A Guide to Function Parameters

2017-02-15

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

www.orckestra.com

C1 CMS

Contents

1 Introduction

Functions play a key role in designing and developing websites in C1 CMS, and their input parameters help make the functions highly adjustable and customizable.

In this guide you will learn how to create function parameters, set default and test values and select proper widgets as well as find out what parameter types are available in C1 CMS.

1.1 Who Should Read This Guide

This guide is intended for developers who want to learn how to create parameters for XSLT, C# inline and SQL functions in C1 CMS.

As a developer, you must be an expert in XML and XSLT, C# or SQL based on what type of functions you create. You should also know how to work with C1 CMS and in its CMS Console.

You need to have access to the Functions perspective with sufficient permissions to create and edit CMS functions. To use the CMS functions on pages and layout templates, you might also need to have access to the Content and Layout perspectives.

1.2 Getting Started

To get started with function parameters, you are supposed to take a number of steps.

Getting Started		
Step	**Activity**	**Chapter or section**
1	Create a function parameter	*Creating Input Parameters*
2	Setting a parameter's properties	*Parameter Properties*
3	Setting the default or test value	*Setting Default and Test Values*
4	Selecting a widget for a parameter	*Selecting Widgets*
5	Working with various parameter types	*Parameter Types*

In the following few chapters, you will learn more about these and other activities.

2 Creating Input Parameters

Input parameters help customize a CMS function's behavior and fine-tune its output.

The values specified in the input parameters can be further used in the function's body.

For example, if your function outputs a list of items, it may output all of them by default. One way is to limit the number of items to display by editing the function's template markup. This would be a hard-coded value, which makes your function inflexible in this respect.

The other and better way is to use a variable in the markup that will get its value from an input parameter set by the user of the function.

In C1 CMS, the following functions can have input parameters:

- XSLT functions
- C# inline functions
- SQL functions

Although these functions differ in the language they are created in, they share the parameter types (simple parameters) and the procedure of creating and using the parameters.

To add an input parameter to a function:

1. Edit a function.
2. Switch to the **Input Parameters** tab.
3. Select **List of input parameters** and click **Add New**. The input parameter's property editor opens on the right.
4. Set its properties where required or necessary.
5. Click **Save**.

The values specified in the input parameters can be further used in function calls and the function's body (XSLT markup, C# code or SQL query).

For information about using parameters within functions, please refer to guides of respective function types.

Now let's have a better look at the parameter properties.

3 Parameter Properties

Each parameter has a number of properties you must or may set:

Parameter naming and help

- **Parameter name**: The name of the parameter. The name is used by the system to identify this parameter. Names must be unique and may not contain spaces and other special characters. Use names like 'Title', 'StartDate', 'LargeImage' etc.
- **Label**: The text that users should see when specifying a value for this parameter. This is the 'human name' for the parameter.
- **Help**: Write a short text that tells the user what to do with the parameter.

Parameter type and values

- **Parameter type**: The type of this parameter.
- **Default value**: You can specify a default value for this parameter. If a parameter has a default value, users are not required to specify it when calling the function.
- **Test value**: When previewing you can test with different input parameter values using this field. If this is left blank, the default value will be used for previews.

Parameter presentation

Widget type: You can select which type of input widget (like a textbox) to use when specifying a value for this parameter. Widgets are only available for simple types.

- **Position**: The position of the parameter. This controls the order of the parameters.

4 Setting Default and Test Values

The purpose of setting default values differs from that of setting test values; however, the steps are the same.

A parameter can often have one or more reasonable values that the user is likely to set the parameter to when using the function. Instead of requiring the user to necessarily set the parameter before using the function, you can set the most likely value for him or her by default, sparing him or her a few clicks in the GUI and thus making the function more user-friendly and usable.

Setting a test value has no effect on the function when the user inserts it in the content; however, it helps debug the function when you are creating it.

To create a default or test value:

1. On the **Input parameters** tab, select the parameter and click in the **Default** (or **Test**) **value** field.
2. In the **Parameter Default** (or **Test**) **value** window in the **Value** field, enter a value.
3. Click **OK**.

Figure 1: Setting the default value

To delete a default or test value:

1. On the **Input parameters** tab, select the parameter and click in the **Default** (or **Test**) **value** field.
2. In the **Parameter Default** (or **Test**) **value** window, click **Advanced View**.
3. Select the function that handles the value (for example, Composite.Constant.Integer) and click **Delete**.
4. Click **OK**.

4.1.1 Using Constants

For simple values, C1 CMS uses one of the matching Constant functions behind-the-scenes:

- Composite.Constant.Boolean
- Composite.Constant.DateTime
- Composite.Constant.Decimal
- Composite.Constant.Guid
- Composite.Constant.Integer
- Composite.Constant.String
- Composite.Constant.XhtmlDocument

You can see what function is used for the parameter by switching to the **Advanced View**.

