**Bộ môn An toàn Thông tin – Khoa MMT&TT**
**Trường Đại học Công nghệ Thông tin (UIT)**

# LAB REPORT

**Subject: Basic network programming**

**(Session 04)**

**Topic: Lab 4**

**Group: Ningguang**

## 1. GENERAL INFORMATION:

Class: NT106.M21.ATCL.1

| Ordinal number | Full name | Student ID | Email |
|---|---|---|---|
| 1 | Truong Thi Hoang Hao | 20520191 | 20520191@gm.uit.edu.vn |
| 2 | Nguyen Dinh Kha | 20520562 | 20520562@gm.uit.edu.vn |

## 2. IMPLEMENTATION CONTENT:

| Ordinal number | Work | Self-assessment result |
|---|---|---|
| 1 | Scenario 01 – Get request | 100% |
| 2 | Scenario 02 – Post request | 100% |
| 3 | Scenario 03 – Download file from website | 100% |
| 4 | Scenario 04 – Web browser | 100% |

**The section below this report is the detailed report document of the implementation team**

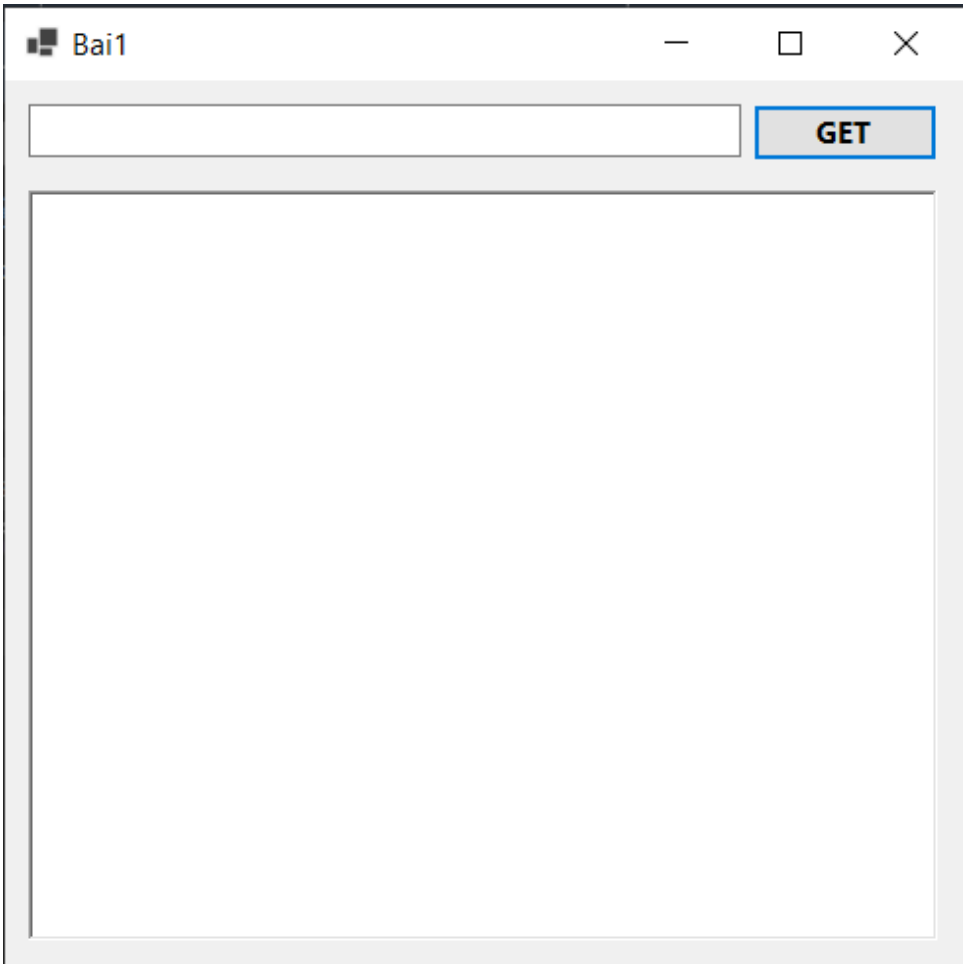
# DETAILED REPORT

## 1. Scenario 01

- Resource: .NET core 6.0
- Objective: Write a program to display the HTML content of any web page.

