Creating Your First XSLT Functions

2017-02-15

Orckestra, Europe
Nygårdsvej 16
DK-2100 Copenhagen
Phone +45 3915 7600

www.orckestra.com

Contents

1 INTRODUCTION 3

1.1 Only little technical background required 3
1.2 Only want the functionality? Get the add-on 3
1.3 Read to learn and create by yourselves 3

2 YOUTUBE FUNCTION 4

2.1 What is it we will make? 4
2.2 How to embed a YouTube video 6
2.2.1 Using the YouTube code from C1 CMS 7
2.3 Your first XSLT function 9
2.3.1 Naming functions 10
2.3.2 Creating an XSLT function 10
2.3.3 Smaller steps: inserting a new function 12
2.4 Rendering YouTube code 13
2.4.1 Learning the basics of XSLT 17
2.4.2 Getting the video code variable with XSLT 17
2.4.3 Specifying input parameters for a function 19
2.4.4 Document your functions 19
2.4.5 Generate documentation on functions 21
2.5 More parameters 21
2.5.1 Default value parameter 21
2.5.2 More uses of an XSLT function without XML 22
2.5.3 Templates and XHTML includes 22

3 VIMEO 23

4 WHAT'S MORE? 25

4.1 Integrating with datatypes 25
4.2 Possible enhancements: full URL, full embed code 25
4.3 And even more? 25

1 Introduction

This document is aimed at people who create very simple XSLT functions in C1 CMS, maybe even their first XSLT function! This could be a super user, or a very experienced editor with some technical background (lots of HTML experience).

Typically, it is a front-end developer just starting to work with C1 CMS or a developer who just starts with C1 CMS.

This document is written quite extensively to give this broad range of personas the full story on how to do this. The more experienced might even need to read very little text and just look at the screenshots.

We will create a function in C1 CMS allowing users to easily add external internet functionalities to their website with C1 CMS. In this case, we will be adding functionality to show **YouTube** and **Vimeo videos embedded** on a C1 CMS site. This means that these videos will play on your website. For an editor, it is just as simple as to add a function to the page and supply the video code, presto.

For you as the creator of the function, it is not even that much harder.

If you want to learn, read on; otherwise, just get the add-on.

1.1 Only little technical background required

To do this by yourselves, you will **only need a little technical background; you don't need to be a programmer**:

- You can read more complex HTML.
- You have seen XSLT code before and managed to understand the very basics of XSLT.
- Oh, and of course, you have access to C1 CMS and are allowed to create functions in there (which is just a matter of security in the manager).

1.2 Only want the functionality? Get the add-on

This functionality will also be available as a add-on; you do not have to do the manual labor of creating it by yourselves. If you simply want to get this on your site, drop the document and find the add-on on the package server.

1.3 Read to learn and create by yourselves

Good you're still here! I will explain how to create the functions yourselves. I'm sure you can use this knowledge to create similar functions for other functionalities!

In this case, we will create an XSLT function. You can also create a function with similar functionality using .NET user control. In our case, XSLT is easier.

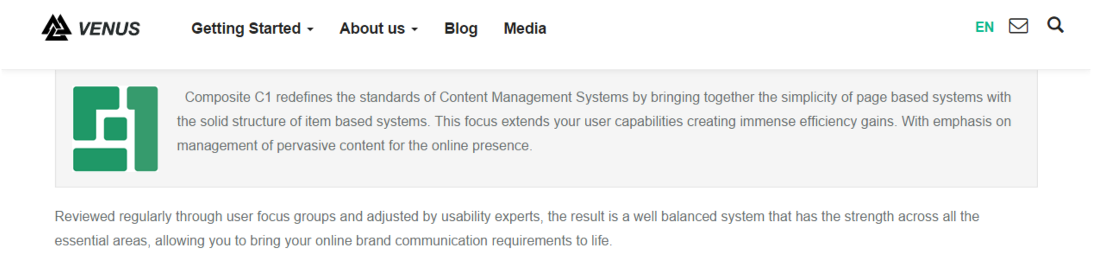
Since we will make an XSLT function, a little XSLT experience is required. But don't fear if you don't know much XSLT. You will only need the very basics to understand and change it.

2 YouTube function

Let's see what we want to create and then make it. Simple steps at a time; this might just be the first XSLT function you are creating.

2.1 What is it we will make?

Our aim here is what you see in the screenshot of a page on my C1 CMS demo site. This part of the template shows two content areas. The one area has pure content…



Composite C1 redefines the standards of Content Management Systems by bringing together the simplicity of page based systems with the solid structure of item based systems. This focus extends your user capabilities creating immense efficiency gains. With emphasis on management of pervasive content for the online presence.

Reviewed regularly through user focus groups and adjusted by usability experts, the result is a well balanced system that has the strength across all the essential areas, allowing you to bring your online brand communication requirements to life.

Figure 1: Showing pure content

…while the other one shows the YouTube video.

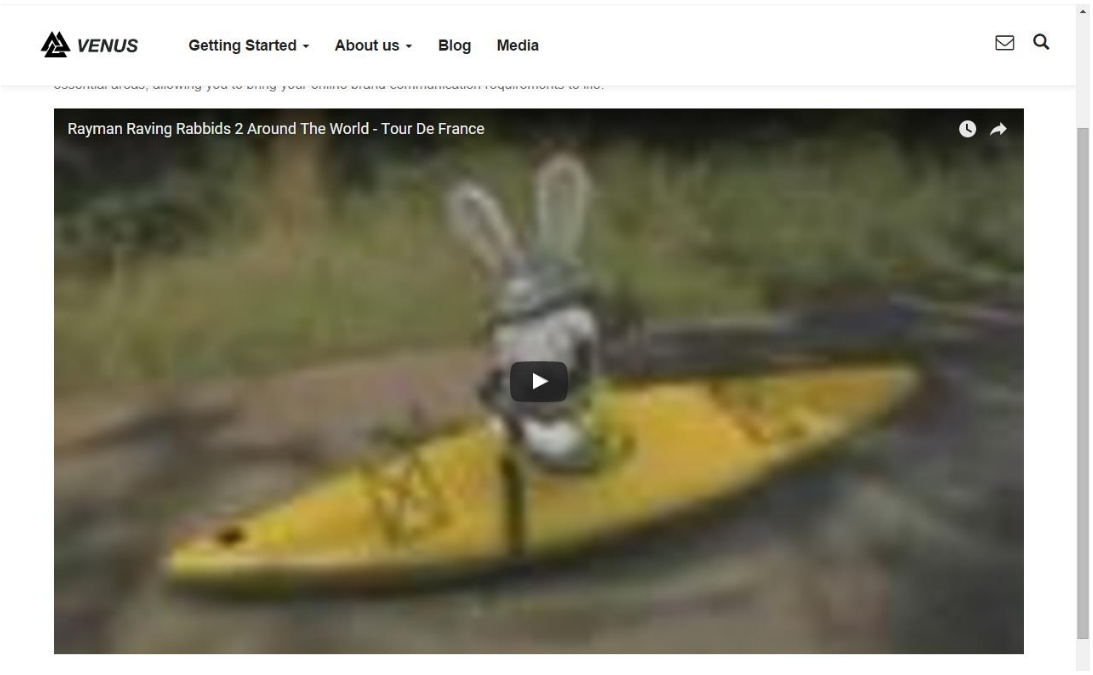
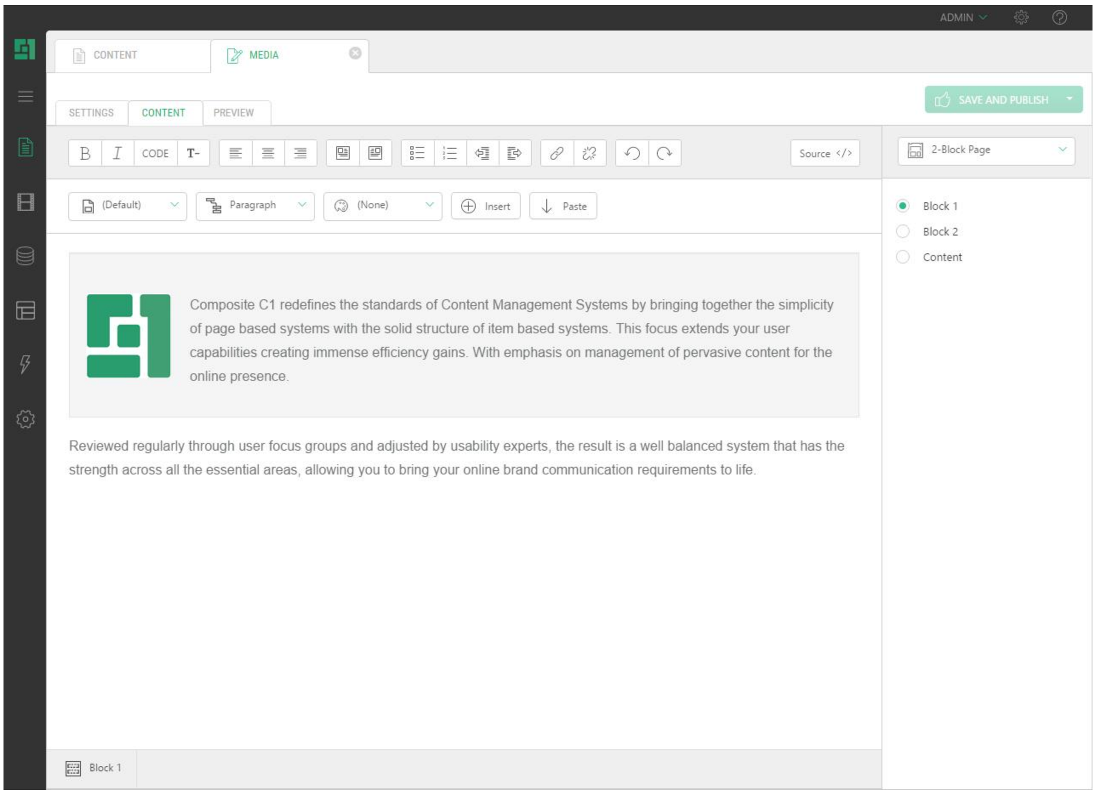


Figure 2: Showing a YouTube video inline

So we want editors to be able to add these to their page. We need to make it simple for them to insert these content elements anywhere they want to. If you don't want the editors to have that much freedom in where to insert and play videos on the pages, you can consider using datatypes - a little more on that later!

Below is the same page in the manager: the template with three content areas the editor can supply content for.



Composite C1 redefines the standards of Content Management Systems by bringing together the simplicity of page based systems with the solid structure of item based systems. This focus extends your user capabilities creating immense efficiency gains. With emphasis on management of pervasive content for the online presence.

