groupie-tracker-filters

Objectives

You must follow the same principles as the first subject.

- Groupie Tracker Filters consists on letting the user filter the artists/bands that will be shown.
- Your project must incorporate at least these four filters:
  - filter by creation date
  - filter by first album date
  - filter by number of members
  - filter by locations of concerts
- Your filters must be of at least these two types:
  - a range filter (filters the results between two values)
  - a check box filter (filters the results by one or multiple selection)

Example

Here is an example of both types of filters:

.

Hints

- You have to pay attention to the locations. For example Seattle, Washington, USA **is part of** Washington, USA.

Instructions

- The backend must be written in **Go**.
- You must handle website errors.
- The code must respect the good practices
- It is recommended to have **test files** for unit testing.

Allowed packages

- Only the standard Go packages are allowed.

This project will help you learn about:

- Manipulation, display and storage of data
- Event creation and display
- JSON files and format
- Go routines

groupie-tracker-geolocalization

Objectives

You must follow the same principles as the first subject.

- Groupie Tracker Geolocalization consists on mapping the different concerts locations of a certain artist/band given by the Client.
- You must use a process of converting addresses (ex: Germany Mainz) into geographic coordinates (ex: 49,59380 8,15052) which you must use to place markers for the concerts locations of a certain artist/band on a map.
- You are free to use the Map API you found more appropriate.

Instructions

- The backend must be written in **Go**
- You must handle website errors
- The code must respect the good practices
- It is recommended to have **test files** for unit testing.

Allowed packages

- Only the standard Go packages are allowed

Usage

- Here is a simple example of the process of giving an address and returning a marker to the location.

This project will help you learn about :

- Manipulation and storage of data
- HTML
- Manipulation of Maps API
- Geolocation, geocoding, etc
- Event creation and display
- JSON files and format

groupie-tracker-search-bar

Objectives

You must follow the same principles as the first subject.

Groupie tracker search bar consists of creating a functional program that searches, inside your website, for a specific text input. So the focus of this project is to create a way for the client to search a member or artist or any other attribute in the data system you made.

- The program should handle at least these search cases :
  - artist/band name
  - members
  - locations
  - first album date
  - creation date
- The program must handle search input as case-insensitive.
- The search bar must have typing suggestions as you write.
  - The search bar must identify and display in each suggestion the individual type of the search cases. (ex: Freddie Mercury -> member)
  - For example if you start writing `"phil"` it should appear as suggestions `Phil Collins - member` and `Phil Collins - artist/band`. This is just an example of a display.

Example

Lets imagine you have created a card system to display the band data. The user can directly search for the band he needs. Here is an example:

- While the user is typing for the member he desires to see, the search bar gives the suggestion of all the possible options.

Instructions

- The program must be written in **Go**.
- The code must respect the **good practices**.

Allowed packages

- Only the standard Go packages are allowed

This project will help you learn about :

- Manipulation, display and storage of data.
- HTML.
- Events creation and display.
- JSON files and format.

groupie-tracker

Objectives

Groupie Trackers consists on receiving a given API and manipulate the data contained in it, in order to create a site, displaying the information.

- It will be given an API, that consists in four parts:
    - The first one, `artists`, containing information about some bands and artists like their name(s), image, in which year they began their activity, the date of their first album and the members.
    - The second one, `locations`, consists in their last and/or upcoming concert locations.
    - The third one, `dates`, consists in their last and/or upcoming concert dates.
    - And the last one, `relation`, does the link between all the other parts, `artists`, `dates` and `locations`.
- Given all this you should build a user friendly website where you can display the bands info through several data visualizations (examples : blocks, cards, tables, list, pages, graphics, etc). It is up to you to decide how you will display it.
- This project also focuses on the creation of events/actions and on their visualization.
    - The event/action we want you to do is known as a client call to the server (client-server). We can say it is a feature of your choice that needs to trigger an action. This action must communicate with the server in order to recieve information, ([request-response])(https://en.wikipedia.org/wiki/Request%E2%80%93response)
    - An event consists in a system that responds to some kind of action triggered by the client, time, or any other factor.

Instructions

- The backend must be written in **Go**.
- The site and server cannot crash at any time.
- All pages must work correctly and you must take care of any errors.
- The code must respect the **good practices**.
- It is recommended to have **test files** for unit testing.

