Apos - Database Manager

V0.1-alpha.2

Generated on Sun Nov 19 2023 16:17:01 for Apos - Database Manager by Doxygen 1.9.8

Sun Nov 19 2023 16:17:01

Chapter 1

Contributing to Apos - Database Manager

Thank you for considering contributing to Apos – Database Manager. It is people like you that make Apos—↩ Database Manager such a great tool.

1.1 Where do I go from here?

If you've noticed a bug or have a feature request, make one! It is generally best if you get confirmation of your bug or approval for your feature request this way before starting to code.

1.2 Fork & create a branch

If this is something you think you can fix, then fork and create a branch with a descriptive name. A good branch name would be (where issue #325 is the ticket you're working on):

```
git checkout -b 325-add-japanese-translations
```

1.3 Get the test suite running

Make sure you're using the latest version of Node.js and npm. Install the project's dependencies:

```
npm install
```

Run the test suite to ensure everything is working correctly:

```
npm test
```

1.4 Implement your fix or feature

At this point, you're ready to make your changes! Feel free to ask for help; everyone is a beginner at first.

1.5 Make a Pull Request

At this point, you should switch back to your main branch and make sure it is up to date with the latest code from the main repository:

```
git remote add upstream git@github.com:DefinitelyNotSimon13/Apos.git
git checkout main
git pull upstream main
```

Then update your feature branch from your local copy of main, and push it!

```
git checkout 325-add-japanese-translations
git rebase main
git push --set-upstream origin 325-add-japanese-translations
```

Go to the Apos – Database Manager repository and create a new Pull Request. Fill out the form and wait for the approval!

1.6 Keeping your Pull Request updated

If a maintainer asks you to “rebase” your PR, they're saying that a lot of code has changed, and that you need to update your branch, so it's easier to merge.

To learn more about rebasing in Git, there are a lot of `good resources` but here's the suggested workflow:

```
git checkout 325-add-japanese-translations
git pull --rebase upstream main
git push --force-with-lease 325-add-japanese-translations
```

1.7 Merging a PR (maintainers only)

A PR can only be merged into main by a maintainer if:

- It is passing CI.
- It has been approved by at least two maintainers. If it was a maintainer who opened the PR, only one extra approval is needed.
- It has no requested changes.
- It is up to date with the current main.

Any maintainer is allowed to merge a PR if these conditions are met.

This `CONTRIBUTING.md` file provides a guide for contributors on how to create a branch, run the test suite, implement a fix or feature, make a pull request, and keep a pull request updated. It also provides instructions for maintainers on how to merge a pull request.

Chapter 2

Apos - Database Manager

Developed by Simon Blum, Apos - Database Manager is a comprehensive Windows application designed to manage SQLite3 databases.

The application provides an intuitive and user-friendly interface for performing a variety of database operations. Whether you need to open, read, edit tables, or execute custom SQLite3 commands, Apos - Database Manager has got you covered.

The name "Apos" is inspired by the ancient Greek word "", which translates to "storage" or "warehouse".

2.1 Key Features

- **Efficient Database Operations**: Open, read, and edit tables within SQLite3 databases with ease.
- **Custom SQLite3 Commands**: Execute custom SQLite3 commands for advanced database operations.
- **Intuitive User Interface**: Navigate through the application effortlessly with the GUI, built using the Qt framework.

2.2 Prerequisites

- Windows Operating System

2.3 Installation

To install Apos - Database Manager, follow these steps:

1. Navigate to the `Releases` page.
2. Download the latest installer.
3. Run the installer and follow the on-screen instructions.

2.4 Usage

1. **Opening a Database**: Launch the application and click on the "Open Database" button to select the desired SQLite3 database file.
2. **Executing Commands**: Enter custom SQLite3 commands in the "Command Box" and click the "Execute" button.
3. **Managing Tables**: Use the GUI to perform various table operations such as "Add," "Update," and "Select Table".

2.5 Project Structure

The project is organized into several directories:

- `/source`: Contains the main entry point for the application (`main.cpp`).
- `/classes/backendClasses`: Contains classes related to the backend logic of the application, such as `StartupHandler` and `ObjectHandler`.
- `/classes/frontendClasses`: Contains classes related to the frontend logic of the application, such as `WindowHandler` and various window classes.
- `/classes/databaseClasses`: Contains classes related to database operations, such as `DatabaseHandler` and `TableHandler`.
- `/resources`: Stores additional resources used by the application, such as translation files and the application logo.
- `/docs`: Contains the Doxygen-generated documentation for the project.

2.6 Documentation

Comprehensive documentation for Apos - Database Manager is available on the `GitHub Pages website` and in the `/docs` directory of this repository.

2.7 Contributing

As a solo developer, I welcome contributions to the project! Please refer to the Contribution Guidelines for details.

2.8 License

Apos - Database Manager is licensed under the LGPL-3.0 License.

2.9 Acknowledgments

- A big thank you to the Qt framework community for their invaluable resources and support.
- [Any other acknowledgments or credits]

Chapter 3

Topic Index

3.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Topic Documentation

8.1 Constructors and Desctructors

Group of constructors and destructors in the application.

Functions

- AposBackend::ObjectHandler::ObjectHandler (QSharedPointer< QApplication > newApplication, QShared↩ Pointer< AposDatabase::DatabaseHandler > newDbHandler, QSharedPointer< AposDatabase::TableHandler > newTableHandler)

  Constructor for the ObjectHandler class.
- AposBackend::StartupHandler::StartupHandler (const QSharedPointer< QApplication > &application)

  Constructor for the StartupHandler class.
- AposDatabase::DatabaseHandler::DatabaseHandler ()

  Constructor for the DatabaseHandler class.
- AposDatabase::TableHandler::TableHandler (QSharedPointer< DatabaseHandler > newDbHandler)

  Constructor for the TableHandler class.
- AposDatabase::TableHandler::TableHandler (QSharedPointer< DatabaseHandler > newDbHandler, const QString &tableName)

  Constructor for the TableHandler class.
- AposDatabase::TableHandler::~TableHandler ()

  Destructor for the TableHandler class.
- AposFrontend::DevWindow::DevWindow (QWidget *parent=nullptr, QSharedPointer< AposBackend::ObjectHandler > objectHandler=nullptr)

  Constructor for the DevWindow class.
- AposFrontend::DevWindow::~DevWindow () override

  Destructor for the DevWindow class.
- AposFrontend::LauncherWindow::LauncherWindow (QWidget *parent=nullptr, QSharedPointer< AposBackend::ObjectHandler > newObjectHandler=nullptr)

  Constructor for the LauncherWindow class.
- AposFrontend::LauncherWindow::~LauncherWindow () override

  Destructor for the LauncherWindow class.
- AposFrontend::SettingsWindow::SettingsWindow (QWidget *parent=nullptr, QSharedPointer< AposBackend::ObjectHandler > newObjectHandler=nullptr)

  Constructor for the SettingsWindow class.
- AposFrontend::SettingsWindow::~SettingsWindow () override

Destructor for the SettingsWindow class.

- AposFrontend::TranslatableWindow::TranslatableWindow ()

  Constructor for the TranslatableWindow class.

- AposFrontend::WindowHandler::WindowHandler (QSharedPointer< AposBackend::ObjectHandler > new↩ ObjectHandler)

  Constructor for the WindowHandler class.

8.1.1 Detailed Description

Group of constructors and destructors in the application.

This group contains all the constructors and destructors used in the application. These functions are responsible for initializing and cleaning up objects.

8.1.2 Function Documentation

8.1.2.1 DatabaseHandler()

```
AposDatabase::DatabaseHandler::DatabaseHandler ( )  [default]
```

Constructor for the DatabaseHandler class.

This constructor initializes the DatabaseHandler object.

8.1.2.2 DevWindow()

```
AposFrontend::DevWindow::DevWindow (
            QWidget * parent = nullptr,
            QSharedPointer< AposBackend::ObjectHandler > objectHandler = nullptr )  [explicit]
```

Constructor for the DevWindow class.

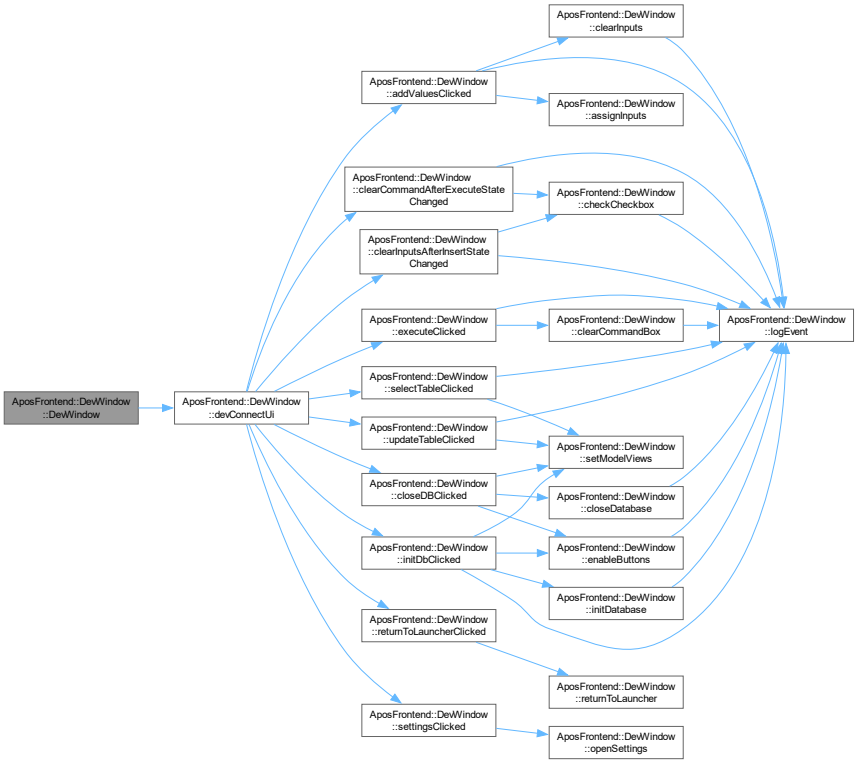
This constructor initializes the DevWindow object with a parent widget and an ObjectHandler object.

Parameters

parent	Pointer to the parent widget.
objectHandler	Shared pointer to the ObjectHandler object.

Definition at line 23 of file devwindow.cpp.

Here is the call graph for this function:



8.1.2.3 LauncherWindow()

```
AposFrontend::LauncherWindow::LauncherWindow (
            QWidget * parent = nullptr,
            QSharedPointer< AposBackend::ObjectHandler > newObjectHandler = nullptr )  [explicit]
```

Constructor for the LauncherWindow class.

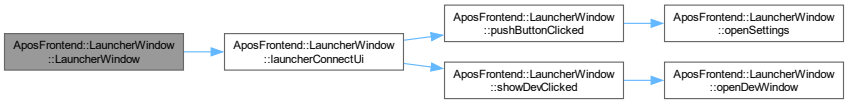
This constructor initializes the LauncherWindow object with a parent widget and an ObjectHandler object.

Parameters

parent	Pointer to the parent widget.
newObjectHandler	Shared pointer to the ObjectHandler object.

Definition at line 26 of file launcherwindow.cpp.

Here is the call graph for this function:



8.1.2.4 ObjectHandler()

```
AposBackend::ObjectHandler::ObjectHandler (
            QSharedPointer< QApplication > newApplication,
            QSharedPointer< AposDatabase::DatabaseHandler > newDbHandler,
            QSharedPointer< AposDatabase::TableHandler > newTableHandler )
```

Constructor for the ObjectHandler class.

This constructor initializes the ObjectHandler object with a QApplication object, a DatabaseHandler object, and a TableHandler object.

Parameters

newApplication	Shared pointer to the QApplication object.
newDbHandler	Shared pointer to the DatabaseHandler object.
newTableHandler	Shared pointer to the TableHandler object.

Definition at line 26 of file objecthandler.cpp.

8.1.2.5 SettingsWindow()

```
AposFrontend::SettingsWindow::SettingsWindow (
            QWidget * parent = nullptr,
            QSharedPointer< AposBackend::ObjectHandler > newObjectHandler = nullptr )  [explicit]
```

Constructor for the SettingsWindow class.

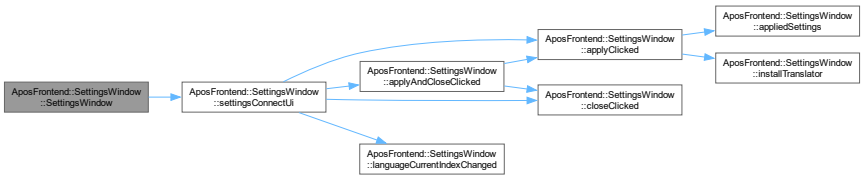
This constructor initializes the SettingsWindow object with a parent widget and an ObjectHandler object.

Parameters

parent	Pointer to the parent widget.
newObjectHandler	Shared pointer to the ObjectHandler object.

Definition at line 27 of file settingswindow.cpp.

Here is the call graph for this function:



8.1.2.6 StartupHandler()

```
AposBackend::StartupHandler::StartupHandler (
            const QSharedPointer< QApplication > & application )  [explicit]
```

Constructor for the StartupHandler class.

This constructor initializes the StartupHandler object with a QApplication object.

Parameters

application	Shared pointer to the QApplication object.

Definition at line 26 of file startuphandler.cpp.

8.1.2.7 TableHandler() [1/2]

```
AposDatabase::TableHandler::TableHandler (
            QSharedPointer< DatabaseHandler > newDbHandler )  [explicit]
```

Constructor for the TableHandler class.

This constructor initializes the TableHandler object with a DatabaseHandler object.

Parameters

newDbHandler	Shared pointer to the DatabaseHandler object.

Definition at line 29 of file tablehandler.cpp.

8.1.2.8 TableHandler() [2/2]

```
AposDatabase::TableHandler::TableHandler (
            QSharedPointer< DatabaseHandler > newDbHandler,
            const QString & tableName )
```

Constructor for the TableHandler class.

This constructor initializes the TableHandler object with a DatabaseHandler object and a table name.

Parameters

newDbHandler	Shared pointer to the DatabaseHandler object.
tableName	The name of the table.

Definition at line 33 of file tablehandler.cpp.

8.1.2.9 TranslatableWindow()

```
AposFrontend::TranslatableWindow::TranslatableWindow ( )  [default]
```

Constructor for the TranslatableWindow class.

This constructor initializes the TranslatableWindow object.

8.1.2.10 WindowHandler()

```
AposFrontend::WindowHandler::WindowHandler (
            QSharedPointer< AposBackend::ObjectHandler > newObjectHandler )  [explicit]
```

Constructor for the WindowHandler class.

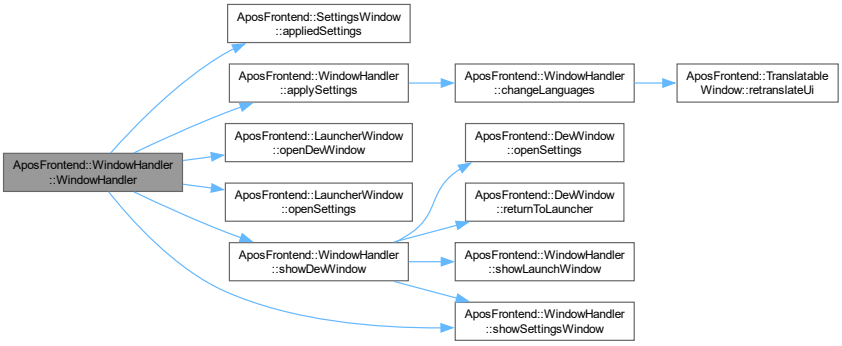
This constructor initializes the WindowHandler object with an ObjectHandler object.

Parameters

newObjectHandler	Shared pointer to the ObjectHandler object.

Definition at line 28 of file windowhandler.cpp.

Here is the call graph for this function:



8.1.2.11 ~DevWindow()

```
AposFrontend::DevWindow::~DevWindow ( )  [override]
```

Destructor for the DevWindow class.

This destructor cleans up the DevWindow object.

Definition at line 40 of file devwindow.cpp.

8.1.2.12 ~LauncherWindow()

```
AposFrontend::LauncherWindow::~LauncherWindow ( )  [override]
```

Destructor for the LauncherWindow class.

This destructor cleans up the LauncherWindow object.

Definition at line 35 of file launcherwindow.cpp.

8.1.2.13 ~SettingsWindow()

```
AposFrontend::SettingsWindow::~SettingsWindow ( )  [override]
```

Destructor for the SettingsWindow class.

This destructor cleans up the SettingsWindow object.

Definition at line 36 of file settingswindow.cpp.

8.1.2.14 ~TableHandler()

```
AposDatabase::TableHandler::~TableHandler ( )
```

Destructor for the TableHandler class.

This destructor cleans up the TableHandler object.

Definition at line 44 of file tablehandler.cpp.

8.2 Database Functions

Group of database functions in the application.

Classes

- class AposBackend::ObjectHandler

  The ObjectHandler class is a part of the application's backend logic.
- class AposDatabase::DatabaseHandler

  Provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.
- class AposDatabase::TableHandler

  Provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

Functions

- void AposDatabase::DatabaseHandler::closeDatabase ()

  Closes the database.
- bool AposDatabase::DatabaseHandler::executeCommand (const QString &command)

  Executes a SQL command.
- void AposDatabase::TableHandler::generateTableModel ()

  Generates a table model.
- void AposDatabase::TableHandler::generateTableModel (const QString &tableName)

  Generates a table model with a specified table name.
- bool AposDatabase::TableHandler::insertIntoTable (const QString &tableName, const QString &value1, const QString &value2, const QString &value3, const QString &value4, const QString &value5)

  Inserts into a table.
- void AposFrontend::DevWindow::initDatabase ()

  Initializes the database.
- void AposFrontend::DevWindow::closeDatabase (const QSharedPointer< AposDatabase::DatabaseHandler > &db)

  Closes the database.
- void AposFrontend::DevWindow::setModelViews ()

  Sets the model views.
- void AposFrontend::DevWindow::setModelViews (const QSharedPointer< QSqlTableModel > &tableModel)

  Sets the model views with a table model.

8.2.1 Detailed Description

Group of database functions in the application.

This group contains all the functions that interact with the database. These functions are responsible for creating, reading, updating, and deleting data in the database.

8.2.2 Function Documentation

8.2.2.1 closeDatabase() [1/2]

```
void AposDatabase::DatabaseHandler::closeDatabase ( )
```

Closes the database.

This function closes the database of the DatabaseHandler object.

Definition at line 34 of file databasehandler.cpp.

8.2.2.2 closeDatabase() [2/2]

```
void AposFrontend::DevWindow::closeDatabase (
            const QSharedPointer< AposDatabase::DatabaseHandler > & db )  [private]
```

Closes the database.

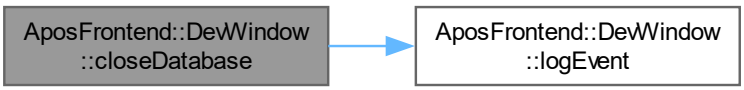
This function closes the database of the DevWindow object.

Parameters

db	Shared pointer to the DatabaseHandler object.

Definition at line 133 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.2.2.3 executeCommand()

```
bool AposDatabase::DatabaseHandler::executeCommand (
            const QString & command )
```

Executes a SQL command.

This function executes a specified SQL command.

Parameters

command	The SQL command to execute.

Returns

Boolean value indicating whether the command was executed successfully.

Definition at line 39 of file databasehandler.cpp.

8.2.2.4 generateTableModel() [1/2]

```
void AposDatabase::TableHandler::generateTableModel ( )
```

Generates a table model.

This function generates a table model for the TableHandler object.

Definition at line 49 of file tablehandler.cpp.

8.2.2.5 generateTableModel() [2/2]

```
void AposDatabase::TableHandler::generateTableModel (
            const QString & tableName )
```

Generates a table model with a specified table name.

This function generates a table model for the TableHandler object with a specified table name.

Parameters

tableName	The name of the table.

Definition at line 57 of file tablehandler.cpp.

8.2.2.6 initDatabase()

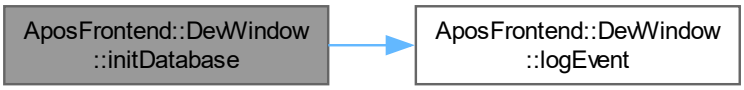
```
void AposFrontend::DevWindow::initDatabase ( )  [private]
```

Initializes the database.

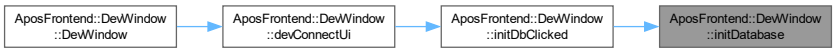
This function initializes the database of the DevWindow object.

Definition at line 115 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.2.2.7 insertIntoTable()

```
bool AposDatabase::TableHandler::insertIntoTable (
            const QString & tableName,
            const QString & value1,
            const QString & value2,
            const QString & value3,
            const QString & value4,
            const QString & value5 )
```

Inserts into a table.

This function inserts into a table with specified values.

Parameters

tableName	The name of the table.
value1	The first value to insert.
value2	The second value to insert.
value3	The third value to insert.
value4	The fourth value to insert.
value5	The fifth value to insert.

Returns

Boolean value indicating whether the insertion was successful.

Definition at line 67 of file tablehandler.cpp.

8.2.2.8 setModelViews() [1/2]

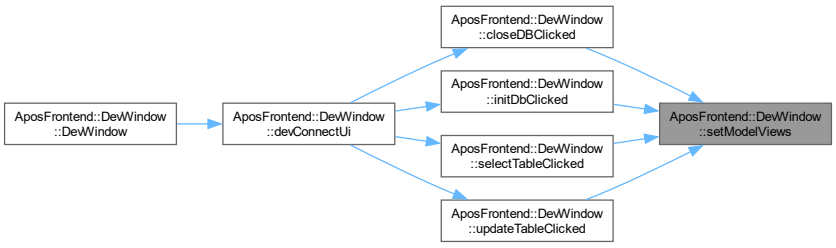
```
void AposFrontend::DevWindow::setModelViews ( )  [private]
```

Sets the model views.

This function sets the model views of the DevWindow object.

Definition at line 101 of file devwindow.cpp.

Here is the caller graph for this function:



8.2.2.9 setModelViews() [2/2]

```
void AposFrontend::DevWindow::setModelViews (
            const QSharedPointer< QSqlTableModel > & tableModel )  [private]
```

Sets the model views with a table model.

This function sets the model views of the DevWindow object with a specified table model.

Parameters

tableModel	Shared pointer to the QSqlTableModel object.

Definition at line 93 of file devwindow.cpp.

8.3 Initialization

Group of initialization functions in the application.

Classes

- class AposBackend::StartupHandler

  Provides the functionality for initializing the application's translator and ObjectHandler.

Functions

- bool AposBackend::ObjectHandler::initDatabaseObject ()

  Initializes the database object.
- bool AposBackend::ObjectHandler::initTableObject (const QString &inputTableName)

  Initializes the table object.
- QSharedPointer< ObjectHandler > AposBackend::StartupHandler::startUp ()

  Initializes the application's translator and ObjectHandler.
- static QSharedPointer< QTranslator > AposBackend::StartupHandler::initTranslator ()

  Initializes the application's translator.
- void AposBackend::StartupHandler::installTranslator ()

  Installs the application's translator.
- QSharedPointer< ObjectHandler > AposBackend::StartupHandler::initObjectHandler ()

  Initializes the application's ObjectHandler.
- bool AposDatabase::DatabaseHandler::initDatabase ()

  Initializes the database.
- void AposFrontend::SettingsWindow::installTranslator ()

  Installs the application's translator.

8.3.1 Detailed Description

Group of initialization functions in the application.

This group contains all the functions that are responsible for initializing various components of the application, such as the QApplication, StartupHandler, ObjectHandler, and WindowHandler objects.

8.3.2 Function Documentation

8.3.2.1 initDatabase()

```
bool AposDatabase::DatabaseHandler::initDatabase ( )
```

Initializes the database.

This function initializes the database of the DatabaseHandler object.

Returns

Boolean value indicating whether the database is initialized.

Definition at line 26 of file databasehandler.cpp.

8.3.2.2 initDatabaseObject()

```
bool AposBackend::ObjectHandler::initDatabaseObject ( )
```

Initializes the database object.

This function initializes the database object of the ObjectHandler object.

Returns

Boolean value indicating whether the database object is initialized.

Definition at line 34 of file objecthandler.cpp.

8.3.2.3 initObjectHandler()

```
QSharedPointer< ObjectHandler > AposBackend::StartupHandler::initObjectHandler ( )  [private]
```

Initializes the application's ObjectHandler.

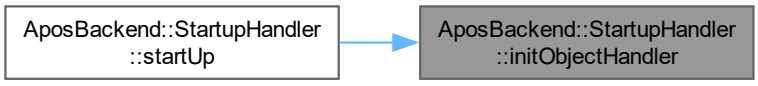
This function initializes the application's ObjectHandler.

Returns

Shared pointer to the initialized ObjectHandler object.

Definition at line 66 of file startuphandler.cpp.