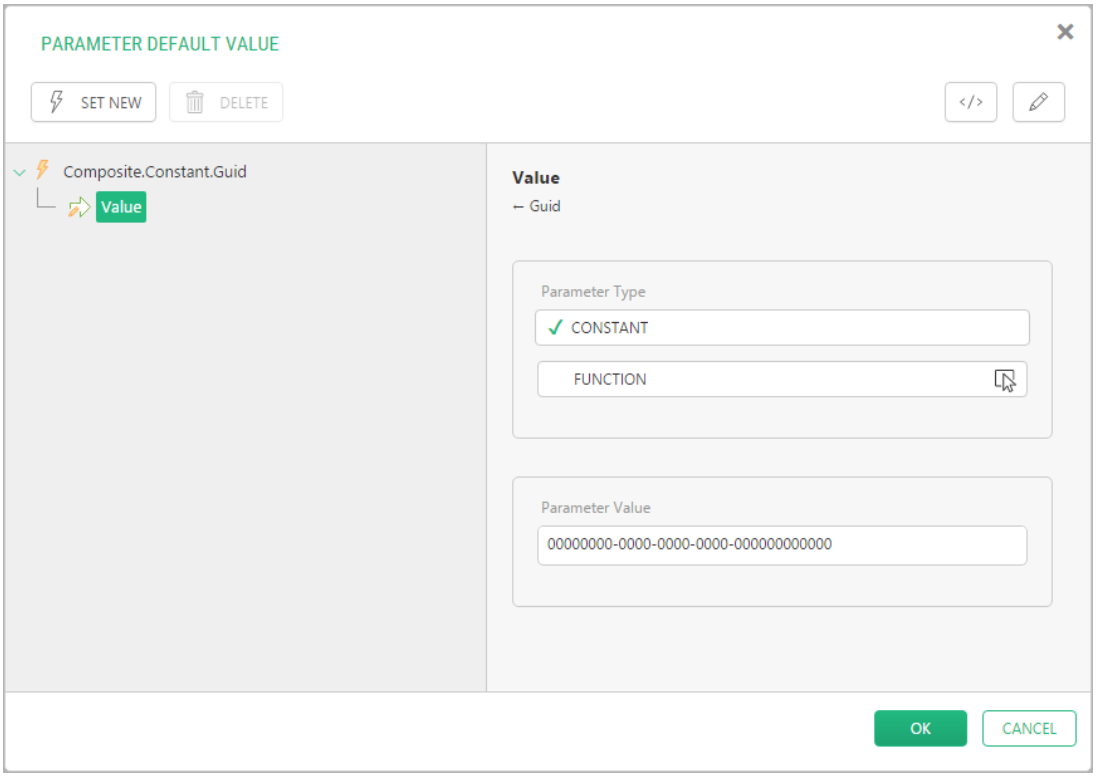

Figure 2: A Constant function in use

4.1.2 Using GetInputParameter

You can use a value from one input parameter in another input parameter by calling the Composite.Utils.GetInputParameter function. The function has one required parameter:

- **Parameter name** (**InputParameterName**): The name of the parameter the value of which you want to use in this parameter.

4.1.3 Using ParseStringToObject

You can specify a string for the input parameter of a specific type (for example, DateTime) and have C1 CMS parse it for you into the corresponding object (DateTime, in the example) - by using Composite.Utils.ParseStringToObject function. The type of the object depends on the input parameter type.

The function has one required parameter:

- **String to parse** (**StringToParse**): The string formatted in a way that can be converted into the type of the expected object.

5 Selecting Widgets

When you select the parameter type, the corresponding widget is pre-selected by default if available. For some types, for example, String, you can select more than one widget.

