**LẬP TRÌNH MẠNG CĂN BẢN – NT106.O21.ANTN**

**BÁO CÁO THỰC HÀNH LAB 4**

# \*\*\*

**Họ và Tên:** Dương Phạm Huy Thông

**MSSV:** 22521431

**Link code:** [tại đây](https://drive.google.com/file/d/1mon7PXvr3xDZQZwWPw_7fu_Oz4k5PXxP/view?usp=drive_link)

*Bài làm báo cáo những phần được yêu cầu bổ sung sau khi đã báo cáo toàn bộ bài Lab tại lớp trong buổi thực hành số 2.*

# BÀI LÀM

**Bài 1: Lấy và hiển thị nội dung và Header**

- Từ URL được nhập vào bởi người dùng, ta thực hiện viết các hàm nhằm lấy thông tin về HTML và Header của một trang web.

A screen shot of a computer program

Description automatically generated

- Trước tiên, ta cần kiểm tra URL được nhập bởi người dùng đã chính xác với định dạng hay chưa, nếu chưa sẽ hiện thông báo cho người dùng.

- Tạo một **WebRequest** từ URL được nhập bởi người dùng để tạo một yêu cầu cho webserver. Sau đó tạo ra một **WebResponse** để nhận các thông điệp trả về từ **WebRequest** đã được tạo.

- Sau khi nhận được thông điệp từ server, ta thực hiện tạo một luồng ghi dữ liệu bằng đối tượng **StreamReader** và cuối cùng trả về giá trị của nội dung được cung cấp bởi server.

A screenshot of a computer program

Description automatically generated

- Với việc lấy thông tin Header mà server phản hồi ta cũng thực hiện tương tự, trong đó ta sẽ bắt đầu với việc tạo một đối tượng **WebRequest** và **WebResponse** tương tự như trên.

- Sau đó ta duyệt qua tất cả các trường được trả về trong Header thông qua đối tượng **WebResponse** và thực hiện them từng đối tượng trong Header vào **ListView** để hiển thị.

A computer screen with text and symbols

Description automatically generated with medium confidence

- Thực hiện sử lý sự kiện **buttonDownload\_Click** để chương trình tiến hành lấy nội dung HTML và Header của trang Web, trong đó yêu cầu người dùng nhập vào URL, và truyền URL vào các hàm chức năng sau đó hiển thị kết quả ra màn hình cho người dùng.

**Demo:**

|  |
| --- |
|  |
|  |

**Bài 2: Tiện ích giả lập – Device Emulator**

- Yêu cầu người dùng nhập vào một URL bất kỳ, sau đó kiếm tra định dạng của URL đã đúng với mong muốn hay không. Nếu đúng sẽ thực hiện tạo một yêu cầu HTTP đối với webserver thông qua đối tượng **HttpWebRequest**.

- Trước khi gửi thông điệp ta thực hiện lựa chọn **User-Agent** từ người dùng lựa chọn (cung cấp sẵn một số giả lập thiết bị) và gửi đến webserver, sau đó tương tự như Bài 1 ta thực hiện tạo luồng để ghi nhận lại những nội dung được phản hồi lại từ webserver.

**Demo:**

|  |
| --- |
|  |
|  |

**Bài 3: Làm việc với API**

**-** Trước tiên, chương trình thu thập các giá trị nhập vào từ hai ô văn bản trên giao diện người dùng (**cityNameText** và **countryCodeText**) để lấy tên thành phố và mã quốc gia.

- Xây dựng URL của **API OpenWeatherMap** bằng cách sử dụng tên thành phố, mã quốc gia và khóa API (apiKey) đã cung cấp. Tham số **units=metric** được sử dụng để yêu cầu dữ liệu nhiệt độ ở đơn vị ***Celsius***.

- Sử dụng **HttpClient** để gửi yêu cầu GET đến URL API đã xây dựng. Việc gửi yêu cầu này được thực hiện bất đồng bộ, cho phép ứng dụng tiếp tục thực thi trong khi đợi phản hồi từ máy chủ.

- Nếu yêu cầu thành công, mã đọc nội dung của phản hồi về dưới dạng một chuỗi sử dụng **ReadAsStringAsync()** và sau đó phân tích chuỗi JSON thành các đối tượng **dynamic** bằng cách sử dụng thư viện **Newtonsoft.Json.**

- Sau khi phân tích và lấy dữ liệu thời tiết từ đối tượng JSON, chương trình sử dụng các thông tin này để cập nhật giao diện người dùng bằng cách đặt các giá trị vào các đối tượng hiển thị thông tin về thành phố, nhiệt độ và mô tả thời tiết.

