

- 1) cd ~/go/src/github.com/
- 2)git clone <https://github.com/patharetush/pucsd2020-pp.git>
- 3)tyanantr mysql nasel tr te install kel
- 4)tyanantr cd /pucsd2020-pp/rest-api/
- 5)shubham@pucsd:~/go/src/github.com/pucsd2020-pp/rest-api/resource/config

```
{
  "host": "",
  "port": 9090,
  "database": {
    "dbname": "restapi",
    "host": "localhost",
    "port": 3306,
    "user": "root", //yamdhe jr tumhi user create kela asel tr to ith takaycha nahitr root
    "password": "q", //aani ith jo password asel to password
    "idle_connection": 10,
    "max_connection": 100
  }
}
```

asel tr root

- 5) shubham@pucsd:~/go/src/github.com/pucsd2020-pp/rest-api/config\$ gedit config.go

```
package config
```

```
import (
    "encoding/json"
    "fmt"
    "io/ioutil"
    "log"
    "os"
)
```

```
const (
    configFile = "/resource/config/rest-api.cfg" //ha path change karava
)
```

```
var (
    _config *RestApiConfig
)
```

```
type RestApiConfig struct {
    Host    string    `json:"host"`
    Port    int        `json:"port"`
    Database MySqlConnection `json:"database"`
}
```

```
type MySqlConnection struct {
    Database    string `json:"dbname"`
    Host        string `json:"host"`
    Port        int    `json:"port"`
    Username    string `json:"user"`
    Password    string `json:"password"`
}
```

```

        IdleConnection int    `json:"idle_connection"`
        MaxConnection  int    `json:"max_connection"`
    }

    func (api *RestApiConfig) String() string {
        byts, _ := json.Marshal(api)
        return string(byts)
    }

    func init() {
        pwd, _ := os.Getwd()
        file, err := ioutil.ReadFile(pwd + configFile) // aani ith pn changes
        if nil != err {
            log.Printf("Error while reading config file: %s:%s", configFile, err.Error())
            os.Exit(1)
        }

        _config = new(RestApiConfig)
        err = json.Unmarshal(file, _config)
        if nil != err {
            log.Printf("Error while reading configuration: %s:%s", configFile, err.Error())
            os.Exit(1)
        }
    }

    func Config() *RestApiConfig {
        return _config
    }

    func (cfg *MysqlConnection) ConnString() string {
        return fmt.Sprintf(
            "%s:%s@tcp(%s:%d)/%s", cfg.Username, cfg.Password,
            cfg.Host, cfg.Port, cfg.Database)
    }

```

7) [shubham@pucsd:~/go/src/github.com/pucsd2020-pp/rest-api/resource/sql](https://github.com/pucsd2020-pp/rest-api/resource/sql)\$

yamadhe jaun donhi file run karaychya

i) `mysql -u username -p <000_create_database.sql`

ii) `mysql -u username -p <001_create_table.sql`

8) `shubham@pucsd:~/go/src/github.com/pucsd2020-pp/rest-api/repository/user$ gedit user.go`

```
package user
```

```

import (
    "context"
    "database/sql"

    "github.com/pucsd2020-pp/rest-api/driver"
    "github.com/pucsd2020-pp/rest-api/model"
)

```

```

type userRepository struct {
    conn *sql.DB
}

func NewUserRepository(conn *sql.DB) *userRepository {
    return &userRepository{conn: conn}
}

func (user *userRepository) GetByID(cntx context.Context, id int64) (interface{}, error) {
    obj := new(model.User)
    return driver.GetById(user.conn, obj, id)
}

func (user *userRepository) Create(cntx context.Context, obj interface{}) (interface{}, error) {
    usr := obj.(model.User)
    //usr := obj.(*model.User) //model cha * kadhaycha
    result, err := driver.Create(user.conn,&usr)
    //result, err := driver.Create(user.conn,usr) // aani ith usr la & takaych same update madhe
pn
    if nil != err {
        return 0, err
    }

    id, _ := result.LastInsertId()
    usr.Id = id
    return id, nil
}

func (user *userRepository) Update(cntx context.Context, obj interface{}) (interface{}, error) {
    usr := obj.(model.User)
    err := driver.UpdateById(user.conn,&usr)
    return obj, err
}

func (user *userRepository) Delete(cntx context.Context, id int64) error {
    obj := &model.User{Id: id}
    return driver.SoftDeleteById(user.conn, obj, id)
}

func (user *userRepository) GetAll(cntx context.Context) ([]interface{}, error) {
    obj := &model.User{}
    return driver.GetAll(user.conn, obj, 0, 0)
}

