DomotIcApp

Generated by Doxygen 1.8.11

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 EP Namespace Reference

Classes

- class Core

  Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.
- class Equipment

  Représente un équipement. Cette classe abstraite est spécialisée par EquipmentKira et EquipmentFibaro.
- class EquipmentFibaro

  Représente un équipement lié à une Fibaro, hérite de Equipment.
- class EquipmentKira

  Représente un équipement lié à une Kira, hérite de Equipment.
- class Node

  Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes Equipment et Room, et n'est donc pas utilisée telle qu'elle.
- class Room

  Représente une pièce de la maison ; Hérite de Node. On peut y retrouver les différents équipements liés à une pièce et les gérer.

Functions

- DllExport Core ∗ Core_New (char ∗file)

  Constructeur statique utilisé pour permettre l'utilisation des objets Core en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport Core ∗ Core_NewFromSave (char ∗file)

  Constructeur statique utilisé pour permettre l'utilisation des objets Core en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur. Après création de l'objet, la méthode Core::load() est appelée sur celui-ci.
- DllExport void Core_Delete (Core ∗core)

  Destructeur statique, pour permettre la destruction des objets Core en passant par la DLL.
- DllExport void Core_SaveAndDelete (Core ∗core)

  Destructeur statique, pour permettre la destruction des objets Core en passant par la DLL. La méthode Core::save() est appelée avant sa destruction.
- DllExport char ∗ Equipment_getIpKira ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

- DllExport char * Equipment_getIpFibaro ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int Equipment_setIpKira (char *new_ip)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int Equipment_setIpFibaro (char *new_ip)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport char * Equipment_getLoginFibaro ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport char * Equipment_getPasswordFibaro ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int Equipment_setLoginFibaro (char *new_login)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int Equipment_setPasswordFibaro (char *new_password)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport EquipmentKira * EquipmentKira_New (char *name, char *ico, Node *parent, int buttonId, int page)

  Constructeur statique utilisé pour permettre l'utilisation des objets EquipmentKira en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void EquipmentKira_Delete (EquipmentKira *eq)

  Destructeur statique, pour permettre la destruction des objets EquipmentKira en passant par la DLL.
- DllExport EquipmentFibaro * EquipmentFibaro_New (char *name, char *ico, Node *parent, int equipmentId, char *action)

  Constructeur statique utilisé pour permettre l'utilisation des objets EquipmentFibaro en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void EquipmentFibaro_Delete (EquipmentFibaro *eq)

  Destructeur statique, pour permettre la destruction des objets EquipmentFibaro en passant par la DLL.
- DllExport Node * Node_New (char *name, char *ico)

  Constructeur statique utilisé pour permettre l'utilisation des objets Node en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void Node_Delete (Node *node)

  Destructeur statique, pour permettre la destruction des objets Node en passant par la DLL.
- void DllImport requeteHttpKira (char *s1, char *s2)

  Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Kira. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode EquipmentKira::sendRequest() ; les paramètres sont construits automatiquement à partir des informations du l'équipement.
- void DllImport requeteHttpFibaro (char *s1, char *s2, char *user, char *pass)

  Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Fibaro. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode EquipmentFibaro::sendRequest() ; les paramètres sont construits automatiquement à partir des informations du l'équipement.
- DllExport Room * Room_New (char *name, char *ico)

  Constructeur statique utilisé pour permettre l'utilisation des objets Room en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void Room_Delete (Room *room)

  Destructeur statique, pour permettre la destruction des objets Room en passant par la DLL.
- __declspec (dllexport) Core *Core_New(char *file)

Variables

- char * ico
- char Node * parent
- char Node int buttonId
- char Node int int page
- char Node int equipmentId
- char Node int char * action

5.1.1 Function Documentation

5.1.1.1 EP::__declspec (dllexport) [new]

5.1.1.2 DllExport void EP::Core_Delete (Core * *core*)

Destructeur statique, pour permettre la destruction des objets Core en passant par la DLL.

Parameters

core	Un pointeur vers l'objet à détruire.

5.1.1.3 DllExport Core* EP::Core_New (char * *file*)

Constructeur statique utilisé pour permettre l'utilisation des objets Core en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

Returns

Un pointeur vers l'objet créé.

5.1.1.4 DllExport Core* EP::Core_NewFromSave (char * *file*)

Constructeur statique utilisé pour permettre l'utilisation des objets Core en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur. Après création de l'objet, la méthode Core::load() est appelée sur celui-ci.

Returns

Un pointeur vers l'objet créé.

5.1.1.5 DllExport void EP::Core_SaveAndDelete (Core * *core*)

Destructeur statique, pour permettre la destruction des objets Core en passant par la DLL. La méthode Core::save() est appelée avant sa destruction.

Parameters

core	Un pointeur vers l'objet à détruire.