A computer screen shot of a program code

Description automatically generated

**Demo:**

A screenshot of a weather login

Description automatically generated

**Bài 4: Trình duyệt cơ bản**

Với bài sau, ta thực hiện 3 yêu cầu:

**a. Xem nội dung Website (Render HTML)**

**A computer screen shot of text

Description automatically generated**

- Từ URL được nhập vào bởi người dùng, ta thực hiện kiểm tra định dạng của URL, nếu không có vấn đề với URL ta sẽ thực hiện render trang HTML thông qua phương thức **Navigate** của **WebBrowser.**

**b. Download Source(Lưu trữ mã nguồn)**

**A screen shot of a computer program

Description automatically generated**

- Tạo một thư mục tạm thời (**src**) trong thư mục hiện tại để lưu trữ tất cả các tài nguyên được tải xuống từ trang web. Sử dụng **HttpClient**, phương thức này tải nội dung HTML của trang web từ URL đã cung cấp.

- Nội dung HTML được lưu vào tệp **index.html** trong thư mục **src**. Sử dụng **HtmlAgilityPack**, phương thức này phân tích HTML để tìm và tải xuống các tài nguyên như hình ảnh, CSS và JavaScript. Nó duyệt qua các thẻ HTML quan trọng như **<img>, <link>, <script>** và tải các tài nguyên từ các thuộc tính như **src** hoặc **href**.

A screen shot of a computer program

Description automatically generated

- **DownloadResourcesAsync** được sử dụng để tải và lưu trữ các tài nguyên (như hình ảnh, CSS, JavaScript) từ trang web đã được tải về. Nó sử dụng thư viện HttpClient để thực hiện các yêu cầu mạng và lưu trữ các tài nguyên tải xuống vào thư mục đích.

- Phương thức này duyệt qua các thẻ HTML quan trọng như **<img>, <link>, <script>** để tìm các tài nguyên cần tải xuống.

- Với mỗi thẻ HTML tìm thấy, nó lấy giá trị của thuộc tính tương ứng (như **src** cho **<img>**, **href** cho **<link>, src** cho **<script>**).

- Nếu thuộc tính này tồn tại, nó xây dựng URL tuyệt đối bằng cách kết hợp **baseUrl** và giá trị thuộc tính, sau đó gọi phương thức **DownloadResourceAsync** để tải và lưu trữ tài nguyên.

A screen shot of a computer code

Description automatically generated

- **DownloadResourceAsync** tải một tài nguyên từ URL và lưu trữ nó vào thư mục đích. Phương thức này sử dụng **HttpClient** để tải nội dung của tài nguyên dưới dạng mảng byte. Sau đó, nó xác định tên tệp và loại tài nguyên (dựa trên phần mở rộng của tên tệp). Nó xác định thư mục đích dựa trên loại tài nguyên và tạo thư mục nếu chưa tồn tại. Cuối cùng, nó lưu trữ tài nguyên vào thư mục đích bằng cách gọi phương thức **WriteBytesToFileAsync**.

A screen shot of a computer program

Description automatically generated

**c. Xem Source (Xem mã nguồn)**

- Lấy ý tưởng từ Bài 1 thực hiện yêu cầu này

A computer screen with text and images

Description automatically generated with medium confidence

**Demo:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Bài 1: Lấy và hiển thị nội dung và Header (Mở rộng)**

- Sử dụng Apache làm webserver và khi chưa cấu hình thông tin sẽ hiển thị như sau:

A screenshot of a computer

Description automatically generated

- Sau đó thực hiện cấu hình:

+ Kích hoạt: *LoadModule headers\_module modules/mod\_headers.so*

+ Thông 2 trường thông tin: *ServerTokens Prod* và *ServerSignature Off* nhằm chỉ định rằng máy chủ sẽ chỉ hiển thị **"ProductOnly" (Prod)** trong các trường **Server response header** thay vì hiển thị phiên bản cụ thể của Apache và chỉ thị để tắt việc hiển thị chữ ký (signature) của máy chủ trong các trang lỗi do máy chủ tạo ra (ví dụ: lỗi 404). Chỉ thị này sẽ không hiển thị bất kỳ thông tin cụ thể nào về máy chủ.

+ Thông thông tin làm giả **User-Agent**: *Header always set User-Agent "Hello NT106"*

- Sau đó thực hiện khởi động lại **apache** để các thay đổi được ghi nhận.

A screenshot of a computer

Description automatically generated

**--Hết--**