TP - Structures et Tableaux

Vous devez développer un programme Go pour gérer une liste de tâches. Chaque tâche est représentée par un nom et un statut (complet ou incomplet). Le programme doit permettre à l'utilisateur de :

1. Ajouter une tâche à la liste en saisissant son nom.
2. Marquer une tâche comme complète en saisissant son nom.
3. Afficher la liste des tâches, en indiquant leur nom et leur statut.

Votre programme devrait afficher le message suivant : "Bienvenue dans le gestionnaire de tâches !"

Ensuite, il devrait afficher un menu avec les options suivantes :

1. Ajouter une tâche
2. Marquer une tâche comme complète
3. Afficher la liste des tâches
4. Quitter

Après chaque action effectuée par l'utilisateur, le menu devrait être réaffiché jusqu'à ce que l'utilisateur choisisse l'option "Quitter".

Solution

```
package main

import "fmt"

type Task struct {
    Name   string
    Status string
}

func main() {
    fmt.Println("Bienvenue dans le gestionnaire de tâches !")

    tasks := make([]Task, 0)

    for {
```

```
fmt.Println("\nMenu:")
fmt.Println("1. Ajouter une tâche")
fmt.Println("2. Marquer une tâche comme complète")
fmt.Println("3. Afficher la liste des tâches")
fmt.Println("4. Quitter")

var choice int
fmt.Print("Votre choix : ")
fmt.Scan(&choice)

switch choice {
case 1:
    var name string
    fmt.Print("Nom de la tâche à ajouter : ")
    fmt.Scan(&name)

    task := Task{
        Name:   name,
        Status: "Incomplet",
    }

    tasks = append(tasks, task)
    fmt.Println("Tâche ajoutée avec succès !")

case 2:
    var name string
    fmt.Print("Nom de la tâche à marquer comme complète : ")
    fmt.Scan(&name)

    for i := range tasks {
        if tasks[i].Name == name {
            tasks[i].Status = "Complet"
            fmt.Println("Tâche marquée comme complète avec succès !")
            break
        }
    }

case 3:
    fmt.Println("Liste des tâches :")
    for _, task := range tasks {
```

```
                fmt.Printf("- %s : %s\n", task.Name, task.Status)
            }

        case 4:
            fmt.Println("Au revoir !")
            return

        default:
            fmt.Println("Choix invalide. Veuillez sélectionner une option valide.")
        }
    }
}
```

Explication du code :

- Une structure `Task` est définie avec deux champs : `Name` pour le nom de la tâche et `Status` pour le statut de la tâche (complet ou incomplet).
- La fonction `main()` est la fonction principale qui est exécutée lorsque le programme est lancé.
- Un slice vide `tasks` est créée pour stocker les tâches ajoutées par l'utilisateur.
- Le programme utilise une boucle `for` pour afficher le menu et traiter les choix de l'utilisateur jusqu'à ce que l'option "Quitter" soit sélectionnée.
- Selon le choix de l'utilisateur, différentes actions sont effectuées :
  - Pour l'option 1, l'utilisateur est invité à saisir le nom de la tâche à ajouter. Une nouvelle tâche est créée avec le statut "Incomplet" et ajoutée au slice `tasks`.
  - Pour l'option 2, l'utilisateur est invité à saisir le nom de la tâche à marquer comme complète. Le statut de la tâche correspondante dans une `tasks` est mis à jour.
  - Pour l'option 3, toutes les tâches dans les `tasks` sont affichées avec leur nom et leur statut.
  - Pour l'option 4, le programme se termine et affiche "Au revoir !".
  - Si l'utilisateur choisit une option invalide, un message d'erreur est affiché.

Revision #2
Created 2023-05-19 13:09:09 UTC by Noé Larrieu-Lacoste
Updated 2023-05-19 13:37:32 UTC by Noé Larrieu-Lacoste