5.1.1.6 DllExport char* EP::Equipment_getIpFibaro ( )

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Returns

L'adresse IP de la Fibaro.

5.1.1.7 DllExport char* EP::Equipment_getIpKira ( )

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Returns

L'adresse IP de la Kira.

5.1.1.8 DllExport char* EP::Equipment_getLoginFibaro ( )

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Returns

Le login de la Fibaro.

5.1.1.9 DllExport char* EP::Equipment_getPasswordFibaro ( )

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Returns

Le password de la Fibaro.

5.1.1.10 DllExport int EP::Equipment_setIpFibaro (char * *new_ip*)

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Parameters

new_ip	La nouvelle adresse IP de la Fibaro.

Returns

0 si l'adresse est incorrecte, 1 sinon.

5.1.1.11 DllExport int EP::Equipment_setIpKira (char * *new_ip*)

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Parameters

new_ip	La nouvelle adresse IP de la Kira.

Returns

0 si l'adresse est incorrecte, 1 sinon.

5.1.1.12 DllExport int EP::Equipment_setLoginFibaro (char * *new_login*)

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Parameters

new_login	Le nouveau login de la Fibaro.

5.1.1.13 DllExport int EP::Equipment_setPasswordFibaro (char * *new_password*)

Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

Parameters

new_password	Le nouveau password de la Fibaro.

5.1.1.14 DllExport void EP::EquipmentFibaro_Delete (EquipmentFibaro * *eq*)

Destructeur statique, pour permettre la destruction des objets EquipmentFibaro en passant par la DLL.

Parameters

eq	Un pointeur vers l'objet à détruire.

5.1.1.15 DllExport EquipmentFibaro* EP::EquipmentFibaro_New (char * *name*, char * *ico*, Node * *parent*, int *equipmentId*, char * *action*)

Constructeur statique utilisé pour permettre l'utilisation des objets EquipmentFibaro en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

Returns

Un pointeur vers l'objet créé.

5.1.1.16 DllExport void EP::EquipmentKira_Delete (EquipmentKira * *eq*)

Destructeur statique, pour permettre la destruction des objets EquipmentKira en passant par la DLL.

Parameters

eq	Un pointeur vers l'objet à détruire.

5.1.1.17 DllExport EquipmentKira* EP::EquipmentKira_New (char * *name*, char * *ico*, Node * *parent*, int *buttonId*, int *page*)

Constructeur statique utilisé pour permettre l'utilisation des objets EquipmentKira en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

Returns

Un pointeur vers l'objet créé.

5.1.1.18 DllExport void EP::Node_Delete (Node * *node*)

Destructeur statique, pour permettre la destruction des objets Node en passant par la DLL.

Parameters

node	Un pointeur vers l'objet à détruire.

5.1.1.19 DllExport Node* EP::Node_New (char * *name*, char * *ico*)

Constructeur statique utilisé pour permettre l'utilisation des objets Node en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

Returns

Un pointeur vers l'objet créé.

5.1.1.20 void DllImport EP::requeteHttpFibaro (char * *s1*, char * *s2*, char * *user*, char * *pass*)

Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Fibaro. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode EquipmentFibaro::sendRequest() ; les paramètres sont construits automatiquement à partir des informations du l'équipement.

Parameters

s1	L'adresse IP de la Fibaro.
s2	La deuxième partie de l'adresse correspondant à la requête HTTP.
user	Le login utilisé pour s'authentifier auprès de la Fibaro.
pass	Le login utilisé pour s'authentifier auprès de la Fibaro.

5.1.1.21 void DllImport EP::requeteHttpKira (char * *s1*, char * *s2*)

Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Kira. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode EquipmentKira::sendRequest() ; les paramètres sont construits automatiquement à partir des informations du l'équipement.

Parameters

s1	L'adresse IP de la Kira.
s2	La deuxième partie de l'adresse correspondant à la requête HTTP.

5.1.1.22 DllExport void EP::Room_Delete (Room * *room*)

Destructeur statique, pour permettre la destruction des objets Room en passant par la DLL.

Parameters

room	Un pointeur vers l'objet à détruire.

5.1.1.23 DllExport Room* EP::Room_New (char * *name*, char * *ico*)

Constructeur statique utilisé pour permettre l'utilisation des objets Room en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.

Returns

Un pointeur vers l'objet créé.

5.1.2 Variable Documentation

5.1.2.1 char Node int char* EP::action

Initial value:

```
{
        return new EquipmentFibaro(name, ico, parent, equipmentId,
      action)
```

5.1.2.2 char Node int EP::buttonId

5.1.2.3 char Node int EP::equipmentId

5.1.2.4 char * EP::ico

Initial value:

```
{
        return new Node(name, ico)
```

5.1.2.5 char Node int int EP::page

Initial value:

```
{
        return new EquipmentKira(name, ico, parent, buttonId,
      page)
```

5.1.2.6 char Node * EP::parent

Chapter 6

Class Documentation

6.1 EP::Core Class Reference

Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.

