versione 1.0
ultima modifica 30/08/2008

classe StreamStorage

realizzazione di una interfaccia C++ per l'accesso ai file di documento (compound files).

#include "streamstorage.h"

Vedi
interfaccia StreamStorage
esempio di utilizzo di StreamStorage

membri della classe StreamStorage

Membri pubblici

-	nessun dato esportato

Costruzione/Distruzione

StreamStorage	costruttore predefinito
~StreamStorage	distruttore predefinito

Documenti

AttachCompoundFile	stabilisce la connessione ad un documento su disco
DetachCompoundFile	chiude il documento correntemente aperto
GetFileName	ritorna il nome del documento aperto
GetFullFileName	ritorna il percorso al documento aperto
DefragCompoundFile	deframmenta un documento
OpenBrowser	visualizza il contenuto del documento aperto
ItemProperties	fornisce alcuni parametri generali di file e direttori

Direttori del documento

ChDir	cambia il direttorio corrente
ChDirRoot	imposta il direttorio corrente su quello radice
ChDirDotDot	sposta il direttorio corrente al livello superiore
MkDir	crea un nuovo direttorio
RmDir	elimina un direttorio esistente
Dir	elenca il contenuto del direttorio corrente

File del documento

OpenFile	apre un file
CloseFile	chiude il file aperto
OpenArchive	apre un file come archivio di serializzazione
CloseArchive	chiude l'archivio aperto
Del	elimina un file da un direttorio
Ren	rinomina un file di un direttorio

StreamStorage::StreamStorage()

StreamStorage()

costruttore predefinito.

Valore di ritorno
nessuno.

Parametri
nessuno.

Note
nessuna.

Vedi
~StreamStorage

StreamStorage::~StreamStorage()

~StreamStorage()

distruttore predefinito.

Valore di ritorno
nessuno.

Parametri
nessuno.

Note
chiude l'eventuale file o archivio aperto e/o l'eventuale documento aperto.

Vedi
StreamStorage

StreamStorage::AttachCompoundFile()

BOOL AttachCompoundFile(LPCTSTR pCompFileName, BOOL readOnly = TRUE, BOOL create = FALSE)

stabilisce la connessione ad un documento su disco.

Valore di ritorno
TRUE se la connessione e' stabilita con successo, FALSE se si e' verificato qualche errore.

Parametri

pCompFileName	percorso al documento sul disco. Puo' essere assoluto o relativo.
readOnly	se TRUE, il documento e' aperto con modalita' a sola lettura.
create	se TRUE ed il documento non esiste, ne crea uno nuovo inizialmente vuoto.

Note
E' possibile connettere un oggetto StreamStorage ad un documento alla volta. Per connettere un altro documento, chiudere prima quello aperto, oppure istanziare un nuovo oggetto StreamStorage. La funzione **AttachCompoundFile** chiude automaticamente ogni documento precedentemente aperto prima di effettuare la connessione richiesta.

Vedi
DetachCompoundFile

StreamStorage::DetachCompoundFile()

BOOL DetachCompoundFile(BOOL defrag = FALSE)

disconnette il documento correntemente aperto.

Valore di ritorno
TRUE se la chiusura ha avuto successo, FALSE se si e' verificato qualche errore.

Parametri
defrag se TRUE, il documento viene compattato dopo la chiusura.

Note
AttachCompoundFile e il distruttore predefinito chiudono automaticamente l'eventuale documento aperto, cosicche' non e' indispensabile (ma comunque raccomandato) richiamare esplicitamente questa funzione.

Vedi
AttachCompoundFile

StreamStorage::GetFileName()

CString GetFileName()

ritorna il nome del documento aperto.

Valore di ritorno
un oggetto CString contenente il nome del documento correntemente aperto.

Parametri
nessuno.

Note
Se l'oggetto StreamStorage non e' attualmente connesso ad alcun documento, la stringa ritornata e' vuota.

Vedi
GetFullFileName

StreamStorage::GetFullFileName()

CString GetFullFileName()

ritorna il percorso al documento aperto.

Valore di ritorno
un oggetto CString contenente il percorso completo del documento correntemente aperto.

Parametri
nessuno.

Note
Se l'oggetto StreamStorage non e' attualmente connesso ad alcun documento, la stringa ritornata e' vuota.

Vedi
GetFileName

StreamStorage::DefragCompoundFile()

BOOL DefragCompoundFile(LPCTSTR pCompFileName)

deframmenta un documento.

Valore di ritorno
TRUE se la compattazione ha avuto successo; in caso contrario FALSE.

Parametri
pCompFileName percorso al documento sul disco. Puo' essere assoluto o relativo.

Note
Il documento da compattare non deve essere attualmente connesso ad oggetti StreamStorage o utilizzato (anche solo limitatamente in lettura) da nessun altro programma o applicazione.

Vedi

StreamStorage::OpenBrowser()

BOOL OpenBrowser(LPCTSTR pTitle = NULL)

visualizza il contenuto del documento aperto.