Here is the caller graph for this function:



8.3.2.4 initTableObject()

```
bool AposBackend::ObjectHandler::initTableObject (
            const QString & inputTableName )
```

Initializes the table object.

This function initializes the table object of the ObjectHandler object with a specified table name.

Parameters

inputTableName	The name of the table.

Returns

Boolean value indicating whether the table object is initialized.

Definition at line 40 of file objecthandler.cpp.

8.3.2.5 initTranslator()

```
QSharedPointer< QTranslator > AposBackend::StartupHandler::initTranslator ( )  [static], [private]
```

Initializes the application's translator.

This function initializes the application's translator.

Returns

Shared pointer to the initialized QTranslator object.

Definition at line 51 of file startuphandler.cpp.

Here is the caller graph for this function:

8.3.2.6 installTranslator() [1/2]

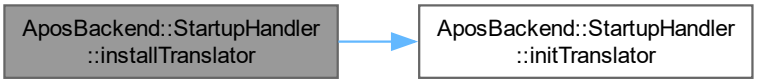
```
void AposBackend::StartupHandler::installTranslator ( )  [private]
```

Installs the application's translator.

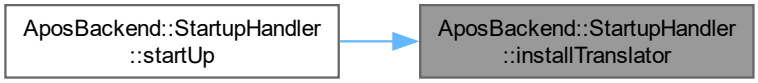
This function installs the application's translator.

Definition at line 44 of file startuphandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.3.2.7 installTranslator() [2/2]

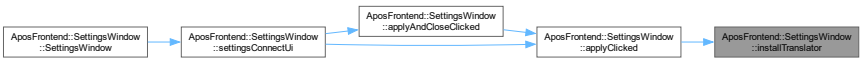
```
void AposFrontend::SettingsWindow::installTranslator ( )  [private]
```

Installs the application's translator.

This function installs the application's translator.

Definition at line 85 of file settingswindow.cpp.

Here is the caller graph for this function:



8.3.2.8 startUp()

```
QSharedPointer< ObjectHandler > AposBackend::StartupHandler::startUp ( )
```

Initializes the application's translator and ObjectHandler.

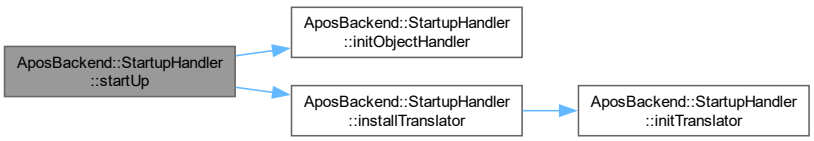
This function initializes the application's translator and ObjectHandler.

Returns

Shared pointer to the initialized ObjectHandler object.

Definition at line 33 of file startuphandler.cpp.

Here is the call graph for this function:



8.4 Log Functions

Group of log functions in the application.

Functions

- void AposFrontend::DevWindow::logEvent (const QString &message, const QSqlError &error)

  Logs an event with a message and a SQL error.
- void AposFrontend::DevWindow::logEvent (const QString &message)

  Logs an event with a message.

8.4.1 Detailed Description

Group of log functions in the application.

This group contains all the functions that are responsible for logging information. These functions are used to log information for debugging and tracking purposes.

8.4.2 Function Documentation

8.4.2.1 logEvent() [1/2]

```
void AposFrontend::DevWindow::logEvent (
            const QString & message )
```

Logs an event with a message.

This function logs an event with a specified message.

Parameters

message	The message of the event.

Definition at line 75 of file devwindow.cpp.

8.4.2.2 logEvent() [2/2]

```
void AposFrontend::DevWindow::logEvent (
            const QString & message,
            const QSqlError & error )
```

Logs an event with a message and a SQL error.

This function logs an event with a specified message and a SQL error.

Parameters

message	The message of the event.
error	The SQL error of the event.

Definition at line 68 of file devwindow.cpp.

8.5 Signal Functions

Group of signal functions in the application.

Signals

- void AposFrontend::DevWindow::returnToLauncher ()
  Signal for returning to the launcher.
- void AposFrontend::DevWindow::openSettings ()
  Signal for opening the settings.
- void AposFrontend::LauncherWindow::openDevWindow ()
  Signal for opening the developer window.
- void AposFrontend::LauncherWindow::openSettings ()
  Signal for opening the settings.
- void AposFrontend::SettingsWindow::appliedSettings ()
  Signal for applying settings.

8.5.1 Detailed Description

Group of signal functions in the application.

This group contains all the signal functions in the application. These functions are used to emit signals that can be handled by slot functions.

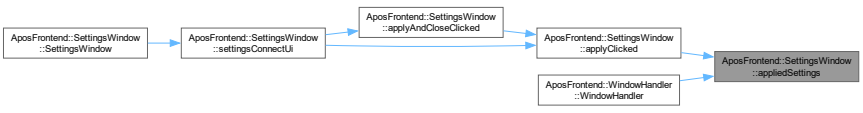
8.5.2 Signals

8.5.2.1 appliedSettings

```
void AposFrontend::SettingsWindow::appliedSettings ( )  [signal]
```

Signal for applying settings.

This signal is emitted when the user applies the settings. Here is the caller graph for this function:

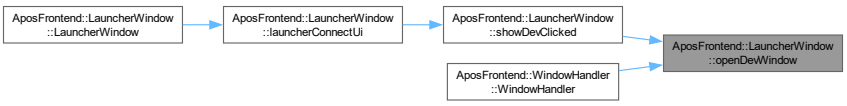


8.5.2.2 openDevWindow

```
void AposFrontend::LauncherWindow::openDevWindow ( )  [signal]
```

Signal for opening the developer window.

This signal is emitted when the user wants to open the developer window. Here is the caller graph for this function:

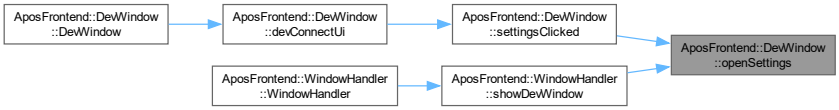


8.5.2.3 openSettings [1/2]

```
void AposFrontend::DevWindow::openSettings ( )  [signal]
```

Signal for opening the settings.

This signal is emitted when the user wants to open the settings. Here is the caller graph for this function:

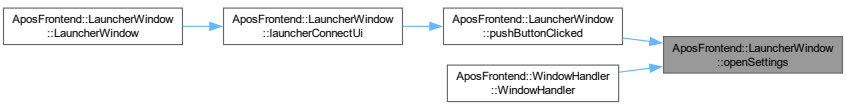


8.5.2.4 openSettings [2/2]

```
void AposFrontend::LauncherWindow::openSettings ( )  [signal]
```

Signal for opening the settings.

This signal is emitted when the user wants to open the settings. Here is the caller graph for this function:

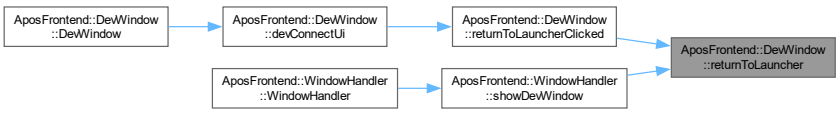


8.5.2.5 returnToLauncher

```
void AposFrontend::DevWindow::returnToLauncher ( )  [signal]
```

Signal for returning to the launcher.

This signal is emitted when the user wants to return to the launcher. Here is the caller graph for this function:



8.6 Slot Functions

Group of slot functions in the application.

Private Slots

- void AposFrontend::DevWindow::initDbClicked ()

  Slot for the 'InitDB' button click event.
- void AposFrontend::DevWindow::closeDBClicked ()

  Slot for the 'CloseDB' button click event.
- void AposFrontend::DevWindow::executeClicked ()

  Slot for the 'Execute' button click event.
- void AposFrontend::DevWindow::selectTableClicked ()

  Slot for the 'SelectTable' button click event.
- void AposFrontend::DevWindow::addValuesClicked ()

  Slot for the 'Add' button click event.
- void AposFrontend::DevWindow::updateTableClicked ()

  Slot for the 'Update' button click event.
- void AposFrontend::DevWindow::clearCommandAfterExecuteStateChanged (int arg1)

  Slot for the 'clearCommandAfterExecute' state change event.
- void AposFrontend::DevWindow::clearInputsAfterInsertStateChanged (int arg1)

  Slot for the 'clearInputsAfterInsert' state change event.
- void AposFrontend::DevWindow::returnToLauncherClicked ()

  Slot for the 'ReturnToLauncher' button click event.
- void AposFrontend::DevWindow::settingsClicked ()

  Slot for the 'Settings' button click event.
- void AposFrontend::LauncherWindow::showDevClicked ()

  Slot for the 'ShowDev' button click event.
- void AposFrontend::LauncherWindow::pushButtonClicked ()

  Slot for the 'PushButton' button click event.
- void AposFrontend::SettingsWindow::closeClicked ()

  Slot for the 'Close' button click event.
- void AposFrontend::SettingsWindow::applyClicked ()

  Slot for the 'Apply' button click event.
- void AposFrontend::SettingsWindow::languageCurrentIndexChanged (int index)

  Slot for the 'Language' combo box index change event.

8.6.1 Detailed Description

Group of slot functions in the application.

This group contains all the slot functions in the application. These functions are used to handle signals emitted by other objects.

8.6.2 Private Slots

8.6.2.1 addValuesClicked

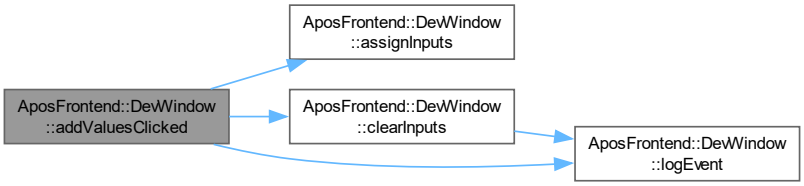
```
void AposFrontend::DevWindow::addValuesClicked ( )  [private], [slot]
```

Slot for the 'Add' button click event.

This slot is triggered when the 'Add' button is clicked.

Definition at line 218 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.2 applyClicked

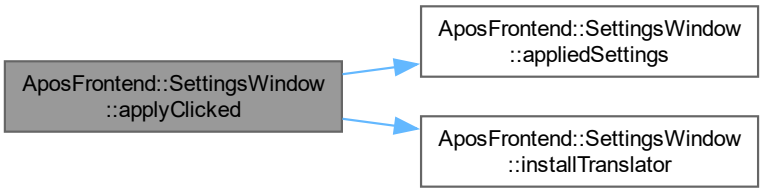
```
void AposFrontend::SettingsWindow::applyClicked ( )  [private], [slot]
```

Slot for the 'Apply' button click event.

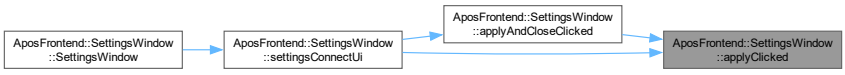
This slot is triggered when the 'Apply' button is clicked.

Definition at line 57 of file settingswindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.2.3 clearCommandAfterExecuteStateChanged

```
void AposFrontend::DevWindow::clearCommandAfterExecuteStateChanged (
            int arg1 )  [private], [slot]
```

Slot for the 'clearCommandAfterExecute' state change event.

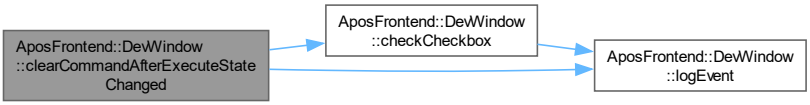
This slot is triggered when the state of the 'clearCommandAfterExecute' checkbox is changed.

Parameters

arg1	The new state of the checkbox.

Definition at line 240 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.4 clearInputsAfterInsertStateChanged

```
void AposFrontend::DevWindow::clearInputsAfterInsertStateChanged (
            int arg1 )  [private], [slot]
```

Slot for the 'clearInputsAfterInsert' state change event.

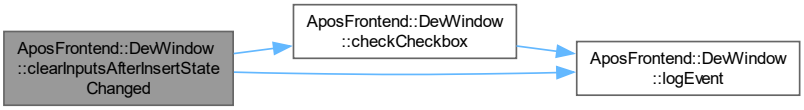
This slot is triggered when the state of the 'clearInputsAfterInsert' checkbox is changed.

Parameters

arg1	The new state of the checkbox.

Definition at line 246 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.5 closeClicked

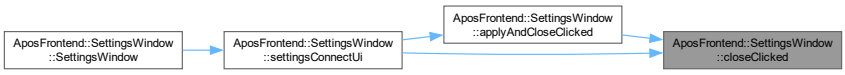
```
void AposFrontend::SettingsWindow::closeClicked ( )  [private], [slot]
```

Slot for the 'Close' button click event.

This slot is triggered when the 'Close' button is clicked.

Definition at line 52 of file settingswindow.cpp.

Here is the caller graph for this function:



8.6.2.6 closeDBClicked

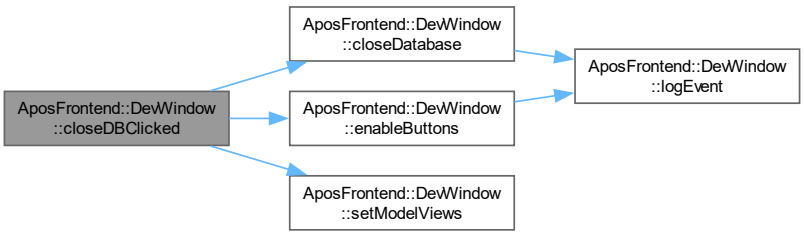
```
void AposFrontend::DevWindow::closeDBClicked ( )  [private], [slot]
```

Slot for the 'CloseDB' button click event.

This slot is triggered when the 'CloseDB' button is clicked.

Definition at line 193 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.7 executeClicked

```
void AposFrontend::DevWindow::executeClicked ( )  [private], [slot]
```

Slot for the 'Execute' button click event.

This slot is triggered when the 'Execute' button is clicked.

Definition at line 199 of file devwindow.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

8.6.2.8 initDbClicked

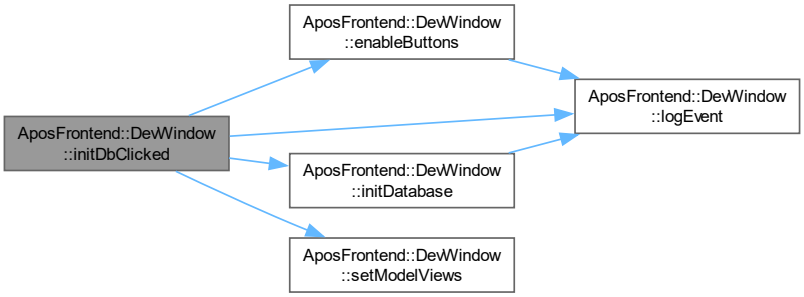
```
void AposFrontend::DevWindow::initDbClicked ( )  [private], [slot]
```

Slot for the 'InitDB' button click event.

This slot is triggered when the 'InitDB' button is clicked.

Definition at line 176 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.9 languageCurrentIndexChanged

```
void AposFrontend::SettingsWindow::languageCurrentIndexChanged (
            int index )  [private], [slot]
```

Slot for the 'Language' combo box index change event.

This slot is triggered when the index of the 'Language' combo box is changed.

Parameters

index	The new index of the combo box.

Definition at line 73 of file settingswindow.cpp.

Here is the caller graph for this function:

8.6.2.10 pushButtonClicked

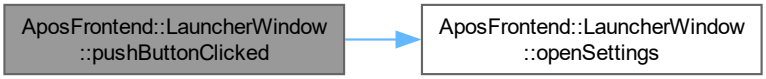
```
void AposFrontend::LauncherWindow::pushButtonClicked ( )  [private], [slot]
```

Slot for the 'PushButton' button click event.

This slot is triggered when the 'PushButton' button is clicked.

Definition at line 51 of file launcherwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.11 returnToLauncherClicked

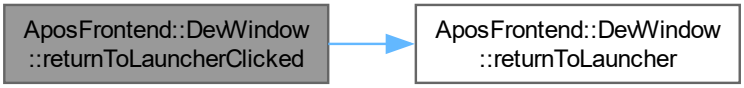
```
void AposFrontend::DevWindow::returnToLauncherClicked ( )  [private], [slot]
```

Slot for the 'ReturnToLauncher' button click event.

This slot is triggered when the 'ReturnToLauncher' button is clicked.

Definition at line 252 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.12 selectTableClicked

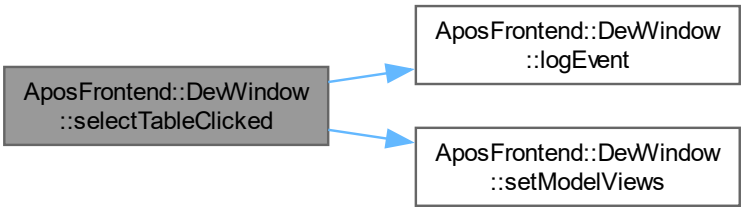
```
void AposFrontend::DevWindow::selectTableClicked ( )  [private], [slot]
```

Slot for the 'SelectTable' button click event.

This slot is triggered when the 'SelectTable' button is clicked.

Definition at line 211 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.13 settingsClicked

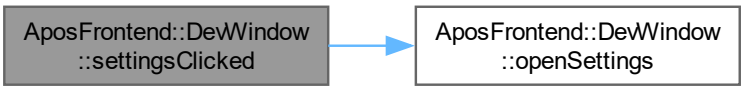
```
void AposFrontend::DevWindow::settingsClicked ( )  [private], [slot]
```

Slot for the 'Settings' button click event.

This slot is triggered when the 'Settings' button is clicked.

Definition at line 256 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.6.2.14 showDevClicked

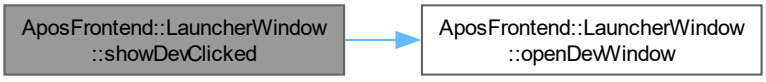
```
void AposFrontend::LauncherWindow::showDevClicked ( )  [private], [slot]
```

Slot for the 'ShowDev' button click event.

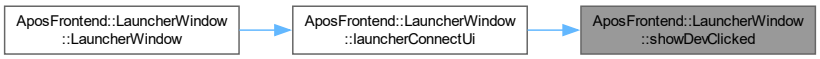
This slot is triggered when the 'ShowDev' button is clicked.

Definition at line 47 of file launcherwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.6.2.15 updateTableClicked

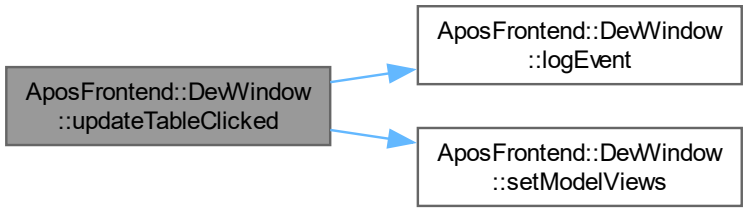
```
void AposFrontend::DevWindow::updateTableClicked ( )  [private], [slot]
```

Slot for the 'Update' button click event.

This slot is triggered when the 'Update' button is clicked.

Definition at line 233 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.7 UI Functions

Group of UI functions in the application.

Classes

- class AposFrontend::LauncherWindow

  Provides the user interface for the launcher window.
- class AposFrontend::SettingsWindow

  Provides the user interface for the settings window.
- class AposFrontend::TranslatableWindow

  An abstract base class that provides a function for retranslating the user interface.
- class AposFrontend::WindowHandler

  Provides the functionality for managing the application's windows.

Functions

- void AposFrontend::DevWindow::retranslateUi () override

  Retranslates the user interface.
- void AposFrontend::DevWindow::enableButtons (bool databaseLoaded)

  Enables or disables the buttons.
- void AposFrontend::DevWindow::assignInputs ()

  Assigns the inputs.
- void AposFrontend::DevWindow::clearInputs (bool clearBool)

  Clears the inputs.
- bool AposFrontend::DevWindow::checkCheckbox (int argCb)

  Checks a checkbox.
- void AposFrontend::DevWindow::clearCommandBox (bool clearBool)

  Clears the command box.
- void AposFrontend::LauncherWindow::retranslateUi () override

  Retranslates the user interface.
- void AposFrontend::SettingsWindow::retranslateUi () override

  Retranslates the user interface.
- virtual void AposFrontend::TranslatableWindow::retranslateUi ()=0

  Retranslates the user interface.
- void AposFrontend::WindowHandler::showLaunchWindow ()

  Shows the launcher window.
- void AposFrontend::WindowHandler::changeLanguages ()

  Changes languages.

Private Slots

- void AposFrontend::WindowHandler::showDevWindow ()

  Shows the developer window.
- void AposFrontend::WindowHandler::showSettingsWindow ()

  Shows the settings window.
- void AposFrontend::WindowHandler::applySettings ()

  Applies settings.

8.7.1 Detailed Description

Group of UI functions in the application.

This group contains all the functions that are responsible for handling the user interface of the application.

8.7.2 Function Documentation

8.7.2.1 assignInputs()

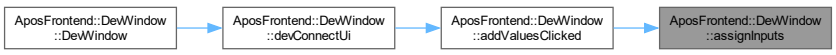
```
void AposFrontend::DevWindow::assignInputs ( )  [private]
```

Assigns the inputs.

This function assigns the inputs of the DevWindow object.

Definition at line 107 of file devwindow.cpp.

Here is the caller graph for this function:



8.7.2.2 changeLanguages()

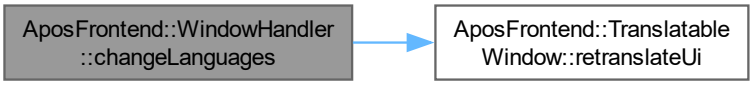
```
void AposFrontend::WindowHandler::changeLanguages ( )  [private]
```

Changes languages.

This function changes the language of the application's user interface.

Definition at line 72 of file windowhandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.7.2.3 checkCheckbox()

```
bool AposFrontend::DevWindow::checkCheckbox (
            int argCb )  [private]
```

Checks a checkbox.

This function checks a checkbox of the DevWindow object based on a specified integer value.

Parameters

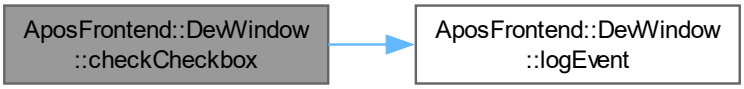
argCb	Integer value indicating the state of the checkbox.

Returns

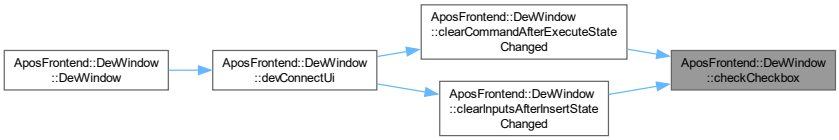
Boolean value indicating whether the checkbox is checked.

Definition at line 142 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.2.4 clearCommandBox()

```
void AposFrontend::DevWindow::clearCommandBox (
            bool clearBool )  [private]
```

Clears the command box.

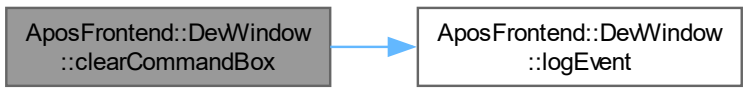
This function clears the command box of the DevWindow object based on a specified boolean value.

Parameters

clearBool	Boolean value indicating whether to clear the command box.

Definition at line 168 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

8.7.2.5 clearInputs()

```
void AposFrontend::DevWindow::clearInputs (
            bool clearBool )  [private]
```

Clears the inputs.

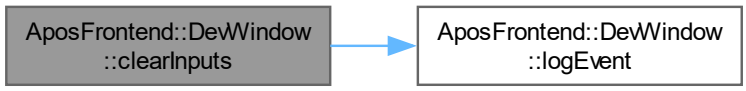
This function clears the inputs of the DevWindow object based on a specified boolean value.

Parameters

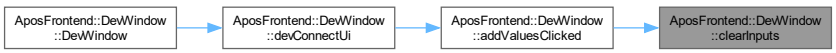
clearBool	Boolean value indicating whether to clear the inputs.

Definition at line 156 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.2.6 enableButtons()

```
void AposFrontend::DevWindow::enableButtons (
            bool databaseLoaded )  [private]
```

Enables or disables the buttons.

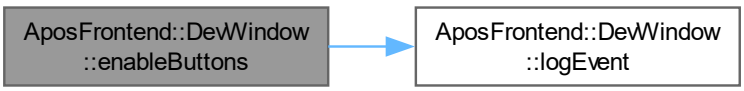
This function enables or disables the buttons of the DevWindow object based on whether the database is loaded.

Parameters

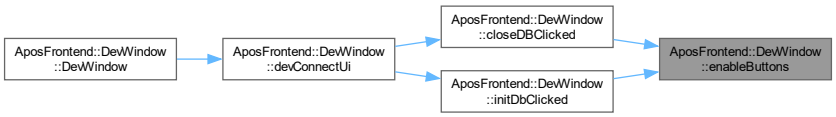
databaseLoaded	Boolean value indicating whether the database is loaded.