`#include <Core.h>`

Public Member Functions

- Core (char *file)

  Le constructeur de Core.summary.
- ~Core ()

  Le destructeur de Core. Vide la liste de pièces.
- int save ()

  Sauvegarde l'état actuel de l'application dans le fichier de sauvegarde.
- int load ()

  Charge l'application depuis le fichier de sauvegarde.
- int addRoom (Room *room)

  Ajoute le pointeur d'une pièce donnée dans m_listRooms. Le nom d'une pièce doit être unique.
- int deleteRoomByIndex (int index)

  Supprime la pièce dont l'index est donné en paramètre.
- int deleteRoomByName (char *name)

  Supprime la pièce dont le nom est donné en paramètre.
- std::vector< Room * > * getRooms ()

  Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec getNumberRooms et getRoomByIndex.
- Room * getRoomByName (char *name)

  Donne un pointeur vers la pièce dont le nom est donné en paramètre.
- Room * getRoomByIndex (int index)

  Donne un pointeur vers la pièce dont l'index est donné en paramètre.
- char * getFileSave ()
- void setFileSave (char *name)
- int getNumberRooms ()
- char * getCOMPort ()
- void setCOMPort (char *port)
- int getThemeId ()
- int getIconSize ()
- void setThemeId (int id)

  Permet de changer le thème utilisé.
- void setIconSize (int size)

  Permet de changer la taille des icônes.

6.1.1 Detailed Description

Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 EP::Core::Core (char * *file*)

Le constructeur de Core.summary.

Parameters

file	Le nom du fichier de sauvegarde.

6.1.2.2 EP::Core::~Core ( )

Le destructeur de Core. Vide la liste de pièces.

6.1.3 Member Function Documentation

6.1.3.1 int EP::Core::addRoom (Room * *room*)

Ajoute le pointeur d'une pièce donnée dans m_listRooms. Le nom d'une pièce doit être unique.

Parameters

room	Le pointeur de la pièce à ajouter.

Returns

0 si tout s'est bien passé, 1 si le nom est déjà pris.

6.1.3.2 int EP::Core::deleteRoomByIndex (int *index*)

Supprime la pièce dont l'index est donné en paramètre.

Parameters

index	L'index de la pièce à supprimer de m_listRooms.

Returns

0 si tout s'est bien passé, 1 si l'index ne correspond à aucune pièce.

6.1.3.3 int EP::Core::deleteRoomByName (char * *name*)

Supprime la pièce dont le nom est donné en paramètre.

Parameters

name	Le nom de la pièce à supprimer de m_listRooms.

Returns

0 si tout s'est bien passé, 1 si le nom ne correspond à aucune pièce.

6.1.3.4 char * EP::Core::getCOMPort ( )

Returns

Le nom du port série utilisé pour communiquer avec le fauteuil.

6.1.3.5 char * EP::Core::getFileSave ( )

Returns

Le nom du fichier de sauvegarde.

6.1.3.6 int EP::Core::getIconSize ( )

Returns

L'identifiant pour la taille des icônes.

6.1.3.7 int EP::Core::getNumberRooms ( )

Returns

Le nombre de pièces de m_listRooms.

6.1.3.8 Room * EP::Core::getRoomByIndex (int *index*)

Donne un pointeur vers la pièce dont l'index est donné en paramètre.

Parameters

index	L'index de la pièce qu'on veut récupérer.

Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

6.1.3.9 Room * EP::Core::getRoomByName (char * *name*)

Donne un pointeur vers la pièce dont le nom est donné en paramètre.

Parameters

name	Le nom de la pièce qu'on veut récupérer.

Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

6.1.3.10 vector< Room * > * EP::Core::getRooms ( )

Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec getNumberRooms et getRoomBy↩ Index.

Returns

L'attribut m_listRooms.

6.1.3.11 int EP::Core::getThemeId ( )

Returns

L'identifiant du thème utilisé.

6.1.3.12 int EP::Core::load ( )

Charge l'application depuis le fichier de sauvegarde.

6.1.3.13 int EP::Core::save ( )

Sauvegarde l'état actuel de l'application dans le fichier de sauvegarde.

6.1.3.14 void EP::Core::setCOMPort (char * *port*)

Parameters

port	Le nom du port série à utiliser pour communiquer avec le fauteuil.

6.1.3.15 void EP::Core::setFileSave (char * *name*)

Parameters

name	Le nouveau fichier de sauvegarde.

6.1.3.16 void EP::Core::setIconSize (int *size*)

Permet de changer la taille des icônes.

Parameters

size	Un chiffre correspondant à la nouvelle taille des icônes.

