Domingo Jakubowski
3 years ago

Introduction

Note: This is a tutorial divided into 3 parts:

1. *Installing Go*
2. *TDD with Go and PostgreSQL*
3. *Database queries on Go with PostgreSQL*

What are we going to build?

The scope of this project is to build from scratch (Literally, from creating the git repository) an example of REST API with Go, including routing, database-fetching (PostgreSQL) and, to start with the right foot, via Test-Driven Development.

Starting PostgreSQL with Docker

First of all, we need to have PostgreSQL installed on your machine. I prefer to use a Docker container with PostgreSQL. Installing a database is a different topic itself, so I will explain shortly how to start a Docker container with a PostgreSQL database:

```
docker run -d -p 5432:5432 --name my-postgres -e POSTGRES_PASSWORD=12345 post
```

To access the database and terminal, we use the following command:

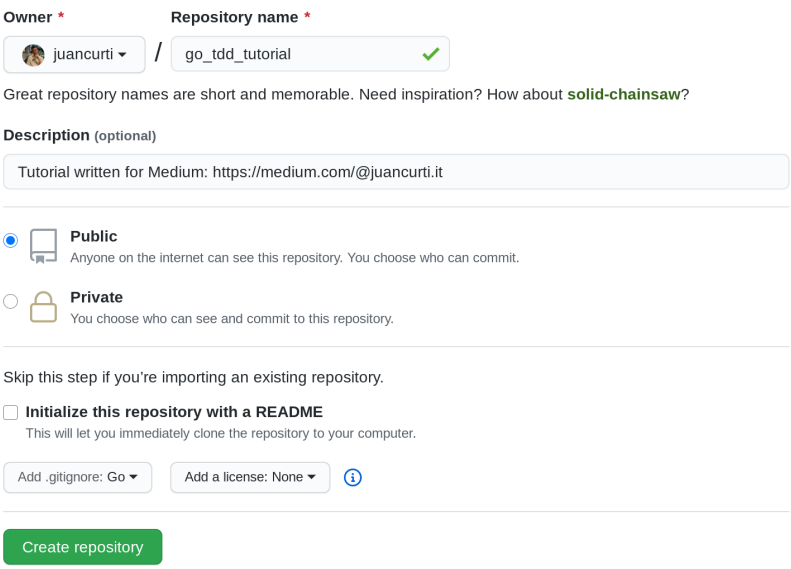
```
sudo docker exec -it my-postgres bash
psql -U postgres
```

If we want to connect locally to the PostgreSQL command line, we use:

```
psql -h localhost -p 5432 -U postgres -W
```

Let's go with Go

We will start creating the Git repository, including the Go .gitignore. I will be using Github.com as a personal choice.



The next step is to clone our git repository, in my case:

```
git clone https://github.com/juancurti/go_tdd_tutorial.git
```

Then, we need to initialize our Go repository, needed in our root folder. This will create a go.mod. Type the following command replacing GITREPOSITORY with your own, without the 'https://' prefix:

```
go mod init github.com/GITREPOSITORY
```

Now we need to fetch the Go modules we're going to use, in this case Gorilla Mux and PostgreSQL.

```
go get -u github.com/gorilla/mux
go get -u github.com/lib/pq
```

```
juancurtiit@penguin:~/tutorials/go_tdd_tutorial
$ go get -u github.com/gorilla/mux
go: downloading github.com/gorilla/mux v1.7.4
go: github.com/gorilla/mux upgrade => v1.7.4
juancurtiit@penguin:~/tutorials/go_tdd_tutorial
$ go get -u github.com/lib/pq
go: downloading github.com/lib/pq v1.7.0
go: github.com/lib/pq upgrade => v1.7.0
juancurtiit@penguin:~/tutorials/go_tdd_tutorial
$
```

Let's start by creating an app.go file for our application. In this file, we will define a struct (App) to hold a reference to the router and the database. To be useful and testable, our App struct will need two methods that initialize and run the application.

```
package main

import (
 "database/sql"
"github.com/gorilla/mux"
 _ "github.com/lib/pq"
)
type App struct {
 Router *mux.Router
 DB *sql.DB
}
func (a *App) Initialize(user, password, dbname string) { }
func (a *App) Run(addr string) {}
```

The Initialize method takes the parameters needed to establish a connection with the PostgreSQL database, while the Run method starts the application.

