Creating Link to Detail XSLT Function

2017-02-15

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

www.orckestra.com

Contents

1 Introduction

This guide presents explanation illustrated with screenshots and code samples on how to create XSLT functions for a link and detail overview of datatype elements (for example, News items or any other type).

Learning topics in this document are:

- Creating an XSLT function
- Using CMS functions instead of programming
- Using CMS functions within CMS functions
- Creating an input parameter filter
- Getting data from a query string

In this example (specifically, for the NewsDetail function), you will see how to create a complex function in C1 CMS using other functions. The NewsDetail function is a more complex part of this job. It is complex but also very interesting because, we will create a filter without actual programming, by using an input parameter!

A .NET developer would probably say: “I can do that easier with a few lines of code”. True, but what is easy relates directly to your skills.

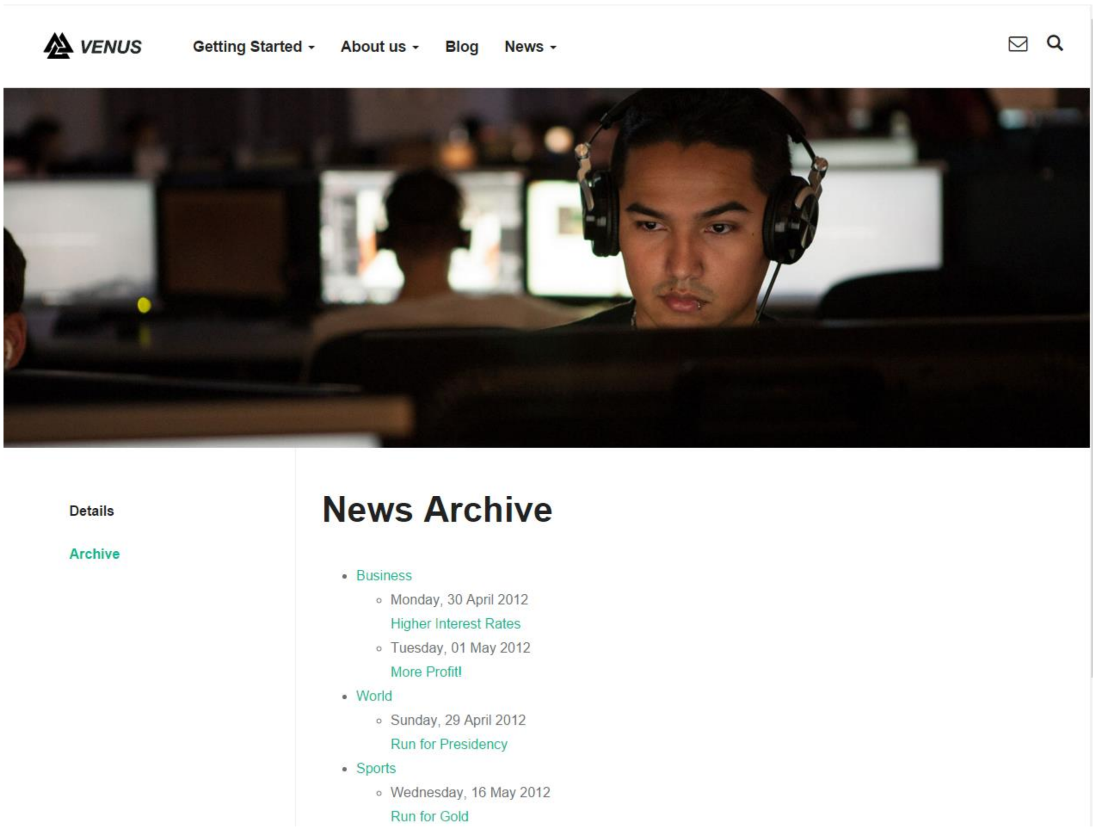
Note: We will do things manually in this example to learn from it. But it is also available as an add-on, so with the deployment wizard, this can all be installed in one go.

Important: In Composite C1 (now C1 CMS) version 5.0 or later, you can quickly build list-to-detail views for a data type by creating a CMS function with a parameter of the RoutedData<T> type. Please see *Data URL Routing* for more information.

2 The Pages

We will make an ad-hoc “news archive” page, e.g: http://www.contoso.com/NewsArchive

(Make sure to replace the server name and port as well as the page name with your own.)



News Archive

- Business
  - Monday, 30 April 2012
    Higher Interest Rates
  - Tuesday, 01 May 2012
    More Profit!
- World
  - Sunday, 29 April 2012
    Run for Presidency
- Sports
  - Wednesday, 16 May 2012
    Run for Gold

Figure 1: The News Archive page

When calling the **NewsArchive** function, you can see that its parameter (**NewsDetailsPage**) points to the “news details” page (via its ID).

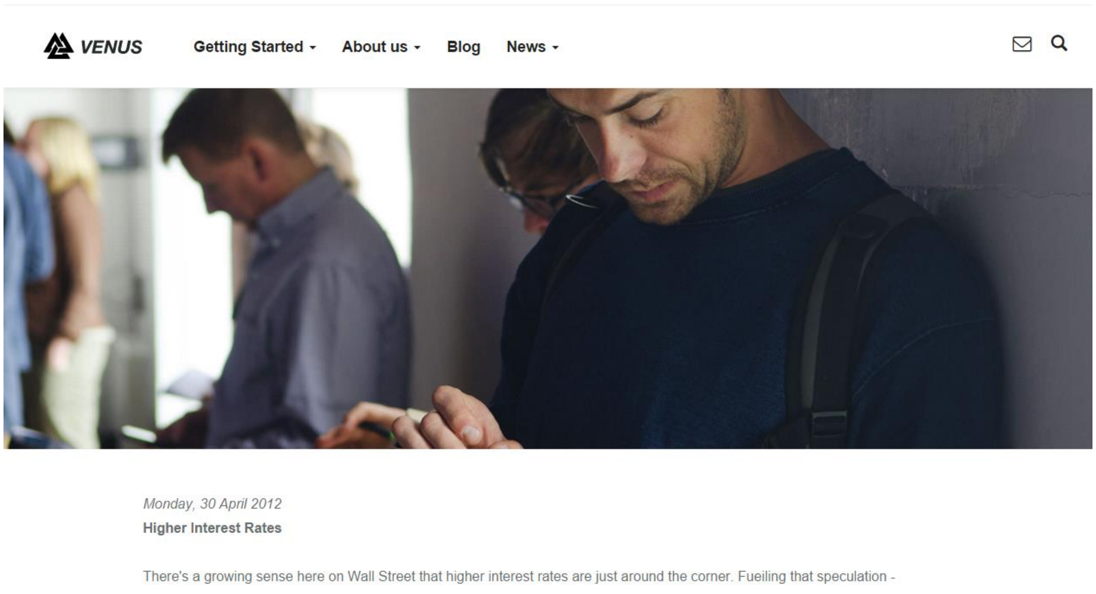
```
<f:function name="Samples.News.NewsArchive"
xmlns:f="http://www.composite.net/ns/function/1.0">
  <f:param name="NewsDetailsPage" value="4bab5861-9a34-478a-828c-
f75dcdaf8729" />
</f:function>
```

Listing 1: Calling the NewsArchive function

Click the link for the news details - and the “news details” page will open as similar to:

http://www.contoso.com/NewsDetails?NewsID=e18873b2-0e0b-4041-92b6-4913892ac3bd&News=Higher%20interest%20rates

(Make sure to replace the server name and port as well as the page name with your own.)



Monday, 30 April 2012

Higher Interest Rates

There's a growing sense here on Wall Street that higher interest rates are just around the corner. Fueling that speculation -

Figure 2: The news details of one news element

On our "news details" page both functions are used: **NewDetails** and **NewsLatest**

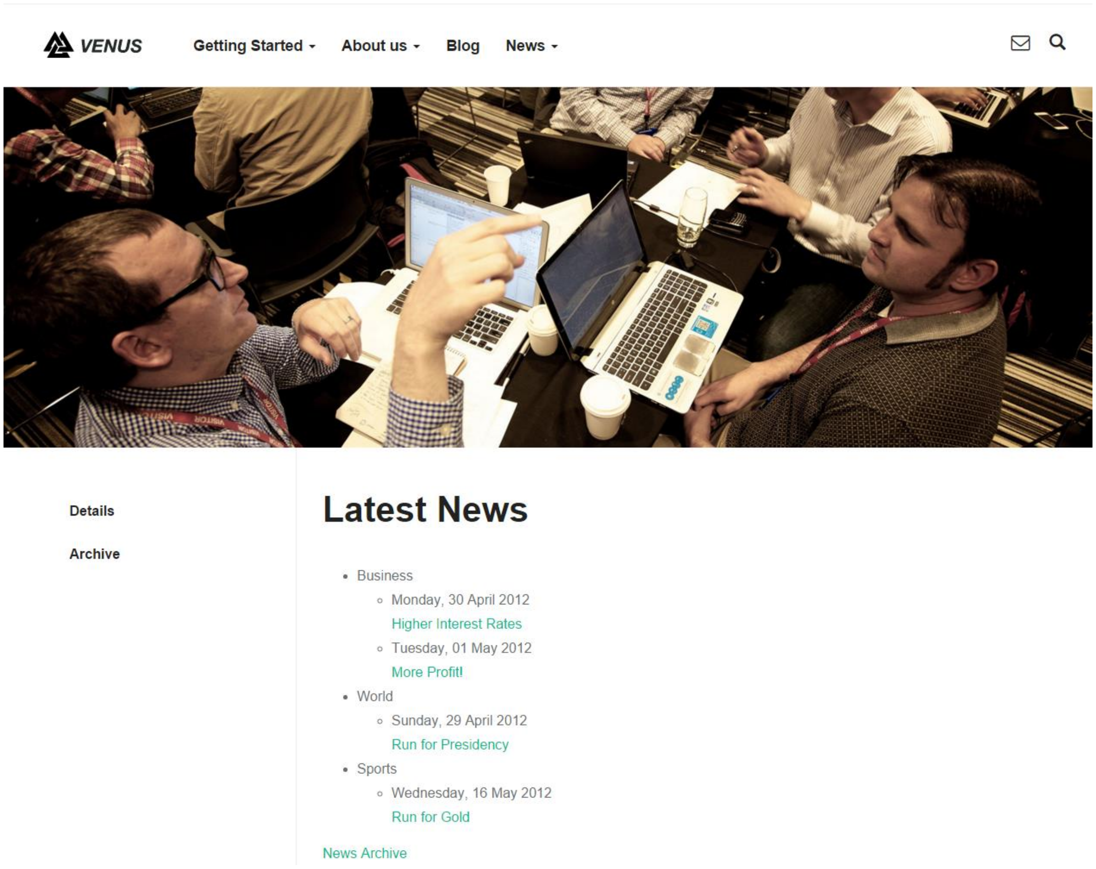
```
<f:function name="Samples.News.NewsDetails"
xmlns:f="http://www.composite.net/ns/function/1.0" />
```