```

9) shubham@pucsd:~/go/src/github.com/pucsd2020-pp/rest-api/driver\$ gedit mysql.go
//ya code madhe jith bold aahe tith comment

package driver

import (

```

"bytes"
"database/sql"
"errors"
"fmt"
"log"
"reflect"
"strings"
"time"

"github.com/pucsd2020-pp/rest-api/config"
"github.com/pucsd2020-pp/rest-api/model"

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

const (
    MYSQL_DRIVER_NAME = "mysql"
    CONN_MAX_LIFETIME  = 30 * 60 * 60 // 30 day
    COLUMN_IGNORE_FLAG = "1"
    COLUMN_PRIMARY     = "primary"
)

func NewMysqlConnection(cfg config.MysqlConnection) (*sql.DB, error) {
    db, err := sql.Open(MYSQL_DRIVER_NAME, cfg.ConnString())
    if err != nil {
        log.Fatalf("Failed to open mysql connection: %v", err)
        return nil, err
    }

    if cfg.IdleConnection > 0 {
        db.SetMaxIdleConns(cfg.IdleConnection)
    }
    if cfg.MaxConnection > 0 {
        db.SetMaxOpenConns(cfg.MaxConnection)
    }
    db.SetConnMaxLifetime(time.Second * CONN_MAX_LIFETIME)

    if err := db.Ping(); err != nil {
        log.Fatalf("Failed to ping mysql: %v", err)
    }

    return db, err
}

// return the placeholder string with given count
func GetPlaceholder(count int) string {
    if count > 0 {
        str := strings.Repeat("?", ", ", count)
        return str[:len(str)-2]
    }

    return ""
}

```

```

}

/**
 * Insert new row
 */
func Create(conn *sql.DB, object model.IModel) (sql.Result, error) {
    rValue := reflect.ValueOf(object)
    rType := reflect.TypeOf(object)

    columns := []string{}
    var params []interface{}

    count := 0
    for idx := 0; idx < rValue.Elem().NumField(); idx++ {
        field := rType.Elem().Field(idx)
        value := rValue.Elem().Field(idx)

        /*if value.IsNil() || COLUMN_IGNORE_FLAG ==
field.Tag.Get("autoincr") ||
        COLUMN_IGNORE_FLAG == field.Tag.Get("ignore") {
            continue
        }*/

        column := field.Tag.Get("column")
        columns = append(columns, column)
        params = append(params, value.Interface())
        count++
    }

    var queryBuffer bytes.Buffer
    queryBuffer.WriteString("INSERT INTO ")
    queryBuffer.WriteString(object.Table())
    queryBuffer.WriteString("(")
    queryBuffer.WriteString(strings.Join(columns, ", "))
    queryBuffer.WriteString(") VALUES(")
    queryBuffer.WriteString(GetPlaceholder(count))
    queryBuffer.WriteString(");")

    query := queryBuffer.String()
    stmt, err := conn.Prepare(query)
    if nil != err {
        log.Printf("Insert Syntax Error: %s\n\tError Query: %s : %s\n",
            err.Error(), object.String(), query)
        return nil, err
    }

    defer stmt.Close()

    result, err := stmt.Exec(params...)
    if nil != err {
        log.Printf("Insert Execute Error: %s\n\tError Query: %s : %s\n",

```

```

        err.Error(), object.String(), query)
    return nil, err
}

return result, nil
}

/**
 * Update existing row with key column
 */
func UpdateById(conn *sql.DB, object model.IModel) error {
    rValue := reflect.ValueOf(object)
    rType := reflect.TypeOf(object)

    columns := []string{}
    var params []interface{}

    keyColumns := []string{}
    var keyParams []interface{}

    for idx := 0; idx < rValue.Elem().NumField(); idx++ {
        field := rType.Elem().Field(idx)
        value := rValue.Elem().Field(idx)

        /*if value.IsNil() ||
            COLUMN_IGNORE_FLAG == field.Tag.Get("ignore") {
            continue
        */

        column := field.Tag.Get("column")
        if COLUMN_PRIMARY == field.Tag.Get("key") {
            keyColumns = append(keyColumns, column+" = ?")
            keyParams = append(keyParams, value.Interface())
        } else {
            columns = append(columns, column+" = ?")
            params = append(params, value.Interface())
        }
    }

    for _, param := range keyParams {
        params = append(params, param)
    }

    var queryBuffer bytes.Buffer
    queryBuffer.WriteString("UPDATE ")
    queryBuffer.WriteString(object.Table())
    queryBuffer.WriteString(" SET ")
    queryBuffer.WriteString(strings.Join(columns, ", "))
    queryBuffer.WriteString(" WHERE ")
    queryBuffer.WriteString(strings.Join(keyColumns, ", "))
    queryBuffer.WriteString(";")