- Steps to implement:

– Step 1: Design an interface that includes a textbox for the user to enter a URL, a "GET" button that sends a request, and a richtextbox to display the information returned from the web server.




– Step 2:

+ Before coding the functionality of the "GET" button, create an object of the HttpClient class to be used throughout the form's operation.

```
// Instantiate HttpClient once and re-used throughout the life of an application.
private static readonly HttpClient client = new HttpClient();
```

+ Next, proceed to code the functionality of the "GET" button:

```
private async void btnGet_Click_1(object sender, EventArgs e)
{
    // Get the url from textbox
    string url = getUrl.Text.Trim();
    if (url != string.Empty)
```

First, add 'async' to the function name because 'await' will be used within this function. Then, process the data in the textbox (the web page URL entered by the user). Confirm there is a value in the textbox before moving to the next step.

```
/*
 * Send a GET request to the url above
 * Return the response body as a string in an asynchronous operation.
 */
string responseString = await client.GetStringAsync(url);
if (responseString != string.Empty)
{
  // Display the response
  view.Text = responseString;
}
```

The GetStringAsync(String) method sends a GET request to the specified Uri and returns the response body as a string. If the response string is not empty, it is displayed in the richtextbox.

## 2. Scenario 02

- Resource: .NET core 6.0
- Objective: Write a program to send arbitrary data to the address: http://www.contoso.com/PostAcceptor.aspx using the POST method

- Steps to implement:

– Step 1: Designing the User Interface:

Textbox for URL Input: A textbox is designed for users to enter the URL of the website to which they want to send a POST request.Textbox for Post Content: Another textbox is provided for users to input the content of the post they wish to send.

Richtextbox for Displaying Server Response: A richtextbox is included in the UI to display information returned from the web server in response to the POST request.

POST Button: A button labeled "POST" is added to the interface. When clicked, it triggers the sending of the POST request with the data provided by the user.


- Step 2: Coding the Functionality of the POST Button:

Initialization of HttpClient Object: Before coding the functionality of the "POST" button, an instance of the HttpClient class is created. This object is used throughout the form's lifecycle to interact with web servers.

```
// Instantiate HttpClient once and re-used throughout the life of an application.
private static readonly HttpClient client = new HttpClient();
```

```
private async void btnPost_Click(object sender, EventArgs e)
{
    // Get the url from text box
    string url = getUrl.Text.Trim();
```

Async Function Declaration: The function that handles the POST request is declared as asynchronous (async) to allow the use of await for asynchronous operations, such as sending the HTTP request and waiting for the response without blocking the UI thread. Processing URL Input: The user-provided URL from the textbox is prepared for the HTTP request.

```
// Get the post info and convert into a JSON string
string json = JsonSerializer.Serialize(post_value.Text.Trim());
// Convert string into HTTP content
StringContent content = new StringContent(json, Encoding.UTF8);
```

Preparing Post Content: The content for the post, entered by the user in the textbox, is converted into a JSON string and then encoded into UTF8. This encoded string is wrapped in a StringContent object (a subclass of HttpContent) for inclusion in the POST request.