6.1.3.17 void EP::Core::setThemeId (int *id*)

Permet de changer le thème utilisé.

Parameters

id	L'identifiant du nouveau thème.

The documentation for this class was generated from the following files:

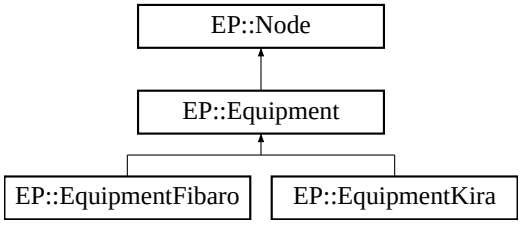
- ModelDll/include/Core.h
- ModelDll/src/Core.cpp

6.2 EP::Equipment Class Reference

Représente un équipement. Cette classe abstraite est spécialisée par EquipmentKira et EquipmentFibaro.

`#include <Equipment.h>`

Inheritance diagram for EP::Equipment:



Public Member Functions

- Equipment (char *name, char *ico, Node *parent, int typeOf)

  Appelle le constructeur de Node, initialise m_roomParent et m_typeOf.
- virtual ~Equipment ()
- virtual int sendRequest ()=0

  Envoie la requête HTTP permettant d'intéragir avec l'équipement.
- int getTypeOf ()
- Node * getNodeParent ()

Static Public Member Functions

- static char * getIpKira ()
- static char * getIpFibaro ()
- static char * getLoginFibaro ()
- static char * getPasswordFibaro ()
- static int setIpKira (char *new_ip)
- static int setIpFibaro (char *new_ip)
- static int setLoginFibaro (char *new_login)
- static int setPasswordFibaro (char *new_password)

Protected Attributes

- Node * m_roomParent

  Un pointeur vers la pièce contenant l'équipement.
- int m_typeOf

  1: pour un équipement lié à la Kira, 2: pour un équipement lié à la Fibaro.

Static Protected Attributes

- static char IP_Kira [15] = "192.168.1.31"

  L'adresse IP de la Kira.
- static char IP_Fibaro [15] = "192.168.81.1"

  L'adresse IP de la Fibaro.
- static char Fibaro_login [300] = "admin"

  Le login de la Fibaro.
- static char Fibaro_password [300] = "admin"

  Le password de la Fibaro.

6.2.1 Detailed Description

Représente un équipement. Cette classe abstraite est spécialisée par EquipmentKira et EquipmentFibaro.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 EP::Equipment::Equipment (char * *name*, char * *ico*, Node * *parent*, int *typeOf*) [inline]

Appelle le constructeur de Node, initialise m_roomParent et m_typeOf.

Parameters

parent	La pièce contenant l'équipement
typeOf	1: pour un équipement lié à la Kira, 2: pour un équipement lié à la Fibaro.

6.2.2.2 virtual EP::Equipment::~Equipment ( ) [inline], [virtual]

6.2.3 Member Function Documentation

6.2.3.1 static char* EP::Equipment::getIpFibaro ( ) [inline], [static]

Returns

L'adresse IP de la Fibaro.

6.2.3.2 static char* EP::Equipment::getIpKira ( ) [inline], [static]

Returns

L'adresse IP de la Kira.

6.2.3.3 static char* EP::Equipment::getLoginFibaro ( ) [inline], [static]

Returns

Le login de la Fibaro.

6.2.3.4 Node* EP::Equipment::getNodeParent ( ) [inline]

Returns

L'attribut m_roomParent.

6.2.3.5 static char* EP::Equipment::getPasswordFibaro ( ) [inline], [static]

Returns

Le password de la Fibaro.

6.2.3.6 int EP::Equipment::getTypeOf ( ) [inline]

Returns

L'attribut m_typeOf.

6.2.3.7 virtual int EP::Equipment::sendRequest ( ) [pure virtual]

Envoie la requête HTTP permettant d'intéragir avec l'équipement.

Implemented in EP::EquipmentFibaro, and EP::EquipmentKira.

6.2.3.8 int EP::Equipment::setIpFibaro (char * *new_ip*) [static]

Parameters

new_ip	La nouvelle adresse IP de la Fibaro.

Returns

0 si l'adresse est incorrecte, 1 sinon.

6.2.3.9 int EP::Equipment::setIpKira (char * *new_ip*) [static]

Parameters

new_ip	La nouvelle adresse IP de la Kira.

Returns

0 si l'adresse est incorrecte, 1 sinon.

6.2.3.10 int EP::Equipment::setLoginFibaro (char * *new_login*) [static]

Parameters

new_login	Le nouveau login de la Fibaro.

6.2.3.11 int EP::Equipment::setPasswordFibaro (char * *new_password*) [static]

Parameters

new_password	Le nouveau password de la Fibaro.