Reviewed regularly through user focus groups and adjusted by usability experts, the result is a well balanced system that has the strength across all the essential areas, allowing you to bring your online brand communication requirements to life.

Figure 3: Block 1 contains nothing but HTML

And Block 2 has the function inserted to play the YouTube video. This is what you will be able to create in a few minutes after you study this document!

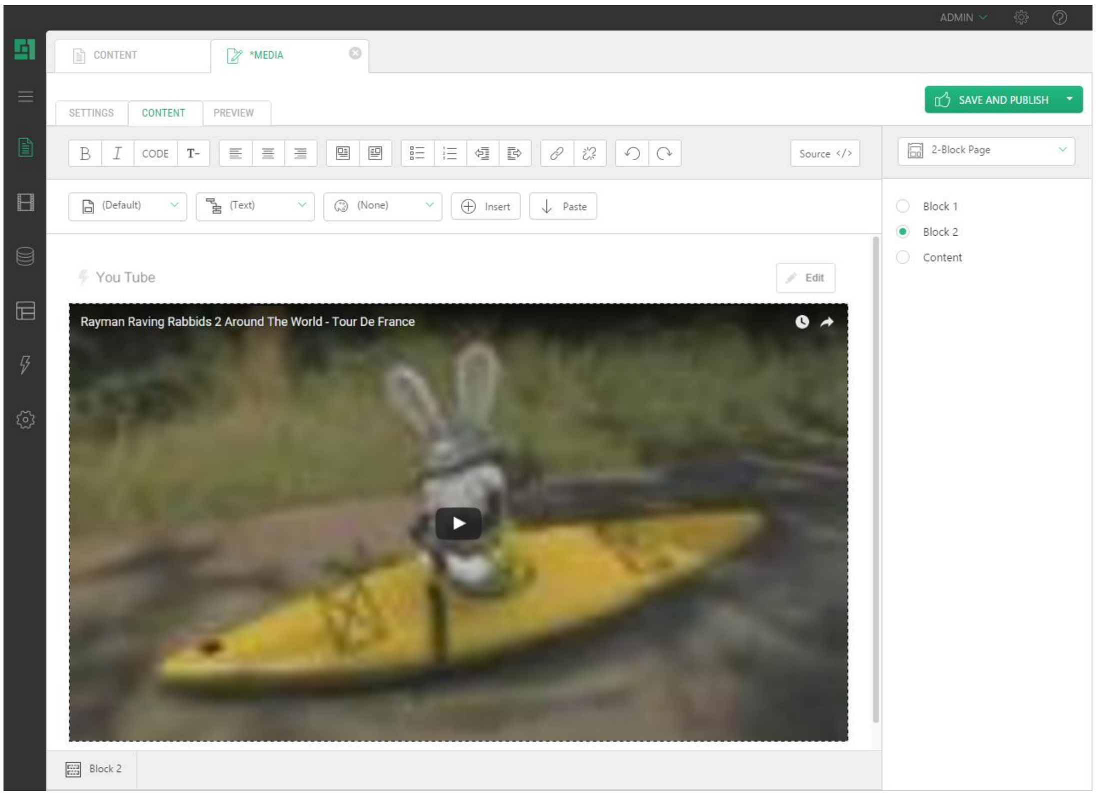


Figure 4: Block 2 includes a YouTube video

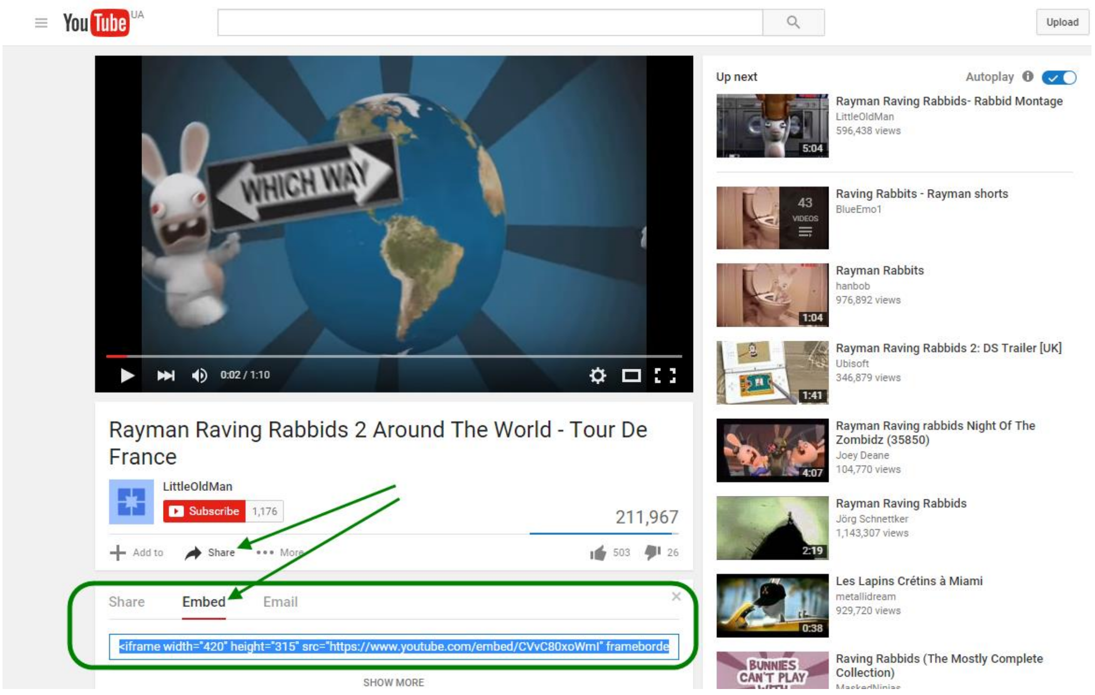
***Note**: In my design the blocks have a fixed height; that is why the content below the function is not visible. I have just left it so in here to make clear that this function can just go anywhere on a page between content elements etc.*

If you do the preview, you will see the video again as in the first screenshot.

2.2 How to embed a YouTube video

There are two ways to go with YouTube videos: link or embed. We want to play the video on our site, not just link to it. (An editor can do that just by supplying a hyperlink but that is boring.) So visit the YouTube website and find the code to embed a video on your site.

On the YouTube site, when having found a video, now get the embed code.



YouTube UA
Upload
WHICH WAY
0:02 / 1:10
Rayman Raving Rabbids 2 Around The World - Tour De France
LittleOldMan
Subscribe 1,176
211,967
Add to
Share
More
503
26
Share
Embed
Email
SHOW MORE
Up next
Autoplay
Rayman Raving Rabbids- Rabbid Montage
Raving Rabbits - Rayman shorts
Rayman Rabbits
Rayman Raving Rabbids 2: DS Trailer [UK]
Rayman Raving rabbids Night Of The Zombidz (35850)
Rayman Raving Rabbids
Les Lapins Crétins à Miami
Raving Rabbids (The Mostly Complete Collection)

```
<iframe width="420" height="315" src="https://www.youtube.com/embed/CVvC80xoWmI" frameborde
```

Figure 5: Video on YouTube

Here is the code copied straight from the site:

```
<object width="425" height="344"><param name="movie"
value="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"></param><pa
ram name="allowFullScreen" value="true"></param><param
name="allowscriptaccess" value="always"></param><embed
src="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"
type="application/x-shockwave-flash" allowscriptaccess="always"
allowfullscreen="true" width="425" height="344"></embed></object>
```

Listing 1: The Embed code from the YouTube website

Have a look at the code: which code is standard and which differs per video? If you're not sure, get another video, get its embed code and compare the two.

Let's beautify the code a bit for readability:

```
<object width="425" height="344">

  <param
  name="movie"
  value="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"></param>
  <param name="allowFullScreen" value="true"></param>

  <param name="allowscriptaccess" value="always"></param>

  <embed
  src="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"
  type="application/x-shockwave-flash"
  allowscriptaccess="always"
  allowfullscreen="true"
  width="425"
  height="344">
  </embed>

</object>
```

Listing 2: The same code formatted for readability

We can see that it has an <object> element with some parameters and an <embed> element.

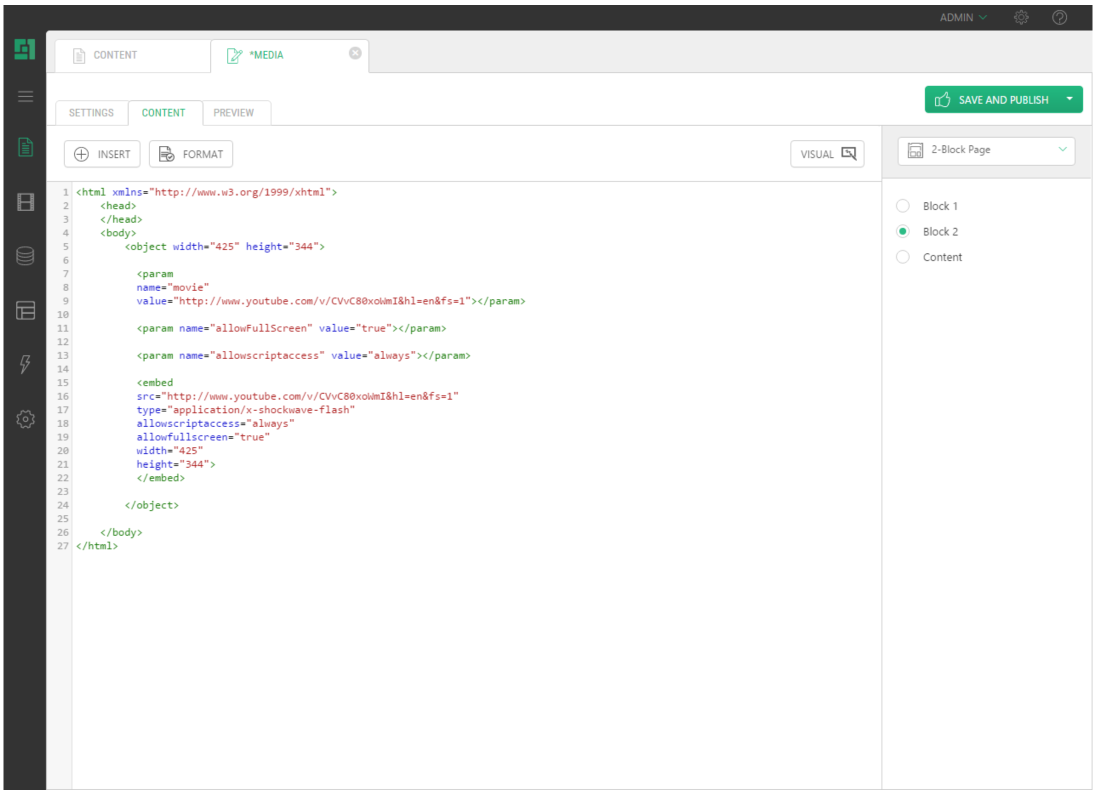
It looks duplicated, which it is. This is how it takes care of different browsers.

