Preventing string extraction

There might be occasions where you need certain files, whole classes or properties on certain forms, or even a whole folder, to be excluded from string extraction. dxgettext doesn't know anything about classes and their properties, because it's not a compiler, and therefore it can't know about the classes and properties you invented. But usually you as the developer know about that, and there's a mechanism to tell dxgettex what you want to get excluded from string extraction.

It's done with a config file, always called *ggexclude.cfg*, that accepts different kinds of information about things that shall be excluded:

- classes
- properties of classes
- certain instances (=objects) of a class in certain files
- folders
- files

This file must be located in the directory where you want to extract strings from (that is the folder where you start "extract translations to template" in the context menu of the Windows-Explorer; in dxgettext command line it's the current working directory or the directory specified in the "-b"-option) . If the file doesn't exist, dxgettext just ignores that and does its normal job. It is a plain text file, consisting of four different kinds of lines: empty lines, comments, section and values.

Comments:
This is a comment, starting with a # at the beginning of each line
  # Leading spaces may be added, but no other leading characters

Sections:
These sections are recognized by *dxgettex*t:
[exclude-form-class]
[exclude-form-class-property]
[exclude-form-instance]
[exclude-file]
[exclude-dir]

Each section is optional and can contain zero to many lines, which are the *values* of that section. Below is a sample file with lots of comments to explain the internal structure and syntax. You have to be aware that the information you provide in this file has to be accurate, and, again, that the parser of *dxgettext* doesn't know about classes and inheritence. That means that if you have a class, say, *TMyEdit* and derived from that, *TMyDerivedEdit*, and you want their *Text* property to be excluded, you need to mention **both** classes in the config file. Leaving *TMyDerivedEdit* out would mean that its *Text* property would still be extracted.

Now the sample file:

```
# Text in square brackets, like "[exclude-dir]", is called a "section".
# Each line that is not empty, not a comment and not a section holds
# exactly 1 "value".
# All lines below a section are scanned for values belonging to that
# section until the next section starts. You can use the same section
# serveral times. It will all be added up.

[exclude-form-class]
# This section prevents a whole class to
# be excluded from scanning in all forms of the folder and subfolders
# where "ggexclude.cfg" is located.
# The format for a value is just "Classname". It's not case-sensitive.
# A wildcard "*" can be used optionally.
# A special case are collections, see [exclude-form-class-property] for that

# Here, everything of TEdit in DFM/XFM-files will be ignored. Remember:
# other classes derived from TEdit have to listed seperatly in order to
# exclude their properties as well. Inheritance is not recognized by dxgettext:
```

```
TEdit

# Visual containers like TPanel or TScrollbox have to be treated slightly
different.
# If you have a TPanel with a TLabel on it, writing "TPanel" would only
# exclude the properties of TPanel itself. If you want to exclude
# everything contained in a TPanel, use the wildcard "*" at the end, like this:
TPanel*
# The following only excludes the properties of TScrollbox itself, but not the
controls
# contained in Scrollboxes (except other classes explicitly listed here, like
# TEdit above):
TScrollbox

[exclude-form-class-property]
# This section prevents a certain property of a class to
# be excluded from scanning in all forms of the folder and subfolders
# where "ggexclude.cfg" is located.
# The format for a value is "Classname.Properyname". It's not case-sensitive. No
wildcards allowed.
# Classname is obvious, the propertyname has to be written the way it
# is written in the form file. If you're in doubt about how a certain property
# has to be stored here, just copy and paste the line from the DFM file here and
# put the classname before that.
# For simple strings the property name is one word:
TLabel.Caption
# ...and for TStrings it's like this:
TListbox.Items.Strings
TMemo.Lines.Strings
TQuery.SQL.Strings

# TEdit is listed in the [exclude-form-class]-section below which means
# that the whole class will be excluded. Listing TEdit.Text here therefore
# has no further effect
TEdit.Text

# A special case are collections in forms (like TDBGridColumns in a TDBGrid,
# TParams in a TQuery or TActionManager.ActionBars). You can exclude only
# the whole collection, but not certain properties of a collection. That
# means as well that in the case of nested collections (see
TActionManager.ActionBars
# in the sample unit "nestedcollections.dfm"), everything that appears below
# the collection with the highest level will be ignored.
# Note that some collections are saved with another name than their
propertyname.
# For example: "TQuery.Params" will be saved as "ParamData" in the form file.
TQuery.ParamData
TDBGrid.Columns
TActionManager.ActionBars
# these lines won't work:
# TDBGrid.Columns.Title.Caption
# TActionManager.ActionBars.ContextItems
# ("ContextItems" is a nested collection, which can hold another nested
collection and so on)

[exclude-form-instance]
# This section prevents a certain instance (=object) of a class in a certain
form file to
# be excluded from scanning.
# Each value is exactly one file with one instance. The format is
# "filename:instancename". On Windows, the "filename" part is not
# case-sensitive. You can use relative or absolute paths.
# Note that if the instance is something like a container or menu,
# everything belonging to that will be excluded.
```

```
# Note also that a frame on a form might contain a component with the
# same name as a component on the form. They would both be excluded.
Unit6.dfm:Popupmenu
Unit6.dfm:Label5

[exclude-file]
# This section prevents a whole file from being scanned.
# Each value is exactly one file. On Windows, it's not case-sensitive.
# You can use relative or absolute paths. Wildcards allowed.

Unit4.dfm
# Using the wildcard ".*" for a file extension means that the following
# matching Delphi-files will be excluded: dfm, xfm, pas, inc, rc, dpr
Unit5.*
# If Unit3 is already excluded by the [exclude-dir] above, because it
# is located in a subfolder of "subfolder", listing it here therefore
# has not further effect
subfolder\subfolder\Unit3.dfm

# you can use absolute paths as well, like this:
# on Windows: D:\test\Unit.pas
# on Linux:   /home/zaphod/projects/MainForm.*

[exclude-dir]
# This section prevents a whole folder and all it's subfolder from being
scanned.
# Each value is exactly one folder. On Windows, it's not case-sensitive.
# You can use relative or absolute paths. No wildcards allowed.

subfolder
# these are valid values as well:
# another\folder
# another\folder\
# Windows: D:\yet\another\folder
# Linux:  /home/zaphod/projects/subfolder/
# You don't have to worry about the path delimiters, both "/" and "\"
# can be used. They are converted to "/" internally
```