Listing 2: Calling the NewDetails function

```
<f:function name=" Samples.News.NewsLatest"
xmlns:f="http://www.composite.net/ns/function/1.0">
  <f:param name="PageSize" value="5" />
  <f:param name="NewsArchivePage" value="0c94fe9d-c1bf-467e-9918-
e7d4b8a4d3ea" />
  <f:param name="NewsDetailsPage" value="4bab5861-9a34-478a-828c-
f75dcdaf8729" />
</f:function>
```

Listing 3: Calling the NewsLatest function

The **NewsLatest** function is almost the same as **NewsArchive** but now it lists a fixed number of articles per category (maximum 5 in this sample):



Details

Archive

Latest News

- Business
  - Monday, 30 April 2012
    Higher Interest Rates
  - Tuesday, 01 May 2012
    More Profit!
- World
  - Sunday, 29 April 2012
    Run for Presidency
- Sports
  - Wednesday, 16 May 2012
    Run for Gold

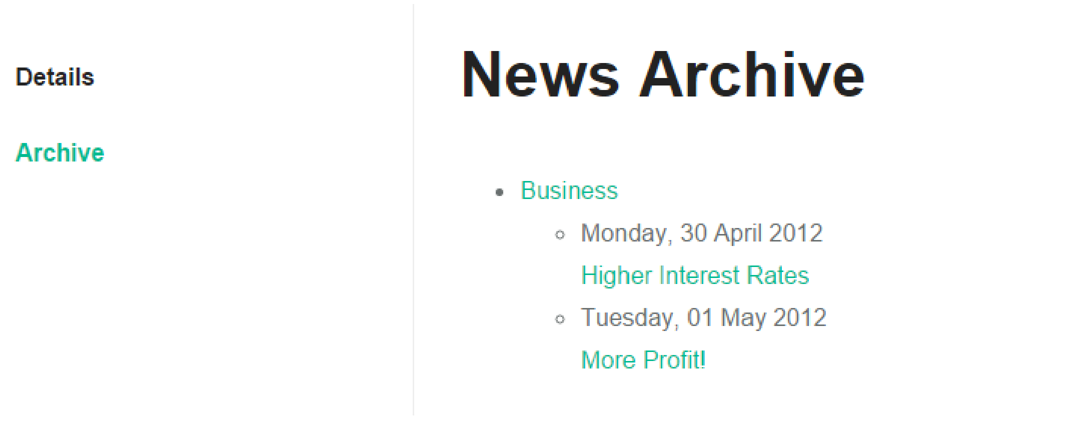
News Archive

Figure 3: The “News Details” article with NewsLatest underneath

Yes, we need a nicer link for this, based on a title without spaces or the like.

So you have a link like this: http://www.contoso.com/NewsArchive?Category=Business

Note: You must avoid duplicate news titles for this to work nicely.



Details

Archive

News Archive

- Business
  - Monday, 30 April 2012
    Higher Interest Rates
  - Tuesday, 01 May 2012
    More Profit!

Figure 4: Using the “Category” query string filter (done in XSLT in this case)

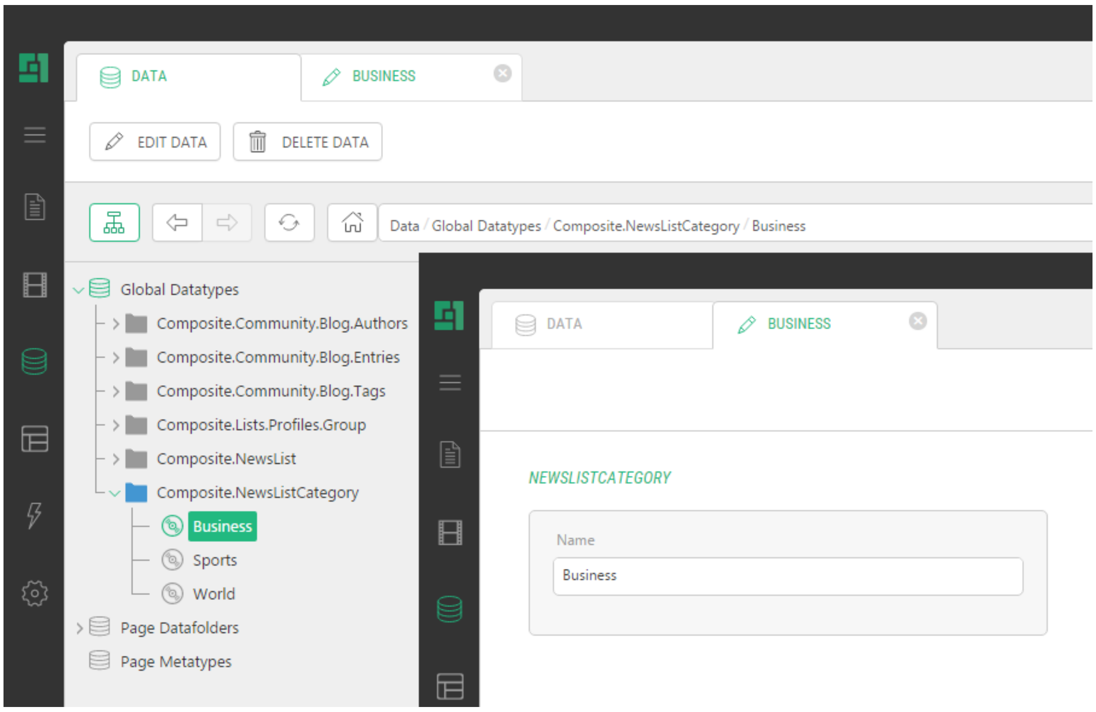
3 The Datatypes

Two datatypes must be present in C1 CMS for the News-related functions to work:

- “News Category” (Samples.News.Category)
- “News Item” (Samples.News.Item)

3.1 News Category

We also introduce some very simple metadata we will use to group our news articles - just one field name and some data for the “News Category” datatype.

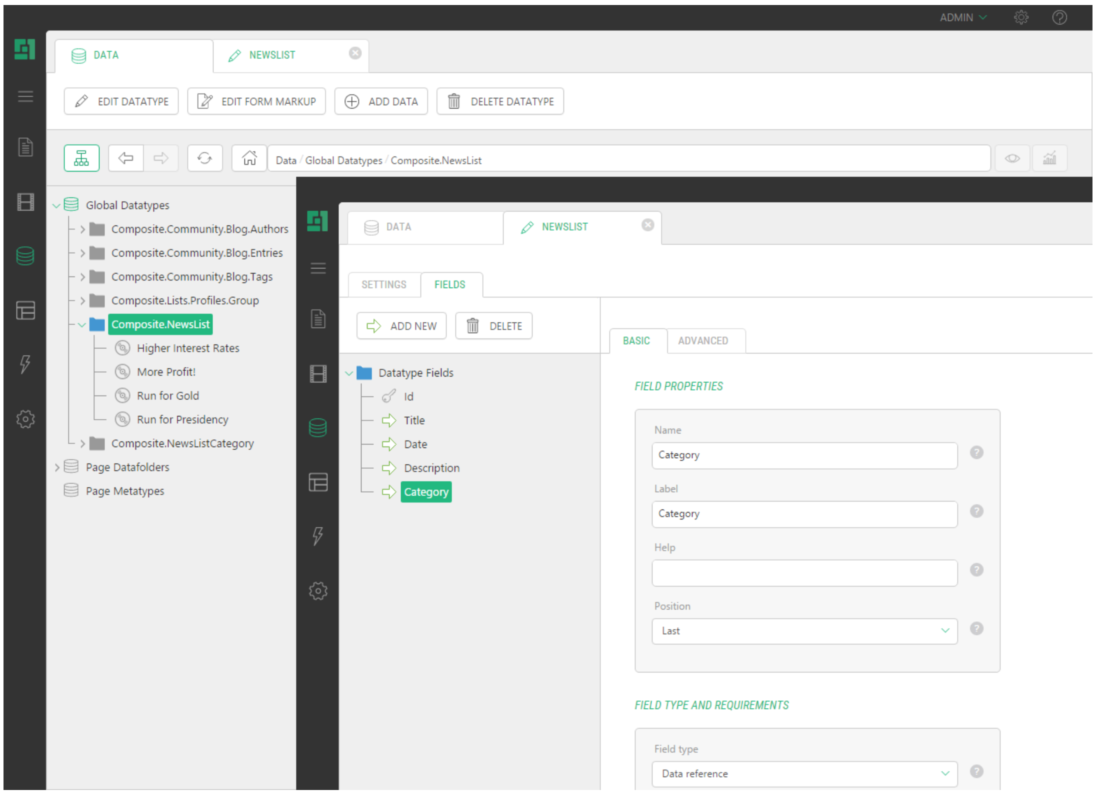

DATA
BUSINESS
EDIT DATA
DELETE DATA
Data / Global Datatypes / Composite.NewsListCategory / Business
Global Datatypes
Composite.Community.Blog.Authors
Composite.Community.Blog.Entries
Composite.Community.Blog.Tags
Composite.Lists.Profiles.Group
Composite.NewsList
Composite.NewsListCategory
Business
Sports
World
Page Datafolders
Page Metatypes
DATA
BUSINESS
NEWSLISTCATEGORY

