AIP Communication Module Template Documentation

CONTENTS

1. Introduction

1.1 Purpose

CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: **create, read, update and delete.** In this document you will get the idea to create a simple CRUD project in Go.

1.2 Overview

You can get the CRUD functionality in go app when you use this project. You can hit PUT, GET, DELETE and POST request and according to the request changes will happen.

2. Code

```
package main

import (
   "encoding/json"
   "fmt"
   "net/http"
   "regexp"
   "strconv"
)

// Movie is a struct
type Movie struct {
   ID       int     `json:"id"`
   Name     string  `json:"name"`
   FilmMaker string  `json:"filmMaker"`
   Rating    float32 `json:"rating"`
}

func findProduct(mid int) (int, error) {
   for i, p := range movies {
      if p.ID == mid {
         return i, nil
      }
   }
   return -1, fmt.Errorf("movie not Found")
}

func returnResponse(w http.ResponseWriter) {
   err := json.NewEncoder(w).Encode(movies)
   if err != nil {
      fmt.Println("There seems to be an error : ", err)
      return
   }
}

func returnError(w http.ResponseWriter, s string) {
   err := json.NewEncoder(w).Encode(s)
   if err != nil {
      fmt.Println("There seems to be an error : ", err)
      return
   }
}
```

```
func mainHandler(w http.ResponseWriter, r *http.Request) {
   w.Header().Set("Content-Type", "application/json")
   if r.Method == http.MethodGet {
      fmt.Println("A GET request.")

      returnResponse(w) // Returns a json of all the Movies in the database
   }

   if r.Method == http.MethodPost {
      fmt.Println("A POST request.")
      var p Movie
      err := json.NewDecoder(r.Body).Decode(&p)
      if err != nil {
         returnError(w, "Couldn't decode your request body.")
         fmt.Println("There seems to be an error : ", err)
         return
      }
      p.ID = len(movies) + 1
      movies = append(movies, p)

      returnResponse(w)
   }

   if r.Method == http.MethodPut {
      fmt.Println("A PUT request.")

      reg := regexp.MustCompile(`/([0-9]+)`)
      g := reg.FindAllStringSubmatch(r.URL.Path, -1)
      strMid := g[0][1]
      mid, err := strconv.Atoi(strMid)
      if err != nil {
         returnError(w, "Couldn't decode the URI.")
         fmt.Println("Not a valid URI.")
         return
      }

      i, err := findProduct(mid)
      if err != nil {
         returnError(w, "Couldn't find the Movie id.")
         fmt.Println("There seems to be an error : ", err)
         return
      }

      var p Movie
      err = json.NewDecoder(r.Body).Decode(&p)
      if err != nil {
         returnError(w, "Couldn't decode your request body.")
```

```
            fmt.Println("There seems to be an error : ", err)
        }
        p.ID = mid
        movies[i] = p

        returnResponse(w)
    }

    if r.Method == http.MethodDelete {
        fmt.Println("A DELETE request.")

        reg := regexp.MustCompile(`/([0-9]+)`)
        g := reg.FindAllStringSubmatch(r.URL.Path, -1)
        strMid := g[0][1]
        mid, err := strconv.Atoi(strMid)
        if err != nil {
            returnError(w, "Couldn't decode the URI.")
            fmt.Println("Not a valid URI.")
            return
        }

        i, err := findProduct(mid)
        fmt.Println("i", i)
        fmt.Println(movies[i])
        if err != nil {
            returnError(w, "Couldn't find the Movie id.")
            fmt.Println("There seems to be an error : ", err)
            return
        }

        movies = append(movies[:i], movies[i+1:]...)

        returnResponse(w)

    }
}

func main() {
    http.HandleFunc("/", mainHandler)
    http.ListenAndServe(":8080", nil)
}

var movies = []Movie{
    Movie{1, "Chungking Express", "Wong Kar Wai", 9.0},
    Movie{ID: 2, Name: "Yi Yi", FilmMaker: "Edward Yang", Rating: 9.0},
    Movie{ID: 3, Name: "Memories of Murder", FilmMaker: "Bong Joon Ho", Rating: 8.9},
}
```