MiscMergeEngine

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

Class Description

A MiscMergeEngine is the heart of the merging object suite. It actually performs the merges. To use it, simply give it a MiscMergeTemplate that has been properly set up with -**setTemplate:**. Next, give it a MiscDictionary object (-**setMergeDictionary:**) that is filled with the contents of the merge fields. The keys are field names and the values associated with the keys are the information that should be substituted for the fields in the merge template. Finally, send a -**merge:** message to start things off. A MiscString will be returned that contains the results of the merge.

The rest of the methods are an API to the internal stat of the engine which may be used to implement MiscMergeCommand subclasses.

To implement MiscMergeCommands, it is important to understand some of the internals of the MiscMergeEngine class.

The main thing to know is that there is an ªoutputº string that is kept throughout the merge and returned at the end. MiscMergeCommands should append strings to it as necessary with the -**appendToOutput:** method.

The MiscMergeEngine resolves field names through a series of symbol tables. Commands can request that arguments be ªresolvedº through these symbol tables with the -**getField:** method. The process is to first look at

the current merge dictionary. If the field name is found as a key in that dictionary, then the value for the key is returned. If not, then the ªlocalº symbol table is searched. The local symbol table may be populated by various MiscMergeCommands and starts out empty for each merge. If the local table doesn't have the value, then the ªparentº merge, if it exists, is consulted, followed by the global symbol table. Somewhere along the way, if a key into the merge dictionary is found, then the resolution is complete and the value is returned. If even the global symbol table doesn't have a desired key, then the key itself is returned, since it could not be resolved.

By doing this extensive resolution, it is possible to use MiscMergeCommands to create aliases for field names. It is also possible to use the global tables to conatin ªdefaultº values for any merge fields that might turn up empty on a particular merge. Note that there are specific methods which may be used to manipulate both the local and global symbol tables, as well as set up the parent merge.

Another special feature of the MiscMergeEngine is that it can carry internal ªvariablesº. A variable is some object that contains state and needs to be accessible throughout a merge. This is useful for groups of MiscMergeCommands that need to pass information between each other, but do not specifically know about each other. A prime example would be the if/else/endif structure supported by the kit. In order to allow nested if statements, a stack is required. The special -**ifStack** method returns this internal variable. However, there is a more general interface, using -**setVariableNamed:** and -**getVariableNamed:** which allows arbitrary variables to be stored and retreived by the engine. The ªif stackº, in fact, uses the above methods with a special internal name. (The accessor method is used to create the stack automatically the first time it is required; the -**getVariableNamed:** method can't do that since it doesn't know the class of the requested variable.) Variables are cleared at the start of a new merge, so only data pertaining to a merge should be stored there. This is, of course, the preferred way for MiscMergeCommands to ªcommunicateº with each other.

One final note about the ªif stackº special variable: if it's state suggests that the engine is walking through an ªinactiveº if block, then all strings sent to be appended to the output will be thrown out until the engine has entered an ªactiveº block. (See MiscIfStack's class description for a deeper understanding.)

The current API should be adequate to perform most things a MiscMergeCommand would want to do. However, it is possible that function would be helpful or that some bit of information is still inaccessible. If this is the case, complain to the author (Don Yacktman, yackd@xmission.com) and he will consider enhancing the API to this object as necessary. Of course, subclasses and categories might also be workable approaches to such deficiencies.

Instance Variables

id **template**;
id **dictionary**;
MiscMergeEngine ***parentMerge**;
MiscDictionary ***symbolTable**;
MiscDictionary ***variables**;
BOOL **mergeInProgress**;
BOOL **abort**;
id **outputString**;
BOOL **outputOK**;
id **driver**;

template	Current merge template.
dictionary	Current merge dictionary.
parentMerge	Parent merge, if any.
symbolTable	Local symbol table.
variables	Storage for merge variables.
mergeInProgress	Set to true if a merge is in progress.
abort	Set to true if a merge should be aborted.
outputString	The output string for the current merge in progress.
outputOK	NO if output to outputString should be discarded.
driver	The object that started the current merge.

Method Types

Creating and setting up an engine
+ newWithTemplate:
± init

Setting up a merge
- setTemplate:
± setMergeDictionary:

Performing a merge
- merge:
± mergeWithDictionary:sender:

Setting a parent MergeEngine for sub-merges
± setParentMerge:
± parentMerge

Primitives for MiscMergeCommands
± abortMerge
± advanceRecord
± appendToOutput:
± dictionary

Handling the local symbol table
- resetSymbolTable
± symbolForKey:
± setSymbol:toValue:
± getField:

Handling merge ªvariablesº
± resetVariables
± getVariableNamed:
± setVariableNamed:to:
± ifStack

Handling the global symbol table
+ resetGlobalSymbolTable
+ setGlobalSymbol:toValue:
+ addDictionaryToGlobalSymbols:
+ addListToGlobalSymbols:

+ globalSymbolForKey:

Class Methods

addDictionaryToGlobalSymbols:

+ **addDictionaryToGlobalSymbols:**(MiscDictionary *)*aDictionary*

Adds the contents of *aDictionary* to the global symbol table. Returns **self**. ***Warning:*** *This is currently unimplemented.*

addListToGlobalSymbols:

