CTC Configurator Framework

Version 2.0.3

Client Tools Consultancy

Table of Contents

1   Introduction

1.1   What is CTC Configurator Framework?

The CTC Configurator environment is a Graphical User Interface for configuring options and features for the generators from Client Tools Consultancy.

The Configurator environment consists of a generic framework and an add-in for each generator. Each generator from CTC is delivered with a Configurator specifically for that generator which adds-in to the Configurator Framework.

The Configurator Framework provides the functionality for managing the user interface and the add-in's.

Depending on the generator being used, this document should be read in conjunction with following documents:

- **CTC ASP.NET WebForms Configurator** document
- **CTC ASP.NET WebForms Generator** document
- **CTC Silverlight Client Configurator** document
- **CTC Silverlight Client Generator** document
- **CTC Smart Client Configurator** document
- **CTC Smart Client Generator** document
- **CTC WCF Services Configurator** document
- **CTC WCF Services Generator** document
- **CTC WPF Client Configurator** document
- **CTC WPF Client Generator** document

1.2   The Concept

A configuration consists of configurable elements, and options and features that can be specified for each element. Configurable elements are generator, application, bundle, language, ispec and field, which are specified in a hierarchy as illustrated below.

Options and features can be configured for any element at any level in a hierarchy and will automatically be inherited by the levels below. An option or a feature can be reconfigured at any level in the hierarchy.

The following illustrates the hierarchy in general:
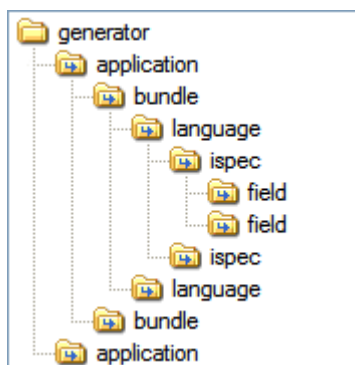
The above diagram shows:

- A generator can have one or more applications and whatever is configured at the generator level will automatically be applied to any application within the generator.

- An application can have one or more bundles and whatever is configured at the application level will automatically be applied to any bundle within the application.

- A bundle can have one or more languages and whatever is configured at the bundle level will automatically be applied to any language within the bundle.

- A language can have one or more ispecs and whatever is configured at the language level will automatically be applied to any ispec within the language.

- An ispec can have one or more fields and whatever is configured at the ispec level will automatically be applied to any field within the ispec.

- A field is the lowest level in the hierarchy that can be configured.

A hierarchy like this provides a very flexible way of configuring options and features. This allows options and features to be specified once and implemented consistently throughout all applications, bundles, languages, ispecs and fields. On the other hand, if an ispec or a field requires a specific option or feature, it is specified at a lower level.

To make the configuration even more flexible, it is also possible to configure fields and ispecs globally. The following shows the complete picture of the hierarchy including fields and ispecs at a global level:
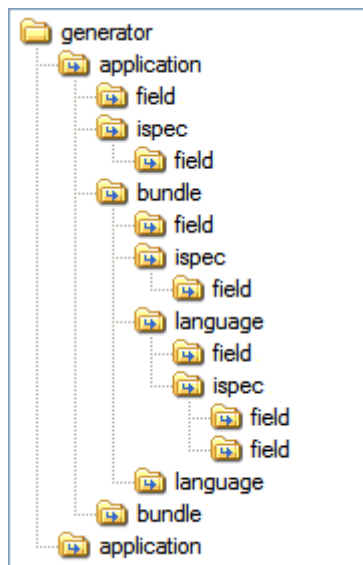

The above diagram shows:

- A field can be configured globally at the application level and this configuration will be applied to the field in whichever ispecs this field occurs when generating a bundle for that application.

- A field can be configured at the bundle level and this configuration will be applied to the field in whichever ispecs this field occurs when generating the bundle.

- A field can be configured at the language level and this configuration will be applied to the field in whichever ispecs this field occurs when generating the bundle.

- A field can be configured at the ispec level and the configuration will be applied to the field wherever this field occurs within the ispec when generating a bundle.

- An ispec can be configured globally at the application level and this configuration will be applied to the ispec in whichever bundle this ispec occurs when generating a bundle for that application.

- An ispec can be configured at the bundle level and this configuration will be applied to the ispec wherever this ispec occurs within the bundle when generating the bundle.

- An ispec can be configured at the language level and this configuration will be applied to the ispec wherever this ispec occurs within the bundle when generating the bundle.

The options and features available depend on the generator. For information about options and features that can be configured for a specific generator, please refer to the help documentation for the Configurator associated with the generator.