2.2.1 Using the YouTube code from C1 CMS

Let's first copy this code to a blank page in C1 CMS. Be sure not to copy the word stuff into your browser. This is code not content.

If you copy and paste it, or better insert as the Word, it will come out as text. What we need is it keeping HTML tags, not content on a page. Therefore, we need to paste it in as code.

- Select to the Content perspective in C1 CMS
- Create a new page or edit an existing page
- Select the Content tab
- Switch to the Source code view (the Source button right on top of Visual editor)
- Paste the HTML in

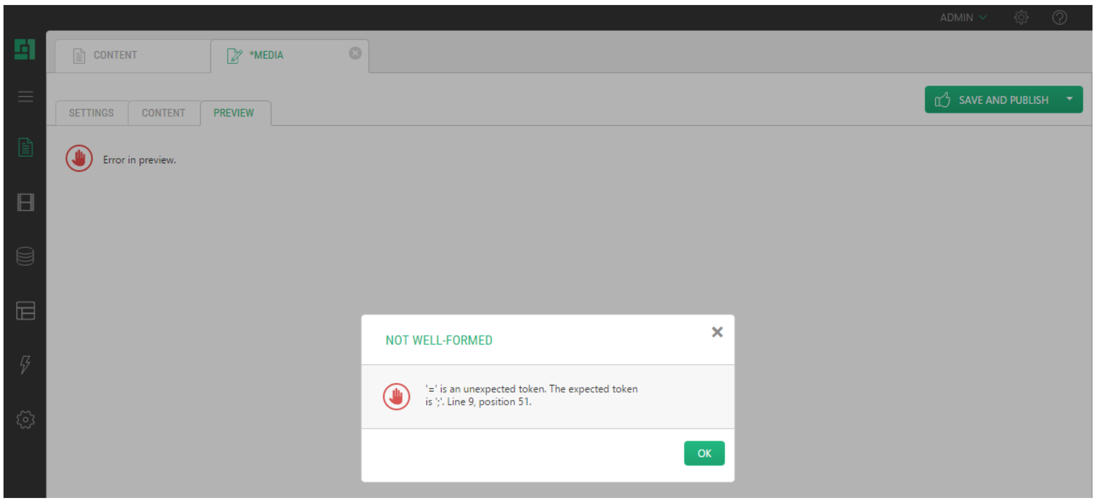


```
 1 <html xmlns="http://www.w3.org/1999/xhtml">
 2     <head>
 3     </head>
 4     <body>
 5         <object width="425" height="344">
 6 
 7           <param
 8           name="movie"
 9           value="http://www.youtube.com/v/CVvC80xoWmI&hl=en&fs=1"></param>
10 
11           <param name="allowFullScreen" value="true"></param>
12 
13           <param name="allowscriptaccess" value="always"></param>
14 
15           <embed
16           src="http://www.youtube.com/v/CVvC80xoWmI&hl=en&fs=1"
17           type="application/x-shockwave-flash"
18           allowscriptaccess="always"
19           allowfullscreen="true"
20           width="425"
21           height="344">
22           </embed>
23 
24         </object>
25 
26     </body>
27 </html>
```

Figure 6: HTML pasted in the Source view

Now click the Preview tab. Oh no! An error… Too bad it does not work!

Don’t be put off immediately. I agree that they are nasty and can be really cryptic but always try to find something in that error message!



Error in preview.

NOT WELL-FORMED

'=' is an unexpected token. The expected token is ';'. Line 9, position 51.

Figure 7: An error displayed in the Preview

It is complaining about ‘=’ at Line 4 Position 60 where it expects a ‘;’. Let’s see if we can fix it there. At Line 4 we see a lot of equal signs. Position 60 must be somewhere at the end:

value="http://www.youtube.com/v/CVvC80xoWmI&hl=en&fs=1

There we see “CVvC80xoWmI&hl=en&fs=1”. Well this is “&hl=” it doesn’t like.

It expects something like “&code;” and it reads something like “&code=” which is not allowed in XHTML, and C1 CMS validates your output for that and, therefore, complains.

In order to ensure validity of strings like that, ampersands must be expressed themselves as an entity reference, i.e. "&amp;" For example:

value="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1

***Note**: Please refer to Using Ampersands in Attribute Values (and Elsewhere) for more information.*

The strange string of characters just before the ampersand “CVvC80xoWmI” is actually the unique YouTube video code. If you look at different <embed> elements, you will see that this one changes for each video. If you change anything there you will see either no video or another one. It is the unique way to specify which video you want to see. That is exactly what we need later on, and it is what we will have the editors specify. So leave that code intact for now.

Usually you will just try to take out pieces that cause problems and see if you can still get it to work. So I took out the “&hl=en&fs=1” part and then it started complaining about another line.

Yes, indeed, there you find the similar one in the <src> value of the <embed> element. Clear them both out and the preview works now! It plays your video (just give it a little bit of time to load it from YouTube)

I am leaving you with the following working code:

```
<object width="425" height="344">
<param
name="movie" value="http://www.youtube.com/v/CVvC80xoWmI"></param>
<param name="allowFullScreen" value="true"></param>
<param name="allowscriptaccess" value="always"></param>
<embed
src="http://www.youtube.com/v/CVvC80xoWmI" type="application/x-shockwave-
flash"
allowscriptaccess="always"
allowfullscreen="true"
width="425"
height="344">
</embed>
</object>
```

Listing 3: The modified code that works now

This code works, and to play another video, you have to change that video code. So we have found the main variable part in here: “CVvC80xoWmI”. So if we want to play a YouTube video, we need to put in all this HTML code and the code of the video we want to see.

To put in some HTML code on a page you can use an XSLT function. This is what we will make now.

To put in an input variable for the video code we can create an input parameter in this function.

2.3 Your first XSLT function

You’re still here, not feared away by that complex error, good! It will not get any more complex than that!

We will now create a XSLT function, which will write some HTML, the HTML we have just managed to get correctly rendered by C1 CMS with the page preview.

To create a new function, select the Functions perspective.

There you can see the list of functions available on your site. It might be empty or list a whole lot of them depending on your site implementation and add-ons installed. I have a bunch of them.

Now you will create the function by right clicking in the XSLT Function part of the Functions navigator and clicking Add Function in the context menu.

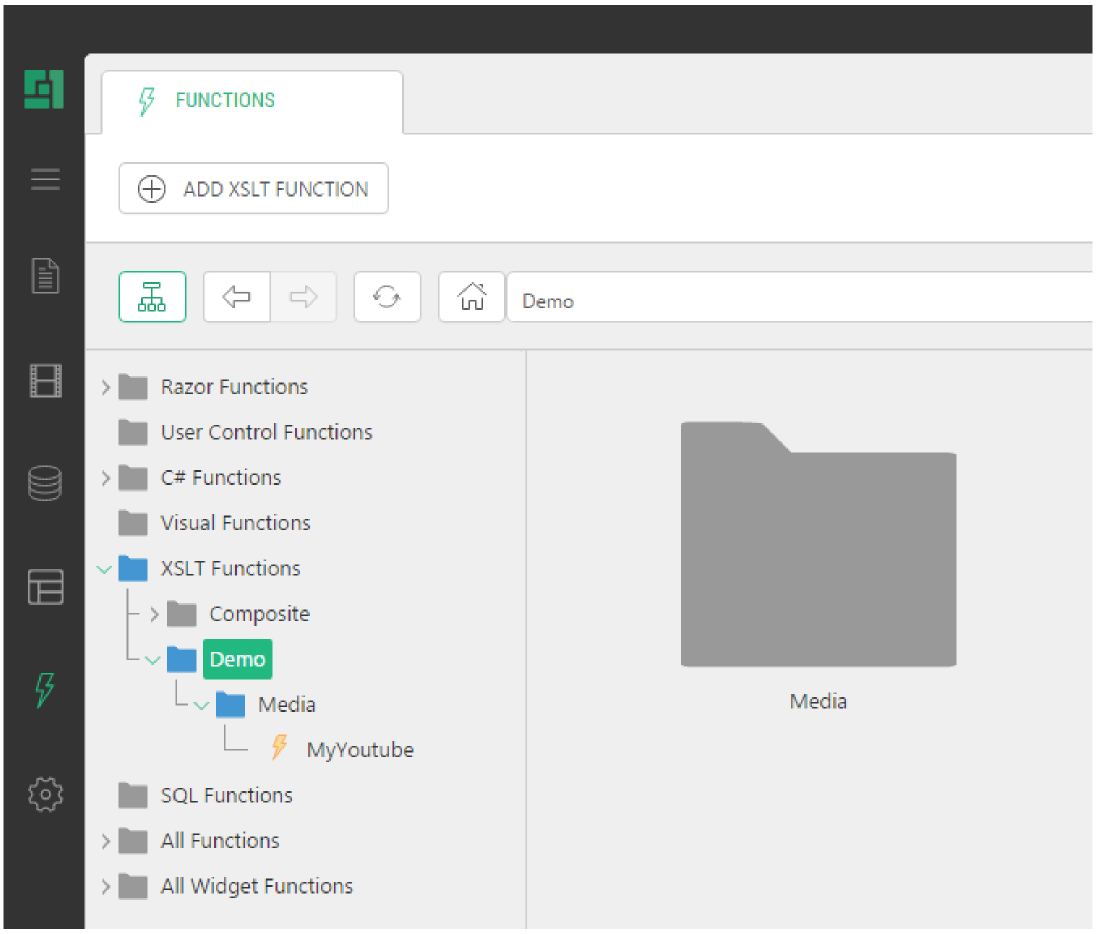


Figure 8: About to add an XSLT function

2.3.1 Naming functions

Think well about the name for a function! When editors will want to add a YouTube video, they must insert the function from this list of functions. Giving a good name will make the function easier to find and group with similar functions.

You might consider using different namespaces. If you are creating the function as a partner, you might have a namespace of “Partner name functions”, or for a specific customer - “Customer name functions”. You can see that C1 CMS uses its own namespace as well: all the add-ons are under within the “Composite” namespace, as well as items in other perspectives such as “Media”. This is important; but for now we will not go deeper into this.