Name: Business

Figure 5: The News Category datatype

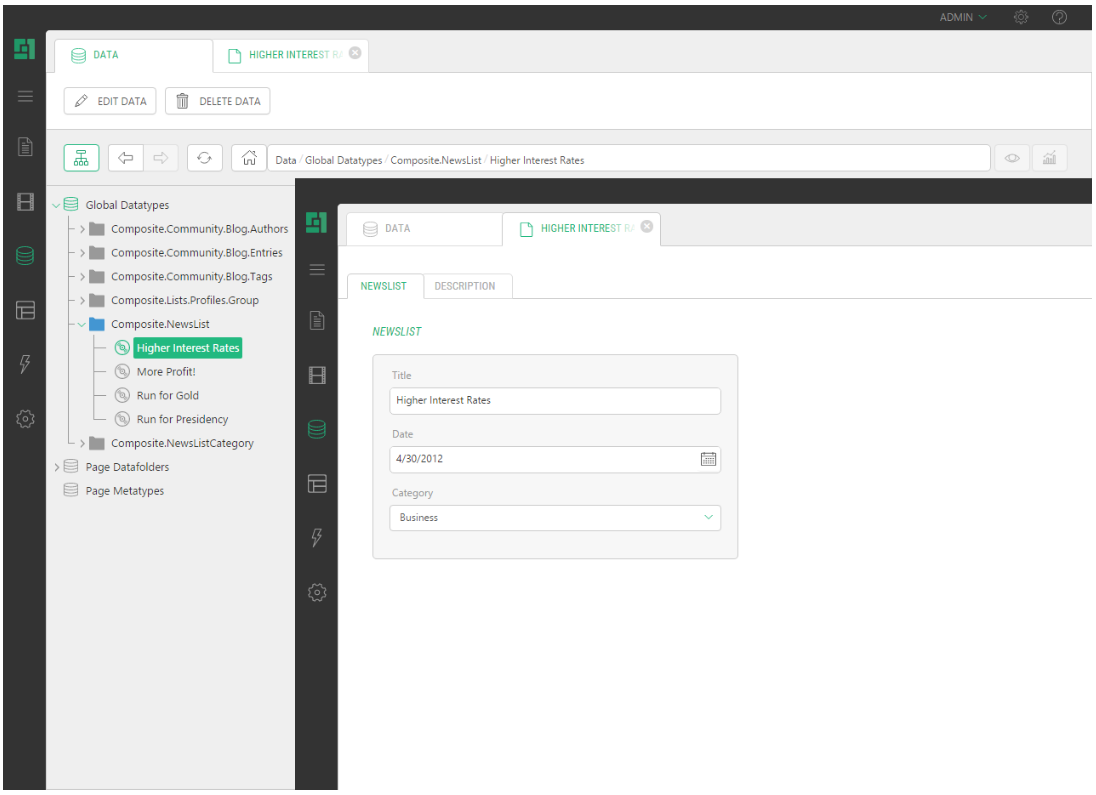
3.2 News Item

“News Item” is another simple datatype with some content, a date field and a field reference to our “News Category” datatype.



Name	Category
Label	Category
Help	
Position	Last
Field type	Data reference

Figure 6: The News Item datatype


ADMIN
DATA
HIGHER INTEREST R/
EDIT DATA
DELETE DATA
Data / Global Datatypes / Composite.NewsList / Higher Interest Rates
Global Datatypes
Composite.Community.Blog.Authors
Composite.Community.Blog.Entries
Composite.Community.Blog.Tags
Composite.Lists.Profiles.Group
Composite.NewsList
Higher Interest Rates
More Profit!
Run for Gold
Run for Presidency
Composite.NewsListCategory
Page Datafolders
Page Metatypes
NEWSLIST
DESCRIPTION
NEWSLIST

Title	Higher Interest Rates
Date	4/30/2012
Category	Business

Figure 7: And some data for the News Item datatype

4 The Functions

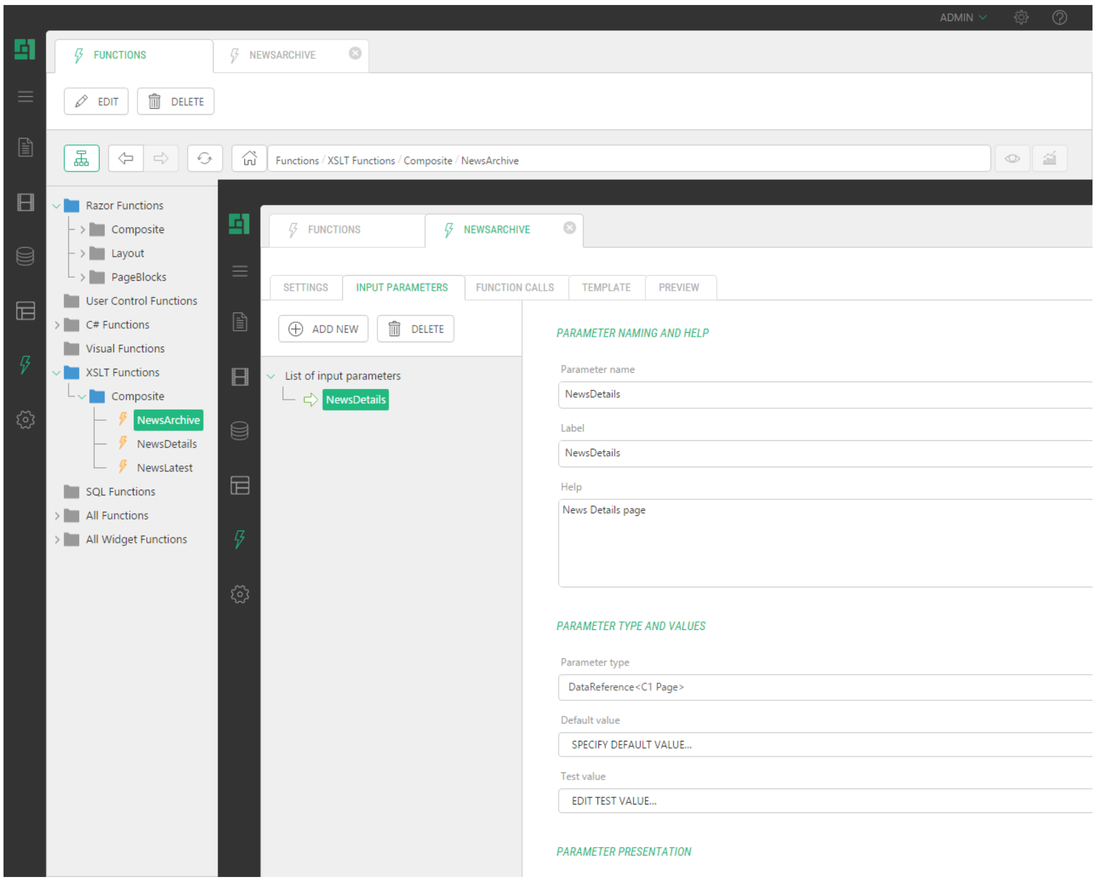
We have three functions here. All the functions are XHTML-based. The NewsArchive and NewsLatest functions are very similar. NewsLatest has a little less functionality than NewsArchive.

So we'll focus on NewsArchive and NewsDetails. NewsDetails is the most complex but also the most interesting one to learn from.

4.1 NewsArchive

This function lists the news articles grouped by category. You can also specify which category you would like to see.

The function needs to return links to the "news details" page so here we have an input parameter called **NewsDetailsPage** which holds that URL.



PARAMETER NAMING AND HELP

Parameter name: NewsDetails

Label: NewsDetails

Help: News Details page

PARAMETER TYPE AND VALUES

Parameter type: DataReference<C1 Page>

Default value: SPECIFY DEFAULT VALUE...

Test value: EDIT TEST VALUE...

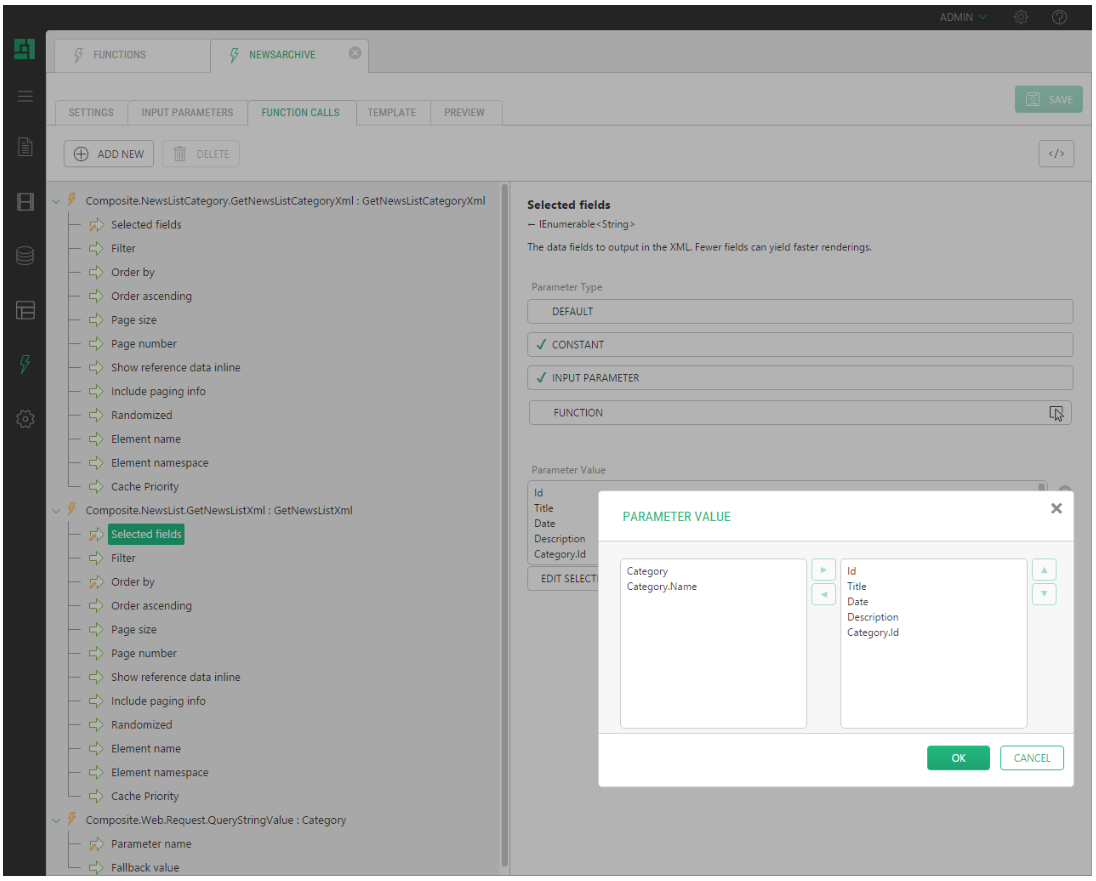
PARAMETER PRESENTATION

Figure 8: The NewsDetailsPage input parameter in NewsArchive

This function calls three other functions to:

- Get the news data (using the generated Get<Type>XML function)
- Get category data (using the generated Get<Type>XML function)
- Get the query string parameter to specify a category (optional)

It is pretty straightforward with the C1 CMS generated functions to get your data in XML.