+ **addListToGlobalSymbols:**(List *)*aList*

Adds the contents of *aList* to the global symbol table. Returns **self**. The keys for the List's contents are generated as ªf0º, ªf1º, and so on.

globalSymbolForKey:

+ (MiscString *)**globalSymbolForKey:**(MiscString *)*name*

Returns the value for the global symbol *name*, if found. Returns **nil** if not found.

newWithTemplate:

+ **newWithTemplate:**(MiscMergeTemplate *)*aTemplate*

Creates and initializes a new MiscMergeEngine instance, setting the current template to *aTemplate*. Returns the newly created object.

resetGlobalSymbolTable
+ **resetGlobalSymbolTable**

Empties the global symbol table. Returns **self**.

setGlobalSymbol:toValue:
+ **setGlobalSymbol:***name* **toValue:***val*

Sets the global symbol *name* to have the value *val*. Returns **self**.

Instance Methods

abortMerge
- **abortMerge**

Aborts the current merge. This means that the merge output will be **nil**, as well. Returns **self**.

advanceRecord
- **advanceRecord**

Attempts to advance to the next merge dictionary while still working with the current output string. This might be used to allow two merges to appear on the same "page" or document, for example. For it to work properly, the driver that started the merge must respond to the -**advanceMergeLoop** method. Returns **self**.

appendToOutput:
- **appendToOutput:**(MiscString *)*newText*

Appends the contents of *newText* to the merge output. Returns **self**.

dictionary

- (MiscDictionary *)**dictionary**

Returns the current merge dictionary.

getField:

- (MiscString *)**getField:**(MiscString *)*fieldName*

Attempts to resolve a field name. If found in the merge dictionary, then the value in the dictionary is returned. If not found there, then a search through the symbol tables is conducted. If there are no local or global symbols named *fieldName* then *fieldName* is returned. If there are local or global symbols, however, then an attempt is made to resolve their values to a key in the merge dictionary. If the local or global symbols exist, but cannot be resolved into values in the merge dictionary, then they local/global value is returned.

This complex search allows aliases to be created for various merge fields so that they may be accessed by different names. It also allows redirection--if a field is missing in a particular merge, the global or local symbol tables can suggest another field to use in its place. Finally, it allows setting of default values in the local or global symbol table. If the field is missing from the merge dictionary, then a default stored in the symbol tables can be used.

getVariableNamed:

- **getVariableNamed:**(MiscString *)*name*

Returns the merge variable named *name*. Returns **nil** if not found.

ifStack

- **ifStack**

Returns the special ªif stackº merge variable, creating it if necessary. The ªif stackº is used by the if/else/endif commands and also used to control turning the output of the merge on and off.

init

- **init**

Initializes a new MiscMergeEngine instance. Returns **self**.

merge:

- (MiscString *)**merge:***sender*

Performs a merge using the current dictionary and template. If successful, then a MiscString containing the results of the merge is returned. If unsuccessful, **nil** is returned. The argument *sender* should be the initiating driver. If not, some commands, such as ªnextº will not work properly.

mergeWithDictionary:sender:

- (MiscString *)**mergeWithDictionary:**(MiscDictionary *)*aDictionary* **sender:***sender*

Initiates a merge with the current template and *aDictionary*. Returns a MiscString containing the output of the merge if successful and **nil** otherwise. The argument *sender* should be the initiating driver. If not, some commands, such as ªnextº will not work properly.

parentMerge

- (MiscMergeEngine *)**parentMerge**

Returns the ªparentº merge engine.

resetSymbolTable

- **resetSymbolTable**

Clears the local symbol table. Returns **self**.

resetVariables
- **resetVariables**

Empties out the merge variables. Returns **self**.

setMergeDictionary:
- **setMergeDictionary:**(MiscDictionary *)*aDictionary*

Sets the current dictionary. The next invocation of -**merge:** will use *aDictionary* as the merge dictionary. Returns **self**.

setParentMerge:
- **setParentMerge:**(MiscMergeEngine *)*aMergeEngine*

Sets the ªparentº merge for this merge engine. If a symbol is undefined in this instance's symbol table, then the parent will be consulted to see if it is defined there. Returns **self**.

setSymbol:toValue:
- **setSymbol:**(MiscString *)*name* **toValue:**(MiscString *)*value*

Adds the symbol *name* to the local symbol table with *value* as its *value*. Returns **self**.

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

Sets the current merge template. All future invocations of -**merge:** will use *aTemplate* as the merge template,

until this method is called again. Returns **self**.

setVariableNamed:to:

- **setVariableNamed:**(MiscString *)*name* **to:***aValue*

Sets the merge variable named *name* to *aValue*. Returns **self**.

symbolForKey:

- (MiscString *)**symbolForKey:**(MiscString *)*name*

Attempts to find a symbol in the symbol table that corresponds to *name*. If found, the value for that symbol is returned, if not, then **nil** is returned. The search is conducted through the local symbol table, through all parent merges' symbol tables, and then through the global symbol table.