Valore di ritorno
TRUE se la finestra e' stata aperta con successo; in caso contrario FALSE.

Parametri
pTitle titolo della finestra di visualizzazione.

Note
Presuppone la connessione ad un documento.

Vedi

StreamStorage::ItemProperties()

BOOL ItemProperties(LPCTSTR pPath, SS_ITEM_INFO* pInfo)

fornisce alcuni parametri generali di file e direttori.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri

pPath	percorso al file o direttorio da esaminare. Puo' essere assoluto o relativo.
pInfo	puntatore ad una struttura **SS_ITEM_INFO** destinato a contenere le informazioni relative all'oggetto specificato.

Note
ItemProperties fornisce le seguenti proprieta' di file e direttorio:

```
struct SS_ITEM_INFO
{
	CString		name;
	SS_ITEM_TYPE	type;
	DWORD		size;
	FILETIME	created;
	FILETIME	accessed;
	FILETIME	modified;
}
```

ove il tipo SS_ITEM_TYPE e' definito come segue:

```
enum SS_ITEM_TYPE { SS_ROOT, SS_DIR, SS_FILE, };
```

La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento.

Vedi
Dir

StreamStorage::ChDir()

CString ChDir()
BOOL ChDir(LPCTSTR pDirPath)

ritorna/cambia il direttorio corrente.

Valore di ritorno
la versione senza parametri ritorna una stringa contenente il percorso assoluto al direttorio corrente, la versione con parametri ritorna TRUE se il comando e' stato eseguito con successo, FALSE in caso contrario.

Parametri
pDirPath percorso assoluto o relativo al nuovo direttorio corrente.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure il direttorio non esiste o non e' permesso accedervi.

Vedi
ChDirRoot, ChDirDotDot, Dir

StreamStorage::ChDirRoot()

BOOL ChDirRoot()

imposta il direttorio corrente su quello radice.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
nessuno.

Note
Comando equivalente a **ChDir(_T("\\"))**; la funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il direttorio radice non e' accessibile.

Vedi
ChDir, ChDirDotDot

StreamStorage::ChDirDotDot()

BOOL ChDirDotDot()

sposta il direttorio corrente al livello superiore.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
nessuno.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il direttorio posto al livello superiore non e' accessibile.

Vedi
ChDir, ChDirRoot

StreamStorage::MkDir()

BOOL MkDir(LPCTSTR pDirPath)

crea un nuovo direttorio.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
pDirPath percorso assoluto o relativo al nuovo direttorio da creare.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il percorso al direttorio da creare non esiste o non e' permesso accedervi.

Vedi
RmDir

StreamStorage::RmDir()

BOOL RmDir(LPCTSTR pDirPath)

elimina un direttorio esistente.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
pDirPath percorso assoluto o relativo al direttorio da eliminare.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il percorso al direttorio da creare non esiste o non e' permesso accedervi. Se il direttorio esiste, esso viene eliminato con tutto il suo contenuto, sotto-direttori compresi.

Vedi
MkDir

StreamStorage::Dir()

BOOL Dir(SS_ITEM_INFO* pItems, int* pCount);

elenca il contenuto del direttorio corrente.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri

pItems	puntatore ad un vettore di strutture **SS_ITEM_INFO** destinato a contenere le informazioni relative agli oggetti contenuti dal direttorio corrente; se impostato a NULL, il trasferimento di dati non ha luogo.
pCount	puntatore ad una variabile **int** che riceve il numero di oggetti memorizzati all'interno del direttorio corrente.

Note
Dir fornisce le seguenti informazioni per ogni oggetto del direttorio corrente:

```
struct SS_ITEM_INFO
{
     CString         name;
     SS_ITEM_TYPE    type;
     DWORD           size;
     FILETIME        created;
     FILETIME        accessed;
     FILETIME        modified;
}
```

ove il tipo SS_ITEM_TYPE e' definito come segue:

```
enum SS_ITEM_TYPE { SS_ROOT, SS_DIR, SS_FILE, };
```

La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento.

Vedi
ChDir, ItemProperties

StreamStorage::OpenFile()

BOOL OpenFile(LPCTSTR pFilePath**, COleStreamFile**** pFile**,** **BOOL** readOnly **= TRUE, BOOL** create **= FALSE)**

apre un file.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri

pFilePath	percorso al file da aprire all'interno del documento. Puo' essere assoluto o relativo.
pFile	puntatore ad un oggetto **COleStreamFile** per l'accesso al file aperto.
readOnly	se TRUE, il file e' aperto con modalita' a sola lettura.
create	se TRUE ed il file non esiste, ne viene creato uno nuovo inizialmente vuoto.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il percorso al file da aprire non esiste o non e' permesso accedervi (es. si richiede un accesso al file di tipo read/write ma il documento e' stato aperto a sola lettura). E' possibile aprire un solo file alla volta. Per aprire un altro file, chiudere prima quello aperto. La funzione **OpenFile** chiude automaticamente ogni file precedentemente aperto prima di aprirne uno nuovo.