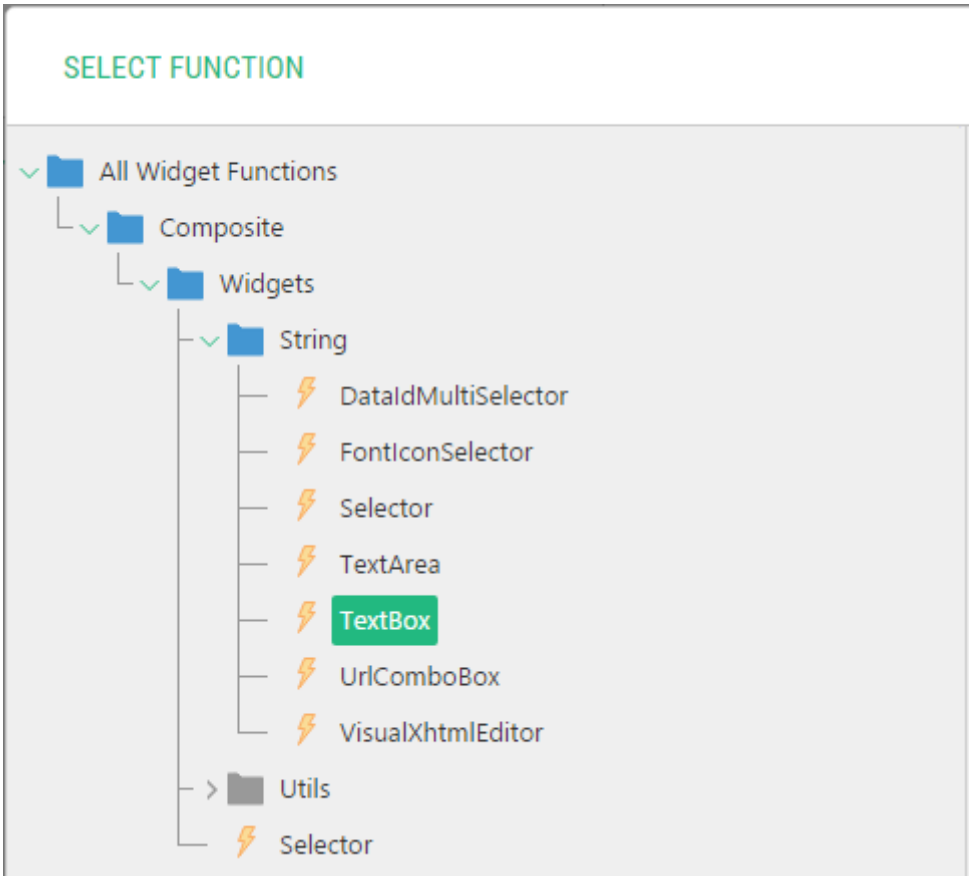

Figure 3: Widgets associated with String

For the function-based parameters, there are no related widgets.

To select a widget:

1. On the **Input parameters** tab, select the parameter and click in the **Widget** field.
2. In the **Parameter Widget** window, click **Add New**. (If a widget has been already selected, select it and click **Delete**.)
3. In the **Select Function** window, expand **All widget functions** and the namespaces the widget belongs to (e.g. Composite.Widgets.Bool).
4. Select the widget (e.g. CheckBox) and click **OK**.
5. In the **Parameter Widget** window, set the parameters of the widget if required or necessary and click **OK**.

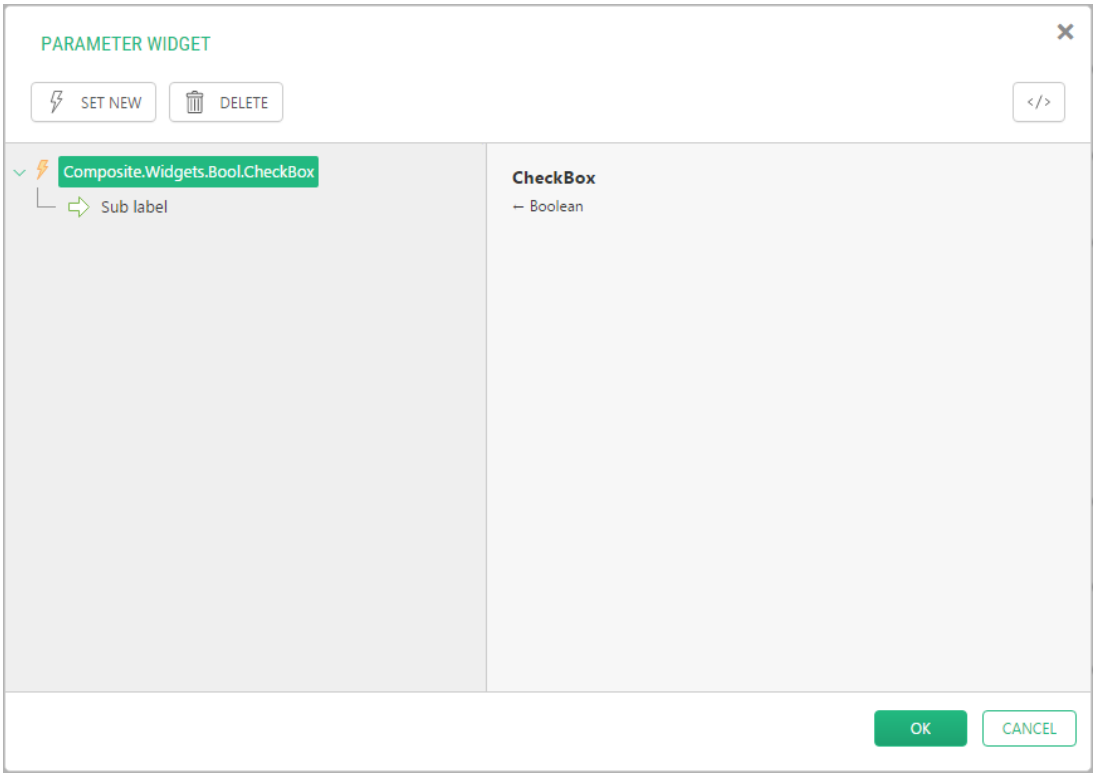

Figure 4: Selecting a widget for the parameter

6 Parameter Types

You can add parameters of various types to your CMS functions.

