FC_Family

Olivier Laviale 2002

COLLABORATORS

	TITLE : FC_Family	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Olivier Laviale 2002	August 24, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 FC_Family **1**

1.1 Feelin : FC_Family . . . 1
1.2 FC_Family / FM_Family_AddMember . . . 1
1.3 FC_Family / FM_Family_RemMember . . . 2
1.4 FC_Family / FM_Family_AddHead . . . 2
1.5 FC_Family / FM_Family_AddTail . . . 2
1.6 FC_Family / FM_Family_Insert . . . 3
1.7 FC_Family / FM_Family_Remove . . . 3
1.8 FC_Family / FM_Family_Sort . . . 3
1.9 FC_Family / FM_Family_Transfer . . . 4
1.10 FC_Family / FA_Child . . . 4
1.11 FC_Family / FA_Family . . . 4
1.12 FC_Family / FA_Family_List . . . 5
1.13 FC_Family / FA_Family_Head . . . 5
1.14 FC_Family / FA_Family_Tail . . . 6

Chapter 1

FC_Family

1.1 Feelin : FC_Family

FC_Family

FC_Family class handles a list of children. This is e.g. the case for Clients, Groups, Menus...

These classes are not subclasses of FC_Family. Family objects are simple objects, but as a subclass of FC_Tool, they know about their owner and can communicate with it. Take a look at FA_Family to see how things work.

FC_Family defines methods and attributes to add and remove children, sort children, and transfer children to other Family objects.

FC_Family automatically set, and unset, the attribute FA_Parent to the owner of the family e.i the real parent of the child.

Methods

FM_AddMember FM_RemMember

FM_Family_AddHead

FM_Family_AddTail FM_Family_Insert

FM_Family_Remove FM_Family_Sort

FM_Family_Transfer

Attributes

FA_Child FA_Family

FA_Family_Head FA_Family_List

FA_Family_Tail

1.2 FC_Family / FM_Family_AddMember

NAME

FM_Family_AddMember -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_AddMember,struct FeelinObject *child)

FUNCTION

Synonimus to FM_Family_AddTail .

1.3 FC_Family / FM_Family_RemMember

NAME

FM_Family_RemMember -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_RemMember,struct FeelinObject *child)

FUNCTION

Synonimus to FM_Family_Remove .

1.4 FC_Family / FM_Family_AddHead

NAME

FM_Family_AddHead -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_AddHead,struct FeelinObject *child)

FUNCTION

Add an object as first object to the family. Classes using FC_Family usually define which types of objects are possible within their family.

INPUTS

child - the object to be added.

SEE ALSO

FM_Family_AddTail FM_Family_Insert

FM_Family_Remove

FA_Child

1.5 FC_Family / FM_Family_AddTail

NAME

FM_Family_AddTail -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_AddTail,struct FeelinObject *child)

FUNCTION

Add an object as last object to the family. Classes using FC_Family usually define which types of objects are possible within their family.

This method has FM_AddMember as synonimus.

INPUTS

obj - the object to be added.

SEE ALSO

FM_Family_AddHead FM_Family_Insert

FM_Family_Remove

FA_Child

1.6 FC_Family / FM_Family_Insert

NAME

FM_Family_Insert -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_Insert,struct FeelinObject *child,struct FeelinObject *pred)

FUNCTION

Add an object after another object to the family. Classes using FC_Family usually define which types of objects are possible within their family.

INPUTS

child - the object to be added. pred - the new object is inserted after this object. pred must of course be a member of the family.

SEE ALSO

FM_Family_AddHead FM_Family_AddTail

FM_Family_Remove

FA_Child

1.7 FC_Family / FM_Family_Remove

NAME

FM_Family_Remove -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_Remove,struct FeelinObject *child)

FUNCTION

Remove an object from a family.

This method has FM_RemMember as synonimus.

INPUTS

child - the object to be removed.

