News Feeder Shared Database Library (nfdb)

Generated by Doxygen 1.8.1.1

Fri Oct 26 2012 14:34:18

Contents

Chapter 1

Namespace Documentation

1.1 nfdb Namespace Reference

Classes

- class Comment

  A class representing a row in the Comments table.
- class CommentController

  A class called to create, update, delete or find comments.
- class ContentPlaceholder

  A class representing a row in the ContentPlaceholders table.
- class ContentPlaceholderController

  A class called to create, update, delete or find content placeholders.
- class DatabaseController

  A class called to connect and disconnect from the database.
- class Feed

  A class representing a row in the Feeds table.
- class FeedController

  A class called to create, update, delete or find feeds.
- class GroupPlaceholder

  A class representing a row in the GroupPlaceholders table.
- class GroupPlaceholderController

  A class called to create, update, delete or find group placeholders.
- class Image

  A class representing a row in the Images table.
- class ImageController

  A class called to create, update, delete or find images.
- class Item

  A class representing a row in the Items table.
- class ItemController

  A class called to create, update, delete or find items.
- class Layout

  A class representing a row in the Layout table.
- class LayoutController

  A class called to create, update, delete or find layouts.
- class Notification

  A class representing a row in the Notification table.

- class NotificationController

  A class called to create, update, delete or find notifications.
- class QueueItem

  A class representing a QueueItem for the queue, derived from the Item table.
- class Session

  A class representing a row in the Session table.
- class SessionController

  A class called to create, update, delete or find sessions.
- class Sheet

  A class representing a row in the Sheet table.
- class SheetController

  A class called to create, update, delete or find sheets.
- class Stat

  A class representing a row in the Stat table.
- class StatController

  A class called to create, update, delete or find stats.
- class User

  A class representing a row in the User table.
- class UserController

  A class called to create, update, delete or find users.

Chapter 2

Class Documentation

2.1 nfdb::Comment Class Reference

A class representing a row in the Comments table.

```
#include <Comment.h>
```

Public Member Functions

- Comment ()

  Initialising Constructor for Comment.
- Comment (int id, int itemId, std::string author, std::string via, nfrd::misc::DateTime date, std::string comment, char ∗avatar, int avatarSize)

  Non-Default Constructor for Comment.
- ∼Comment ()

  Destructor for Comment.
- void Destroy ()

  Cleans up any memory held by the Comment.

Public Attributes

- int id
- int itemId
- std::string author
- std::string via
- nfrd::misc::DateTime date
- std::string comment
- char ∗ avatar
- int avatarSize

2.1.1 Detailed Description

A class representing a row in the Comments table.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 nfdb::Comment::Comment ( ) `[inline]`

Initialising Constructor for Comment.

Sets avatar to NULL

2.1.2.2 nfdb::Comment::Comment (int *id*, int *itemId*, std::string *author*, std::string *via*, nfrd::misc::DateTime *date*, std::string *comment*, char ∗ *avatar*, int *avatarSize*) `[inline]`

Non-Default Constructor for Comment.

Parameters

id	int identifier of the comment
itemId	int representing the id of the item the comment was made on
author	string representing who made the comment
via	string representing where the comment was made
date	datetime of the date the comment was made
comment	string representing the actual comment text
avatar	char∗ representing the avatar image of who sent the comment, nullable
avatarSize	int representing the number of bytes in the avatar image, 0 if not included

2.1.2.3 nfdb::Comment::∼Comment ( ) `[inline]`

Destructor for Comment.

Does not perform any actions

2.1.3 Member Function Documentation

2.1.3.1 void nfdb::Comment::Destroy ( ) `[inline]`

Cleans up any memory held by the Comment.

2.1.4 Member Data Documentation

2.1.4.1 std::string nfdb::Comment::author

2.1.4.2 char∗ nfdb::Comment::avatar

2.1.4.3 int nfdb::Comment::avatarSize

2.1.4.4 std::string nfdb::Comment::comment

2.1.4.5 nfrd::misc::DateTime nfdb::Comment::date

2.1.4.6 int nfdb::Comment::id

2.1.4.7 int nfdb::Comment::itemId

2.1.4.8 std::string nfdb::Comment::via

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

- include/Comment.h

2.2 nfdb::CommentController Class Reference

A class called to create, update, delete or find comments.

```
#include <CommentController.h>
```

Public Member Functions

- CommentController ()

  Initialising Constructor for CommentController, opens a database connection.
- ∼CommentController ()

  Deallocates any dynamic memory and closes the database connection.
- Comment ∗ GetCommentById (int id)

  Find the comment with that id.
- std::vector< Comment ∗ > GetCommentsByItemId (int itemId)

  Find all comments that belong to the item of that id.
- std::vector< Comment ∗ > GetAllComments ()

  Find all comments.
- int AddComment (Comment &comment)

  Insert the comment into the database.
- void UpdateComment (Comment &comment)

  Update the relevant comment in the database.
- void UpdateComment (int id, int ∗itemId, std::string ∗author, std::string ∗via, nfrd::misc::DateTime ∗date, std-::string ∗comment, char ∗avatar, int avatarSize)

  Update the relevant comment in the database, NULLS passed if that parameter is not to be updated.
- void RemoveComment (int id)

  Delete the comment from the database.
- Comment ∗ GenerateComment (sql::ResultSet &rs)

  Generate a comment based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.2.1 Detailed Description

A class called to create, update, delete or find comments.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 CommentController::CommentController ( )

Initialising Constructor for CommentController, opens a database connection.

2.2.2.2 CommentController::∼CommentController ( )

Deallocates any dynamic memory and closes the database connection.

2.2.3 Member Function Documentation

2.2.3.1 int CommentController::AddComment (Comment & *comment*)

Insert the comment into the database.

Parameters

comment	comment to be added to the database

Returns

id of the newly added comment, -1 for error

2.2.3.2 Comment ∗ CommentController::GenerateComment (sql::ResultSet & *rs*)

Generate a comment based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a comment

Returns

the generated comment

2.2.3.3 vector< Comment ∗ > CommentController::GetAllComments ( )

Find all comments.

Returns

all comments in the database

2.2.3.4 Comment ∗ CommentController::GetCommentById (int *id*)

Find the comment with that id.

Parameters

id	primary key id of the comment

Returns

the comment with that id

2.2.3.5 vector< Comment ∗ > CommentController::GetCommentsByItemId (int *itemId*)

Find all comments that belong to the item of that id.

Parameters

itemId	id of the item to find comments for

Returns

the comments that belong to that item

2.2.3.6 void CommentController::RemoveComment (int *id*)

Delete the comment from the database.

Parameters

comment	comment to be removed from the database

2.2.3.7 void CommentController::UpdateComment (Comment & *comment*)

Update the relevant comment in the database.

Parameters

comment	comment to be updated

2.2.3.8 void nfdb::CommentController::UpdateComment (int *id,* int ∗ *itemId,* std::string ∗ *author,* std::string ∗ *via,* nfrd::misc::DateTime ∗ *date,* std::string ∗ *comment,* char ∗ *avatar,* int *avatarSize*)

Update the relevant comment in the database, NULLS passed if that parameter is not to be updated.

Parameters

id	int identifier of the comment
itemId	int∗ representing the id of the item the comment was made on, nullable
author	string∗ representing who made the comment, nullable
via	string∗ representing where the comment was made, nullable
date	datetime of the date the comment was made, nullable
comment	string∗ representing the actual comment text, nullable
avatar	char∗ representing the avatar image of who sent the comment, nullable
avatarSize	int representing the number of bytes in the avatar image, 0 if not included

2.2.4 Member Data Documentation

2.2.4.1 sql::Connection∗ nfdb::CommentController::conn `[private]`

Connection to the database.

2.2.4.2 DatabaseController∗ nfdb::CommentController::dbc `[private]`

Database controller used to connect to the database.

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

- include/CommentController.h
- src/CommentController.cpp

2.3 nfdb::ContentPlaceholder Class Reference

A class representing a row in the ContentPlaceholders table.

```
#include <ContentPlaceholder.h>
```

Public Member Functions

- ContentPlaceholder ()

  Initialising Constructor for ContentPlaceholder.
- ContentPlaceholder (int id, int sheetId, int type, int column, int order)

  Non-Default Constructor for ContentPlaceholder.
- ∼ContentPlaceholder ()

  Destructor for ContentPlaceholder, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the ContentPlaceholder.

Public Attributes

- int id
- int sheetId
- int type
- int column
- int order

2.3.1 Detailed Description

A class representing a row in the ContentPlaceholders table.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 nfdb::ContentPlaceholder::ContentPlaceholder ( ) `[inline]`

Initialising Constructor for ContentPlaceholder.

2.3.2.2 nfdb::ContentPlaceholder::ContentPlaceholder (int *id*, int *sheetId*, int *type*, int *column*, int *order*) `[inline]`

Non-Default Constructor for ContentPlaceholder.

Parameters

id	int identifier of the contentplaceholder
sheetId	int representing the id of the sheet this placeholder is located on
type	int representing what type of content placeholder it is
column	int representing which column the placeholder is located in
order	int representing what order this placeholder is in the column

2.3.2.3 nfdb::ContentPlaceholder::∼ContentPlaceholder ( ) `[inline]`

Destructor for ContentPlaceholder, does not perform any actions.

2.3.3 Member Function Documentation

2.3.3.1 void nfdb::ContentPlaceholder::Destroy ( ) `[inline]`

Cleans up any memory held by the ContentPlaceholder.

2.3.4 Member Data Documentation

2.3.4.1 int nfdb::ContentPlaceholder::column

2.3.4.2 int nfdb::ContentPlaceholder::id

2.3.4.3 int nfdb::ContentPlaceholder::order

2.3.4.4 int nfdb::ContentPlaceholder::sheetId

2.3.4.5 int nfdb::ContentPlaceholder::type

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

- include/ContentPlaceholder.h

2.4 nfdb::ContentPlaceholderController Class Reference

A class called to create, update, delete or find content placeholders.