Composite.NewsListCategory.GetNewsListCategoryXml : GetNewsListCategoryXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Composite.NewsList.GetNewsListXml : GetNewsListXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Composite.Web.Request.QueryStringValue : Category
- Parameter name
- Fallback value

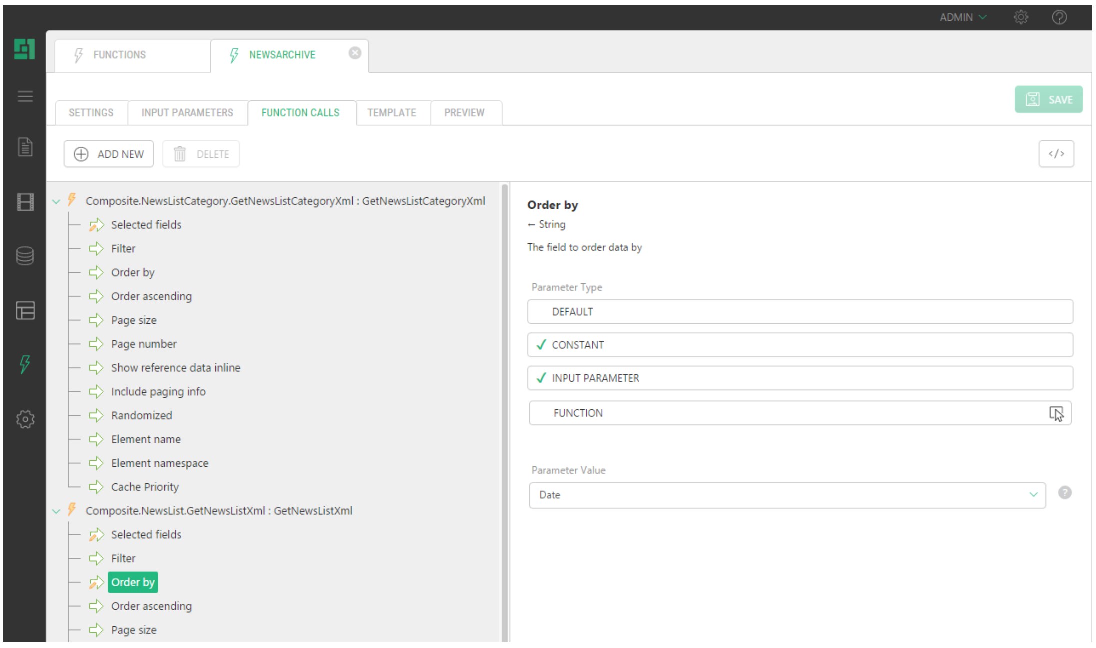
Selected fields
← IEnumerable<String>
The data fields to output in the XML. Fewer fields can yield faster renderings.

Parameter Type
DEFAULT
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Value
Id
Title
Date
Description
Category.Id

Category	Id
Category.Name	Title
	Date
	Description
	Category.Id

Figure 9: Getting fields in XML



Composite.NewsListCategory.GetNewsListCategoryXml : GetNewsListCategoryXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

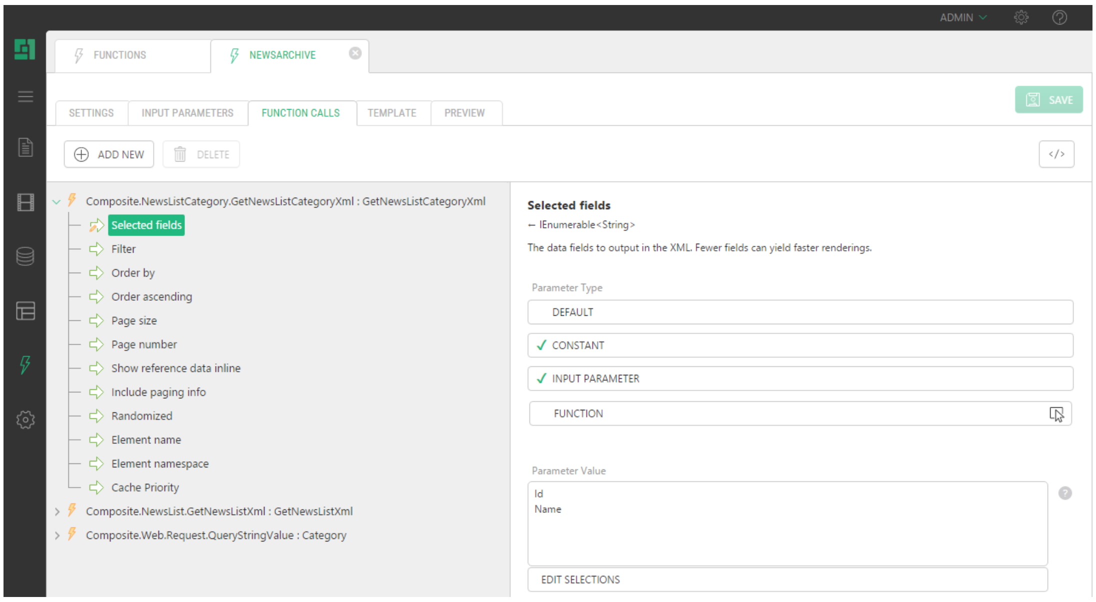
Composite.NewsList.GetNewsListXml : GetNewsListXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size

Order by
← String
The field to order data by

Parameter Type
DEFAULT
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Value
Date

Figure 10: Sorting items by date



Composite.NewsListCategory.GetNewsListCategoryXml : GetNewsListCategoryXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Composite.NewsList.GetNewsListXml : GetNewsListXml
Composite.Web.Request.QueryStringValue : Category

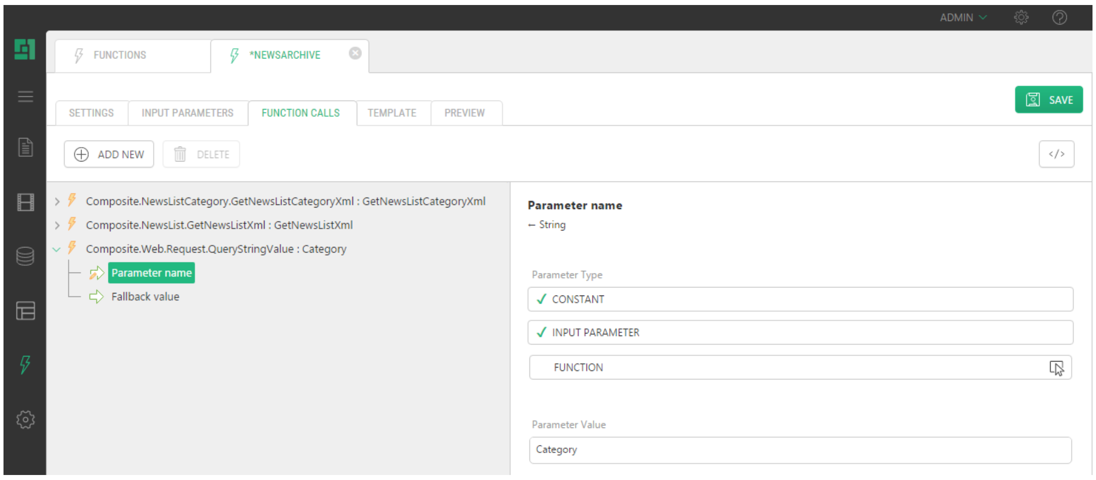
Selected fields
– IEnumerable<String>
The data fields to output in the XML. Fewer fields can yield faster renderings.

Parameter Type
DEFAULT
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Value
Id
Name

EDIT SELECTIONS

Figure 11: Getting the category data and fields



Composite.NewsListCategory.GetNewsListCategoryXml : GetNewsListCategoryXml
Composite.NewsList.GetNewsListXml : GetNewsListXml
Composite.Web.Request.QueryStringValue : Category
- Parameter name
- Fallback value

Parameter name
– String

Parameter Type
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Value
Category

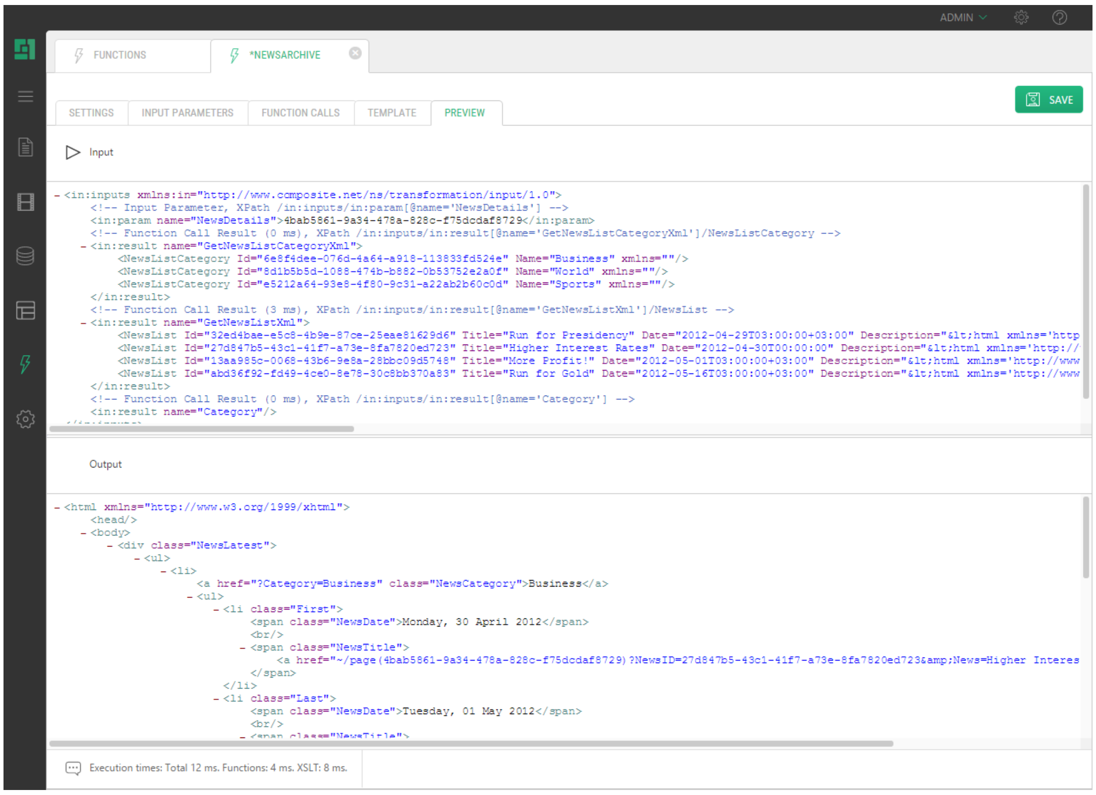
Figure 12: Getting the “?Category=Sport” value from the query string

The logic is implemented in XSLT.