SEE ALSO

FM_Family_AddHead FM_Family_AddTail

FM_Family_Insert

FA_Child

1.8 FC_Family / FM_Family_Sort

NAME

FM_Family_Sort -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_Sort,struct FeelinObject *child1, ..., struct FeelinObject *childn,ULONG NULL)

FUNCTION

Sort the children of a family.

INPUTS

Array that contains all the children of the family in the desired order. The array must be terminated with a NULL entry.

SEE ALSO

FA_Child

1.9 FC_Family / FM_Family_Transfer

NAME

FM_Family_Transfer -- (00.00)

SYNOPSIS

F_Do(obj,FM_Family_Transfer,APTR family)

FUNCTION

All the children of the family are removed and added to another family in the same order.

INPUTS

family - the destination family.

SEE ALSO

FA_Child

1.10 FC_Family / FA_Child

NAME

FA_Child -- (00.00) [I..], struct FeelinObject *

FUNCTION

You supply a pointer to a previously created Feelin object here. This object will be added to the family at family creation time. Of course you can specify any number of child objects, limited only by available memory.

Normally, the value for a FA_Child tag is a direct call to another F_NewObjA(), children are generated on the fly.

When a family is disposed, all of its children will also get deleted. If you supply a NULL pointer as child, the family object will fail and previously dispose all valid children found in the taglist.

This behaviour makes it possible to generate a complete family within one single (but long) F_NewObjA() call. Error checking is not necessary since every error, even if it occurs in a very deep nesting level, will cause the complete call to fail without leaving back any previously created object.

SEE ALSO

FM_Family_AddHead FM_Family_AddTail

FM_Family_Insert FM_Family_Remove

FA_Family

1.11 FC_Family / FA_Family

NAME

FA_Family -- (00.00) [..G], APTR

FUNCTION

This attribute is designed to communicate with the owner of the Family object. Classes with Family capabilities are not subclasses of the FC_Family, they use a Family object as a Tool.

The attribute FA_Family may be used by classes to return a pointer to their Family object.

EXAMPLE

FM_New: if (F_NewObj(FC_Family, FA_Tool_Owner, obj, /* Pointer to the object holding the Family Object */ TAG_MORE, tags, /* TagItems that comes with the FM_New method */ TAG_DONE)) {

/* If some children have failed i.e FA_Child is found NULL in the taglist, all children in the taglist will be disposed by the FC_Family, and the Family object will fail to create. Do not save Family object pointer here !! This will be done automatically if everything was ok using the FA_Family attribute, just check for NULL */

return obj; }

FM_Dispose: Self->Family = (APTR)F_DisposeObj(Self->Family);

/* When Family object is disposed all children are disposed too. Remember that F_DisposeObj() always return NULL and handles NULL pointers */

FM_Set: while (item = NextTagItem(&Tags)) { switch (item->ti_Tag) { case FA_Family: LOD->Family = item->ti_Data; break; /* Only save Family object pointer here */ ... } }

FM_Get: while (item = NextTagItem(&tags)) { save = item->ti_Data;

switch (item->ti_Tag) { case FA_Family: *save = LOD->Family; break; /* This is recommended */ ... } }

SEE ALSO

FA_Child

1.12 FC_Family / FA_Family_List

NAME

FA_Family_List -- (00.00) [..G], struct FeelinList *

FUNCTION

Returns a pointer to a struct FeelinList which contains the children of a family object. You must parse this list with F_NextObject().

SEE ALSO

FA_Child

1.13 FC_Family / FA_Family_Head

NAME

FA_Family_Head -- (00.00) [..G], struct FeelinObject *

FUNCTION

Returns a pointer to the first object of the family.

SEE ALSO

FA_Family_List FA_Family_Tail

1.14 FC_Family / FA_Family_Tail

NAME

FA_Family_Tail -- (00.00) [..G], struct FeelinObject *

FUNCTION

Returns a pointer to the last object of the family.

SEE ALSO

FA_Family_Head FA_Family_List