```

```

query := queryBuffer.String()
//      log.Println("Update statement is: %s", query)
stmt, err := conn.Prepare(query)
if nil != err {
    log.Printf("Update Syntax Error: %s\n\tError Query: %s : %s\n",
        err.Error(), object.String(), query)
    return err
}

defer stmt.Close()
_, err = stmt.Exec(params...)
if nil != err {
    log.Printf("Update Execute Error: %s\n\tError Query: %s : %s\n",
        err.Error(), object.String(), query)
}

return err
}

func GetById(conn *sql.DB, object model.IModel, id int64) (model.IModel, error) {
    rValue := reflect.ValueOf(object)
    rType := reflect.TypeOf(object)

    columns := []string{}
    pointers := make([]interface{}, 0)

    for idx := 0; idx < rValue.Elem().NumField(); idx++ {
        field := rType.Elem().Field(idx)
        if COLUMN_IGNORE_FLAG == field.Tag.Get("ignore") {
            continue
        }

        column := field.Tag.Get("column")
        columns = append(columns, column)
        pointers = append(pointers, rValue.Elem().Field(idx).Addr().Interface())
    }

    var queryBuffer bytes.Buffer

    queryBuffer.WriteString("SELECT ")
    queryBuffer.WriteString(strings.Join(columns, ", "))
    queryBuffer.WriteString(" FROM ")
    queryBuffer.WriteString(object.Table())
    queryBuffer.WriteString(" WHERE id = ?")

    query := queryBuffer.String()
    //      log.Printf("GetById sql: %s\n", query)
    row, err := conn.Query(query, id)

    if nil != err {
        log.Printf("Error conn.Query: %s\n\tError Query: %s\n", err.Error(), query)
    }
}

```

```

        return nil, err
    }

    defer row.Close()
    if row.Next() {
        if nil != err {
            log.Printf("Error row.Columns(): %s\n\tError Query: %s\n", err.Error(),
query)
                return nil, err
            }

            err = row.Scan(pointers...)
            if nil != err {
                log.Printf("Error: row.Scan: %s\n", err.Error())
                return nil, err
            }
        } else {
            return nil, errors.New(fmt.Sprintf("Entry not found for id: %d", id))
        }

        return object, nil
    }
}

```

```

func GetAll(conn *sql.DB, object model.IModel, limit, offset int64) ([]interface{}, error) {
    rValue := reflect.ValueOf(object)
    rType := reflect.TypeOf(object)

    columns := []string{}
    pointers := make([]interface{}, 0)

    for idx := 0; idx < rValue.Elem().NumField(); idx++ {
        field := rType.Elem().Field(idx)
        if COLUMN_IGNORE_FLAG == field.Tag.Get("ignore") {
            continue
        }

        column := field.Tag.Get("column")
        columns = append(columns, column)
        pointers = append(pointers, rValue.Elem().Field(idx).Addr().Interface())
    }

    var queryBuffer bytes.Buffer
    var params []interface{}

    queryBuffer.WriteString("SELECT ")
    queryBuffer.WriteString(strings.Join(columns, ", "))
    queryBuffer.WriteString(" FROM ")
    queryBuffer.WriteString(object.Table())
    if 0 != limit && 0 != offset {
        queryBuffer.WriteString(" LIMIT ? OFFSET ?")
        params = append(params, limit)
        params = append(params, offset)
    }
}

```



```

    }

    query := queryBuffer.String()
    //      log.Printf("GetById sql: %s\n", query)
    row, err := conn.Query(query, params...)

    if nil != err {
        log.Printf("Error conn.Query: %s\n\tError Query: %s\n", err.Error(), query)
        return nil, err
    }

    defer row.Close()
    objects := make([]interface{}, 0)
    for row.Next() {
        if nil != err {
            log.Printf("Error row.Columns(): %s\n\tError Query: %s\n", err.Error(),
query)

            return nil, err
        }

        err = row.Scan(pointers...)
        if nil != err {
            log.Printf("Error: row.Scan: %s\n", err.Error())
            return nil, err
        }

        objects = append(objects, object)
    }

    return objects, nil
}