`#include <ContentPlaceholderController.h>`

Public Member Functions

- ContentPlaceholderController ()

  Initialising Constructor for ContentPlaceholderController, opens a database connection.
- ∼ContentPlaceholderController ()

  Deallocates any dynamic memory and closes the database connection.
- ContentPlaceholder ∗ GetContentPlaceholderById (int id)

  Find the content placeholder with that id.
- std::vector< ContentPlaceholder ∗ > GetContentPlaceholdersBySheetId (int sheetId)

  Find all content placeholders that belong to the sheet of that id.
- std::vector< ContentPlaceholder ∗ > GetAllContentPlaceholders ()

  Find all content placeholders.
- int AddContentPlaceholder (ContentPlaceholder &contentPlaceholder)

  Insert the content placeholder into the database.
- void UpdateContentPlaceholder (ContentPlaceholder &contentPlaceholder)

  Update the relevant content placeholder in the database.
- void UpdateContentPlaceholder (int id, int ∗sheetId, int ∗wid, bool ∗column, int ∗order)

  Update the relevant content placeholder in the database, NULLs passed if that parameter is not to be updated.
- void RemoveContentPlaceholder (int id)

  Delete the content placeholder from the database.
- ContentPlaceholder ∗ GenerateContentPlaceholder (sql::ResultSet &rs)

  Generate a content placeholder based on the data in a result set row.

Private Attributes

- DatabaseController $*$ dbc

  Database controller used to connect to the database.
- sql::Connection $*$ conn

  Connection to the database.

2.4.1 Detailed Description

A class called to create, update, delete or find content placeholders.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 ContentPlaceholderController::ContentPlaceholderController ( )

Initialising Constructor for ContentPlaceholderController, opens a database connection.

2.4.2.2 ContentPlaceholderController::$\sim$ContentPlaceholderController ( )

Deallocates any dynamic memory and closes the database connection.

2.4.3 Member Function Documentation

2.4.3.1 int ContentPlaceholderController::AddContentPlaceholder (ContentPlaceholder & *contentPlaceholder*)

Insert the content placeholder into the database.

Parameters

content	placeholder content placeholder to be added to the database

Returns

id of the added content placeholder, -1 for error

2.4.3.2 ContentPlaceholder $*$ ContentPlaceholderController::GenerateContentPlaceholder (sql::ResultSet & *rs*)

Generate a content placeholder based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a content placeholder

Returns

the generated content placeholder

2.4.3.3 vector$<$ ContentPlaceholder $*$ $>$ ContentPlaceholderController::GetAllContentPlaceholders ( )

Find all content placeholders.

Returns

all content placeholders in the database

2.4.3.4 ContentPlaceholder $*$ ContentPlaceholderController::GetContentPlaceholderById (int *id*)

Find the content placeholder with that id.

Parameters

id	primary key id of the content placeholder

Returns

the content placeholder with that id

2.4.3.5 vector$<$ ContentPlaceholder $*$ $>$ ContentPlaceholderController::GetContentPlaceholdersBySheetId (int *sheetId*)

Find all content placeholders that belong to the sheet of that id.

Parameters

sheetId	id of the sheet to find content placeholders for

Returns

the content placeholders that belong to that sheet

2.4.3.6 void ContentPlaceholderController::RemoveContentPlaceholder (int *id*)

Delete the content placeholder from the database.

Parameters

content-Placeholder	content placeholder to be removed from the database

2.4.3.7 void ContentPlaceholderController::UpdateContentPlaceholder (ContentPlaceholder & *contentPlaceholder*)

Update the relevant content placeholder in the database.

Parameters

content-Placeholder	content placeholder to be updated

2.4.3.8 void ContentPlaceholderController::UpdateContentPlaceholder (int *id,* int $*$ *sheetId,* int $*$ *wid,* bool $*$ *column,* int $*$ *order*)

Update the relevant content placeholder in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the contentplaceholder
sheetId	int representing the id of the sheet this placeholder is located on, nullable
type	int representing what type of content placeholder it is, nullable
column	int representing which column the placeholder is located in, nullable
order	int representing what order this placeholder is in the column, nullable

2.4.4 Member Data Documentation

2.4.4.1 sql::Connection∗ nfdb::ContentPlaceholderController::conn `[private]`

Connection to the database.

2.4.4.2 DatabaseController∗ nfdb::ContentPlaceholderController::dbc `[private]`

Database controller used to connect to the database.

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

- include/ContentPlaceholderController.h
- src/ContentPlaceholderController.cpp

2.5 nfdb::DatabaseController Class Reference

A class called to connect and disconnect from the database.

`#include <DatabaseController.h>`

Public Member Functions

- DatabaseController ()

  Initialising Constructor for DatabaseController.
- ∼DatabaseController ()

  Deallocates any dynamic memory.
- sql::Connection ∗ Connect ()

  Connects to the database and returns the active connection.
- void Disconnect ()

  Disconnects from the database.

Private Attributes

- sql::Connection ∗ conn

  Active connection to the database.

2.5.1 Detailed Description

A class called to connect and disconnect from the database.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 DatabaseController::DatabaseController ( )

Initialising Constructor for DatabaseController.

2.5.2.2 DatabaseController::∼DatabaseController ( )

Deallocates any dynamic memory.

2.5.3 Member Function Documentation

2.5.3.1 Connection ∗ DatabaseController::Connect ( )

Connects to the database and returns the active connection.

Returns

the connection to the database

2.5.3.2 void DatabaseController::Disconnect ( )

Disconnects from the database.

2.5.4 Member Data Documentation

2.5.4.1 sql::Connection∗ nfdb::DatabaseController::conn `[private]`

Active connection to the database.

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

- include/DatabaseController.h
- src/DatabaseController.cpp

2.6 nfdb::Feed Class Reference

A class representing a row in the Feeds table.

`#include <Feed.h>`

Public Member Functions

- Feed ()

  Initialising Constructor for Feed.
- Feed (int id, std::string url, std::string name, int frequency, nfrd::misc::DateTime ∗lastUpdate, std::string ∗category, int ∗type, char ∗favIcon, int iconSize)

  Non-Default Constructor for Feed.
- ∼Feed ()

  Destructor for Feed.
- void Destroy ()

  Cleans up any memory held by the Feed.

Public Attributes

- int id
- std::string url
- std::string name
- int frequency
- nfrd::misc::DateTime ∗ lastUpdate
- std::string ∗ category
- int ∗ type
- char ∗ favIcon
- int iconSize

2.6.1 Detailed Description

A class representing a row in the Feeds table.

2.6.2 Constructor & Destructor Documentation

2.6.2.1 nfdb::Feed::Feed ( ) [inline]

Initialising Constructor for Feed.

Sets lastUpdate, category, type and favIcon to NULL

2.6.2.2 nfdb::Feed::Feed (int *id,* std::string *url,* std::string *name,* int *frequency,* nfrd::misc::DateTime ∗ *lastUpdate,* std::string ∗ *category,* int ∗ *type,* char ∗ *favIcon,* int *iconSize*) [inline]

Non-Default Constructor for Feed.

Parameters

id	int identifier of the feed
url	string representing the url of the RSS feed
name	string representing the display name of the feed
frequency	int representing how often this feed is updated
lastUpdate	datetime representing when this feed was last crawled/updated, nullable
category	string∗ representing what category this feed is in, nullable
type	int∗ representing the type of feed, nullable
favIcon	char∗ representing the favourite icon image of that feed, nullable
iconSize	int representing the size of the feed's favourite icon, 0 if no icon is provided

2.6.2.3 nfdb::Feed::∼Feed ( ) [inline]

Destructor for Feed.

Does not perform any actions

2.6.3 Member Function Documentation

2.6.3.1 void nfdb::Feed::Destroy ( ) [inline]

Cleans up any memory held by the Feed.

2.6.4 Member Data Documentation

2.6.4.1 std::string∗ nfdb::Feed::category

2.6.4.2 char∗ nfdb::Feed::favIcon

2.6.4.3 int nfdb::Feed::frequency

2.6.4.4 int nfdb::Feed::iconSize

2.6.4.5 int nfdb::Feed::id

2.6.4.6 nfrd::misc::DateTime∗ nfdb::Feed::lastUpdate

2.6.4.7 std::string nfdb::Feed::name

2.6.4.8 int∗ nfdb::Feed::type

2.6.4.9 std::string nfdb::Feed::url

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

- include/Feed.h

2.7 nfdb::FeedController Class Reference

A class called to create, update, delete or find feeds.

```
#include <FeedController.h>
```

Public Member Functions

- FeedController ()

  Initialising Constructor for FeedController, opens a database connection.
- ∼FeedController ()

  Deallocates any dynamic memory and closes the database connection.
- Feed ∗ GetFeedById (int id)

  Find the feed with that id.
- std::vector< Feed ∗ > GetFeedsByCphId (int cphId)

  Find all feeds that are contained within that CPH.
- std::vector< Feed ∗ > GetFeedsByUsername (std::string username)

  Find all feeds that belong to the user of that username.
- std::vector< Feed ∗ > GetAllFeeds ()

  Find all feeds.
- std::vector< QueueItem ∗ > GetQueueFeeds ()

  Find all feeds which are due to be crawled again.
- std::vector< Feed ∗ > GetNewFeeds (int id)

  Get all of the feeds which are newer than the passed id.
- int AddFeed (Feed &feed)

  Insert the feed into the database.
- void UpdateFeed (Feed &feed)

  Update the relevant feed in the database.

- void UpdateFeed (int id, std::string ∗url, std::string ∗name, int ∗frequency, nfrd::misc::DateTime ∗lastUpdate, std::string ∗category, int ∗type, char ∗favIcon, int ∗iconSize)

  Update the relevant feed in the database, NULLs passed if that parameter is not to be updated.
- void RemoveFeed (int id)

  Delete the feed from the database.
- Feed ∗ GenerateFeed (sql::ResultSet &rs)

  Generate a feed based on the data in a result set row.
- QueueItem ∗ GenerateQueueItem (sql::ResultSet &rs)

  Generate a queue item based on the data in a result set row.
- int GetNumberOfFeedUsers (int id)

  Get the number of users currently subscribed to that feed.
- void UpdateLastUpdateTime (int feedid)

  Touch the feed, setting lastupdate to now.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.7.1 Detailed Description

A class called to create, update, delete or find feeds.

2.7.2 Constructor & Destructor Documentation

2.7.2.1 FeedController::FeedController ( )

Initialising Constructor for FeedController, opens a database connection.

2.7.2.2 FeedController::∼FeedController ( )

Deallocates any dynamic memory and closes the database connection.