Note: If you have a lot of categories and you filter them, it is better to use the filter on the GetItemXml function.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-
prefixes="xsl in"
        xmlns:in="http://www.composite.net/ns/transformation/input/1.0"
        xmlns="http://www.w3.org/1999/xhtml"
        xmlns:msxsl="urn:schemas-microsoft-com:xslt"
        xmlns:user="urn:my-scripts"
        >

  <msxsl:script language="C#" implements-prefix="user">
    <![CDATA[
            public string FormatDate(DateTime Date)
            {
                  return Date.ToString("dddd, dd MMMM yyyy");
            }
      ]]>
  </msxsl:script>
  <xsl:param name="items" select="/in:inputs/in:result[@name='GetItemXml']"
/>
  <xsl:param name="categories"
select="/in:inputs/in:result[@name='GetCategoryXml']" />
  <xsl:param name="category"
select="/in:inputs/in:result[@name='Category']" />

  <xsl:template match="/">
    <html>
      <head></head>
      <body>
        <div class="NewsLatest">
          <ul>
            <xsl:apply-templates mode="NewsCategory"
select="$categories/*[$category = '' or $category = @Name]">
            </xsl:apply-templates>

          </ul>
        </div>
      </body>
    </html>
  </xsl:template>
  <xsl:template mode="NewsCategory" match="*">
    <xsl:variable name="id" select="@Id" />
    <li>
      <a href="?Category={@Name}" class="NewsCategory" ><xsl:value-of
select="@Name" /></a>
      <ul>
        <xsl:apply-templates mode="NewsItem" select="$items/*[@Category.Id
= $id]">
          <xsl:sort select="@Date" order="ascending"/>
        </xsl:apply-templates>
      </ul>
    </li>
  </xsl:template>

  <xsl:template mode="NewsItem" match="*">
    <li>
      <xsl:if test="position() = 1" >
        <xsl:attribute name="class" >First</xsl:attribute>
      </xsl:if>
      <xsl:if test="position() = last()" >
        <xsl:attribute name="class" >Last</xsl:attribute>
      </xsl:if>
      <span class="NewsDate">
<xsl:value-of select="user:FormatDate(@Date)"/>
      </span><br/>
      <span class="NewsTitle">
        <a
href="~/page({/in:inputs/in:param[@name='NewsDetailsPage']})?NewsID={@Id}&a
mp;News={@Title}">
          <xsl:value-of select="@Title"/>
        </a>
      </span>
    </li>
  </xsl:template>
</xsl:stylesheet>
```

Listing 4: The NewsArchive template code



```
- <in:inputs xmlns:in="http://www.composite.net/ns/transformation/input/1.0">
    <!-- Input Parameter, XPath /in:inputs/in:param[@name='NewsDetails'] -->
    <in:param name="NewsDetails">4bab5861-9a34-478a-828c-f75dcdaf8729</in:param>
    <!-- Function Call Result (0 ms), XPath /in:inputs/in:result[@name='GetNewsListCategoryXml']/NewsListCategory -->
  - <in:result name="GetNewsListCategoryXml">
        <NewsListCategory Id="6e8f4dee-076d-4a64-a918-113833fd524e" Name="Business" xmlns=""/>
        <NewsListCategory Id="8d1b5b5d-1088-474b-b882-0b53752e2a0f" Name="World" xmlns=""/>
        <NewsListCategory Id="e5212a64-93e8-4f80-9c31-a22ab2b60c0d" Name="Sports" xmlns=""/>
    </in:result>
    <!-- Function Call Result (3 ms), XPath /in:inputs/in:result[@name='GetNewsListXml']/NewsList -->
  - <in:result name="GetNewsListXml">
        <NewsList Id="32ed4bae-e5c8-4b9e-87ce-25eae81629d6" Title="Run for Presidency" Date="2012-04-29T03:00:00+03:00" Description="&lt;html xmlns='http
        <NewsList Id="27d847b5-43c1-41f7-a73e-8fa7820ed723" Title="Higher Interest Rates" Date="2012-04-30T00:00:00" Description="&lt;html xmlns='http://
        <NewsList Id="13aa985c-0068-43b6-9e8a-28bbc09d5748" Title="More Profit!" Date="2012-05-01T03:00:00+03:00" Description="&lt;html xmlns='http://www
        <NewsList Id="abd36f92-fd49-4ce0-8e78-30c8bb370a83" Title="Run for Gold" Date="2012-05-16T03:00:00+03:00" Description="&lt;html xmlns='http://www
    </in:result>
    <!-- Function Call Result (0 ms), XPath /in:inputs/in:result[@name='Category'] -->
    <in:result name="Category"/>
```