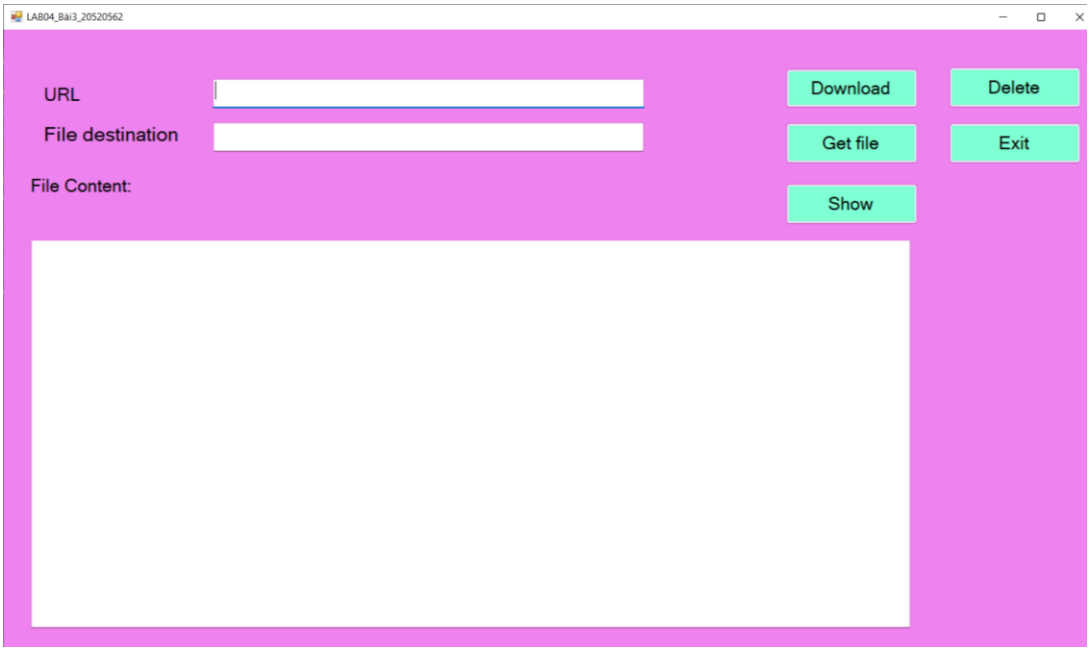
```
// Send a POST request to the url above as an asynchronous operation.
HttpResponseMessage responseMess = await client.PostAsync(url, content);
// Serialize the HTTP content to a string as an asynchronous operation.
string response = responseMess.Content.ReadAsStringAsync().Result;
// Display the response
view.Text = response;
```

The POST request is sent using the HttpClient object, with the StringContent object containing the post content. The response from the server is then displayed in the richtextbox.

3. **Scenario 03:**
   - **Resource:** Attached compressed File
   - **Objective:** Write a program to download the content of any website from any URL and write it to an HTML file, then display the website content on a form

   - **Steps to implement:**
   - Step 1: Designing the Interface:

Create a user interface with two text boxes (textURL and textFileDes) and five buttons (Download, Get File, Show, Delete, Exit).
The Download button is for downloading web content to a specified file. The URL is entered in the textURL textbox, and the file destination is specified by the Get File button, which selects an existing empty HTML file for the content to be saved into.
The Show button displays the content of the downloaded HTML file in a textbox, presumably called txtView.
The Delete button is for clearing the displayed data, allowing for new data to be entered and processed.
The Exit button closes the application.




– Step 2: Initialization and Reading Content:

```
namespace LAB04_Bài3_20520562
{
    3 references
    public partial class Form1 : Form
    {
        WebClient wc = new WebClient();
```

```
1 reference
private string OpenToRead(string s)
{
    string clarification = "";

    // hàm OpenRead dùng để mở luồng có thể đọc được cho dữ liệu được tải xuống từ tài nguyên với URL được chỉ định là chuỗi.
    Stream str = wc.OpenRead(s);
    // Dùng StreamReader để đọc file
    StreamReader sr = new StreamReader(str);
    // hàm .Peek() để trả về ký tự có sẵn tiếp theo nhưng không tiêu thụ nó.
    while (sr.Peek() > -1)
    {
        clarification += sr.ReadLine();
    }
```

- Initialize a WebClient object to fetch web content.
- Implement an OpenToRead function using WebClient's OpenRead() method from the System.IO library to read the content of a web page into a stream.
- Use StreamReader to read from the file and the Peek() method to check for the next available character without consuming it, which helps in reading the stream efficiently.