2.7.3 Member Function Documentation

2.7.3.1 int FeedController::AddFeed (Feed & *feed*)

Insert the feed into the database.

Parameters

feed	feed to be added to the database

Returns

id of the added feed, -1 for an error

2.7.3.2 Feed $*$ FeedController::GenerateFeed (sql::ResultSet & *rs*)

Generate a feed based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a feed

Returns

the generated feed

2.7.3.3 QueueItem $*$ FeedController::GenerateQueueItem (sql::ResultSet & *rs*)

Generate a queue item based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a feed

Returns

the generated queue item

2.7.3.4 vector$<$ Feed $*$ $>$ FeedController::GetAllFeeds ( )

Find all feeds.

Returns

all feeds in the database

2.7.3.5 Feed $*$ FeedController::GetFeedById (int *id*)

Find the feed with that id.

Parameters

id	primary key id of the feed

Returns

the feed with that id

2.7.3.6 vector$<$ Feed $*$ $>$ FeedController::GetFeedsByCphId (int *cphId*)

Find all feeds that are contained within that CPH.

Parameters

cphId	id of the cph to find feeds for

Returns

the feeds that belong to that cph

2.7.3.7 vector< Feed * > FeedController::GetFeedsByUsername (std::string *username*)

Find all feeds that belong to the user of that username.

Parameters

username	username of the user to find feeds for

Returns

the feeds that belong to that user

2.7.3.8 vector< Feed * > FeedController::GetNewFeeds (int *id*)

Get all of the feeds which are newer than the passed id.

Parameters

id	the id of the last feed that has been added to the queue

Returns

feeds which are newer than the passed id

2.7.3.9 int FeedController::GetNumberOfFeedUsers (int *id*)

Get the number of users currently subscribed to that feed.

Parameters

id	int identifying the feed

Returns

the number of users who have that feed

2.7.3.10 vector< QueueItem * > FeedController::GetQueueFeeds ( )

Find all feeds which are due to be crawled again.

Returns

feeds which are due to be crawled again

2.7.3.11 void FeedController::RemoveFeed (int *id*)

Delete the feed from the database.

Parameters

feed	feed to be removed from the database

2.7.3.12 void FeedController::UpdateFeed (Feed & *feed*)

Update the relevant feed in the database.

Parameters

feed	feed to be updated

2.7.3.13 void nfdb::FeedController::UpdateFeed (int *id*, std::string ∗ *url*, std::string ∗ *name*, int ∗ *frequency*, nfrd::misc::DateTime ∗ *lastUpdate*, std::string ∗ *category*, int ∗ *type*, char ∗ *favIcon*, int ∗ *iconSize*)

Update the relevant feed in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the feed
url	string representing the url of the RSS feed, nullable
name	string representing the display name of the feed, nullable
frequency	int representing how often this feed is updated, nullable
lastUpdate	datetime representing when this feed was last crawled/updated, nullable
category	string∗ representing what category this feed is in, nullable
type	int∗ representing the type of feed, nullable
favIcon	char∗ representing the favourite icon image of that feed, nullable
iconSize	int representing the size of the feed's favourite icon, 0 if no icon is provided

2.7.3.14 void FeedController::UpdateLastUpdateTime (int *feedid*)

Touch the feed, setting lastupdate to now.

Parameters

feedid	int identifying the feed

2.7.4 Member Data Documentation

2.7.4.1 sql::Connection∗ nfdb::FeedController::conn `[private]`

Connection to the database.

2.7.4.2 DatabaseController∗ nfdb::FeedController::dbc `[private]`

Database controller used to connect to the database.

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

- include/FeedController.h
- src/FeedController.cpp

2.8 nfdb::GroupPlaceholder Class Reference

A class representing a row in the GroupPlaceholders table.

`#include <GroupPlaceholder.h>`

Public Member Functions

- GroupPlaceholder ()
  Initialising Constructor for GroupPlaceholder.
- GroupPlaceholder (int id)
  Non-Default Constructor for GroupPlaceholder.
- ∼GroupPlaceholder ()
  Destructor for GroupPlaceholder, does not perform any actions.
- void Destroy ()
  Cleans up any memory held by the GroupPlaceholder.

Public Attributes

- int id

2.8.1 Detailed Description

A class representing a row in the GroupPlaceholders table.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 nfdb::GroupPlaceholder::GroupPlaceholder ( ) `[inline]`

Initialising Constructor for GroupPlaceholder.

2.8.2.2 nfdb::GroupPlaceholder::GroupPlaceholder (int *id*) `[inline]`

Non-Default Constructor for GroupPlaceholder.

Parameters

id	int identifier of the group placeholder

2.8.2.3 nfdb::GroupPlaceholder::∼GroupPlaceholder ( ) `[inline]`

Destructor for GroupPlaceholder, does not perform any actions.

2.8.3 Member Function Documentation

2.8.3.1 void nfdb::GroupPlaceholder::Destroy ( ) `[inline]`

Cleans up any memory held by the GroupPlaceholder.

2.8.4 Member Data Documentation

2.8.4.1 int nfdb::GroupPlaceholder::id

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

- include/GroupPlaceholder.h

2.9 nfdb::GroupPlaceholderController Class Reference

A class called to create, update, delete or find group placeholders.

`#include <GroupPlaceholderController.h>`

Public Member Functions

- GroupPlaceholderController ()

  Initialising Constructor for GroupPlaceholderController, opens a database connection.
- ∼GroupPlaceholderController ()

  Deallocates any dynamic memory and closes the database connection.
- GroupPlaceholder GetGroupPlaceholderById (int id)

  Find the group placeholder with that id.
- std::vector< GroupPlaceholder > GetAllGroupPlaceholders ()

  Find all group placeholders.
- void AddGroupPlaceholder (GroupPlaceholder &groupPlaceholder)

  Insert the group placeholder into the database.
- void UpdateGroupPlaceholder (GroupPlaceholder &groupPlaceholder)

  Update the relevant group placeholder in the database.
- void RemoveGroupPlaceholder (GroupPlaceholder &groupPlaceholder)

  Delete the group placeholder from the database.
- GroupPlaceholder GenerateGroupPlaceholder (sql::ResultSet &rs)

  Generate a group placeholder based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.9.1 Detailed Description

A class called to create, update, delete or find group placeholders.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 GroupPlaceholderController::GroupPlaceholderController ( )

Initialising Constructor for GroupPlaceholderController, opens a database connection.

2.9.2.2 GroupPlaceholderController::∼GroupPlaceholderController ( )

Deallocates any dynamic memory and closes the database connection.

2.9.3 Member Function Documentation

2.9.3.1 void GroupPlaceholderController::AddGroupPlaceholder (**GroupPlaceholder** & *groupPlaceholder*)

Insert the group placeholder into the database.

Parameters

group-Placeholder	group placeholder to be added to the database

2.9.3.2 **GroupPlaceholder** GroupPlaceholderController::GenerateGroupPlaceholder (sql::ResultSet & *rs*)

Generate a group placeholder based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a group placeholder

Returns

the generated group placeholder

2.9.3.3 vector< **GroupPlaceholder** > GroupPlaceholderController::GetAllGroupPlaceholders ( )

Find all group placeholders.

Returns

all group placeholders in the database

2.9.3.4 **GroupPlaceholder** GroupPlaceholderController::GetGroupPlaceholderById (int *id*)

Find the group placeholder with that id.

Parameters

id	primary key id of the group placeholder

Returns

the group placeholder with that id

2.9.3.5 void GroupPlaceholderController::RemoveGroupPlaceholder (**GroupPlaceholder** & *groupPlaceholder*)

Delete the group placeholder from the database.

Parameters

group-Placeholder	group placeholder to be removed from the database

2.9.3.6 void GroupPlaceholderController::UpdateGroupPlaceholder (GroupPlaceholder & *groupPlaceholder*)

Update the relevant group placeholder in the database.

Parameters

group-Placeholder	group placeholder to be updated

2.9.4 Member Data Documentation

2.9.4.1 sql::Connection∗ nfdb::GroupPlaceholderController::conn `[private]`

Connection to the database.

2.9.4.2 DatabaseController∗ nfdb::GroupPlaceholderController::dbc `[private]`

Database controller used to connect to the database.

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

- include/GroupPlaceholderController.h
- src/GroupPlaceholderController.cpp

2.10 nfdb::Image Class Reference

A class representing a row in the Images table.

`#include <Image.h>`

Public Member Functions

- Image ()
  Initialising Constructor for Image, sets image to NULL.
- Image (int id, int itemId, char ∗image, int imageSize, std::string url)
  Non-Default Constructor for Image.
- ∼Image ()
  Destructor for Image, does not perform any actions.
- void Destroy ()
  Cleans up any memory held by the Image.

Public Attributes

- int id
- int itemId
- char ∗ image
- int imageSize
- std::string url

2.10.1 Detailed Description

A class representing a row in the Images table.

2.10.2 Constructor & Destructor Documentation

2.10.2.1 nfdb::Image::Image ( ) `[inline]`

Initialising Constructor for Image, sets image to NULL.

2.10.2.2 nfdb::Image::Image (int *id*, int *itemId*, char ∗ *image*, int *imageSize*, std::string *url*) `[inline]`

Non-Default Constructor for Image.

Parameters

id	int identifier of the image
itemId	int representing the id of the item this image relates to
image	char∗ representing the actual image array, nullable
imageSize	int representing the size of the image in bytes, 0 if no image supplied
url	string representing the url of the image

2.10.2.3 nfdb::Image::∼Image ( ) `[inline]`

Destructor for Image, does not perform any actions.

2.10.3 Member Function Documentation

2.10.3.1 void nfdb::Image::Destroy ( ) `[inline]`

Cleans up any memory held by the Image.

