some configuration steps. With the new Web form a new type of Web content is created.

This How-To Guide explains how to create new Web forms for Web Page Composer. To walk you through the steps necessary to complete the task, we have used an example Web form called *Example*.

2. Background Information

Make sure that the following components have been successfully installed:

- SAP NetWeaver Developer Studio
- SAP NetWeaver 7.3

3. Prerequisites

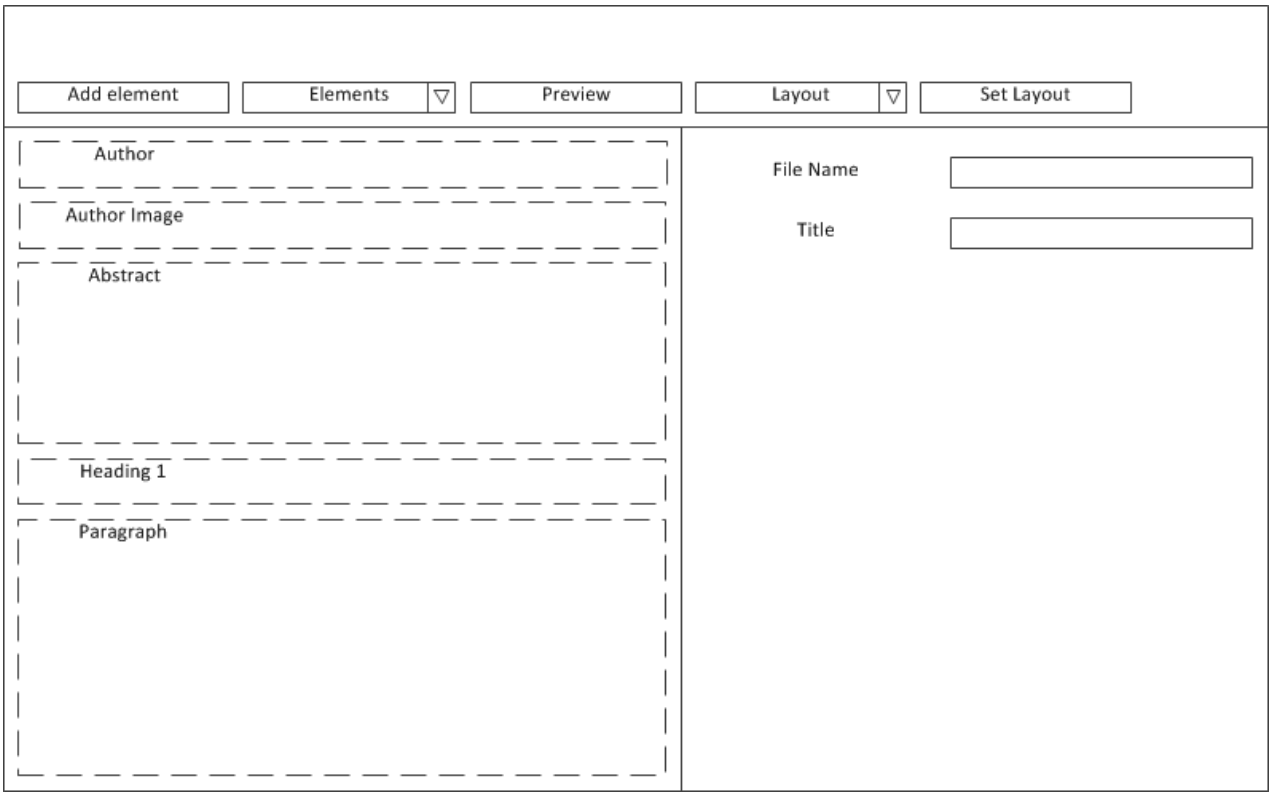
Before you go on, make sure that the following prerequisites are fulfilled:

- The *System Admin* and *Content Admin* portal roles are assigned to your user.
- The *WPC Editor* role is assigned to your user (required for testing purposes only).

4. Creating New Web Forms

In our example we will create a Web form called *Example*. This Web form will contain the following elements: *Author*, *Author Image*, *Abstract*, *Heading 1* and *Paragraph*. The form will also have the following document properties: *File Name* and *Title*.

The graphic below provides you with an overview of the screen areas, the elements and the properties of the *Example* Web form. As you can see in the header, it will be possible to add elements, have a preview and change the layout.



To be able to use the new Web form, walk through the following steps:

1. Create the necessary XML and XSL files.
2. Work through the configuration steps for new Web forms.
3. Restart the portal.

4.1 Create the necessary XML and XSL Files

4.1.1 Define the Web form

Create an XML file and name it *wpc_example.xml* (the attached ZIP file contains this file as an example). The XML definition contains information about the document schema.

In our example XML file we use the following description:

```
<documenttype id="wpc_example" description="xml.xlbl.wpc_example"
showpreview="true" showelementlist="true">
   <properties>
```

```
      <property id="fileName" description="xml.xlbl.filename"
type="inputfield" size="25" isrequired="true" isfilename="true" />
      <property id="title" description="xml.xlbl.title" type="inputfield"
size="25" isrequired="true" />
   </properties>
   <elements>
      <element id="author" description="xml.xlbl.author"
type="peoplepicker" size="25" default="true"/>
      <element id="authorimage" description="xml.xlbl.author_image"
type="imageselect" default="true"/>
      <element id="abstract" description="xml.xlbl.abstract"
type="textedit" size="5" default="true" />
      <element id="heading1" description="xml.xlbl.heading1"
type="inputfield" size="50" default="true"/>
      <element id="heading2" description="xml.xlbl.heading2"
type="inputfield" size="50" default="false"/>
      <element id="paragraph" description="xml.xlbl.paragraph"
type="htmledit" default="true"/>
      <element id="image" description="xml.xlbl.image" type="imageselect"/>
      <element id="furtherreading" description="xml.xlbl.related_links"
type="wpclink" hastitle="true" />
   </elements>
</documenttype>
```

Explanatory notes:

<documenttype> Element

The <documenttype> element is the root element. Its attributes describe the general settings for the editor when editing documents of this type.

The <documenttype> element must contain exactly one <properties> element and exactly one <elements> element. The following table provides and overview of the available attributes of the <documenttype> element:

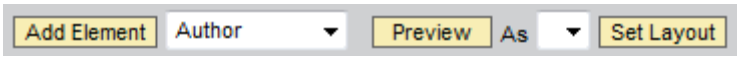
Attribute	Required and Default Value	Description
ID	Required	Document type ID. This ID must be unique among all configured document types.
description	Required	Document type ID. This ID must be unique among all configured document types.
showpreview	Optional Deafaul: false	Defines whether or not the framework contains the preview function. This function allows users to display a preview with all available stylesheets.
showelementlist	Optional Default: false	Defines whether or not the framework allows users to insert additional elements into the document.

		If you set this attribute to `true`, users can insert all elements for which the `singleinstance` attribute is set to `false`.

In our example we define the attributes `showpreview` and `showelementlist` as follows:

- `showpreview="true"`
- `showelementlist="true"`

We do so in order to be able to use the preview function and to be able to insert additional elements into the new document of the type *Example* when editing in the portal:



<properties> Section

The <properties> section must contain one or more <property> elements. These elements define the document properties.

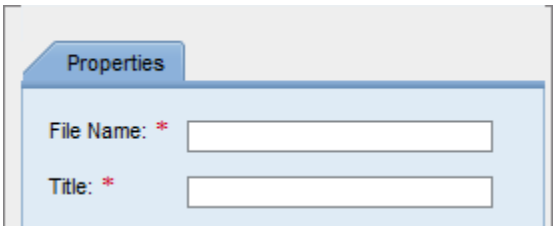
The following table provides an overview of the available attributes of the <property> element:

Attribute	Required and Default Value	Description
`ID`	Required	Unique property ID. This ID must be unique within the document type.
`description`	Required	Key for the country-specific description of the property. For localization purposes, the system uses the bundle file for the KM configuration object that the XML references.
`type`	Required	Specifies the editor component that the system uses to render the property and generate the XML output for the property element. The value must match the name of the component in the KM configuration. You can find the components in the standard system in the KM configuration at *Web Page Composer* → *Editor* → *Editor Components* → *Components.*
`size`	Optional Default: `40`	Size of the property. This must be an integer value. If the editor component supports this, the system uses this value to define the size of the

		components during rendering. If a property is rendered as an input field, for example, the system uses this value for the size of the input field. This attribute is only evaluated if the component supports it.
`isrequired`	Optional Default: `false`	Defines whether or not an entry is mandatory. This attribute is only evaluated if the component supports it.
`isfilename`	Optional Default: `false`	Defines whether or not the system uses the value as the name of the resource created. This attribute may be set to true in only one property within the document type.
`defaultvalue`	Optional	Default value for the property.
`maxlength`	Optional Default: `-1` (unlimited)	Maximum length of the property value. For example, this attribute defines the maximum length of input fields. This attribute is only evaluated if the component supports it.
`hastitle`	Optional Default: `true`	Defines whether or not the property has a title field. Some components use this attribute to determine whether or not an input field is displayed for a title (for example, for an ALT text). This attribute is only evaluated if the component supports it.
`property`	Optional	ID of a KM property in which the property value is stored. The property must be defined in the service for properties and metadata (*System Administration → System Configuration → Knowledge Management → Content Management → Global Services → Property Metadata Service → Properties*) and it must be assigned to the *wpc_wcm* namespace (namespace alisas: *wpc*).

		Example: *wpc_wcm_rss*

In our example we define the *File Name* and *Title* properties, which are visible in the *Properties* section of the Web form. The (*) asterisk next to the property names indicates that these attributes are mandatory (`isrequired="true"`).