All CMS functions support simple parameter types. XSLT and C# inline functions also support data reference and other types of parameters.

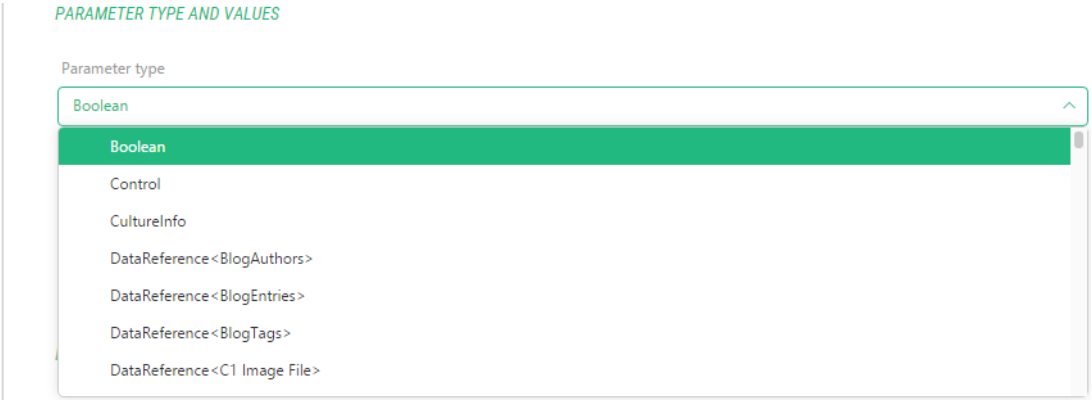

Figure 5: Parameter types listed

The values of many parameters can be set in the GUI as constant values via their associated widgets or as dynamic values via calls to related functions (e.g. String).

The values of some parameter types can only be set via calls to related functions (e.g. UserControl).

In some cases, you may need to use a parameter of specific type if you intentionally want to use the input parameter as a call to a certain function (e.g. Void).

6.1 Simple Parameters

When editing a CMS function, you can use these simple parameter types:

- Boolean
- DateTime
- Decimal
- Guid
- Int32
- String

They are all associated with one or more standard widgets by default and their use is straightforward.

You can also use some related functions to set the values of these parameters. For example, you can concatenate two strings with the Composite.Utils.String.JoinTwo functions for a parameter of the String type instead of using a constant value.

Please note that these are the parameter types SQL functions are limited to.

6.2 Data Reference Parameters

Parameters of the data reference type can reference both built-in and custom data types in C1 CMS. The built-in data types include:

- C1 Page
- C1 Image File
- C1 Media File

- C1 Media Folder

Custom data types are all those global data types, page data folders and page meta types created in C1 CMS.

Data reference parameters come in two flavors:

- DataReference <datatype>
- NullDataReference <datatype>

When used, a DataReference parameter contains a reference to a specific data type and is required.

A NullDataReference parameter can also contain a null reference (i.e. no reference to a specific data type) and thus serves as an optional parameter.

For the built-in data types, ad-hoc selectors are used as widgets:

- PageSelector
- Optional PageSelector
- ImageSelector
- MediaFileSelector
- MediaFileFolderSelector

For both built-in and custom data types, these two selectors are used as widgets:

- Selector
- OptionalSelector

When you specify a custom data type as a parameter type in a function, the selector widget will list all the items of this data type by using the field set as the “Title field” in the data type.