2.10.4 Member Data Documentation

2.10.4.1 int nfdb::Image::id

2.10.4.2 char∗ nfdb::Image::image

2.10.4.3 int nfdb::Image::imageSize

2.10.4.4 int nfdb::Image::itemId

2.10.4.5 std::string nfdb::Image::url

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

- include/Image.h

2.11 nfdb::ImageController Class Reference

A class called to create, update, delete or find images.

```
#include <ImageController.h>
```

Public Member Functions

- ImageController ()

  Initialising Constructor for ImageController, opens a database connection.
- ∼ImageController ()

  Deallocates any dynamic memory and closes the database connection.
- Image ∗ GetImageById (int id)

  Find the image with that id.
- std::vector< Image ∗ > GetImagesByItemId (int itemId)

  Find all images that belong to the item of that id.
- std::vector< Image ∗ > GetAllImages ()

  Find all images.
- int AddImage (Image &image)

  Insert the image into the database.
- void UpdateImage (Image &image)

  Update the relevant image in the database.
- void UpdateImage (int id, int ∗itemId, char ∗image, int ∗imageSize, std::string ∗url)

  Update the relevant image in the database, NULLs passed if that parameter is not to be updated.
- void RemoveImage (int id)

  Delete the image from the database.
- Image ∗ GenerateImage (sql::ResultSet &rs)

  Generate a image based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.11.1 Detailed Description

A class called to create, update, delete or find images.

2.11.2 Constructor & Destructor Documentation

2.11.2.1 ImageController::ImageController ( )

Initialising Constructor for ImageController, opens a database connection.

2.11.2.2 ImageController::∼ImageController ( )

Deallocates any dynamic memory and closes the database connection.

2.11.3 Member Function Documentation

2.11.3.1 int ImageController::AddImage (Image & *image*)

Insert the image into the database.

Parameters

image	image to be added to the database

Returns

id of the newly added image, -1 for an error

2.11.3.2 Image $*$ ImageController::GenerateImage (sql::ResultSet & *rs*)

Generate a image based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a image

Returns

the generated image

2.11.3.3 vector$<$ Image $*$ $>$ ImageController::GetAllImages ( )

Find all images.

Returns

all images in the database

2.11.3.4 Image $*$ ImageController::GetImageById (int *id*)

Find the image with that id.

Parameters

id	primary key id of the image

Returns

the image with that id

2.11.3.5 vector$<$ Image $*$ $>$ ImageController::GetImagesByItemId (int *itemId*)

Find all images that belong to the item of that id.

Parameters

itemId	id of the item to find images for

Returns

the images that belong to that item

2.11.3.6 void ImageController::RemoveImage (int *id*)

Delete the image from the database.

Parameters

image	image to be removed from the database

2.11.3.7 void ImageController::UpdateImage (**Image** & *image*)

Update the relevant image in the database.

Parameters

image	image to be updated

2.11.3.8 void nfdb::ImageController::UpdateImage (int *id,* int ∗ *itemId,* char ∗ *image,* int ∗ *imageSize,* std::string ∗ *url*)

Update the relevant image in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the image
itemId	int representing the id of the item this image relates to, nullable
image	char∗ representing the actual image array, nullable
imageSize	int representing the size of the image in bytes, 0 if no image supplied
url	string representing the url of the image, nullable

2.11.4 Member Data Documentation

2.11.4.1 sql::Connection∗ nfdb::ImageController::conn `[private]`

Connection to the database.

2.11.4.2 **DatabaseController**∗ nfdb::ImageController::dbc `[private]`

Database controller used to connect to the database.

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

- include/ImageController.h
- src/ImageController.cpp

2.12 nfdb::Item Class Reference

A class representing a row in the Items table.

`#include <Item.h>`

Public Member Functions

- Item ()
  Initialising Constructor for Item, sets postDate, author and geolocation to NULL.
- Item (int id, int feedId, std::string title, std::string url, std::string content, nfrd::misc::DateTime ∗postDate, std::string ∗author, std::string ∗geolocation)
  Non-Default Constructor for Item.
- ∼Item ()
  Destructor for Item, does not perform any actions.
- void Destroy ()
  Cleans up any memory held by the Item.

Public Attributes

- int id
- int feedId
- std::string title
- std::string url
- std::string content
- nfrd::misc::DateTime ∗ postDate
- std::string ∗ author
- std::string ∗ geolocation

2.12.1 Detailed Description

A class representing a row in the Items table.

2.12.2 Constructor & Destructor Documentation

2.12.2.1 nfdb::Item::Item ( ) [inline]

Initialising Constructor for Item, sets postDate, author and geolocation to NULL.

2.12.2.2 nfdb::Item::Item (int *id,* int *feedId,* std::string *title,* std::string *url,* std::string *content,* nfrd::misc::DateTime ∗ *postDate,* std::string ∗ *author,* std::string ∗ *geolocation*) [inline]

Non-Default Constructor for Item.

Parameters

id	int identifier of the item
feedId	int representing the id of the feed this item relates to
title	string representing the display name of the item
url	string representing the url of the item
content	string representing the body content of the item
postDate	datetime representing the date the item was published, nullable
author	string∗ representing the author of the item, nullable
geolocation	string∗ representing the geolocation information parsed for that item, nullable

2.12.2.3 nfdb::Item::∼Item ( ) [inline]

Destructor for Item, does not perform any actions.

2.12.3 Member Function Documentation

2.12.3.1 void nfdb::Item::Destroy ( ) `[inline]`

Cleans up any memory held by the Item.

2.12.4 Member Data Documentation

2.12.4.1 std::string∗ nfdb::Item::author

2.12.4.2 std::string nfdb::Item::content

2.12.4.3 int nfdb::Item::feedId

2.12.4.4 std::string∗ nfdb::Item::geolocation

2.12.4.5 int nfdb::Item::id

2.12.4.6 nfrd::misc::DateTime∗ nfdb::Item::postDate

2.12.4.7 std::string nfdb::Item::title

2.12.4.8 std::string nfdb::Item::url

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

- include/Item.h

2.13 nfdb::ItemController Class Reference

A class called to create, update, delete or find items.

`#include <ItemController.h>`

Public Member Functions

- ItemController ()

  Initialising Constructor for ItemController, opens a database connection.
- ∼ItemController ()

  Deallocates any dynamic memory and closes the database connection.
- Item ∗ GetItemById (int id)

  Find the item with that id.
- std::vector< Item ∗ > GetItemsByFeedId (int feedId)

  Find all items that belong to the feed of that id.
- std::vector< Item ∗ > GetAllItems ()

  Find all items.
- int AddItem (Item &item)

  Insert the item into the database.
- void UpdateItem (Item &item)

  Update the relevant item in the database.
- void UpdateItem (int id, int ∗feedId, std::string ∗title, std::string ∗url, std::string ∗content, nfrd::misc::DateTime ∗postDate, std::string ∗author, std::string ∗geolocation)

Update the relevant item in the database, NULLs passed if that parameter is not to be updated.

- void RemoveItem (int id)

  Delete the item from the database.

- Item ∗ GenerateItem (sql::ResultSet &rs)

  Generate a item based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.

- sql::Connection ∗ conn

  Connection to the database.

2.13.1 Detailed Description

A class called to create, update, delete or find items.

2.13.2 Constructor & Destructor Documentation

2.13.2.1 ItemController::ItemController ( )

Initialising Constructor for ItemController, opens a database connection.

2.13.2.2 ItemController::∼ItemController ( )

Deallocates any dynamic memory and closes the database connection.

2.13.3 Member Function Documentation

2.13.3.1 int ItemController::AddItem (Item & *item*)

Insert the item into the database.

Parameters

item	item to be added to the database

Returns

id of the newly added item, -1 for an error

2.13.3.2 Item ∗ ItemController::GenerateItem (sql::ResultSet & *rs*)

Generate a item based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a item

Returns

the generated item

2.13.3.3 vector< Item ∗ > ItemController::GetAllItems ( )

Find all items.

Returns

all items in the database

2.13.3.4 Item ∗ ItemController::GetItemById (int *id*)

Find the item with that id.

Parameters

id	primary key id of the item

Returns

the item with that id

2.13.3.5 vector< Item ∗ > ItemController::GetItemsByFeedId (int *feedId*)

Find all items that belong to the feed of that id.

Parameters

feedId	id of the feed to find items for

Returns

the items that belong to that feed

2.13.3.6 void ItemController::RemoveItem (int *id*)

Delete the item from the database.

Parameters

item	item to be removed from the database

2.13.3.7 void ItemController::UpdateItem (Item & *item*)

Update the relevant item in the database.

Parameters

item	item to be updated

2.13.3.8 void nfdb::ItemController::UpdateItem (int *id,* int ∗ *feedId,* std::string ∗ *title,* std::string ∗ *url,* std::string ∗ *content,* nfrd::misc::DateTime ∗ *postDate,* std::string ∗ *author,* std::string ∗ *geolocation*)

Update the relevant item in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the item
feedId	int representing the id of the feed this item relates to, nullable
title	string representing the display name of the item, nullable
url	string representing the url of the item, nullable
content	string representing the body content of the item, nullable
postDate	datetime representing the date the item was published, nullable
author	string∗ representing the author of the item, nullable
geolocation	string∗ representing the geolocation information parsed for that item, nullable

2.13.4 Member Data Documentation

2.13.4.1 sql::Connection∗ nfdb::ItemController::conn `[private]`

Connection to the database.

2.13.4.2 DatabaseController∗ nfdb::ItemController::dbc `[private]`

Database controller used to connect to the database.

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

- include/ItemController.h
- src/ItemController.cpp

2.14 nfdb::Layout Class Reference

A class representing a row in the Layout table.

`#include <Layout.h>`

Public Member Functions

- Layout ()

  Initialising Constructor for Layout.
- Layout (int id)

  Non-Default Constructor for Layout.
- ∼Layout ()

  Destructor for Layout, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the Layout.

Public Attributes

- int id

2.14.1 Detailed Description

A class representing a row in the Layout table.

2.14.2 Constructor & Destructor Documentation

2.14.2.1 nfdb::Layout::Layout ( ) `[inline]`

Initialising Constructor for Layout.

2.14.2.2 nfdb::Layout::Layout (int *id*) `[inline]`

Non-Default Constructor for Layout.

Parameters

id	int identifier of the layout

2.14.2.3 nfdb::Layout::∼Layout ( ) `[inline]`

Destructor for Layout, does not perform any actions.

2.14.3 Member Function Documentation

2.14.3.1 void nfdb::Layout::Destroy ( ) `[inline]`

Cleans up any memory held by the Layout.

2.14.4 Member Data Documentation

2.14.4.1 int nfdb::Layout::id

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

- include/Layout.h

2.15 nfdb::LayoutController Class Reference

A class called to create, update, delete or find layouts.

```
#include <LayoutController.h>
```

Public Member Functions

- LayoutController ()

  Initialising Constructor for LayoutController, opens a database connection.
- ∼LayoutController ()

  Deallocates any dynamic memory and closes the database connection.
- Layout GetLayoutById (int id)

  Find the layout with that id.