`<elements>` Section

The `<elements>` section of the document type must contain one or more `<element>` elements. These elements define the content of the editor.

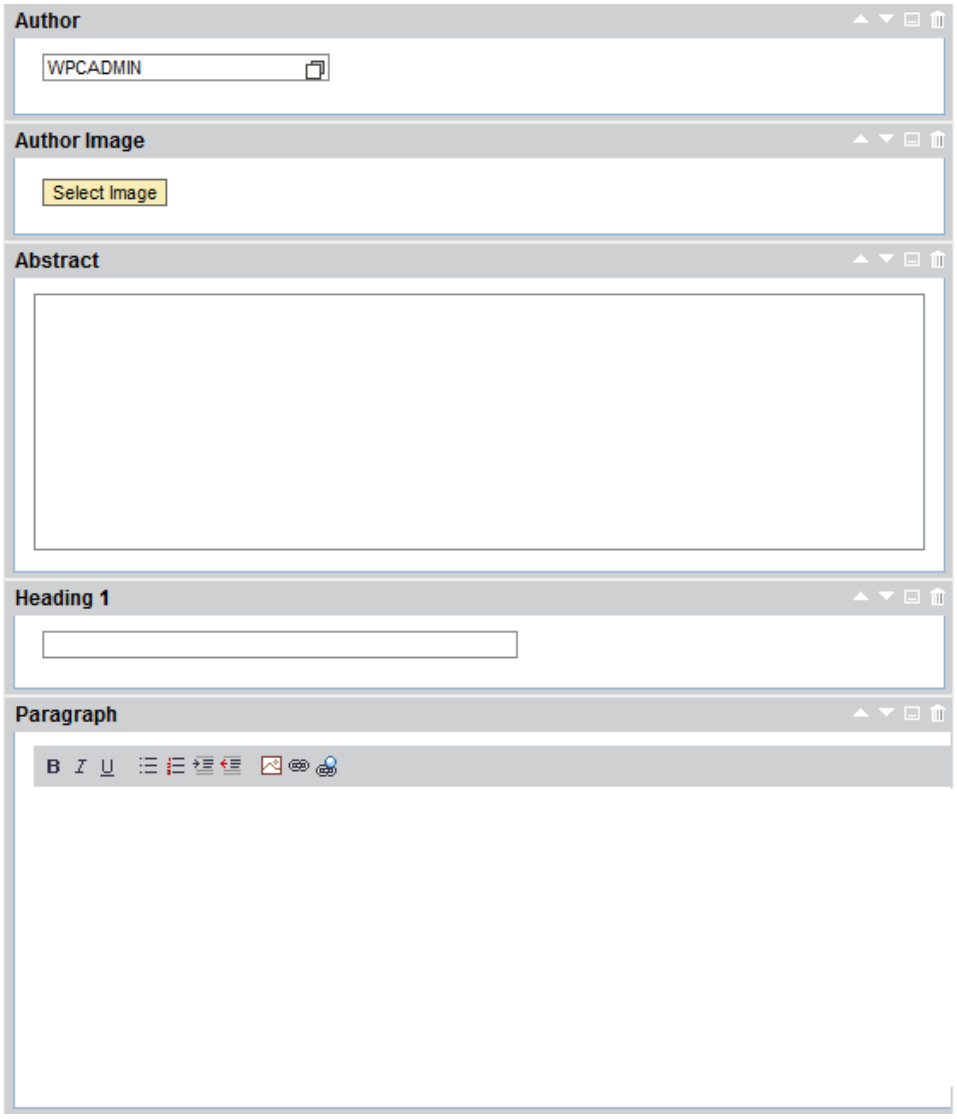
The following table provides an overview of the available attributes of the `<element>` element:

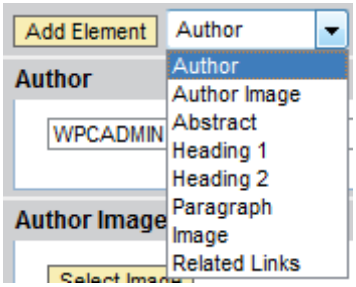
Attribute	Required and Default Value	Description
`ID`	Required	Element ID. This ID can be freely chosen but must be unique within the document type.
`description`	Required	Key for the country-specific description of the property. For localization purposes, the system uses the bundle file for the KM configuration object that the XML references.
`type`	Required	Specifies the editor component that the system uses to render the property and generate the XML output for the property element. The value must match the name of the component in the KM configuration.
`size`	Optional Default: 40	Size of the property. This must be an integer value. If the editor component supports this, the system uses this value to define the size of the components during rendering. If a property is rendered as an input field, for example, the system uses this value for the size of the input field.

		This attribute is only evaluated if the component supports it.
`isrequired`	Optional Default: false	Defines whether or not an entry is mandatory. This attribute is only evaluated if the component supports it.
`defaultvalue`	Optional	Default value for the property.
`maxlength`	Optional Default: -1 (unlimited)	Maximum length of the property value. For example, this attribute defines the maximum length of input fields. This attribute is only evaluated if the component supports it.
`hastitle`	Optional Default: true	Defines whether or not the property has a title field. Some components use this attribute to determine whether or not an input field is displayed for a title (for example, for an ALT text). This attribute is only evaluated if the component supports it.
`property`	Optional	ID of a KM property in which the property value is stored. You must define define the property in the property metadata service (*System Administration → System Configuration → Knowledge Management → Content Management → Global Services → Property Metadata Service → Properties*) and assign it to the *wpc_wcm* namespace (namespace alisas: *wpc*). Example: *wpc_wcm_rss*
`singleinstance`	Optional Default: false	Defines whether or not the element is part of the Add Element list. If this attribute has the value false, users can add new instances of the element (provided this is activated for the document type).
`default`	Optional Default: false	Defines whether or not the element is included in the document by default when users

		create a new document. If this value is false, users have to add the element to the document manually. If the singleinstance attribute is set to true and the default attribute is set to false, users cannot add the element to the document in any way.
`nodelete`	Optional Default: false	Defines whether or not the element can be deleted from the document.

In our example we define the elements `author`, `authorimage`, `abstract`, `heading1` and `paragraph` as default. The elements `heading2`, `image` and `wpclink` can be added as required.

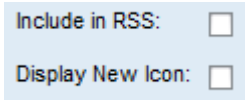
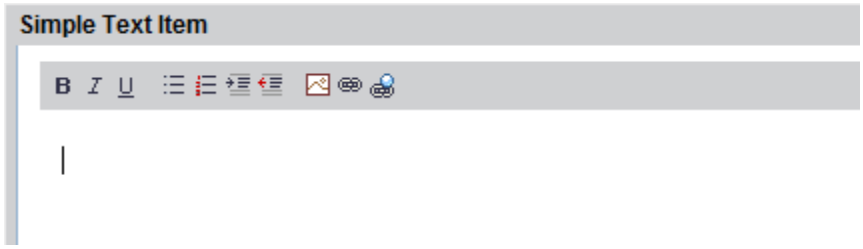


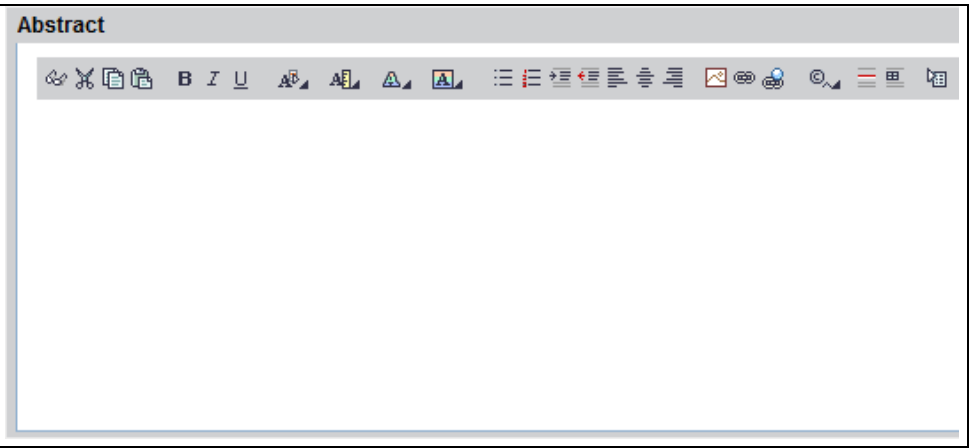
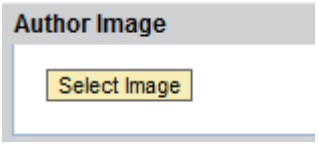
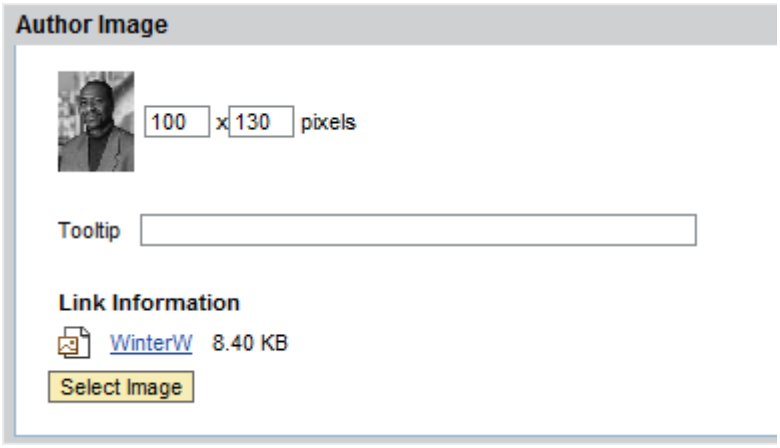
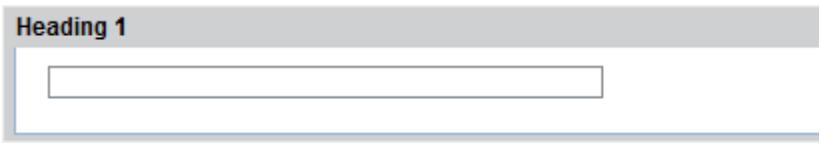