```

```

func DeleteById(conn *sql.DB, object model.IModel, id int64) (sql.Result, error) {
    var queryBuffer bytes.Buffer
    queryBuffer.WriteString("DELETE FROM ")
    queryBuffer.WriteString(object.Table())
    queryBuffer.WriteString(" WHERE id = ?")

    query := queryBuffer.String()
    //      log.Println("Delete statement is: %s", query)
    stmt, err := conn.Prepare(query)
    if nil != err {
        log.Printf("Delete Syntax Error: %s\n\tError Query: %s : %s\n",
            err.Error(), object.String(), query)
        return nil, err
    }

    defer stmt.Close()
    result, err := stmt.Exec(id)
    if nil != err {
        log.Printf("Delete Execute Error: %s\n\tError Query: %s : %s\n",
            err.Error(), object.String(), query)
    }
}

```

```

    }

    return result, err
}

func SoftDeleteById(conn *sql.DB, object model.IModel, id int64) error {
    var queryBuffer bytes.Buffer
    queryBuffer.WriteString("UPDATE ")
    queryBuffer.WriteString(object.Table())
    queryBuffer.WriteString(" SET deleted = 1 WHERE id = ?")

    query := queryBuffer.String()
    //      log.Println("Delete statement is: %s", query)
    stmt, err := conn.Prepare(query)
    if nil != err {
        log.Printf("Delete Syntax Error: %s\n\tError Query: %s : %s\n",
            err.Error(), object.String(), query)
        return err
    }

    defer stmt.Close()
    _, err = stmt.Exec(id)
    if nil != err {
        log.Printf("Delete Execute Error: %s\n\tError Query: %s : %s\n",
            err.Error(), object.String(), query)
    }

    return err
}

```

10) then run

```
shubham@pucsd:~/go/src/github.com/pucsd2020-pp/rest-api$ go run main.go
```

10)shubham@pucsd:~\$curl -X POST -d

```
'{"first_name":"shubham","last_name":"shincholkar","email":"chincholkar1711@gmail.com","password":"1224578","contact_number":"9422749610","updated_by":0}'
```

```
http://localhost:9090/webapi/v1/user
```

```
{"status":200,"data":
```

```
{"first_name":"shubham","last_name":"shincholkar","email":"chincholkar1711@gmail.com","password":"1224578","contact_number":"9422749610","updated_by":0},"message":"Error 1062:
```

```
Duplicate entry 'chincholkar1711@gmail.com' for key 'email'"}(base) shubham@pucsd:~$ ^C
```

(base)

```
shubham@pucsd:~$ curl -X POST -d
```

```
'{"first_name":"namdev","last_name":"survase","email":"namdev_survase@gmail.com","password":"2245788","contact_number":"9422749610","updated_by":0}'
```

```
http://localhost:9090/webapi/v1/user
```

```
{"status":200,"data":
```

```
{"first_name":"namdev","last_name":"survase","email":"namdev_survase@gmail.com","password":"2245788","contact_number":"9422749610","updated_by":0}}(base) shubham@pucsd:~$ ^C
```

(base)

```
shubham@pucsd:~$ curl -X DELETE http://localhost:9090/webapi/v1/user/2
```

```
{"status":200,"data":"User deleted successfully"}(base) shubham@pucsd:~$ ^C
```

```
(base)
shubham@pucsd:~$ curl -X DELETE http://localhost:9090/webapi/v1/user/7
{"status":200,"data":"User deleted successfully"}(base) shubham@pucsd:~$ ^C
(base)
shubham@pucsd:~$ curl -X PUT -d
'{"first_name":"Shubham","last_name":"Chincholkar","email":"Chincholkar1711@gmail.com","password":"224578","contact_number":"9422749610","updated_by":0}'
http://localhost:9090/webapi/v1/user/5
{"status":200,"data":
{"id":5,"first_name":"Shubham","last_name":"Chincholkar","email":"Chincholkar1711@gmail.com","password":"224578","contact_number":"9422749610","updated_by":0}}
(base) shubham@pucsd:~$ curl -X PUT -d
'{"first_name":"shubhu","last_name":"chincholkar","email":"chincholkar1711@gmail.com","password":"224578","contact_number":"9422749610","updated_by":0}'
http://localhost:9090/webapi/v1/user/5
{"status":200,"data":
{"id":5,"first_name":"shubhu","last_name":"chincholkar","email":"chincholkar1711@gmail.com","password":"224578","contact_number":"9422749610","updated_by":0}}
```

```
11) shubham@pucsd:~$ mysql -u root -p restapi
select * from user_detail;
```