Если вы видите что-то необычное, просто сообщите мне.

Echo framework + GORM = Огненно быстрое Golang приложение на стороне сервера. Пример аутентификации.

В статье, я хочу показать пример реализации входа, выхода и регистрации используя Go фреймворк Echo и Gorm для postgreSQL. Для аутентификации пользователей будет использоваться JWT(Json web token).

Для начала создадим `main.go`:

```
package main

import (
	"app1/helper"
	"app1/models"
	"fmt"
	"github.com/jinzhu/gorm"
	_ "github.com/jinzhu/gorm/dialects/postgres"
	"github.com/labstack/echo"
	"github.com/labstack/echo/middleware"
)

func main() {
	configuration := helper.GetConfig()
```

```
	gormParameters := fmt.Sprintf("host=%s port=%s dbname=%s user=%s password=%s sslmode=disable",
	configuration.DbHost, configuration.DbPort, configuration.DbName, configuration.DbUsername,
	configuration.DbPassword)
	gormDB, err := gorm.Open("postgres", gormParameters)
	if err != nil {
		panic("failed to connect database")
	}
	helper.GormDB = gormDB

	// Migrate the schema (tables): User, Authentication
	helper.GormDB.AutoMigrate(&helper.User{})
	helper.GormDB.AutoMigrate(&helper.Authentication{})
	helper.GormDB.Model(&helper.Authentication{}).AddForeignKey("user_id", "users(id)", "CASCADE",
	"CASCADE")

	echoFramework := echo.New()
	echoFramework.Use(middleware.Logger()) // log
	echoFramework.Use(middleware.CORS())   // CORS from Any Origin, Any Method

	echoGroupUseJWT := echoFramework.Group("/api/v1")
	echoGroupUseJWT.Use(middleware.JWT([]byte(configuration.EncryptionKey)))
	echoGroupNoJWT := echoFramework.Group("/api/v1")

	// /api/v1/users : logged in users
	echoGroupUseJWT.POST("/users/logout", models.Logout)

	// /api/v1/users : public accessible
	echoGroupNoJWT.POST("/users", models.CreateUser)
	echoGroupNoJWT.POST("/users/login", models.Login)

	defer helper.GormDB.Close()
	echoFramework.Logger.Fatal(echoFramework.Start(":1323"))
	}
```

Второе - создадим `globals.go`:

```
package helper

import (
```

```
	"regexp"
	"time"
	"github.com/jinzhu/gorm"

)

type Configuration struct {
	EncryptionKey string
	DbHost        string
	DbPort        string
	DbName        string
	DbUsername    string
	DbPassword    string
}

var configuration Configuration
type ModelBase struct {
	ID        int       `gorm:"primary_key"`
	CreatedAt time.Time `json:"-"`
	UpdatedAt time.Time `json:"-"`
}

type User struct {
	ModelBase        // replaces gorm.Model
	Email     string `gorm:"not null; unique"`
	Password  string `gorm:"not null" json:"-"`
	Name      string `gorm:"not null; type:varchar(100)"` // unique_index
}

type Authentication struct {
	ModelBase
	User   User `gorm:"foreignkey:UserID; not null"`
	UserID int
	Token  string `gorm:"type:varchar(200); not null"`
}

type CustomHTTPSuccess struct {
	Data string `json:"data"`
}
```

```
type ErrorType struct {
	Code    int    `json:"code"`
	Message string `json:"message"`
}

type CustomHTTPError struct {
	Error ErrorType `json:"error"`
}

var GormDB *gorm.DB
func init() {
	configuration = Configuration{
		EncryptionKey: "F61L8L7CUCGN0NK6336I8TFP9Y2ZOS43",
		DbHost:        "localhost",
		DbPort:        "5432",
		DbName:        "postgres",
		DbUsername:    "postgres",
		DbPassword:    "postgres",
	}
}

func GetConfig() Configuration {
	return configuration
}

func ValidateEmail(email string) (matchedString bool) {
	re := regexp.MustCompile("^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-
]{0,61}[a-zA-Z0-9])?(?:\\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$")
	matchedString = re.MatchString(email)
	return
}
```

