DB migration server

The purpose of this server is to do db migration for version changes of the database. As an overview, it will contain a table to track database migrations. It is packaged in a docker server that can be accessed via web. When the service is invoked, it will locate the folders specifed in its registry, and executes the files located in the migrations folder in order. The executions are logged in its system database and it will only execute files starting from the latest file.

This should be called manually everytime we have pull updates. The migration files will be included in the path of the module under the migrations folder. The first filename for a module script will be the main sql script that builds the database. So we can install a module the first time by invoking this service.

This service should be simple and would only work one way, i.e. we do not apply rollbacks or migration down executions in case of errors. If an error is found, we should not continue, thats why instead of doing a rollback, we just restore a backup if it cant be helped. It is assumed that before deploying this to the client, one must have an exact copy of the db structure of the client and test in a staging area before deploying to test if the scripts are ok.

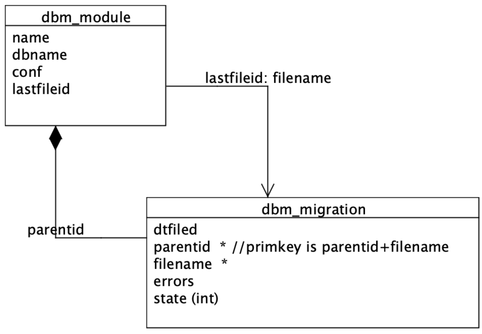
As to the strategy on how to migrate files, whether to keep the column or drop the column, we will leave it to the developer of the module or who is in charge for migrating databases.

Filenames must follow a convention: Recommended is the ff: format

```
```[yyyy][MM]-[0%4d]-descriptivename.[sql|js|svc]```
```

## Database Structure

The ff. is the database structure:




**dbm_module**

| field | description |
|---|---|
| name | name of the module. also corresponds to the name of the folder where migrations are located |
| dbname | name of the database. |
| conf | the database configuration setting in json or any complex format. contains db url, username and pwd |
| lastfileid | links to dbm_migration. the last line executed |

**dbm_migration**

| field | description |
|---|---|
| parentid | links to dbm_module.name |
| filename | name of file |
| dtfiled | date executed |
| errors | log errors here if any |
| state | 0=unprocessed 1=processed |

## How it works

1. After a pull update is done, access the server via url example: http://localhost:5000/dbmigrations/build or http://localhost:5000/dbmigrations/build/modulename to build a specific module
2. The server loads dbm_module table and gets the module names.
3. For each modulename, the server scans for files in its local folder called **dbm-root** : /dbm-root/
4. For each module folder, it scans files under the migrations folder, gets all the filenames and insert ignore it in **dbm_migration** table.
5. From the dbm_module table, get the lastfileid and select items where filename > lastfiledid. If the lastfileid is null, it will get all filenames.
6. For each filename in No. 5, run the processor. The processor is a service that executes the contents of the file. Each file is identified by its extension as follows ( we can add as many handlers in the future):

- .sql = executes database files
- .js = executes js
- .sv = executes a service (might be for updating rules)

7. Once a file is successfully executed, it updates the dbm_module.lastfileid and updates dbm_migration.state to 1 and dtfiled to now.
8. If there are errors, errors are logged in the dbm_module and the process is stopped. It will be displayed in the browser

## Sample docker-compose and folder

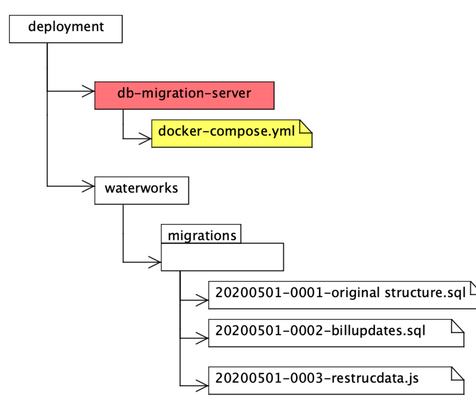
```
version: "3"

services:
  db-migration-server:
    image: ramesesinc/db-migration-server:0.0.1
    container_name: db-migration-server
    restart: always
    logging:
      driver: "json-file"
      options:
        max-file: "5"
        max-size: 10m

    environment:
      TZ: "Asia/Manila"

    ports:
      - "5000:5000"

    volumes:
      - ../waterworks2/migrations:/dbm-root/waterworks2/migrations
```



Sample script

```
USE ${dbname};

CREATE TABLE ....
```
```