2.3.2 Creating an XSLT function

To create a new function:

1. Select the Functions perspective
2. Select the XSLT functions in the navigator’s tree
3. Click Add Function

4. Specify the name
5. Specify the namespace
6. Click OK

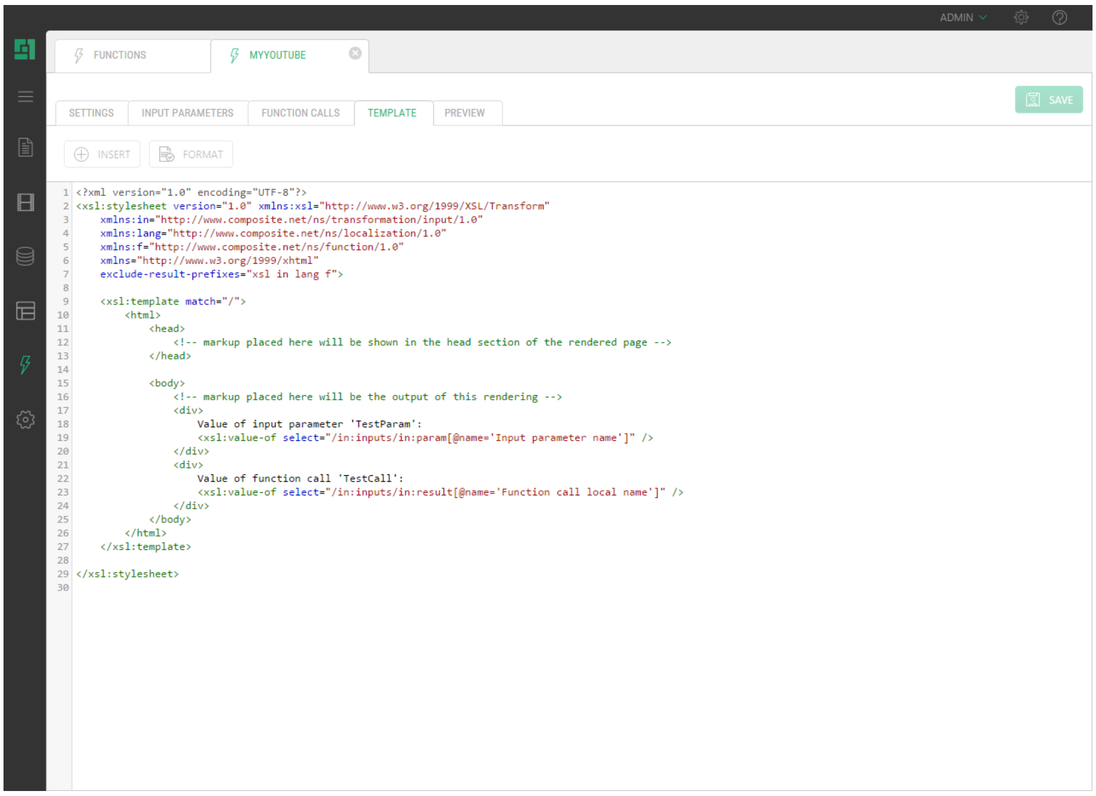
I named the function MyYoutube and entered “Demo.Media” as the namespace. Meaning you will have a namespace called Media below the already existing Demo namespace.

***Note**: namespaces are case-sensitive, so if you type “demo.media”, you will have two “demo” namespaces: “Demo” and “demo”, which might not be your idea of ordering functions.)*

Output type XHTML

Leave the output type as XHTML, so that we will have XHTML generated. The other choice is XML. Putting it on XHTML will make the function available in the Insert Function list of Visual Editor. In most cases, you will be creating XHTML functions.

When going much more advanced on, you might want to create XML that is used in other functions; in that case, you’ll have to change the output type.



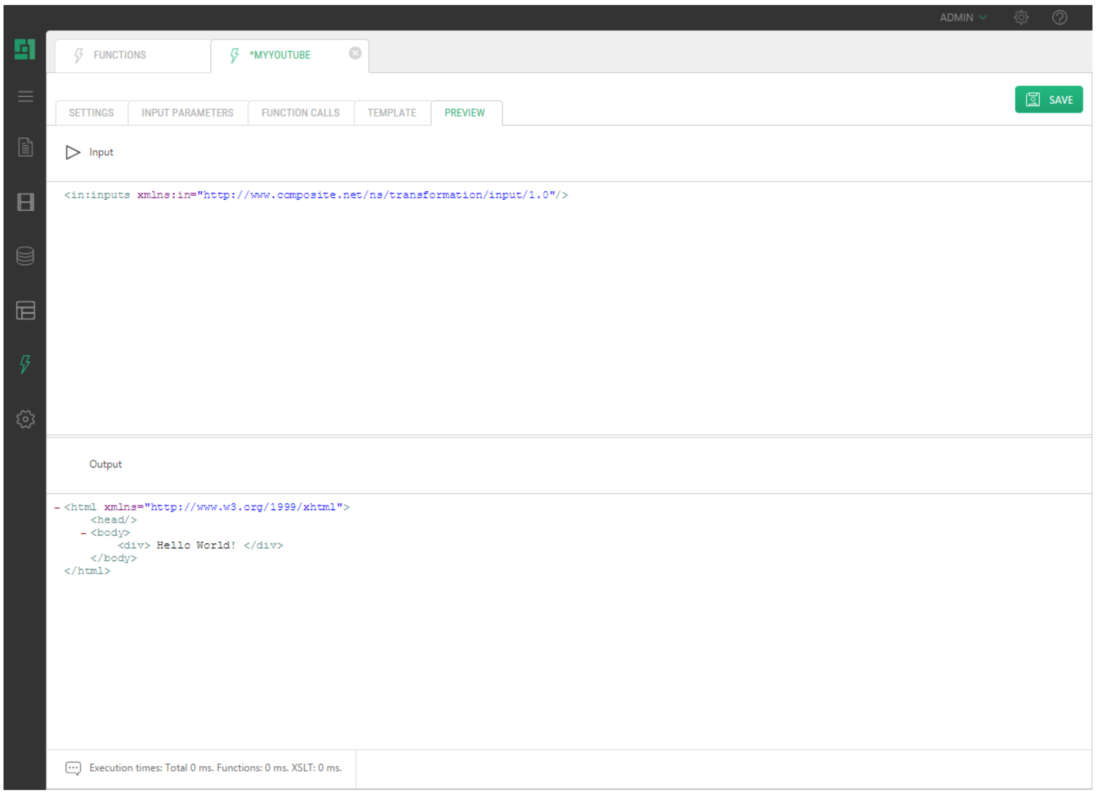
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:in="http://www.composite.net/ns/transformation/input/1.0"
    xmlns:lang="http://www.composite.net/ns/localization/1.0"
    xmlns:f="http://www.composite.net/ns/function/1.0"
    xmlns="http://www.w3.org/1999/xhtml"
    exclude-result-prefixes="xsl in lang f">

    <xsl:template match="/">
        <html>
            <head>
                <!-- markup placed here will be shown in the head section of the rendered page -->
            </head>

            <body>
                <!-- markup placed here will be the output of this rendering -->
                <div>
                    Value of input parameter 'TestParam':
                    <xsl:value-of select="/in:inputs/in:param[@name='Input parameter name']" />
                </div>
                <div>
                    Value of function call 'TestCall':
                    <xsl:value-of select="/in:inputs/in:result[@name='Function call local name']" />
                </div>
            </body>
        </html>
    </xsl:template>

</xsl:stylesheet>
```

Figure 9: Editing an XSLT function

The function is created. The namespace now shows in the function tree.

An XSLT function editor view consists of several tabs. It opens on the Template tab, which shows the XSLT. This is where you specify the output of the function. Here you will add the XHTML and any XSLT needed to render XHTML. We will get to the other tabs later.

Let’s start simple, with “hello world”. You can replace the entire inner <body> section with some of your html. Click on the Preview tab to see the result:



```
<in:inputs xmlns:in="http://www.composite.net/ns/transformation/input/1.0"/>
```