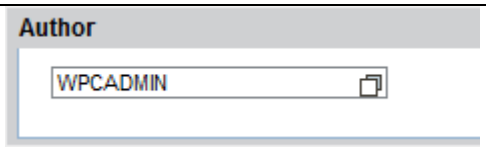
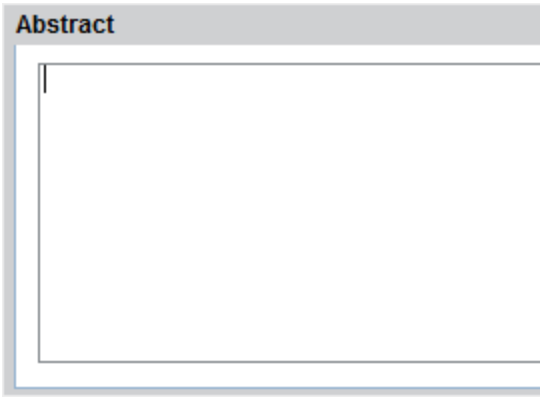
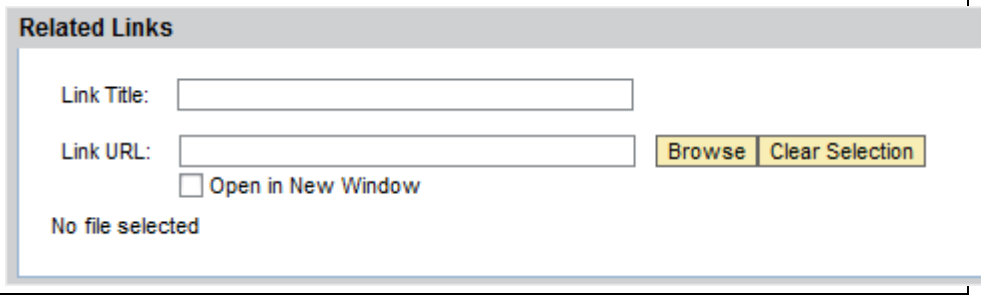
Available element types

To find the editor components delivered by SAP, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Editor* → *Editor Components* → *Components*.

The following table describes all the editor components delivered by SAP:

Component ID	Description
`checkbox`	Checkbox component This editor component provides a standard checkbox that is used to enable or disable standard Web Page Composer functionality (for example, *Include in RSS*, *Show Table of Contents*, *Display New Icon*). 
`htmledit`	HTML edit component This editor component provides a field with text editing functionality and basic formatting features. It also allows Web page content to be copied and pasted. When using the Copy/Paste functionality, HTML characters and tags are escaped. 
`htmleditadvanced`	Advanced HTML edit component This editor component provides a field with text editing functionality, functionality for creating tables, and advanced formatting features. The Advanced HTML edit component offers the same capabilities as the HTML Editor used in KM.

	
`imageselect`	Image Select component This editor component provides a field for image selection. Users can browse for images stored in Knowledge Management. The component also allows users to resize the selected image.  After selection of an image this screen appears: 
`inputfield`	Input field component This editor component provides a single-line input field where users can enter text. 
`peoplepicker`	People Picker Component This editor component provides a user selection dialog box that allows user selection from User Management. The user selection dialog box is the same as the one used in Knowledge Management and Collaboration.

	
`textedit`	Text edit component This editor component provides a multi-line input field where users can enter text. 
`wpclink`	Link component This editor component provides a field that allows users to create links. Users can enter the title of the link and the link URL. Using the *Browse* functionality, users can also create links to documents stored in Knowledge Management. 

4.1.2 Define the presentation of the Web form

Create an XSL file and name it *wpc_example.xsl* (the attached ZIP file contains this file as an example).

In our example *wpc_example.xsl* file we use the following description:

```
<?xml version="1.0"?>
<!DOCTYPE stylesheet [
   <!ENTITY apos   "'" ><!-- replace ' with html escape character for ' -->
]>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:wpc="com.sap.nw.wpc.km.service.linkmanager.XsltHelper">
    <xsl:output method="html"/>
   <xsl:template match="/">
      <div id="wpccas" class="wpcclearfix">
         <a name="top">
         </a> <!-- anchor tag for any Back-to-Top links  -->
         <div id="casmain">
         <div id="wpccascontent" >
```

```
            <h1 class="wpch1">
               <xsl:value-of disable-output-escaping="yes"
select="document/properties/property[@type='title']"/>
               <xsl:if
test="document/properties/property[@type='displayNewIcon']/@value='true'">
                  <span class="new">
                     <xsl:value-of disable-output-escaping="yes"
select="wpc:getString('xsl.xmsg.new', string(document/@locale))"/>
                  </span>
               </xsl:if>
            </h1>
            <xsl:if
test="document/properties/property[@type='showTOC']/@value='true'">
               <h3>
                  <xsl:value-of disable-output-escaping="yes"
select="wpc:getString('xsl.xtit.table_of_contents',
string(document/@locale))"/>
               </h3>
               <ul class="toc">
                  <xsl:for-each select="document/elements/element">
                     <xsl:if test="@type='heading1'">
                        <li>
                           <a class="wpca">
                              <xsl:value-of disable-output-escaping="yes"
select="current()" />
                              <xsl:attribute name="href">
                                 #section1
                                 <xsl:value-of disable-output-
escaping="yes" select="position()"/>
                              </xsl:attribute>
                           </a>
                        </li>
                     </xsl:if>
                     <xsl:if test="@type='heading2'">
                        <ul class="wpcul">
                           <a class="wpca">
                              <xsl:value-of disable-output-escaping="yes"
select="current()" />
                              <xsl:attribute name="href">
                                 #section2
                                 <xsl:value-of disable-output-
escaping="yes" select="position()"/>
                              </xsl:attribute>
                           </a>
                        </ul>
                     </xsl:if>
                  </xsl:for-each>
               </ul>
            </xsl:if>
            <xsl:for-each select="document/elements/element">
               <xsl:if test="@type='abstract'">
                  <p class="wpcp">
                     <xsl:value-of disable-output-escaping="yes"
select="current()" />
                  </p>
               </xsl:if>
               <xsl:if test="@type='authorimage'">
                  <xsl:if test="string-length(@height)!=0 and string-
length(@width)!=0">
                     <img alt="Author Photo">
```

```
                        <xsl:attribute name="src">
                           <xsl:value-of disable-output-escaping="yes"
select="wpc:getWebDavAccess(string(current()))"/>
                        </xsl:attribute>
                        <xsl:if test="string-length(@height)!=0">
                           <xsl:attribute name="height">
                              <xsl:value-of disable-output-escaping="yes"
select="@height"/>
                           </xsl:attribute>
                        </xsl:if>
                        <xsl:if test="string-length(@width)!=0">
                           <xsl:attribute name="width">
                              <xsl:value-of disable-output-escaping="yes"
select="@width"/>
                           </xsl:attribute>
                        </xsl:if>
                     </img>
                  </xsl:if>
               </xsl:if>
               <xsl:if test="@type='author'">
                  <p class="wpcp">
                     <xsl:value-of disable-output-escaping="yes"
select="wpc:getDisplayName(string(current()))"/>
                  </p>
               </xsl:if>
               <xsl:if test="@type='heading1'">
                  <h2 class="wpch2">
                     <xsl:attribute name="id">
                        section1
                        <xsl:value-of disable-output-escaping="yes"
select="position()"/>
                     </xsl:attribute>
                     <xsl:value-of disable-output-escaping="yes"
select="current()" />
                  </h2>
               </xsl:if>
               <xsl:if test="@type='heading2'">
                  <h5 class="wpch5">
                     <xsl:attribute name="id">
                        section2
                        <xsl:value-of disable-output-escaping="yes"
select="position()"/>
                     </xsl:attribute>
                     <xsl:value-of disable-output-escaping="yes"
select="current()" />
                  </h5>
               </xsl:if>
               <xsl:if test="@type='paragraph'">
                  <p class="wpcp">
                     <xsl:value-of disable-output-escaping="yes"
select="current()" />
                  </p>
                  <div class="backtotop">
                     <a href="#"
onclick="javascript:window.location.reload(false)">
                        <xsl:value-of disable-output-escaping="yes"
select="wpc:getString('xsl.xmsg.back_to_top', string(/document/@locale))"/>
                     </a>
                  </div>
               </xsl:if>
```