- std::vector< Layout > GetAllLayouts ()

  Find all layouts.
- void AddLayout (Layout &layout)

  Insert the layout into the database.
- void UpdateLayout (Layout &layout)

  Update the relevant layout in the database.
- void RemoveLayout (Layout &layout)

  Delete the layout from the database.
- Layout GenerateLayout (sql::ResultSet &rs)

  Generate a layout based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.15.1 Detailed Description

A class called to create, update, delete or find layouts.

2.15.2 Constructor & Destructor Documentation

2.15.2.1 LayoutController::LayoutController ( )

Initialising Constructor for LayoutController, opens a database connection.

2.15.2.2 LayoutController::∼LayoutController ( )

Deallocates any dynamic memory and closes the database connection.

2.15.3 Member Function Documentation

2.15.3.1 void LayoutController::AddLayout (Layout & *layout*)

Insert the layout into the database.

Parameters

layout	layout to be added to the database

2.15.3.2 Layout LayoutController::GenerateLayout (sql::ResultSet & *rs*)

Generate a layout based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a layout

Returns

the generated layout

2.15.3.3 vector< Layout > LayoutController::GetAllLayouts ( )

Find all layouts.

Returns

all layouts in the database

2.15.3.4 Layout LayoutController::GetLayoutById (int *id*)

Find the layout with that id.

Parameters

id	primary key id of the layout

Returns

the layout with that id

2.15.3.5 void LayoutController::RemoveLayout (Layout & *layout*)

Delete the layout from the database.

Parameters

layout	layout to be removed from the database

2.15.3.6 void LayoutController::UpdateLayout (Layout & *layout*)

Update the relevant layout in the database.

Parameters

layout	layout to be updated

2.15.4 Member Data Documentation

2.15.4.1 sql::Connection∗ nfdb::LayoutController::conn `[private]`

Connection to the database.

2.15.4.2 DatabaseController∗ nfdb::LayoutController::dbc `[private]`

Database controller used to connect to the database.

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

- include/LayoutController.h

- src/LayoutController.cpp

2.16 nfdb::Notification Class Reference

A class representing a row in the Notification table.

```
#include <Notification.h>
```

Public Member Functions

- Notification ()

  Initialising Constructor for Notification.
- Notification (int id, std::string username, int sheetId)

  Non-Default Constructor for Notification.
- ∼Notification ()

  Destructor for Notification, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the Notification.

Public Attributes

- int id
- std::string username
- int sheetId

2.16.1 Detailed Description

A class representing a row in the Notification table.

2.16.2 Constructor & Destructor Documentation

2.16.2.1 nfdb::Notification::Notification ( ) `[inline]`

Initialising Constructor for Notification.

2.16.2.2 nfdb::Notification::Notification (int *id,* std::string *username,* int *sheetId*) `[inline]`

Non-Default Constructor for Notification.

Parameters

id	int identifier of the notification
username	string representing the username of the user this notification relates to
sheetId	int representing the id of the sheet this notification relates to

2.16.2.3 nfdb::Notification::∼Notification ( ) `[inline]`

Destructor for Notification, does not perform any actions.

2.16.3 Member Function Documentation

2.16.3.1 void nfdb::Notification::Destroy ( ) `[inline]`

Cleans up any memory held by the Notification.

2.16.4 Member Data Documentation

2.16.4.1 int nfdb::Notification::id

2.16.4.2 int nfdb::Notification::sheetId

2.16.4.3 std::string nfdb::Notification::username

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

- include/Notification.h

2.17 nfdb::NotificationController Class Reference

A class called to create, update, delete or find notifications.

`#include <NotificationController.h>`

Public Member Functions

- NotificationController ()

  Initialising Constructor for NotificationController, opens a database connection.
- ∼NotificationController ()

  Deallocates any dynamic memory and closes the database connection.
- Notification ∗ GetNotificationById (int id)

  Find the notification with that id.
- std::vector< Notification ∗ > GetNotificationsBySheetId (int sheetId)

  Find all notifications that belong to the sheet of that id.
- std::vector< Notification ∗ > GetNotificationsByUsername (std::string username)

  Find all notifications that belong to the user of that username.
- std::vector< Notification ∗ > GetAllNotifications ()

  Find all notifications.
- int AddNotification (Notification &notification)

  Insert the notification into the database.
- void UpdateNotification (Notification &notification)

  Update the relevant notification in the database.
- void UpdateNotification (int id, std::string ∗username, int ∗sheetId)

  Update the relevant notification in the database, NULLs passed if that parameter is not to be updated.
- void RemoveNotification (int id)

  Delete the notification from the database.
- void AddFeedUpdateNotifications (int feedid)

  Insert notifications based on the feed updated.
- Notification ∗ GenerateNotification (sql::ResultSet &rs)

  Generate a notification based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.17.1 Detailed Description

A class called to create, update, delete or find notifications.

2.17.2 Constructor & Destructor Documentation

2.17.2.1 NotificationController::NotificationController ( )

Initialising Constructor for NotificationController, opens a database connection.

2.17.2.2 NotificationController::∼NotificationController ( )

Deallocates any dynamic memory and closes the database connection.

2.17.3 Member Function Documentation

2.17.3.1 void NotificationController::AddFeedUpdateNotifications (int *feedid*)

Insert notifications based on the feed updated.

Parameters

feedid	id of the feed that was updated

2.17.3.2 int NotificationController::AddNotification (Notification & *notification*)

Insert the notification into the database.

Parameters

notification	notification to be added to the database

Returns

id of the newly added notification, -1 for an error

2.17.3.3 Notification ∗ NotificationController::GenerateNotification (sql::ResultSet & *rs*)

Generate a notification based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a notification

Returns

the generated notification

2.17.3.4 vector< Notification ∗ > NotificationController::GetAllNotifications ( )

Find all notifications.

Returns

all notifications in the database

2.17.3.5 Notification ∗ NotificationController::GetNotificationById (int *id*)

Find the notification with that id.

Parameters

id	primary key id of the notification

Returns

the notification with that id

2.17.3.6 vector< Notification ∗ > NotificationController::GetNotificationsBySheetId (int *sheetId*)

Find all notifications that belong to the sheet of that id.

Parameters

sheetId	id of the sheet to find notifications for

Returns

the notifications that belong to that sheet

2.17.3.7 vector< Notification ∗ > NotificationController::GetNotificationsByUsername (std::string *username*)

Find all notifications that belong to the user of that username.

Parameters

username	username of the user to find notifications for

Returns

the notifications that belong to that user

2.17.3.8 void NotificationController::RemoveNotification (int *id*)

Delete the notification from the database.

Parameters

id	id of the notification to be removed from the database

2.17.3.9 void NotificationController::UpdateNotification (Notification & *notification*)

Update the relevant notification in the database.

Parameters

notification	notification to be updated

2.17.3.10 void nfdb::NotificationController::UpdateNotification (int *id,* std::string ∗ *username,* int ∗ *sheetId*)

Update the relevant notification in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the notification
username	string representing the username of the user this notification relates to, nullable
sheetId	int representing the id of the sheet this notification relates to, nullable

2.17.4 Member Data Documentation

2.17.4.1 sql::Connection∗ nfdb::NotificationController::conn `[private]`

Connection to the database.

2.17.4.2 DatabaseController∗ nfdb::NotificationController::dbc `[private]`

Database controller used to connect to the database.

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

- include/NotificationController.h
- src/NotificationController.cpp

2.18 nfdb::QueueItem Class Reference

A class representing a QueueItem for the queue, derived from the Item table.

`#include <QueueItem.h>`

Public Member Functions

- QueueItem ()

  Initialising Constructor for QueueItem, sets type to NULL.
- QueueItem (int id, int frequency, int ∗type, int numUsers)

  Non-Default Constructor for QueueItem.
- ∼QueueItem ()

  Destructor for QueueItem, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the QueueItem.

Public Attributes

- int id
- int frequency
- int ∗ type
- int numUsers

2.18.1 Detailed Description

A class representing a QueueItem for the queue, derived from the Item table.

2.18.2 Constructor & Destructor Documentation

2.18.2.1 nfdb::QueueItem::QueueItem ( ) [inline]

Initialising Constructor for QueueItem, sets type to NULL.

2.18.2.2 nfdb::QueueItem::QueueItem (int *id*, int *frequency*, int ∗ *type*, int *numUsers*) [inline]

Non-Default Constructor for QueueItem.

Parameters

id	int identifier of the queue item, gotten from item id
frequency	int representing the frequency of updates on this item
type	int∗ representing the type of item, nullable
numUsers	int representing the number of users who are using that item

2.18.2.3 nfdb::QueueItem::∼QueueItem ( ) [inline]

Destructor for QueueItem, does not perform any actions.

2.18.3 Member Function Documentation

2.18.3.1 void nfdb::QueueItem::Destroy ( ) [inline]

Cleans up any memory held by the QueueItem.

2.18.4 Member Data Documentation

2.18.4.1 int nfdb::QueueItem::frequency

2.18.4.2 int nfdb::QueueItem::id

2.18.4.3 int nfdb::QueueItem::numUsers

2.18.4.4 int∗ nfdb::QueueItem::type

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

- include/QueueItem.h

2.19 nfdb::Session Class Reference

A class representing a row in the Session table.

```
#include <Session.h>
```

Public Member Functions

- Session ()

  Initialising Constructor for Session.
- Session (int id, std::string username, std::string key, nfrd::misc::DateTime time)

  Non-Default Constructor for Session.
- ∼Session ()

  Destructor for Session, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the Session.

Public Attributes

- int id
- std::string username
- std::string key
- nfrd::misc::DateTime time

2.19.1 Detailed Description

A class representing a row in the Session table.

2.19.2 Constructor & Destructor Documentation

2.19.2.1 nfdb::Session::Session ( ) `[inline]`

Initialising Constructor for Session.

2.19.2.2 nfdb::Session::Session (int *id*, std::string *username*, std::string *key*, nfrd::misc::DateTime *time*) `[inline]`

Non-Default Constructor for Session.

