In this tutorial, we are going to see an example program to learn how to do database CRUD operations using Golang and MySQL. CRUD is an acronym for Create, Read, Update, and Delete. CRUD operations are basic data manipulation for database.  
  
In this example, we are going to create an interface as database front end to handle these operations. We have Employee table containing Employee information like id, name and city. With this table, we have to perform CRUD using MySQL.

**Step 1: Prepare and Import MySQL driver into your project**

Using Git Bash first install driver for Go's MySQL database package. Run below command and install MySQL driver's

go get -u github.com/go-sql-driver/mysql

Now create Goblog Database

1. Open PHPMyAdmin/SQLyog or what ever MySQL database management tool that you are using.  
2. Create a new database "goblog"

**Step 2: Creating the Employee Table**

Execute the following SQL query to create a table named Employee inside your MySQL database. We will use this table for all of our future operations.

DROP TABLE IF EXISTS `employee`;

CREATE TABLE `employee` (

`id` int(6) unsigned NOT NULL AUTO\_INCREMENT,

`name` varchar(30) NOT NULL,

`city` varchar(30) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=latin1;

**Step 3: Creating Struct, Handler and Handler Function**

Let's create a file named main.go and put the following code inside it.  
We usually import database/sql and use sql to execute database queries on the database.  
Function dbConn opens connection with MySQL driver.  
We will create Employee struct that has following properties: Id, Name and City.

package main

import (

"database/sql"

"log"

"net/http"

"text/template"

\_ "github.com/go-sql-driver/mysql"

)

type Employee struct {

Id int

Name string

City string

}

func dbConn() (db \*sql.DB) {

dbDriver := "mysql"

dbUser := "root"

dbPass := "root"

dbName := "goblog"

db, err := sql.Open(dbDriver, dbUser+":"+dbPass+"@/"+dbName)

if err != nil {

panic(err.Error())

}

return db

}

var tmpl = template.Must(template.ParseGlob("form/\*"))

func Index(w http.ResponseWriter, r \*http.Request) {

db := dbConn()

selDB, err := db.Query("SELECT \* FROM Employee ORDER BY id DESC")

if err != nil {

panic(err.Error())

}

emp := Employee{}

res := []Employee{}

for selDB.Next() {

var id int

var name, city string

err = selDB.Scan(&id, &name, &city)

if err != nil {

panic(err.Error())

}

emp.Id = id

emp.Name = name

emp.City = city

res = append(res, emp)

}

tmpl.ExecuteTemplate(w, "Index", res)

defer db.Close()

}

func Show(w http.ResponseWriter, r \*http.Request) {

db := dbConn()

nId := r.URL.Query().Get("id")

selDB, err := db.Query("SELECT \* FROM Employee WHERE id=?", nId)

if err != nil {

panic(err.Error())

}

emp := Employee{}

for selDB.Next() {

var id int

var name, city string

err = selDB.Scan(&id, &name, &city)

if err != nil {

panic(err.Error())

}

emp.Id = id

emp.Name = name

emp.City = city

}

tmpl.ExecuteTemplate(w, "Show", emp)

defer db.Close()

}

func New(w http.ResponseWriter, r \*http.Request) {

tmpl.ExecuteTemplate(w, "New", nil)

}

func Edit(w http.ResponseWriter, r \*http.Request) {

db := dbConn()

nId := r.URL.Query().Get("id")

selDB, err := db.Query("SELECT \* FROM Employee WHERE id=?", nId)

if err != nil {

panic(err.Error())

}

emp := Employee{}

for selDB.Next() {

var id int

var name, city string

err = selDB.Scan(&id, &name, &city)

if err != nil {

panic(err.Error())

}

emp.Id = id

emp.Name = name

emp.City = city

}

tmpl.ExecuteTemplate(w, "Edit", emp)

defer db.Close()

}

func Insert(w http.ResponseWriter, r \*http.Request) {

db := dbConn()

if r.Method == "POST" {

name := r.FormValue("name")

city := r.FormValue("city")

insForm, err := db.Prepare("INSERT INTO Employee(name, city) VALUES(?,?)")

if err != nil {

panic(err.Error())

}

insForm.Exec(name, city)

log.Println("INSERT: Name: " + name + " | City: " + city)

}

defer db.Close()

http.Redirect(w, r, "/", 301)

}

func Update(w http.ResponseWriter, r \*http.Request) {

db := dbConn()

if r.Method == "POST" {

name := r.FormValue("name")

city := r.FormValue("city")

id := r.FormValue("uid")

insForm, err := db.Prepare("UPDATE Employee SET name=?, city=? WHERE id=?")

if err != nil {

panic(err.Error())

}

insForm.Exec(name, city, id)

log.Println("UPDATE: Name: " + name + " | City: " + city)

}

defer db.Close()

http.Redirect(w, r, "/", 301)

