Creating Custom Settings in Wealth-Lab Pro®

Introduction

In Wealth-Lab Pro®, several object types can implement custom settings that can then be modified by the user at run time. Custom settings are either mandatory or optional, depending on the extension class being implemented. Some of the extendable types that can implement custom settings are:

- Chart Styles
- Chart Drawing Objects

When implementing a class that extends Wealth-Lab Pro in a particular way, for example creating a new Chart Style, you typically either derive your class from a certain base class or implement a specific interface. For your extension class to also implement custom settings, it must also implement the **ICustomSettings** interface.

The **ICustomSettings** interface serves two purposes:

1. It exposes a user interface (in the form of a UserControl) that displays the settings and allows the user to modify them.
2. It allows the settings values to be persisted and initialized.

The interfaces described below are contained in the Fidelity.Components.dll assembly, and in the Fidelity.Components namespace.

ICustomSettings interface

The interface contains four methods that you must implement in your class.

```
UserControl GetSettingsUI();
```

You should return an instance of a **UserControl** that contains the GUI the represents your class' custom settings. The size of the **UserControl** is dependant on the extension that you are providing custom settings for, see the documentation for the specific extension for guidelines. Generally, the **UserControl's** width should conform to the recommended guideline, but Wealth-Lab Pro will dynamically resize the height dialog box or control that will host the UserControl.

Before returning the **UserControl**, be sure to set its user interface elements so that they correctly represent the current state of the settings in the object itself.

```
void ChangeSettings(UserControl ui);
```

Wealth-Lab Pro will call this method when the user has made changes to the user interface that you provided above. You are passed back the instance of the **UserControl** that you returned in the call to **GetSettingsUI**. At this point you should read the user interface elements on the **UserControl** and update the internal state of the object based on the changed settings.

```
void ReadSettings(ISettingsHost host);
```

Wealth-Lab Pro will call this method when your object needs to initialize its custom settings. Use the **ReadSettingsValue** method in the supplied **ISettingsHost** parameter to read the settings one by one and set the internal state of your object.

```
void WriteSettings(ISettingsHost host);
```

Wealth-Lab Pro will call this method when it needs to persist the custom settings in your object. Call the **WriteSettingsValue** method in the supplied **ISettingsHost** to persist each of the custom settings your object maintains.

ISettingsHost interface

This interface allows some or all of the custom settings that your class maintains to be persisted. Wealth-Lab Pro uses this mechanism to save selected custom settings to the application settings file, and to make sure the previously selected settings values are applied to new objects of the same type that are created. ISettingsHost provides a number of overloaded **Get** and **Set** methods that allow you to get and set a particular settings value, by key, for various data types.

```
bool ContainsKey(string key);
```

Allows you to query whether a particular settings value was ever persisted to the settings data file.

```
string Get(string key, string defaultValue);
void Set(string key, string value);
```

Reads/writes settings that are **String** values.

```
bool Get(string key, bool defaultValue);
void Set(string key, bool value);
```

Read/writes settings that are **Boolean** values.

```
Color Get(string key, Color defaultValue);
void Set(string key, Color color);
```

Reads/writes settings that are **System.Drawing.Color** values.

```
int Get(string key, int defaultValue);
void Set(string key, int value);
```

Reads/writes settings that are **Int32** values.

```
double Get(string key, double defaultValue);
void Set(string key, double value);
```

Reads/writes settings that are **Double** values.

```
DateTime Get(string key, DateTime defaultValue);
void Set(string key, DateTime value);
```

Reads/writes settings that are **DateTime** values.

```
Font Get(string key, Font defaultFont);
void Set(string key, Font value);
```

Reads/writes settings that are **System.Drawing.Font** values.

```
bool Get(Form frm, string key);
void Set(Form frm, string key);
```

Reads and writes a form's location, size, maximized state and Tag property (as a string).