We will also create a main.go file, the entry point of the application, with the following code:

```
package main

import "os"
func main() {
 a := App{}
 a.Initialize(
   os.Getenv("APP_DB_USERNAME"),
   os.Getenv("APP_DB_PASSWORD"),
   os.Getenv("APP_DB_NAME"),
 )
 a.Run(":8001")
}
```

To secure our code, and most programmers will agree, we must avoid exposing credentials within our code, which will be pushed to Github and be (potentially) available to millions. We will use environment variables. We will export, then, our credentials typing the following commands in the terminal:

```
export APP_DB_USERNAME=postgres
export APP_DB_PASSWORD=
export APP_DB_NAME=postgres
```

Lastly, creating a CRUD application, we need something to be requested. We will use a generic 'Product' for the sake of the tutorial. For this, we will add a model.go file with the following:

```
package main

import (
 "database/sql"
)
type product struct {
 ID int `json:"id"`
 Name string `json:"name"`
 Price float64 `json:"price"`
}
func (p *product) getProduct(db *sql.DB) error {
 return errors.New("Not implemented")
}
func (p *product) updateProduct(db *sql.DB) error {
 return errors.New("Not implemented")
}
func (p *product) deleteProduct(db *sql.DB) error {
 return errors.New("Not implemented")
}
func (p *product) createProduct(db *sql.DB) error {
 return errors.New("Not implemented")
}
func getProducts(db *sql.DB, start, count int) ([]product, error) {
 return nil, errors.New("Not implemented")
}
```

Now that we have the base models created, we can start writing our tests.

To start testing as we develop, we can start running tests against the database, specifically to ensure it is properly set up. We will use a TestMain function in a main_test.go file, and use an "a" variable that represents the application we want to test. We will write the following code, and then I will explain what we've just written:

```
package main

import (
 "os"
 "testing"
 "log"
)
var a App
func TestMain(m *testing.M) {
 a.Initialize(
 os.Getenv("APP_DB_USERNAME"),
 os.Getenv("APP_DB_PASSWORD"),
 os.Getenv("APP_DB_NAME"),
)
ensureTableExists()
 code := m.Run()
```

```
 clearTable()
 os.Exit(code)
}
func ensureTableExists() {
 if _, err := a.DB.Exec(tableCreationQuery); err != nil {
 log.Fatal(err)
 }
}
func clearTable() {
 a.DB.Exec("DELETE FROM products")
 a.DB.Exec("ALTER SEQUENCE products_id_seq RESTART WITH 1")
}
const tableCreationQuery = `CREATE TABLE IF NOT EXISTS products
(
 id SERIAL,
 name TEXT NOT NULL,
 price NUMERIC(10,2) NOT NULL DEFAULT 0.00,
 CONSTRAINT products_pkey PRIMARY KEY (id)
)`
```

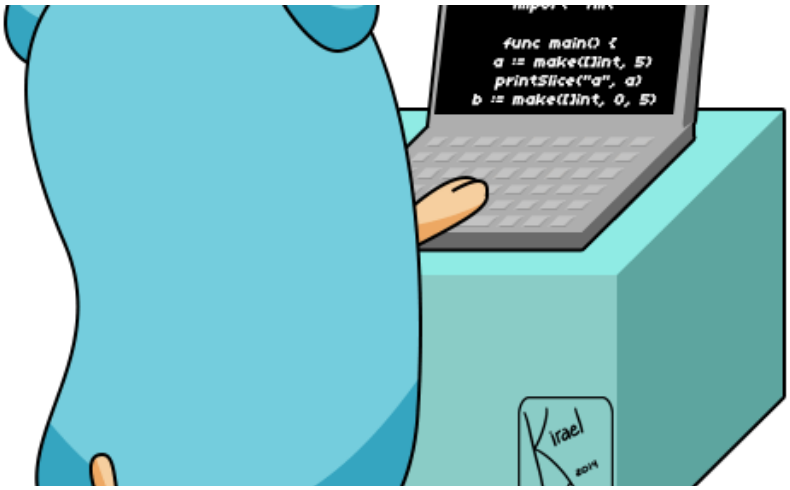
On the function "ensureTableExists", we make sure the table we want to test is available. In order to do this, we created a constant with the creation of the SQL script.