*L'utilizzo dell'interfaccia COleStreamFile ottenuta dalla chiamata **OpenFile** dovrebbe limitarsi ai metodi COleStreamFile::Read e COleStreamFile::Write. Qualunque altro metodo puo' generare effetti collaterali incontrollabili.*

Vedi
CloseFile, OpenArchive

StreamStorage::CloseFile()

BOOL CloseFile(BOOL trunc = FALSE)

chiude il file aperto.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
trunc se TRUE, il file e' troncato alla posizione del puntatore di lettura/scrittura.

Note
nessuna.

Vedi
OpenFile

StreamStorage::OpenArchive()

BOOL OpenFile(LPCTSTR pFilePath**, CArchive**** pArchive**,**
BOOL load **= TRUE, BOOL** create **= FALSE)**

apre un file come archivio di serializzazione.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri

pFilePath	percorso al file da aprire all'interno del documento. Puo' essere assoluto o relativo.
pArchive	puntatore ad un oggetto **CArchive** per l'accesso serializzato al file aperto.
load	se TRUE, l'archivio e' aperto con modalita' a sola lettura.
create	se TRUE e l'archivio non esiste, ne viene creato uno nuovo inizialmente vuoto.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il percorso al file di archivio da aprire non esiste o non e' permesso accedervi (es. si richiede un accesso al file di tipo read/write ma il documento e' stato aperto a sola lettura). E' possibile aprire un solo file di archivio alla volta. Per aprire un altro archivio, chiudere prima quello aperto. La funzione **OpenArchive** chiude automaticamente ogni file di archivio precedentemente aperto prima di aprirne uno nuovo.

*L'utilizzo dell'interfaccia CArchive ottenuta dalla chiamata **OpenArchive** dovrebbe limitarsi ai metodi CArchive::operator<< e CArchive::operator>>. Qualunque altro metodo puo' generare effetti collaterali incontrollabili.*

Vedi
CloseArchive, OpenFile

StreamStorage::CloseArchive()

BOOL CloseArchive()

chiude l'archivio aperto.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
nessuno.

Note
nessuna.

Vedi
OpenArchive

StreamStorage::Del()

BOOL Del(LPCTSTR pFilePath)

elimina un file da un direttorio.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri
pFilePath percorso al file da eliminare.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il percorso al file da eliminare non esiste o non e' permesso accedervi (es. il documento e' stato aperto a sola lettura).

Vedi
Ren

StreamStorage::Ren()

BOOL Ren(LPCTSTR pFilePath, LPCTSTR pNewName)

rinomina un file di un direttorio.

Valore di ritorno
TRUE se il comando e' stato eseguito con successo; in caso contrario FALSE.

Parametri

pFilePath	percorso al file da rinominare.
pNewName	nuovo nome del file.

Note
La funzione genera un errore se l'oggetto StreamStorage non e' connesso con alcun documento oppure se il percorso al file da rinominare non esiste o non e' permesso accedervi (es. il documento e' stato aperto a sola lettura).

Vedi
Del

esempio di utilizzo di StreamStorage

```
// no special purpose construction needed
StreamStorage ss;

// connect to the desired compound file
LPCTSTR pCompoundFileName = ...
ss.AttachCompoundFile(pFileName);


// travel the directory structure using either
// absolute or relative path specifications
LPCTSTR pPath = ...
ss.ChDir(pPath);

// same as "CD .." DOS command
ss.ChDirDotDot();

// same as "CD \" DOS command
ss.ChDirRoot();

// modify the directory structure creating new directories...
LPCTSTR pNewDirPath = ...
ss.MkDir(pNewDirPath);

// or removing old ones (directories are removed even if they are not empty)
LPCTSTR pOldDirPath = ...
ss.RmDir(pOldDirPath);


// inspect the current directory contents
int nItems;
ss.Dir(NULL,&nItems);

StreamStorage::SS_ITEM_INFO* pInfo = new StreamStorage::SS_ITEM_INFO[nItems];
ss.Dir(pInfo,&nItems);


// you can access files with a CFile like interface
COleStreamFile* pFile;
LPCTSTR pFilePath = ...
ss.OpenFile(pFilePath,&pFile);

// you should use only the CFile Read/Write member functions
pFile->Read(...);
pFile->Write(...);

// remember to close the opened file
ss.CloseFile();


// you can access files with a CArchive like interface
CArchive* pArchive;
LPCTSTR pFilePath = ...
ss.OpenArchive(pFilePath,&pArchive);

// you should use only the CArchive << >> operators member functions
CObject serial;
*pArchive << serial;
*pArchive >> serial;

// remember to close the opened archive
ss.CloseArchive();


// release the compound file
ss.DetachCompoundFile();
```