6.2.4 Member Data Documentation

6.2.4.1 char EP::Equipment::Fibaro_login = "admin" [static], [protected]

Le login de la Fibaro.

6.2.4.2 char EP::Equipment::Fibaro_password = "admin" [static], [protected]

Le password de la Fibaro.

6.2.4.3 char EP::Equipment::IP_Fibaro = "192.168.81.1" [static], [protected]

L'adresse IP de la Fibaro.

6.2.4.4 char EP::Equipment::IP_Kira = "192.168.1.31" [static], [protected]

L'adresse IP de la Kira.

6.2.4.5 Node* EP::Equipment::m_roomParent [protected]

Un pointeur vers la pièce contenant l'équipement.

6.2.4.6 int EP::Equipment::m_typeOf [protected]

1: pour un équipement lié à la Kira, 2: pour un équipement lié à la Fibaro.

The documentation for this class was generated from the following files:

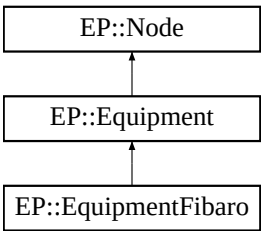
- ModelDll/include/Equipment.h
- ModelDll/src/Equipment.cpp

6.3 EP::EquipmentFibaro Class Reference

Représente un équipement lié à une Fibaro, hérite de Equipment.

`#include <Equipment.h>`

Inheritance diagram for EP::EquipmentFibaro:



Public Member Functions

- EquipmentFibaro (char *name, char *ico, Node *parent, int equipmentId, char *action)
  Appelle le constructeur de Equipment et initialise m_equipmentId et m_action.
- virtual ~EquipmentFibaro ()
- virtual int sendRequest ()
  Exécute la requête HTTP correspondant à cet équipement.
- int getEquipmentId ()
- char * getAction ()

Additional Inherited Members

6.3.1 Detailed Description

Représente un équipement lié à une Fibaro, hérite de Equipment.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 EP::EquipmentFibaro::EquipmentFibaro (char * *name*, char * *ico*, Node * *parent*, int *equipmentId*, char * *action*)

Appelle le constructeur de Equipment et initialise m_equipmentId et m_action.

Parameters

equipment↩ Id	L'identifiant correspondant à l'équipement dans l'interface web de la Fibaro.
action	L'action que l'équipement devra effectuer.

6.3.2.2 EP::EquipmentFibaro::~EquipmentFibaro ( ) [virtual]

6.3.3 Member Function Documentation

6.3.3.1 char * EP::EquipmentFibaro::getAction ( )

Returns

L'action que l'équipement devra effectuer.

6.3.3.2 int EP::EquipmentFibaro::getEquipmentId ( )

Returns

L'identifiant correspondant à l'équipement dans l'interface web de la Fibaro.

6.3.3.3 int EP::EquipmentFibaro::sendRequest ( ) [virtual]

Exécute la requête HTTP correspondant à cet équipement.

Implements EP::Equipment.

The documentation for this class was generated from the following files:

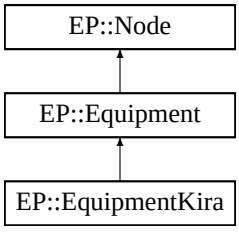
- ModelDll/include/Equipment.h
- ModelDll/src/Equipment.cpp

6.4 EP::EquipmentKira Class Reference

Représente un équipement lié à une Kira, hérite de Equipment.

`#include <Equipment.h>`

Inheritance diagram for EP::EquipmentKira:



Public Member Functions

- EquipmentKira (char *name, char *ico, Node *parent, int buttonId, int page)

  Appelle le constructeur de Equipment et initialise m_buttonId et m_page.
- virtual ~EquipmentKira ()
- virtual int sendRequest ()

  Exécute la requête HTTP correspondant à cet équipement.
- int getButtonId ()
- int getPageNumber ()
- int setButtonId (int new_id)
- int setPageNumber (int new_PageNumber)

Additional Inherited Members

6.4.1 Detailed Description

Représente un équipement lié à une Kira, hérite de Equipment.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 EP::EquipmentKira::EquipmentKira (char * *name,* char * *ico,* Node * *parent,* int *buttonId,* int *page*)

Appelle le constructeur de Equipment et initialise m_buttonId et m_page.

Parameters

buttonId	Le numéro du bouton correspondant à l'équipement dans l'interface web de la Kira.
page	La page du bouton correspondant à l'équipement dans l'interface web de la Kira.

6.4.2.2 EP::EquipmentKira::~EquipmentKira ( ) [virtual]

6.4.3 Member Function Documentation

6.4.3.1 int EP::EquipmentKira::getButtonId ( )

Returns

Le numéro du bouton correspondant à l'équipement dans l'interface web de la Kira.