Parameters

id	int identifier of the session
username	string representing the username of the user this session relates to
key	string representing the unique key of the session
time	datetime representing the time the session was created

2.19.2.3 nfdb::Session::∼Session ( ) `[inline]`

Destructor for Session, does not perform any actions.

2.19.3 Member Function Documentation

2.19.3.1 void nfdb::Session::Destroy ( ) [inline]

Cleans up any memory held by the Session.

2.19.4 Member Data Documentation

2.19.4.1 int nfdb::Session::id

2.19.4.2 std::string nfdb::Session::key

2.19.4.3 nfrd::misc::DateTime nfdb::Session::time

2.19.4.4 std::string nfdb::Session::username

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

- include/Session.h

2.20 nfdb::SessionController Class Reference

A class called to create, update, delete or find sessions.

`#include <SessionController.h>`

Public Member Functions

- SessionController ()

  Initialising Constructor for SessionController, opens a database connection.
- ∼SessionController ()

  Deallocates any dynamic memory and closes the database connection.
- Session ∗ GetSessionById (int id)

  Find the session with that id.
- std::vector< Session ∗ > GetSessionsByUsername (std::string username)

  Find all sessions that belong to the item of that id.
- Session ∗ GetSessionByKey (std::string key)

  Find the sessions with that key.
- std::vector< Session ∗ > GetAllSessions ()

  Find all sessions.
- int AddSession (Session &session)

  Insert the session into the database.
- void UpdateSession (Session &session)

  Update the relevant session in the database.
- void UpdateSession (int id, std::string ∗username, std::string ∗key, nfrd::misc::DateTime ∗time)

  Update the relevant session in the database, NULLs passed if that parameter is not to be updated.
- void RemoveSession (int id)

  Delete the session from the database.
- Session ∗ GenerateSession (sql::ResultSet &rs)

  Generate a session based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.20.1 Detailed Description

A class called to create, update, delete or find sessions.

2.20.2 Constructor & Destructor Documentation

2.20.2.1 SessionController::SessionController ( )

Initialising Constructor for SessionController, opens a database connection.

2.20.2.2 SessionController::∼SessionController ( )

Deallocates any dynamic memory and closes the database connection.

2.20.3 Member Function Documentation

2.20.3.1 int SessionController::AddSession (Session & *session*)

Insert the session into the database.

Parameters

session	session to be added to the database

Returns

id of the newly added session, -1 for an error

2.20.3.2 Session ∗ SessionController::GenerateSession (sql::ResultSet & *rs*)

Generate a session based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a session

Returns

the generated session

2.20.3.3 vector< Session ∗ > SessionController::GetAllSessions ( )

Find all sessions.

Returns

all sessions in the database

2.20.3.4 Session ∗ SessionController::GetSessionById (int *id*)

Find the session with that id.

Parameters

id	primary key id of the session

Returns

the session with that id

2.20.3.5 Session ∗ SessionController::GetSessionByKey (std::string *key*)

Find the sessions with that key.

Parameters

key	key of the session

Returns

the session with that key

2.20.3.6 vector< Session ∗ > SessionController::GetSessionsByUsername (std::string *username*)

Find all sessions that belong to the item of that id.

Parameters

username	username of the user to find sessions for

Returns

the sessions that belong to that user

2.20.3.7 void SessionController::RemoveSession (int *id*)

Delete the session from the database.

Parameters

session	session to be removed from the database

2.20.3.8 void SessionController::UpdateSession (Session & *session*)

Update the relevant session in the database.

Parameters

session	session to be updated

2.20.3.9 void nfdb::SessionController::UpdateSession (int *id*, std::string ∗ *username*, std::string ∗ *key*, nfrd::misc::DateTime ∗ *time*)

Update the relevant session in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the session
username	string representing the username of the user this session relates to, nullable
key	string representing the unique key of the session, nullable
time	datetime representing the time the session was created, nullable

2.20.4 Member Data Documentation

2.20.4.1 sql::Connection∗ nfdb::SessionController::conn `[private]`

Connection to the database.

2.20.4.2 DatabaseController∗ nfdb::SessionController::dbc `[private]`

Database controller used to connect to the database.

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

- include/SessionController.h
- src/SessionController.cpp

2.21 nfdb::Sheet Class Reference

A class representing a row in the Sheet table.

`#include <Sheet.h>`

Public Member Functions

- Sheet ()

  Initialising Constructor for Sheet.
- Sheet (int id, std::string name, std::string username, nfrd::misc::DateTime updated, int layoutId)

  Non-Default Constructor for Sheet.
- ∼Sheet ()

  Destructor for Sheet, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the Sheet.

Public Attributes

- int id
- std::string name

- std::string username
- nfrd::misc::DateTime updated
- int layoutId

2.21.1 Detailed Description

A class representing a row in the Sheet table.

2.21.2 Constructor & Destructor Documentation

2.21.2.1 nfdb::Sheet::Sheet ( ) [inline]

Initialising Constructor for Sheet.

2.21.2.2 nfdb::Sheet::Sheet (int *id*, std::string *name*, std::string *username*, nfrd::misc::DateTime *updated*, int *layoutId*) [inline]

Non-Default Constructor for Sheet.

Parameters

id	int identifier of the sheet
name	string representing the display name of the sheet
username	string representing the username of the user this sheet relates to
updated	datetime representing the time that sheet was last updated
layoutid	int representing the identifier of the layout the sheet uses

2.21.2.3 nfdb::Sheet::∼Sheet ( ) [inline]

Destructor for Sheet, does not perform any actions.

2.21.3 Member Function Documentation

2.21.3.1 void nfdb::Sheet::Destroy ( ) [inline]

Cleans up any memory held by the Sheet.

2.21.4 Member Data Documentation

2.21.4.1 int nfdb::Sheet::id

2.21.4.2 int nfdb::Sheet::layoutId

2.21.4.3 std::string nfdb::Sheet::name

2.21.4.4 nfrd::misc::DateTime nfdb::Sheet::updated

2.21.4.5 std::string nfdb::Sheet::username

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

- include/Sheet.h

2.22 nfdb::SheetController Class Reference

A class called to create, update, delete or find sheets.

`#include <SheetController.h>`

Public Member Functions

- SheetController ()

  Initialising Constructor for SheetController, opens a database connection.
- ∼SheetController ()

  Deallocates any dynamic memory and closes the database connection.
- Sheet ∗ GetSheetById (int id)

  Find the sheet with that id.
- std::vector< Sheet ∗ > GetSheetsByUsername (std::string username)

  Find all sheets that belong to that user.
- std::vector< Sheet ∗ > GetAllSheets ()

  Find all sheets.
- int AddSheet (Sheet &sheet)

  Insert the sheet into the database.
- void UpdateSheet (Sheet &sheet)

  Update the relevant sheet in the database.
- void UpdateSheet (int id, std::string ∗name, std::string ∗username, nfrd::misc::DateTime ∗updated, int ∗layoutId)

  Update the relevant sheet in the database, NULLs passed if that parameter is not to be updated.
- void RemoveSheet (int id)

  Delete the sheet from the database.
- Sheet ∗ GenerateSheet (sql::ResultSet &rs)

  Generate a sheet based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.22.1 Detailed Description

A class called to create, update, delete or find sheets.

2.22.2 Constructor & Destructor Documentation

2.22.2.1 SheetController::SheetController ( )

Initialising Constructor for SheetController, opens a database connection.

2.22.2.2 SheetController::∼SheetController ( )

Deallocates any dynamic memory and closes the database connection.

2.22.3 Member Function Documentation

2.22.3.1 int SheetController::AddSheet (Sheet & *sheet*)

Insert the sheet into the database.

Parameters

sheet	sheet to be added to the database

Returns

id of the newly added sheet, -1 for an error

2.22.3.2 Sheet ∗ SheetController::GenerateSheet (sql::ResultSet & *rs*)

Generate a sheet based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a sheet

Returns

the generated sheet

2.22.3.3 vector< Sheet ∗ > SheetController::GetAllSheets ( )

Find all sheets.

Returns

all sheets in the database

2.22.3.4 Sheet ∗ SheetController::GetSheetById (int *id*)

Find the sheet with that id.

Parameters

id	primary key id of the sheet

Returns

the sheet with that id

2.22.3.5 vector< Sheet ∗ > SheetController::GetSheetsByUsername (std::string *username*)

Find all sheets that belong to that user.

Parameters

username	username of the user to find sheets for

Returns

the sheets that belong to that user

2.22.3.6 void SheetController::RemoveSheet (int *id*)

Delete the sheet from the database.

Parameters

sheet	sheet to be removed from the database

2.22.3.7 void SheetController::UpdateSheet (Sheet & *sheet*)

Update the relevant sheet in the database.

Parameters

sheet	sheet to be updated

2.22.3.8 void nfdb::SheetController::UpdateSheet (int *id,* std::string ∗ *name,* std::string ∗ *username,* nfrd::misc::DateTime ∗ *updated,* int ∗ *layoutId*)

Update the relevant sheet in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the sheet
name	string representing the display name of the sheet, nullable
username	string representing the username of the user this sheet relates to, nullable
updated	datetime representing the time that sheet was last updated, nullable
layoutid	int representing the identifier of the layout the sheet uses, nullable

2.22.4 Member Data Documentation

2.22.4.1 sql::Connection∗ nfdb::SheetController::conn `[private]`

Connection to the database.

2.22.4.2 DatabaseController∗ nfdb::SheetController::dbc `[private]`

Database controller used to connect to the database.

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

- include/SheetController.h
- src/SheetController.cpp

2.23 nfdb::Stat Class Reference

A class representing a row in the Stat table.

`#include <Stat.h>`

Public Member Functions

- Stat ()

  Initialising Constructor for Stat.
- Stat (int id, int users, int sheets, int feeds, int items, int comments)

  Non-Default Constructor for Stat.
- ∼Stat ()

  Destructor for Stat, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the Stat.

Public Attributes

- int id
- int users
- int sheets
- int feeds
- int items
- int comments

2.23.1 Detailed Description

A class representing a row in the Stat table.

2.23.2 Constructor & Destructor Documentation

2.23.2.1 nfdb::Stat::Stat ( ) [inline]

Initialising Constructor for Stat.

2.23.2.2 nfdb::Stat::Stat (int *id*, int *users*, int *sheets*, int *feeds*, int *items*, int *comments*) [inline]

Non-Default Constructor for Stat.