Definition at line 82 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.2.7 retranslateUi() [1/4]

```
void AposFrontend::DevWindow::retranslateUi ( )  [override], [virtual]
```

Retranslates the user interface.

This function retranslates the user interface of the DevWindow object.

Implements AposFrontend::TranslatableWindow.

Definition at line 260 of file devwindow.cpp.

8.7.2.8 retranslateUi() [2/4]

```
void AposFrontend::LauncherWindow::retranslateUi ( )  [override], [virtual]
```

Retranslates the user interface.

This function retranslates the user interface of the LauncherWindow object.

Implements AposFrontend::TranslatableWindow.

Definition at line 55 of file launcherwindow.cpp.

8.7.2.9 retranslateUi() [3/4]

```
void AposFrontend::SettingsWindow::retranslateUi ( )  [override], [virtual]
```

Retranslates the user interface.

This function retranslates the user interface of the SettingsWindow object.

Implements AposFrontend::TranslatableWindow.

Definition at line 48 of file settingswindow.cpp.

8.7.2.10 retranslateUi() [4/4]

```
virtual void AposFrontend::TranslatableWindow::retranslateUi ( )  [pure virtual]
```

Retranslates the user interface.

This is a pure virtual function that needs to be implemented by derived classes. It is used to retranslate the user interface of the window.

Implemented in AposFrontend::DevWindow, AposFrontend::LauncherWindow, and AposFrontend::SettingsWindow.

Here is the caller graph for this function:

8.7.2.11 showLaunchWindow()

```
void AposFrontend::WindowHandler::showLaunchWindow ( )
```

Shows the launcher window.

This function shows the launcher window of the application.

Definition at line 40 of file windowhandler.cpp.

Here is the caller graph for this function:

8.7.3 Private Slots

8.7.3.1 applySettings

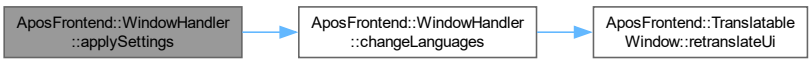
```
void AposFrontend::WindowHandler::applySettings ( )  [private], [slot]
```

Applies settings.

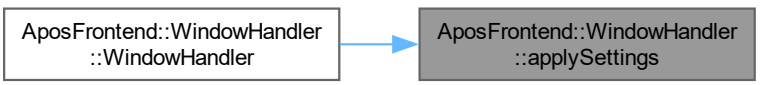
This slot is triggered when the user applies the settings.

Definition at line 68 of file windowhandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.2 showDevWindow

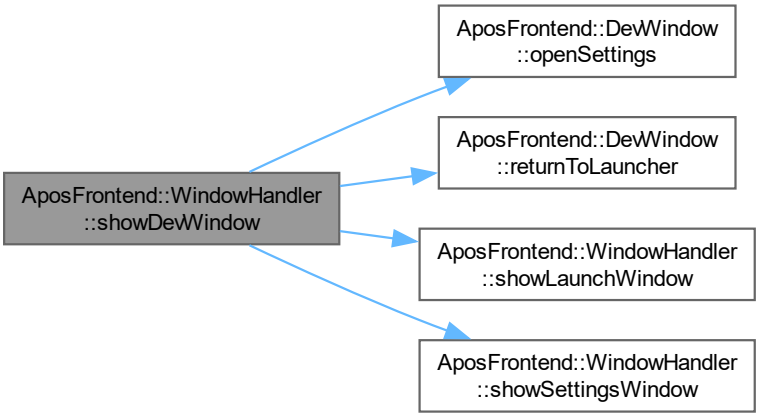
```
void AposFrontend::WindowHandler::showDevWindow ( )  [private], [slot]
```

Shows the developer window.

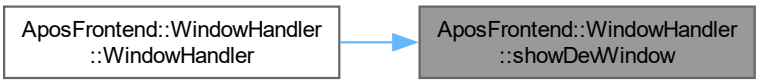
This slot is triggered when the user wants to open the developer window.

Definition at line 49 of file windowhandler.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.3.3 showSettingsWindow

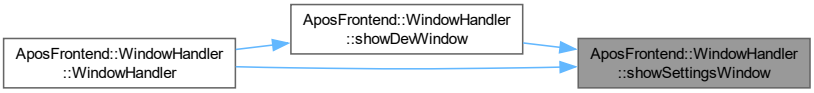
```
void AposFrontend::WindowHandler::showSettingsWindow ( )  [private], [slot]
```

Shows the settings window.

This slot is triggered when the user wants to open the settings window.

Definition at line 64 of file windowhandler.cpp.

Here is the caller graph for this function:



8.8 Utility Functions

Group of utility functions in the application.

Group of utility functions in the application.

This group contains all the utility functions in the application. These functions provide various utility services such as string manipulation, data conversion, etc.

8.9 Variables

Group of variables in the application.

Variables

- QSharedPointer< QApplication > AposBackend::ObjectHandler::ptrApplication

  Shared pointer to the QApplication object.
- QSharedPointer< AposDatabase::DatabaseHandler > AposBackend::ObjectHandler::ptrDbHandler

  Shared pointer to the DatabaseHandler object.
- QSharedPointer< AposDatabase::TableHandler > AposBackend::ObjectHandler::ptrTableHandler

  Shared pointer to the TableHandler object.
- QSharedPointer< QApplication > AposBackend::StartupHandler::ptrApplication

  Shared pointer to the QApplication object.
- QSharedPointer< ObjectHandler > AposBackend::StartupHandler::ptrObjectHandler

  Shared pointer to the ObjectHandler object.
- QSqlError AposDatabase::DatabaseHandler::lastSqlError

  The SQL error of the DatabaseHandler object.
- QSqlDatabase AposDatabase::DatabaseHandler::activeDatabase

  The active database of the DatabaseHandler object.
- QSharedPointer< QSqlDatabase > AposDatabase::DatabaseHandler::ptrActiveDatabase

  Shared pointer to the active database.
- QString AposDatabase::DatabaseHandler::databasePath = R"(C:\Users\Clean\Documents\Projekte\Apos-DatabaseManager\Project\resources\defaultDatabase\userDatabase.db)"

  The path to the database.
- QString AposDatabase::TableHandler::activeTableName = "userTable"

  The active table name of the TableHandler object.
- QSharedPointer< DatabaseHandler > AposDatabase::TableHandler::ptrDbHandler

  Shared pointer to the DatabaseHandler object.

- QSharedPointer< QSqlTableModel > AposDatabase::TableHandler::ptrTableModel
  Shared pointer to the QSqlTableModel object.
- QSqlError AposDatabase::TableHandler::lastTableError
  The last table error of the TableHandler object.
- Ui::DevWindow * AposFrontend::DevWindow::ui
  Pointer to the user interface of the DevWindow object.
- QSharedPointer< AposBackend::ObjectHandler > AposFrontend::DevWindow::ptrObjectHandler = nullptr
  Shared pointer to the ObjectHandler object.
- QString AposFrontend::DevWindow::input1
  String values for the inputs.
- bool AposFrontend::DevWindow::clearCommand = false
  Boolean values for clearing the command and the inputs.
- Ui::LauncherWindow * AposFrontend::LauncherWindow::ui
  Pointer to the user interface of the LauncherWindow object.
- QSharedPointer< AposBackend::ObjectHandler > AposFrontend::LauncherWindow::objectHandler = nullptr
  Shared pointer to the ObjectHandler object.
- Ui::SettingsWindow * AposFrontend::SettingsWindow::ui
  Pointer to the user interface of the SettingsWindow object.
- QSharedPointer< AposBackend::ObjectHandler > AposFrontend::SettingsWindow::ptrObjectHandler
  Shared pointer to the ObjectHandler object.
- QSharedPointer< QTranslator > AposFrontend::SettingsWindow::ptrTranslator
  Shared pointer to the QTranslator object.
- int AposFrontend::SettingsWindow::languageIndex {}
  The index of the selected language.
- int AposFrontend::SettingsWindow::tempLanguageIndex {}
  The temporary index of the selected language.
- bool AposFrontend::SettingsWindow::languageChanged {}
  Indicates whether the language has changed.
- QSharedPointer< LauncherWindow > AposFrontend::WindowHandler::ptrLauncherWindow
  Shared pointer to the LauncherWindow object.
- QSharedPointer< DevWindow > AposFrontend::WindowHandler::ptrDevWindow
  Shared pointer to the DevWindow object.
- QSharedPointer< SettingsWindow > AposFrontend::WindowHandler::ptrSettingsWindow
  Shared pointer to the SettingsWindow object.
- QSharedPointer< AposBackend::ObjectHandler > AposFrontend::WindowHandler::ptrObjectHandler
  Shared pointer to the ObjectHandler object.

8.9.1 Detailed Description

Group of variables in the application.

This group contains all the variables used in the application. These variables are used to store data and pass it between functions.

8.9.2 Variable Documentation

8.9.2.1 activeDatabase

```
QSqlDatabase AposDatabase::DatabaseHandler::activeDatabase  [private]
```

The active database of the DatabaseHandler object.

This variable is used to store the active database of the DatabaseHandler object.

Definition at line 101 of file databasehandler.hpp.

8.9.2.2 activeTableName

```
QString AposDatabase::TableHandler::activeTableName = "userTable"  [private]
```

The active table name of the TableHandler object.

This variable is used to store the active table name of the TableHandler object.

Definition at line 141 of file tablehandler.hpp.

8.9.2.3 clearCommand

```
bool AposFrontend::DevWindow::clearCommand = false  [private]
```

Boolean values for clearing the command and the inputs.

These boolean values are used to determine whether to clear the command, and the inputs of the DevWindow object.

Definition at line 362 of file devwindow.hpp.

8.9.2.4 databasePath

```
QString AposDatabase::DatabaseHandler::databasePath = R"(C:\Users\Clean\Documents\Projekte\↩
Apos-DatabaseManager\Project\resources\defaultDatabase\userDatabase.db)"  [private]
```

The path to the database.

This variable is used to store the path to the database.

Definition at line 115 of file databasehandler.hpp.

8.9.2.5 input1

```
QString AposFrontend::DevWindow::input1  [private]
```

String values for the inputs.

These string values are used to store the inputs of the DevWindow object.

Definition at line 353 of file devwindow.hpp.

8.9.2.6 languageChanged

```
bool AposFrontend::SettingsWindow::languageChanged {}  [private]
```

Indicates whether the language has changed.

This variable is used to indicate whether the language has changed.

Definition at line 158 of file settingswindow.hpp.

8.9.2.7 languageIndex

```
int AposFrontend::SettingsWindow::languageIndex {}  [private]
```

The index of the selected language.

This variable is used to store the index of the selected language.

Definition at line 144 of file settingswindow.hpp.

8.9.2.8 lastSqlError

```
QSqlError AposDatabase::DatabaseHandler::lastSqlError  [private]
```

The SQL error of the DatabaseHandler object.

This variable is used to store the SQL error of the DatabaseHandler object.

Definition at line 94 of file databasehandler.hpp.

8.9.2.9 lastTableError

```
QSqlError AposDatabase::TableHandler::lastTableError  [private]
```

The last table error of the TableHandler object.

This variable is used to store the last table error of the TableHandler object.

Definition at line 162 of file tablehandler.hpp.

8.9.2.10 objectHandler

```
QSharedPointer<AposBackend::ObjectHandler> AposFrontend::LauncherWindow::objectHandler =
nullptr  [private]
```

Shared pointer to the ObjectHandler object.

This shared pointer is used to access the ObjectHandler object.

Definition at line 117 of file launcherwindow.hpp.

8.9.2.11 ptrActiveDatabase

```
QSharedPointer<QSqlDatabase> AposDatabase::DatabaseHandler::ptrActiveDatabase  [private]
```

Shared pointer to the active database.

This shared pointer is used to access the active database of the DatabaseHandler object.

Definition at line 108 of file databasehandler.hpp.

8.9.2.12 ptrApplication [1/2]

```
QSharedPointer<QApplication> AposBackend::ObjectHandler::ptrApplication  [private]
```

Shared pointer to the QApplication object.

This shared pointer is used to access the QApplication object.

Definition at line 140 of file objecthandler.hpp.

8.9.2.13 ptrApplication [2/2]

```
QSharedPointer<QApplication> AposBackend::StartupHandler::ptrApplication  [private]
```

Shared pointer to the QApplication object.

This shared pointer is used to access the QApplication object.

Definition at line 95 of file startuphandler.hpp.

8.9.2.14 ptrDbHandler [1/2]

```
QSharedPointer<AposDatabase::DatabaseHandler> AposBackend::ObjectHandler::ptrDbHandler  [private]
```

Shared pointer to the DatabaseHandler object.

This shared pointer is used to access the DatabaseHandler object.

Definition at line 147 of file objecthandler.hpp.

8.9.2.15 ptrDbHandler [2/2]

```
QSharedPointer<DatabaseHandler> AposDatabase::TableHandler::ptrDbHandler  [private]
```

Shared pointer to the DatabaseHandler object.

This shared pointer is used to access the DatabaseHandler object.

Definition at line 148 of file tablehandler.hpp.

8.9.2.16 ptrDevWindow

```
QSharedPointer<DevWindow> AposFrontend::WindowHandler::ptrDevWindow  [private]
```

Shared pointer to the DevWindow object.

This shared pointer is used to access the DevWindow object.

Definition at line 105 of file windowhandler.hpp.

8.9.2.17 ptrLauncherWindow

```
QSharedPointer<LauncherWindow> AposFrontend::WindowHandler::ptrLauncherWindow  [private]
```

Shared pointer to the LauncherWindow object.

This shared pointer is used to access the LauncherWindow object.

Definition at line 98 of file windowhandler.hpp.

8.9.2.18 ptrObjectHandler [1/4]

```
QSharedPointer<ObjectHandler> AposBackend::StartupHandler::ptrObjectHandler  [private]
```

Shared pointer to the ObjectHandler object.

This shared pointer is used to access the ObjectHandler object.

Definition at line 102 of file startuphandler.hpp.

8.9.2.19 ptrObjectHandler [2/4]

```
QSharedPointer<AposBackend::ObjectHandler> AposFrontend::DevWindow::ptrObjectHandler = nullptr
[private]
```

Shared pointer to the ObjectHandler object.

This shared pointer is used to access the ObjectHandler object.

Definition at line 344 of file devwindow.hpp.

8.9.2.20 ptrObjectHandler [3/4]

```
QSharedPointer<AposBackend::ObjectHandler> AposFrontend::SettingsWindow::ptrObjectHandler
[private]
```

Shared pointer to the ObjectHandler object.

This shared pointer is used to access the ObjectHandler object.

Definition at line 130 of file settingswindow.hpp.

8.9.2.21 ptrObjectHandler [4/4]

```
QSharedPointer<AposBackend::ObjectHandler> AposFrontend::WindowHandler::ptrObjectHandler
[private]
```

Shared pointer to the ObjectHandler object.

This shared pointer is used to access the ObjectHandler object.

Definition at line 119 of file windowhandler.hpp.

8.9.2.22 ptrSettingsWindow

```
QSharedPointer<SettingsWindow> AposFrontend::WindowHandler::ptrSettingsWindow  [private]
```

Shared pointer to the SettingsWindow object.

This shared pointer is used to access the SettingsWindow object.

Definition at line 112 of file windowhandler.hpp.

8.9.2.23 ptrTableHandler

```
QSharedPointer<AposDatabase::TableHandler> AposBackend::ObjectHandler::ptrTableHandler  [private]
```

Shared pointer to the TableHandler object.

This shared pointer is used to access the TableHandler object.

Definition at line 154 of file objecthandler.hpp.

8.9.2.24 ptrTableModel

```
QSharedPointer<QSqlTableModel> AposDatabase::TableHandler::ptrTableModel  [private]
```

Shared pointer to the QSqlTableModel object.

This shared pointer is used to access the QSqlTableModel object.

Definition at line 155 of file tablehandler.hpp.

8.9.2.25 ptrTranslator

```
QSharedPointer<QTranslator> AposFrontend::SettingsWindow::ptrTranslator  [private]
```

Shared pointer to the QTranslator object.

This shared pointer is used to access the QTranslator object.

Definition at line 137 of file settingswindow.hpp.

8.9.2.26 tempLanguageIndex

```
int AposFrontend::SettingsWindow::tempLanguageIndex {}  [private]
```

The temporary index of the selected language.

This variable is used to store the temporary index of the selected language.

Definition at line 151 of file settingswindow.hpp.

8.9.2.27 ui [1/3]

```
Ui::DevWindow* AposFrontend::DevWindow::ui  [private]
```

Pointer to the user interface of the DevWindow object.

This pointer is used to access the user interface of the DevWindow object.

Definition at line 335 of file devwindow.hpp.

8.9.2.28 ui [2/3]

```
Ui::LauncherWindow* AposFrontend::LauncherWindow::ui  [private]
```

Pointer to the user interface of the LauncherWindow object.

This pointer is used to access the user interface of the LauncherWindow object.

Definition at line 110 of file launcherwindow.hpp.

8.9.2.29 ui [3/3]

```
Ui::SettingsWindow* AposFrontend::SettingsWindow::ui  [private]
```

Pointer to the user interface of the SettingsWindow object.

This pointer is used to access the user interface of the SettingsWindow object.

Definition at line 123 of file settingswindow.hpp.

Chapter 9

Namespace Documentation

9.1 AposBackend Namespace Reference

Classes

- class ObjectHandler

  The ObjectHandler class is a part of the application's backend logic.
- class StartupHandler

  Provides the functionality for initializing the application's translator and ObjectHandler.

9.2 AposDatabase Namespace Reference

Classes

- class DatabaseHandler

  Provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.
- class TableHandler

  Provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

9.3 AposFrontend Namespace Reference

Classes

- class DevWindow

  Provides the user interface for the developer window.
- class LauncherWindow

  Provides the user interface for the launcher window.
- class SettingsWindow

  Provides the user interface for the settings window.
- class TranslatableWindow

  An abstract base class that provides a function for retranslating the user interface.
- class WindowHandler

  Provides the functionality for managing the application's windows.

9.4 AppInitialization Namespace Reference

Functions

- QSharedPointer< AposBackend::StartupHandler > initializeStartupHandler (const QSharedPointer< QApplication > &newApp)

  Initialize the StartupHandler object.
- QSharedPointer< AposBackend::ObjectHandler > initializeObjectHandler (const QSharedPointer< AposBackend::StartupHandler > &startupHandler)

  Initialize the ObjectHandler object.
- QSharedPointer< AposFrontend::WindowHandler > initializeWindowHandler (const QSharedPointer< AposBackend::ObjectHandler > &objectHandler)

  Initialize the WindowHandler object.

9.4.1 Detailed Description

< Include the StartupHandler class < Include the WindowHandler class < Include the QApplication class < Include the QDebug class for debugging < Include the QScopedPointer class for memory management

9.4.2 Function Documentation

9.4.2.1 initializeObjectHandler()

```
QSharedPointer< AposBackend::ObjectHandler > AppInitialization::initializeObjectHandler (
            const QSharedPointer< AposBackend::StartupHandler > & startupHandler )
```

Initialize the ObjectHandler object.

This function initializes the StartupHandler object with the QApplication object, then uses the StartupHandler object to initialize and return the ObjectHandler object.

Parameters

startupHandler	Pointer to the StartupHandler object. This is used to initialize the ObjectHandler object.

Returns

Unique pointer to the initialized ObjectHandler object

Exceptions

std::runtime_error	if the QApplication pointer is null or if the ObjectHandler fails to initialize.

Definition at line 131 of file main.cpp.

Here is the caller graph for this function:

9.4.2.2 initializeStartupHandler()

```
QSharedPointer< AposBackend::StartupHandler > AppInitialization::initializeStartupHandler (
            const QSharedPointer< QApplication > & newApp )
```

Initialize the StartupHandler object.

This function initializes the StartupHandler object with the QApplication object.

Parameters

newApp	Shared pointer to the QApplication object. This is used to initialize the StartupHandler object.

Returns

Shared pointer to the initialized StartupHandler object

Exceptions

std::runtime_error	if the QApplication pointer is null.

Definition at line 120 of file main.cpp.

Here is the caller graph for this function:

9.4.2.3 initializeWindowHandler()

```
QSharedPointer< AposFrontend::WindowHandler > AppInitialization::initializeWindowHandler (
            const QSharedPointer< AposBackend::ObjectHandler > & objectHandler )
```

Initialize the WindowHandler object.

This function initializes the WindowHandler object with the ObjectHandler object and shows the launch window.

Parameters

objectHandler	Unique pointer to the ObjectHandler object. This is used to initialize the WindowHandler object.

Returns

Unique pointer to the initialized WindowHandler object

Exceptions

std::runtime_error	if the ObjectHandler pointer is null, or if the WindowHandler object fails to initialize.

Definition at line 143 of file main.cpp.

Here is the caller graph for this function:

9.5 Ui Namespace Reference

Chapter 10

Class Documentation

10.1 AposBackend::ObjectHandler Class Reference

The ObjectHandler class is a part of the application's backend logic.

`#include <objecthandler.hpp>`

Public Member Functions

- ObjectHandler (QSharedPointer< QApplication > newApplication, QSharedPointer< AposDatabase::DatabaseHandler > newDbHandler, QSharedPointer< AposDatabase::TableHandler > newTableHandler)

  Constructor for the ObjectHandler class.
- bool initDatabaseObject ()

  Initializes the database object.
- bool initTableObject (const QString &inputTableName)

  Initializes the table object.
- void setActiveTableName (const QString &newActiveTableName)

  Sets the active table name.
- QSharedPointer< QSqlDatabase > getActiveDatabase () const

  Gets the active database.
- QSharedPointer< AposDatabase::TableHandler > getPtrTableHandler () const

  Gets the table handler.
- QSharedPointer< AposDatabase::DatabaseHandler > getPtrDbHandler () const

  Gets the database handler.
- const QString & getActiveTableName () const

  Gets the active table name.
- const QSqlError & getTableSqlError () const

  Gets the table SQL error.
- const QSharedPointer< QApplication > & getPtrApplication () const

  Gets the QApplication object.

Private Attributes

- QSharedPointer< QApplication > ptrApplication

  Shared pointer to the QApplication object.
- QSharedPointer< AposDatabase::DatabaseHandler > ptrDbHandler

  Shared pointer to the DatabaseHandler object.
- QSharedPointer< AposDatabase::TableHandler > ptrTableHandler

  Shared pointer to the TableHandler object.

10.1.1 Detailed Description

The ObjectHandler class is a part of the application's backend logic.

Provides the functionality for initializing the database and table objects, setting the active table name, and getting the active database, table handler, database handler, active table name, and table SQL error.

See also

AposDatabase::DatabaseHandler

AposDatabase::TableHandler

QApplication

QSharedPointer

QDebug

QtSql

Definition at line 48 of file objecthandler.hpp.

10.1.2 Member Function Documentation

10.1.2.1 getActiveDatabase()

```
QSharedPointer< QSqlDatabase > AposBackend::ObjectHandler::getActiveDatabase ( ) const
```

Gets the active database.

This function gets the active database of the ObjectHandler object.

Returns

Shared pointer to the active QSqlDatabase object.

Definition at line 70 of file objecthandler.cpp.

10.1.2.2 getActiveTableName()

```
const QString & AposBackend::ObjectHandler::getActiveTableName ( ) const
```

Gets the active table name.

This function gets the active table name of the ObjectHandler object.

Returns

The active table name.

Definition at line 62 of file objecthandler.cpp.

10.1.2.3 getPtrApplication()

```
const QSharedPointer< QApplication > & AposBackend::ObjectHandler::getPtrApplication ( ) const
```

Gets the QApplication object.

This function gets the QApplication object of the ObjectHandler object.

Returns

Shared pointer to the QApplication object.

Definition at line 78 of file objecthandler.cpp.

10.1.2.4 getPtrDbHandler()

```
QSharedPointer< AposDatabase::DatabaseHandler > AposBackend::ObjectHandler::getPtrDbHandler (
) const
```

Gets the database handler.

This function gets the database handler of the ObjectHandler object.

Returns

Shared pointer to the DatabaseHandler object.

Definition at line 58 of file objecthandler.cpp.

10.1.2.5 getPtrTableHandler()

```
QSharedPointer< AposDatabase::TableHandler > AposBackend::ObjectHandler::getPtrTableHandler (
) const
```

Gets the table handler.

This function gets the table handler of the ObjectHandler object.

Returns

Shared pointer to the TableHandler object.

Definition at line 54 of file objecthandler.cpp.

10.1.2.6 getTableSqlError()

```
const QSqlError & AposBackend::ObjectHandler::getTableSqlError ( ) const
```

Gets the table SQL error.

This function gets the table SQL error of the ObjectHandler object.

Returns

The table SQL error.

Definition at line 66 of file objecthandler.cpp.

10.1.2.7 setActiveTableName()

```
void AposBackend::ObjectHandler::setActiveTableName (
            const QString & newActiveTableName )
```

Sets the active table name.

