**Отчет по лабораторной работе № 0**

Тарасов А. Т.

Группа ИУ9-31Б

**Цель:** Разработка простейшего web-сервера на языке GO на основе пакета net/http.

Заявленной цели соответствуют следующие **задачи:**

1. Изучить принимаемые web-сервером параметры, реализовать передачу данных методом GET.

2. Реализовать вывод форматированного гипертекста:

контекстное меню в виде гиперссылок, при клике на гиперссылку выполняется подмена контекста / адресация;

3. Реализовать получение данных из различных RSS- каналов и вывод результатов обработки данных в структурированном виде.

4. Реализовать сортировку данных по дате публикации.

**Исходный код программы:**

package main

/// Импорт библиотек

import (

    "fmt";

    "net/http";

    "github.com/mmcdole/gofeed"

    "html/template"

    "sort"

    "time"

    "log";

    )

/// Структура для отображения контента

type ViewData struct {

    Items []\*ItemView

    Urls []string

    Names []string

}

/// Структура для отображения новости

type ItemView struct {

    Title string

    Content template.HTML

    Source string

    Date time.Time

}

var feedParser = gofeed.NewParser()

var urls []string

var names []string

/// Преобразование из структуры библиотеки gofeed в ItemView

func MapItem(item \*gofeed.Item, Source string) \*ItemView {

    return &ItemView{Title : item.Title,

                    Content : template.HTML(item.Description),

                    Source : Source,

                    Date : \*item.PublishedParsed}

}

/// Парсинг новостей

func GetItems(url string) \*[]\*ItemView  {

    feedData, err := feedParser.ParseURL("http://" + url)

    if err!= nil {

        log.Fatal("ListenAndServe: ", nil)

    }

    fmt.Println("Parsed: " + url)

    items := make([]\*ItemView, len(feedData.Items))

    for i:= 0; i < len(feedData.Items); i++{

        items[i] = MapItem(feedData.Items[i], feedData.Title)

    }

    return &items

}

/// Обработчик по умолчанию

func HomeRouterHandler(w http.ResponseWriter, r \*http.Request) {

    allItems := make([]\*ItemView, 0)

    for \_, url := range urls {

        items := GetItems(url)

        allItems = append(allItems, (\*items)...)

    }

    /// Сортируем новости по дате и времени публикации

    sort.SliceStable(allItems, func(i, j int) bool {

        return allItems[i].Date.Unix() > allItems[j].Date.Unix()

    })

    data := ViewData{Items : allItems, Urls : urls, Names : names}

    t, \_ := template.ParseFiles("./index.html")

    t.Execute(w, data)

}

/// Обработчик url конкретного новостного ресурса

func PageHandler(url string) func (w http.ResponseWriter, r \*http.Request) {

    return func (w http.ResponseWriter, r \*http.Request) {

        items := GetItems(url)

        data := ViewData{Items : \*items, Urls : urls, Names : names}

        t, \_ := template.ParseFiles("./index.html")

        t.Execute(w, data)

    }

}

/// Точка входа в программу, назначаем обработчики

/// инициализируем глобальные переменные с url'ами и названиями

/// новостных ресурсов

func main() {

    urls = []string{"lenta.ru/rss", "news.ap-pa.ru/rss.xml"}

    names = []string{"lenta", "news"}

    http.HandleFunc("/", HomeRouterHandler)

    for i := 0; i < len(urls); i++{

        http.HandleFunc("/" + names[i], PageHandler(urls[i]))

    }

    err := http.ListenAndServe(":9034", nil)

    if err!= nil {

        log.Fatal("ListenAndServe: ", nil)

    }

}

**Пример работы программы:**