}

func Delete(w http.ResponseWriter, r \*http.Request) {

db := dbConn()

emp := r.URL.Query().Get("id")

delForm, err := db.Prepare("DELETE FROM Employee WHERE id=?")

if err != nil {

panic(err.Error())

}

delForm.Exec(emp)

log.Println("DELETE")

defer db.Close()

http.Redirect(w, r, "/", 301)

}

func main() {

log.Println("Server started on: http://localhost:8080")

http.HandleFunc("/", Index)

http.HandleFunc("/show", Show)

http.HandleFunc("/new", New)

http.HandleFunc("/edit", Edit)

http.HandleFunc("/insert", Insert)

http.HandleFunc("/update", Update)

http.HandleFunc("/delete", Delete)

http.ListenAndServe(":8080", nil)

}

**Step 4: Creating Template files**

Now it's time to build the Template files of our CRUD application. Create form folder at same location where we have created main.go.

a)Let's create a file named Index.tmpl inside the form folder and put the following code inside it.

{{ define "Index" }}

{{ template "Header" }}

{{ template "Menu" }}

<h2> Registered </h2>

<div class="table-responsive">

<table class="table">

<thead>

<tr>

<td>ID</td>

<td>Name</td>

<td>City</td>

<td>View</td>

<td>Edit</td>

<td>Delete</td>

</tr>

</thead>

<tbody>

{{ range . }}

<tr>

<td>{{ .Id }}</td>

<td> {{ .Name }} </td>

<td>{{ .City }} </td>

<td><a href="/show?id={{ .Id }}">View</a></td>

<td><a href="/edit?id={{ .Id }}">Edit</a></td>

<td><a href="/delete?id={{ .Id }}" onclick=" return confirm('Are You sure you want to delete?')">Delete</a><td>

</tr>

{{ end }}

</tbody>

</table>

</div>

{{ template "Footer" }}

{{ end }}

b)Now create another file named Header.tmpl inside the same form folder and put the following code inside it.

{{ define "Header" }}

<!DOCTYPE html>

<html lang="en-US">

<head>

<title>Golang Mysql Curd Example</title>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<style type="text/css">

container.{

padding: 50px;

}

</style>

</head>

<body>

<div class="container">

{{ end }}

c)Now create another file named Footer.tmpl inside the same form folder and put the following code inside it.

{{ define "Footer" }}

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

</div>

</div>

</div>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

</body>

</html>

{{ end }}

d)Now create another file named Menu.tmpl inside the same form folder and put the following code inside it.

{{ define "Menu" }}

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="#">Navbar</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item active">

<a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="/new">Add New</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Country List</a>

</li>

<li class="nav-item">

<a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>

</li>

</ul>

</div>

</nav>

<div class="row">

<div class="col-lg-12 col-md-12 col-sm-12 ">

{{ end }}

e)Next, we have to create Show.tmpl file for item details page, so again create this file in form folder.

{{ define "Show" }}

{{ template "Header" }}

{{ template "Menu" }}

<h2> Register {{ .Id }} </h2>

<p>Name: {{ .Name }}</p>

<p>City: {{ .City }}</p><br /> <a href="/edit?id={{ .Id }}">Edit</a></p>

{{ template "Footer" }}

{{ end }}

f)Now we create new blade file for create new item, it's call New.tmpl file inside form.

{{ define "New" }}

{{ template "Header" }}

{{ template "Menu" }}

<form method="POST" action="insert">

<div class="form-group">

<label for="exampleInputEmail1">Country</label>

<input type="text" class="form-control" name="name" required="true" placeholder="Enter Country">

</div>

<div class="form-group">

<label for="exampleInputPassword1">City</label>

<input type="text" class="form-control" name="city" required="true" placeholder="City">

</div>

<button type="submit" class="btn btn-primary">Save user</button>

</form>

{{ template "Footer" }}

{{ end }}

g)At last, we need to create Edit.tmpl file for update item, so again create this file in form folder.

{{ define "Edit" }}

{{ template "Header" }}

{{ template "Menu" }}

<h2>Edit Name and City</h2>

<form method="POST" action="update">

<input type="hidden" name="uid" value="{{ .Id }}" />

<div class="form-group">

<label for="exampleInputEmail1">Country</label>

<input type="text" class="form-control" name="name" value="{{ .Name }}" placeholder="Enter Country">

</div>

<div class="form-group">

<label for="exampleInputPassword1">City</label>

<input type="text" class="form-control" name="city" value="{{ .City }}" placeholder="City">

</div>

<button type="submit" class="btn btn-primary">Save user</button>

</form>

{{ template "Footer" }}

{{ end }}

**After a long journey finally we've created all files of our CRUD application with Golang and MySQL.**

### Run the following command

go run main.go

### Load the following URL

http:*//localhost:8080/*