This function sets the active table name of the ObjectHandler object.

Parameters

newActiveTableName	The new active table name.

Definition at line 74 of file objecthandler.cpp.

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

- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/backendClasses/objecthandler.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/backendClasses/objecthandler.cpp

10.2 AposBackend::StartupHandler Class Reference

Provides the functionality for initializing the application's translator and ObjectHandler.

```
#include <startuphandler.hpp>
```

Public Member Functions

- StartupHandler (const QSharedPointer< QApplication > &application)

  Constructor for the StartupHandler class.
- QSharedPointer< ObjectHandler > startUp ()

  Initializes the application's translator and ObjectHandler.

Private Member Functions

- void installTranslator ()

  Installs the application's translator.
- QSharedPointer< ObjectHandler > initObjectHandler ()

  Initializes the application's ObjectHandler.

Static Private Member Functions

- static QSharedPointer< QTranslator > initTranslator ()

  Initializes the application's translator.

Private Attributes

- QSharedPointer< QApplication > ptrApplication

  Shared pointer to the QApplication object.
- QSharedPointer< ObjectHandler > ptrObjectHandler

  Shared pointer to the ObjectHandler object.

10.2.1 Detailed Description

Provides the functionality for initializing the application's translator and ObjectHandler.

The StartupHandler class is a part of the application's backend logic. It interacts with the QApplication object and uses the ObjectHandler class to manage the application's objects.

See also

ObjectHandler
QApplication
QSharedPointer
QDebug
QtSql
QTranslator
QLocale

Definition at line 49 of file startuphandler.hpp.

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

- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/backendClasses/startuphandler.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/backendClasses/startuphandler.cpp

10.3 AposDatabase::DatabaseHandler Class Reference

Provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.

```
#include <databasehandler.hpp>
```

Public Member Functions

- DatabaseHandler ()

  Constructor for the DatabaseHandler class.
- bool initDatabase ()

  Initializes the database.
- void closeDatabase ()

  Closes the database.
- bool executeCommand (const QString &command)

  Executes a SQL command.
- QSharedPointer< QSqlDatabase > getActiveDatabase ()

  Gets the active database.
- const QSqlError & getSqlError () const

  Gets the SQL error.

Private Attributes

- QSqlError lastSqlError

  The SQL error of the DatabaseHandler object.
- QSqlDatabase activeDatabase

  The active database of the DatabaseHandler object.
- QSharedPointer< QSqlDatabase > ptrActiveDatabase

  Shared pointer to the active database.
- QString databasePath = R"(C:\Users\Clean\Documents\Projekte\Apos-DatabaseManager\Project\resources\default↩ Database\userDatabase.db)"

  The path to the database.

10.3.1 Detailed Description

Provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.

The DatabaseHandler class is a part of the application's backend logic.

See also

QtSql

QSqlDatabase

QSqlError

QSharedPointer

QDebug

Definition at line 40 of file databasehandler.hpp.

10.3.2 Member Function Documentation

10.3.2.1 getActiveDatabase()

```
QSharedPointer< QSqlDatabase > AposDatabase::DatabaseHandler::getActiveDatabase ( )
```

Gets the active database.

This function gets the active database of the DatabaseHandler object.

Returns

Shared pointer to the active QSqlDatabase object.

Definition at line 52 of file databasehandler.cpp.

10.3.2.2 getSqlError()

```
const QSqlError & AposDatabase::DatabaseHandler::getSqlError ( ) const
```

Gets the SQL error.

This function gets the SQL error of the DatabaseHandler object.

Returns

The SQL error.

Definition at line 56 of file databasehandler.cpp.

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

- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/databaseClasses/databasehandler.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/databaseClasses/databasehandler.cpp

10.4 AposDatabase::TableHandler Class Reference

Provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

```
#include <tablehandler.hpp>
```

Public Member Functions

- TableHandler (QSharedPointer< DatabaseHandler > newDbHandler)

  Constructor for the TableHandler class.
- TableHandler (QSharedPointer< DatabaseHandler > newDbHandler, const QString &tableName)

  Constructor for the TableHandler class.
- ~TableHandler ()

  Destructor for the TableHandler class.
- void generateTableModel ()

  Generates a table model.
- void generateTableModel (const QString &tableName)

  Generates a table model with a specified table name.
- bool insertIntoTable (const QString &tableName, const QString &value1, const QString &value2, const QString &value3, const QString &value4, const QString &value5)

  Inserts into a table.
- void setActiveTableName (const QString &newActiveTableName)

  Sets the active table name.
- QSharedPointer< QSqlTableModel > getTableModel ()

  Gets the table model.
- const QString & getActiveTableName () const

  Gets the active table name.
- const QSqlError & getLastTableError () const

  Gets the last table error.

Private Attributes

- QString activeTableName = "userTable"

  The active table name of the TableHandler object.
- QSharedPointer< DatabaseHandler > ptrDbHandler

  Shared pointer to the DatabaseHandler object.
- QSharedPointer< QSqlTableModel > ptrTableModel

  Shared pointer to the QSqlTableModel object.
- QSqlError lastTableError

  The last table error of the TableHandler object.

10.4.1 Detailed Description

Provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

The TableHandler class is a part of the application's backend logic.

See also

DatabaseHandler

QSharedPointer

QDebug

QtSql

QSqlTableModel

QSqlError

QSqlQuery

QSqlDatabase

Definition at line 48 of file tablehandler.hpp.

10.4.2 Member Function Documentation

10.4.2.1 getActiveTableName()

```
const QString & AposDatabase::TableHandler::getActiveTableName ( ) const
```

Gets the active table name.

This function gets the active table name of the TableHandler object.

Returns

The active table name.

Definition at line 92 of file tablehandler.cpp.

10.4.2.2 getLastTableError()

```
const QSqlError & AposDatabase::TableHandler::getLastTableError ( ) const
```

Gets the last table error.

This function gets the last table error of the TableHandler object.

Returns

The last table error.

Definition at line 100 of file tablehandler.cpp.

10.4.2.3 getTableModel()

```
QSharedPointer< QSqlTableModel > AposDatabase::TableHandler::getTableModel ( )
```

Gets the table model.

This function gets the table model of the TableHandler object.

Returns

Shared pointer to the QSqlTableModel object.

Definition at line 104 of file tablehandler.cpp.

10.4.2.4 setActiveTableName()

```
void AposDatabase::TableHandler::setActiveTableName (
            const QString & newActiveTableName )
```

Sets the active table name.

This function sets the active table name of the TableHandler object.

Parameters

newActiveTableName	The new active table name.

Definition at line 96 of file tablehandler.cpp.

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

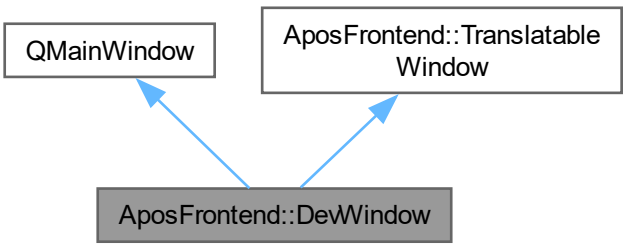
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/databaseClasses/tablehandler.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/databaseClasses/tablehandler.cpp

10.5 AposFrontend::DevWindow Class Reference

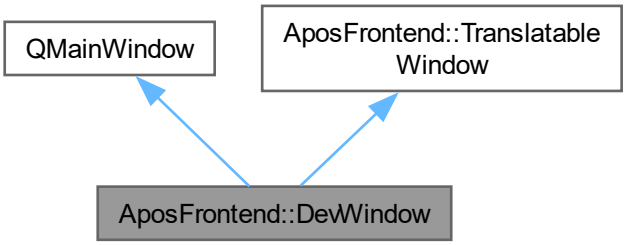
Provides the user interface for the developer window.

`#include <devwindow.hpp>`

Inheritance diagram for AposFrontend::DevWindow:



Collaboration diagram for AposFrontend::DevWindow:



Signals

- void returnToLauncher ()

  Signal for returning to the launcher.
- void openSettings ()

  Signal for opening the settings.

Public Member Functions

- DevWindow (QWidget *parent=nullptr, QSharedPointer< AposBackend::ObjectHandler > objectHandler=nullptr)

  Constructor for the DevWindow class.
- ~DevWindow () override

  Destructor for the DevWindow class.
- void logEvent (const QString &type, const QString &message)

  Logs an event with a type and a message.
- void logEvent (const QString &message, const QSqlError &error)

  Logs an event with a message and a SQL error.
- void logEvent (const QString &message)

  Logs an event with a message.
- void retranslateUi () override

  Retranslates the user interface.

Public Member Functions inherited from AposFrontend::TranslatableWindow

- TranslatableWindow ()

  Constructor for the TranslatableWindow class.

Private Slots

- void initDbClicked ()

  Slot for the 'InitDB' button click event.
- void closeDBClicked ()

  Slot for the 'CloseDB' button click event.
- void executeClicked ()

  Slot for the 'Execute' button click event.
- void selectTableClicked ()

  Slot for the 'SelectTable' button click event.
- void addValuesClicked ()

  Slot for the 'Add' button click event.
- void updateTableClicked ()

  Slot for the 'Update' button click event.
- void clearCommandAfterExecuteStateChanged (int arg1)

  Slot for the 'clearCommandAfterExecute' state change event.
- void clearInputsAfterInsertStateChanged (int arg1)

  Slot for the 'clearInputsAfterInsert' state change event.
- void returnToLauncherClicked ()

  Slot for the 'ReturnToLauncher' button click event.
- void settingsClicked ()

  Slot for the 'Settings' button click event.

Private Member Functions

- bool devConnectUi ()
- void initDatabase ()

  Initializes the database.
- void closeDatabase (const QSharedPointer< AposDatabase::DatabaseHandler > &db)

  Closes the database.
- void setModelViews ()

  Sets the model views.
- void setModelViews (const QSharedPointer< QSqlTableModel > &tableModel)

  Sets the model views with a table model.
- void enableButtons (bool databaseLoaded)

  Enables or disables the buttons.
- void assignInputs ()

  Assigns the inputs.
- void clearInputs (bool clearBool)

  Clears the inputs.
- bool checkCheckbox (int argCb)

  Checks a checkbox.
- void clearCommandBox (bool clearBool)

  Clears the command box.

Private Attributes

- Ui::DevWindow * ui

  Pointer to the user interface of the DevWindow object.
- QSharedPointer< AposBackend::ObjectHandler > ptrObjectHandler = nullptr

  Shared pointer to the ObjectHandler object.
- QString input1

  String values for the inputs.
- QString input2
- QString input3
- QString input4
- QString input5
- bool clearCommand = false

  Boolean values for clearing the command and the inputs.
- bool clearInput = false

10.5.1 Detailed Description

Provides the user interface for the developer window.

The DevWindow class provides the functionality for logging events, retranslating the user interface, initializing and closing the database, setting model views, enabling buttons, assigning inputs, checking checkboxes, and clearing inputs and the command box.

See also

QMainWindow

TranslatableWindow

AposBackend::ObjectHandler

Definition at line 49 of file devwindow.hpp.

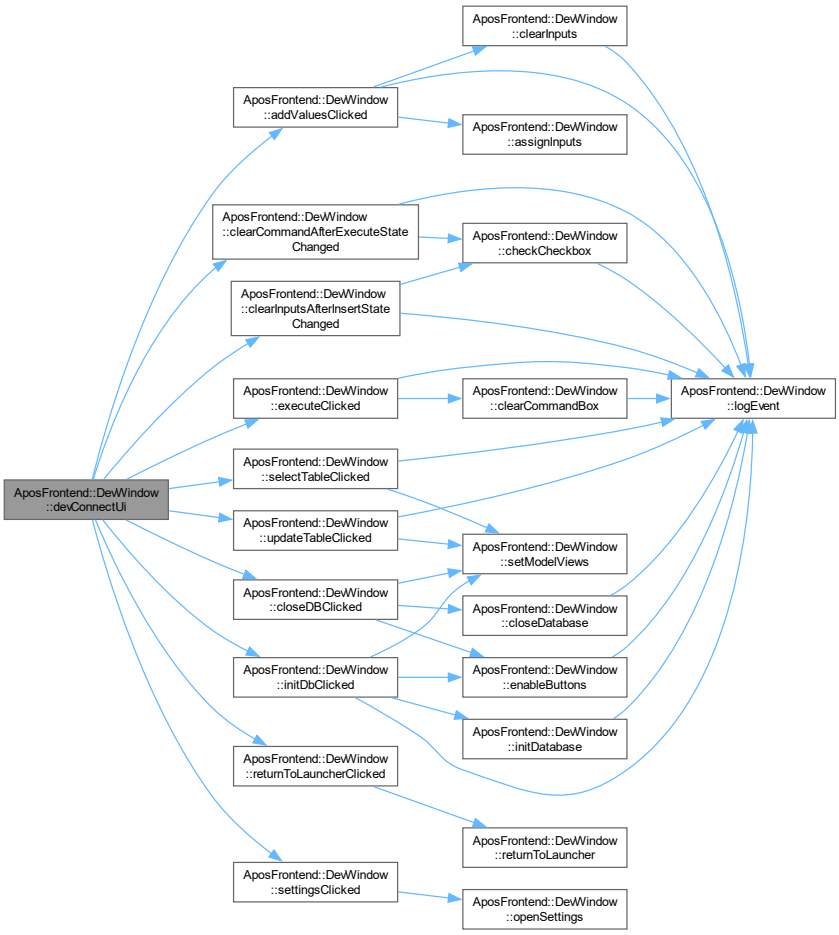
10.5.2 Member Function Documentation

10.5.2.1 devConnectUi()

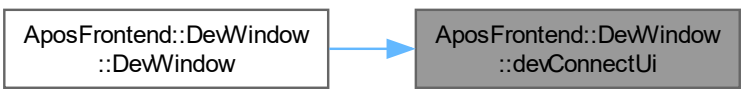
```
bool AposFrontend::DevWindow::devConnectUi ( )  [private]
```

Definition at line 44 of file devwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.2.2 logEvent()

```
void AposFrontend::DevWindow::logEvent (
            const QString & type,
            const QString & message )
```

Logs an event with a type and a message.

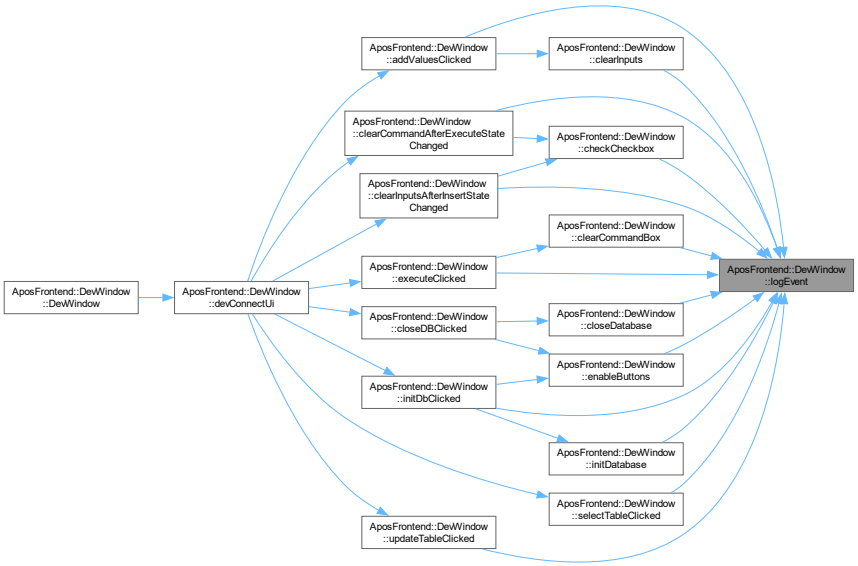
This function logs an event with a specified type and message.

Parameters

type	The type of the event.
message	The message of the event.

Definition at line 61 of file devwindow.cpp.

Here is the caller graph for this function:



10.5.3 Member Data Documentation

10.5.3.1 clearInput

```
bool AposFrontend::DevWindow::clearInput = false  [private]
```

Definition at line 362 of file devwindow.hpp.

10.5.3.2 input2

```
QString AposFrontend::DevWindow::input2  [private]
```

Definition at line 353 of file devwindow.hpp.

10.5.3.3 input3

```
QString AposFrontend::DevWindow::input3  [private]
```

Definition at line 353 of file devwindow.hpp.

10.5.3.4 input4

```
QString AposFrontend::DevWindow::input4  [private]
```

Definition at line 353 of file devwindow.hpp.

10.5.3.5 input5

```
QString AposFrontend::DevWindow::input5  [private]
```

Definition at line 353 of file devwindow.hpp.

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

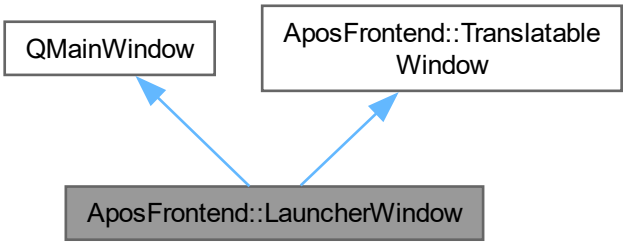
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/devwindow.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/devwindow.cpp

10.6 AposFrontend::LauncherWindow Class Reference

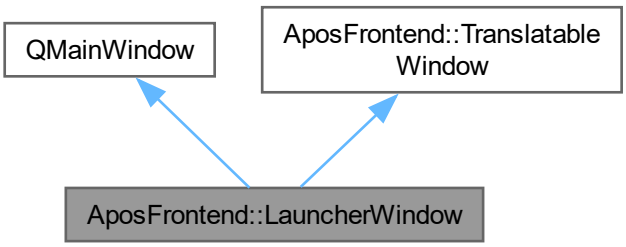
Provides the user interface for the launcher window.

```
#include <launcherwindow.hpp>
```

Inheritance diagram for AposFrontend::LauncherWindow:



Collaboration diagram for AposFrontend::LauncherWindow:



Signals

- void openDevWindow ()

  Signal for opening the developer window.
- void openSettings ()

  Signal for opening the settings.

Public Member Functions

- LauncherWindow (QWidget *parent=nullptr, QSharedPointer< AposBackend::ObjectHandler > newObjectHandler=nullptr)

  Constructor for the LauncherWindow class.
- ~LauncherWindow () override

  Destructor for the LauncherWindow class.
- void retranslateUi () override

  Retranslates the user interface.

Public Member Functions inherited from AposFrontend::TranslatableWindow

- TranslatableWindow ()

  Constructor for the TranslatableWindow class.

Private Slots

- void showDevClicked ()

  Slot for the 'ShowDev' button click event.
- void pushButtonClicked ()

  Slot for the 'PushButton' button click event.

Private Member Functions

- bool launcherConnectUi ()

Private Attributes

- Ui::LauncherWindow * ui

  Pointer to the user interface of the LauncherWindow object.
- QSharedPointer< AposBackend::ObjectHandler > objectHandler = nullptr

  Shared pointer to the ObjectHandler object.

10.6.1 Detailed Description

Provides the user interface for the launcher window.

The LauncherWindow class provides the functionality for opening the developer window and settings. It interacts with the QMainWindow and TranslatableWindow classes and uses the ObjectHandler class to manage the application's objects.

See also

AposBackend::ObjectHandler

TranslatableWindow

QMainWindow

Definition at line 45 of file launcherwindow.hpp.

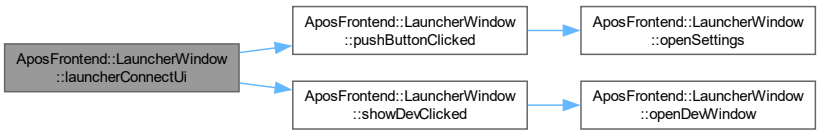
10.6.2 Member Function Documentation

10.6.2.1 launcherConnectUi()

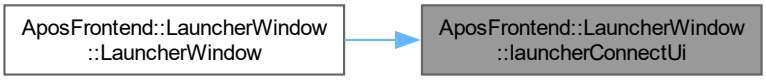
```
bool AposFrontend::LauncherWindow::launcherConnectUi ( )  [private]
```

Definition at line 39 of file launcherwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

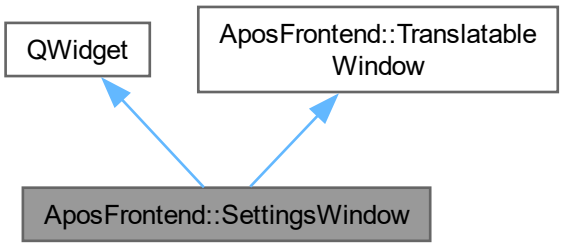
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/launcherwindow.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/launcherwindow.cpp

10.7 AposFrontend::SettingsWindow Class Reference

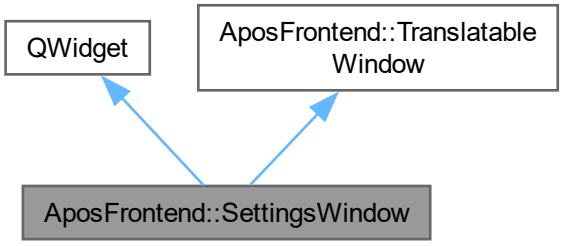
Provides the user interface for the settings window.

`#include <settingswindow.hpp>`

Inheritance diagram for AposFrontend::SettingsWindow:



Collaboration diagram for AposFrontend::SettingsWindow:



Signals

- void appliedSettings ()

  Signal for applying settings.

Public Member Functions

- SettingsWindow (QWidget *parent=nullptr, QSharedPointer< AposBackend::ObjectHandler > newObjectHandler=nullptr)

  Constructor for the SettingsWindow class.
- ~SettingsWindow () override

  Destructor for the SettingsWindow class.
- void retranslateUi () override

  Retranslates the user interface.

Public Member Functions inherited from AposFrontend::TranslatableWindow

- TranslatableWindow ()

  Constructor for the TranslatableWindow class.

Private Slots

- void closeClicked ()

  Slot for the 'Close' button click event.
- void applyClicked ()

  Slot for the 'Apply' button click event.
- void applyAndCloseClicked ()
- void languageCurrentIndexChanged (int index)

  Slot for the 'Language' combo box index change event.

Private Member Functions

- void settingsConnectUi ()
- void installTranslator ()

  Installs the application's translator.

Private Attributes

- Ui::SettingsWindow * ui

  Pointer to the user interface of the SettingsWindow object.
- QSharedPointer< AposBackend::ObjectHandler > ptrObjectHandler

  Shared pointer to the ObjectHandler object.
- QSharedPointer< QTranslator > ptrTranslator

  Shared pointer to the QTranslator object.
- int languageIndex {}

  The index of the selected language.
- int tempLanguageIndex {}

  The temporary index of the selected language.
- bool languageChanged {}

  Indicates whether the language has changed.

10.7.1 Detailed Description

Provides the user interface for the settings window.

The SettingsWindow class provides the functionality for applying settings and retranslating the user interface. It interacts with the QWidget and TranslatableWindow classes and uses the ObjectHandler class to manage the application's objects.

See also

AposBackend::ObjectHandler

TranslatableWindow

QWidget

QTranslator

QSharedPointer

Definition at line 49 of file settingswindow.hpp.

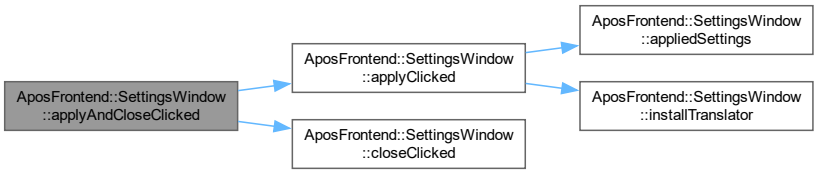
10.7.2 Member Function Documentation

10.7.2.1 applyAndCloseClicked

```
void AposFrontend::SettingsWindow::applyAndCloseClicked ( )  [private], [slot]
```

Definition at line 67 of file settingswindow.cpp.

Here is the call graph for this function:



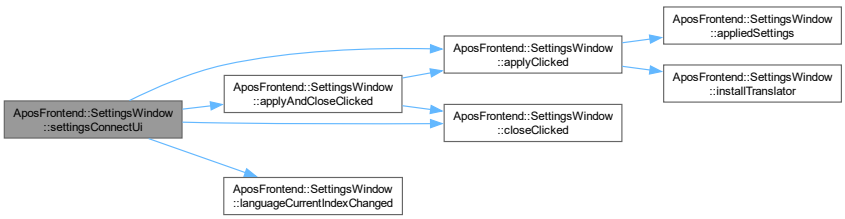
Here is the caller graph for this function:

10.7.2.2 settingsConnectUi()

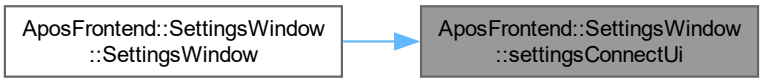
```
void AposFrontend::SettingsWindow::settingsConnectUi ( )  [private]
```

Definition at line 40 of file settingswindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

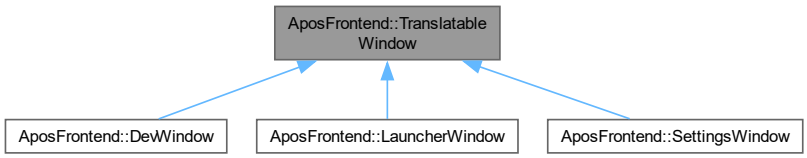
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/settingswindow.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/settingswindow.cpp

10.8 AposFrontend::TranslatableWindow Class Reference

An abstract base class that provides a function for retranslating the user interface.