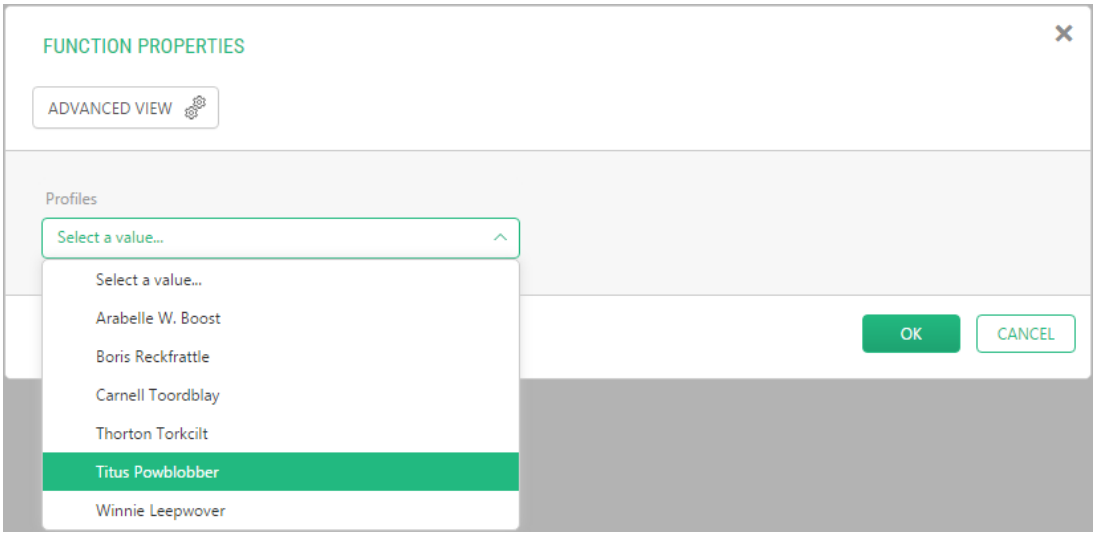

Figure 6: Setting a parameter of the data reference type

6.3 Other Types of Parameters

In addition to the simple parameters, there are two more parameter types whose values can be set via associated widgets in the GUI:

- XhtmlDocument
- Type

And these are the parameters the values of which can be only set with related functions:

- CultureInfo
- Object
- UserControl
- XElement

There are also a few more parameter types used more to make calls to specific functions:

- Void
- XSLT Extension Definitions
- Expression Functions (Predicates)
- Enumerables
- Property Validator Builders

6.3.1 XhtmlDocument

The XhtmlDocument parameter accepts valid XHTML content as its value.

The widget used on this parameter is Composite.Widgets.XhtmlDocument.VisualXhtmlEditor that represents an XHTML editor in both Visual and Source modes.

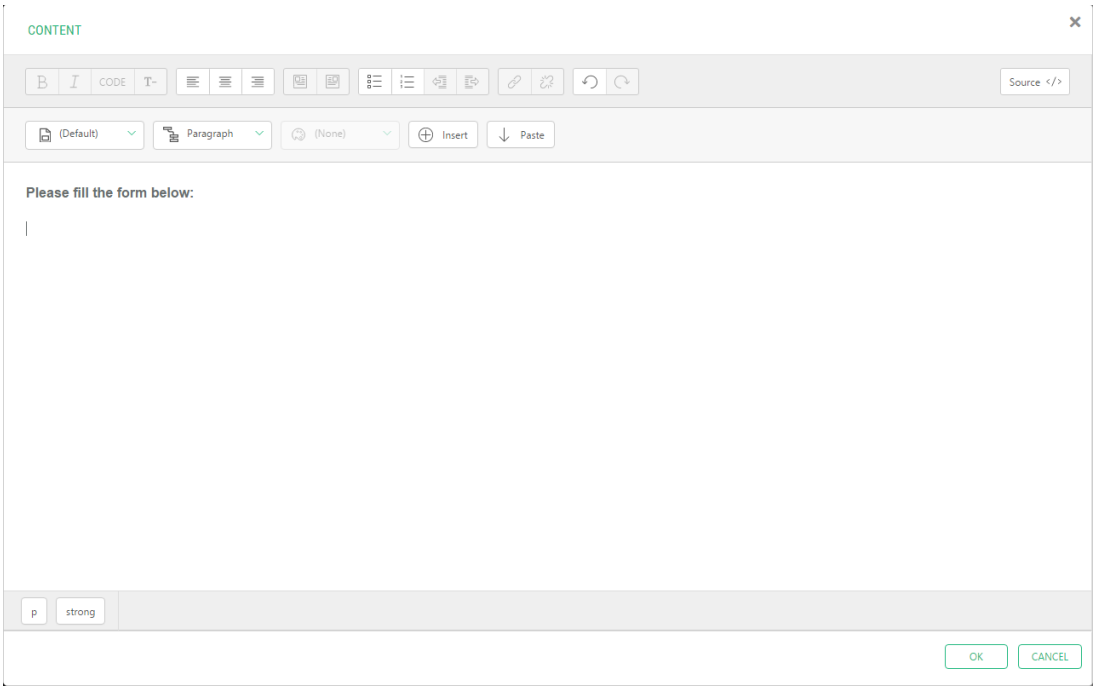

Figure 7: The VisualXhtmlEditor widget

You can use the MarkupParser extension function on the value of this parameter type when rendering input parameters in XSLT.

6.3.2 Type

The Type parameter accepts a data type as its value.

The widget used on this parameter is Composite.Widgets.Type.DataTypeSelector that lists all the data types available in C1 CMS - both built-in and custom. The custom data types include global data types, page data folders and page meta types.

