MiscMergeDriver

Inherits From: Object
Declared In: misckit/MiscMergeDriver.h

Class Description

A MiscMergeDriver is used to merge an ASCII template with several dictionaries filled with key/value pairs. Each dictionary will be used in turn to generate a new output ªdocumentº.

If you only need to generate a single merge, you may wish to simply use a MiscMergeEngine object. If you have several merges to perform, then a MiscMergeDriver implements the required loop to generate the required merges, as well as supporting a protocol that allows the merge engine some control over the loop. If you create your own loop, instead of using a MiscMergeDriver instance, some of the merge commands such as ªnextº will be ignored rather than performing the desired function.

To use a MiscMergeDriver you must provide it with a template, dictionaries to merge into the template, and, optionally, a MiscMergeEngine instance. If a MiscMergeEngine is not provided, one will be created to perform the merge. To set up a merge template, use the -**setTemplate:** method. It expects an instance of the MiscMergeTemplate class, which comes from an ASCII file or from a MiscString object.

The data to be merged into the template is set up using the -**setMergeData:** method. The data should be

stored as key/value pairs in a MiscDictionary object for each merge to be performed. Place all the dictionaries into a List object and use the List object as the argument to -**setMergeData:**.

Finally, use the -**doMerge:** method to perform the desired merge operation. The results will be returned as a List object with a MiscString corresponding to each MiscDictionary in the List provided to the MiscMergeDriver by the most recent -**setData:** message. For example, the third MiscString will contain the results from the merge with the third MiscDictionary. If the Merge returned no result (due to an error or an ªomitº command, for example) then the MiscString will be empty.

If you wish to use a specific subclass of MiscMergeEngine to perform the merge, then use the -**setEngine:** method to set up the engine before calling -**doMerge:**. This engine will be used for all subsequent merges unless -**setEngine:** is sent again.

For more information, please see the IntroMiscMerge.rtfd document. It describes the syntax of the merge language and built-in commands available. The MiscMergeArchitecture.rtfd document describes the architecutre of the various classes used to perform merging operations and how to add custom commands to the framework.

Instance Variables

MiscMergeTemplate ***template**;
List ***dictionaries**;
List ***output**;
MiscMergeEngine ***engine**;
BOOL **merging**;
int _**mergeLoopIndex**;

template	MiscMergeTemplate for merging

dictionaries	List of MiscDictionaries used for merges
output	The output list that will be returned by -**doMerge:**
engine	The merge engine to be used for merges
merging	YES if merging, NO if not
_mergeLoopIndex	Index to **dictionaries** when merge is in progress

Method Types

Accessing the template	- template - setTemplate:
Accessing the data	- mergeData - setMergeData:
Performing a merge	- doMerge:
Accessing the engine	- engine - setEngine:

Instance Methods

doMerge:

- (List *)**doMerge:***sender*

Sets up a merge engine, if necessary, and performs a merge of the template with the MiscDictionaries in the data List. Any engines created will be destroyed after the merge; engines set using -**setEngine** will persist, however.

A List object populated with MiscStrings will be returned. There is a one-to-one correspondence between the index of the return MiscStrings in the List and the MiscDictionaries' indices in the List that was provided via the most recent -**setMergeData**. Thus, if there were six dictionaries used for merging, six MiscStrings will be returned, as the result of six merges. Note that the ªnextº command will cause a MiscMergeEngine to attempt to skip forward to the next MiscDictionary, while still performing a single merge. In this case, an empty MiscString will be inserted in the output List as a placeholder and the final merge result will be put in the slot corresponding to the last dictionary used. Merges that fail or are halted due to an ªomitº command will also be represented by an empty MiscString in the output.

engine

- (MiscMergeEngine *)**engine**

Returns the merge engine, an instance of MiscMergeEngine, that will be used to perform a merge. If no engine has been set up, then **nil** is returned.

mergeData

- (List *)**mergeData**

Returns the List of MiscDictionaries that will be used for the next merge.

setEngine:

- **setEngine:**(MiscMergeEngine *)*anEngine*

Sets up an engine to be used for merging. If no engine is set, a temporary engine will be created before and used during a merge. It will be destroyed after it is used. Engines set using -**setEngine:** will not be destroyed at the end of a merge and will be used for subsequent merges as well. Setting a new engine will not free the old engine; the MiscMergeDriver does not ªownº the engine; it only makes use of it. This way, the same engine

could be used by several MiscMergeDriver instances. Setting the engine to **nil** will revert to the default create/use/destroy pattern. The engine cannot be changed while a merge loop is in progress. Returns **self** if successful or **nil** if failure occurs.

setMergeData:

- **setMergeData:**(List *)*aList*

Sets the List of MiscDictionaries that will be used for the next merge. Returns **self** upon success and **nil** upon failure. This method fails if a merge is in progress.

setTemplate:

- **setTemplate:**(MiscMergeTemplate *)*aTemplate*

Sets the MiscMergeTemplate that will be used for the next merge. Returns **self** upon success and **nil** upon failure. This method fails if a merge is in progress.

template

- (MiscMergeTemplate *)**template**

Returns the MiscMergeTemplate that will be used for the next merge.