`#include <translatablewindow.hpp>`

Inheritance diagram for AposFrontend::TranslatableWindow:



Public Member Functions

- TranslatableWindow ()

  Constructor for the TranslatableWindow class.
- virtual void retranslateUi ()=0

  Retranslates the user interface.

10.8.1 Detailed Description

An abstract base class that provides a function for retranslating the user interface.

The TranslatableWindow class is a part of the application's frontend logic. It is used as a base class for other classes that need to retranslate their user interface.

Definition at line 26 of file translatablewindow.hpp.

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

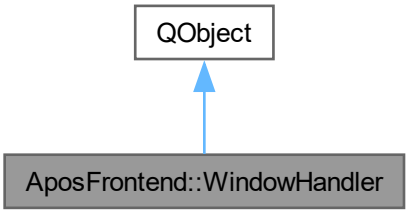
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/translatablewindow.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/translatablewindow.cpp

10.9 AposFrontend::WindowHandler Class Reference

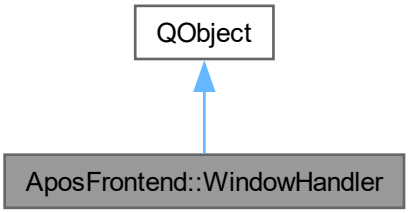
Provides the functionality for managing the application's windows.

`#include <windowhandler.hpp>`

Inheritance diagram for AposFrontend::WindowHandler:



Collaboration diagram for AposFrontend::WindowHandler:



Public Member Functions

- WindowHandler (QSharedPointer< AposBackend::ObjectHandler > newObjectHandler)
  Constructor for the WindowHandler class.
- void showLaunchWindow ()
  Shows the launcher window.

Private Slots

- void showDevWindow ()
  Shows the developer window.
- void showSettingsWindow ()
  Shows the settings window.
- void applySettings ()
  Applies settings.

Private Member Functions

- void changeLanguages ()

  Changes languages.

Private Attributes

- QSharedPointer< LauncherWindow > ptrLauncherWindow

  Shared pointer to the LauncherWindow object.
- QSharedPointer< DevWindow > ptrDevWindow

  Shared pointer to the DevWindow object.
- QSharedPointer< SettingsWindow > ptrSettingsWindow

  Shared pointer to the SettingsWindow object.
- QSharedPointer< AposBackend::ObjectHandler > ptrObjectHandler

  Shared pointer to the ObjectHandler object.

10.9.1 Detailed Description

Provides the functionality for managing the application's windows.

The WindowHandler class is a part of the application's frontend logic. It provides the functionality for showing the launcher window, developer window, and settings window. It also applies settings and changes languages.

See also

AposBackend::ObjectHandler

LauncherWindow

DevWindow

SettingsWindow

QSharedPointer

QObject

Definition at line 48 of file windowhandler.hpp.

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

- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/windowhandler.hpp
- C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/Project/classes/frontendClasses/windowhandler.cpp

Chapter 11

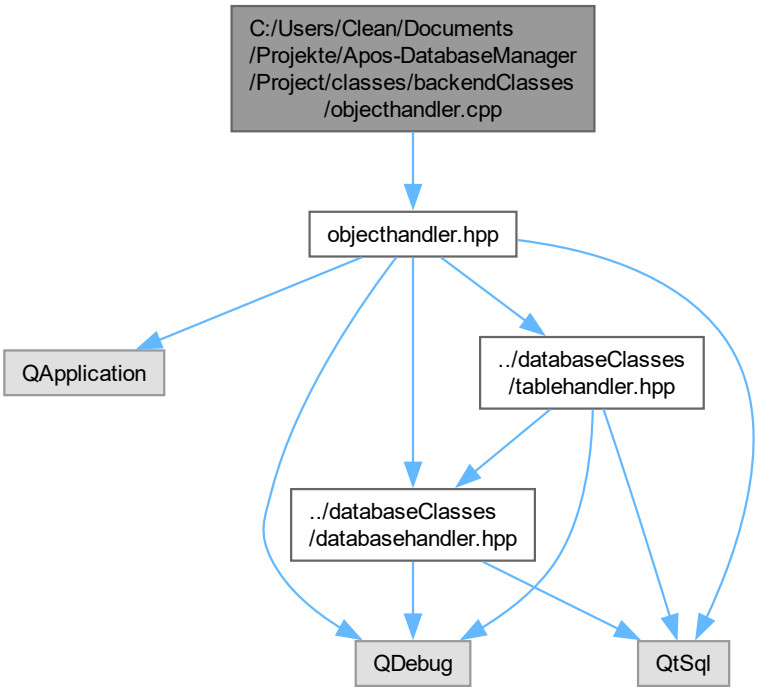
File Documentation

11.1 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/backendClasses/objecthandler.cpp File Reference

Source file for the ObjectHandler class.

```
#include "objecthandler.hpp"
```

Include dependency graph for objecthandler.cpp:



Namespaces

- namespace AposBackend

11.1.1 Detailed Description

Source file for the ObjectHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the ObjectHandler class, which is a part of the application's backend logic. The ObjectHandler class provides the functionality for initializing the database and table objects, setting the active table name, and getting the active database, table handler, database handler, active table name, and table SQL error.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposDatabase::DatabaseHandler

AposDatabase::TableHandler

QApplication

QSharedPointer

QDebug

QtSql

Definition in file objecthandler.cpp.

11.2 objecthandler.cpp

Go to the documentation of this file.

```
00001
00022 #include "objecthandler.hpp"
00023
00024 namespace AposBackend {
00025
      //-------------------------------------------------------------------------------------------------------------//
00026     ObjectHandler::ObjectHandler(QSharedPointer<QApplication> newApplication,
00027                                  QSharedPointer<AposDatabase::DatabaseHandler> newDbHandler,
00028                                  QSharedPointer<AposDatabase::TableHandler> newTableHandler) {
00029         ptrApplication = newApplication;
00030         ptrDbHandler = newDbHandler;
00031         ptrTableHandler = newTableHandler;
00032     }
00033
      //-------------------------------------------------------------------------------------------------------------//
00034     bool ObjectHandler::initDatabaseObject() {
00035         bool initializedDatabaseObject = false;
```

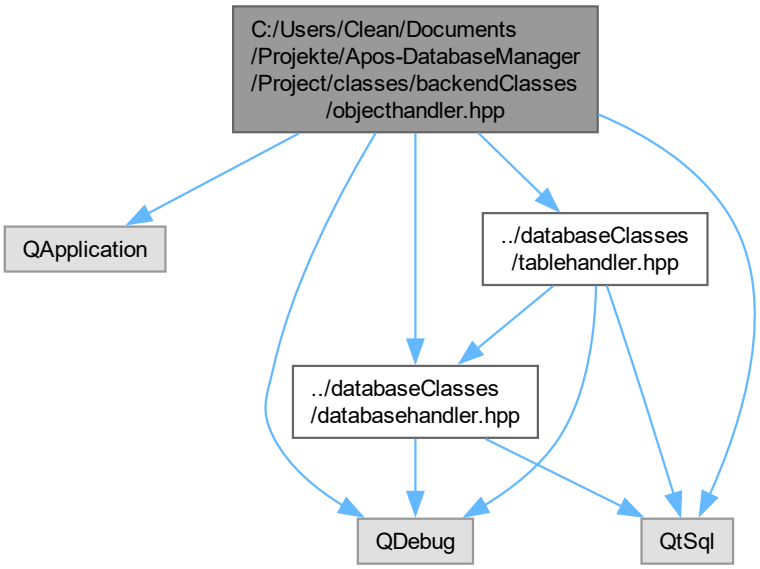
```
00036         initializedDatabaseObject = ptrDbHandler->initDatabase();
00037         return initializedDatabaseObject;
00038     }
00039
      //-----------------------------------------------------------------------------------------------------------//
00040     bool ObjectHandler::initTableObject(const QString &inputTableName) {
00041         bool initializedTableObject = false;
00042         try {
00043             ptrTableHandler->generateTableModel(inputTableName);
00044             initializedTableObject = true;
00045         }
00046         catch (const std::exception &e) {
00047             qDebug() « "Error: " « e.what();
00048             initializedTableObject = false;
00049         }
00050         return initializedTableObject;
00051
00052     }
00053
      //-----------------------------------------------------------------------------------------------------------//
00054     QSharedPointer<AposDatabase::TableHandler> ObjectHandler::getPtrTableHandler() const {
00055         return ptrTableHandler;
00056     }
00057
      //-----------------------------------------------------------------------------------------------------------//
00058     QSharedPointer<AposDatabase::DatabaseHandler> ObjectHandler::getPtrDbHandler() const {
00059         return ptrDbHandler;
00060     }
00061
      //-----------------------------------------------------------------------------------------------------------//
00062     const QString &ObjectHandler::getActiveTableName() const {
00063         return ptrTableHandler->getActiveTableName();
00064     }
00065
      //-----------------------------------------------------------------------------------------------------------//
00066     const QSqlError &ObjectHandler::getTableSqlError() const {
00067         return ptrTableHandler->getLastTableError();
00068     }
00069
      //-----------------------------------------------------------------------------------------------------------//
00070     [[maybe_unused]] QSharedPointer<QSqlDatabase> ObjectHandler::getActiveDatabase() const {
00071         return ptrDbHandler->getActiveDatabase();
00072     }
00073
      //-----------------------------------------------------------------------------------------------------------//
00074     [[maybe_unused]] void ObjectHandler::setActiveTableName(const QString &newActiveTableName) {
00075         ptrTableHandler->setActiveTableName(newActiveTableName);
00076     }
00077
      //-----------------------------------------------------------------------------------------------------------//
00078     const QSharedPointer<QApplication> &ObjectHandler::getPtrApplication() const {
00079         return ptrApplication;
00080     }
00081
00082 }
00083
00084
00085
00086
```

11.3 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/backendClasses/objecthandler.hpp File Reference

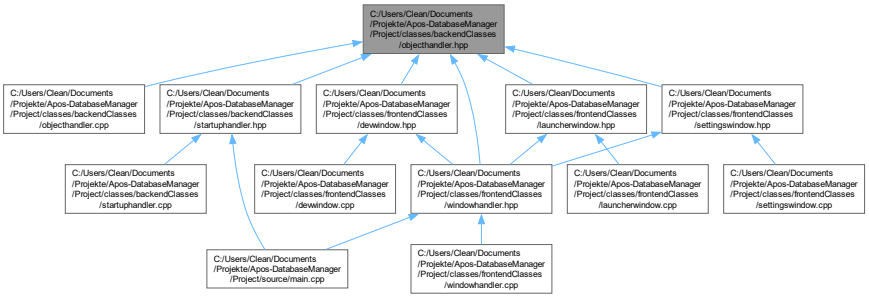
Header file for the ObjectHandler class.

```
#include <QApplication>
#include <QDebug>
#include <QtSql>
#include "../databaseClasses/databasehandler.hpp"
#include "../databaseClasses/tablehandler.hpp"
```

Include dependency graph for objecthandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposBackend::ObjectHandler

  The ObjectHandler class is a part of the application's backend logic.

Namespaces

- namespace AposBackend

11.3.1 Detailed Description

Header file for the ObjectHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the ObjectHandler class, which is a part of the application's backend logic. The ObjectHandler class provides the functionality for initializing the database and table objects, setting the active table name, and getting the active database, table handler, database handler, active table name, and table SQL error.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposDatabase::DatabaseHandler

AposDatabase::TableHandler

QApplication

QSharedPointer

QDebug

QtSql

Definition in file objecthandler.hpp.

11.4 objecthandler.hpp

Go to the documentation of this file.

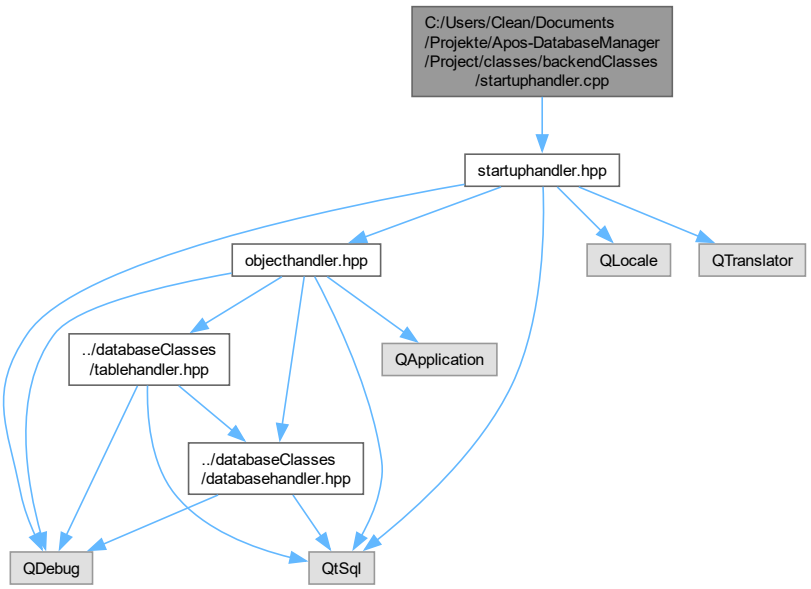
```
00001
00023 #pragma once
00024
00025 #include <QApplication>
00026 #include <QDebug>
00027 #include <QtSql>
00028
00029 #include "../databaseClasses/databasehandler.hpp"
00030 #include "../databaseClasses/tablehandler.hpp"
00031
00032
      //-----------------------------------------------------------------------------------------------------------/
00033 namespace AposBackend {
00048     class ObjectHandler {
00049     public:
00058         ObjectHandler(QSharedPointer<QApplication> newApplication,
00059                       QSharedPointer<AposDatabase::DatabaseHandler> newDbHandler,
00060                       QSharedPointer<AposDatabase::TableHandler> newTableHandler);
```

```
00061
00068         bool initDatabaseObject();
00069
00077         bool initTableObject(const QString &inputTableName);
00078
00085         [[maybe_unused]] void setActiveTableName(const QString &newActiveTableName);
00086
00093         [[maybe_unused]] [[nodiscard]] QSharedPointer<QSqlDatabase> getActiveDatabase() const;
00094
00101         [[nodiscard]] QSharedPointer<AposDatabase::TableHandler> getPtrTableHandler() const;
00102
00109         [[nodiscard]] QSharedPointer<AposDatabase::DatabaseHandler> getPtrDbHandler() const;
00110
00117         [[nodiscard]] const QString &getActiveTableName() const;
00118
00125         [[nodiscard]] const QSqlError &getTableSqlError() const;
00126
00133         [[nodiscard]] const QSharedPointer<QApplication> &getPtrApplication() const;
00134     private:
00140         QSharedPointer<QApplication> ptrApplication;
00141
00147         QSharedPointer<AposDatabase::DatabaseHandler> ptrDbHandler;
00148
00154         QSharedPointer<AposDatabase::TableHandler> ptrTableHandler;
00155     };
00156 }
```

11.5 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/backendClasses/startuphandler.cpp File Reference

Source file for the StartupHandler class.

`#include "startuphandler.hpp"`
Include dependency graph for startuphandler.cpp:



Namespaces

- namespace AposBackend

11.5.1 Detailed Description

Source file for the StartupHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the StartupHandler class, which is a part of the application's backend logic. The StartupHandler class provides the functionality for initializing the application's translator and ObjectHandler.

Note

The application is part of a student project and is not intended for commercial use.

See also

ObjectHandler

QApplication

QSharedPointer

QDebug

QtSql

QTranslator

QLocale

Definition in file startuphandler.cpp.

11.6 startuphandler.cpp

Go to the documentation of this file.

```
00001
00022 #include "startuphandler.hpp"
00023
00024 namespace AposBackend {
00025
      //-----------------------------------------------------------------------------------------------------------//
00026     StartupHandler::StartupHandler(const QSharedPointer<QApplication> &application) {
00027         if (application == nullptr) {
00028             throw std::runtime_error("QApplication pointer is null");
00029         }
00030         this->ptrApplication = application;
00031     }
00032
      //-----------------------------------------------------------------------------------------------------------//
00033     QSharedPointer<ObjectHandler> StartupHandler::startUp() {
00034         try {
00035             installTranslator();
00036             ptrObjectHandler = initObjectHandler();
```

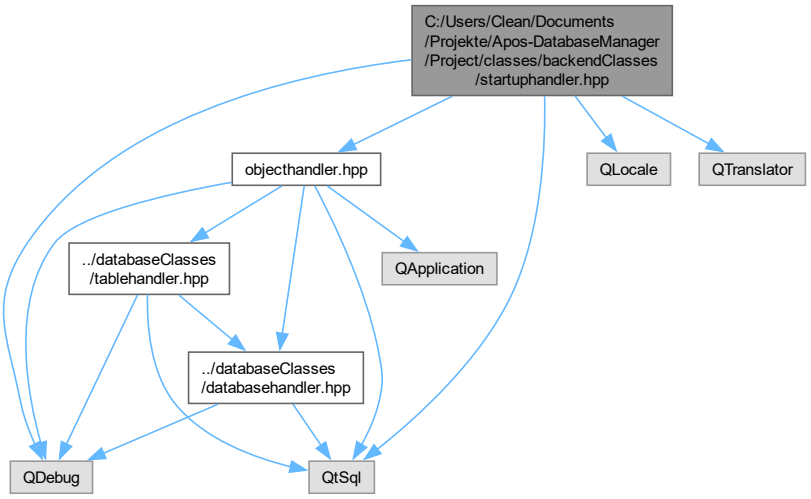
```
00037         } catch (const std::exception &e) {
00038             qDebug() « "Exception caught in StartupHandler::startUp: " « e.what();
00039             ptrObjectHandler = nullptr;
00040         }
00041         return ptrObjectHandler;
00042     }
00043
      //-----------------------------------------------------------------------------------------------------------//
00044     void StartupHandler::installTranslator() {
00045         QSharedPointer<QTranslator> translator = initTranslator();
00046         if (ptrApplication->installTranslator(translator.data())) {
00047             qDebug() « "Translator installed";
00048         }
00049     }
00050
      //-----------------------------------------------------------------------------------------------------------//
00051     QSharedPointer<QTranslator> StartupHandler::initTranslator() {
00052         QSharedPointer<QTranslator> translator(new QTranslator());
00053         const QStringList uiLanguages = QLocale::system().uiLanguages();
00054         for (const QString &locale: uiLanguages) {
00055             const QString baseName = "Apos-DatabaseManager_" + QLocale(locale).name();
00056             if (translator->load(":/i18n/" + baseName)) {
00057                 break;
00058             }
00059         }
00060         if (translator->isEmpty()) {
00061             throw std::runtime_error("Failed to load translator");
00062         }
00063         return translator;
00064     }
00065
      //-----------------------------------------------------------------------------------------------------------//
00066     QSharedPointer<ObjectHandler> StartupHandler::initObjectHandler() {
00067         QSharedPointer<AposDatabase::DatabaseHandler> dbHandler(new AposDatabase::DatabaseHandler());
00068         QSharedPointer<AposDatabase::TableHandler> tableHandler(new
      AposDatabase::TableHandler(dbHandler));
00069         if (dbHandler == nullptr || tableHandler == nullptr) {
00070             throw std::runtime_error("Failed to initialize DatabaseHandler or TableHandler");
00071         }
00072         return QSharedPointer<ObjectHandler>(new ObjectHandler(ptrApplication, dbHandler,
      tableHandler));
00073     }
00074
00075 }
```

11.7 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/backendClasses/startuphandler.hpp File Reference

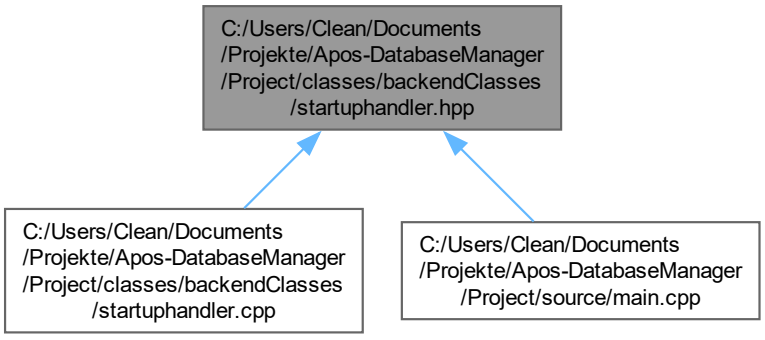
Header file for the StartupHandler class.

```
#include <QDebug>
#include <QtSql>
#include <QLocale>
#include <QTranslator>
#include "objecthandler.hpp"
```

Include dependency graph for startuphandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposBackend::StartupHandler

  Provides the functionality for initializing the application's translator and ObjectHandler.

Namespaces

- namespace AposBackend

11.7.1 Detailed Description

Header file for the StartupHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the StartupHandler class, which is a part of the application's backend logic. The StartupHandler class provides the functionality for initializing the application's translator and ObjectHandler.

Note

The application is part of a student project and is not intended for commercial use.

See also

ObjectHandler
QApplication
QSharedPointer
QDebug
QtSql
QTranslator
QLocale

Definition in file startuphandler.hpp.

11.8 startuphandler.hpp

Go to the documentation of this file.

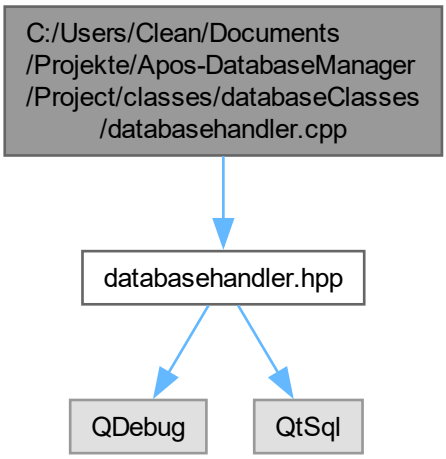
```
00001
00022 #pragma once
00023
00024 #include <QDebug>
00025 #include <QtSql>
00026 #include <QLocale>
00027 #include <QTranslator>
00028
00029 #include "objecthandler.hpp"
00030
00031
      //-----------------------------------------------------------------------------------------------------------------/
00032 namespace AposBackend {
00049     class StartupHandler {
00050     public:
00057         explicit StartupHandler(const QSharedPointer<QApplication> &application);
00058
00065         QSharedPointer<ObjectHandler> startUp();
00066     private:
00073         static QSharedPointer<QTranslator> initTranslator();
00074
00080         void installTranslator();
00081
00088         QSharedPointer<ObjectHandler> initObjectHandler();
00089
00095         QSharedPointer<QApplication> ptrApplication;
00096
00102         QSharedPointer<ObjectHandler> ptrObjectHandler;
00103     };
00104 }
```

11.9 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/databaseClasses/databasehandler.cpp File Reference

Source file for the DatabaseHandler class.

```
#include "databasehandler.hpp"
```

Include dependency graph for databasehandler.cpp:



Namespaces

- namespace AposDatabase

11.9.1 Detailed Description

Source file for the DatabaseHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the DatabaseHandler class, which is a part of the application's backend logic. The DatabaseHandler class provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.

Note

The application is part of a student project and is not intended for commercial use.

See also

QtSql

QSqlDatabase

QSqlError

QSharedPointer

QDebug

Definition in file databasehandler.cpp.

11.10 databasehandler.cpp

Go to the documentation of this file.