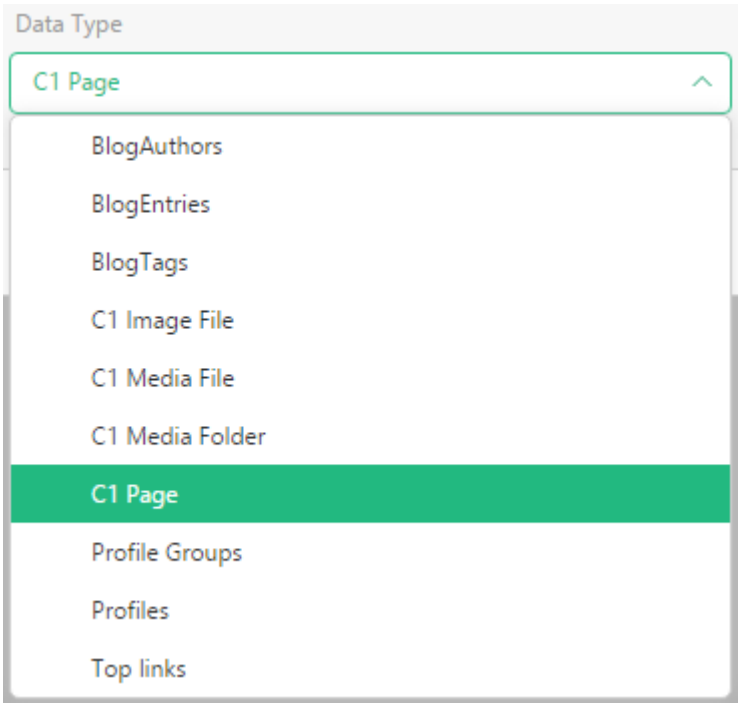

Figure 8: Setting a parameter of the Type type

In the markup, a custom data type must be referred to by its full name.

```
<f:param name="DataType" value="Composite.Community.Blog.Entries"
xmlns:f="http://www.composite.net/ns/function/1.0"/>
```

Listing 1: Setting the value to a custom data type (dynamic)

The static data types as well as the built-in data types have their own pattern for reference: the full name and the name of the assembly.

```
<f:param name="TestField" value="Composite.Data.Types.IImageFile,Composite"
xmlns:f="http://www.composite.net/ns/function/1.0" />
<f:param name="TestField" value="Composite.Data.Types.IMediaFile,Composite"
xmlns:f="http://www.composite.net/ns/function/1.0" />
<f:param name="TestField"
value="Composite.Data.Types.IMediaFileFolder,Composite"
xmlns:f="http://www.composite.net/ns/function/1.0" />
<f:param name="TestField" value="Composite.Data.Types.IPage,Composite"
xmlns:f="http://www.composite.net/ns/function/1.0" />
```

Listing 2: Setting the value to a built-in data type

6.3.3 CultureInfo

The CultureInfo type expects a predefined CultureInfo name as its value (for example, “en-US” or “da-DK”).

To allow the user to specify the CultureInfo name as a string, the Composite.Utils.ParseStringToObject function should be used.

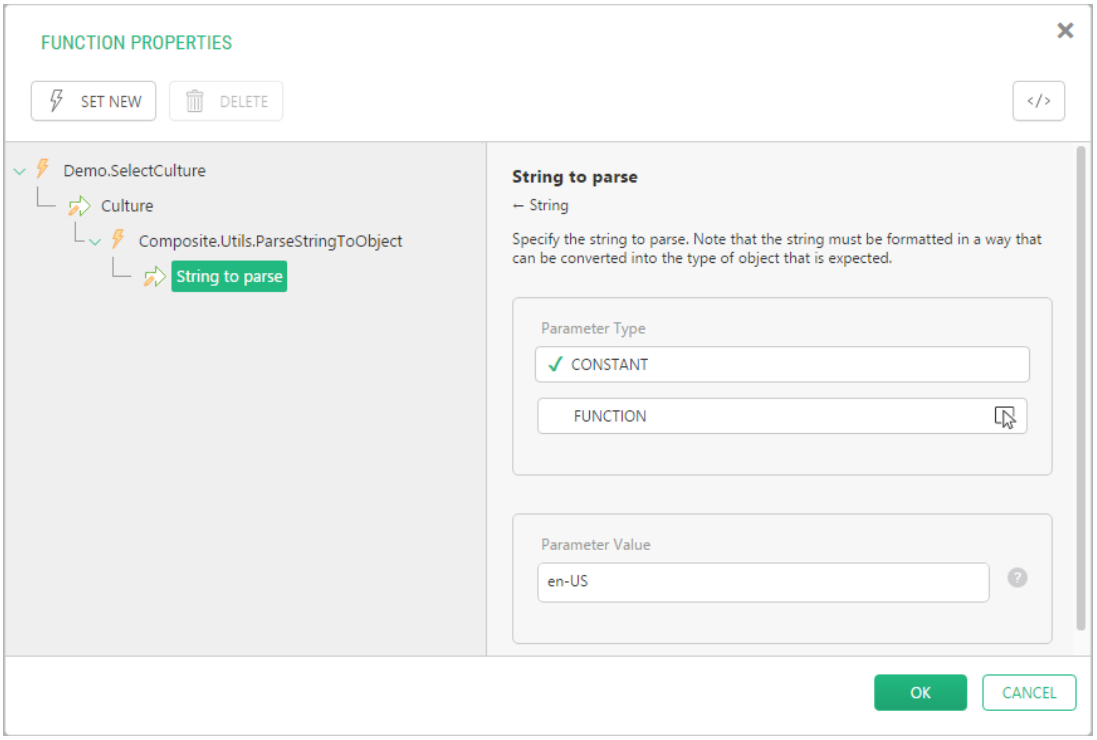

Figure 9: Parsing a string to a CultureInfo object

If necessary, the parameter can be also set to the CultureInfo currently used in the system with the Composite.Utils.Globalization.CurrentCulture.