6.4.3.2 int EP::EquipmentKira::getPageNumber ( )

Returns

Le numéro de page du bouton correspondant à l'équipement dans l'interface web de la Kira.

6.4.3.3 int EP::EquipmentKira::sendRequest ( ) [virtual]

Exécute la requête HTTP correspondant à cet équipement.

Implements EP::Equipment.

6.4.3.4 int EP::EquipmentKira::setButtonId (int *new_id*)

Parameters

new_id	Le nouveau numéro du bouton.

6.4.3.5 int EP::EquipmentKira::setPageNumber (int *new_PageNumber*)

Parameters

new_PageNumber	Le nouveau numéro de page du bouton.

The documentation for this class was generated from the following files:

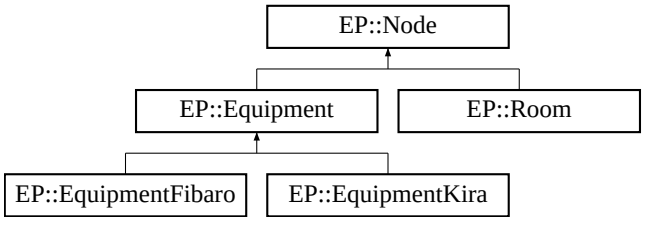
- ModelDll/include/Equipment.h
- ModelDll/src/Equipment.cpp

6.5 EP::Node Class Reference

Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes Equipment et Room, et n'est donc pas utilisée telle qu'elle.

```
#include <Node.h>
```

Inheritance diagram for EP::Node:



Public Member Functions

- Node (char *name, char *ico)

  Constructeur, initialise les attributs m_name et m_ico.
- virtual ~Node (void)

  Destructeur, ne fait rien de particulier.
- char * getName ()
- void setName (char *name)

  Change le nom de ce noeud pour celui passé en paramètre.
- char * getIco ()
- void setIco (char *ico)

  Change l'icône du noeud.

Protected Attributes

- char m_name [100]

  Le nom du noeud.
- char m_ico [100]

  L'icône du noeud qui sera affichée dans l'application.
- EP::Node * m_parent

  Le noeud parent de celui-ci. Sera nul pour les pièces (Room), et correspondra à la pièce contenante dans le cas d'un équipement (Equipment).

6.5.1 Detailed Description

Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes Equipment et Room, et n'est donc pas utilisée telle qu'elle.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 EP::Node::Node (char * *name*, char * *ico*)

Constructeur, initialise les attributs m_name et m_ico.

6.5.2.2 EP::Node::~Node (void) [virtual]

Destructeur, ne fait rien de particulier.

6.5.3 Member Function Documentation

6.5.3.1 char * EP::Node::getIco ( )

Returns

Le nom du fichier correspondant à l'icône de ce noeud.

6.5.3.2 char * EP::Node::getName ( )

Returns

Le nom du noeud.

6.5.3.3 void EP::Node::setIco (char * *ico*)

Change l'icône du noeud.

Parameters

ico	La nouvelle icône du noeud.

6.5.3.4 void EP::Node::setName (char * *name*)

Change le nom de ce noeud pour celui passé en paramètre.

Parameters

name	Le nouveau nom du noeud.

6.5.4 Member Data Documentation

6.5.4.1 char EP::Node::m_ico[100] [protected]

L'icône du noeud qui sera affichée dans l'application.

6.5.4.2 char EP::Node::m_name[100] [protected]

Le nom du noeud.

6.5.4.3 EP::Node* EP::Node::m_parent [protected]

Le noeud parent de celui-ci. Sera nul pour les pièces (Room), et correspondra à la pièce contenante dans le cas d'un équipement (Equipment).

The documentation for this class was generated from the following files:

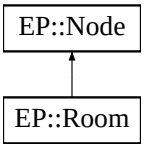
- ModelDll/include/Node.h
- ModelDll/src/Node.cpp

6.6 EP::Room Class Reference

Représente une pièce de la maison ; Hérite de Node. On peut y retrouver les différents équipements liés à une pièce et les gérer.

`#include <Room.h>`

Inheritance diagram for EP::Room:



Public Member Functions

- Room (char *name, char *ico)

  Appele le constructeur de Node.
- ~Room ()

  Détruit la liste d'équipements.
- int addEquipment (Equipment *equip)

  Ajoute le pointeur d'un équipement donné dans m_listEquipments. Le nom d'un équipement doit être unique.
- int deleteEquipmentByIndex (int index)

  Supprime l'équipement dont l'index est donné en paramètre.
- int deleteEquipmentByName (char *name)

  Supprime l'équipement dont le nom est donné en paramètre.
- std::vector< Equipment * > * getEquipments ()

  Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec getNumberEquipments et getEquipmentByIndex.
- Equipment * getEquipmentByName (char *name)

  Donne un pointeur vers l'équipement dont le nom est donné en paramètre.
- Equipment * getEquipmentByIndex (int index)

  Donne un pointeur vers l'équipement dont l'index est donné en paramètre.
- int getNumberEquipments ()

Additional Inherited Members

6.6.1 Detailed Description

Représente une pièce de la maison ; Hérite de Node. On peut y retrouver les différents équipements liés à une pièce et les gérer.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 EP::Room::Room (char * *name*, char * *ico*)

Appele le constructeur de Node.

