# Using Connection Handlers

James P. Gilbert

2024-07-18

## Contents



## Introduction

`ConnectionHandler` classes are R6 instances that intent to provide consistent manners to connect to relational database instances on top of `DatabaseConnector` utilities. These are designed for long running applications, such as Plumber APIs or Shiny applications and would generally be encapsulated in other objects (such as a DataMigrationManager).

## Basic usage

### Creating an instance

Creating a connection handler only requires a connection details object

```
connectionDetails <- DatabaseConnector::createConnectionDetails("sqlite", server = "MyDb.sqlite")
connectionHandler <- ConnectionHandler$new(connectionDetails)
```

### Pooled connections

In applications such as Shiny apps that require many long running or concurrent requests, pooled connections are often required. In this case, a polled connection handler should be used. Instantiation is similar to above:

```
connectionDetails <- DatabaseConnector::createConnectionDetails("sqlite", server = "MyDb.sqlite")
connectionHandler <- PooledConnectionHandler$new(connectionDetails)
```

These classes should behave identically in terms of queries as they implement a common set of functions. See the `pool::dbPool` class for information regarding pooled connections.

### Querying a database

Submitting queries to a database is straightforward and uses `SqlRender` parameterization, for example:

```
result <- connectionHandler$queryDb("SELECT * FROM my_table WHERE id = @id", id = 1)
```

Similarly, SQL execution can occur

```
result <- connectionHandler$executeSql("CREATE TABLE foo (id INT);")
```

Note that the above queries render and translate using `SqlRender` functionality. Should direct querying or execution be required the functions `connectionHander$queryFunction` and `connectionHandler$executeFunction` can be used, respectively.