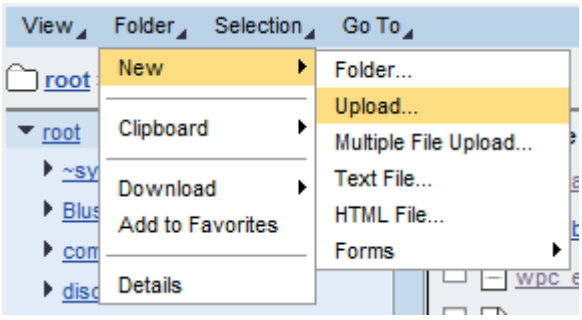
```
                <xsl:if test="@type='image'">
                    <img>
                        <xsl:attribute name="src">
                            <xsl:value-of disable-output-escaping="yes"
select="wpc:getWebDavAccess(string(current()))"/>
                        </xsl:attribute>
                        <xsl:if test="string-length(@height)!=0">
                            <xsl:attribute name="height">
                                <xsl:value-of disable-output-escaping="yes"
select="@height"/>
                            </xsl:attribute>
                        </xsl:if>
                        <xsl:if test="string-length(@width)!=0">
                            <xsl:attribute name="width">
                                <xsl:value-of disable-output-escaping="yes"
select="@width"/>
                            </xsl:attribute>
                        </xsl:if>
                    </img>
                </xsl:if>
            </xsl:for-each>
            <xsl:variable name="rlCount"
select="count(document/elements/element[@type='furtherreading']/@rid)"/>

                <xsl:if test="$rlCount">
                    <div id="wpccassidebar">
                        <div class="wpcbox">
                            <div class="wpcboxhead">
                                <xsl:value-of disable-output-escaping="yes"
select="wpc:getString('xsl.xmsg.related_links',
string(/document/@locale))"/>
                            </div>
                            <ul class="wpcul">
                                <xsl:for-each
select="document/elements/element">
                                    <xsl:if test="@type='furtherreading'">
                                        <li>
                                            <a class="wpca">
                                                <xsl:attribute name="href">
                                                    <xsl:value-of disable-output-
escaping="yes" select="wpc:getAccessLink(string(@rid),
string(/document/@locale))"/>
                                                </xsl:attribute>
                                                <xsl:value-of disable-output-
escaping="yes" select="@title"/>
                                            </a>
                                        </li>
                                    </xsl:if>
                                </xsl:for-each>
                            </ul>
                        </div>
                    </div>
                </xsl:if>
            </div>
        </div>
    </div>
  </xsl:template>
</xsl:stylesheet>
```

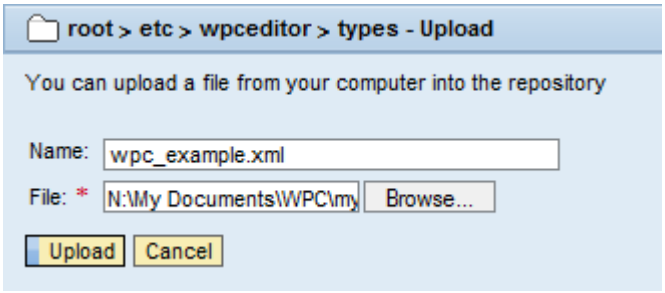
4.1.3 Upload the XML and XSL files to the *etc* repository

In the portal, navigate to *Content Management* → *KM Content* → *etc* → *wpceditor*

1. Upload the XML file in the *types* folder.
   a. Locate the *types* folder and open it.
   b. In the menu bar, choose *Folder* → *New* → *Upload.*

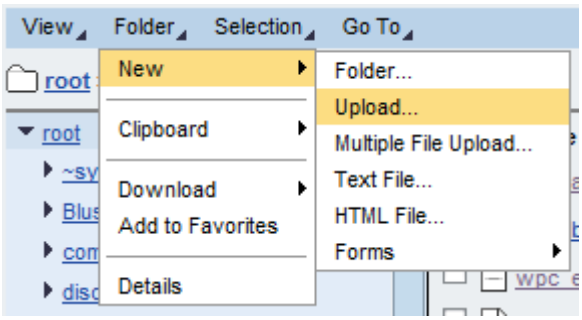


   d. To browse your local repository, choose *Browse.*
   e. Select the XML file, make sure that *wpc_example.xml* is entered as a file name, and then choose *Upload*.



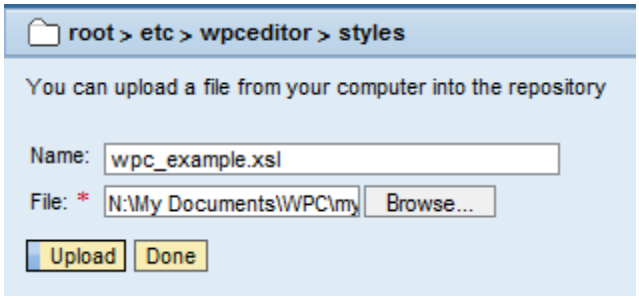
   a. Choose *Done*, and then continue with uploading the XSL file.

2. Upload the XSL file in the *styles* folder.
   a. Locate the styles folder and open it.



   b. In the menu bar, choose *Folder* → *New* → *Upload*.
   c. To browse your local repository, choose Browse.

d. Select the file, make sure that *wpc_example.xsl* is entered as a file name and the choose *Upload*.



e. Choose *Done*, and then continue with the configuration steps for new Web forms.

Note

Alternatively, you can use SAP NetWeaver Portal Drive to upload the files to their specific repository.

4.2 Work through the configuration steps for new Web forms

To create a new Web form, you must perform the following configuration steps:

1. Create a new resource type.
2. Create an iView for the new Web form.
3. Create a new Web resource type.
4. Reference the XSL file in the stylesheet object.
5. Create a new stylesheet group.
6. Reference the XML file in the document type.
7. Restart the portal.

4.2.1 Create a new resource type

You require a separate resource type in KMfor each new Web form.

1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Global Services* → *Resource Types* → *Resource Types.*
2. To create a new resource type, choose *New*.
3. Enter values for the `Name` and `Resource Type ID` parameters

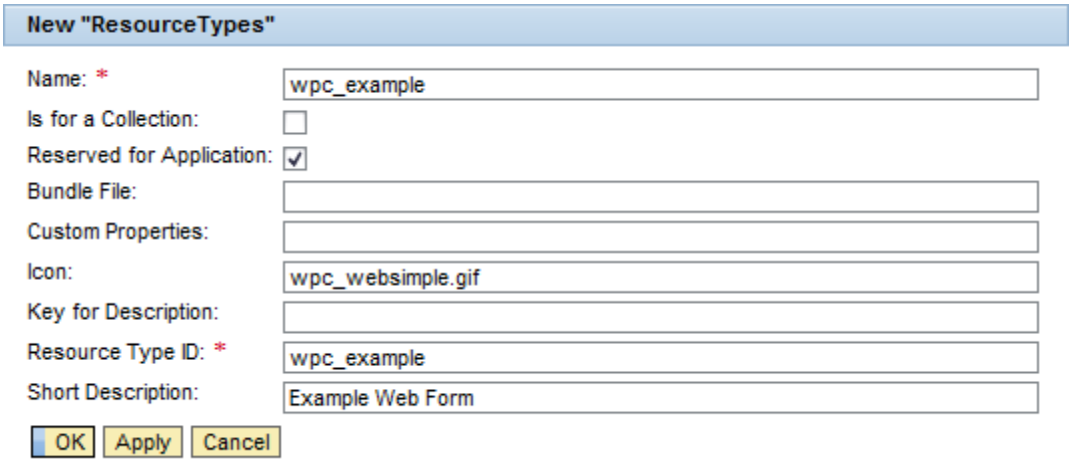
   Note

   Enter the same information in both parameters.
4. Activate the `Reserved for Application` parameter so that the Resource Type can only be used for Web Page Composer purposes.
5. Leave the fields `Bundle File` and `Key for Description` empty. (Chapter 5 of this guide explains how to configure your custom bundle files.)
6. Optional: Specify the image to be used as a symbol for the resource type.

You can optionally specify an image to be used as an icon for your Web form in the page container and for the corresponding content template. To do this, specify the file name (for example, `.<myicon>.gif)` and, if necessary, the subfolder name (for example, `<subfolder>/<myicon>.gif)` in the `Icon` parameter.

Note

The file must be stored in the folder `/etc/public/mimes/images` or in a subfolder thereof.



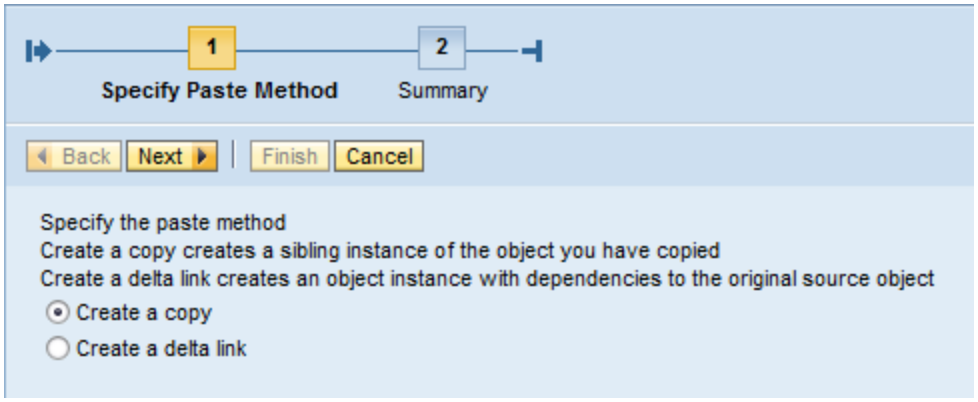
Note

The icon specified in the screenshot above is the default icon for the paragraph.

4.2.2 Create an iView for the new Web form

You create a Web content template iView. You use the Web content template iView when creating Web content in a page.

1. In the portal, choose *Content Administration* → *Portal Content Management* → *Portal Content.*
2. In the Portal Catalog, browse to *Portal Content* → *Templates* → *WPC Templates* → *iViews* → *WPC Content.*
3. In the WPC Content folder, create an iView for the Web Content Template of the new Web Content Type by copying an existing iView instance.
   a. Choose one of the standard iView instances (for example *Article*), right-click the object, and then choose *Copy*.
   b. Right-click the WPC Content folder, and from the context menu choose *Paste*. The *Copy* wizard appears.
   c. Choose *Create a copy* and then choose *Finish* to execute the paste action.

