Technical Module Description: MPDControl.py

Version: 1.0 Date: January 10, 2025

Authors: Henk Stevens, Olaf Mastenbroek, Onno Janssen

Organization: Stichting Oradio

License: GNU GPL

# Module Purpose

The `MPDControl.py` module provides a Python-based interface for managing music playback on the Music Player Daemon (MPD) system, used in the Oradio device. It offers robust playback control, safe connection handling, and playlist management features. The module includes error handling, automatic reconnection, and can operate as both an importable library and a standalone interactive test tool for developers.

# Main Features

* Safe MPD client connection with automatic reconnection and error monitoring.
* Playback control: play, pause, stop, next track, and preset selection.
* Dynamic playlist selection from stored playlists or directories.
* Database update functionality with cancellation and timeout handling.
* Support for searching songs by artist or title pattern.
* Play a single song immediately after the current song and remove it after playback.
* Service restart functionality for MPD.
* Standalone interactive mode with menu-based testing for developers.

# Key Components and Flow

## Initialization (\_\_init\_\_)

Sets up MPD connection parameters and starts a background thread to maintain the connection and handle automatic reconnection.

## \_connect()

Handles the actual connection process to the MPD server and sets the crossfade parameter to 5 seconds on each connection.

## Playback Controls

Includes methods `play()`, `pause()`, `stop()`, `next()`, and `play\_preset(preset)` for controlling playback. Presets are resolved from the configuration file containing playlist names or directory paths.

## Database Management

`update\_mpd\_database()` handles database updates with timeout and cancellation support. Can be executed in a separate thread via `start\_update\_mpd\_database\_thread()`.

## Song Management

`play\_song(song)` inserts a specific song into the queue for immediate playback and removes it after completion.

## Playlist and Directory Listing

`get\_lists()` retrieves all available stored playlists and directories. `get\_songs(list)` lists the songs in a given playlist or directory.

## Search Functionality

`search(pattern)` finds songs by matching artist or title attributes against a given pattern.

## Service Management

`restart\_mpd\_service()` restarts the MPD service using systemctl for system-level recovery.

## Connection Maintenance

The `\_maintain\_connection()` thread checks connection health and logs status or errors at regular intervals.

# Usage Example (Standalone Test Mode)

When executed directly (`python3 MPDControl.py`), the module starts an interactive test mode. This mode provides a text-based menu allowing manual playback control, playlist selection, database updates, song searching, and stress testing the MPD command interface.