```
- <html xmlns="http://www.w3.org/1999/xhtml">
    <head/>
  - <body>
      - <div class="NewsLatest">
          - <ul>
              - <li>
                    <a href="?Category=Business" class="NewsCategory">Business</a>
                  - <ul>
                      - <li class="First">
                            <span class="NewsDate">Monday, 30 April 2012</span>
                            <br/>
                          - <span class="NewsTitle">
                                <a href="~/page(4bab5861-9a34-478a-828c-f75dcdaf8729)?NewsID=27d847b5-43c1-41f7-a73e-8fa7820ed723&amp;News=Higher Interes
                            </span>
                        </li>
                      - <li class="Last">
                            <span class="NewsDate">Tuesday, 01 May 2012</span>
                            <br/>
```

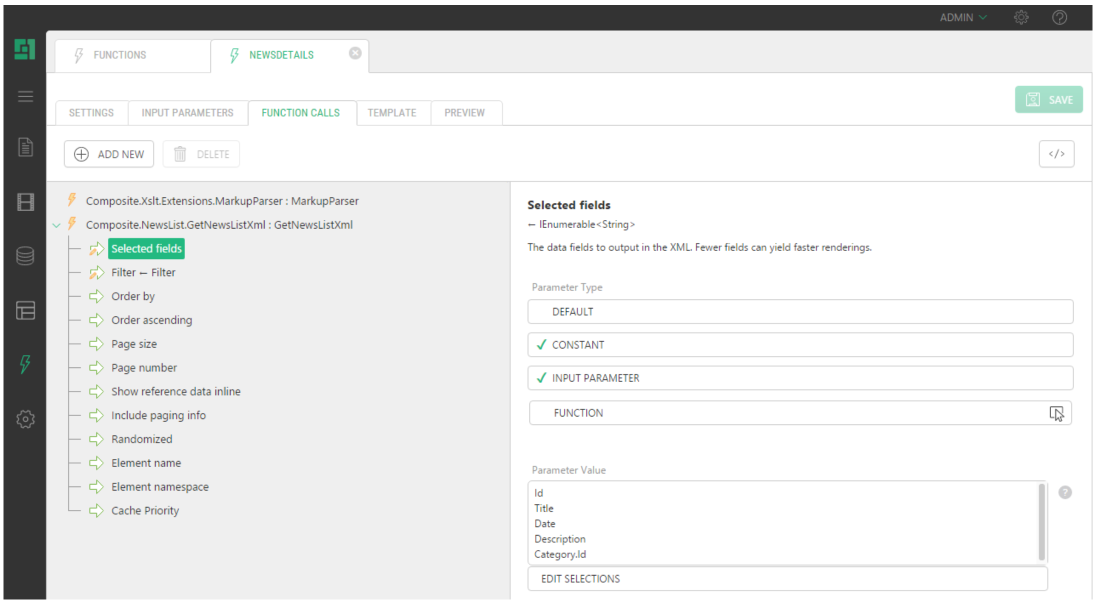
Execution times: Total 12 ms. Functions: 4 ms. XSLT: 8 ms.

Figure 13: Previewing the function

4.2 NewsDetails

This is one of the most interesting parts. We will need filtering and do that using CMS functions. And no programming at all!

We want to get a news article so that we use the Get<Type>XML function to get the XML for our news article and specify which fields we want to retrieve to build our news details:



Composite.Xslt.Extensions.MarkupParser : MarkupParser
Composite.NewsList.GetNewsListXml : GetNewsListXml
- Selected fields
- Filter ← Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Selected fields
← IEnumerable<String>
The data fields to output in the XML. Fewer fields can yield faster renderings.

Parameter Type
DEFAULT
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Value
Id
Title
Date
Description
Category.Id

EDIT SELECTIONS

Figure 14: Selected fields

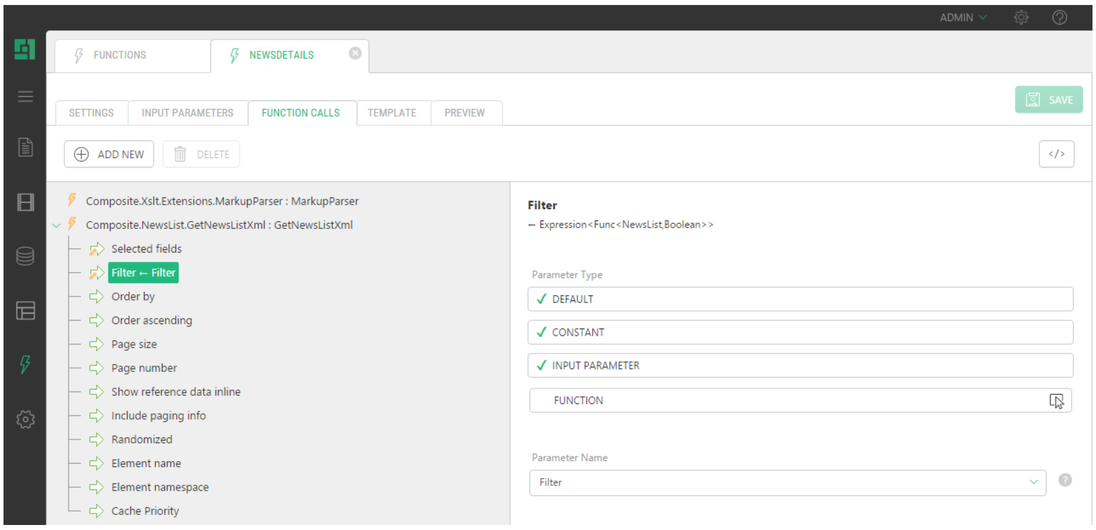
But we don’t want all articles. We only want a specific item, which we specify by using the query string’s news ID. So we will filter to only get that specific News Item element instead of getting all of them

The filter expects the expression function <Func<News Item,Boolean>>.

Basically this expression walks along all the News Item items, evaluates the News Item data and returns a Boolean to say if it passes the filter’s True / False.

We can build such an expression using an input parameter.

After we have built the expression, we need to indicate that we want to use that input parameter by specifying its name. We will call it “Filter” (see the text below the following screenshot). When you set up the “Filter” parameter (see below), you will be able to switch to the Input Parameter type and select the input parameter:



Composite.Xslt.Extensions.MarkupParser : MarkupParser
Composite.NewsList.GetNewsListXml : GetNewsListXml
- Selected fields
- Filter ← Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Filter
← Expression<Func<NewsList,Boolean>>

Parameter Type
✓ DEFAULT
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Name
Filter

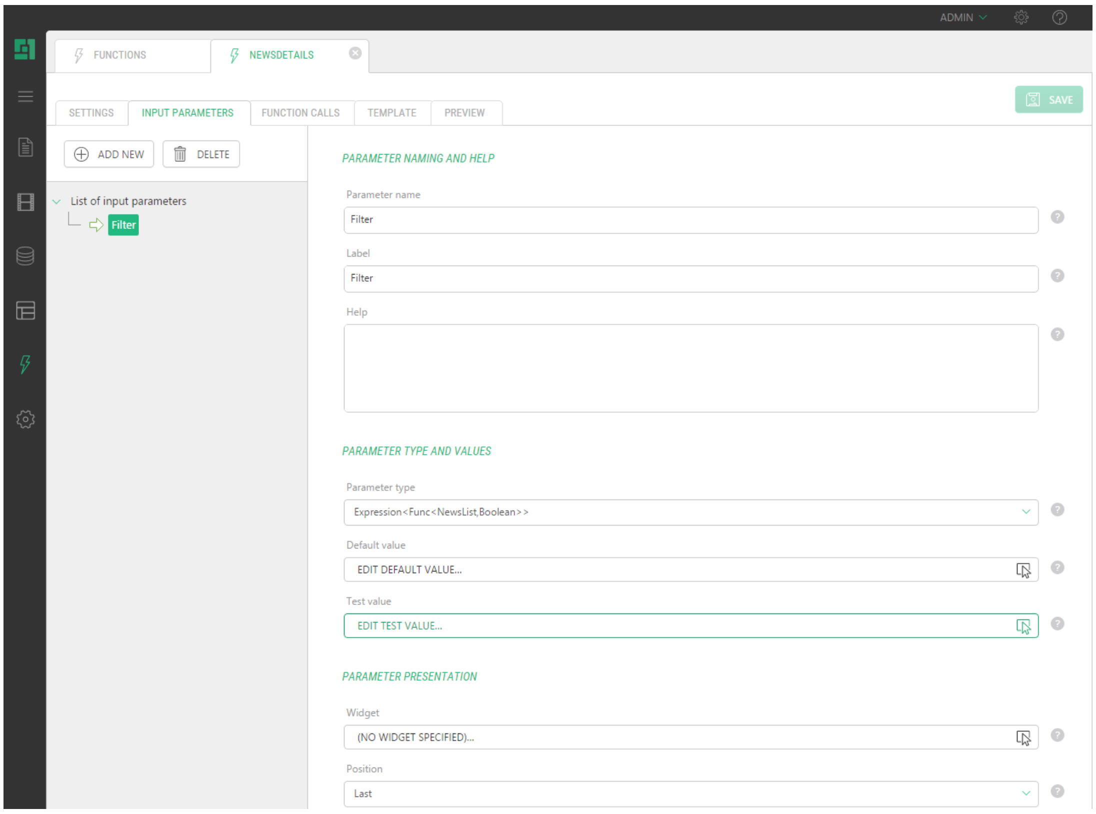
Figure 15: Naming the input parameter

4.2.1 Building a filter expression using an input parameter

So now we need to make an expression function **<func<News Item,Boolean>>** (also called a predicate function in .NET), which will be used as our filter in the GetItemXml function.

We need to give it a name. (In the previous section, we decided to call this input parameter “Filter”).

In the parameter type, select the Expression<Func<News Item,Boolean>>.



PARAMETER NAMING AND HELP

Parameter name: Filter

Label: Filter

Help:

PARAMETER TYPE AND VALUES

Parameter type: Expression<Func<NewsList,Boolean>>

Default value: EDIT DEFAULT VALUE...

Test value: EDIT TEST VALUE...

PARAMETER PRESENTATION

Widget: (NO WIDGET SPECIFIED)...

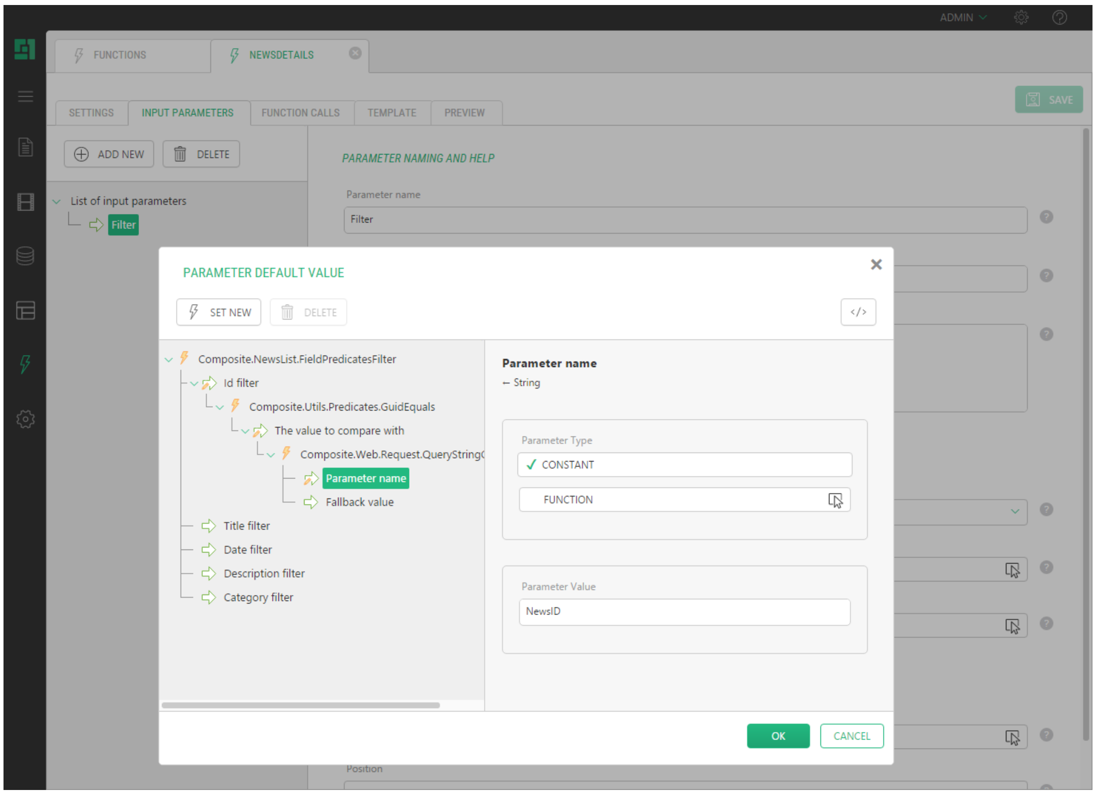
Position: Last

Figure 16: The input parameter expression

We will filter on a specific ID using a GUID.

We make a complex function call and take the GUID from the query string’s NewsID:

1. For the default value, call News Item’s GetPredicatesFilter function.
2. For its IdFilter parameter, call Composite.Utils.Predicates.GuidEquals.
3. For it’s the Value to compare with parameter, call Composite.Web.Request.QueryStringGuidValue and set its Parameter name to “NewsID”.



PARAMETER NAMING AND HELP

Parameter name: Filter

List of input parameters
- Filter

PARAMETER DEFAULT VALUE

- Composite.NewsList.FieldPredicatesFilter
  - Id filter
    - Composite.Utils.Predicates.GuidEquals
      - The value to compare with
        - Composite.Web.Request.QueryString(
          - Parameter name
          - Fallback value
  - Title filter
  - Date filter
  - Description filter
  - Category filter

Parameter name
– String

Parameter Type
✓ CONSTANT
FUNCTION

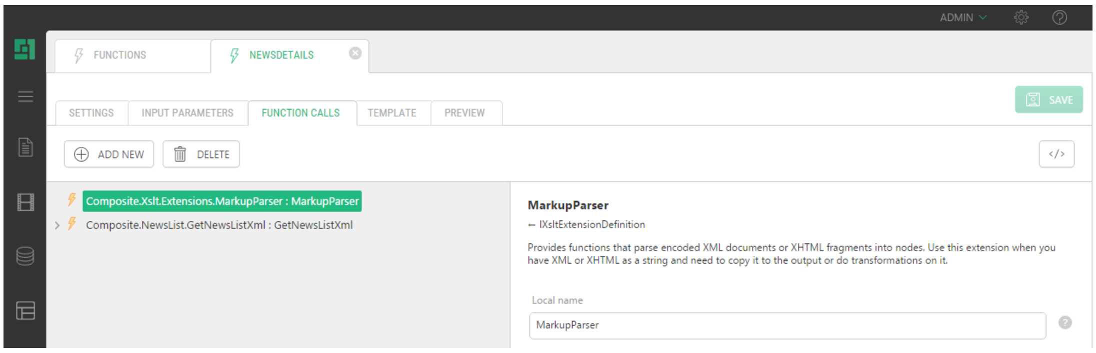
Parameter Value
NewsID

Position

Figure 17: Editing the default value to filter on an ID and using NewsID from the query string

So this is where you could decide not to use the GUID but rather filter on the title (or an extra “no spaces” title field) and get a string from the query string.

That was almost it. We still need the XSLT to create our detailed page out of the XML.



- Composite.Xslt.Extensions.MarkupParser : MarkupParser
- Composite.NewsList.GetNewsListXml : GetNewsListXml

MarkupParser
– IXsltExtensionDefinition

Provides functions that parse encoded XML documents or XHTML fragments into nodes. Use this extension when you have XML or XHTML as a string and need to copy it to the output or do transformations on it.

Local name
MarkupParser

Figure 18: XSLT extension

In our XSLT, we will use an XSLT extension: xmlns:mp="#MarkupParserExtensions" - to parse the HTML from the news article body:

```
<div class="NewsDescription">
  <xsl:copy-of select="mp:ParseXhtmlBodyFragment(@Description)"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"/>
</div>
```

Listing 5: Using the ParseXhtmlBodyFragment markup parser extension

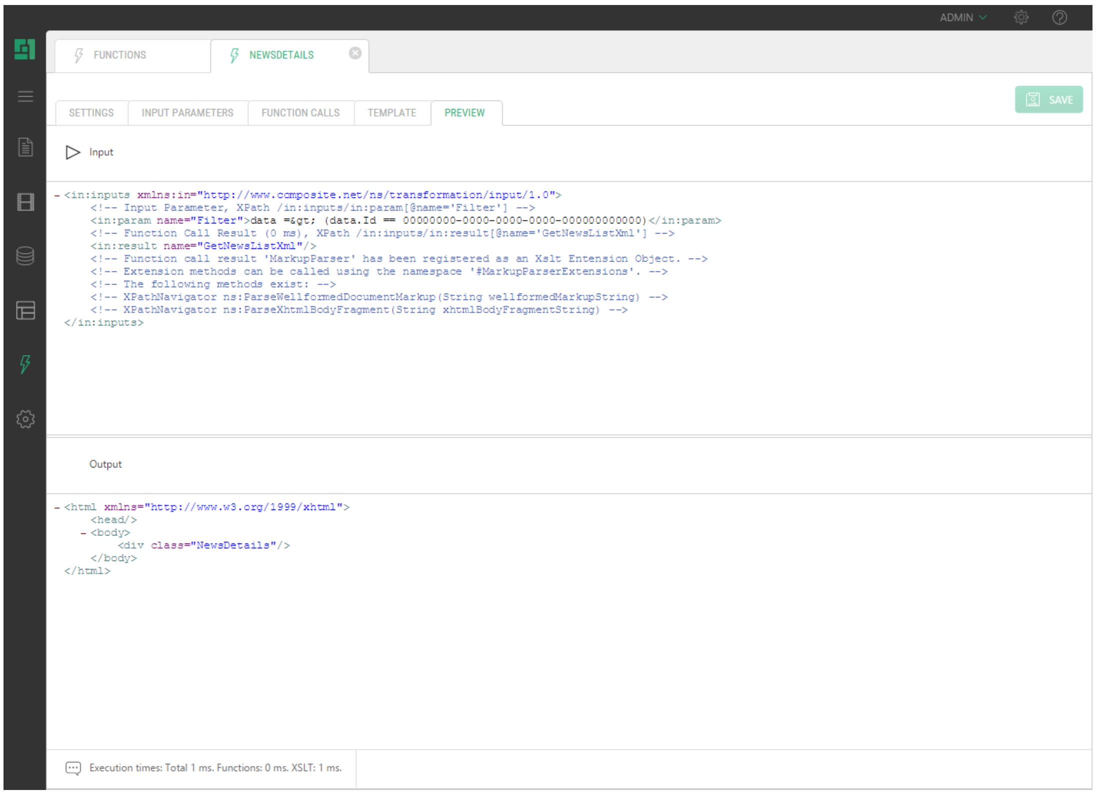
We will also use XSLT to render the news details:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-
prefixes="xsl in"
```

```
        xmlns:in="http://www.composite.net/ns/transformation/input/1.0"
        xmlns="http://www.w3.org/1999/xhtml"
        xmlns:msxsl="urn:schemas-microsoft-com:xslt"
        xmlns:user="urn:my-scripts"
        xmlns:mp="#MarkupParserExtensions"
        >

  <msxsl:script language="C#" implements-prefix="user">
    <![CDATA[
            public string FormatDate(DateTime Date)
            {
               return Date.ToString("dddd, dd MMMM yyyy");
            }
        ]]>
  </msxsl:script>

        <xsl:template match="/">
                <html>
                        <head>
                        </head>
                        <body>
                          <div class="NewsDetails">
                            <xsl:apply-templates mode="NewsItem"
select="/in:inputs/in:result[@name='GetItemXml']/*"></xsl:apply-templates>
                          </div>
                        </body>
                </html>
        </xsl:template>
        <xsl:template mode="NewsItem" match="*">
                <div class="NewsItem">
                        <div class="NewsDate">
                                <xsl:value-of
select="user:FormatDate(@Date)"/>
                        </div>
                        <div class="NewsTitle">
                                <xsl:value-of select="@Title"/>
                        </div>
                        <div class="NewsDescription">
                                <xsl:copy-of
select="mp:ParseXhtmlBodyFragment(@Description)"/>
                        </div>
                </div>
        </xsl:template>
</xsl:stylesheet>
```

Listing 6: NewsDetails

Note: You might want to specify a GUID as a test value so that you could preview your function. In our case, there is nothing to preview since our current filter returns no valid element.



```
- <in:inputs xmlns:in="http://www.composite.net/ns/transformation/input/1.0">
    <!-- Input Parameter, XPath /in:inputs/in:param[@name='Filter'] -->
    <in:param name="Filter">data =&gt; (data.Id == 00000000-0000-0000-0000-000000000000)</in:param>
    <!-- Function Call Result (0 ms), XPath /in:inputs/in:result[@name='GetNewsListXml'] -->
    <in:result name="GetNewsListXml"/>
    <!-- Function call result 'MarkupParser' has been registered as an Xslt Entension Object. -->
    <!-- Extension methods can be called using the namespace '#MarkupParserExtensions'. -->
    <!-- The following methods exist: -->
    <!-- XPathNavigator ns:ParseWellformedDocumentMarkup(String wellformedMarkupString) -->
    <!-- XPathNavigator ns:ParseXhtmlBodyFragment(String xhtmlBodyFragmentString) -->
  </in:inputs>
```

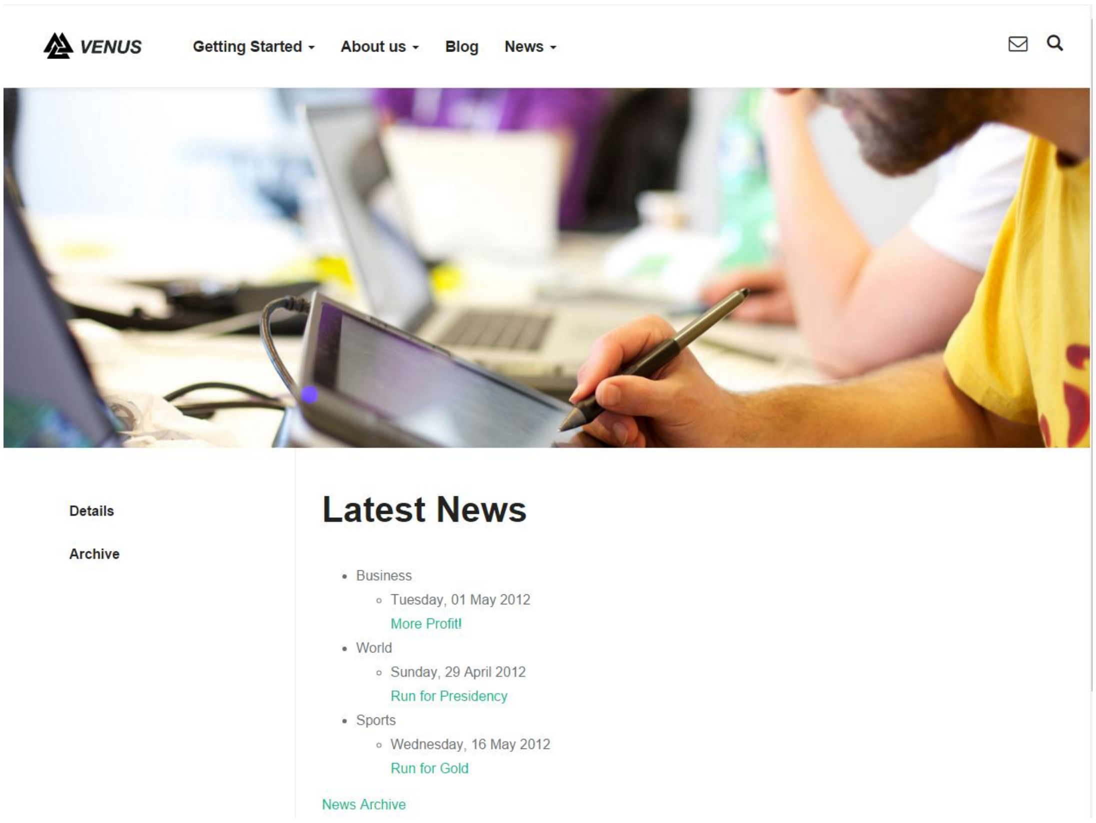
```
- <html xmlns="http://www.w3.org/1999/xhtml">
    <head/>
  - <body>
        <div class="NewsDetails"/>
    </body>
  </html>
```

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

Figure 19: Previewing the function

4.3 NewsLatest

This function is similar to the NewsArchive function, except that it shows a fixed maximum number of articles per category.