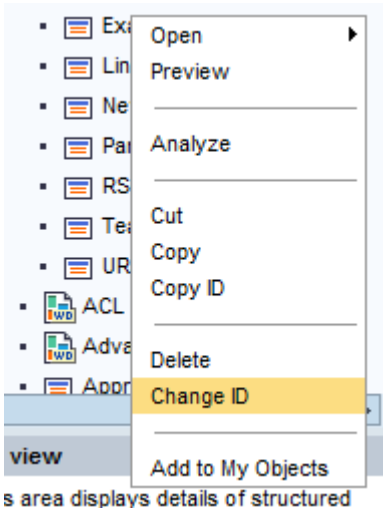


4. Configure the properties of the newly created iView.
   a. In the context menu of the iView you have created, choose *Open* → *Properties*.
   b. In the property editor choose *Properties* → *All*.
   c. Change the value of the following properties:

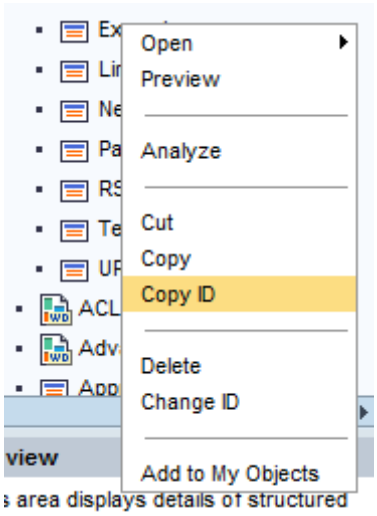
Name	Mandatory Change	Value
`com.sap.nw.wpc.WebContentType`	Yes	Enter the name of the Web resource type you created.
`com.sap.portal.ContentTemplate`	Yes	Enter the name of the Web resource type you created.
`name`	Yes	Enter a name for your iView. If you do not change the value of the property, the display name of the iView remains the same as the display name of the source object from which you created the copy.

   You do not need to change any other properties.
   d. Save your changes and close the property editor.

5. Optional: Change the ID of the iView according to your requirements.
   a. In the context menu of the iView, choose *Change ID*.
   b. Follow the on-screen instructions in the wizard. To apply the changes, choose *Finish*.



6. Copy the complete ID of the iView to your clipboard
   a. Right-click the iView and then choose *Copy ID*.



4.2.3 Create a new Web resource type

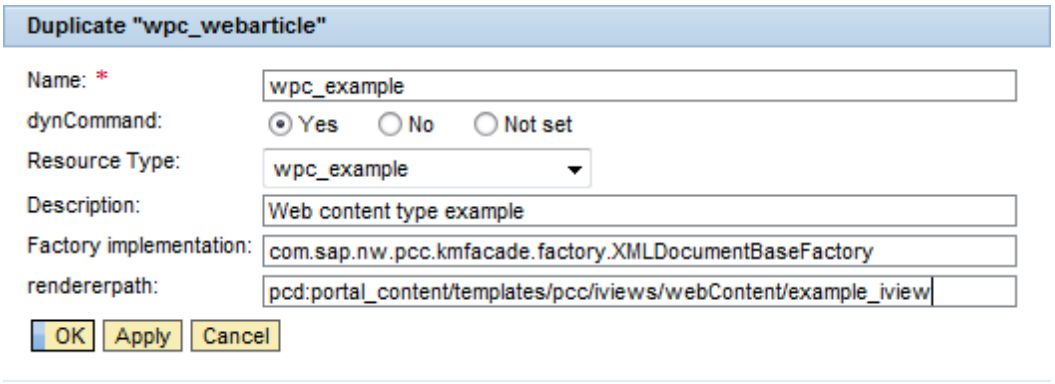
To assign the resource type, you require a separate Web resource type in the configuration of the Web Page Composer.

1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Web Content Types* → *Web Resource Type.*
2. To create the Web resource type for the page with predefined values, search for the *wpc_webarticle* instance, select it, and choose *Duplicate*.

   Make sure that the following values are entered in the following parameters:

Parameter	Value
`Name`	Change the name. Use the name of the Resource Type used in the

	previous step.
`Resource Type`	Specifies the assigned Resource Type. Choose the Resource Type you just created for the Web Form.
`Description`	Description of the Resource Type.
`Factory implementation`	Specifies the factory implementation required to create instances of this Web Resource Type. You must use `com.sap.nw.pcc.kmfacade.factory.XMLDocumentBaseFactory` for all Resource Types that rely on the default rendering mechanism (Web Resource Types that are base on an XML definition and use XSL transformation). Therefore, do not change the value in this parameter, unless you use a custom rendering mechanism.
`Renderpath`	Enter the complete ID of the Web Content Template iView of the new Web Content Type. Right-click the input field and choose paste.



4.2.4 Reference the XSL file in the stylesheet object

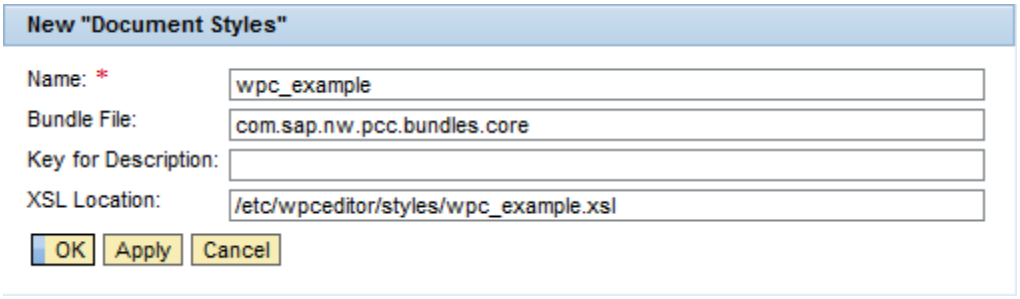
1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Editor* → *Stylesheets* → *StyleSheets* → *Document Styles.*

   Choose *New* to create a new stylesheet object and specify the path to the XSL file and the other parameters.The following table provides an overview of the parameters of the stylesheet object:

Parameter	Required
`Name`	Yes
`Bundle File`	No
`Key for Description`	No
`XSL Location`	No

   Keep the default entry for the `Bundle File` field. Once the language bundle is defined, default entry will be replaced with the new definition..

Leave the `Key for Description` field empty. Once the language bundle is defined, the default entry will be replaced with the new definition.

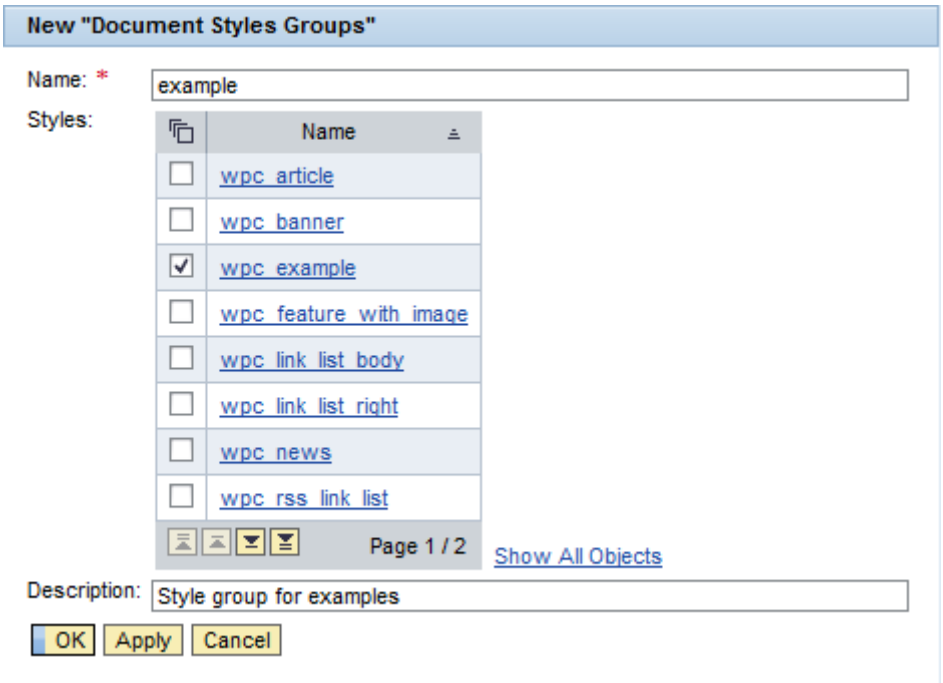


4.2.5 Create a new stylesheet group

For technical reasons, the stylesheet object must be entered in a stylesheet group.