2   The User Interface

The Configurator uses an xml file as the storage media for the configuration details. The user interface, as illustrated below, provides assistance for entering and editing the configuration details as well as ensuring the integrity of the details kept in the configuration file.
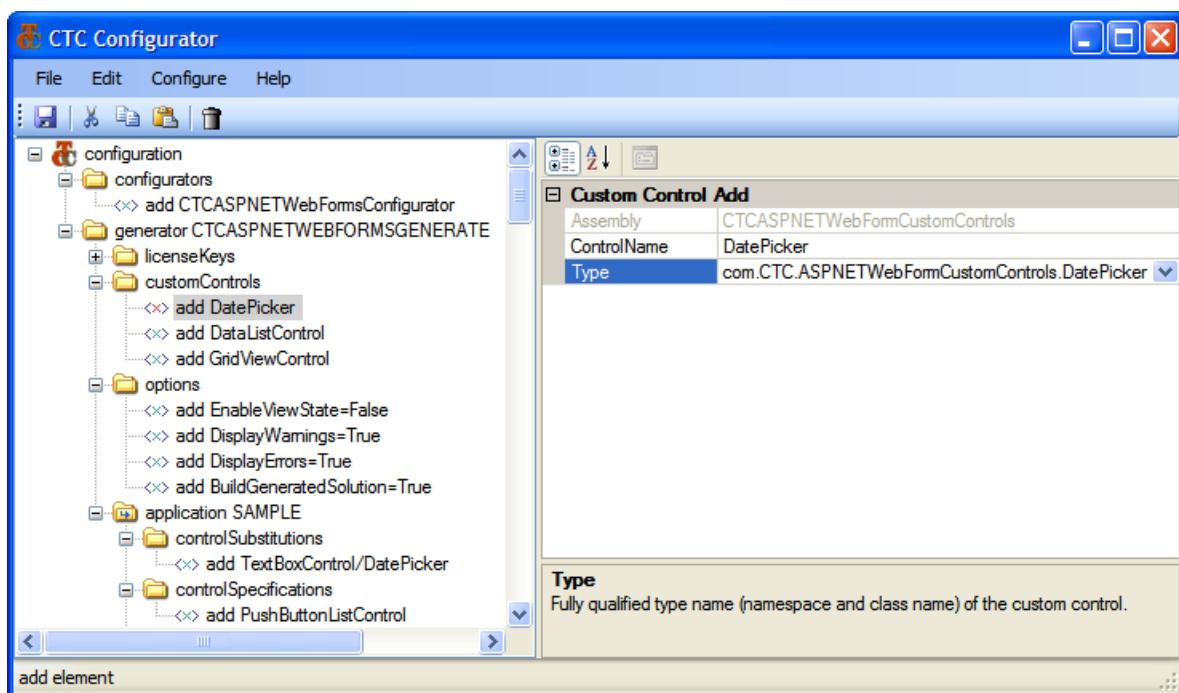

The left hand side of the user interface provides a view of the configuration file and displays the information in a tree structure to show the hierarchical relationship of the various elements.

The right hand side displays the properties of a configuration element when selected. It allows for entry of values, where possible by selection from a dropdown list of valid suggestions.

To make it easy to add and delete configuration elements, a context menu is provided for each configuration element in the tree structure.

2.1   Main Menu Items
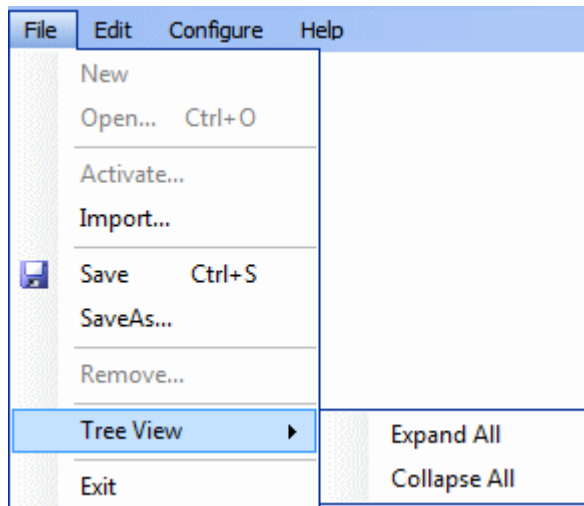
The **New** function creates a new configuration. This function is only available when the Multi Configuration Files feature is activated.

The **Open** function opens an existing configuration file. This function is only available when the Multi Configuration Files feature is activated.

The **Activate** function activates a configuration. A configuration marked as active will be used by the generator. This function is only available when the Multi Configuration Files feature is activated.

The **Import** function imports a configuration file. This activates the Multi Configuration Files feature.

The **Save** function writes the current state of the configuration information to the current open configuration file. When the Multi Configuration Files feature is not activated, a file named CTCGeneratorConfig.xml located in the same directory as the Configurator executable, which is the [CERoot]\bin directory, will be created.

The **SaveAs** function saves a configuration to a file. This activates the Multi Configuration Files feature.

The **Remove** function removes a configuration from the list of available configurations. This function is only available when the Multi Configuration Files feature is activated.

The **Tree View/Expand All** function expands all child nodes of the current selected node in the tree structure.