```
- <html xmlns="http://www.w3.org/1999/xhtml">
      <head/>
    - <body>
          <div> Hello World! </div>
      </body>
  </html>
```

Execution times: Total 0 ms. Functions: 0 ms. XSLT: 0 ms.

Figure 10: Previewing the XSLT function.

2.3.3 Smaller steps: inserting a new function

If you want to take smaller steps, you can now save this function and test it.

For the limited functionality it has now, it only shows “Hello World!” in my case.

To do so:

1. From the Content perspective, edit a page,
2. Switch to the Visual view if necessary
3. Insert the function (Insert > Function)
4. Then click to expand Demo.Media and you will see your function there.
5. Select it and click OK.

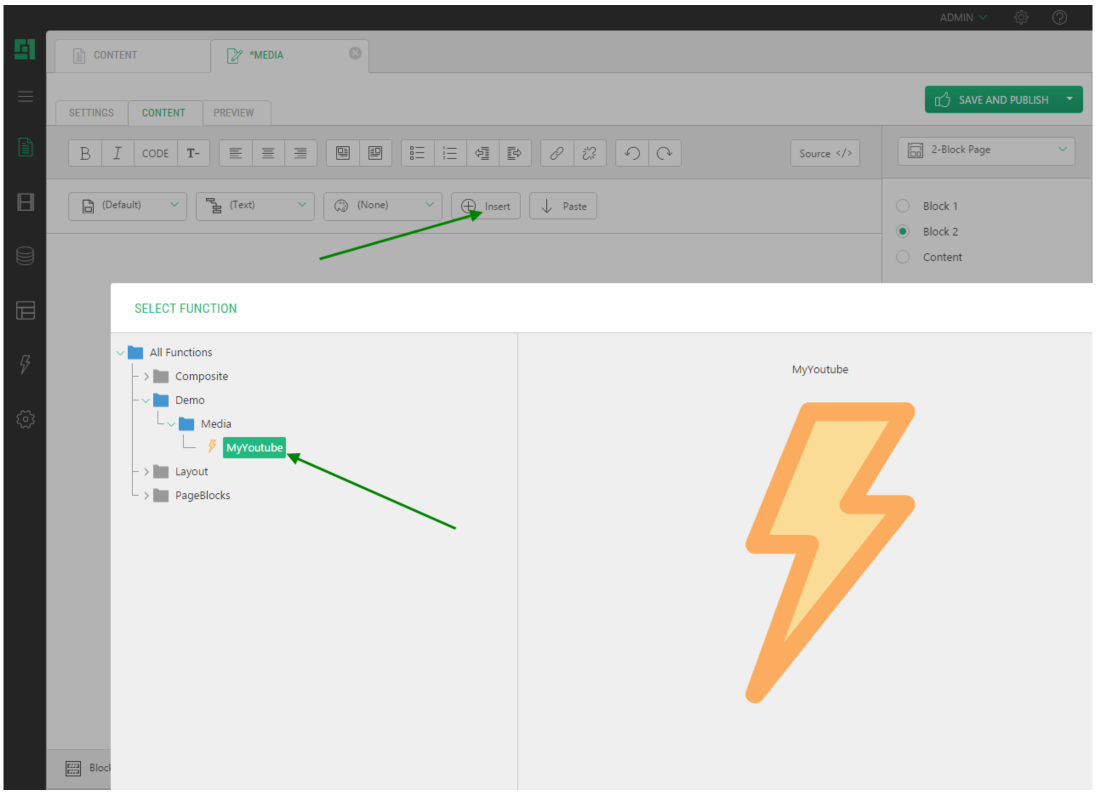
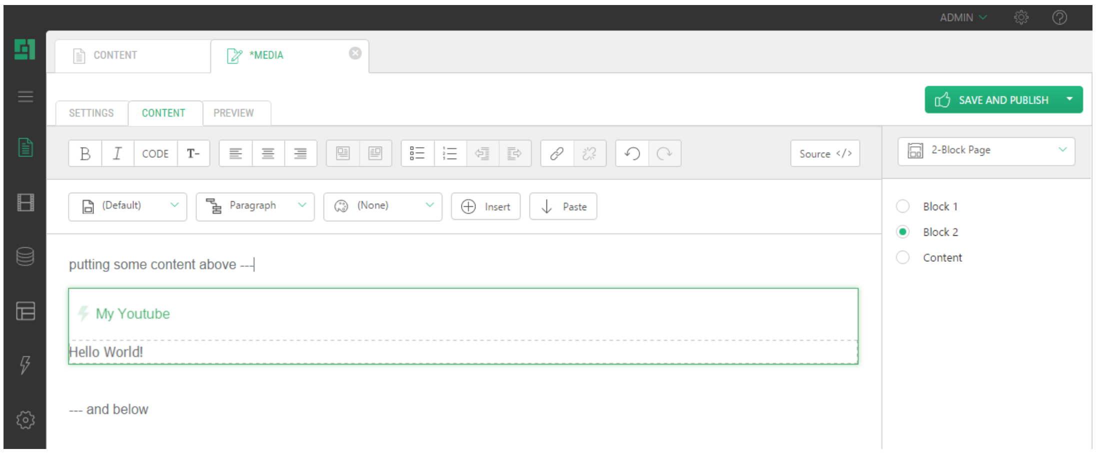


Figure 11: Inserting the function

Now you can see the green box with your function’s name appeared on a page. Add some content above and below your function. Click Preview to see the result.



putting some content above ---

My Youtube

Hello World!

--- and below

Figure 12: The function has been inserted

Okay, that works. Now let’s go back to the function again and add something better than just “Hello World!” to it.

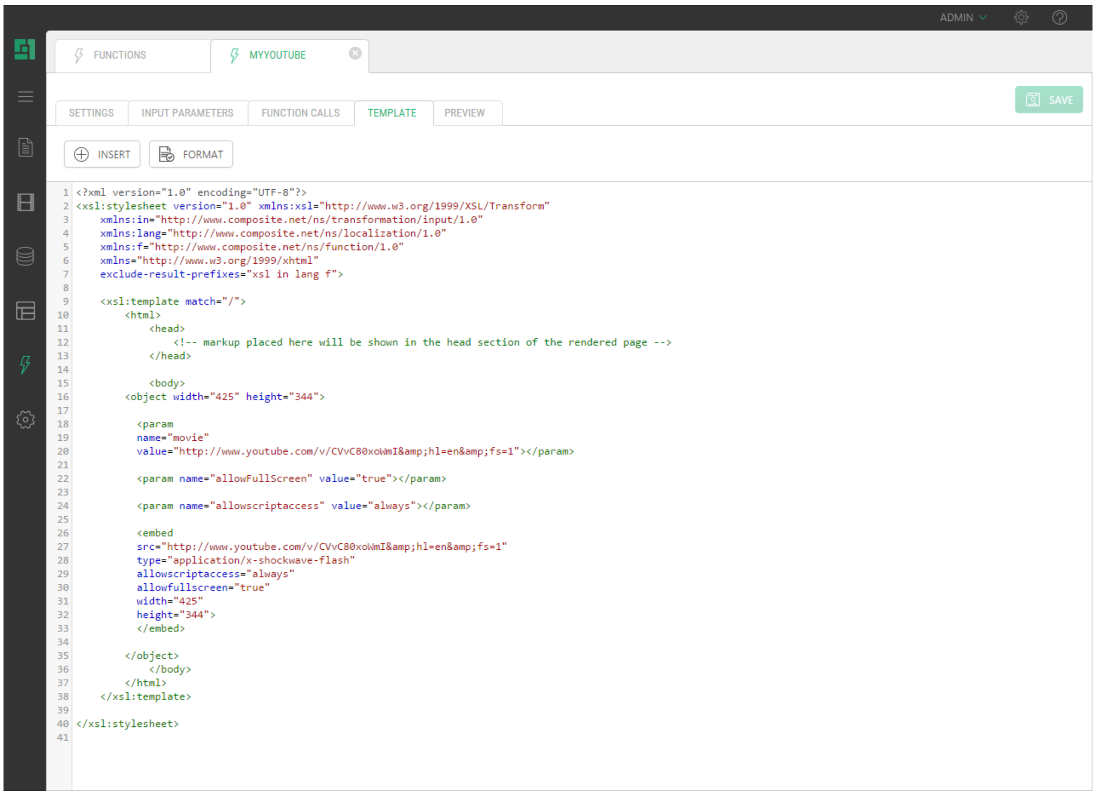
2.4 Rendering YouTube code

Now remove “HELLO WORLD” and add that YouTube code we made earlier. Hopefully, you kept it in Notepad somewhere (Personally I use Notepad++, a free, very handy editor with many cool features, I often have multiple documents open in where I paste parts of code). Otherwise go back in this document to the proper section and copy it again.

- Copy the YouTube object embed code

- Paste the code in the <body> section of the XSLT template.
- Save the function.

If you took the smaller steps, you can switch to the Content perspective again. If you still have that page with your function open, you can click on the Preview tab again. This is a great way of testing your function on a page. Click back and forth from the function and content section (the same works well when modifying templates)

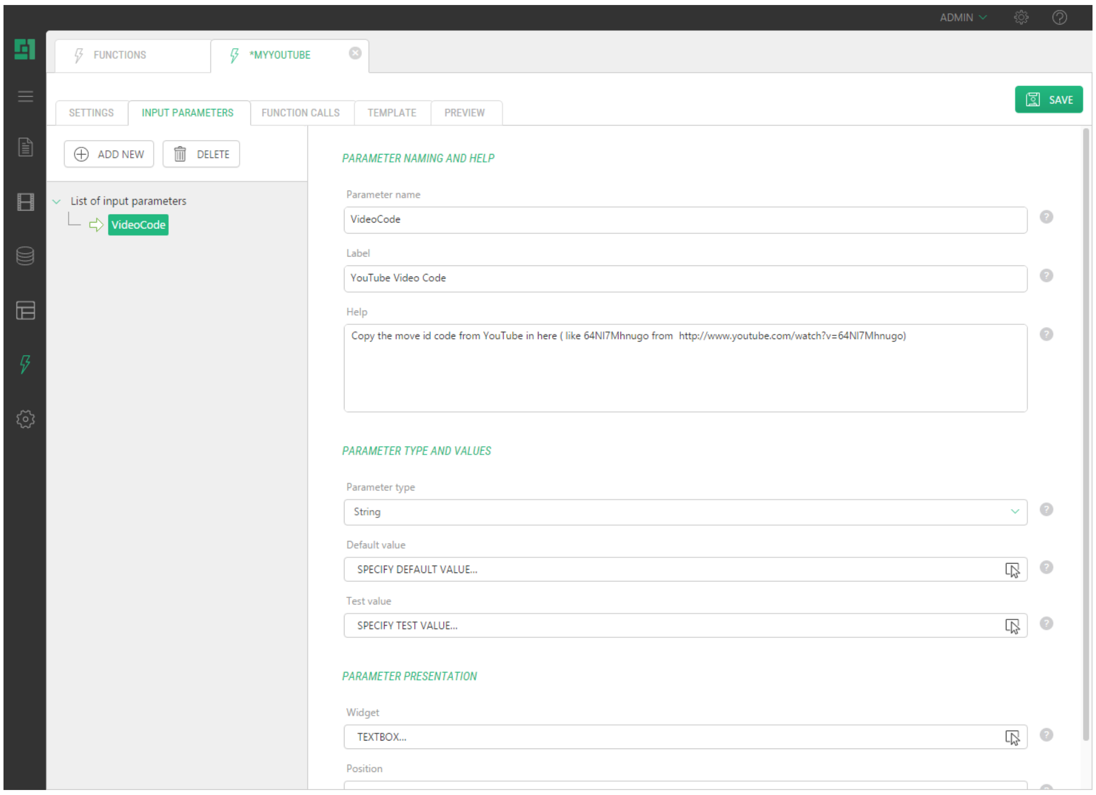


```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:in="http://www.composite.net/ns/transformation/input/1.0"
    xmlns:lang="http://www.composite.net/ns/localization/1.0"
    xmlns:f="http://www.composite.net/ns/function/1.0"
    xmlns="http://www.w3.org/1999/xhtml"
    exclude-result-prefixes="xsl in lang f">

    <xsl:template match="/">
        <html>
            <head>
                <!-- markup placed here will be shown in the head section of the rendered page -->
            </head>

            <body>
        <object width="425" height="344">

          <param
          name="movie"
          value="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"></param>

          <param name="allowFullScreen" value="true"></param>

          <param name="allowscriptaccess" value="always"></param>

          <embed
          src="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"
          type="application/x-shockwave-flash"
          allowscriptaccess="always"
          allowfullscreen="true"
          width="425"
          height="344">
          </embed>

