Consuming REST API

LAB 09

Learning Outcomes

After completing this lab you should be able to

- Make HTTP request using Go code
- Build a REST Client which consumes REST APIs

Sample Code

`https://github.com/betsegawlemma/go-rest-client`

The API to be consumed

`https://reqres.in/`

URI for listing all Users in a given page

`https://reqres.in/api/users?page=2`

URI for getting a single User by its id

`https://reqres.in/api/users/2`

GET https://reqres.in/api/users?page=2


```
{
    "page": 2,
    "per_page": 6,
    "total": 12,
    "total_pages": 2,
    "data": [
        {
            "id": 7,
            "email": "michael.lawson@reqres.in",
            "first_name": "Michael",
            "last_name": "Lawson",
            "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/follettkyle/128.jpg"
        },
        {
            "id": 8,
            "email": "lindsay.ferguson@reqres.in",
            "first_name": "Lindsay",
            "last_name": "Ferguson",
            "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/araa3185/128.jpg"
        },
```

GET https://reqres.in/api/users/2


```
{
    "data": {
        "id": 2,
        "email": "janet.weaver@reqres.in",
        "first_name": "Janet",
        "last_name": "Weaver",
        "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/josephstein/128.jpg"
    }
}
```

Creating Data Structure

```
// User represents User data
type User struct {
    ID        int    `json:"id"`
    Email     string `json:"email"`
    FirstName string `json:"first_name"`
    LastName  string `json:"last_name"`
    Avatar    string `json:"avatar"`
}
```

Creating Data Structure

```
// SingleData represents a single User
type SingleData struct {
    User User `json:"data"`
}

// CollectionData represents collection of Users
type CollectionData struct {
    Users []User `json:"data"`
}
```

Fetch Single User

```
var baseURL =
"https://reqres.in/api/
users/"
```

```
//FetchUser fetchs a single user by its id
func FetchUser(id int) (*User, error) {
    client := &http.Client{}
    URL := fmt.Sprintf("%s%d", baseURL, id)
    req, _ := http.NewRequest("GET", URL, nil)
    res, err := client.Do(req)
    //res, err := client.Get(URL)
    if err != nil {
        return nil, err
    }
    userdata := &SingleData{}
    body, err := ioutil.ReadAll(res.Body)
    if err != nil {
        return nil, err
    }
    err = json.Unmarshal(body, userdata)
    if err != nil {
        return nil, err
    }
    return &userdata.User, nil
}
```

The client making GET request on the API
GET
https://reqres.in/api/users/1

Reading the response body in JSON format

Converting the JSON object into the SingleData struct form

Returning the User object

```go
//FetchUser fetchs a single user by its id
func FetchUser(id int) (*User, error) {
    client := &http.Client{}
    URL := fmt.Sprintf("%s%d", baseURL, id)
    req, _ := http.NewRequest("GET", URL, nil)
    res, err := client.Do(req)
    //res, err := client.Get(URL)
    if err != nil {
        return nil, err
    }
    userdata := &SingleData{}
    body, err := ioutil.ReadAll(res.Body)
    if err != nil {
        return nil, err
    }
    err = json.Unmarshal(body, userdata)
    if err != nil {
        return nil, err
    }
    return &userdata.User, nil
}
```

Fetching Users

```
var baseURL =
"https://reqres.in/api/
users/"
```

```
// FetchUsers fetchs all users on a given page
func FetchUsers(page int) ([]User, error) {
    client := &http.Client{}
    URL := fmt.Sprintf("%s?page=%d", baseURL, page)
    req, _ := http.NewRequest("GET", URL, nil)
    res, err := client.Do(req)
    //res, err := client.Get(URL)
    if err != nil {
        return nil, err
    }
    usdata := &CollectionData{}
    body, err := ioutil.ReadAll(res.Body)
    if err != nil {
        return nil, err
    }
    err = json.Unmarshal(body, usdata)
    if err != nil {
        return nil, err
    }
    return usdata.Users, nil
}
```

The client making GET request on the API
GET
https://reqres.in/api/users?page=2

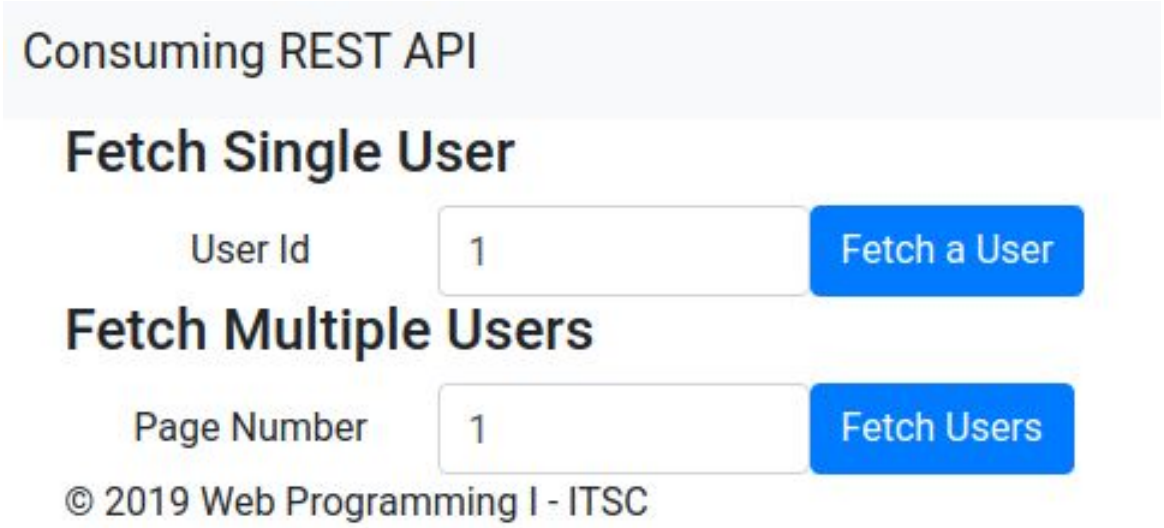
Reading the response body in JSON format