Теперь реализуем метод аутентификации в `user.go`:

```
package models

import (
	"app1/helper"
	"fmt"
	"net/http"
```

```
	"strconv"
	jwt "github.com/dgrijalva/jwt-go"
	"github.com/labstack/echo"
	)

	func CreateUser(c echo.Context) error {
		m := echo.Map{}
		if err := c.Bind(&m); err != nil {
			// return err
		}

		name := m["name"].(string)
		email := m["email"].(string)
		password := m["password"].(string)
		confirmPassword := m["confirm_password"].(string)
		if password == "" || confirmPassword == "" || name == "" || email == "" {
			return echo.NewHTTPError(http.StatusBadRequest, "Please enter name, email and password")
		}
		if password != confirmPassword {
			return echo.NewHTTPError(http.StatusBadRequest, "Confirm password is not same to password
	provided")
		}
		if helper.ValidateEmail(email) == false {
			return echo.NewHTTPError(http.StatusBadRequest, "Please enter valid email")
		}
		if bCheckUserExists(email) == true {
			return echo.NewHTTPError(http.StatusBadRequest, "Email provided already exists")
		}
		configuration := helper.GetConfig()
		enc, _ := helper.EncryptString(password, configuration.EncryptionKey)
		user1 := helper.User{Name: name, Email: email, Password: enc}

		// globals.GormDB.NewRecord(user) // => returns `true` as primary key is blank
		helper.GormDB.Create(&user1)
		token := jwt.New(jwt.SigningMethodHS256)
		claims := token.Claims.(jwt.MapClaims)
		claims["name"] = user1.Name
		claims["email"] = user1.Email
		t, err := token.SignedString([]byte(configuration.EncryptionKey)) // "secret" >> EncryptionKey
		if err != nil {
```

```
    return err
  }
  authentication := helper.Authentication{}
  if helper.GormDB.First(&authentication, "user_id =?", user1.ID).RecordNotFound() {
    // insert
    helper.GormDB.Create(&helper.Authentication{User: user1, Token: t})
  } else {
    authentication.User = user1
    authentication.Token = t
    helper.GormDB.Save(&authentication)
  }
  return c.JSON(http.StatusOK, map[string]string{
    "token": t,
  })
}
func bCheckUserExists(email string) bool {
  user1 := helper.User{}
  if helper.GormDB.Where(&helper.User{Email: email}).First(&user1).RecordNotFound() {
    return false
  }
  return true
}
func ValidateUser(email, password string, c echo.Context) (bool, error) {
  fmt.Println("validate")
  var user1 helper.User
  if helper.GormDB.First(&user1, "email =?", email).RecordNotFound() {
    return false, nil
  }
  configuration := helper.GetConfig()
  decrypted, _ := helper.DecryptString(user1.Password, configuration.EncryptionKey)
  if password == decrypted {
    return true, nil
  }
  return false, nil
}

func Login(c echo.Context) error {
  m := echo.Map{}
  if err := c.Bind(&m); err != nil {
    // return err
```

```
	}
	email := m["email"].(string)
	password := m["password"].(string)
	var user1 helper.User
	if helper.GormDB.First(&user1, "email =?", email).RecordNotFound() {
		_error := helper.CustomHTTPError{
			Error: helper.ErrorType{
				Code:    http.StatusBadRequest,
				Message: "Invalid email & password",
			},
		}
		return c.JSONPretty(http.StatusBadGateway, _error, "  ")
	}
	configuration := helper.GetConfig()
	decrypted, _ := helper.DecryptString(user1.Password, configuration.EncryptionKey)
	if password == decrypted {
		token := jwt.New(jwt.SigningMethodHS256)
		claims := token.Claims.(jwt.MapClaims)
		claims["name"] = user1.Name
		claims["email"] = user1.Email
		claims["id"] = user1.ModelBase.ID
		t, err := token.SignedString([]byte(configuration.EncryptionKey)) // "secret" >> EncryptionKe
		if err != nil {
			return err
		}
		authentication := helper.Authentication{}
		if helper.GormDB.First(&authentication, "user_id =?", user1.ID).RecordNotFound() {
			// insert
			helper.GormDB.Create(&helper.Authentication{User: user1, Token: t})
		} else {
			// update
			authentication.User = user1
			authentication.Token = t
			helper.GormDB.Save(&authentication)
		}
		return c.JSON(http.StatusOK, map[string]string{
			"token": t,
		})
	} else {
		_error := helper.CustomHTTPError{
```

```
□□□Error: helper.ErrorType{
□□□□Code:    http.StatusBadRequest,
□□□□Message: "Invalid email & password",
□□□},
□□}
□□return c.JSONPretty(http.StatusBadGateway, _error, "  ")
□}
}
func Logout(c echo.Context) error {
□tokenRequester := c.Get("user").(*jwt.Token)
□claims := tokenRequester.Claims.(jwt.MapClaims)
□fRequesterID := claims["id"].(float64)
□iRequesterID := int(fRequesterID)
□sRequesterID := strconv.Itoa(iRequesterID)
□requester := helper.User{}
□if helper.GormDB.First(&requester, "id =?", sRequesterID).RecordNotFound() {
□□return echo.ErrUnauthorized
□}
□authentication := helper.Authentication{}
□if helper.GormDB.First(&authentication, "user_id =?", requester.ModelBase.ID).RecordNotFound()
{
□□return echo.ErrUnauthorized
□}
□helper.GormDB.Delete(&authentication)
□return c.String(http.StatusAccepted, "")
}
```

Пропущенный функции шифроки\расшифровки. Напишите мне чтобы их увидеть.

Revision #2
Created 2022-08-11 20:17:34 UTC by gasick
Updated 2023-04-16 19:30:04 UTC by gasick