6.3.4 Object

The Object is a generic parameter type used for input parameters whose value cannot be defined at the design time, and thus, can basically be of any type. Since the Object type is the parent type in .NET, it makes sense to use it as the most common type.

6.3.5 UserControl

The UserControl type expects an ASP.NET User Control as its value. The value can be set by calling the Composite.AspNet.LoadUserControl function, which loads an ASP.NET User Control (.ascx file) from a specific path, for example, “~/Controls/MyControl.ascx”.

6.3.6 XElement

The XML type expects an XML element, the fundamental XML construct, as its value. The XML element can have no, one or more attributes and include content made up by other XML element. In general, the parameter of this type expects valid XML.

To set its value, you can use of these CMS functions:

- **Composite.Core.Xml.LoadFile**: Loads a local XML file at a specific relative path
- **Composite.Core.Xml.LoadUrl**: Loads a remote XML file by a specific URL

Custom functions that return a value as an XElement can be also used.

6.3.7 Void

You can use a function to provide a value for the input parameter. The function must return a value of the same type as selected for the input parameter. For example, if you set the input parameter type to “string”, the functions that return a string will be available in the Select Function window. Others will be filtered out.

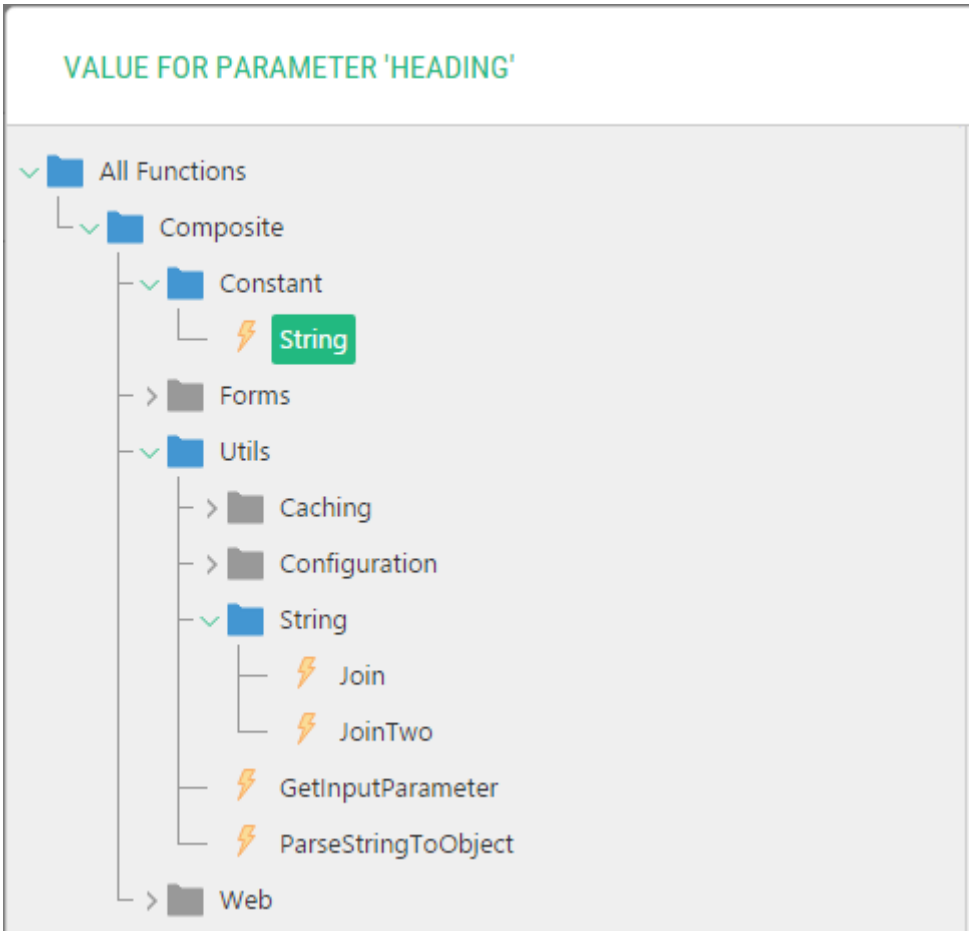

Figure 10: Functions filtered based on the parameter type (String)

Some CMS functions return no value. In fact, what they return is “void” (“null” as a matter of fact). This ensures that they are never suggested as functions for any specific type you can set the input parameter to.

