Inteligentne Aplikacje Internetowe

Laboratorium 1 – część 2

Opracowanie prostej aplikacji internetowej umożliwiającej ocenę obiektów

Serwer HTTP

Celem tej sekcji jest zaprezentowanie prostego serwera http

Jednym z najprostszych sposobów uruchomienia serwera jest wywołanie funkcji **http.ListenAndServe** - pierwszym parametrem jest adres (można go pominąć) i port, drugi jest opcjonalny i będzie omówiony w kolejnych laboratoriach:

```
http.ListenAndServe(":8080", nil)
```

W momencie uruchomienia powyższej funkcji kod programu **zatrzymuje się** na tym poleceniu i rozpoczyna obsługę zapytań serwera. Zapytania serwera należy wcześniej ustawić, jednym z najprostszych sposobów jest obsługa wzorców (*ang.* pattern) i w momencie ich zgodności wywołanie konkretnej funkcji programu – służy do tego metoda **http.HandleFunc**. Przedstawia to poniższy przykład:

```
package main

import (
    "fmt"
    "net/http"
)

func indexFunc(w http.ResponseWriter, r *http.Request) {
    // zwrócenie strony głównej
    fmt.Fprintf(w, "<html><body>STRONA GŁÓWNA</body></html>")
}

func itemFunc(w http.ResponseWriter, r *http.Request) {
    // zwrócenie strony /item/* (* - oznacza dowolny ciąg znaków)
    fmt.Fprintf(w, "<html><body>STRONA ITEM<br>ADRES: ")
    fmt.Fprintf(w, r.RequestURI)
    fmt.Fprintf(w, "<br>METODA: ")
    fmt.Fprintf(w, r.Method)
    fmt.Fprintf(w, "</body></html>")
}

func main() {
    http.HandleFunc("/", indexFunc)
    http.HandleFunc("/item/", itemFunc)
    http.ListenAndServe("localhost:8080", nil)
}
```

Powyższy przykład można uruchomić i przetestować adresy:
localhost:8080/ localhost:8080/item/ localhost:8080/items localhost:8080/item/15

Wymuszenie zatrzymania serwera można uzyskać przez skrót w terminalu Ctrl + C

W powyższym przykładzie zawartość stron jest generowana "ręcznie". Istnieje jednak wiele innych możliwości przesyłania treści stron internetowych oraz plików, np.:

- Zwracanie statycznych plików:

```
http.ServeFile(w, r, "pages/strona.html")
```

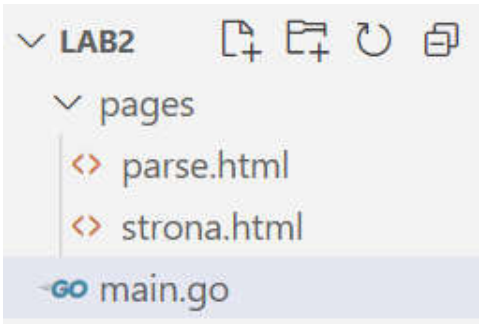
- Zwracanie parsowanych plików .html (w takich plikach można umieszczać treść dynamicznie), czyli wykorzystanie tzw. szablonów (*ang.* template):

```
tmpl, _ := template.ParseFiles("pages/parse.html")
tmpl.Execute(w, [data])
```

W powyższym kodzie [data] oznacza opcjonalną strukturę danych przekazywanych do strony. Znak _ oznacza że drugi parameter zwracany przez ParseFiles nie będzie wykorzystywany. Podmianę danych umożliwia zapis na stronie .html pomiędzy nawiasami {{ oraz }}. Dostęp do pól struktury odbywa się przez zapis: .NazwaPola (pola struktury muszą być publiczne – pisane z dużych liter – więcej o dostępie do pól w kolejnych laboratoriach).