All our tests are executed by calling m.Run(), after which we call clearTable() to clean the database up.

Finally, we need to implement the Initialize method of App in app.go, to establish a connection with the database and initialize the router.

We need to add the "fmt" and "log" first in order to start with our initialize method.

#api #tutorial #google #backend #go



ITNEXT.IO
Go Tutorial: TDD with Go and PostgreSQL [Part II]
TDD with Go and PostgreSQL

1.85 GEEK

Eclipse Plugin

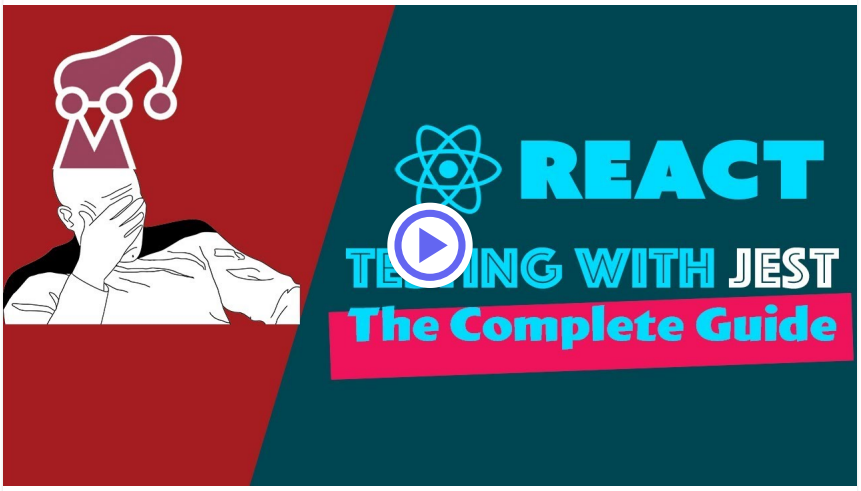
Contact www.jr-database-tools.c

www.jr-database-tools.com

Domingo Jakubowski
2 years ago

Complete Guide to Component testing with Jest for beginners. Crash course on Jest and mocking

#react #jest #testing



YOUTUBE.COM
React Unit Testing with Jest & React-testing-library
Complete Guide to Component testing with Jest for beginners. Crash course on Jest and mockin...

11.80 GEEK

Domingo Jakubowski
2 years ago

> *"Learn how to integrate the FlutterFire plugins Authentication, Cloud Firestore, Remote Config, Crashlytics, and Analytics in a Flutter app."*

Firebase helps you develop, measure, improve, and grow your mobile app. It's backed

BLOG.LOGROCKET.COM

Add Firebase to Your Flutter App with FlutterFire Plugins

Learn how to integrate the FlutterFire plugins Authentication, Cloud Firestore, Remote Config, Cr...

7.95 GEEK

Domingo Jakubowski

2 years ago

> *"With Vue Native, developers can get the best of both the Vue.js and React Native ecosystems when building apps."*

Vue Native is a JavaScript framework designed to build cross-platform mobile

BLOG.LOGROCKET.COM

Building Mobile Apps with Vue Native

With Vue Native, developers can get the best of both the Vue.js and React Native ecosystems wh...

