Database Migrations

Contents

Migrations

By default Luminus uses the Migratus library for database migration and schema management. When you select the `+mysql`, or `+postgres` profiles a migrations configuration will be added to the `project.clj` file in your application.

Migrations with Migratus

The development database configuration should placed in the `dev-config.edn` file. This file specifies your local configuration and should **not** be checked into the source repository.

```
{:database-url "jdbc:postgresql://localhost/myapp_dev?user=appuser&pas
```

For production, the configuration is expected to be present in the environment. An example would be to have a shell variable called `DATABASE_URL` that points to the URL. See the official cprop documentation for the complete list of configuration options.

Migratus is invoked from the `-main` function in the `<app>.core` namespace of the application.

By default, the SQL migration scripts are expected to be found in the `resources/migrations` directory in the root of the project. A custom directory can be set in the `<app>.db.migrations` namespace. With the directory created we can start adding our migrations SQL scripts there.

Build Tool: lein

Topics

Your First Application
REPL Driven Developme
Application Profiles
HTML Templating
Static Assets
ClojureScript
Routing
RESTful Services
Request types
Response types
Websockets
Middleware
Sessions and Cookies
Input Validation
Security
Component Lifecycle
Database Access
• **Database Migrations**
Logging
Internationalization
Testing
Server Tuning
Environment Variables
Deployment
Useful Libraries
Sample Applications
Upgrading
Clojure Resources

Books

Migration ids are not assumed to be incremented integers and are considered for completion independently. The recommended way to keep migrations ordered is by prefixing the current date to the name of the script.

Before migrations can be run, we have to ensure that the database connection has been started. By default, the connection is defined in the `<app>.db.core` namespace as:

```
(defstate ^:dynamic *db*
  :start (conman/connect! {:jdbc-url (env :database-url)})
  :stop (conman/disconnect! *db*))
```

When the `<app>.db.core` namespace has to be referenced in the proejct for the connection will be started automatically. To start the connection manually, run the following command in the REPL:

```
(mount.core/start  #'<app>.db.core/*db*)
```

Let's create two scripts, one for the migration and the other for the rollback. The files can be generated from the REPL as follows:

```
(user/create-migration "add-users-table")
```

This will generate the appropriate files in the migrations directory. We'll update the `up` migrations file with the script to create the table:

`resources/migrations/201506120401-add-users-table.up.sql`

```
CREATE TABLE users (id INT, name VARCHAR(25));
```

Then we'll update the corresponding `down` migrations file with script to drop it:

`resources/migrations/201506120401-add-users-table.down.sql`

```
DROP TABLE users;
```

The migrations can now be invoked by running:

```
lein run migrate
```

Rolling back is done by running:

```
lein run rollback
```

To apply specific migrations, run the above commands with the desired migrations ids:

```
lein run migrate 201506104553 201506120401
lein run rollback 201506104553 201506120401
```

The migrations will be packaged in the applications when it's compiled. This allows the application to apply its own migrations when deployed to the server:

```
lein uberjar
java -jar target/uberjar/<app>.jar migrate
```

Popular Migrations Alternatives

Below is a list of all the popular migration libraries that are currently available:

Drift - Drift is a migration library written in Clojure. Drift works much like Rails migrations where a directory in your project contains all of the migration files. Drift will detect which migration files need to be run and run them as appropriate.
Ragtime - a general migration framework, with an implementation for database migrations.
Joplin - Joplin is a library for flexible datastore migration and seeding