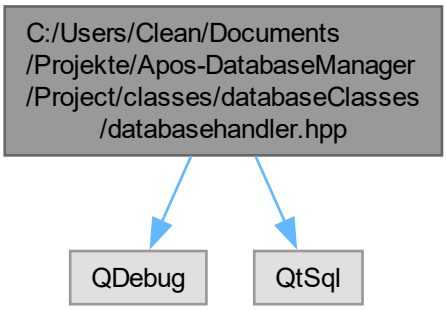
```
00001
00020 #include "databasehandler.hpp"
00021
00022 namespace AposDatabase {
00023
      //-------------------------------------------------------------------------------------------------------------------//
00024     DatabaseHandler::DatabaseHandler() = default;
00025
      //-------------------------------------------------------------------------------------------------------------------//
00026     bool DatabaseHandler::initDatabase() {
00027
00028         activeDatabase = QSqlDatabase::addDatabase("QSQLITE", "db1");
00029         activeDatabase.setDatabaseName(databasePath);
00030         ptrActiveDatabase = QSharedPointer<QSqlDatabase>(&activeDatabase);
00031         return activeDatabase.open();
00032     }
00033
      //-------------------------------------------------------------------------------------------------------------------//
00034     void DatabaseHandler::closeDatabase() {
00035         activeDatabase.close();
00036         QSqlDatabase::removeDatabase("db1");
00037     }
00038
      //-------------------------------------------------------------------------------------------------------------------//
00039     bool DatabaseHandler::executeCommand(const QString &command) {
00040         bool queryExecuted = false;
00041         QSqlQuery query(activeDatabase);
00042         if (!query.exec(command)) {
00043             lastSqlError = query.lastError();
00044             qDebug() << lastSqlError.text();
00045             queryExecuted = false;
00046         } else {
00047             queryExecuted = true;
00048         }
00049         return queryExecuted;
00050     }
00051
      //-------------------------------------------------------------------------------------------------------------------//
00052     QSharedPointer<QSqlDatabase> DatabaseHandler::getActiveDatabase() {
00053         return ptrActiveDatabase;
00054     }
00055
      //-------------------------------------------------------------------------------------------------------------------//
00056     const QSqlError &DatabaseHandler::getSqlError() const {
00057         return lastSqlError;
00058     }
00059 }
```

11.11 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/databaseClasses/databasehandler.hpp File Reference

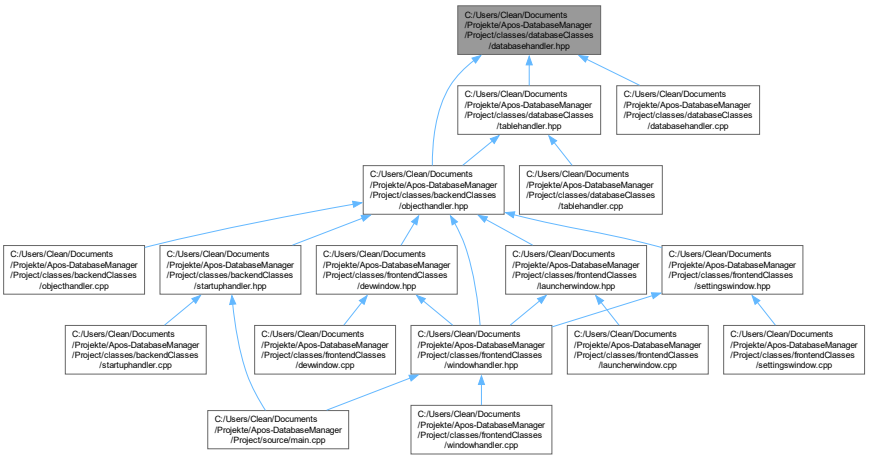
Header file for the DatabaseHandler class.

```
#include <QDebug>
#include <QtSql>
```

Include dependency graph for databasehandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposDatabase::DatabaseHandler

  Provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.

Namespaces

- namespace AposDatabase

11.11.1 Detailed Description

Header file for the DatabaseHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the DatabaseHandler class, which is a part of the application's backend logic. The DatabaseHandler class provides the functionality for initializing and closing the database, executing SQL commands, and getting the active database and SQL error.

Note

The application is part of a student project and is not intended for commercial use.

See also

QtSql

QSqlDatabase

QSqlError

QSharedPointer

QDebug

Definition in file databasehandler.hpp.

11.12 databasehandler.hpp

Go to the documentation of this file.

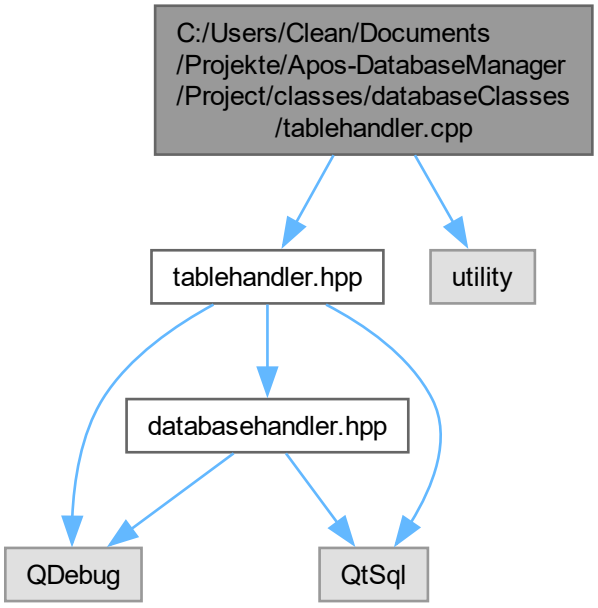
```
00001
00021 #pragma once
00022
00023 #include <QDebug>
00024 #include <QtSql>
00025
00026
      //-----------------------------------------------------------------------------------------------------------------------/
00027 namespace AposDatabase {
00040     class DatabaseHandler {
00041     public:
00047         DatabaseHandler();
00048
00055         bool initDatabase();
00056
00062         void closeDatabase();
00063
00071         bool executeCommand(const QString &command);
00072
00079         QSharedPointer<QSqlDatabase> getActiveDatabase();
00080
00087         [[nodiscard]] const QSqlError &getSqlError() const;
00088     private:
00094         QSqlError lastSqlError;
00095
00101         QSqlDatabase activeDatabase;
00102
00108         QSharedPointer<QSqlDatabase> ptrActiveDatabase;
00109
00115         QString databasePath =
      R"(C:\Users\Clean\Documents\Projekte\Apos-DatabaseManager\Project\resources\defaultDatabase\userDatabase.db)";
00116     };
00117 }
```

11.13 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/databaseClasses/tablehandler.cpp File Reference

Source file for the TableHandler class.

```
#include "tablehandler.hpp"
#include <utility>
```

Include dependency graph for tablehandler.cpp:



Namespaces

- namespace AposDatabase

11.13.1 Detailed Description

Source file for the TableHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the TableHandler class, which is a part of the application's backend logic. The TableHandler class provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

Note

The application is part of a student project and is not intended for commercial use.

See also

DatabaseHandler

QSharedPointer

QDebug

QtSql

QSqlTableModel

QSqlError

QSqlQuery

QSqlDatabase

Definition in file tablehandler.cpp.

11.14 tablehandler.cpp

Go to the documentation of this file.

```
00001
00023 #include "tablehandler.hpp"
00024
00025 #include <utility>
00026
00027 namespace AposDatabase {
00028
      //-----------------------------------------------------------------------------------------------------------//
00029     TableHandler::TableHandler(QSharedPointer<DatabaseHandler> newDbHandler) {
00030         ptrDbHandler = std::move(newDbHandler);
00031     }
00032
      //-----------------------------------------------------------------------------------------------------------//
00033     TableHandler::TableHandler(QSharedPointer<DatabaseHandler> newDbHandler, const QString &tableName)
      {
00034         ptrDbHandler = std::move(newDbHandler);
00035         activeTableName = tableName;
00036         ptrTableModel = QSharedPointer<QSqlTableModel>(
00037                 new QSqlTableModel(nullptr, *ptrDbHandler->getActiveDatabase()));
00038         ptrTableModel->setTable(tableName);
00039         if (!ptrTableModel->select()) {
00040             throw std::runtime_error("Failed to select table");
00041         }
00042     }
00043
      //-----------------------------------------------------------------------------------------------------------//
00044     TableHandler::~TableHandler() {
00045         ptrTableModel = nullptr;
00046         qDebug() << "TableHandler destroyed";
00047     }
00048
      //-----------------------------------------------------------------------------------------------------------//
00049     void TableHandler::generateTableModel() {
00050         ptrTableModel = QSharedPointer<QSqlTableModel>(new QSqlTableModel(nullptr,
      *ptrDbHandler->getActiveDatabase()));
00051         ptrTableModel->setTable(activeTableName);
00052         if (!ptrTableModel->select()) {
00053             throw std::runtime_error("Failed to select table");
00054         }
00055     }
00056
      //-----------------------------------------------------------------------------------------------------------//
00057     void TableHandler::generateTableModel(const QString &tableName) {
00058         activeTableName = tableName;
00059         ptrTableModel = QSharedPointer<QSqlTableModel>(new QSqlTableModel(nullptr,
      *ptrDbHandler->getActiveDatabase()));
00060         ptrTableModel.data()->setTable(activeTableName);
00061         if (!ptrTableModel.data()->select()) {
00062             qDebug() << "Failed to select table:" << ptrTableModel->lastError();
```

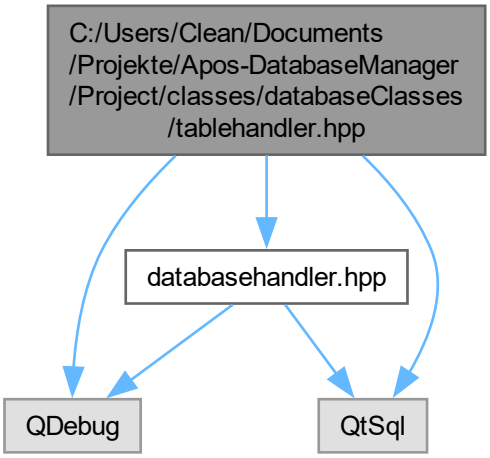
```
00063             throw std::runtime_error("Failed to select table");
00064         }
00065     }
00066
      //-----------------------------------------------------------------------------------------------------------//
00067     bool TableHandler::insertIntoTable(const QString &tableName, const QString &value1, const QString
      &value2,
00068                                        const QString &value3, const QString &value4, const QString
      &value5) {
00069         bool querrySuccess = false;
00070         QSqlQuery query(*ptrDbHandler->getActiveDatabase());
00071         if (!query.prepare(
00072                 QString("INSERT INTO %1 VALUES (:value1, :value2, :value3, :value4,
      :value5)").arg(tableName))) {
00073             lastTableError = query.lastError();
00074             querrySuccess = false;
00075
00076         } else {
00077             query.bindValue(":value1", value1);
00078             query.bindValue(":value2", value2);
00079             query.bindValue(":value3", value3);
00080             query.bindValue(":value4", value4);
00081             query.bindValue(":value5", value5);
00082             if (!query.exec()) {
00083                 lastTableError = query.lastError();
00084                 querrySuccess = false;
00085             } else {
00086                 querrySuccess = true;
00087             }
00088         }
00089         return querrySuccess;
00090     }
00091
      //-----------------------------------------------------------------------------------------------------------//
00092     const QString &TableHandler::getActiveTableName() const {
00093         return activeTableName;
00094     }
00095
      //-----------------------------------------------------------------------------------------------------------//
00096     void TableHandler::setActiveTableName(const QString &newActiveTableName) {
00097         activeTableName = newActiveTableName;
00098     }
00099
      //-----------------------------------------------------------------------------------------------------------//
00100     const QSqlError &TableHandler::getLastTableError() const {
00101         return lastTableError;
00102     }
00103
      //-----------------------------------------------------------------------------------------------------------//
00104     QSharedPointer<QSqlTableModel> TableHandler::getTableModel() {
00105         return ptrTableModel;
00106     }
00107 }
```

11.15 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/databaseClasses/tablehandler.hpp File Reference

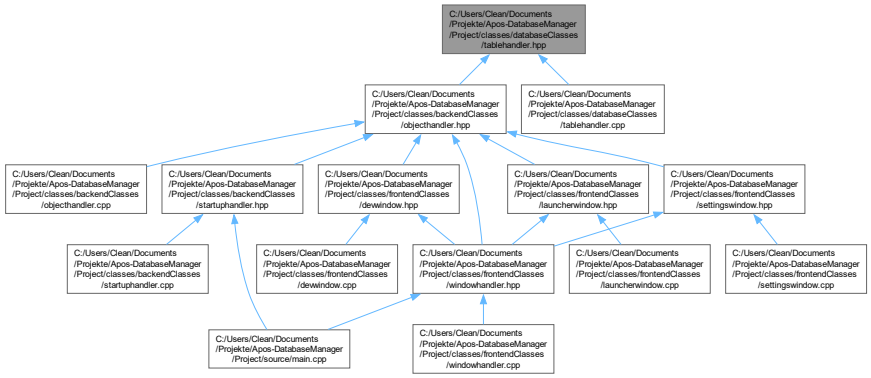
Header file for the TableHandler class.

```
#include <QDebug>
#include <QtSql>
#include "databasehandler.hpp"
```

Include dependency graph for tablehandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposDatabase::TableHandler

  Provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

Namespaces

- namespace AposDatabase

11.15.1 Detailed Description

Header file for the TableHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the TableHandler class, which is a part of the application's backend logic. The TableHandler class provides the functionality for generating a table model, inserting into a table, and getting the active table name, table model, and last table error.

Note

The application is part of a student project and is not intended for commercial use.

See also

DatabaseHandler

QSharedPointer

QDebug

QtSql

QSqlTableModel

QSqlError

QSqlQuery

QSqlDatabase

Definition in file tablehandler.hpp.

11.16 tablehandler.hpp

Go to the documentation of this file.

```
00001
00024 #pragma once
00025
00026 #include <QDebug>
00027 #include <QtSql>
00028
00029 #include "databasehandler.hpp"
00030
00031
      //-------------------------------------------------------------------------------------------------------------/
00032 namespace AposDatabase {
00048     class TableHandler {
00049     public:
00056         explicit TableHandler(QSharedPointer<DatabaseHandler> newDbHandler);
00057
```

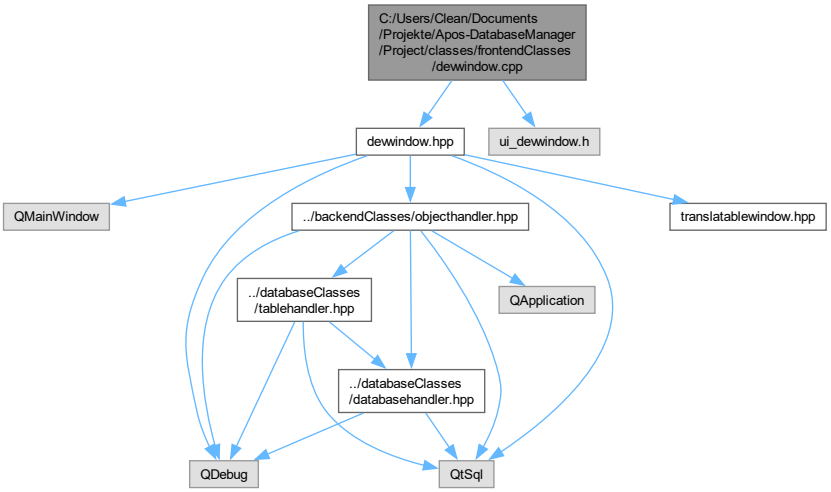
```
00065         TableHandler(QSharedPointer<DatabaseHandler> newDbHandler, const QString &tableName);
00066
00072         ~TableHandler();
00073
00079         void generateTableModel();
00080
00087         void generateTableModel(const QString &tableName);
00088
00101         bool insertIntoTable(const QString &tableName, const QString &value1, const QString &value2,
00102                              const QString &value3, const QString &value4, const QString &value5);
00103
00110         void setActiveTableName(const QString &newActiveTableName);
00111
00118         QSharedPointer<QSqlTableModel> getTableModel();
00119
00126         [[nodiscard]] const QString &getActiveTableName() const;
00127
00134         [[nodiscard]] const QSqlError &getLastTableError() const;
00135     private:
00141         QString activeTableName = "userTable";
00142
00148         QSharedPointer<DatabaseHandler> ptrDbHandler;
00149
00155         QSharedPointer<QSqlTableModel> ptrTableModel;
00156
00162         QSqlError lastTableError;
00163     };
00164 }
```

11.17 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/devwindow.cpp File Reference

Source file for the DevWindow class.

```
#include "devwindow.hpp"
#include "ui_devwindow.h"
```

Include dependency graph for devwindow.cpp:



Namespaces

- namespace AposFrontend

11.17.1 Detailed Description

Source file for the DevWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the DevWindow class, which is a part of the application's frontend logic. The DevWindow class inherits from QMainWindow and TranslatableWindow, and it provides the user interface for the developer window.

Note

The application is part of a student project and is not intended for commercial use.

See also

QMainWindow

TranslatableWindow

AposBackend::ObjectHandler

Definition in file devwindow.cpp.

11.18 devwindow.cpp

Go to the documentation of this file.

```
00001
00018 #include "devwindow.hpp"
00019 #include "ui_devwindow.h"
00020
00021 namespace AposFrontend {
00022
      //-----------------------------------------------------------------------------------------------------------------//
00023     DevWindow::DevWindow(QWidget *parent, QSharedPointer<AposBackend::ObjectHandler> newObjectHandler)
00024             : QMainWindow(parent), ui(new Ui::DevWindow) {
00025         qDebug() << "DevWindow constructor called";
00026         ui->setupUi(this);
00027         qDebug() << "DevWindow ui setup";
00028         ptrObjectHandler = qMove(newObjectHandler);
00029         qDebug() << "DevWindow ptrObjectHandler moved";
00030         //TODO: implement Logger
00031         devConnectUi();
00032         qDebug() << "DevWindow connected UiElements";
00033
00034
00035
00036
00037
00038     }
```

```
00039
      //-----------------------------------------------------------------------------------------------------------------//
00040     DevWindow::~DevWindow() {
00041         delete ui;
00042     }
00043
      //-----------------------------------------------------------------------------------------------------------------//
00044     bool DevWindow::devConnectUi() {
00045         //TODO: implement Logger
00046         //Buttons
00047         connect(ui->btnInitDB, &QPushButton::clicked, this, &DevWindow::initDbClicked);
00048         connect(ui->btnCloseDB, &QPushButton::clicked, this, &DevWindow::closeDBClicked);
00049         connect(ui->btnExecute, &QPushButton::clicked, this, &DevWindow::executeClicked);
00050         connect(ui->btnSelectTable, &QPushButton::clicked, this, &DevWindow::selectTableClicked);
00051         connect(ui->btnAdd, &QPushButton::clicked, this, &DevWindow::addValuesClicked);
00052         connect(ui->btnUpdate, &QPushButton::clicked, this, &DevWindow::updateTableClicked);
00053         connect(ui->inReturnToLauncher, &QPushButton::clicked, this,
      &DevWindow::returnToLauncherClicked);
00054         connect(ui->inSettings, &QPushButton::clicked, this, &DevWindow::settingsClicked);
00055         //Checkboxes
00056         connect(ui->clearCommandAfterExecute, &QCheckBox::stateChanged, this,
      &DevWindow::clearCommandAfterExecuteStateChanged);
00057         connect(ui->clearInputsAfterInsert, &QCheckBox::stateChanged, this,
      &DevWindow::clearInputsAfterInsertStateChanged);
00058         return true;
00059     }
00060
      //-----------------------------------------------------------------------------------------------------------------//
00061     void DevWindow::logEvent(const QString &type,const QString &message) {
00062         //TODO: implement Logger
00063         ui->outLog->append("Log |  " + type + ": " + message);
00064         //TODO: implement Logger
00065         qDebug() << "Logged:  " + type + " - " + message;
00066     }
00067
      //-----------------------------------------------------------------------------------------------------------------//
00068     void DevWindow::logEvent(const QString &message,const QSqlError &error) {
00069         //TODO: implement Logger
00070         ui->outLog->append("Log | " + message + "-" + error.text());
00071         //TODO: implement Logger
00072         qDebug() << "Logged: " + message + " - " + error.text();
00073     }
00074
      //-----------------------------------------------------------------------------------------------------------------//
00075     void DevWindow::logEvent(const QString &message) {
00076         //TODO: implement Logger
00077         ui->outLog->append("Log | " + message);
00078         //TODO: implement Logger
00079         qDebug() << "Logged: " + message;
00080     }
00081
      //-----------------------------------------------------------------------------------------------------------------//
00082     void DevWindow::enableButtons(bool databaseLoaded) {
00083         ui->btnAdd->setEnabled(databaseLoaded);
00084         ui->btnCloseDB->setEnabled(databaseLoaded);
00085         ui->btnUpdate->setEnabled(databaseLoaded);
00086         ui->btnExecute->setEnabled(databaseLoaded);
00087         ui->btnSelectTable->setEnabled(databaseLoaded);
00088         ui->btnInitDB->setEnabled(!databaseLoaded);
00089         //TODO: implement Logger
00090         logEvent("action", "Buttons enabled/disabled");
00091     }
00092
      //-----------------------------------------------------------------------------------------------------------------//
00093     void DevWindow::setModelViews(const QSharedPointer<QSqlTableModel>& tableModel) {
00094         //TODO: implement Logger
00095         qDebug() <<"SharedPointer.data(): " << tableModel->database();
00096         ui->outTable->setModel(tableModel.data());
00097         ui->outColumn->setModel(tableModel.data());
00098         ui->outList->setModel(tableModel.data());
00099     }
00100
      //-----------------------------------------------------------------------------------------------------------------//
00101     void DevWindow::setModelViews() {
00102         ui->outTable->setModel(nullptr);
00103         ui->outColumn->setModel(nullptr);
00104         ui->outList->setModel(nullptr);
00105     }
00106
      //-----------------------------------------------------------------------------------------------------------------//
00107     void DevWindow::assignInputs() {
00108         input1 = ui->inInput1->text();
00109         input2 = ui->inInput2->text();
00110         input3 = ui->inInput3->text();
00111         input4 = ui->inInput4->text();
00112         input5 = ui->inInput5->text();
00113     }
```

```
00114
      //-----------------------------------------------------------------------------------------------------------------//
00115     void DevWindow::initDatabase() {
00116         if (!ptrObjectHandler->initDatabaseObject()) {
00117             //TODO: implement Logger
00118             logEvent("Error initiating database", ptrObjectHandler->getPtrDbHandler()->getSqlError());
00119
00120         }
00121         else {
00122             //TODO: implement Logger
00123             logEvent("action", "Database initiated");
00124             //TODO: implement Logger
00125             logEvent("action",
00126                      QString(ptrObjectHandler->getPtrDbHandler()->getActiveDatabase()->databaseName())
      + " opened");
00127             //TODO: implement Logger
00128             logEvent("status", ptrObjectHandler->getPtrDbHandler()->getActiveDatabase()->isOpen()
00129                                                                                 ? "Database open":
      "Database closed");
00130         }
00131     }
00132
      //-----------------------------------------------------------------------------------------------------------------//
00133     void DevWindow::closeDatabase(const QSharedPointer<AposDatabase::DatabaseHandler>& db) {
00134         db->closeDatabase();
00135         //TODO: implement Logger
00136         logEvent("action", "Database closed");
00137         //TODO: implement Logger
00138         logEvent("status",
00139                  ptrObjectHandler->getPtrDbHandler()->getActiveDatabase()->isOpen() ? "Database open"
      : "Database closed");
00140     }
00141
      //-----------------------------------------------------------------------------------------------------------------//
00142     bool DevWindow::checkCheckbox(int argCb) {
00143         bool checked = false;
00144         if (argCb == 2) {
00145             checked = true;
00146         } else if (argCb == 0) {
00147             checked = false;
00148         } else {
00149             //TODO: implement Logger
00150             logEvent("warning", "Something went wrong!");
00151             checked = false;
00152         }
00153         return checked;
00154     }
00155
      //-----------------------------------------------------------------------------------------------------------------//
00156     void DevWindow::clearInputs(bool clearBool) {
00157         if (clearBool) {
00158             ui->inInput1->clear();
00159             ui->inInput2->clear();
00160             ui->inInput3->clear();
00161             ui->inInput4->clear();
00162             ui->inInput5->clear();
00163             //TODO: implement Logger
00164             logEvent("action", "cleared inputs");
00165         }
00166     }
00167
      //-----------------------------------------------------------------------------------------------------------------//
00168     void DevWindow::clearCommandBox(bool clearBool) {
00169         if (clearBool) {
00170             ui->inCommand->clear();
00171             //TODO: implement Logger
00172             logEvent("action", "tried to clear command-line");
00173         }
00174     }
00175
      //-----------------------------------------------------------------------------------------------------------------//
00176     void DevWindow::initDbClicked() {
00177         initDatabase();
00178         //TODO: implement Logger
00179         qDebug() << "Database initialized";
00180         if(!ptrObjectHandler->initTableObject("userTable")){
00181             //TODO: implement Logger
00182             logEvent("Error initiating table", ptrObjectHandler->getTableSqlError());
00183             return;
00184         }
00185         //TODO: implement Logger
00186         qDebug() << "TableHander initialized";
00187         setModelViews(ptrObjectHandler->getPtrTableHandler()->getTableModel());
00188         //TODO: implement Logger
00189         qDebug() << "ModelViews set";
00190         enableButtons(true);
00191     }
```