6.10 GEEK

Domingo Jakubowski

2 years ago

> *"Follow this guide to implement the Stripe payment system in your React Native apps with the new Stripe React Native SDK."*

Although Stripe is only the second most popular payment gateway (PayPal remains

BLOG.LOGROCKET.COM

How to Implement the Stripe Payment System in React Native Apps

Learn how to implement the Stripe payment system in your React Native apps with the new Strip...

4.65 GEEK

Domingo Jakubowski
2 years ago

In **TypeScript**, *enums*, or enumerated types, are data structures of constant length that hold a set of constant values. Each of these constant values is known as a *member* of the enum. Enums are useful when setting properties or values that can only be a certain number of possible values. One common example is the suit value of a single

DIGITALOCEAN.COM

How To Use Enums in TypeScript

Learn how to use Enums in TypeScript. We will explain the syntax used to create enum types, the ...

3.90 GEEK

Domingo Jakubowski
2 years ago

Sign Data using Symmetric-key Algorithm Encryption

ska

Lets you easily sign data, using symmetric-key algorithm encryption. Allows you to

3.40 GEEK

Domingo Jakubowski
2 years ago

The performance of your tools, and especially your IDE, has a significant impact on your development experience. For a long time, performance was a pain point for Kotlin developers, especially those who worked with large codebases or with complex code structures. This is why for the past year we've been focusing on improving this

BLOG.JETBRAINS.COM

Kotlin IDE Performance
Kotlin IDE Performance. Performance was a pain point for Kotlin developers. This is why for the p...

3.20 GEEK

Domingo Jakubowski
2 years ago

CSS has default keywords for various values. In this article I'm going to talk about three of them: `initial`, `inherit`, and the relatively new one, `unset`.

There's a good chance that although most web developers have encountered them, many of even the most experienced ones don't fully understand them

ELAD.MEDIUM.COM
Understanding the "Initial", "Inherit" and "Unset" CSS Keywords
CSS has default keywords for various values. In this article I'm going to talk about three of them: ...

2.95 GEEK

Domingo Jakubowski
2 years ago

> *"Improve your programming skills by creating games using Python."*

C++ is one of the best programming languages for game development. Being an interpreter based programming language, Python is not much preferred for game

MEDIUM.COM

Game Development Projects using Python

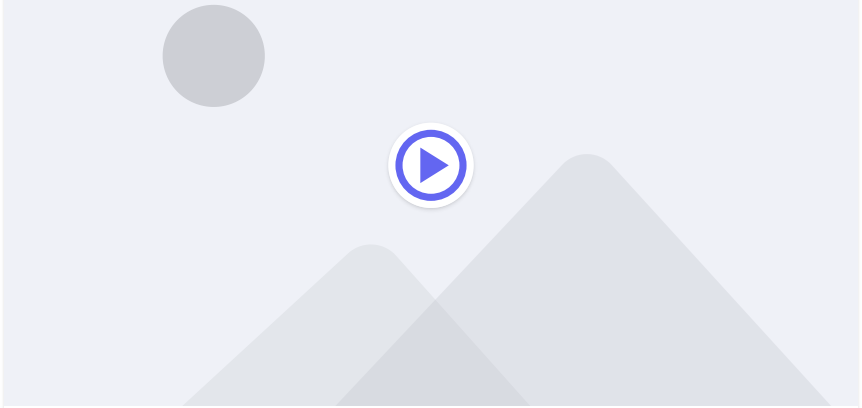
Improve your programming skills by creating games using Python. I will take you through some ...

2.70 GEEK

Domingo Jakubowski
2 years ago

In this episode, I take a look at the SvelteKit examples. I explore each, walking through their new code. We see a few bugs and work in progress spots. As SvelteKit is in public beta, this is to be expected.

Timestamps:



YOUTUBE.COM

First Look at the SvelteKit Examples! SvelteKit Public Beta and Svelte

In this episode, I take a look at the SvelteKit examples. I explore each, walking through their new ...

6.45 GEEK