1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Editor* → *Stylesheets* → *StyleSheet groups* → *Document Style Groups.*
2. Choose *New* to create a new stylesheet group, and then choose the stylesheet object that you have just created. Specify the other parameters:

Parameter	Required
`Name`	Yes
`Styles`	No
`Description`	No



4.2.6 Reference the XML file in the document type

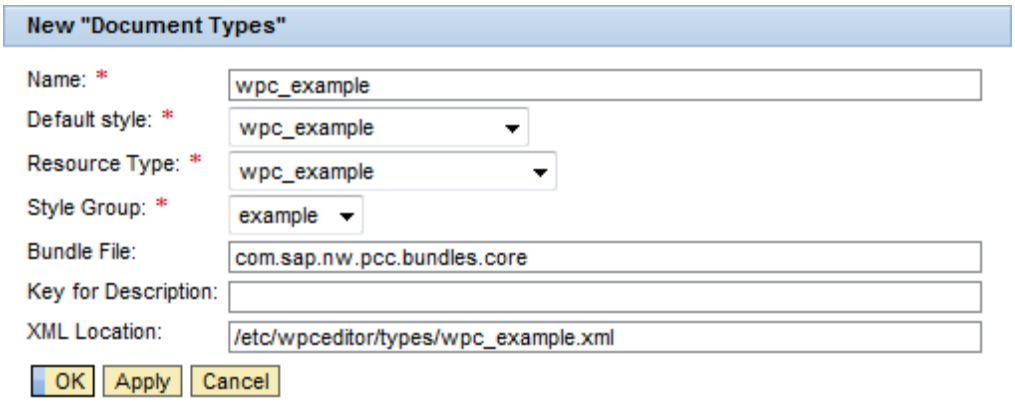
In addition to referencing the XSL file, you must also reference the XML file that describes the new Web form. Create a new document type for the Web form and specify the path to the XML file.

1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Editor* → *Document Types* → *Document Types.*
2. Choose *New* to create a new document type, and then choose the stylesheet group that you have just created. Specify the other parameters:

Parameter	Required
`Name`	Yes
`Default style`	Yes
`Resource Type`	Yes
`Style Group`	Yes
`Bundle File`	No
`Key for Description`	No
`XML Location`	No

Keep the default entry for the `Bundle File` field. Once the language bundle is defined, the default entry will be replaced.

Leave the `Key for Description` field empty. Once the language bundle is defined, the default entry will be replaced.



4.3 Reload editor configuration

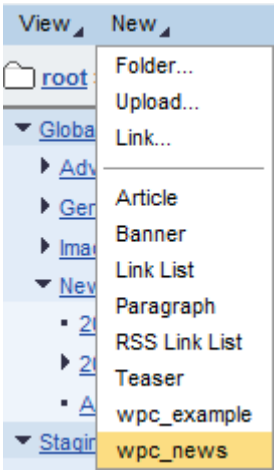
For the changes to take effect, you must reload editor configuration. This can be done by restarting the portal or by launching the *ReloadEditorConfigComponent* application via the following URL: <portal url>/irj/servlet/prt/portal/prtroot/com.sap.nw.wpc.designtime.ReloadEditorConfigComponent

4.4 Result

After you have completed all the steps, all users assigned to the WPC *Editor* role can use your new Web form to create Web content. This Web content can be used within Web Page Composer to build Web pages.

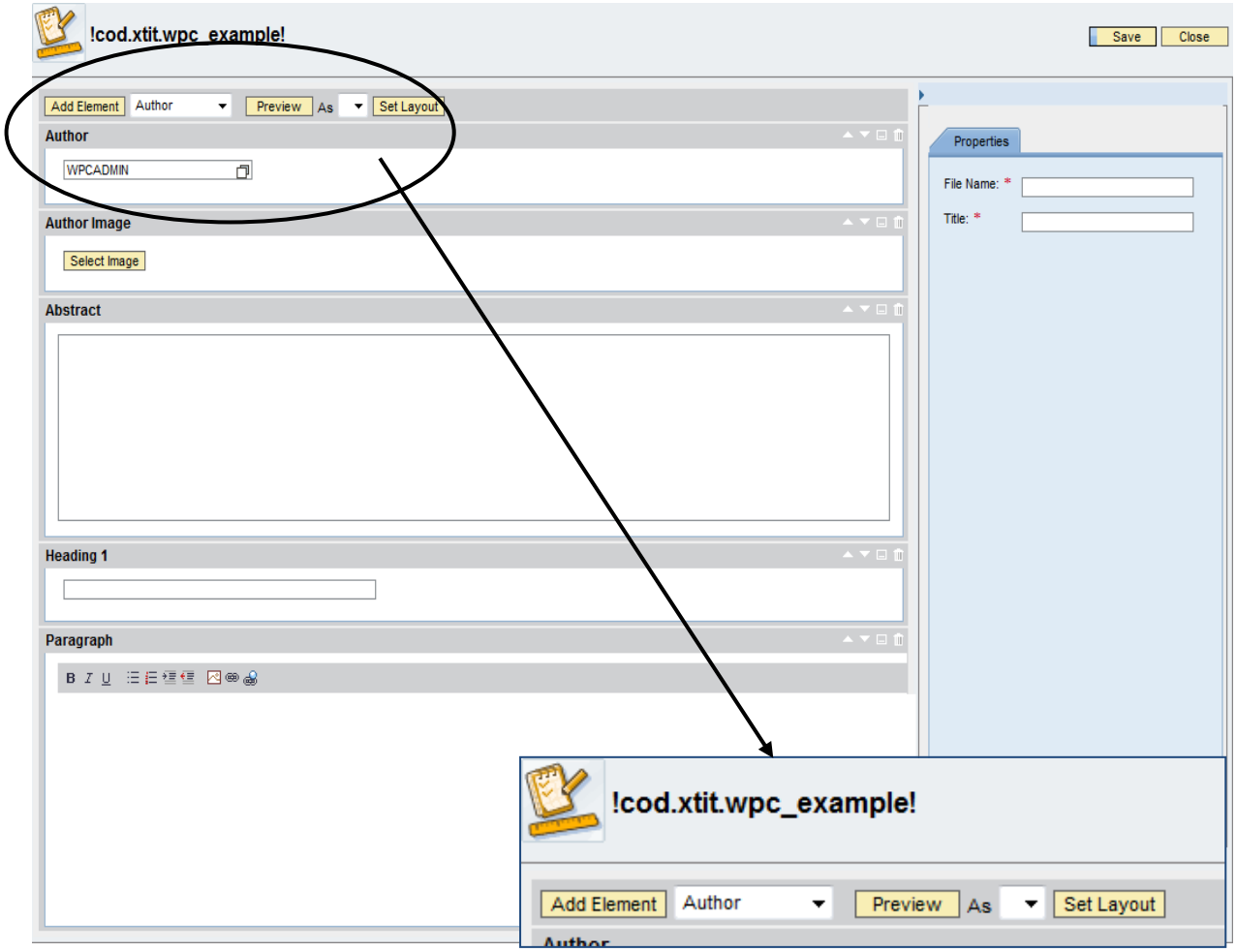
4.4.1 Creating Web content in Content Library

You can create Web content using the new Web form in the Content Library.



4.4.2 Creating Web content in Content Editor

You can also create Web content using the new Web form in the content editor.



When you first open the Web form it should look like the form above, with an empty entry for *Preview*. When you open it for a second time, the *Preview* field must contain the the entry: `cod.xlbl.wpc_example`.

5. Changing and Translating Labels for Web Content

For all new configuration objects required for creating and using a new Web form, you can specify bundle files that contain keys and labels for translation purposes.

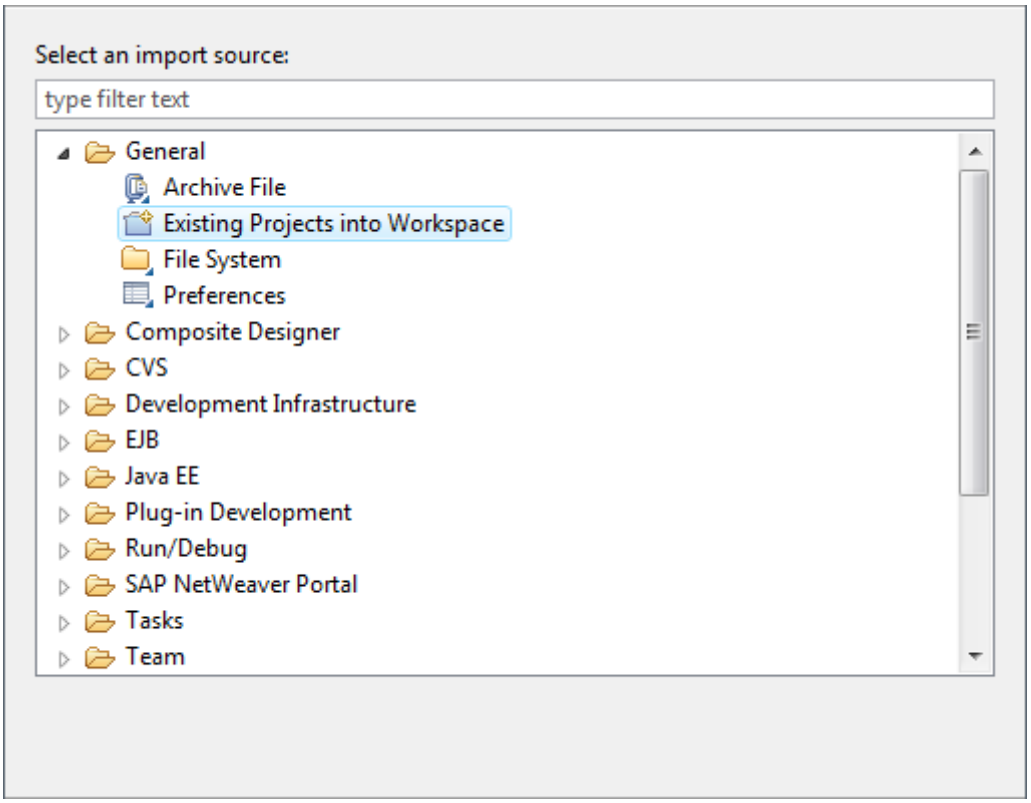
You use a bundle file specification to:

- Display the label of the new Web form commands.
- Display the labels of the new Web form.
- Display the label of the new Web form in the container description.