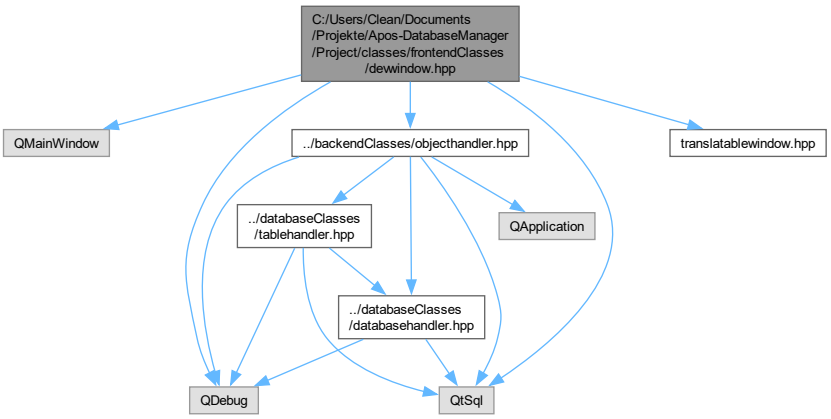
```
00192
      //-----------------------------------------------------------------------------------------------------------------//
00193     void DevWindow::closeDBClicked() {
00194         setModelViews();
00195         closeDatabase(ptrObjectHandler->getPtrDbHandler());
00196         enableButtons(false);
00197     }
00198
      //-----------------------------------------------------------------------------------------------------------------//
00199     void DevWindow::executeClicked() {
00200         if (!ptrObjectHandler->getPtrDbHandler()->executeCommand(ui->inCommand->toPlainText())) {
00201             //TODO: implement Logger
00202             logEvent("Error executing command", ptrObjectHandler->getPtrDbHandler()->getSqlError());
00203             clearCommandBox(clearCommand);
00204             return;
00205         }
00206         clearCommandBox(clearCommand);
00207         //TODO: implement Logger
00208         logEvent("action", "Command executed");
00209     }
00210
      //-----------------------------------------------------------------------------------------------------------------//
00211     void DevWindow::selectTableClicked() {
00212         ptrObjectHandler->getPtrTableHandler()->generateTableModel();
00213         //TODO: implement Logger
00214         logEvent("action", "Table selected");
00215         setModelViews(ptrObjectHandler->getPtrTableHandler()->getTableModel());
00216     }
00217
      //-----------------------------------------------------------------------------------------------------------------//
00218     void DevWindow::addValuesClicked() {
00219         assignInputs();
00220         if
      (!ptrObjectHandler->getPtrTableHandler()->insertIntoTable(ptrObjectHandler->getActiveTableName(),
      input1, input2,
00221                                                                 input3,
00222                                                                 input4, input5)) {
00223             //TODO: implement Logger
00224             logEvent("Insert Error", ptrObjectHandler->getTableSqlError());
00225             clearInputs(clearInput);
00226             return;
00227         }
00228         clearInputs(clearInput);
00229         //TODO: implement Logger
00230         logEvent("action", "Values inserted");
00231     }
00232
      //-----------------------------------------------------------------------------------------------------------------//
00233     void DevWindow::updateTableClicked() {
00234         ptrObjectHandler->getPtrTableHandler()->generateTableModel();
00235         setModelViews(ptrObjectHandler->getPtrTableHandler()->getTableModel());
00236         //TODO: implement Logger
00237         logEvent("action", "Table view updated");
00238     }
00239
      //-----------------------------------------------------------------------------------------------------------------//
00240     void DevWindow::clearCommandAfterExecuteStateChanged(int arg1) {
00241         clearCommand = checkCheckbox(arg1);
00242         //TODO: implement Logger
00243         logEvent("status", "Command will clear after execute: " + QString(clearCommand ? "true" :
      "false"));
00244     }
00245
      //-----------------------------------------------------------------------------------------------------------------//
00246     void DevWindow::clearInputsAfterInsertStateChanged(int arg1) {
00247         clearInput = checkCheckbox(arg1);
00248         //TODO: implement Logger
00249         logEvent("status", "Inputs will be cleared after execution: " + QString(clearInput ? "true" :
      "false"));
00250     }
00251
      //-----------------------------------------------------------------------------------------------------------------//
00252     void DevWindow::returnToLauncherClicked() {
00253         emit returnToLauncher();
00254     }
00255
      //-----------------------------------------------------------------------------------------------------------------//
00256     void DevWindow::settingsClicked() {
00257         emit openSettings();
00258     }
00259
      //-----------------------------------------------------------------------------------------------------------------//
00260     void DevWindow::retranslateUi() {
00261         ui->retranslateUi(this);
00262     }
00263
00264 }
```

11.19 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/devwindow.hpp File Reference

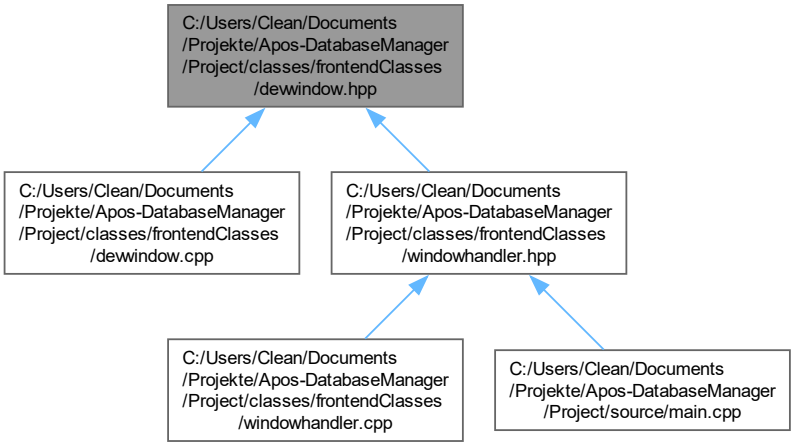
Header file for the DevWindow class.

```
#include <QMainWindow>
#include <QDebug>
#include <QtSql>
#include "../backendClasses/objecthandler.hpp"
#include "translatablewindow.hpp"
```

Include dependency graph for devwindow.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposFrontend::DevWindow

  Provides the user interface for the developer window.

Namespaces

- namespace Ui
- namespace AposFrontend

11.19.1 Detailed Description

Header file for the DevWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the DevWindow class, which is a part of the application's frontend logic. The DevWindow class inherits from QMainWindow and TranslatableWindow, and it provides the user interface for the developer window.

Note

The application is part of a student project and is not intended for commercial use.

See also

TranslatableWindow

AposBackend::ObjectHandler

QMainWindow

QSharedPointer

QDebug

QtSql

Definition in file devwindow.hpp.

11.20 devwindow.hpp

Go to the documentation of this file.

```
00001
00022 #pragma once
00023
00024 #include <QMainWindow>
00025 #include <QDebug>
00026 #include <QtSql>
00027
00028 #include "../backendClasses/objecthandler.hpp"
00029 #include "translatablewindow.hpp"
00030
00031
      //-----------------------------------------------------------------------------------------------------------------/
00032 namespace Ui { class DevWindow; }
00033
      //-----------------------------------------------------------------------------------------------------------------/
00034 namespace AposFrontend {
00049     class DevWindow : public QMainWindow, public TranslatableWindow {
00050     Q_OBJECT
00051     public:
00062         explicit DevWindow(QWidget *parent = nullptr, QSharedPointer<AposBackend::ObjectHandler>
      objectHandler = nullptr);
00063
00071         ~DevWindow() override;
00072
00073
00074
00075         //TODO: Replace with logging class
00085         void logEvent(const QString &type, const QString &message);
00086
00097         void logEvent(const QString &message, const QSqlError &error);
00098
00108         void logEvent(const QString &message);
00109
00117         void retranslateUi() override;
00118     signals:
00125         void returnToLauncher();
00126
00134         void openSettings();
00135     private slots:
00143         void initDbClicked();
00144
00152         void closeDBClicked();
00153
00161         void executeClicked();
00162
00170         void selectTableClicked();
00171
00179         void addValuesClicked();
00180
00188         void updateTableClicked();
00189
00199         void clearCommandAfterExecuteStateChanged(int arg1);
00200
00210         void clearInputsAfterInsertStateChanged(int arg1);
00211
00219         void returnToLauncherClicked();
00220
00228         void settingsClicked();
00229
00230     private:
00231         //TODO: Add documentation
00232         bool devConnectUi();
00233
00241         void initDatabase();
00242
00252         void closeDatabase(const QSharedPointer<AposDatabase::DatabaseHandler> &db);
00253
00261         void setModelViews();
00262
00272         void setModelViews(const QSharedPointer<QSqlTableModel>& tableModel);
00273
00283         void enableButtons(bool databaseLoaded);
00284
00292         void assignInputs();
00293
00303         void clearInputs(bool clearBool);
00304
00315         bool checkCheckbox(int argCb);
00316
00326         void clearCommandBox(bool clearBool);
00327
00335         Ui::DevWindow *ui;
```

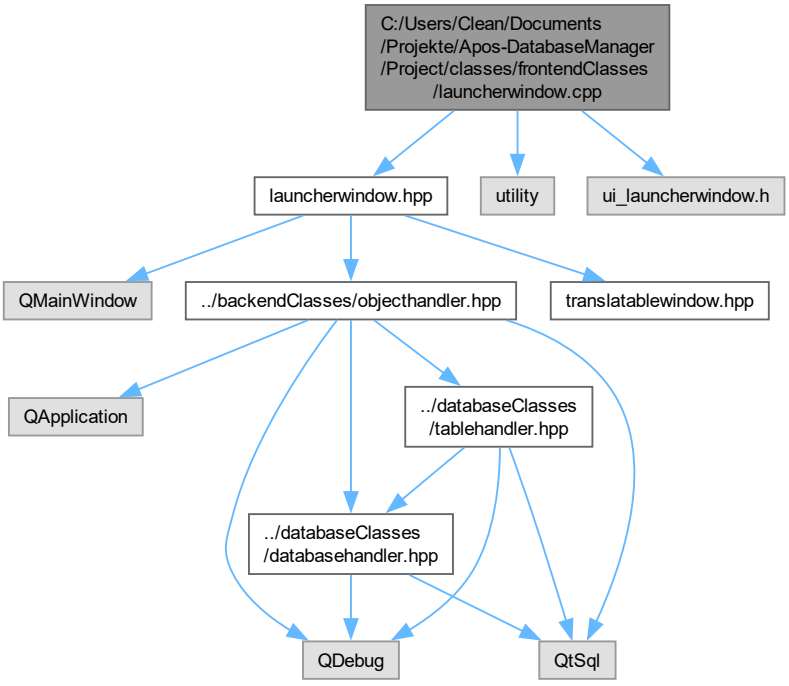
```
00336
00344         QSharedPointer<AposBackend::ObjectHandler> ptrObjectHandler = nullptr;
00345
00353         QString input1, input2, input3, input4, input5;
00354
00362         bool clearCommand = false, clearInput = false;
00363     };
00364
00365 }
```

11.21 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/launcherwindow.cpp File Reference

Source file for the LauncherWindow class.

```
#include "launcherwindow.hpp"
#include <utility>
#include "ui_launcherwindow.h"
```

Include dependency graph for launcherwindow.cpp:



Namespaces

- namespace AposFrontend

11.21.1 Detailed Description

Source file for the LauncherWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the LauncherWindow class, which is a part of the application's frontend logic. The LauncherWindow class inherits from QMainWindow and TranslatableWindow, and it provides the user interface for the launcher window.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposBackend::ObjectHandler

TranslatableWindow

QMainWindow

Definition in file launcherwindow.cpp.

11.22 launcherwindow.cpp

Go to the documentation of this file.

```
00001
00018 #include "launcherwindow.hpp"
00019
00020 #include <utility>
00021 #include "ui_launcherwindow.h"
00022
00023
00024 namespace AposFrontend {
00025
      //-----------------------------------------------------------------------------------------------------------//
00026     LauncherWindow::LauncherWindow(QWidget *parent, QSharedPointer<AposBackend::ObjectHandler>
      newObjectHandler) :
00027             QMainWindow(parent),
00028             ui(new Ui::LauncherWindow) {
00029         ui->setupUi(this);
00030         objectHandler = std::move(newObjectHandler);
00031         //TODO: implement Logger
00032         launcherConnectUi();
00033     }
00034
      //-----------------------------------------------------------------------------------------------------------//
00035     LauncherWindow::~LauncherWindow() {
00036         delete ui;
00037     }
```

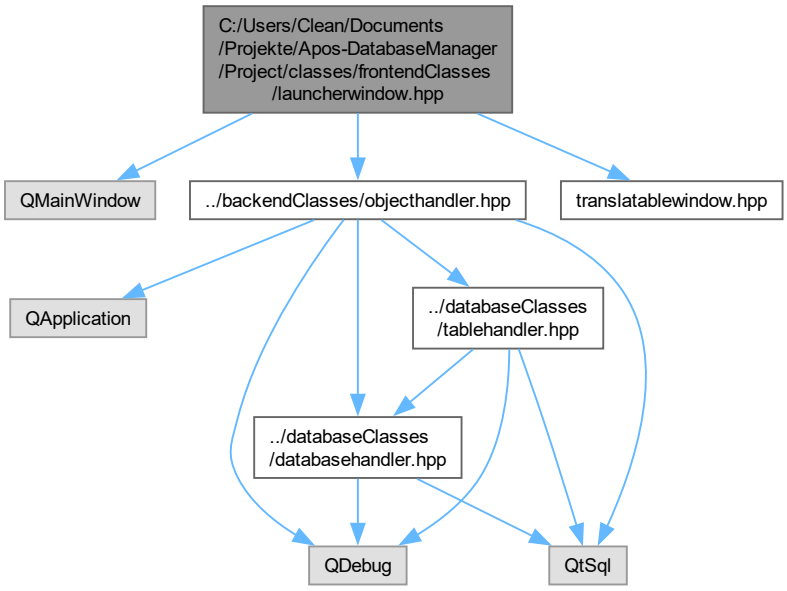
```
00038
      //-----------------------------------------------------------------------------------------------------------//
00039     bool LauncherWindow::launcherConnectUi(){
00040         //TODO: implement Logger
00041         connect(ui->btnShowDev, SIGNAL(clicked()), this, SLOT(showDevClicked()));
00042         //TODO: implement Logger
00043         connect(ui->btnSettings, SIGNAL(clicked()), this, SLOT(pushButtonClicked()));
00044         return true;
00045     }
00046
      //-----------------------------------------------------------------------------------------------------------//
00047     void LauncherWindow::showDevClicked() {
00048         emit openDevWindow();
00049     }
00050
      //-----------------------------------------------------------------------------------------------------------//
00051     void LauncherWindow::pushButtonClicked() {
00052         emit openSettings();
00053     }
00054
      //-----------------------------------------------------------------------------------------------------------//
00055     void LauncherWindow::retranslateUi() {
00056     }
00057 }
```

11.23 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/launcherwindow.hpp File Reference

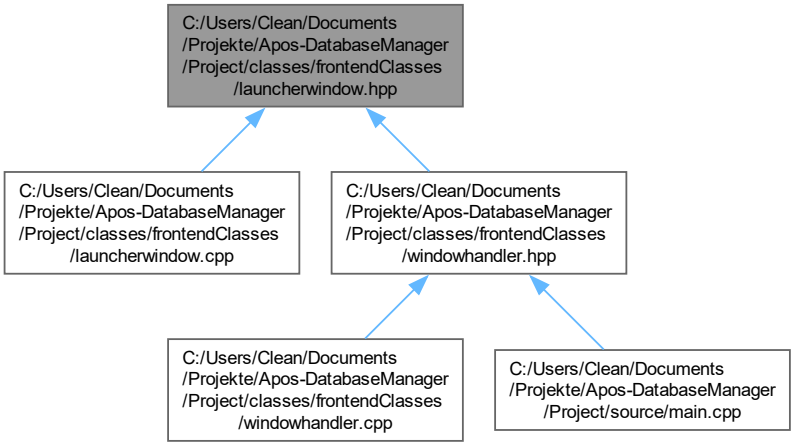
Header file for the LauncherWindow class.

```
#include <QMainWindow>
#include "../backendClasses/objecthandler.hpp"
#include "translatablewindow.hpp"
```

Include dependency graph for launcherwindow.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposFrontend::LauncherWindow

  Provides the user interface for the launcher window.

Namespaces

- namespace Ui
- namespace AposFrontend

11.23.1 Detailed Description

Header file for the LauncherWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the LauncherWindow class, which is a part of the application's frontend logic. The LauncherWindow class inherits from QMainWindow and TranslatableWindow, and it provides the user interface for the launcher window.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposBackend::ObjectHandler

TranslatableWindow

QMainWindow

Definition in file launcherwindow.hpp.

11.24 launcherwindow.hpp

Go to the documentation of this file.

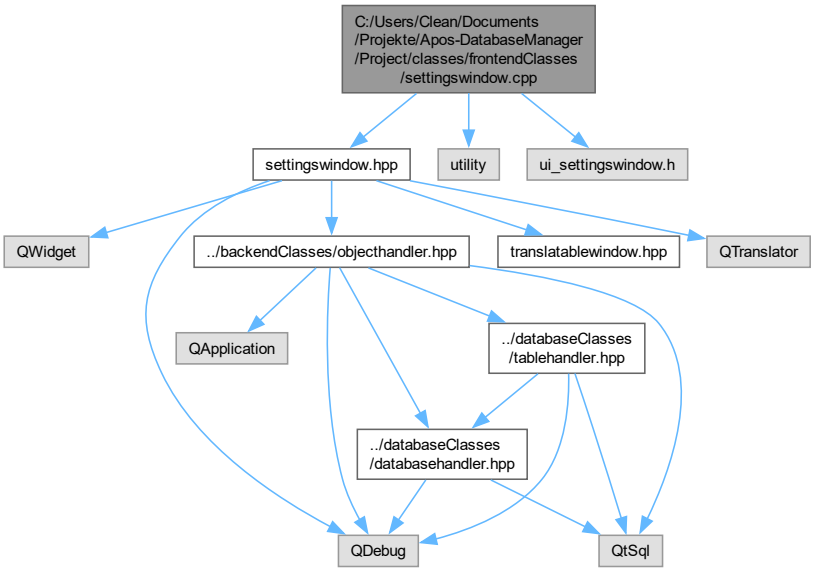
```
00001
00019 #pragma once
00020
00021 #include <QMainWindow>
00022
00023 #include "../backendClasses/objecthandler.hpp"
00024 #include "translatablewindow.hpp"
00025
00026
      //-----------------------------------------------------------------------------------------------------------------/
00027 namespace Ui {
00028     class LauncherWindow;
00029 }
00030
      //-----------------------------------------------------------------------------------------------------------------/
00031 namespace AposFrontend {
00045     class LauncherWindow : public QMainWindow, public TranslatableWindow {
00046     Q_OBJECT
00047     public:
00055         explicit LauncherWindow(QWidget *parent = nullptr,
00056                                 QSharedPointer<AposBackend::ObjectHandler> newObjectHandler =
      nullptr);
00057
00063         ~LauncherWindow() override;
00064
00065         ;
00066
00072         void retranslateUi() override;
00073     signals:
00079         void openDevWindow();
00080
00086         void openSettings();
00087     private slots:
00093         void showDevClicked();
00094
00100         void pushButtonClicked();
00101     private:
00102         //TODO: Add documentation
00103         bool launcherConnectUi();
00104
00110         Ui::LauncherWindow *ui;
00111
00117         QSharedPointer<AposBackend::ObjectHandler> objectHandler = nullptr;
00118     };
00119 }
```

11.25 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/settingswindow.cpp File Reference

Source file for the SettingsWindow class.

```
#include "settingswindow.hpp"
#include <utility>
#include "ui_settingswindow.h"
```

Include dependency graph for settingswindow.cpp:



Namespaces

- namespace AposFrontend

11.25.1 Detailed Description

Source file for the SettingsWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the SettingsWindow class, which is a part of the application's frontend logic. The SettingsWindow class inherits from QWidget and TranslatableWindow, and it provides the user interface for the settings window.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposBackend::ObjectHandler

TranslatableWindow

QWidget

QTranslator

QSharedPointer

Definition in file settingswindow.cpp.

11.26 settingswindow.cpp

Go to the documentation of this file.

```
00001
00020 #include "settingswindow.hpp"
00021
00022 #include <utility>
00023 #include "ui_settingswindow.h"
00024
00025
00026 namespace AposFrontend {
00027     SettingsWindow::SettingsWindow(QWidget *parent, QSharedPointer<AposBackend::ObjectHandler>
      newObjectHandler) :
00028             QWidget(parent),
00029             ui(new Ui::SettingsWindow) {
00030         ui->setupUi(this);
00031         ptrObjectHandler = std::move(newObjectHandler);
00032         ptrTranslator = QSharedPointer<QTranslator>(new QTranslator);
00033         settingsConnectUi();
00034     }
00035
00036     SettingsWindow::~SettingsWindow() {
00037         delete ui;
00038     }
00039
00040     void SettingsWindow::settingsConnectUi() {
00041         //TODO: implement Logger
00042         connect(ui->btnClose, SIGNAL(clicked()), this, SLOT(closeClicked()));
00043         connect(ui->btnApply, SIGNAL(clicked()), this, SLOT(applyClicked()));
00044         connect(ui->btnApplyAndClose, SIGNAL(clicked()), this, SLOT(applyAndCloseClicked()));
00045         connect(ui->inLanguage, SIGNAL(currentIndexChanged(int)), this,
      SLOT(languageCurrentIndexChanged(int)));
00046     }
00047
00048     void SettingsWindow::retranslateUi() {
00049         ui->retranslateUi(this);
00050     }
00051
00052     void SettingsWindow::closeClicked() {
00053         this->hide();
00054     }
00055
00056
00057     void SettingsWindow::applyClicked() {
00058         if (languageChanged) {
00059             qDebug() << "New Language will be applied";
00060
00061             languageIndex = tempLanguageIndex;
00062             installTranslator();
00063         }
00064         emit appliedSettings();
00065     }
00066
00067     void SettingsWindow::applyAndCloseClicked() {
00068         applyClicked();
00069         closeClicked();
00070     }
00071
```

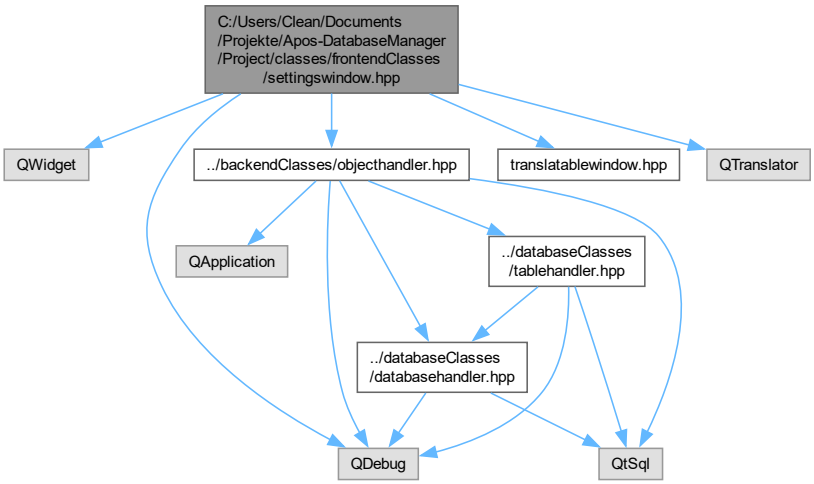
```
00072
00073     void SettingsWindow::languageCurrentIndexChanged(int index) {
00074         tempLanguageIndex = index;
00075         qDebug() « "checkboxIndex: " « index « "tempIndex: " « tempLanguageIndex « "index: " «
      languageIndex;
00076         if (tempLanguageIndex == languageIndex) {
00077             qDebug() « "Language not changed";
00078             languageChanged = false;
00079             return;
00080         }
00081         qDebug() « "Language changed";
00082         languageChanged = true;
00083     }
00084
00085     void SettingsWindow::installTranslator() {
00086         qDebug() « "Language Index: " « languageIndex;
00087
00088         QTranslator *translator = ptrTranslator.data();
00089         if(ptrObjectHandler->getPtrApplication()->removeTranslator(translator)){
00090             qDebug() « "removed translator";
00091         }
00092         else{
00093             qDebug() « "could not remove translator";
00094         }
00095         ptrTranslator = QSharedPointer<QTranslator>(new QTranslator);
00096         switch (languageIndex) {
00097             case 0:
00098                 (void)ptrTranslator->load(":/i18n/Apos-DatabaseManager_en_GB");
00099                 qDebug() « "tried to load english";
00100                 translator = ptrTranslator.data();
00101                 (void)ptrObjectHandler->getPtrApplication()->installTranslator(translator);
00102                 break;
00103             case 1:
00104                 (void)ptrTranslator->load(":/i18n/Apos-DatabaseManager_de_DE");
00105                 qDebug() « "tried to load german";
00106                 translator = ptrTranslator.data();
00107                 (void)ptrObjectHandler->getPtrApplication()->installTranslator(translator);
00108                 break;
00109             default:
00110                 qDebug() « "no language selected";
00111                 break;
00112         }
00113     }
00114 }
```

11.27 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/settingswindow.hpp File Reference

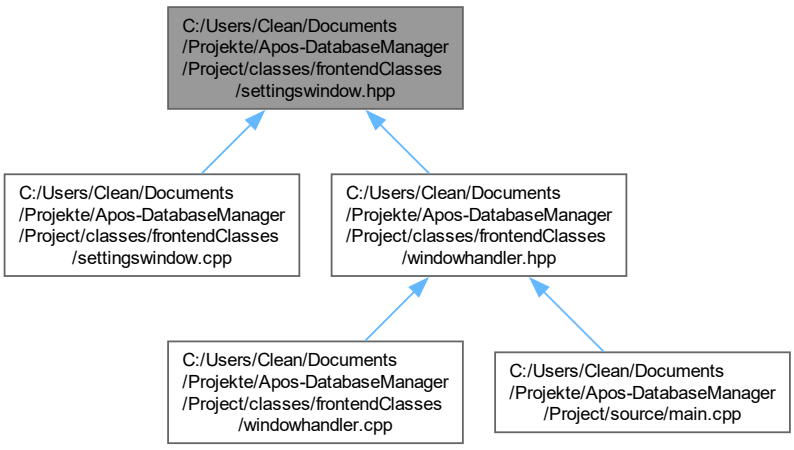
Header file for the SettingsWindow class.

```
#include <QWidget>
#include <QDebug>
#include "../backendClasses/objecthandler.hpp"
#include "translatablewindow.hpp"
#include <QTranslator>
```

Include dependency graph for settingswindow.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposFrontend::SettingsWindow

  Provides the user interface for the settings window.

