AdvancedFormatProvider class and plug-ins

For platform: Microsoft .net 1.1 and higher.

AdvancedFormatProvider is an IFormatProvider which means it works with String.Format() and other consumers of this interface. For example:

```
text = String.Format(AdvancedFormatProvider.Current, "{0:C-$}", 100.0)
```

It supports all existing formats supplied by the .net framework, plus those included with this product and you create.

AdvancedFormatProvider makes it easy to drop in new formatters, tools that convert a value into a string based on a format string in the `{#:here}` token. (For example, `{0:c}` and `{1:yyyy-MM-dd}`.)

The formatters are registered globally so you only install them once, in Application_Start.

It pre-installs support for these formatters:

- **IntegerFormatter** - For use with integers, or to format decimals like integers. Uses the symbol "i" or "I" such as `{0:I}` and `{0:i}`. It is similar to the `N0` formatter built into .net, except it has an option to omit group separators.
  The symbol can be followed with these modifiers:
  - `{0:I-,}` - Omit the group separator character. When not present, the group separator is used.
  - `{0:I,}` - Include the group separator. Not necessary because when there is no comma, it includes the group separator.
  - `{0:I#}` - The number of digits to show where # is shown. If the value is less than this, lead zeros are used to fill.

  Examples:

`{0:I}`, `{0:I,}`	10000 -> "10,000"	(includes group separators)
`{0:I3}`, `{0:I,3}`	1 -> "001"	(includes group separators and fills to 3 digits)
`{0:I-,}`	10000 -> "10000"	(omits group separators)
`{0:I-,7}`	11234 -> "0011234"	(omits group separators and fills to 8 digits)

- **CurrencyAndPercentFormatter** - For use with numbers to format them either as a currency or percent. This overrides the native formatters, using the same format strings ("c", "C", "p", "P"), adding rules to omit the group separator, currency or percent symbol, or decimal digits when all zero.
  The initial character can be followed with these modifiers:
  - `{0:C-$}` - Omit the currency symbol. When not present, the currency symbol is used.
  - `{0:C$}` - Include the currency symbol. Not necessary because when there is no $, it means include the currency symbol
  - `{0:P-%}` - Omit the percent symbol. When not present, the percent symbol is used.
  - `{0:P%}` - Include the percent symbol. Not necessary because when there is no %, it means include the percent symbol
  - `{0:C-,}` or `{0:P-,}` - Omit the group separator character. When not present, the group separator is used.
  - `{0:C,}` or `{0:P,}` - Include the group separator. Not necessary because when there is no comma, it includes the group separator.
  - `{0:C-0}` or `{0:P-0}` - Omit the floating point part when its zero (the value would be the same as an integer).
  - `{0:C#}` or `{0:P#}` - The number of floating point digits to show where # is shown. If not specified, it uses the culture's rule. Use 0 to remove the floating point part. However, if not specified and the value is already an integer, the floating point part is always removed.

Examples for currency:

o `{0:C}`, `{0:C,}`	5.0 -> "$5.00" 5000.0 -> "$5,000.00" 50 -> "$50"	(includes currency symbol and group separators) (because it is an integer passed in)
o `{0:C-$}`	5.0 -> "5.00" 5000.0 -> "5,000.00"	(omits currency symbol; includes group separators)
o `{0:C-$-,}`	5000.0 -> "5000.00"	(omits currency symbol and group separators)
o `{0:C3}`, `{0:C,3}`	5.00 -> "$5.000"	(includes group separators and requires 3 decimal digits)
o `{0:C0}`, `{0:C,0}`	50.0 -> "$50"	(includes group separators and removes decimal digits),
o `{0:C-,}`	5000.0 -> "$5000.00"	(omits group separators)
o `{0:C-,3}`	5000.0 -> "$5000.000"	(omits group separators and requires 3 decimal digits)
o `{0:C-,0}`	5000.0 -> "$5000"	(omits group separators and removes decimal digits)
o `{0:C-0}`	5000.00 -> "$5,000" 5000.01 -> "$5,000.01"	(removes decimal part when zero; includes currency symbol and group separators)
o `{0:C-$-,-0}`	5000.0 -> "5000" 5000.01 -> "5000.01"	(removes decimal part when zero; omits currency symbol and group separators)