        </object>
            </body>
        </html>
    </xsl:template>

</xsl:stylesheet>
```

Figure 13: Pasting the YouTube HTML embed code in the function

But this will always show the same YouTube video. Now we need to make it dynamic. An editor should supply that video code. To do so, we can use the Input Parameters of a XSLT function.

1. Click on the Input Parameters tab
2. Click Add New (input parameter)
3. Type “VideoCode” in the field name. This is the name of our variable, we need this later in the specific casing
4. Specify the Label and Help fields to give more friendly names and explanation about what this input parameter is used for. For example, I have specified “Copy the move id code from YouTube in here (like 64NI7Mhnugo from http://www.youtube.com/watch?v=64NI7Mhnugo)” in the Help field.



Parameter name: VideoCode

Label: YouTube Video Code

Help: Copy the move id code from YouTube in here (like 64NI7Mhnugo from http://www.youtube.com/watch?v=64NI7Mhnugo)

Parameter type: String

Default value: SPECIFY DEFAULT VALUE...

Test value: SPECIFY TEST VALUE...

Widget: TEXTBOX...

Position

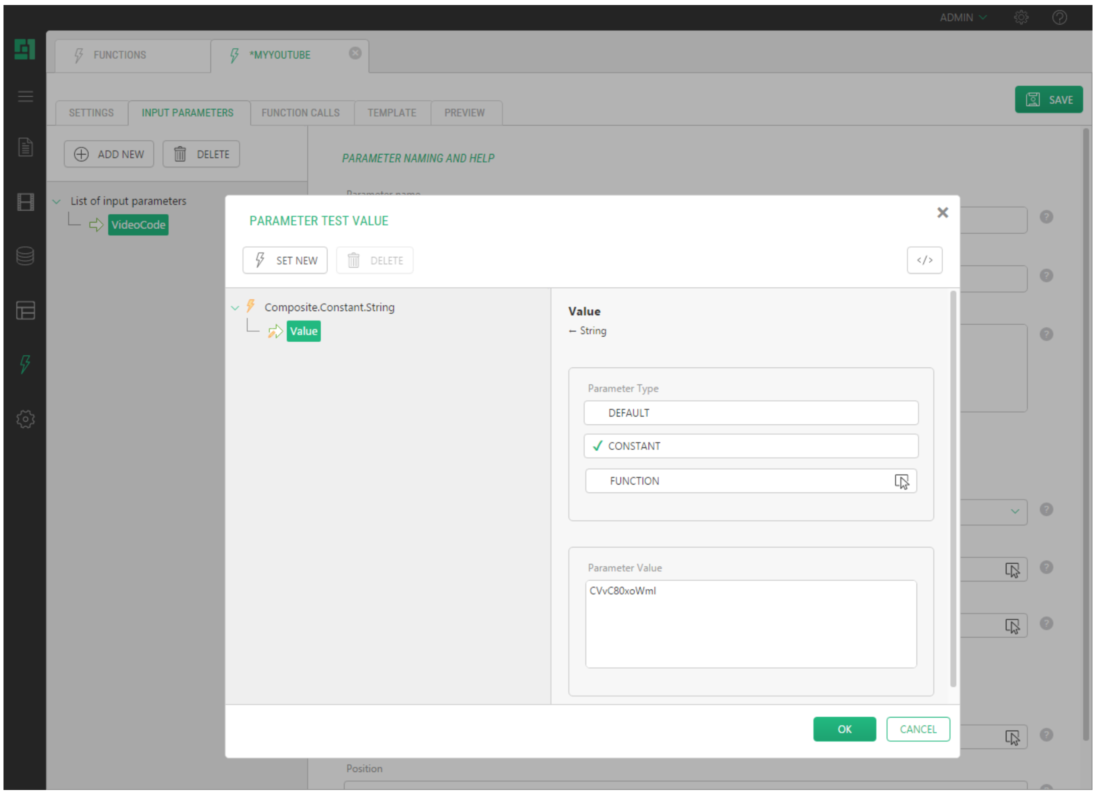
Figure 14: Adding an input parameter

In the Parameter type and values, you specify, of which type the VideoCode parameter is and what the value of the parameter will be, or better where we will get it from.

We need something like “CVvC80xoWmI” to fit in there, which is a string.

Let’s create a test value with that string in there:

1. Click the **Test value**. Two windows will pop up.
2. Click **Set New**.
3. Select the function **Composite | Constant | String**. The function opens and shows that it has a parameter : **Value**)
4. Specify the value for the String constant function by choosing **Constant** and then pasting our video code in.



Composite.Constant.String
Value

Value
– String

Parameter Type
DEFAULT
✓ CONSTANT
FUNCTION

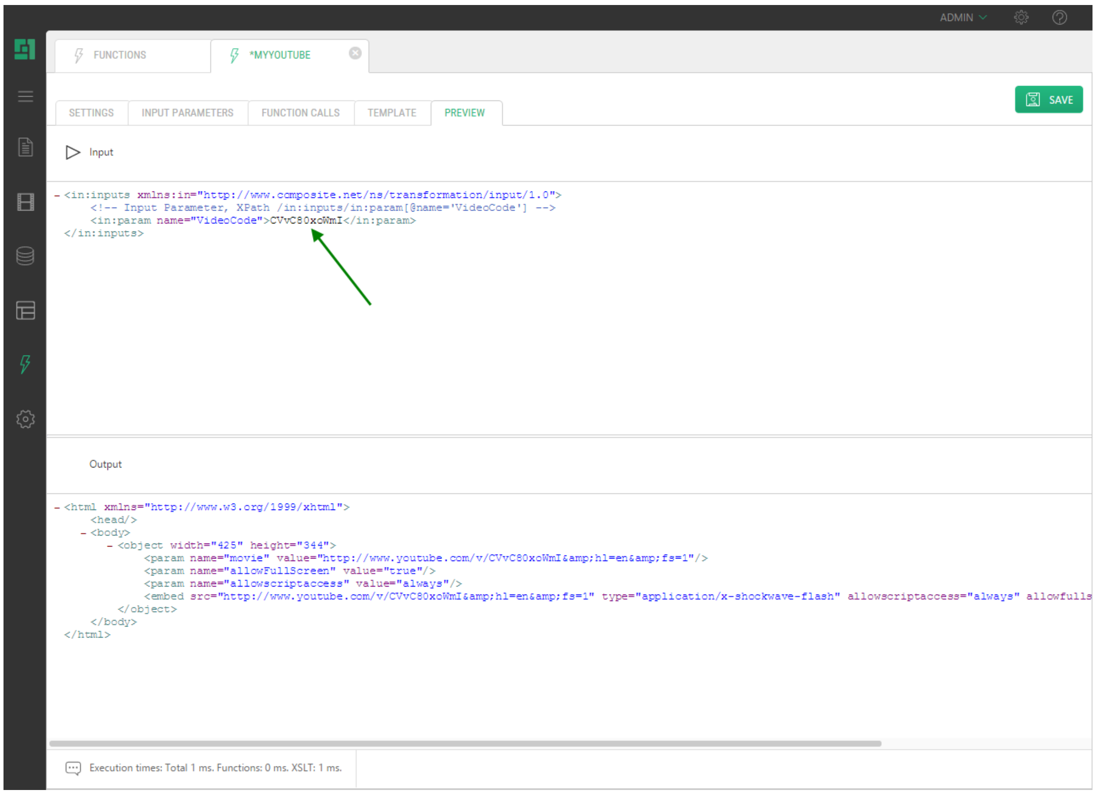
Parameter Value
CVvC80xoWmI

Figure 15: Specifying the constant value –video code

5. Click OK.

You see, you are using CMS functions and input parameters, just like you do with your own right now.

6. Click the Preview tab.



```
- <in:inputs xmlns:in="http://www.composite.net/ns/transformation/input/1.0">
    <!-- Input Parameter, XPath /in:inputs/in:param[@name='VideoCode'] -->
    <in:param name="VideoCode">CVvC80xoWmI</in:param>
  </in:inputs>
```

```
- <html xmlns="http://www.w3.org/1999/xhtml">
    <head/>
  - <body>
    - <object width="425" height="344">
          <param name="movie" value="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1"/>
          <param name="allowFullScreen" value="true"/>
          <param name="allowscriptaccess" value="always"/>
          <embed src="http://www.youtube.com/v/CVvC80xoWmI&amp;hl=en&amp;fs=1" type="application/x-shockwave-flash" allowscriptaccess="always" allowfulls
      </object>
    </body>
  </html>
```

Execution times: Total 1 ms. Functions: 0 ms. XSLT: 1 ms.

Figure 16: Previewing the modified function

You can see our input parameter with the test value appearing in the preview. We will now get that parameter into the XHTML we have specified in the Template. This is where we need to do a little XSLT coding.

2.4.1 Learning the basics of XSLT

This document is not intended to teach you that; but it will give you some bits to start with.

XSLT is quite complex to master but it is quite easy to change if you learn the basics. XSLT works very well on reuse. (Installing add-ons from our package server will provide you with good examples, although some are aimed at a more complex level.)

Once you have gathered some XSLTs, do a lot of googling and practicing in, for example, the free XRay XML Editor. You will learn a lot about XSLT. Focus on learning the basics, see how XSLT looks. You don’t have to be able to write XSLT from scratch nor understand it fully to use and modify it! You will often be able to use a complex XSLT by only changing some simple bits and pieces!

Learn the XPath, learn the template basics and spot the xsl:value-of bits. These are the basic things you need to change in XSLTs. Paste the input XML from your functions into your editor (XRay or better XMLSpy) and code the XSLT in there until you are satisfied, then paste the XSLT back in here.

You will not learn anything specific to C1 CMS here. Just open standards XML and XSLT that you can reuse in many other different projects and products. You can also consult other people knowing nothing of C1 CMS but knowing XSLT.

To start learning XSLT I have created a page on a personal site with some links and tips to resources and other interesting things: http://Vinnie.nerdnet.nl/xslt.

2.4.2 Getting the video code variable with XSLT

As was said, we will not try to teach you XSLT in here, so here we go with only short bits.

We have <in:param name="videocode">CVvC80xoWmI</in:param> which needs to go in our XHTML.

Create a variable in XSLT and set the value with our XML using XPath query:

```
<xsl:param name="videocode" select="/in:inputs/in:param[@name='videocode']"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" />
```

Create the $videocode variable and select the value from our <**in:inputs** …> <**in:param name**="videocode">…

Now to refer to variables in XSLT, you use something like this:

```
<xsl:value-of select="@attribute"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" />
<xsl:value-of select="$variable"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" />
```

With @ you are referring to XML attributes, with $ you are referring to XSLT variables.

If you want a variable within an XHTML element you will use the {$variable notation}

```
<strong><xsl:value-of select="$variable"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" /></strong>
<img src="{$variable}"/>
```

So to put the $videocode in the right spot, you will get to

```
<param name="movie" value="http://www.youtube.com/v/{$videocode}" />
```

The same counts for:

```
<embed src="http://www.youtube.com/v/{$videocode}" />
```

Test it using the preview, and see that the test value appears in the XHTML.

Voila! you managed to pull an input parameter into the output XHTML.

Okay, we managed to get an input parameter into our XHTML but it is still a constant value. We want the editors to specify the value for the videocode input parameter themselves.

This is what the default value is for. Until now we have been having the test value do this. This test value is not required in your function, but it can be handy for testing your function.

1. Save your function.
2. Switch to the Content perspective again.
3. Preview.

Another error!

Instead of your function a **[Error in rendering]** appears! Nasty, but again don't be put off! I'm actually 'forcing' these errors onto you to experience them and now to deal with them.

1. Click on the error.
2. Read the error in the popup