Parameters

id	int identifier of the stat
users	int representing the number of users in the database
sheets	int representing the number of sheets in the database
feeds	int representing the number of feeds in the database
items	int representing the number of items in the database
comments	int representing the number of comments in the database

2.23.2.3 nfdb::Stat::∼Stat ( ) [inline]

Destructor for Stat, does not perform any actions.

2.23.3 Member Function Documentation

2.23.3.1 void nfdb::Stat::Destroy ( ) [inline]

Cleans up any memory held by the Stat.

2.23.4 Member Data Documentation

2.23.4.1 int nfdb::Stat::comments

2.23.4.2 int nfdb::Stat::feeds

2.23.4.3 int nfdb::Stat::id

2.23.4.4 int nfdb::Stat::items

2.23.4.5 int nfdb::Stat::sheets

2.23.4.6 int nfdb::Stat::users

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

- include/Stat.h

2.24 nfdb::StatController Class Reference

A class called to create, update, delete or find stats.

`#include <StatController.h>`

Public Member Functions

- StatController ()

  Initialising Constructor for StatController, opens a database connection.
- ∼StatController ()

  Deallocates any dynamic memory and closes the database connection.
- Stat ∗ GetStatById (int id)

  Find the stat with that id.
- Stat ∗ GetLatestStat ()

  Find the latest statistic.
- std::vector< Stat ∗ > GetAllStats ()

  Find all stats.
- int AddStat (Stat &stat)

  Insert the stat into the database.
- void UpdateStat (Stat &stat)

  Update the relevant stat in the database.
- void UpdateStat (int id, int ∗users, int ∗sheets, int ∗feeds, int ∗items, int ∗comments)

  Update the relevant stat in the database, NULLs passed if that parameter is not to be updated.
- void RemoveStat (int id)

  Delete the stat from the database.
- void GenerateNewStat ()

  Generate the new statistics and store in the database.
- Stat ∗ GenerateStat (sql::ResultSet &rs)

  Generate a stat based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

    Database controller used to connect to the database.

- sql::Connection ∗ conn

    Connection to the database.

2.24.1 Detailed Description

A class called to create, update, delete or find stats.

2.24.2 Constructor & Destructor Documentation

2.24.2.1 StatController::StatController ( )

Initialising Constructor for StatController, opens a database connection.

2.24.2.2 StatController::∼StatController ( )

Deallocates any dynamic memory and closes the database connection.

2.24.3 Member Function Documentation

2.24.3.1 int StatController::AddStat (Stat & *stat*)

Insert the stat into the database.

Parameters

stat	stat to be added to the database

Returns

id of the newly added stat, -1 for an error

2.24.3.2 void StatController::GenerateNewStat ( )

Generate the new statistics and store in the database.

2.24.3.3 Stat ∗ StatController::GenerateStat (sql::ResultSet & *rs*)

Generate a stat based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a stat

Returns

the generated stat

2.24.3.4 vector< Stat * > StatController::GetAllStats ( )

Find all stats.

Returns

all stats in the database

2.24.3.5 Stat * StatController::GetLatestStat ( )

Find the latest statistic.

Returns

the stat with the latest id

2.24.3.6 Stat * StatController::GetStatById (int *id*)

Find the stat with that id.

Parameters

id	primary key id of the stat

Returns

the stat with that id

2.24.3.7 void StatController::RemoveStat (int *id*)

Delete the stat from the database.

Parameters

stat	stat to be removed from the database

2.24.3.8 void StatController::UpdateStat (Stat & *stat*)

Update the relevant stat in the database.

Parameters

stat	stat to be updated

2.24.3.9 void StatController::UpdateStat (int *id,* int * *users,* int * *sheets,* int * *feeds,* int * *items,* int * *comments*)

Update the relevant stat in the database, NULLs passed if that parameter is not to be updated.

Parameters

id	int identifier of the stat
users	int representing the number of users in the database, nullable
sheets	int representing the number of sheets in the database, nullable

feeds	int representing the number of feeds in the database, nullable
items	int representing the number of items in the database, nullable
comments	int representing the number of comments in the database, nullable

2.24.4 Member Data Documentation

2.24.4.1 sql::Connection∗ nfdb::StatController::conn `[private]`

Connection to the database.

2.24.4.2 DatabaseController∗ nfdb::StatController::dbc `[private]`

Database controller used to connect to the database.

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

- include/StatController.h
- src/StatController.cpp

2.25 nfdb::User Class Reference

A class representing a row in the User table.

`#include <User.h>`

Public Member Functions

- User ()

  Initialising Constructor for User, sets picture to NULL.
- User (std::string username, std::string password, char ∗picture, int pictureSize, nfrd::misc::DateTime registered, std::string realname, std::string email, int layout, bool admin, bool https)

  Non-Default Constructor for User.
- ∼User ()

  Destructor for User, does not perform any actions.
- void Destroy ()

  Cleans up any memory held by the User.

Public Attributes

- std::string username
- std::string password
- char ∗ picture
- int pictureSize
- nfrd::misc::DateTime registered
- std::string realname
- std::string email
- int layout
- bool admin
- bool https

2.25.1 Detailed Description

A class representing a row in the User table.

2.25.2 Constructor & Destructor Documentation

2.25.2.1 nfdb::User::User ( ) [inline]

Initialising Constructor for User, sets picture to NULL.

2.25.2.2 nfdb::User::User (std::string *username,* std::string *password,* char ∗ *picture,* int *pictureSize,* nfrd::misc::DateTime *registered,* std::string *realname,* std::string *email,* int *layout,* bool *admin,* bool *https*) [inline]

Non-Default Constructor for User.

Parameters

username	string identifier of the user
password	string representing the password of this user
picture	char∗ representing this users avatar, nullable
pictureSize	int representing the size in bytes of this users avatar, 0 for no picture
registered	datetime representing the date this user registered
realname	string representing the full name of this user
email	string representing the email address of this user
layout	int representing the which layout, dark or light, the user is using
admin	bool representing whether this user is an administrator
https	bool representing whether this user has ssl activated on their account

2.25.2.3 nfdb::User::∼User ( ) [inline]

Destructor for User, does not perform any actions.

2.25.3 Member Function Documentation

2.25.3.1 void nfdb::User::Destroy ( ) [inline]

Cleans up any memory held by the User.

2.25.4 Member Data Documentation

2.25.4.1 bool nfdb::User::admin

2.25.4.2 std::string nfdb::User::email

2.25.4.3 bool nfdb::User::https

2.25.4.4 int nfdb::User::layout

2.25.4.5 std::string nfdb::User::password

2.25.4.6 char∗ nfdb::User::picture

2.25.4.7 int nfdb::User::pictureSize

2.25.4.8 std::string nfdb::User::realname

2.25.4.9 nfrd::misc::DateTime nfdb::User::registered

2.25.4.10 std::string nfdb::User::username

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

- include/User.h

2.26 nfdb::UserController Class Reference

A class called to create, update, delete or find users.

`#include <UserController.h>`

Public Member Functions

- UserController ()

  Initialising Constructor for UserController, opens a database connection.
- ∼UserController ()

  Deallocates any dynamic memory and closes the database connection.
- User ∗ GetUserByUsername (std::string username)

  Find the user with that id.
- std::vector< User ∗ > GetAllUsers ()

  Find all users.
- int AddUser (User &user)

  Insert the user into the database.
- void UpdateUser (User &user)

  Update the relevant user in the database.
- void UpdateUser (std::string username, std::string ∗password, char ∗picture, int ∗pictureSize, nfrd::misc::-DateTime ∗registered, std::string ∗realname, std::string ∗email, int ∗layout, bool ∗admin, bool ∗https)

  Update the relevant user in the database, NULLs passed if that parameter is not to be updated.
- void RemoveUser (std::string username)

  Delete the user from the database.
- int GetNumberOfUsers ()

  Get the current number of users in the database.
- User ∗ GenerateUser (sql::ResultSet &rs)

  Generate a user based on the data in a result set row.

Private Attributes

- DatabaseController ∗ dbc

  Database controller used to connect to the database.
- sql::Connection ∗ conn

  Connection to the database.

2.26.1 Detailed Description

A class called to create, update, delete or find users.

2.26.2 Constructor & Destructor Documentation

2.26.2.1 UserController::UserController ( )

Initialising Constructor for UserController, opens a database connection.

2.26.2.2 UserController::∼UserController ( )

Deallocates any dynamic memory and closes the database connection.

2.26.3 Member Function Documentation

2.26.3.1 int UserController::AddUser (User & *user*)

Insert the user into the database.

Parameters

user	user to be added to the database

Returns

id of the newly added user, -1 for an error

2.26.3.2 User ∗ UserController::GenerateUser (sql::ResultSet & *rs*)

Generate a user based on the data in a result set row.

Parameters

rs	result set pointing at the current row for generating a user

Returns

the generated user

2.26.3.3 vector< User ∗ > UserController::GetAllUsers ( )

Find all users.

Returns

all users in the database

2.26.3.4 int UserController::GetNumberOfUsers ( )

Get the current number of users in the database.

Returns

the number of users in the database

2.26.3.5 User ∗ UserController::GetUserByUsername (std::string *username*)

Find the user with that id.

Parameters

id	primary key id of the user

Returns

the user with that id

2.26.3.6 void UserController::RemoveUser (std::string *username*)

Delete the user from the database.

Parameters

user	user to be removed from the database

2.26.3.7 void UserController::UpdateUser (User & *user*)

Update the relevant user in the database.

Parameters

user	user to be updated

2.26.3.8 void nfdb::UserController::UpdateUser (std::string *username,* std::string ∗ *password,* char ∗ *picture,* int ∗ *pictureSize,* nfrd::misc::DateTime ∗ *registered,* std::string ∗ *realname,* std::string ∗ *email,* int ∗ *layout,* bool ∗ *admin,* bool ∗ *https*)

Update the relevant user in the database, NULLs passed if that parameter is not to be updated.

