FC_Family

Olivier LAVIALE 2004

COLLABORATORS

	TITLE : FC_Family		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Olivier LAVIALE 2004	January 13, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FC_Family

1.1 Feelin : FC_Family

FC_Family -- 03.00

IDs: Static Super: NONE Include: <libraries/feelin.h>

This class handles a list of children. This is e.g. the case for applications, groups, menus... These classes are not subclasses of FC_Family, but FC_Family objects know about their owner and can communicate with it. Take a look at the FA_Family attribute to see how things work.

Generic methods defined by FC_Object to add and remove children are supported by the class. No FeelinNode structure, that should be embended in the object, is used. Instead, objects are referenced in nodes linked in a private list. Thus, any kind of object can be added to a family.

The FA_Parent attribute of the object added, or removed, is automatically set, and unset, to the owner of the family, the real parent of the child.

CHANGES

FM_New FM_Dispose

FM_AddMember FM_RemMember

ATTRIBUTES

FA_Family FA_Family_Head

FA_Family_Tail

TYPES

FFamilyNode

VARIABLES

DBG_FAMILY_ADDMEMBER DBG_FAMILY_REMMEMBER

1.2 FC_Family / FM_AddMember

NAME

FM_AddMember

CHANGE

This method adds an object to a family.

Before actually adding the object, the method checks if the object is already added, in which case the method logs a message with the level FV_ERLV_DEV. If the object is not already a member, the FM_Connect method is invoked on the object with the owner of the family as argument. If the method result is different than FALSE, the object is added according to the position requested.

NOTES

The class do not use any embended struct FeelinNode to link objects. Instead, objects are referenced in nodes linked to a private list. Thus, any kind of object can be added to the family.

A message can be logged for each object added if you set the variable DBG_FAMILY_ADDMEMBER to 1.

SEE ALSO

FM_RemMember

1.3 FC_Family / FM_Dispose

NAME

FM_Dispose

CHANGE

For each object in the family, the owner of the family is invoked with the FM_RemMember method and an object as argument, then the object is disposed with the F_DisposeObj() function.

1.4 FC_Family / FM_New

NAME

FM_New

CHANGE

For each object defined by FA_Child found in the taglist, the family's owner receives a FM_AddMember method with the object as message (struct FS_AddMember).

If an object is NULL (probably creation failure...) or if the FM_AddMember method returns FALSE, objects already member of the family are removed with the method FM_RemMember (invoked on the owner of the family). Then, the taglist is read again and every object defined by FA_Child is disposed with the F_DisposeObj() function.

NOTES

The owner of the family is defined with the FA_Family_Owner attribute, this attribute is very important. If the owner of the family is not defined, objects won't be added automatically.

1.5 FC_Family / FM_RemMember

NAME

FM_RemMember

CHANGE

This method removes an object from a family.

Before actually removing the object, the method checks if the object is really a member of the family. If the object is not a member of the family, the method logs a message with the level FV_ERLV_DEV, and returns FALSE.

If the object is a member of the family, it is invoked with the method FM_Disconnect, then the object is removed from the family and the method returns TRUE.

NOTES

A message can be logged for each object removed if you set the variable DBG_FAMILY_ADDMEMBER to 1.

SEE ALSO

FM_AddMember

1.6 FC_Family / FA_Family

NAME

FA_Family -- (01.00) [..G], FObject

FUNCTION

This attribute is designed to communicate with the owner of the family Classes with family capabilities are not subclasses of FC_Family, they use it as a supporting tool.

The attribute FA_Family may be used by classes to return a pointer to their FC_Family object (recommanded).

EXAMPLE

F_METHOD(FObject,mNew) { ...

if (F_NewObj(FC_Family, FA_Family_Owner, Obj, // Pointer to the object FC_Family Object TAG_MORE, Msg)) // Tag-items that comes with the FM_New method

{

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

return Obj; } return NULL; }

F_METHOD(void,mDispose) { ...

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

F_DisposeObj(LOD -> Family); LOD -> Family = NULL;

F_SUPERDO(); }

F_METHOD(void,mSet) { ...

while (F_DynamicNTI(&Tags,&item,Class) switch (item.ti_Tag) { ...

/* Only save Family object pointer here */

case FA_Family: LOD -> Family = (FObject)(item.ti_Data); break;

... } }

F_METHOD(void,mGet) { ...

while (F_DynamicNTI(&Tags,&item,Class)) switch (item.ti_Tag) { ...

/* This is recommended */

case FA_Family: F_STORE(LOD -> Family); break;

... }

SEE ALSO

FA_Child

1.7 FC_Family / FA_Family_Head

NAME

FA_Family_Head -- (03.00) [..G], FFamilyNode *

FUNCTION

Returns a pointer a FFamilyNode holding the first object of the family.

SEE ALSO

FA_Family_Tail

1.8 FC_Family / FA_Family_Tail

NAME

FA_Family_Tail -- (03.00) [..G], FFamilyNode *

FUNCTION

Returns a pointer a FFamilyNode holding the last object of the family.

SEE ALSO

FA_Family_Head

1.9 FC_Family / FFamilyNode

NAME

FFamilyNode -- (03.00)

STRUCT

struct FeelinFamilyNode { struct FeelinFamilyNode *Next; struct FeelinFamilyNode *Prev; FObject Object; };

typedef struct FeelinFamilyNode FFamilyNode;

FUNCTION

The class do not use any embended struct FeelinNode to link objects. Instead, objects are referenced in FFamilyNodes linked to a private list. Thus, any kind of object can be added to the family.

1.10 FC_Family / DBG_FAMILY_ADDMEMBER

NAME

DBG_FAMILY_ADDMEMBER -- (03.00)

FUNCTION

If this system variable is TRUE, the FM_AddMember method logs a message for each object added. A FC_DOSNotify object is used to watch the system variable "ENV:Feelin/DBG_FAMILY_ADDMEMBER" and react on changes.

SEE ASLO

DBG_FAMILY_REMMEMBER

1.11 FC_Family / DBG_FAMILY_REMMEMBER

NAME

DBG_FAMILY_REMMEMBER -- (03.00)

FUNCTION

If this system variable is TRUE, the FM_RemMember method logs a message for each object added. A FC_DOSNotify object is used to watch the system variable "ENV:Feelin/DBG_FAMILY_REMMEMBER" and react on changes.

SEE ASLO

DBG_FAMILY_ADDMEMBER