```
System.InvalidOperationException: Failed to get value for function
'Demo.Media.MyYoutube'. ---> System.ArgumentException: Missing parameter
'videocode' (type of System.String)
   at
Composite.Functions.FunctionRuntimeTreeNode.GetValue(FunctionContextContain
er contextContainer)
   --- End of inner exception stack trace ---
   at
Composite.Functions.FunctionRuntimeTreeNode.GetValue(FunctionContextContain
er contextContainer)
   at
Composite.Renderings.Page.PageRenderer.ExecuteEmbeddedFunctions(XElement
element, FunctionContextContainer contextContainer)
```

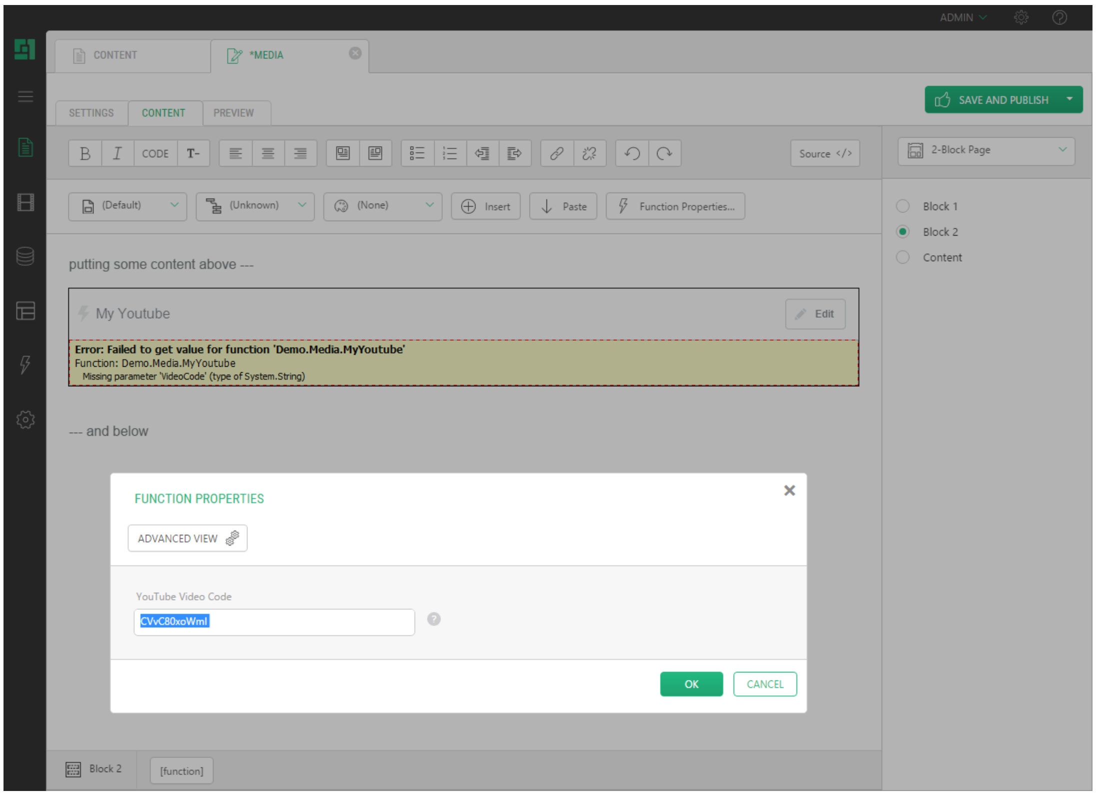
Listing 4: Error text

It reads like a lot of gibberish until you look a little bit closer and see the Missing parameter 'videocode' - hey! that is our variable! And it is missing. Right, we have not specified it yet. This is because we had already inserted our function before it had any input parameters.

2.4.3 Specifying input parameters for a function

Select the green box of your function on the page and click the function Properties button (or right-click the green box). Now you will have to specify your videocode.

Or better yet, remove your function and insert it again. When you insert it you will see that it automatically prompts for the missing videocode parameter. This is how the editors will now work with the function.


putting some content above ---

My Youtube

Error: Failed to get value for function 'Demo.Media.MyYoutube'
Function: Demo.Media.MyYoutube
Missing parameter 'VideoCode' (type of System.String)

--- and below

FUNCTION PROPERTIES

YouTube Video Code: CVvC80xoWml

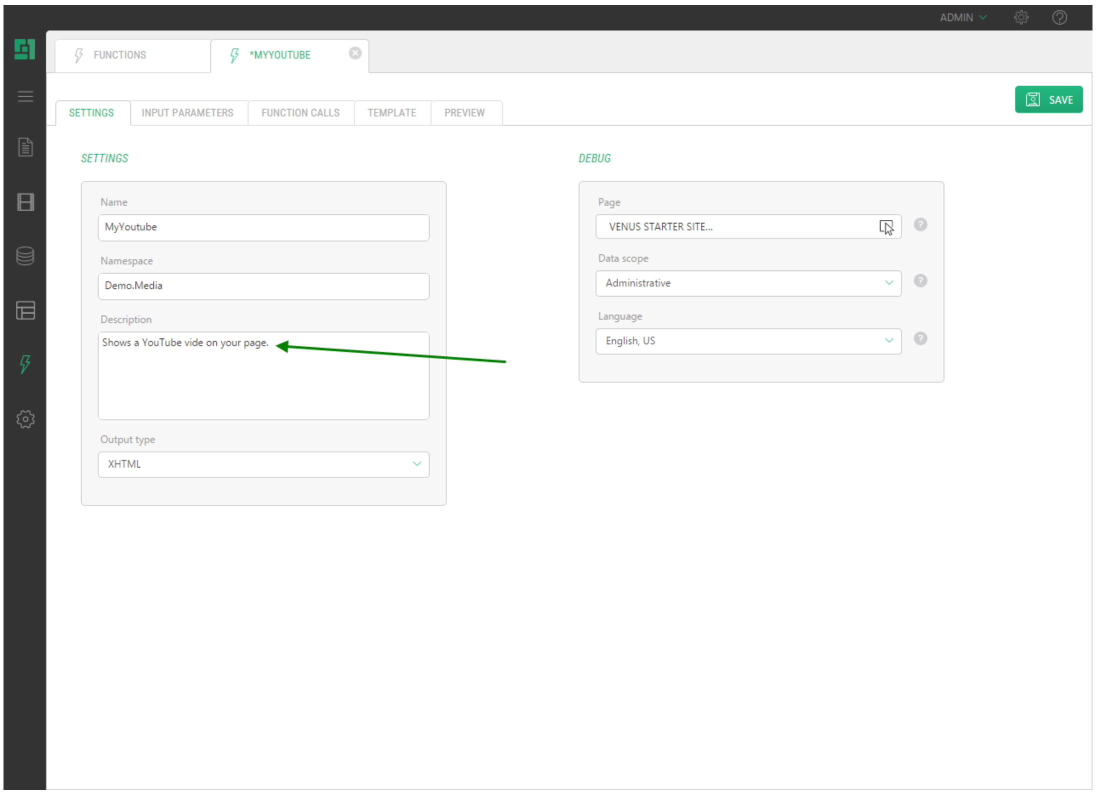
Figure 17: The function prompting for the required parameter

If there is no default value for a parameter, the function will automatically prompt for it. Note the names of the label and help. The videocode parameter is only used in the function itself. (*Sorry, my help text is a bit confusing in here mentioning movie id, which should be a video code.*)

2.4.4 Document your functions

You can also give a description for the whole function:

- Go to the settings tab of your function and type some text in there.



Name: MyYoutube

Namespace: Demo.Media

Description: Shows a YouTube vide on your page.

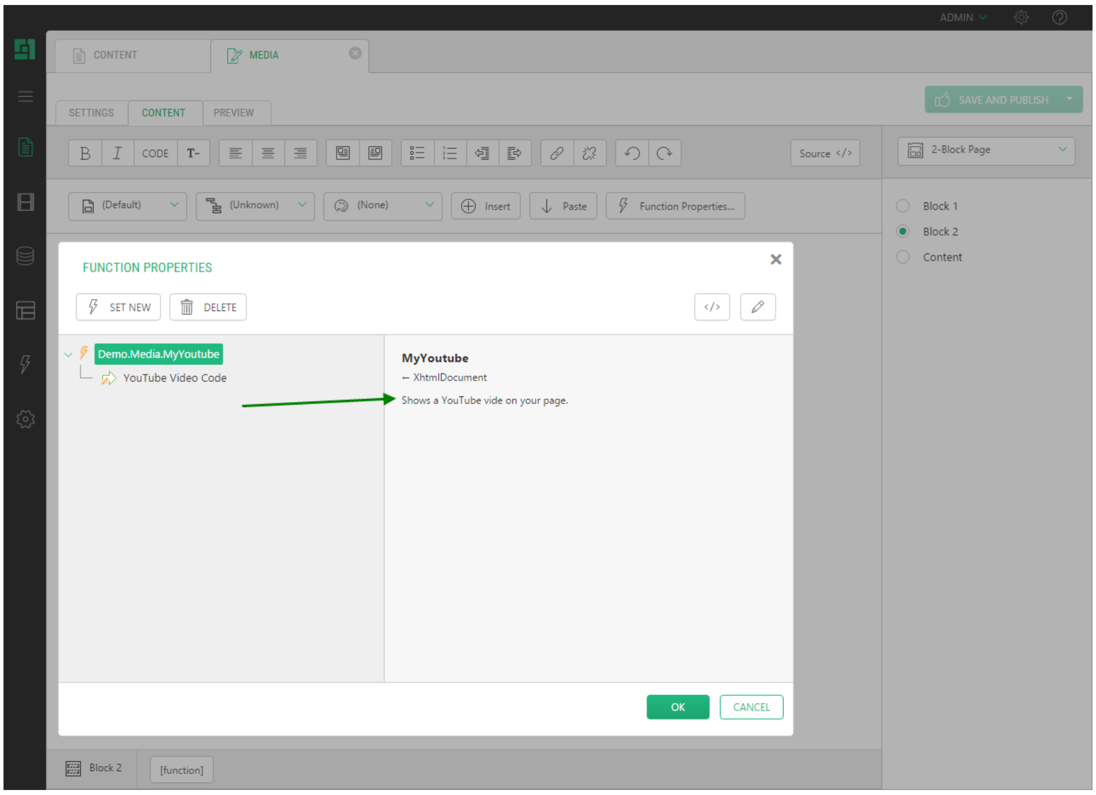
Output type: XHTML

Page: VENUS STARTER SITE...

Data scope: Administrative

Language: English, US

Figure 18: Providing a description for the function



Demo.Media.MyYoutube
YouTube Video Code

MyYoutube
← XhtmlDocument
Shows a YouTube vide on your page.

Figure 19: The function shows its description

2.4.5 Generate documentation on functions

You can also generate automatic information of all your functions. C1 CMS will list all the descriptions, different parameters and help text for all the functions. So it is a very good practice to put some text in there for both your editors and for documentation.

To generate documentation for the functions:

1. From the Function perspective, select All functions.
2. Click Generate Documentation on the toolbar.

Similarly, you can do with All widget functions. You might as well generate on a single function or a namespace with a number of functions in a similar manner.

2.5 More parameters

If you look at the object embed code, you see there are more parameters in there. You can also turn some of these parameters into input parameters of the function, just as you did with the videocode parameter.