6.6.2.2 EP::Room::~Room ( )

Détruit la liste d'équipements.

6.6.3 Member Function Documentation

6.6.3.1 int EP::Room::addEquipment (Equipment * *equip*)

Ajoute le pointeur d'un équipement donné dans m_listEquipments. Le nom d'un équipement doit être unique.

Parameters

equip	Le pointeur de l'équipement à ajouter.

Returns

0 si tout s'est bien passé, 1 si le nom est déjà pris.

6.6.3.2 int EP::Room::deleteEquipmentByIndex (int *index*)

Supprime l'équipement dont l'index est donné en paramètre.

Parameters

index	L'index de l'équipement à supprimer de m_listEquipments.

Returns

0 si tout s'est bien passé, 1 si l'index ne correspond à aucun équipement.

6.6.3.3 int EP::Room::deleteEquipmentByName (char * *name*)

Supprime l'équipement dont le nom est donné en paramètre.

Parameters

name	Le nom de l'équipement à supprimer de m_listEquipments.

Returns

0 si tout s'est bien passé, 1 si le nom ne correspond à aucun équipement.

6.6.3.4 Equipment * EP::Room::getEquipmentByIndex (int *index*)

Donne un pointeur vers l'équipement dont l'index est donné en paramètre.

Parameters

index	L'index de l'équipement qu'on veut récupérer.

Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

6.6.3.5 Equipment * EP::Room::getEquipmentByName (char * *name*)

Donne un pointeur vers l'équipement dont le nom est donné en paramètre.

Parameters

name	Le nom de l'équipement qu'on veut récupérer.

Returns

Un pointeur vers l'équipement demandé, NULL si aucun ne correspond.

6.6.3.6 vector< Equipment * > * EP::Room::getEquipments ( )

Ne pas utiliser cette méthode à partir de la DLL. A la place on peut itérer avec getNumberEquipments et get↩ EquipmentByIndex.

Returns

L'attribut m_listEquipments.

6.6.3.7 int EP::Room::getNumberEquipments ( )

Returns

Le nombre d'équipement que la pièce contient

The documentation for this class was generated from the following files:

- ModelDll/include/Room.h
- ModelDll/src/Room.cpp

Chapter 7

File Documentation

7.1 ModelDll/include/Core.h File Reference

```
#include "Room.h"
```

Classes

- class EP::Core

  Le coeur de l'application, contient la liste des pièces, les paramètres de l'application ainsi que les méthodes pour gérer ceux-ci.

Namespaces

- EP

Macros

- #define DllExport __declspec(dllexport)
- #define FILESAVE_NAME_SIZE 500

Functions

- DllExport Core * EP::Core_New (char *file)

  Constructeur statique utilisé pour permettre l'utilisation des objets Core en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport Core * EP::Core_NewFromSave (char *file)

  Constructeur statique utilisé pour permettre l'utilisation des objets Core en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur. Après création de l'objet, la méthode Core::load() est appelée sur celui-ci.
- DllExport void EP::Core_Delete (Core *core)

  Destructeur statique, pour permettre la destruction des objets Core en passant par la DLL.
- DllExport void EP::Core_SaveAndDelete (Core *core)

  Destructeur statique, pour permettre la destruction des objets Core en passant par la DLL. La méthode Core::save() est appelée avant sa destruction.

7.1.1 Macro Definition Documentation

7.1.1.1 #define DllExport __declspec(dllexport)

7.1.1.2 #define FILESAVE_NAME_SIZE 500

7.2 ModelDll/include/Equipment.h File Reference

```
#include "Node.h"
```

Classes

- class EP::Equipment

  Représente un équipement. Cette classe abstraite est spécialisée par EquipmentKira et EquipmentFibaro.
- class EP::EquipmentKira

  Représente un équipement lié à une Kira, hérite de Equipment.
- class EP::EquipmentFibaro

  Représente un équipement lié à une Fibaro, hérite de Equipment.

Namespaces

- EP

Macros

- #define DllExport __declspec(dllexport)

Functions

- DllExport char * EP::Equipment_getIpKira ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport char * EP::Equipment_getIpFibaro ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int EP::Equipment_setIpKira (char *new_ip)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int EP::Equipment_setIpFibaro (char *new_ip)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport char * EP::Equipment_getLoginFibaro ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport char * EP::Equipment_getPasswordFibaro ()

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int EP::Equipment_setLoginFibaro (char *new_login)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.
- DllExport int EP::Equipment_setPasswordFibaro (char *new_password)

  Utiliser cette méthode plutôt que la méthode statique de la classe équipement lorsqu'on passe par la DLL.