If you need to use one of these functions in the input parameter (for example, Web.Response.Redirect), you should set the input parameter to the Void type.

6.3.8 XSLT Extension Definition

A parameter of the XSLT extension definition type expects an XSLT extension definition as its value.

The value (the extension itself) is not added to the input XML document, but rather hooked into the list of extensions that can be called from XSLT. You can “inject” these extensions into the CMS function system, and then call the injected classes from XSLT.

If any XSLT extensions are available in C1 CMS, you can specifically select them as an input parameter:

XsltExtensionDefinition<extension>

For example, XsltExtensionDefinition<MailingListXsltExtensions>

The IXsltExtensionDefinition type as an input parameter has no XSLT extension definition associated with it at the design time and must be defined when the function with this parameter is actually used. All the XSLT extension definitions will be thus available. For example, Composite.Xslt.Extensions.DateFormatting.

For information about making custom XSLT extension definitions, please refer to "Adding an XSLT Extension".

6.3.9 Function Expressions (Predicates)

A parameter of the function expression type expects a function expression (expression tree) as its value.

As parameters in CMS functions, the function expressions can be called with the following simple types: Boolean, DateTime, Decimal, Guid, Int32, String as well as all the data types available in C1 CMS - both built-in and custom. (They can be also called with nullable counterparts of Boolean, DateTime, Decimal, Guid, Int32.)

- Expression <Func <[type], Boolean>>
- Expression <Func <[datatype], Boolean>>
- Expression <Func <Nullable<[type]>, Boolean> >

They always return a Boolean.

Based on the type the expression is called with, specific expression-building functions are available for it (Composite.Utils.Predicates).

For example, with Expression <Func <String, bool>>, you can use the following functions:

- StringContains
- StringEndsWith
- StringEquals
- StringInCommaSeparatedList
- StringInList
- StringNoValue
- StringStartsWith

For data types, a corresponding FieldPredicatesFilter is used to give access to expression-building functions on each field according to the type of the latter.

The function expressions are a convenient tool to filter data provided by data-centric functions (Get<Type>Xml).

For example, it can be used to retrieve, render and present only one data item based on its ID as a query parameter in the URL.

6.3.10 Enumerables

For an input parameter that holds a list of read-only values, you should use an enumerable type - IEnumerable. IEnumerable implements an enumerable list of strings, XML elements (XElement) or CultureInfo objects:

- IEnumerable<string>
- IEnumerable<XElement>
- IEnumerable <CultureInfo>

Its value can be set with a function call.

For example, the Composite.Utils.String.Split function returns an enumerable list of strings and Composite.Utils.Globalization.AllCultures returns an enumerable list of CultureInfo objects.

Functions that return XML can provide enumerable lists of XElements. The data-centric functions (Get<Type>Xml) can be their example.

In the function markup, when the parameter type is IEnumerable, you should use one or more "paramelement" elements to list the fields to include the values of in the output.

```
<f:function xmlns:f="http://www.composite.net/ns/function/1.0"
name="Composite.Data.Types.IPage.GetIPageXml">
  <f:param name="PropertyNames">
    <f:paramelement value="Id" />
    <f:paramelement value="Title" />
  </f:param>
</f:function>
```

Listing 3: Using paramelement elements with the IEnumerable parameter

6.3.11 Property Validators Builders

As its value, the property validator builder type expects an object capable of validating a value of a simple type such as DateTime, Decimal, Guid, Int32 and String. Hence, there are five property validator builder types available in C1 CMS:

- PropertyValidatorBuilder <DateTime>
- PropertyValidatorBuilder <Decimal>
- PropertyValidatorBuilder <Guid>
- PropertyValidatorBuilder <Int32>
- PropertyValidatorBuilder <String>

Their value should be set by invoking one of the Composite.Utils.Validation functions.

For each validated simple type, there is at least one validation function, which ensures that the value is not null (*NotNullValidation), for example, StringNotNullValidation.

Besides, there are a few validation functions specific to the Decimal, Integer and String types:

- DecimalPrecisionValidation
- IntegerRangeValidation
- RegularExpressionValidation
- StringLengthValidation

7 Special Uses of Parameters

In this guide XSLT functions are used for examples of creating CMS function parameters. As you know, the CMS Functions are not limited to XSLT and you can use, for example, Razor functions, which have their own way of setting parameters.

Besides, you can programmatically create and set the parameters.

You can use Dependency Injections to define and set parameters of specified types automatically.

Please see “Dependency Injections on Parameters” for more information.

You can use the parameter of the RoutedData<T> type to quickly build a list-to-details view for a data type with a CMS function.

Please see “Data URL Routing” for more information.