# Classes

|  |  |
| --- | --- |
| **Full name of the class** | **Brief Description** |
| User | Represents the users (and administrators) who access the system. |
| Sheet | Represents a sheet/view which displays the feeds in the system. |
| Feed | Represents a news (or other content) feed in the system. |
| Item | Represents a news item or piece of content from a feed |
| Image | Represents an image for a particular item. |
| Comment | Represents a comment made by a user on a particular item. |
| Session | Represents one of the of the currently active user sessions. |
| Stat | Represents statistics logged by the backend. |
| Notification | Represents the notifications shown to the user. |
| Layout | Represents the layout of a sheet in terms of rows and columns. |
| GroupPlaceholder | Represents an area where content placeholders can be placed on a sheet. |
| ContentPlaceholder | An abstract class, including only common settings for an area where feed content will be displayed. |
| RSSContentPlaceholder | A sub-class of ContentPlaceholder, representing an area where RSS feed content is displayed. |
| WeatherContentPlaceholder | A sub-class of ContentPlaceholder, representing an area where weather content is displayed. |

## Entities

These classes are simply structures that represent the tables in the database. They consist of a number of private members that relate to the columns in the tables and accessor/mutator methods to operate on these variables.

### User

Entity representing the users (and administrators) who access the system. It contains members such as username and password.

### Sheet

Entity representing a sheet/view which displays the feeds in the system. It contains members such as the user it belongs to and last updated date.

### Feed

Entity representing a news (or other content) feed in the system. It contains members such as the feed type and …

### Item

Entity representing a news item or piece of content from a feed. It contains members such as title and author.

### Image

Entity representing an image for a particular item. It consists of the actual picture data in binary form and a link to the original image.

### Comment

Entity representing a comment made by a user on a particular item. It consists of the actual content and the author who wrote it.

### Session

Entity representing a record of the currently active user sessions.

### Stat

Entity representing statistics logged by the backend.

### Notification

Entity representing the notifications shown to the user.

### Layout

Entity representing the layout of a sheet in terms of rows and columns.

### GroupPlaceholder

Entity representing an area where content placeholders can be placed on a sheet.

### ContentPlaceholder

Entity representing an area where feed content will be displayed. This is an abstract class and it is further defined by content placeholders representing specific types of content (such as news, weather, forums). Each sub-class will define settings that specify how its specific type of content is displayed.

## Controllers

These classes are the interfaces used by the other sub-systems to interact with the database. They consist of a number of methods which are used to find, modify, delete and create entities. Each controller interacts solely with its related entity class. They return entities (individually or in collections) or responses based upon which of these actions they are performing.

|  |  |
| --- | --- |
| **Full name of the class** | **Brief Description** |
| UserController | called to create, update, delete or find users. It also includes methods which are used to authenticate and change the user’s password. |
| SheetController | Class called to create, update, delete or find sheets. |
| FeedController | Class called to create, update, delete or find feeds. |
| ItemController | Represents a news item or piece of content from a feed. |
| ImageController | Class called to create, update, delete or find items. |
| CommentController | Class called to create, update, delete or find comments. |
| SessionController | Class called to create, delete or find sessions. |
| StatController | Class called to create, delete or find statistic records. |
| NotificationController | Class called to create, delete or find user notifications. |
| LayoutController | Class called to create, update, delete or find sheet layouts. |
| GroupPlaceholderController | Class called to create, update, delete or find group placeholders. |
| ContentPlaceholderController | Class called to create, update, delete or find content placeholders. Its methods access both standard ContentPlaceholder records, the sub-classes of ContentPlaceholder. |
| DatabaseController | A Singleton class which represents the connection (or more correctly pool of connections) to the database. It contains methods which perform all database interactions, namely connecting and disconnecting from the database and executing queries. It also contains members that hold basic database information (such as the open connection, DBName, URL, etc). |

### UserController

Class called to create, update, delete or find users. It also includes methods which are used to authenticate and change the user’s password.

### SheetController

Class called to create, update, delete or find sheets.

### FeedController

Class called to create, update, delete or find feeds.

### ItemController

Class called to create, update, delete or find items.

### ImageController

Class called to create, update, delete or find images.

### CommentController

Class called to create, update, delete or find comments.

### SessionController

Class called to create, delete or find sessions.

### StatController

Class called to create, delete or find statistic records.

### NotificationController

Class called to create, delete or find user notifications.

### LayoutController

Class called to create, update, delete or find sheet layouts.

### GroupPlaceholderController

Class called to create, update, delete or find group placeholders.

### ContentPlaceholderController

Class called to create, update, delete or find content placeholders. Its methods access both the standard ContentPlaceholder records, and use that to access the specific type of ContentPlaceholder.

### DatabaseController

This class differs from the other controllers as it is not directly accessibly from most of the sub-systems (aside from the back-end for establishing the database connections). It is a Singleton class which represents the connection (or more correctly pool of connections) to the database. As such it contains methods which perform all database interactions, namely connecting and disconnecting from the database and executing queries. It also contains members that hold basic database information (such as the open connection, DBName, URL, etc).

## Utilities

These classes are used by the controllers to perform commonly used operations.

### QueryBuilder

This class is used by all controllers in the subsystem. The other classes create instances of this class within their own methods to create queries when needed. Its methods create a query string based on parameters passed into them by the other controllers and the type of query to be created.

### DateTime

This class is used to represent the ‘Date and Time’ values stored within SQL.

# Shared Database Library

The Shared Database Library is a C++ library used by the frontend, backend and web service sub-systems as a standard interface to access the Newsfeeder database. The library itself provides methods that can be called to access and modify the database entities and records via SQL statements. This library uses the MYSQL++ library to perform standard database tasks (such as executing a query) and instead focuses on generating application specific queries and retrieving the results.

This system was added to ensure all database calls within the system are performed in a standardised way and as such will achieve the same results. It was also added to separate the database connection/query generation from the other sub-systems.

As the library needs to be used by the different systems which are written in different programming languages (PHP for frontend, C++ for backend, Java for the web service), the library provides a number of wrappers/interfaces for each specific version (these are generated by SWIG, Simplified Wrapper and Interface Generator).

# Architecture

The Shared Database Library is designed using the Model View Controller pattern. Where the models are classes representing of the database records that are to be altered/retrieved and the controllers are the classes called in order to modify these models. The views in relation to this library are the other sub-systems which access it (namely the front-end, back-end and API web service) as they are using the library to retrieve the information from the models for use.

The Model View Controller pattern was chosen because the shared database library is to act as a standard interface used by the other sub-systems. As such, there are multiple ways to view and interact with the data and each subsystem will interact differently. This pattern enables the other sub-systems (acting as views) to interact with the database simply by using the library’s associated controller classes. The additional advantage of choosing this pattern is that any major changes regarding database access/interaction (such as changing to another database type or changing the schema) means the changes only have to be made once, in this library, rather than on all of the other subsystems.

The shared database library also makes use of the Singleton pattern in regards to the DatabaseController class. It uses the singleton here to prevent to prevent the system from opening too many connections to the database. Instead, the library will retrieve connections from the DatabaseController connection pool.