Do przetestowania powyższych funkcjonalności należy dodać do folderu roboczego folder **pages** oraz umieścić w nim pliki: strona.html (o dowolnej treści HTML) i parse.html o następującej zawartości:

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <table>
            <tr>
                <th>Imię</th>
                <th>Indeks</th>
            </tr>
            <tr>
                <td>{{ .Name }}</td>
                <td>{{ .Index }}</td>
            </tr>
        </table>
    </body>
</html>
```



Następnie należy dodać w funkcji main obsługę dwóch kolejnych wzorców (oczywiście przed rozpoczęciem startu serwera):

```
    http.HandleFunc("/strona/", stronaFunc)
    http.HandleFunc("/parse/", parseFunc)
```

Do kodu programu należy również dodać strukturę student (będzie zawierała pola .Name oraz .Index, które będą przekazywane do strony parse.html) oraz funkcje stronaFunc i parseFunc. Implementację tych elementów przedstawia następujący kod źródłowy:

```go
func stronaFunc(w http.ResponseWriter, r *http.Request) {
    // zwrócenie statycznej strony strona.html
    http.ServeFile(w, r, "pages/strona.html")
}

type student struct {
    Name  string
    Index int
}

func parseFunc(w http.ResponseWriter, r *http.Request) {
    // zwrócenie strony o dynamicznej zawartości
    tmpl, _ := template.ParseFiles("pages/parse.html")
    tmpl.Execute(w, &student{"Jan", 12345})
}
```

W celu przetestowania drugiej części przykładu należy uruchomić poniższe adresy w przeglądarce:
localhost:8080/strona/ localhost:8080/parse/

Następnie można zmienić pola struktury student na małą literę (name, index) oraz właściwości w pliku parse.html na {{ .name }} oraz {{ .index }} i sprawdzić czy strona działa poprawnie.

Zadanie 1

Celem zadania jest stworzenie prostego serwera zapisującego odsłony produktów do pliku. Na podstawie takich odsłon możliwe jest późniejsze ocenienie popularności produktów.

1. Utworzyć nowy folder roboczy (np. LAB1-Z1).
2. Utworzyć plik **main.go** z funkcją main oraz folder **pages**
3. W pliku main.go utworzyć strukturę ***item*** zawierającą następujące pola:
   Name (string), Price (float)
4. Utworzyć serwer http obsługujący stronę /item/{id} przez funkcję itemFunc.
5. Funkcja itemFunc powinna wyświetlać stronę z danymi produktu o wybranym identyfikatorze. Dane produktów powinny być odczytywane z tablicy globalnej jako przykład. Realizację tej części zadania należy zacząć od dodania następującego kodu:

```go
var items = [3]item{
    {"OnePlus 10 Pro", 4500},
    {"Asus X513EA", 2000},
    {"Samsung UE55TU7022K", 1900},
}

func itemFunc(w http.ResponseWriter, r *http.Request) {
    id, _ := strconv.Atoi(r.PathValue("id")) // pobranie ID przedmiotu
    // zadanie (miejsce poniżej na własny kod)
    // wyświetić stronę z informacjami o przedmiocie o danym id
    // korzystanie z tablicy globalnej: - &items[id]
}
```

Uwaga: wymagany jest język Go w wersji 1.22 lub wyższej

6. Po implementacji kodu sprawdzić działanie programu dla następujących adresów:
   localhost:8080/item/ localhost:8080/item/0 localhost:8080/item/1
   localhost:8080/item/2 localhost:8080/item/3 localhost:8080/item/zet
7. Zmodyfikować kod funkcji itemFunc tak, aby przy wyświetlaniu informacji o konkretnym produkcie program zapisywał do pliku logs.txt identyfikator przedmiotu oraz bieżącą datę. Dopis danych do plików może wyglądać następująco:

```
file, _ := os.OpenFile("logs.txt", os.O_APPEND|os.O_CREATE, 0600)
file.WriteString(fmt.Sprint(id) + "\t" + time.Now().String() + "\n")
file.Close()
```

8. Wynik zaprezentować prowadzącemu w celu zaliczenia laboratorium.
9. **Zadanie na plus**
   Rozszerzyć działanie programu tak, aby w przypadku gdy przedmiotu nie ma w tablicy items wyświetlała się strona z błędem – nie znaleziono przedmiotu lub indeks jest większy niż liczba przedmiotów w tablicy.
10. **Zadanie na dwa plusy**
    Zamiast przedmiotów (item) wyświetlać na stronie informacje o samochodach (samochody wczytać tak jak w poprzednim laboratorium z pliku **cars.txt**, ale wynik funkcji wczytującej umieścić w zmiennej globalnej **cars**). Znaleźć samodzielnie sposób na wyświetlenie strony z wszystkimi linkami do podstron z samochodami.