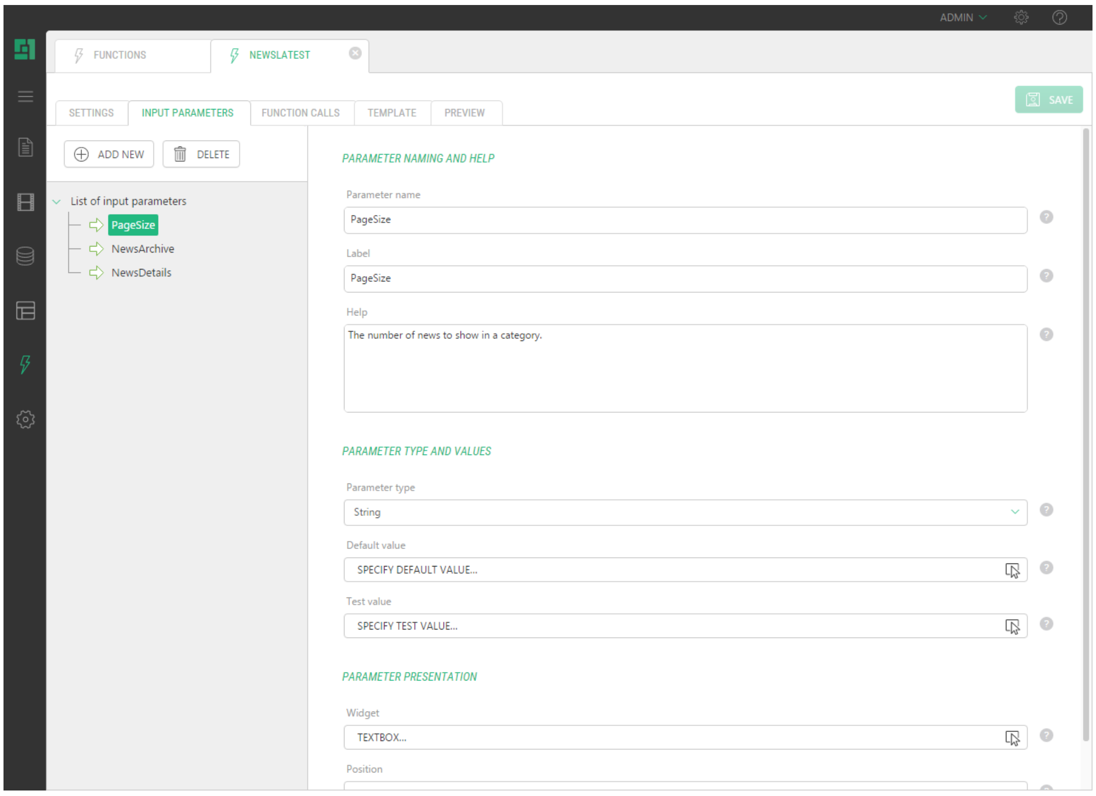


Latest News

- Business
  - Tuesday, 01 May 2012
    More Profit!
- World
  - Sunday, 29 April 2012
    Run for Presidency
- Sports
  - Wednesday, 16 May 2012
    Run for Gold

News Archive

Figure 20: NewLatest with the PageSize parameter set to 1 (1 news in each category)



List of input parameters
- PageSize
- NewsArchive
- NewsDetails

PARAMETER NAMING AND HELP

Parameter name: PageSize

Label: PageSize

Help: The number of news to show in a category.

PARAMETER TYPE AND VALUES

Parameter type: String

Default value: SPECIFY DEFAULT VALUE...

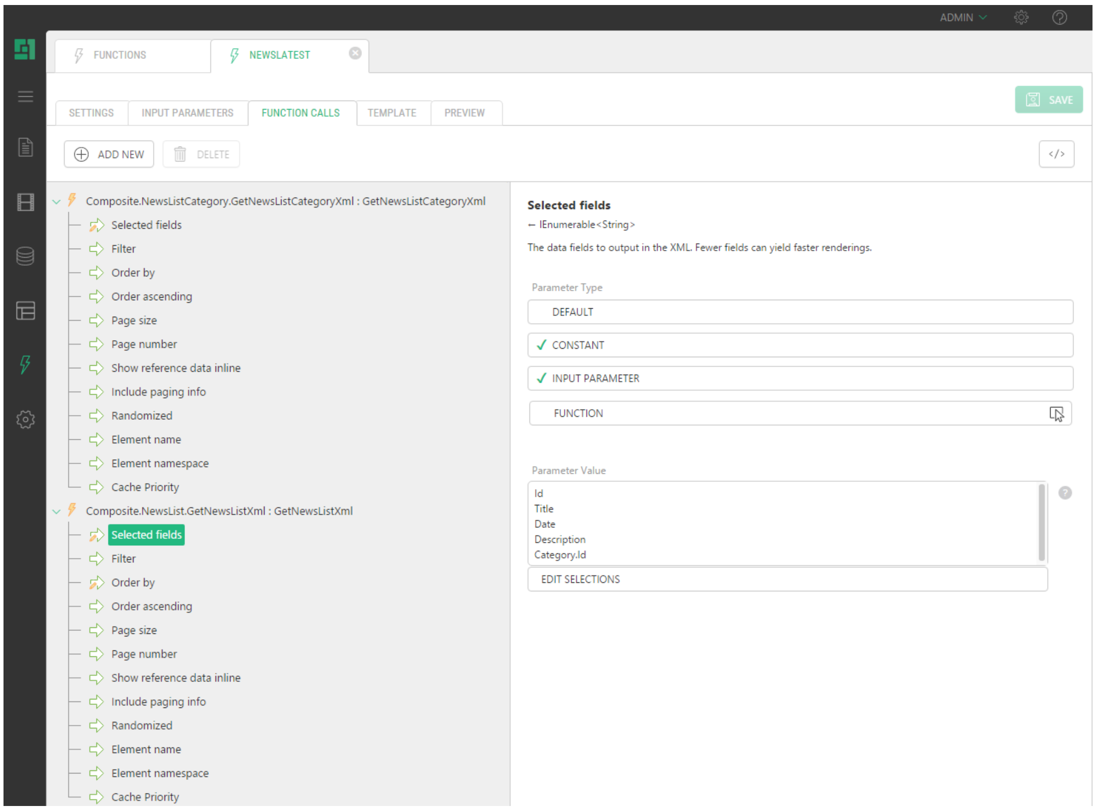
Test value: SPECIFY TEST VALUE...

PARAMETER PRESENTATION

Widget: TEXTBOX...

Position

Figure 21: The PageSize parameter and parameters for hyperlinks



Composite.NewsListCategory.GetNewsListCategoryXml : GetNewsListCategoryXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Composite.NewsList.GetNewsListXml : GetNewsListXml
- Selected fields
- Filter
- Order by
- Order ascending
- Page size
- Page number
- Show reference data inline
- Include paging info
- Randomized
- Element name
- Element namespace
- Cache Priority

Selected fields
– IEnumerable<String>
The data fields to output in the XML. Fewer fields can yield faster renderings.

Parameter Type
DEFAULT
✓ CONSTANT
✓ INPUT PARAMETER
FUNCTION

Parameter Value
Id
Title
Date
Description
Category.Id
EDIT SELECTIONS

Figure 22: Nothing really new here

It is the XSLT that takes care of the counting.

Note: There is no filter in here. If you have loads of news items, this might give the performance you want. In such a case, it would be better to use a filter on the GetItemXml to limit the total number of XML items to work with.

The problem is that if you want 5 items per category and you have 3 categories, you don’t know how many news items you need to retrieve to fill each category up to 5 items.

You could filter on a date. You could use the page size (for example, 25, and hopefully you get enough variation in category news items in there). You could also do it perfectly and filter on the Category and PageSize with your amount but then you will have to add a function for each list.

You should work it out. Only keep in mind that this sample is not what you want to use on something like 1000 news items without changes.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-
prefixes="xsl in"
        xmlns:in="http://www.composite.net/ns/transformation/input/1.0"
        xmlns="http://www.w3.org/1999/xhtml"
        xmlns:msxsl="urn:schemas-microsoft-com:xslt"
        xmlns:user="urn:my-scripts"
        >

  <msxsl:script language="C#" implements-prefix="user">

    <![CDATA[
            public string FormatDate(DateTime Date)
            {
                  return Date.ToString("dddd, dd MMMM yyyy");
            }
```

```
      ]]>

  </msxsl:script>

  <xsl:param name="items" select="/in:inputs/in:result[@name='GetItemXml']"
/>
  <xsl:param name="categories"
select="/in:inputs/in:result[@name='GetCategoryXml']" />
  <xsl:param name="pagesize" select="/in:inputs/in:param[@name='PageSize']"
/>
  <xsl:template match="/">
    <html>
      <head>
        <!-- markup placed here will be shown in the head section of the
rendered page -->
      </head>
      <body>
        <div class="NewsLatest">
          <ul>
            <xsl:apply-templates mode="NewsCategory"
select="$categories/*">
            </xsl:apply-templates>

          </ul>
          <a href="~/page({/in:inputs/in:param[@name='NewsArchivePage']})">
            News Archive
          </a>
        </div>
      </body>
    </html>
  </xsl:template>
  <xsl:template mode="NewsCategory" match="*">
    <xsl:variable name="id" select="@Id" />
    <li>
      <xsl:value-of select="@Name" />
      <ul>
        <xsl:apply-templates mode="NewsItem" select="$items/*[@Category.Id
= $id][position() &gt; count($items/*[@Category.Id = $id]) - $pagesize]">
          <xsl:sort select="@Date" order="ascending"/>
        </xsl:apply-templates>
      </ul>
    </li>
  </xsl:template>

  <xsl:template mode="NewsItem" match="*">
    <li>
      <xsl:if test="position() = 1" >
        <xsl:attribute name="class" >First</xsl:attribute>
      </xsl:if>
      <xsl:if test="position() = last()" >
        <xsl:attribute name="class" >Last</xsl:attribute>
      </xsl:if>
      <span class="NewsDate">
        <xsl:value-of select="user:FormatDate(@Date)"/>
      </span><br/>
      <span class="NewsTitle">
        <a
href="~/page({/in:inputs/in:param[@name='NewsDetailsPage']})?NewsID={@Id}&a
mp;News={@Title}">
          <xsl:value-of select="@Title"/>
        </a>
      </span>
    </li>
  </xsl:template>
</xsl:stylesheet>
```

Listing 7: The NewsLatest template code