

# MiloDB Client

## Dev Guide

FrozenWolf

### Table of Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	File Structure .....	2
1.1.1	Root Directory .....	2
1.1.2	Build Directory .....	2
1.2	Setup Virtual Environment .....	2
1.2.1	Create Environment .....	3
1.2.2	Activate Environment .....	3
1.2.3	Install Packages .....	3
1.2.4	Verify Setup .....	4
<b>2</b>	<b>Execution</b>	<b>5</b>
2.1	Run Scripts .....	5
2.2	Building the Client .....	5

# 1 Introduction

MiloDB is a Python application intended to run on Windows and Linux, and development can also be done on either Windows or Linux. There shouldn't be any constraints that prevent the application running or being developed under other operating systems, but those are currently not supported.

This guide is intended for developers and assumes you have unpacked the source package to a suitable location.

## 1.1 File Structure

The `milodb-src` directory will be referred to as the 'top' directory throughout this guide.

```

milodb-src/
├── build/ ..... Output directory for all build artefacts
├── root/ ..... Working directory for execution
│   ├── default-dark.css ..... Style sheet when generating summaries
│   └── default-light.css ..... (default) Style sheet when generating summaries
├── src/
│   ├── entry_*.pyw ..... Entry point scripts for each application
│   ├── milodb_*/ ..... Main application source code
│   └── milodb*_test/ ..... Unit tests
├── tools/
│   ├── __init__.py ..... Python tools package definition
│   ├── milodb-*.spec ..... pyinstaller specification files
│   ├── requirements-*.txt ..... Third-party Python package dependencies
│   ├── svg_to_data_uri.py ..... Converts an SVG file to an HTML embeddable image data URI
│   └── svg_to_ico.sh ..... Converts an SVG file to a Windows icon
├── run*.py ..... Shortcut scripts to run entry point scripts from root
├── build_client.py ..... Builds the client application for the current platform
├── cspell* ..... Spelling configuration
├── LICENSE.txt ..... Source code license
├── milodb-changelog.md ..... Source code change log
├── milodb-developer-guide.pdf ..... This document
└── pyproject.toml ..... Python project configuration

```

### 1.1.1 Root Directory

The `root` directory is where the client applications expect to find the database and configuration files when run, therefore copying (or soft-linking) the database files into the `root` directory is recommended. This is done to reduce the clutter of configuration files etc. that can clog up the top directory.

See section 2.1-Run Scripts for more information.

### 1.1.2 Build Directory

The `build` directory is where all of the build output goes, including the executables, database, documentation, and source and binary distributions. If the `build` directory doesn't exist, it will be created automatically. The `build` directory can be deleted at any time.

## 1.2 Setup Virtual Environment

Use of a virtual environment is strongly recommended to allow you to install Python packages in isolation from your main Python installation, and also to ensure you have specific versions of packages. Unless otherwise stated, all commands given here are expected to be run from the top directory.

### 1.2.1 Create Environment

Creating the virtual environment only has to be done once, after that it can be reused. Create it as follows:

Windows / Linux

```
python -m venv .venv
```

This uses the `venv` module to create a new directory, `.venv` for the virtual environment. The virtual environment contains a copy of Python and no third-party packages. The directory doesn't have to be called `.venv`, it can be anything you like, but this document will assume it's `.venv`. Similarly, the directory doesn't have to be in the top directory, it can be anywhere you prefer.

### 1.2.2 Activate Environment

Now activate the virtual environment:

Windows

```
.venv\Scripts\activate.bat
```

Linux

```
source .venv/bin/activate
```

This activation *only* affects the current shell/terminal by changing its environment so that Python and PIP now run from within the virtual environment directory `.venv`, and third-party package installations will also go into `.venv`. When starting a new shell/terminal, you need to run the activate script again.

The `.venv` directory can be deleted if you want to clean out the packages you've installed into it, or use a different Python version, after which you can create another one.

### 1.2.3 Install Packages

MiloDB depends on external Python packages, all of which are listed under `tools/requirements-*.txt`.

You don't need to install the build packages unless you explicitly want to build the binaries manually. See section 2.2-Building the Client for more information.

For the sake of simplicity, install all of the non-build requirements in one go with `pip`:

Windows

```
pip install -r tools\requirements-run.txt -r tools\requirements-develop.txt
```

Linux

```
pip install -r tools/requirements-run.txt -r tools/requirements-develop.txt
```

### 1.2.4 Verify Setup

You should now be able to run all of the `run_*.py` scripts in the top directory, for example:

Windows

```
python run_gui.py
```

Linux

```
./run_gui.py
```

## 2 Execution

### 2.1 Run Scripts

All of the `run_*.py` scripts in the top directory are convenience scripts that change the current working directory to `root` and then execute the associated script from `src/entry_*.py`. See section 1.1.1-Root Directory for why this is done.

For example, the following:

Is equivalent to:

Windows

```
python run_gui.py
```

Windows

```
cd root
python ../src/entry_gui.py
cd ..
```

Linux

```
./run_gui.py
```

Linux

```
cd root
../src/entry_gui.py
cd ..
```

### 2.2 Building the Client

To build the client for the current operating system, run the following:

Windows

```
python build_client.py
```

Linux

```
./build_client.py
```

This build script doesn't depend on the virtual environment described in section 1.2-Setup Virtual Environment as it creates a clean one using the appropriate `requirements-*.txt` files. All outputs from the build script go into the `build` directory.

To verify that a built executable can be run successfully, it is recommended to run it relative to the `root` directory as follows:

Windows

```
cd root
../build/bin/milodb-gui.exe
```

Linux

```
cd root
../build/bin/milodb-gui
```