Examples for percent:

o `{0:P}`, `{0:P,}`	0.50 -> "50.00 %" 50.0 -> "5,000.00 %" 50 -> "5,000 %"	(includes percent symbol and group separators) (because it is an integer passed in)
o `{0:P-%}`	0.50 -> "50.00" 50.0 -> "5,000.00"	(omits percent symbol; includes group separators)
o `{0:P-%-,}`	50.0 -> "5000.00"	(omits percent symbol and group separators)
o `{0:P3}`, `{0:P,3}`	0.50 -> "50.000 %"	(includes group separators and requires 3 decimal digits)
o `{0:P0}`, `{0:P,0}`	50.0 -> "$5,000"	(includes group separators and removes decimal digits)
o `{0:P-,}`	50.0 -> "5000.0 %"	(omits group separators)
o `{0:P-,3}`	50.0 -> "5000.000 %"	(omits group separators and requires 3 decimal digits),
o `{0:P-,0}`	50.0 -> "5000 %"	(omits group separators and removes decimal digits)
o `{0:P-0}`	0.50 -> "50 %" 0.501 -> "50.1 %"	(removes decimal part when zero; includes percent symbol and group separators)
o `{0:P-%-,-0}`	0.50 -> "50" 0.501 -> "50.1"	(removes decimal part when zero; omits percent symbol and group separators)

Add more by creating a class that implements the IAdvancedFormatterPlugIn interface.

Getting started

Either add the Visual Studio project into your application or add the source code files.

Using the Visual Studio project file

After adding the project, add a reference from your application to the project.

Using the source code

The source code files are in C#. If your application is not in C#, use the Visual Studio project file.
Add the source code files to your application. The source code files do not specify a namespace. Consider adding your own application's namespace to them.

Using AdvancedFormatProvider

You can either call String.Format() passing in the AdvancedFormatProvider as the first parameter, or call AdvancedFormatProvider.Format(), with similar parameters to String.Format.
AdvancedFormatProvider.Format() lets you avoid creating an instance of AdvancedFormatProvider and makes it easy to pass in a specific System.Globalization.CultureInfo object.

Do not need to pass a CultureInfo object

```
text = String.Format(AdvancedFormatProvider.Current, "text with tokens",
value1, value2, etc.)
text = AdvancedFormatProvider.Format("text with tokens", value1, value2,
etc.)
```

Examples

```
text = String.Format(AdvancedFormatProvider.Current, "The {0} has a value
of {1:C}.", "book", 10.0)
text = AdvancedFormatProvider.Format("The {0} has a value of {1:C}.",
"book", 10.0
```

Result:
"The book has a value of $10.00."

Need to pass a CultureInfo object

```
text = String.Format(new AdvancedFormatProvider(cultureInfo), "text with
tokens", value1, value2, etc.)
text = AdvancedFormatProvider.Format(cultureInfo, "text with tokens",
value1, value2, etc.)
```

Examples

```
text = String.Format(AdvancedFormatProvider.Current, "The {0} has a value
of {1:C}.", "book", 10.0)
text = AdvancedFormatProvider.Format("The {0} has a value of {1:C}.",
"book", 10.0
```

Result:
"The book has a value of $10.00."

Creating your own plug-in

Create a class that implements the IAdvancedFormatterPlugIn interface. See the code in the IAdvancedFormatProvider.cs source code file.
To add your class to AdvancedFormatProvider, in application startup code, call `AdvancedFormatProvider.RegisterFormatPlugIn(new YourClass())` passing an instance of your class. Your object will be maintained globally.
Do not attempt to modify it after application startup.