Contents

[Create DAO 1](#_Toc57538968)

[Creating a server that implements the REST API 2](#_Toc57538969)

[Creating the html that will use AJAX to link to the server and provide a user interface 3](#_Toc57538970)

[Configuration File 3](#_Toc57538971)

[IN THE READMEFILE 3](#_Toc57538972)

# Create DAO

Create table on datarepresentation database, on mysqlcommand line

use datarepresentation;

create table stock2(

    -> id INT AUTO\_INCREMENT PRIMARY KEY,

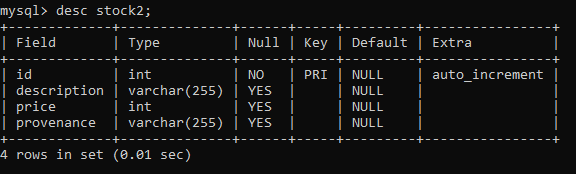
    -> description varchar(255),

    -> price int,

    -> provenance varchar(255)

    -> );

Save these commands as initdb.sql



Use code from <https://github.com/andrewbeattycourseware/dataRepresenation2020>

Provide and Test functions.

The test document is used for this purpose but it is not a required piece of the project submission.

Create 🗸 (print("\*Return from Create function is\*",(str(returnValueCreate))) to the screen

Get All 🗸 print("\*getAll output from the stock2 table is as follows\*\n",(str(returnValuegetAll)))

findById🗸 it works like this when you input the id on the testdocument

result = stockDao.findById(111);

print (result)

Update The function works but it doesn’t get put onto the table unless Create is used. The current coding only allows one (hardcoded?) item at a time to be tested.

returnValue = stockDao.update(item)

print("\*after update function\*",(str(returnValue)))

\*after update function\* {'id': 112, 'description': '6 Free Range Eggs', 'price': 600, 'provenance': 'Magner Farm'}

Delete X Not working!! **Fix that up**

When you run this, it runs alright but I can’t see that it actually does anything!

returnValue = stockDao.delete(item['id'])

print("\*after delete\*",(str(returnValue)))

# [Creating a server that implements the REST API](https://web.microsoftstream.com/video/2748d036-41fd-433a-a9be-938bff2dd9f7?list=studio)

Start by recycling a server.py programme

The server programme should be simple, with the “heavy lifting” carried out in the DAO

Leave out all the fancy stuff and just test using curl that the basic functions (that don’t really do anything yet) run. This is to check the interface works. It takes up about half of the instructional video!

(I called this version serverBasic)

Then link a version to update (I called it server.py) to the DAO using the same code as is in the first line of test

from BookDAO import bookDao

OR

from stock2DAO import stockDao

Replace like this

return jsonify({}) becomes return jsonify(bookDao.create(book))

Check them all again with that additional link

getAll 🗸 curl http://127.0.0.1:5000/stock

Create 🗸curl -X POST -d "{\"description\":\"Apple Juice Box\", \"price\":1250, \"provenance\":\"The Apple Farm, Cahir\"}" -H "Content-Type:application/json" http://127.0.0.1:5000/stock

Update 🗸 curl -X PUT -d "{\"description\":\"new lad\", \"price\":567, \"provenance\":\"Cashel\"}" -H "Content-Type:application/json" http://127.0.0.1:5000/stock/115

findByID 🗸 curl <http://127.0.0.1:5000/stock/109>

Delete curl -X DELETE <http://127.0.0.1:5000/stock/115>

# Creating the [html](https://learnonline.gmit.ie/mod/url/view.php?id=64740) that will use AJAX to link to the server and provide a user interface

Working along with the lecture video 9.6

-user interface needs to have Display Table on one page, Edit screen on the other

Id is my unique key so don’t enter that when you are creating a new item- it is generated from the database via the server.

Designed the page

Ajax call to the database

When server.py is running the abindex.html is linked to it

body {

  background-image: url('pots.jpg');

  background-repeat: no-repeat; -->

Launch <http://127.0.0.1:5000/index.html> to see your db contentspdate

It is working with functions to

Create a new stock item

Update existing stock items

Delete

Crud, baby!

# Configuration File

This file will allow to STORE environment specific variables outside of my code so the code can run on different machines.

I can’t control where the application is run e.g. a colleague’s machine which is not a PC like mine

The purpose of the configuration file is to create the initial settings for the project, thus avoiding hardcoded data. My colleague (for example) can amend this file to include their own local settings (e.g., user name and password) before proceeding.

Similar to requirements.txt – what is needed is set out clearly in a separate file and the code can just do it’s thing.

The simplest way to write one is simply to write a separate file that contains Python. Make sure to add the name of this file to the .gitignore file to avoid uploading it accidentally…or bad things will happen to the person that downloads it… their own configuration files will be destroyed. There are lots of different ways to do this. Other formats are json, yaml, ini, xml.

cd C:\ALL MY STUFF\GMIT\dataRepresentation\dataRepresentation\Week8

.\venv\Scripts\activate.bat

(venv) λ set FLASK\_APP=server

(venv) λ set FLASK\_ENV=development

(venv) λ flask run

Running on <http://127.0.0.1:5000/>

λ cd C:\ALL MY STUFF\GMIT\dataRepresentation\dataRepresentation\Week9\Walkthrough

server.py

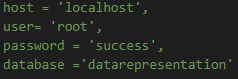
Running on <http://127.0.0.1:5000/>

cd C:\ALL MY STUFF\GMIT\dataRepresentation\ProjectPrep

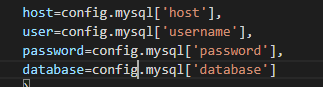
1.Make a config.py file that has all the credentials needed to access the mysql database

2. Amend the DAO so that it calls the contents of that file instead of having host/user/password/database details hardcoded in.

It looked like this



Now it looks like this



# In the .gitignore file

Anything that should not be read including

/venv (confirmed!)

\*config.py

# In the README file

* description the .sql file that has a create command for your table(s).
* Link to a hosted version of your application.
* An [overview](https://learnonline.gmit.ie/mod/url/view.php?id=64757) that has any other information you think I need.