5.1 Import the bundle file project

To be able to implement your own bundle files for the Web Page Composer configuration, you must import the attached ZIP file into your SAP NetWeaver Developer Studio.

1. Extract the attached ZIP file to your file system.
2. In SAP NetWeaver Developer Studio, navigate to *File* → *Import*.
3. Choose *General* → *Existing Projects into Workspace.*



4. Choose *Next*.
5. Choose *Browse* and navigate to the directory where you have extracted the ZIP file.
6. Select the *CustomBundleProject* project, and then choose *Finish*. The project will be imported to your SAP NetWeaver Developer Studio.

5.2 Edit the bundle file

You can create several property files for language specific translations. In each property file you have to maintain a key/label pair for all configurations and labels for which you need a display name instead of a technical ID.

In each language bundle you maintain the value texts in the corresponding language. If you need an additional language specific bundle such as Spanish, you have to create the corresponding property file with the language suffix (for example, for Spanish, the suffix is *es*).

5.2.1 Edit the WebFormBundle.properties file

Let's start with the *WebFormBundle.properties* file. This is the bundle file that covers the default language which. In our example, the default language is English.

You first need to check which labels have to be maintained in the file

1. Navigate to *CustomBundleProject* → *src.api* → *com* → *customer.nw.km.wpc.bundle* → *WebFormBundle.properties*.
2. Open the file and maintain the keys.

   The following table lists the keys that needed to be configured:

Key	Description
`xml.xlbl.wpc_example`	This key will be used for displaying the label of the new Web form commands and for displaying the label of the new Web form in the container description.
`cod.xtit.wpc_example`	This is the key for the name of the Web form in the editor.
`cod.xlbl.wpc_example`	This is the key for the name of the Web form in the preview listing.

3. Enter a value for each of the keys.
4. If you plan to use your own keys for elements or properties in the XML file, you need to maintain the key/label pairs in your bundle property file additionally.

   EXAMPLE

   In our example the image element is displayed as *Author Image*.

   To rename this element to *Picture*, you need to maintain the. following entry:

   ```
   xml.xlbl.image=Picture
   ```

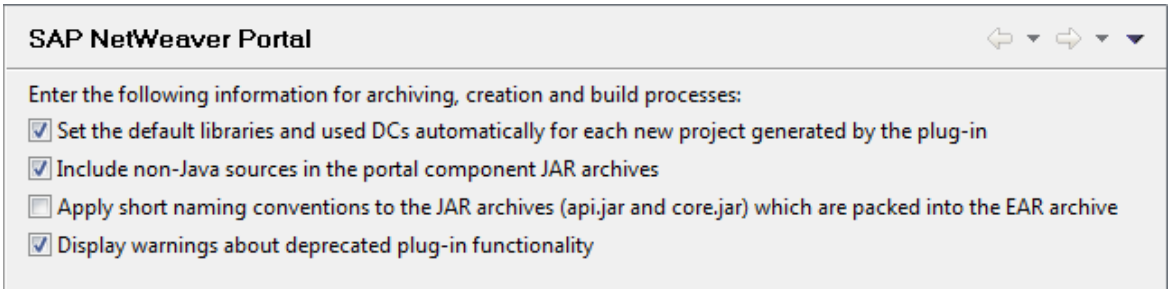
The property file should contain the following entries:

```
xml.xlbl.wpc_example=Example
cod.xtit.wpc_example=Example
```

```
cod.xlbl.wpc_example=Example
```

5.3 Configure the settings in SAP NetWeaver Developer Studio

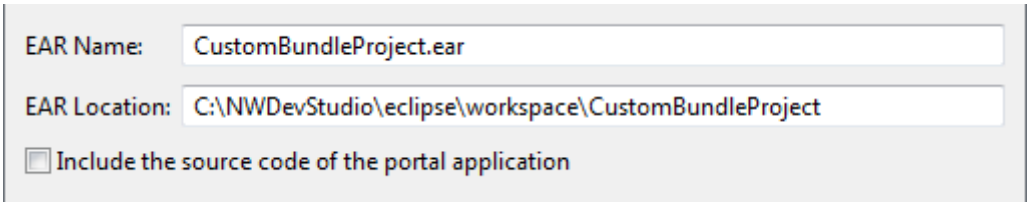
1. Navigate to *Window* → *Preferences* → *SAP NetWeaver Portal*.
2. Ensure that the following checkboxes are selected:



3. Choose *Apply*, and then choose *OK* to close the preferences editor.

5.4 Export the project as an EAR file and deploy it to the portal

1. In SAP NetWeaver Developer Studio, right-click your project's root folder *CustomBundleProject*, and then choose *Export*.
2. Navigate to *SAP NetWeaver Portal* → *EAR file,* and then choose *Next*.
3. From the list of projects, select *CustomBundleProject,* and then choose *Next*.
4. To export your EAR file, choose *Finish*.



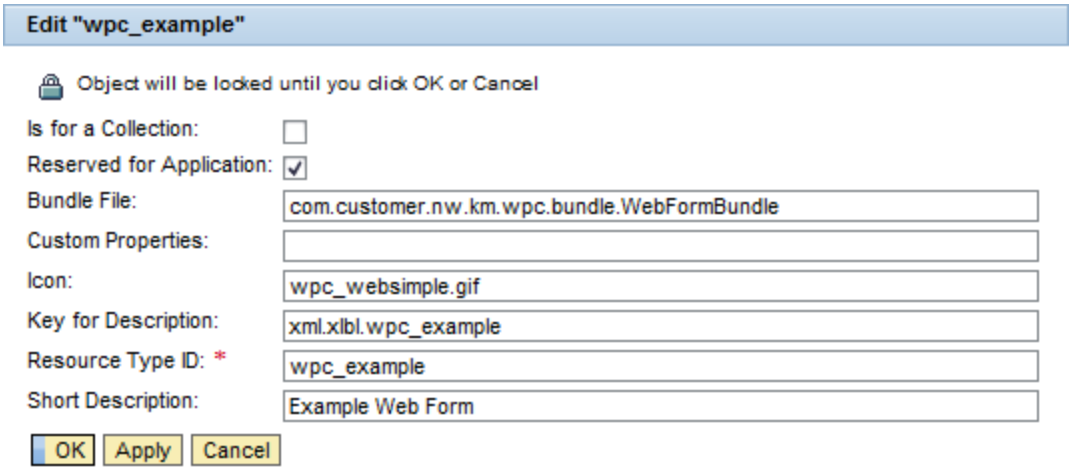
5. Deploy the EAR file to the portal. If you need assistance, contact your administrator.

5.5 Reference the bundle file in KM configuration

In the configuration of the following configuration objects, enter the name of the bundle file and specify the keys.

5.5.1 Reference the bundle file in the resource type

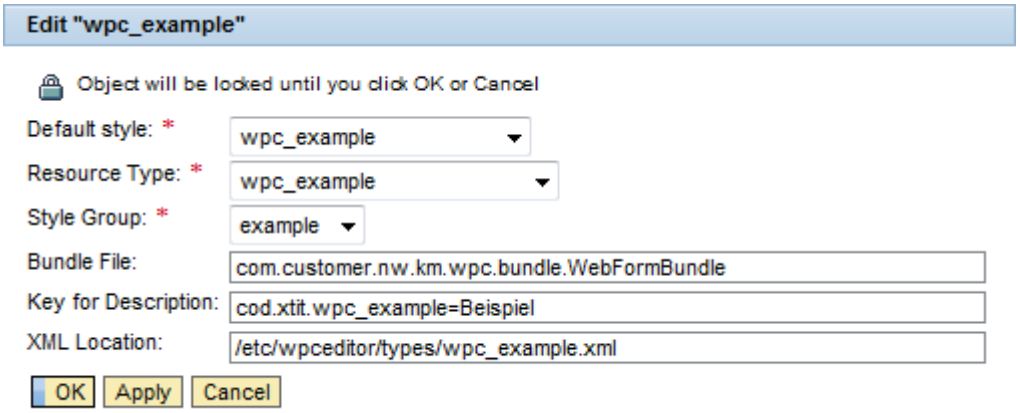
1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Global Services* → *Resource Types* → *Resource Types*.
2. Filter the newly created *wpc_example* resource type.
3. Open *wpc_example* for editing and add the following configuration:
   - `Bundle File`: **`com.customer.nw.km.wpc.bundle.WebFormBundle`**
   - `Key for Description`: **`xml.xlbl.wpc_example`**



4. To save your changes, choose *OK*.

5.5.2 Reference the bundle file in the document types

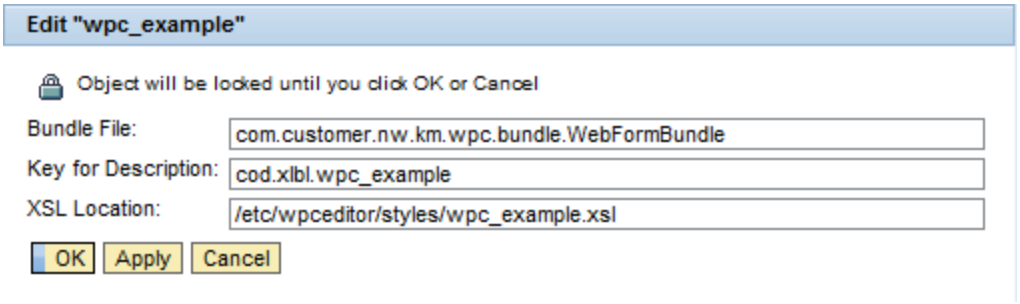
1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Editor* → *Document Types* → *Document Types.*
2. Filter for the newly created *wpc_example* document type.
3. Open *wpc_example* for editing and add the following configuration:
   - `Bundle File`: **`com.customer.nw.km.wpc.bundle.WebFormBundle`**
   - `Key for Description`: **`cod.xtit.wpc_example`**



4. To save your changes, choose *OK*.

5.5.3 Reference the bundle file in the stylesheets

1. In the portal, navigate to *System Administration* → *System Configuration* → *Knowledge Management* → *Content Management* → *Web Page Composer* → *Editor* → *Stylesheet* → *StyleSheets* → *Document Styles.*
2. Filter for the newly created *wpc_example* document style
3. Open *wpc_example* for editing and add the following configuration:
   - `Bundle File`: **`com.customer.nw.km.wpc.bundle.WebFormBundle`**
   - `Key for Description`: **`cod.xlbl.wpc_example`**



4. Choose *OK* for saving the changes.

5.6 Restart the portal

In order for the changes to take effect, you must restart the portal.