```
<embed
src="http://www.youtube.com/v/CVvC80xoWmI" type="application/x-shockwave-
flash"
allowscriptaccess="always"
allowfullscreen="true"
width="425"
height="344" />
```

Listing 5: Parameters of the embed element

To something like:

```
<object width="{$width}" height="{$height}">
  <param name="movie" value="http://www.youtube.com/v/{$videocode}" />
  <param name="allowFullScreen" value="{$allowfullscreen}" />
  <param name="allowscriptaccess" value="always" />
  <embed src="http://www.youtube.com/v/{$videocode}"
  width="{$width}" height="{$height}"
  movie="http://www.youtube.com/v/{$videocode}" type="application/x-
shockwave-flash"
  allowFullScreen="{$allowfullscreen}" allowscriptaccess="always" />
</object>
```

Listing 6: Implementing and using more parameters

You will have to think about what it would be nice to specify and what it would not.

More parameters do make your functions more flexible, but using too many of them will make them harder to use for editors. When implementing more parameters, it makes much sense to use default values so that the editors have to specify as fewer values as possible.

Also think about the types of parameters, don’t stick with strings, and use Booleans and integers where possible. It will prevent you from getting wrong values. C1 CMS is heavy on strong typing, preventing many errors and bugs before they might occur.

2.5.1 Default value parameter

To create a default value, do the same as you did with the test value. It makes sense to do so except for the very minimum of default parameters that need to be specified by the user. By not specifying a default value, you force it to be specified by the user. In this case, I would specify default parameters for all the additional parameters you might choose to add and only leave the default value for the videocode unspecified.

2.5.2 More uses of an XSLT function without XML

We used all except for the Function calls tab. This tab is actually a very important tab you will use in many XSLT functions except for those functions where you don't need to get data from the C1 CMS repository.

Most functions will use the Function calls section, they do so to get content from somewhere and then transform the xml with the XSLT. You will learn this in other documents on XSLT functions.

2.5.3 Templates and XHTML includes

Another good example of using XSLT functions, like we did without using the XML function, is for including XHTML in templates. When you work with templates, you will often find pieces of HTML, which are the same in several templates (like CSS and JavaScript includes, disclaimers, headers, footers, web statistics scripts, Google analytics etc). In those cases, it is good to put that HTML in one place, instead of having it in x places (x templates). It means that you will most likely put the HTML in the XSLT function (the way we started out) and add the function to the templates.

Some might view this as a "downside":

- they want to change something in the HTML
- open the template
- find out the code is not there
- need to go to the function and edit it there

But this it is not a downside!

It is true, you will have to take a few extra steps, but:

- you change it for all x templates at once
- it prevents you from forgetting a single template
- it prevents you from ending up with different pieces where you actually want is one!

So it is a good practice to make your code better to manage!

3 Vimeo

So we have made a function for YouTube. Let's make one for Vimeo, another online video service, as well.

Visit vimeo.com and examine at the site and find out the way to embed their videos.

Their URL's to a video is something like this: http://www.vimeo.com/3826770

So they use a large integer for uniquely identifying a video.

They provide an embed tag code generator as well:

```
<object width="400" height="301"><param name="allowfullscreen" value="true"
/><param name="allowscriptaccess" value="always" /><param name="movie"
value="http://vimeo.com/moogaloop.swf?clip_id=3826770&amp;server=vimeo.com&
amp;show_title=1&amp;show_byline=1&amp;show_portrait=0&amp;color=&amp;fulls
creen=1" /><embed
src="http://vimeo.com/moogaloop.swf?clip_id=3826770&amp;server=vimeo.com&am
p;show_title=1&amp;show_byline=1&amp;show_portrait=0&amp;color=&amp;fullscr
een=1" type="application/x-shockwave-flash" allowfullscreen="true"
allowscriptaccess="always" width="400" height="301"></embed></object><br
/><a href="http://vimeo.com/3826770"> Rihanna - Disturbia (Live@VMA
2008)</a> from <a href="http://vimeo.com/user1477864">Rihanna More</a> on
<a href="http://vimeo.com">Vimeo</a>
```

Listing 7: Vimeo's embed video code

Ehm, doesn't this look familiar?

I guess I can leave the rest to you!

I would surely add the width and height parameters in here, in YouTube they are mostly the same but with Vimeo they differ a lot. Unfortunately I have not yet found out how to request the correct values from Vimeo. Users can use the preview and the Vimeo embed code to pull the correct sizes. Tell them to do so with your function description and help text on your function.

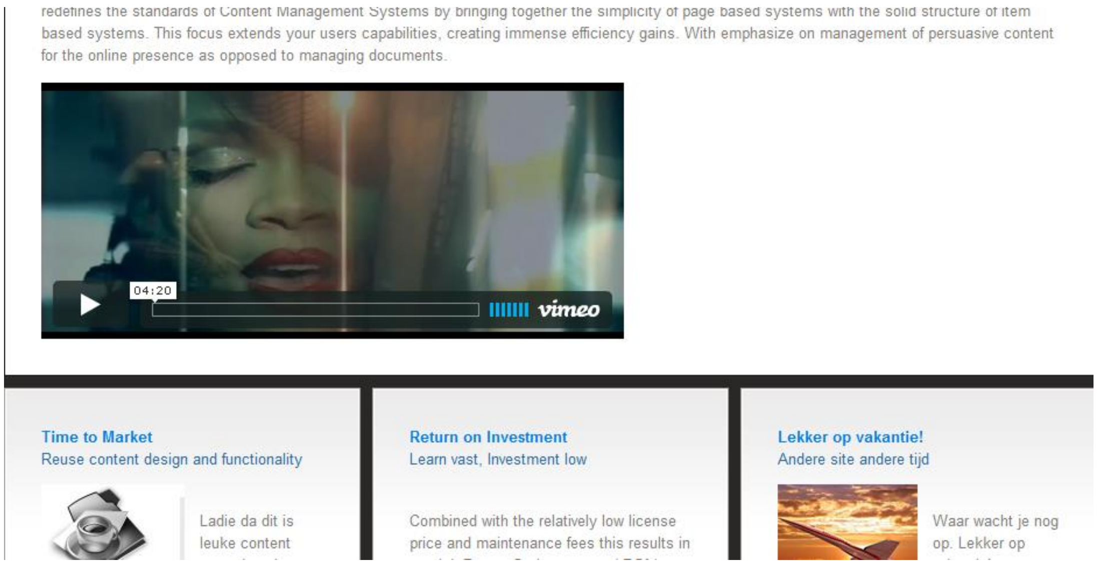
And they allow for a little more customizing, you can specify colors etc to better match your website!

I came up with this XSLT part in the <body> section:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...
<!-- Vincent van Gentevoort nov 2008
optional attributes
 show_title,  show_byline  show_portrait  fullscreen  color  00adef  blue

http://vimeo.com/moogaloop.swf?clip_id={$moviecode}&amp;server=vimeo.com&am
p;show_title=0&amp;show_byline=0&amp;show_portrait=0&amp;color=&amp;fullscr
een=0
-->
<xsl:param name="moviecode" select="/in:inputs/in:param[@name='moviecode']"
/>
<xsl:param name="height" select="/in:inputs/in:param[@name='height']" />
<xsl:param name="width" select="/in:inputs/in:param[@name='width']" />
<xsl:param name="allowfullscreen"
select="/in:inputs/in:param[@name='allowfullscreen']" />
      <xsl:template match="/">
            <html>
                  <head />
                  <body>
 <object width="{$width}" height="{$height}">
 <param name="allowfullscreen" value="true" />
 <param name="allowscriptaccess" value="always" />
 <param name="movie"
 value="http://vimeo.com/moogaloop.swf?clip_id={$moviecode}" />
 <embed
 src="http://vimeo.com/moogaloop.swf?clip_id={$moviecode}"
 type="application/x-shockwave-flash"
 allowfullscreen="false"
 allowscriptaccess="always"
width="{$width}" height="{$height}">
 </embed></object>
...
              </body>
       </html>
</xsl:template>
</xsl:stylesheet>
```

Listing 8: Using the embed code in the function

This results in something like this:


redefines the standards of Content Management Systems by bringing together the simplicity of page based systems with the solid structure of item based systems. This focus extends your users capabilities, creating immense efficiency gains. With emphasize on management of persuasive content for the online presence as opposed to managing documents.

Time to Market
Reuse content design and functionality

Ladie da dit is leuke content

Return on Investment
Learn vast, Investment low

Combined with the relatively low license price and maintenance fees this results in

Lekker op vakantie!
Andere site andere tijd

Waar wacht je nog op. Lekker op

Figure 20: Part of the rendered page with the Vimeo function

4 What’s more?

The topic of creating XSLT function – and particularly, that of creating functions that embeds codes from other websites - cannot be limited to the guidelines in the previous chapters. However, the above guidelines give you a good start with XSLT functions.

You can of course do many other things in the area outlined above. In the following few short sections, you can learn what you can do in addition to what you have just learned.

4.1 Integrating with datatypes

In this case, we made functions that provide editors with a way to freely integrate video on pages wherever they like. You can also go for a more structured approach. For example, if you have a datatype called “News” and can often find a video to go with that you can make the videocode a field of the datatype. From there it actually is quite similar to what we have done in this guide, but now we specify in the function rendering the datatype that we also put in the code we have done above.

(XSLT Functions rendering datatypes is described in another document where the XML part also comes in.)

4.2 Possible enhancements: full URL, full embed code

Our functions have a little downside; they still require the editors to do some additional things, picking the right code from the URL or embed code.

It would be even better to allow pasting of a full URL, or even better the full embed code (which also holds the width and height), in which case the input parameter “videocode” could hold the full URL or embed code and the XSLT would retrieve the right values from out of there. But that requires some advanced parsing in the XHTML.

4.3 And even more?

There are many more online services that you can embed in a similar manner: online photos, online presentations, Google maps etc.

Have fun and let me know what you have come up with!

Oh, and of course, you can turn a function into an add-on so you don’t have to manually recreate it each time for different customers. But that story is part of another document.