Namespaces

- namespace Ui
- namespace AposFrontend

11.27.1 Detailed Description

Header file for the SettingsWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the SettingsWindow class, which is a part of the application's frontend logic. The SettingsWindow class inherits from QWidget and TranslatableWindow, and it provides the user interface for the settings window.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposBackend::ObjectHandler

TranslatableWindow

QWidget

QTranslator

QSharedPointer

Definition in file settingswindow.hpp.

11.28 settingswindow.hpp

Go to the documentation of this file.

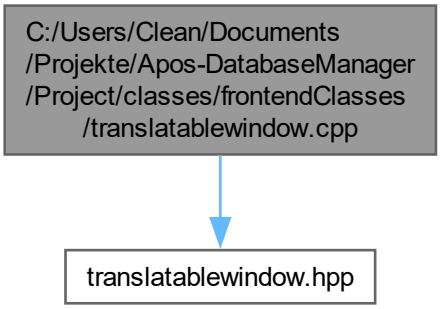
```
00001
00021 #pragma once
00022
00023 #include <QWidget>
00024 #include <QDebug>
00025 #include "../backendClasses/objecthandler.hpp"
00026 #include "translatablewindow.hpp"
00027 #include <QTranslator>
00028
00029
      //-----------------------------------------------------------------------------------------------------------------/
00030 namespace Ui {
00031     class SettingsWindow;
00032 }
00033
      //-----------------------------------------------------------------------------------------------------------------/
00034 namespace AposFrontend {
00049     class SettingsWindow : public QWidget, public TranslatableWindow {
00050     Q_OBJECT
00051     public:
```

```
00059         explicit SettingsWindow(QWidget *parent = nullptr,
00060                                 QSharedPointer<AposBackend::ObjectHandler> newObjectHandler =
    nullptr);
00061
00067         ~SettingsWindow() override;
00068
00074         void retranslateUi() override;
00075     signals:
00081         void appliedSettings();
00082     private slots:
00088         void closeClicked();
00089
00095         void applyClicked();
00096
00097         //TODO: add Documentation
00098         void applyAndCloseClicked();
00099
00106         void languageCurrentIndexChanged(int index);
00107     private:
00108         //TODO: add Documentation
00109         void settingsConnectUi();
00110
00116         void installTranslator();
00117
00123         Ui::SettingsWindow *ui;
00124
00130         QSharedPointer<AposBackend::ObjectHandler> ptrObjectHandler;
00131
00137         QSharedPointer<QTranslator> ptrTranslator;
00138
00144         int languageIndex{};
00145
00151         int tempLanguageIndex{};
00152
00158         bool languageChanged{};
00159     };
00160 }
```

11.29 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/translatablewindow.cpp File Reference

Source file for the TranslatableWindow class.

`#include "translatablewindow.hpp"`
Include dependency graph for translatablewindow.cpp:



Namespaces

- namespace AposFrontend

11.29.1 Detailed Description

Source file for the TranslatableWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the TranslatableWindow class, which is a part of the application's frontend logic. The TranslatableWindow class is an abstract base class that provides a function for retranslating the user interface.

Note

The application is part of a student project and is not intended for commercial use.

Definition in file translatablewindow.cpp.

11.30 translatablewindow.cpp

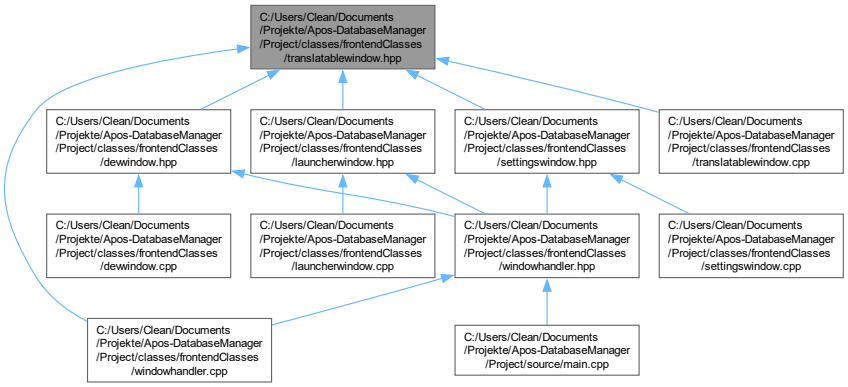
Go to the documentation of this file.

```
00001
00014 #include "translatablewindow.hpp"
00015 namespace AposFrontend {
00016
      //-----------------------------------------------------------------------------------------------------------------//
00017     TranslatableWindow::TranslatableWindow() = default;
00018 }
00019
```

11.31 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/translatablewindow.hpp File Reference

Header file for the TranslatableWindow class.

This graph shows which files directly or indirectly include this file:



Classes

- class AposFrontend::TranslatableWindow

  An abstract base class that provides a function for retranslating the user interface.

Namespaces

- namespace AposFrontend

11.31.1 Detailed Description

Header file for the TranslatableWindow class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the TranslatableWindow class, which is a part of the application's frontend logic. The TranslatableWindow class is an abstract base class that provides a function for retranslating the user interface.

Note

The application is part of a student project and is not intended for commercial use.

Definition in file translatablewindow.hpp.

11.32 translatablewindow.hpp

Go to the documentation of this file.

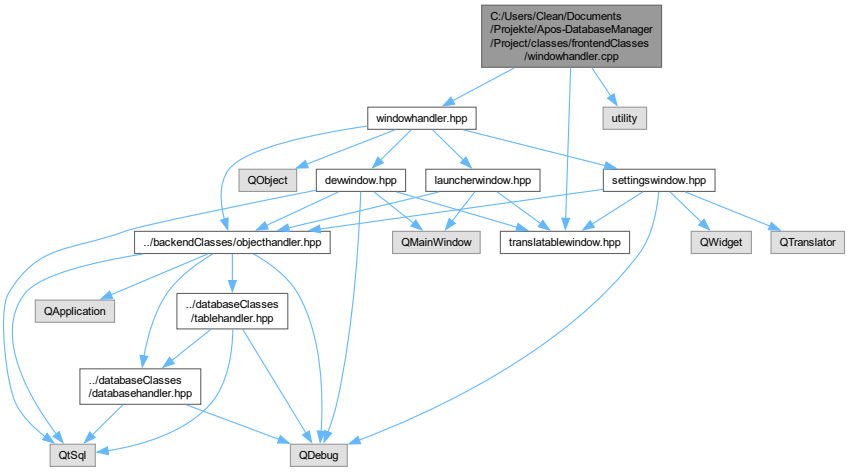
```
00001
00015 #pragma once
00016
00017
      //-----------------------------------------------------------------------------------------------------------/
00018 namespace AposFrontend {
00026     class TranslatableWindow {
00027     public:
00033         TranslatableWindow();
00034
00041         virtual void retranslateUi() = 0;
00042     };
00043 }
```

11.33 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/windowhandler.cpp File Reference

Source file for the WindowHandler class.

```
#include "windowhandler.hpp"
#include <utility>
#include "translatablewindow.hpp"
```

Include dependency graph for windowhandler.cpp:



Namespaces

- namespace AposFrontend

11.33.1 Detailed Description

Source file for the WindowHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the implementation of the WindowHandler class, which is a part of the application's frontend logic. The WindowHandler class provides the functionality for managing the application's windows, including the launcher window, developer window, and settings window. It interacts with the LauncherWindow, DevWindow, and SettingsWindow classes and uses the ObjectHandler class to manage the application's objects.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposBackend::ObjectHandler

LauncherWindow

DevWindow

SettingsWindow

QSharedPointer

QObject

Definition in file windowhandler.cpp.

11.34 windowhandler.cpp

Go to the documentation of this file.

```
00001
00022 #include "windowhandler.hpp"
00023
00024 #include <utility>
00025 #include "translatablewindow.hpp"
00026
00027 namespace AposFrontend {
00028     WindowHandler::WindowHandler(QSharedPointer<AposBackend::ObjectHandler> newObjectHandler) {
00029
00030         ptrObjectHandler = qMove(newObjectHandler);
00031         ptrLauncherWindow = QSharedPointer<LauncherWindow>(new LauncherWindow(nullptr,
      ptrObjectHandler));
00032         ptrSettingsWindow = QSharedPointer<SettingsWindow>(new SettingsWindow(nullptr,
      ptrObjectHandler));
00033         (void)QObject::connect(ptrLauncherWindow.data(), &LauncherWindow::openDevWindow, this,
      &WindowHandler::showDevWindow,
00034                                Qt::DirectConnection);
00035         (void)QObject::connect(ptrLauncherWindow.data(), &LauncherWindow::openSettings, this,
      &WindowHandler::showSettingsWindow);
00036         (void)QObject::connect(ptrSettingsWindow.data(), &SettingsWindow::appliedSettings, this,
      &WindowHandler::applySettings);
00037
00038     }
00039
00040     void WindowHandler::showLaunchWindow() {
00041         if (ptrDevWindow != nullptr) {
00042             if (!ptrDevWindow->isHidden()) {
00043                 ptrDevWindow->hide();
00044             }
00045         }
00046         ptrLauncherWindow->show();
00047     }
00048
00049     void WindowHandler::showDevWindow() {
00050         ptrLauncherWindow->hide();
00051         if (ptrDevWindow == nullptr) {
00052             ptrDevWindow = QSharedPointer<DevWindow>(new DevWindow(nullptr, ptrObjectHandler));
00053             QObject::connect(ptrDevWindow.data(), &DevWindow::returnToLauncher, this,
      &WindowHandler::showLaunchWindow,
00054                              Qt::DirectConnection);
00055             QObject::connect(ptrDevWindow.data(), &DevWindow::openSettings, this,
      &WindowHandler::showSettingsWindow,
00056                              Qt::DirectConnection);
00057         }
00058         if(ptrDevWindow != nullptr){
00059             ptrDevWindow->show();
00060         }
00061
00062     }
00063
00064     void WindowHandler::showSettingsWindow() {
00065         ptrSettingsWindow->show();
00066     }
```

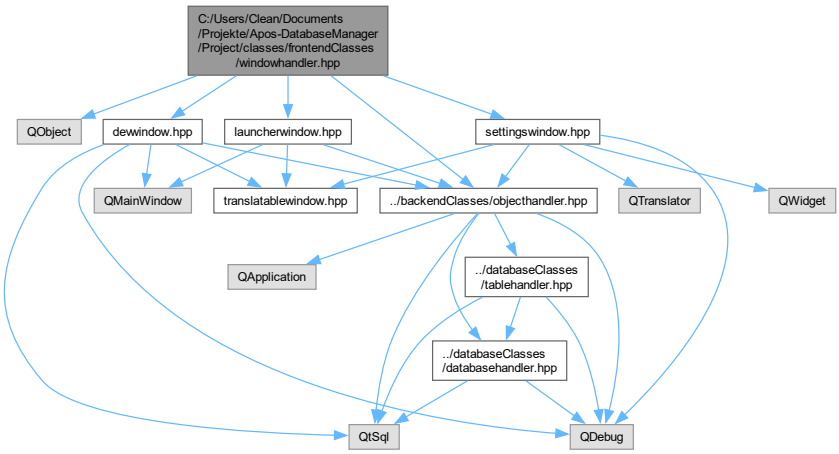
```
00067
00068     void WindowHandler::applySettings() {
00069         changeLanguages();
00070     }
00071
00072     void WindowHandler::changeLanguages() {
00073         QWidgetList openWindows = ptrObjectHandler->getPtrApplication()->topLevelWidgets();
00074         for (QWidget* widget: std::as_const(openWindows)) {
00075             auto *tw = dynamic_cast<TranslatableWindow *>(widget);
00076             if (tw != nullptr) {
00077                 qDebug() « "Dynamic cast pointer adress " « tw;
00078                 tw->retranslateUi();
00079             }
00080         }
00081         qDebug() « openWindows;
00082     }
00083 }
```

11.35 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/classes/frontendClasses/windowhandler.hpp File Reference

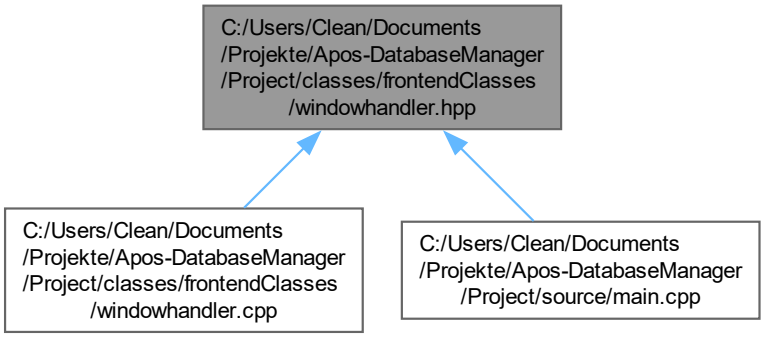
Header file for the WindowHandler class.

```
#include <QObject>
#include "launcherwindow.hpp"
#include "devwindow.hpp"
#include "settingswindow.hpp"
#include "../backendClasses/objecthandler.hpp"
```

Include dependency graph for windowhandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class AposFrontend::WindowHandler

  Provides the functionality for managing the application's windows.

Namespaces

- namespace AposFrontend

11.35.1 Detailed Description

Header file for the WindowHandler class.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the declaration of the WindowHandler class, which is a part of the application's frontend logic. The WindowHandler class provides the functionality for managing the application's windows, including the launcher window, developer window, and settings window. It interacts with the LauncherWindow, DevWindow, and Settings↩
Window classes and uses the ObjectHandler class to manage the application's objects.

Note

The application is part of a student project and is not intended for commercial use.

See also

AposBackend::ObjectHandler

LauncherWindow

DevWindow

SettingsWindow

QSharedPointer

QObject

Definition in file windowhandler.hpp.

11.36 windowhandler.hpp

Go to the documentation of this file.

```
00001
00023 #pragma once
00024
00025 #include <QObject>
00026 #include "launcherwindow.hpp"
00027 #include "devwindow.hpp"
00028 #include "settingswindow.hpp"
00029 #include "../backendClasses/objecthandler.hpp"
00030
00031
      //-----------------------------------------------------------------------------------------------------------------/
00032 namespace AposFrontend {
00048     class WindowHandler : public QObject {
00049     public:
00056         explicit WindowHandler(QSharedPointer<AposBackend::ObjectHandler> newObjectHandler);
00057
00063         void showLaunchWindow();
00064     private slots:
00070         void showDevWindow();
00071
00077         void showSettingsWindow();
00078
00084         void applySettings();
00085     private:
00091         void changeLanguages();
00092
00098         QSharedPointer<LauncherWindow> ptrLauncherWindow;
00099
00105         QSharedPointer<DevWindow> ptrDevWindow;
00106
00112         QSharedPointer<SettingsWindow> ptrSettingsWindow;
00113
00119         QSharedPointer<AposBackend::ObjectHandler> ptrObjectHandler;
00120     };
00121 }
```

11.37 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/CONTRIBUTING.md File Reference

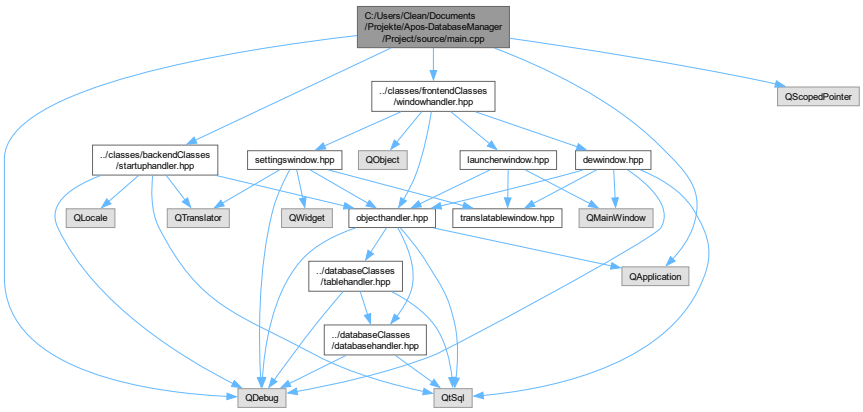
11.38 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/README.md File Reference

11.39 C:/Users/Clean/Documents/Projekte/Apos-DatabaseManager/↩ Project/source/main.cpp File Reference

Main entry point for the application.

```
#include "../classes/backendClasses/startuphandler.hpp"
#include "../classes/frontendClasses/windowhandler.hpp"
#include <QApplication>
#include <QDebug>
#include <QScopedPointer>
```

Include dependency graph for main.cpp:



Namespaces

- namespace AppInitialization

Functions

- QSharedPointer< AposBackend::StartupHandler > AppInitialization::initializeStartupHandler (const QSharedPointer< QApplication > &newApp)

  Initialize the StartupHandler object.
- QSharedPointer< AposBackend::ObjectHandler > AppInitialization::initializeObjectHandler (const QSharedPointer< AposBackend::StartupHandler > &startupHandler)

  Initialize the ObjectHandler object.
- QSharedPointer< AposFrontend::WindowHandler > AppInitialization::initializeWindowHandler (const QSharedPointer< AposBackend::ObjectHandler > &objectHandler)

  Initialize the WindowHandler object.
- int main (int argc, char *argv[])

  Main function.

11.39.1 Detailed Description

Main entry point for the application.

Author

Simon Blum

Date

13.11.2023

Version

0.1_alpha.2 @license LGPL-V3

This file contains the main function, which represents the entry point for the application. It initializes the QApplication, StartupHandler, and ObjectHandler objects. It also creates a WindowHandler object and shows the launch window.

The application is built using the Qt framework and follows the object-oriented programming paradigm. The main function initializes the necessary objects and starts the application's event loop. The QApplication object encapsulates the functionality of Qts core application class for GUI-based applications. The StartupHandler, ObjectHandler, and WindowHandler classes are part of the application's backend and frontend logic.

@deviation MISRA 3-1-3 The argv parameter in the main function is a pointer to an array of C-style strings. This array is not explicitly sized, which violates MISRA rule 3-1-3. However, the parameters of the main function are defined by the C++ standard, and changing them would not be compliant with the standard. This deviation is considered acceptable because the size of the argv array is managed by the runtime environment, and the array is guaranteed to be null-terminated. Therefore, the risk of out-of-bounds access is minimal.

@deviation MISRA 7-3-1 The main – defined as qMain – function is part of the global namespace, which violates MISRA rule 7-3-1. However, since the qMain function acts as the main function within the Qt framework, this deviation is considered acceptable.

Note

The application is part of a student project and is not intended for commercial use.

See also

QApplication

StartupHandler

ObjectHandler

WindowHandler

Definition in file main.cpp.

11.39.2 Function Documentation

11.39.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Main function.

This is the main function, which is the entry point for the application. It initializes the QApplication, ObjectHandler, and WindowHandler objects, and starts the application's event loop.

Parameters

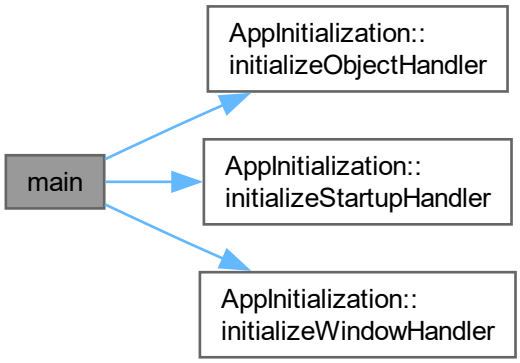
argc	Argument count
argv	Argument vector

Returns

int Application exit status

Definition at line 98 of file main.cpp.

Here is the call graph for this function:



11.40 main.cpp

Go to the documentation of this file.

```
00001
00039 // Local includes
      ------------------------------------------------------------------------------------------------ //
00040 #include "../classes/backendClasses/startuphandler.hpp"
00041 #include "../classes/frontendClasses/windowhandler.hpp"
00042 // System includes
      ------------------------------------------------------------------------------------------------ //
00043 #include <QApplication>
00044 #include <QDebug>
00045 #include <QScopedPointer>
00046 // Declaration of
      functions------------------------------------------------------------------------------------------ //
00047 namespace AppInitialization {
00057     QSharedPointer<AposBackend::StartupHandler> initializeStartupHandler(const
      QSharedPointer<QApplication>& newApp);
00058
00069     QSharedPointer<AposBackend::ObjectHandler> initializeObjectHandler(
00070             const QSharedPointer<AposBackend::StartupHandler>& startupHandler);
00071 //
      ------------------------------------------------------------------------------------------------------------------------------------
      //
00083     QSharedPointer<AposFrontend::WindowHandler>
00084     initializeWindowHandler(const QSharedPointer<AposBackend::ObjectHandler>& objectHandler);
00085 }
00086 // Implementation of functions
      ------------------------------------------------------------------------------ //
00098 int main(int argc, char *argv[]) { // NOLINT(clion-misra-cpp2008-3-1-3, clion-misra-cpp2008-7-3-1)
```

```
00099     int returnStatus = -1; // Initialize return status to -1 (error state)
00100     try {
00101         QSharedPointer<QApplication> application(new QApplication(argc, argv));
00102         qDebug() « "Application Object initialized";
00103 
00104         QSharedPointer<AposBackend::StartupHandler>
00105                 startupHandler = AppInitialization::initializeStartupHandler(application);
00106         QSharedPointer<AposBackend::ObjectHandler>
00107                 objectHandler = AppInitialization::initializeObjectHandler(startupHandler);
00108         QSharedPointer<AposFrontend::WindowHandler> windowHandler =
    AppInitialization::initializeWindowHandler(
00109                 objectHandler);
00110 
00111         returnStatus = QApplication::exec(); // Update return status
00112     } catch (const std::exception &e) {
00113         qDebug() « "Exception caught in main: " « e.what();
00114     }
00115     return returnStatus; // Single point of exit
00116 }
00117 //
    ----------------------------------------------------------------------------------------------------------------
    //
00118 namespace AppInitialization {
00119 
    //-------------------------------------------------------------------------------------------------------------//
00120     QSharedPointer<AposBackend::StartupHandler> initializeStartupHandler(const
    QSharedPointer<QApplication>& newApp) {
00121         if (newApp == nullptr) {
00122             throw std::runtime_error("QApplication pointer is null");
00123         }
00124 
00125         QSharedPointer<AposBackend::StartupHandler> startupHandler(new
    AposBackend::StartupHandler(newApp));
00126         qDebug() « "StartupHandler Object initialized";
00127         return startupHandler;
00128     }
00129 
    //-------------------------------------------------------------------------------------------------------------//
00130     QSharedPointer<AposBackend::ObjectHandler>
00131     initializeObjectHandler(const QSharedPointer<AposBackend::StartupHandler>& startupHandler) {
00132 
00133         QSharedPointer<AposBackend::ObjectHandler> objectHandler(startupHandler->startUp());
00134         if (objectHandler == nullptr) {
00135             throw std::runtime_error("Failed to initialize ObjectHandler");
00136         }
00137         qDebug() « "ObjectHandler Object initialized";
00138 
00139         return objectHandler;
00140     }
00141 
    //-------------------------------------------------------------------------------------------------------------//
00142     QSharedPointer<AposFrontend::WindowHandler>
00143     initializeWindowHandler(const QSharedPointer<AposBackend::ObjectHandler>& objectHandler) {
00144         if (objectHandler == nullptr) {
00145             throw std::runtime_error("ObjectHandler pointer is null");
00146         }
00147 
00148         QSharedPointer<AposFrontend::WindowHandler> windowHandler(new
    AposFrontend::WindowHandler(objectHandler));
00149         windowHandler->showLaunchWindow();
00150         qDebug() « "After DevWindow Show";
00151 
00152         return windowHandler;
00153     }
00154 }
00155 // End of file main.cpp
    ------------------------------------------------------------------------------------------ //
00156 // Doxygen-Groups
    ------------------------------------------------------------------------------------------------ //
00157 
```

Index