The **Tree View/Collapse All** function collapses all child nodes of the current selected node in the tree structure.

The **Exit** function terminates the Configurator. The user will be warned if changes have not been saved on exit.
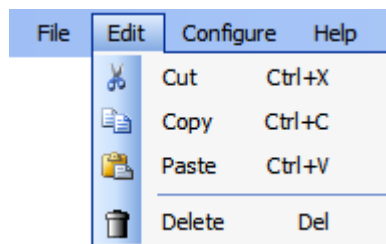
The **Edit** menu provides the standard Cut, Copy, Paste and Delete of configuration nodes in the tree view. In addition, Drag and Drop is also available for moving and copying configuration nodes. Holding down the Ctrl key while dragging and dropping a node causes the node to be copied. The default behaviour when not pressing the Ctrl key is to move the node.

Cut/Copy/Paste and Drag/Drop is only available within the same configuration file. It cannot be performed across multiple configuration files.
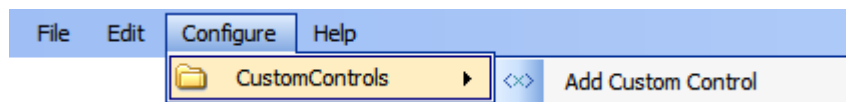
The **Configure** menu is context sensitive and changes depending on the configuration element selected in the tree structure.
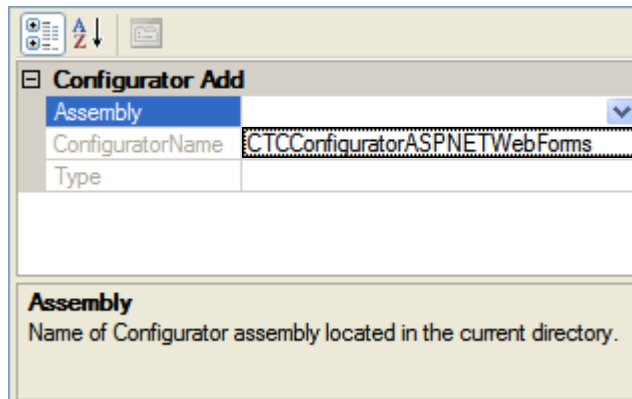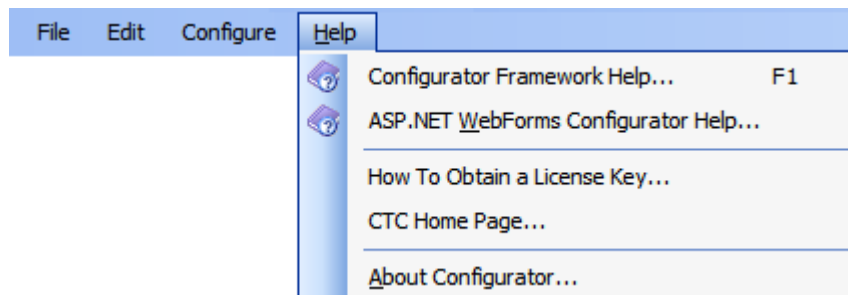

3   Configurator Add-In's

Each generator from CTC has its own unique Configurator. After installing a generator and the Configurator Framework, the first task is to add the Configurator for the generator you wish to configure. A Configurator add-in is a .dll file located in the same directory as the Configurator executable.

3.1   Adding a Configurator

A Configurator is added by right-click on the 'configurators' node in the tree structure and selecting the 'Add Configurator' item from the context menu. This provides the following properties in the property window:

The dropdown list of the Assembly property is pre-filled with a list of Configurator assemblies found in the current directory.

When selecting the appropriate Configurator assembly, the ConfiguratorName and Type properties will automatically be filled. In addition, a 'generator' node for this Configurator will automatically be added to the tree structure and a menu item providing access to the Help Documentation for the Configurator will be added to the Main Help menu.

When changing/renaming the Assembly property of an existing Configurator, the 'generator' element associated with the 'old' Configurator will be unloaded. This means the 'generator' element will be removed from the tree structure, however, it will not be deleted from the xml config file. When later adding the same Configurator again, the 'generator' element will be re-inserted in the tree structure. Warnings asking for confirmation to proceed with the renaming will popup.

3.2   Deleting a Configurator

A Configurator is deleted by right-click on the 'add [configurator-name]' element and selecting the 'Delete' item from the context menu.

When deleting a Configurator, the entire 'generator' element associated with the Configurator will be deleted. Warnings asking for confirmation to proceed with the deletion will popup.

A 'generator' element cannot be deleted directly. As described, selecting and deleting the Configurator associated with the 'generator' element will delete the generator.

4   Location of Config File

The configuration file named 'CTCGeneratorConfig.xml' is located in the bin sub-folder of the Component Enabler Root directory. E.g.:

C:\CE3.3\bin\CTCGeneratorConfig.xml