Parameters

username	string identifier of the user
password	string representing the password of this user, nullable
picture	char∗ representing this users avatar, nullable
pictureSize	int representing the size in bytes of this users avatar, 0 for no picture
registered	datetime representing the date this user registered, nullable
realname	string representing the full name of this user, nullable
email	string representing the email address of this user, nullable
layout	int representing the which layout, dark or light, the user is using, nullable
admin	bool representing whether this user is an administrator, nullable
https	bool representing whether this user has ssl activated on their account, nullable

2.26.4 Member Data Documentation

2.26.4.1 sql::Connection∗ nfdb::UserController::conn `[private]`

Connection to the database.

2.26.4.2 DatabaseController* nfdb::UserController::dbc [private]

Database controller used to connect to the database.

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

- include/UserController.h
- src/UserController.cpp

Chapter 3

File Documentation

3.1 include/Comment.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::Comment

  A class representing a row in the Comments table.

Namespaces

- namespace nfdb

3.1.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.1.2 DESCRIPTION

Defines the struct class representing a Comment record

3.2 include/CommentController.h File Reference

```
#include "Comment.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::CommentController

  A class called to create, update, delete or find comments.

Namespaces

- namespace nfdb

3.2.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.2.2 DESCRIPTION

Defines a class called to create, update, delete or find comments.

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.2.3 DESCRIPTION

Defines a class called to connect and disconnect from the database.

3.3 include/ContentPlaceholder.h File Reference

Classes

- class nfdb::ContentPlaceholder

  A class representing a row in the ContentPlaceholders table.

Namespaces

- namespace nfdb

3.3.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.3.2 DESCRIPTION

Defines the struct class representing a Comment record

3.4 include/ContentPlaceholderController.h File Reference

```
#include "ContentPlaceholder.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
```

Classes

- class nfdb::ContentPlaceholderController

  A class called to create, update, delete or find content placeholders.

Namespaces

- namespace nfdb

3.4.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.4.2 DESCRIPTION

Defines a class called to create, update, delete or find content placeholders.

3.5 include/DatabaseController.h File Reference

```
#include <cppconn/driver.h>
#include <cppconn/connection.h>
#include <string>
```

Classes

- class nfdb::DatabaseController

  A class called to connect and disconnect from the database.

Namespaces

- namespace nfdb

3.6 include/Feed.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::Feed

  A class representing a row in the Feeds table.

Namespaces

- namespace nfdb

3.6.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.6.2 DESCRIPTION

Defines the struct class representing a Feed record

3.7 include/FeedController.h File Reference

```
#include "Feed.h"
#include "QueueItem.h"
#include "DatabaseController.h"
#include "nfrd/DateTime.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::FeedController

  A class called to create, update, delete or find feeds.

Namespaces

- namespace nfdb

3.7.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.7.2 DESCRIPTION

Defines a class called to create, update, delete or find feeds.

3.8 include/GroupPlaceholder.h File Reference

Classes

- class nfdb::GroupPlaceholder

  A class representing a row in the GroupPlaceholders table.

Namespaces

- namespace nfdb

3.8.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.8.2 DESCRIPTION

Defines the struct class representing a GroupPlaceholder record

3.9 include/GroupPlaceholderController.h File Reference

```
#include "GroupPlaceholder.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
```

Classes

- class nfdb::GroupPlaceholderController

  A class called to create, update, delete or find group placeholders.

Namespaces

- namespace nfdb

3.9.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.9.2 DESCRIPTION

Defines a class called to create, update, delete or find group placeholders.

3.10 include/Image.h File Reference

```
#include <cstdlib>
#include <string>
```

Classes

- class nfdb::Image

  A class representing a row in the Images table.

Namespaces

- namespace nfdb

3.10.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.10.2 DESCRIPTION

Defines the struct class representing an Image record

3.11 include/ImageController.h File Reference

```
#include "Image.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
```

Classes

- class nfdb::ImageController

  A class called to create, update, delete or find images.

Namespaces

- namespace nfdb

3.11.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.11.2 DESCRIPTION

Defines a class called to create, update, delete or find images.

3.12 include/Item.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::Item

  A class representing a row in the Items table.

Namespaces

- namespace nfdb

3.12.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.12.2 DESCRIPTION

Defines the struct class representing an Item record

3.13 include/ItemController.h File Reference

```
#include "Item.h"
#include "DatabaseController.h"
#include "nfrd/DateTime.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::ItemController

  A class called to create, update, delete or find items.

Namespaces

- namespace nfdb

3.13.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.13.2 DESCRIPTION

Defines a class called to create, update, delete or find items.

3.14 include/Layout.h File Reference

Classes

- class nfdb::Layout

  A class representing a row in the Layout table.

Namespaces

- namespace nfdb

3.14.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.14.2 DESCRIPTION

Defines the struct class representing a Layout record

3.15 include/LayoutController.h File Reference

```
#include "Layout.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
```

Classes

- class nfdb::LayoutController

  A class called to create, update, delete or find layouts.

Namespaces

- namespace nfdb

3.15.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.15.2 DESCRIPTION

Defines a class called to create, update, delete or find layouts.

3.16 include/Notification.h File Reference

```
#include <string>
```

Classes

- class nfdb::Notification

  A class representing a row in the Notification table.

Namespaces

- namespace nfdb

3.16.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.16.2 DESCRIPTION

Defines the struct class representing a Notification record

3.17 include/NotificationController.h File Reference

```
#include "Notification.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::NotificationController

  A class called to create, update, delete or find notifications.

Namespaces

- namespace nfdb

3.17.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.17.2 DESCRIPTION

Defines a class called to create, update, delete or find notifications.

3.18 include/QueueItem.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::QueueItem

  A class representing a QueueItem for the queue, derived from the Item table.

Namespaces

- namespace nfdb

3.18.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.18.2 DESCRIPTION

Defines the struct class representing a QueueItem record, derived from the Item class

3.19 include/Session.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::Session

  A class representing a row in the Session table.

Namespaces

- namespace nfdb

3.19.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.19.2 DESCRIPTION

Defines the struct class representing a Session record

3.20 include/SessionController.h File Reference

```
#include "Session.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::SessionController

  A class called to create, update, delete or find sessions.

Namespaces

- namespace nfdb

3.20.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.20.2 DESCRIPTION

Defines a class called to create, update, delete or find sessions.

3.21 include/Sheet.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::Sheet

  A class representing a row in the Sheet table.

Namespaces

- namespace nfdb

3.21.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.21.2 DESCRIPTION

Defines the struct class representing a Sheet record

3.22 include/SheetController.h File Reference

```
#include "Sheet.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::SheetController

  A class called to create, update, delete or find sheets.

Namespaces

- namespace nfdb

3.22.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.22.2 DESCRIPTION

Defines a class called to create, update, delete or find sheets.

3.23 include/Stat.h File Reference

Classes

- class nfdb::Stat

  A class representing a row in the Stat table.

Namespaces

- namespace nfdb

3.23.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.23.2 DESCRIPTION

Defines the struct class representing a Stat record

3.24 include/StatController.h File Reference

```
#include "Stat.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
```

Classes

- class nfdb::StatController

    A class called to create, update, delete or find stats.

Namespaces

- namespace nfdb

3.24.1 Detailed Description

Author

Michael Boge `mjb932@uow.edu.au`

Date

5/8/12

3.24.2 DESCRIPTION

Defines a class called to create, update, delete or find stats.

3.25 include/User.h File Reference

```
#include <string>
#include "nfrd/DateTime.h"
```

Classes

- class nfdb::User

    A class representing a row in the User table.

Namespaces

- namespace nfdb

3.25.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.25.2 DESCRIPTION

Defines the struct class representing a User record

3.26 include/UserController.h File Reference

```
#include "User.h"
#include "DatabaseController.h"
#include <cppconn/connection.h>
#include <cppconn/resultset.h>
#include <vector>
#include <string>
```

Classes

- class nfdb::UserController
  A class called to create, update, delete or find users.

Namespaces

- namespace nfdb

3.26.1 Detailed Description

Author

Michael Boge mjb932@uow.edu.au

Date

5/8/12

3.26.2 DESCRIPTION

Defines a class called to create, update, delete or find users.

3.27 src/CommentController.cpp File Reference

```
#include "CommentController.h"
#include <cppconn/prepared_statement.h>
#include <sstream>
```

3.28 src/ContentPlaceholderController.cpp File Reference

```
#include "ContentPlaceholderController.h"
#include <cppconn/prepared_statement.h>
```

3.29 src/DatabaseController.cpp File Reference

```
#include "DatabaseController.h"
#include <cppconn/exception.h>
```

Variables

- const string DBHOST = "tcp://nbs.im:3306"
- const string USER = "newsfeeder"
- const string PASSWORD = "Ewv3M5dMFHvEWSGT"
- const string DATABASE = "newsfeeder_nf"

3.29.1 Variable Documentation

3.29.1.1 const string DATABASE = "newsfeeder_nf"

3.29.1.2 const string DBHOST = "tcp://nbs.im:3306"

3.29.1.3 const string PASSWORD = "Ewv3M5dMFHvEWSGT"

3.29.1.4 const string USER = "newsfeeder"

3.30 src/FeedController.cpp File Reference

```
#include "FeedController.h"
#include <cppconn/prepared_statement.h>
#include <sstream>
```

3.31 src/GroupPlaceholderController.cpp File Reference

```
#include "GroupPlaceholderController.h"
#include <cppconn/prepared_statement.h>
```

3.32 src/ImageController.cpp File Reference

```
#include "ImageController.h"
#include <cppconn/prepared_statement.h>
#include <sstream>
```

3.33 src/ItemController.cpp File Reference

```
#include "ItemController.h"
#include <cppconn/prepared_statement.h>
#include <cppconn/exception.h>
```

3.34 src/LayoutController.cpp File Reference

```
#include "LayoutController.h"
#include <cppconn/prepared_statement.h>
```

3.35 src/NotificationController.cpp File Reference

```
#include "NotificationController.h"
#include <cppconn/prepared_statement.h>
```

3.36 src/SessionController.cpp File Reference

```
#include "SessionController.h"
#include <cppconn/prepared_statement.h>
```

3.37 src/SheetController.cpp File Reference

```
#include "SheetController.h"
#include <cppconn/prepared_statement.h>
```

3.38 src/StatController.cpp File Reference

```
#include "StatController.h"
#include <cppconn/prepared_statement.h>
```

3.39 src/UserController.cpp File Reference

```
#include "UserController.h"
#include <cppconn/prepared_statement.h>
#include <iostream>
#include <sstream>
```

Index