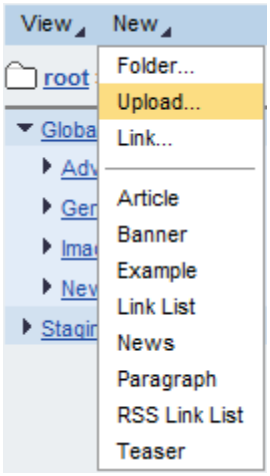
Important

It is essential to restart the whole portal. Otherwise the complete changes will not take effect.

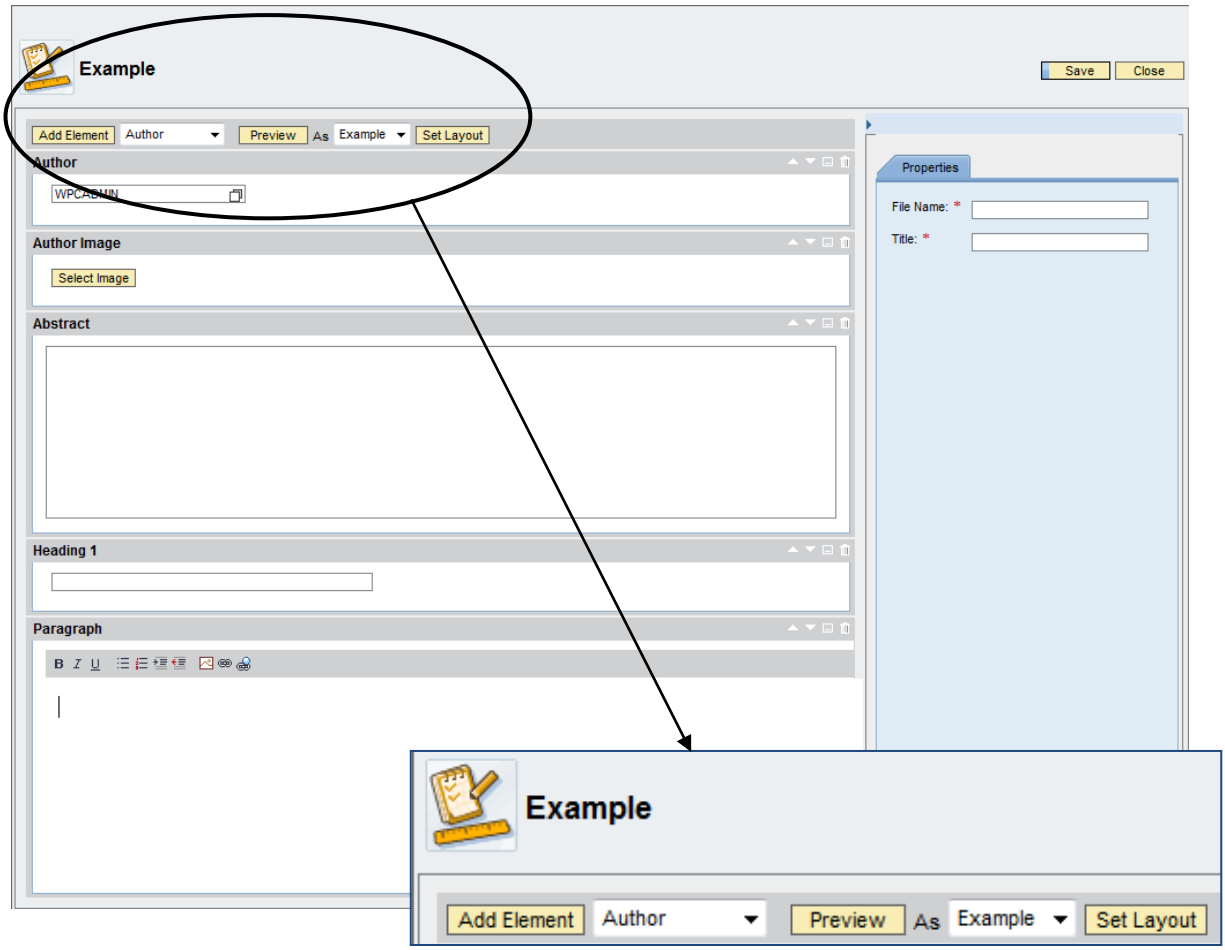
5.7 Result

After you have completed all the steps, all users assigned to the *WPC Editor* role can use your new Web form to create Web content. This Web content can be used within Web Page Composer to build Web pages.

5.7.1 Creating Web content in Content Library



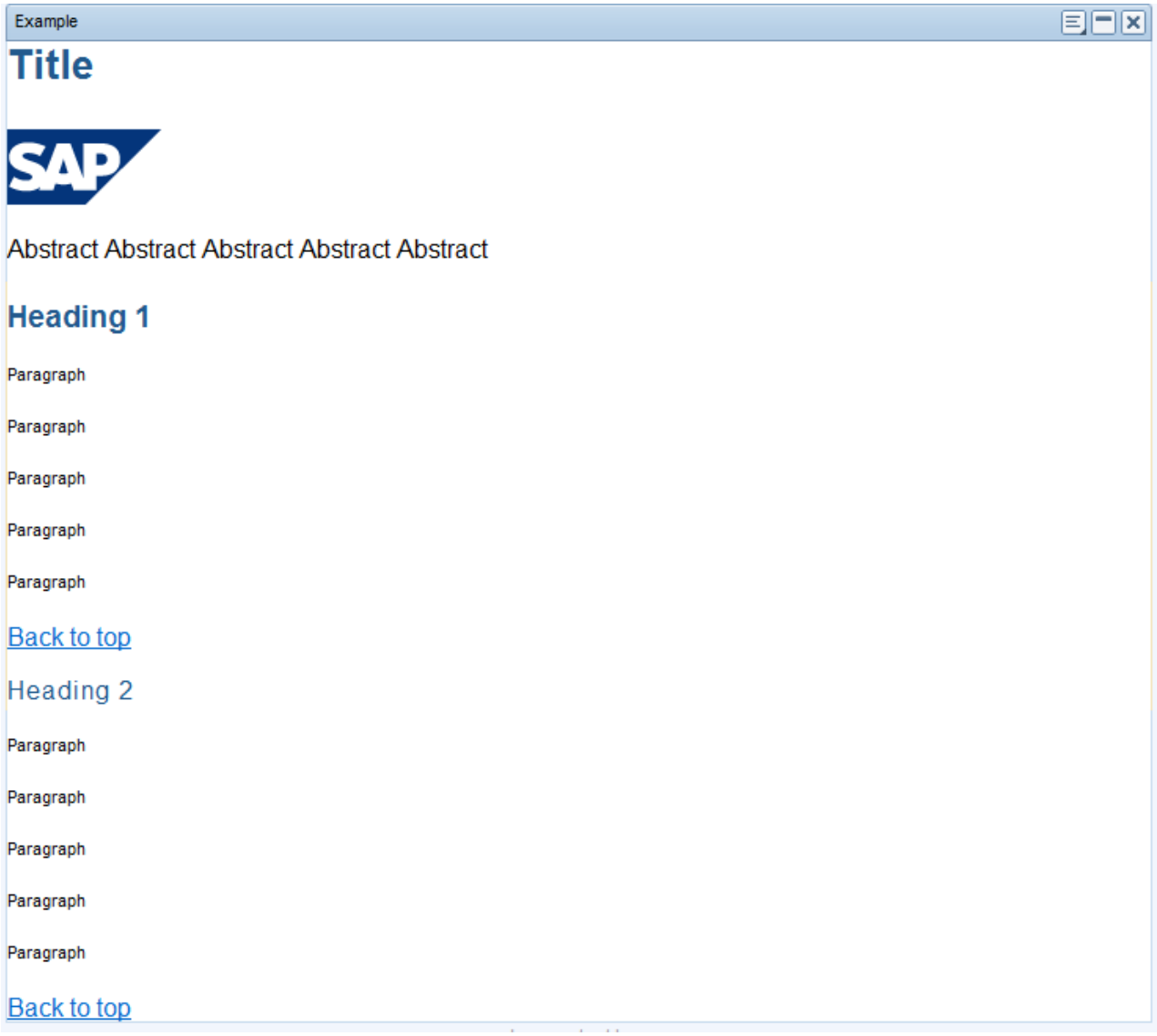
5.7.2 Creating Web content in Content Editor



6. Result

After you have completed the configuration steps, you can create new Web content with the new Web form.

After you have edited the iView, the Web content item should look like the one in the screenshot below:



7. Appendix

Appendix A – List of abbreviations

Abbreviation	Description
KM	Knowledge Management
EAR	Enterprise Application Archive
NWDS	SAP NetWeaver Developer Studio

www.sdn.sap.com/irj/sdn/howtoguides