Converting the JSON object into the CollectionData struct form

Returning the []User object

```go
// FetchUsers fetchs all users on a given page
func FetchUsers(page int) ([]User, error) {
    client := &http.Client{}
    URL := fmt.Sprintf("%s?page=%d", baseURL, page)
    req, _ := http.NewRequest("GET", URL, nil)
    res, err := client.Do(req)
    //res, err := client.Get(URL)
    if err != nil {
        return nil, err
    }
    usdata := &CollectionData{}
    body, err := ioutil.ReadAll(res.Body)
    if err != nil {
        return nil, err
    }
    err = json.Unmarshal(body, usdata)
    if err != nil {
        return nil, err
    }
    return usdata.Users, nil
}
```

The Web Client form



Form: Fetch Single User

```
<h4>Fetch Single User</h4>
   <form class="form-inline" method="post" action="/user">
       <label for="id" class="col-sm-2 col-form-label">User Id</label>
       <input type="number" form-control" name="id" id="id" value="1" >
       <button type="submit" class="btn btn-primary">Fetch a User</button>
   </form>
```

Form: Fetch Multiple Users

```
<h4>Fetch Multiple Users</h4>
   <form class="form-inline" method="post" action="/users">
       <label for="page" class="col-sm-2 col-form-label">Page Number</label>
       <input type="number" name="page" id="page" value="1">
       <button type="submit" class="btn btn-primary">Fetch Users</button>
   </form>
```

main function

```
var tmpl = template.Must(template.ParseGlob("ui/templates/*.html"))

func main() {

    mux := http.NewServeMux()
    fs := http.FileServer(http.Dir("ui/assets"))
    mux.Handle("/assets/", http.StripPrefix("/assets/", fs))

    mux.HandleFunc("/", index)
    mux.HandleFunc("/user", singleUser)
    mux.HandleFunc("/users", allUsers)

    http.ListenAndServe(":8181", mux)
}
```

singleUser handler

```
func singleUser(w http.ResponseWriter, r *http.Request) {
    idraw := r.FormValue("id")
    id, _ := strconv.Atoi(idraw)

    user, err := data.FetchUser(id)

    if err != nil {
        w.WriteHeader(http.StatusNoContent)
        tmpl.ExecuteTemplate(w, "error.layout", nil)
    }
    tmpl.ExecuteTemplate(w, "user.layout", user)
}
```

user.layout Page

```
{{ define "user.layout" }}
{{ template "navbar" . }}
{{ template "user.content" . }}
{{ template "footer" . }}
{{ end }}
{{ define "user.content" }}
<div class="container">
   <h2>A Single User</h2>
   <p><b>ID:</b> {{ .ID}} </p>
   <p><b>Email:</b> {{ .Email }}</p>
   <p><b>FirstName:</b> {{ .FirstName }}</p>
   <p><b>LastName:</b>{{ .LastName }}</p>
   <p><b>Avatar</b></p>
   <img src="{{.Avatar}}" width="70rem" height="70rem" />
</div>
{{ end }}
```

allUsers Handler

```
func allUsers(w http.ResponseWriter, r *http.Request) {
    pageraw := r.FormValue("page")
    page, err := strconv.Atoi(pageraw)

    users, err := data.FetchUsers(page)

    if err != nil {
        w.WriteHeader(http.StatusNoContent)
        tmpl.ExecuteTemplate(w, "error.layout", nil)
    }
    tmpl.ExecuteTemplate(w, "users.layout", users)
}
```

users.layout Page

```
{{ define "users.layout" }}
{{ template "navbar" . }}
{{ template "users.content" . }}
{{ template "footer" . }}
{{ end }}
{{ define "users.content" }}
```

```
{{ define "users.content" }}
<div class="container">
        <h2>List of Users</h2>
    <table class="table table-striped">
        <thead> ...   </thead>
        <tbody>
            {{ range . }}
          <tr>
            <td> {{ .ID }} </td>
            <td> {{ .Email }} </td>
            <td> {{ .FirstName }} </td>
            <td> {{ .LastName }} </td>
            <td> <img src="{{.Avatar}}" /> </td>
          </tr>
          {{ end }}
          </tbody>
          </table>
</div>
{{ end }}
```