Allowed packages

- Only the standard Go packages are allowed.

Usage

- You can see an example of a RESTful API here

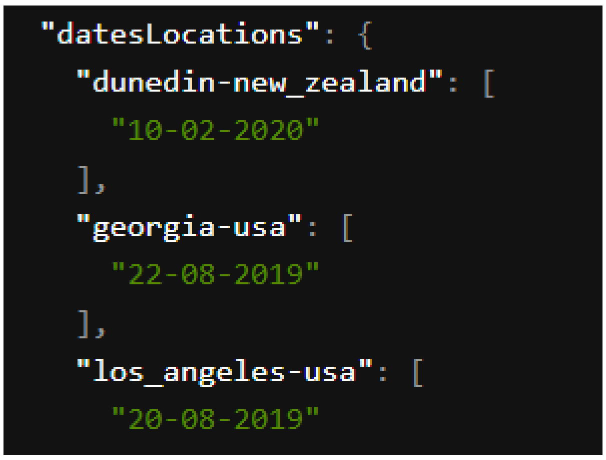
This project will help you learn about :

- Manipulation and storage of data.
- JSON files and format.
- HTML.
- Event creation and display.
- Client-server.

🎵 Groupie Tracker

Relations

Vous l'avez peut-être remarqué, mais la partie `relation` de l'API est un peu différente des autres, elle contient ces données :



```
"datesLocations": {
  "dunedin-new_zealand": [
    "10-02-2020"
  ],
  "georgia-usa": [
    "22-08-2019"
  ],
  "los_angeles-usa": [
    "20-08-2019"
```

Les clés étant dynamiques, on ne peut pas les utiliser de manière standard en JSON. On va donc utiliser un autre procédé.

Admettons que nous ne connaissons pas le nom des clés, nous allons utiliser un type *map* pour notre `datesLocations` :

```
type Date struct {
  Id int `json:"id"`
  DatesLocations map[string][]string `json:"datesLocations"`
}
```

Qu'est-ce qu'on fait ici ? Basiquement, chaque endroit est "relié" à un tableau de string qui correspond à une date, soit pour chaque string > []string. Ne connaissant pas le nom de la clé initiale, on dit à DatesLocations de "mapper" chaque string qu'il trouve, donc par exemple `dunedin-new_zealand` à un tableau de string lui appartenant.

Variables Go vers Javascript

Il est possible de faire passer ses variables Go en Javascript, c'est un peu tricky mais ça marche. Vous avez besoin de faire ça pour effectuer la barre de recherche et réussir le bonus `geolocalization`. Pour se faire, commencez par alimenter votre fichier .js comme ceci

```
let tableauAvecTousLesArtistes = []
let unSeulArtiste = ""
```

Puis, selon ce que vous voulez faire, modifiez votre HTML comme suit. Si vous souhaitez ajouter tous les artistes à une array en Javascript :

```
{{ range .Artists }}
<!-- Votre code -->
<script>
    tableauAvecTousLesArtistes.push("{{ . }}")
</script>
{{ end }}
```

Si vous voulez uniquement passer une seule valeur, en backend Go, ça restera classique, il faudra juste vérifier que vous ayez bien la bonne valeur passée dans le template exécuté :

```
package groupie

type Artist struct {
  Name string `json:"name"`
}

data := Artist{
  Name: "Micheal Jackson"
}

tmpl.ExecuteTemplate(w, "layout", Artist)
```

```
<script>
    unSeulArtiste = "{{ .Name }}"
    console.log(unSeulArtiste) // Vous devriez avoir Micheal Jackson d'affiché
</script>
```

Une fois ceci fait, vous avez la possibilité de tout coder dans la balise script, ce qui est très moche et déconseillé, ou alors vous avez la possibilité de renseigner un fichier .js pour rentrer votre code :

```
<script src="static/script.js"></script>
```

Tada ! La/les variable(s) que vous avez passé devrai(en)t se retrouver ici.

Range clé - valeur

Pour afficher en HTML au choix votre clé, votre valeur, ou les deux, venant d’une map, la syntaxe est simple :

```
{{ range $key, $value := .VotreMap }}
   <p>{{ $key }}</p>: <p>{{ $value }}</p>
{{ end }}
```