- DllExport EquipmentKira * EP::EquipmentKira_New (char *name, char *ico, Node *parent, int buttonId, int page)

  Constructeur statique utilisé pour permettre l'utilisation des objets EquipmentKira en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void EP::EquipmentKira_Delete (EquipmentKira *eq)

  Destructeur statique, pour permettre la destruction des objets EquipmentKira en passant par la DLL.
- DllExport EquipmentFibaro * EP::EquipmentFibaro_New (char *name, char *ico, Node *parent, int equipmentId, char *action)

  Constructeur statique utilisé pour permettre l'utilisation des objets EquipmentFibaro en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void EP::EquipmentFibaro_Delete (EquipmentFibaro *eq)

  Destructeur statique, pour permettre la destruction des objets EquipmentFibaro en passant par la DLL.

7.2.1 Macro Definition Documentation

7.2.1.1 #define DllExport __declspec(dllexport)

7.3 ModelDll/include/Node.h File Reference

Classes

- class EP::Node

  Représente un noeud de l'arbre (une pièce ou un équipement). Cette classe est spécialisée par les classes Equipment et Room, et n'est donc pas utilisée telle qu'elle.

Namespaces

- EP

Macros

- #define DllExport __declspec(dllexport)

Functions

- DllExport Node * EP::Node_New (char *name, char *ico)

  Constructeur statique utilisé pour permettre l'utilisation des objets Node en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void EP::Node_Delete (Node *node)

  Destructeur statique, pour permettre la destruction des objets Node en passant par la DLL.

7.3.1 Macro Definition Documentation

7.3.1.1 #define DllExport __declspec(dllexport)

7.4 ModelDll/include/RequeteHttp.h File Reference

Namespaces

- EP

Macros

- #define DllImport __declspec(dllimport)

Functions

- void DllImport EP::requeteHttpKira (char *s1, char *s2)

  Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Kira. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode EquipmentKira::sendRequest() ; les paramètres sont construits automatiquement à partir des informations du l'équipement.
- void DllImport EP::requeteHttpFibaro (char *s1, char *s2, char *user, char *pass)

  Cette méthode permet d'envoyer une requête HTTP pour un équipement lié à la Fibaro. Cette méthode est un import de la DLL RequeteHttp.dll et est utilisée uniquement par la méthode EquipmentFibaro::sendRequest() ; les paramètres sont construits automatiquement à partir des informations du l'équipement.

7.4.1 Macro Definition Documentation

7.4.1.1 #define DllImport __declspec(dllimport)

7.5 ModelDll/include/Room.h File Reference

```
#include <vector>
#include "Equipment.h"
```

Classes

- class EP::Room

  Représente une pièce de la maison ; Hérite de Node. On peut y retrouver les différents équipements liés à une pièce et les gérer.

Namespaces

- EP

Macros

- #define DllExport __declspec(dllexport)

Functions

- DllExport Room * EP::Room_New (char *name, char *ico)

  Constructeur statique utilisé pour permettre l'utilisation des objets Room en passant par la DLL. Les paramètres sont les mêmes que ceux du constructeur.
- DllExport void EP::Room_Delete (Room *room)

  Destructeur statique, pour permettre la destruction des objets Room en passant par la DLL.

7.5.1 Macro Definition Documentation

7.5.1.1 #define DllExport __declspec(dllexport)

7.6 ModelDll/src/Core.cpp File Reference

```
#include "../include/Core.h"
#include <fstream>
#include <string>
#include <iostream>
```

Namespaces

- EP

Functions

- EP::__declspec (dllexport) Core *Core_New(char *file)

7.7 ModelDll/src/Equipment.cpp File Reference

```
#include "..\include\Equipment.h"
#include "..\include\RequeteHttp.h"
#include <iostream>
```

Namespaces

- EP

Functions

- EP::__declspec (dllexport) Core *Core_New(char *file)

Variables

- char * EP::ico
- char Node * EP::parent
- char Node int EP::buttonId
- char Node int int EP::page
- char Node int EP::equipmentId
- char Node int char * EP::action

7.8 ModelDll/src/Node.cpp File Reference

```
#include "../include/Node.h"
#include <iostream>
```

Namespaces

- EP

Functions

- EP::__declspec (dllexport) Core *Core_New(char *file)

7.9 ModelDll/src/Room.cpp File Reference

```
#include "../include/Room.h"
```

Namespaces

- EP

Functions

- EP::__declspec (dllexport) Core *Core_New(char *file)

Index