– Step3: Implementing Button Functionalities:

+ For the Download button:

```
// hàm .Trim() Trả về một chuỗi mới trong đó tất cả các lần xuất hiện hàng đầu
// và theo dõi của một tập hợp các ký tự được chỉ định từ chuỗi hiện tại được loại bỏ.
if (textURL.Text.Trim() == "")
{
    MessageBox.Show("ERROR !!! Please enter the correct URL. ");
    // hàm .Focus() có tác dụng trả về true nếu yêu cầu lấy nét đầu vào thành công; nếu không, false.
    textURL.Focus();
    return;
}
```

- Verify the URL format and the destination path before downloading the content using the WebClient's DownloadFile method. If the operation is successful, indicate via a message box.

```
if (fileSaveDes.Text.Trim() == "")
{
    MessageBox.Show("ERROR !!! Please enter the correct file destination. ");
    fileSaveDes.Focus();
    return;
}
```

- Proceed to check if the textbox fileSaveDes.Text is empty or not; if it is empty, display a messagebox notifying of the error and requesting the correct path to the empty html file that needs to store the downloaded data.

```
// dùng try catch để kiểm tra dữ liệu khi download file về
try
{
    wc.DownloadFile(textURL.Text, fileSaveDes.Text);

    //
    txtView.Text = OpenToRead(fileSaveDes.Text);
    MessageBox.Show("File downloaded successfully !!! ");
}
// dùng WebException: ngoại lệ được thrown đi khi một lỗi xảy ra trong khi acccessing mạng thông qua một giao thức có thể cắm.
catch (WebException wex)
{
    // Nhận được một thông báo mô tả ngoại lệ hiện tại.
    txtView.Text = wex.Message;
    // .Message để thông báo lỗi, thông báo giúp giải thích nguyên nhân của ngoại lệ hoặc để một chuỗi trống
}
```

- Use the DownloadFile property of WebClient to get the content of the webpage to save into the file specified above.
- Reuse the OpenToRead() method to read the page content into a stream. In this case, the stream is the textbox txtView.Text.
- Use try – catch to check the data when downloading the file. If the file is downloaded successfully, a messagebox will display that the file has been downloaded successfully.

+ For the Get file button:

```
private void btnGetFileDes_Click_1(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.ShowDialog();
    fileSaveDes.Text = ofd.FileName;
    FileStream fs = new FileStream(ofd.FileName, FileMode.OpenOrCreate);
    StreamReader sr = new StreamReader(fs);
    string s = sr.ReadToEnd();
    txtView.Text = s;
    fs.Close();
}
```

- Open the OpenDialog to browse the folder needed. The FileStream class is used to read data from a file, and the StreamReader class functions to read the file.
- Use the ReadToEnd function to read all the data and push it into the textbox txtView.Text

+ For the Show button (the show button plays a role in displaying the content of the .html file)

```
// button show sẽ hiển thị nội dung của file .html trên textbox txtView
1 reference
private void btnShow_Click(object sender, EventArgs e)
{
    if (fileSaveDes.Text.Trim() == "")
    {
        MessageBox.Show("ERRROR !!! Please enter the correct file destination. ");
        fileSaveDes.Focus();
        return;
    }
```

- Check if the .html file is empty, if the .html file is in the correct format or not; if not, display a messagebox that the correct format of the html file must be entered.

```
// khai báo FileStream để đọc và viết dữ liệu vào file
FileStream fs = new FileStream(fileSaveDes.Text, FileMode.Open);

//  Note: Byte arrays can represent any values
//  but each invidual byte can only hold a certain range
byte[] bit = new Byte[fs.Length];
fs.Read(bit, 0, (int)fs.Length);
fs.Close();

string s = Encoding.UTF8.GetString(bit);
txtView.Text = s;
}
```

- Declare FileStream to read and write data into the file.
- Initialize a bit variable used to save the value in bytes.
- Initialize the string s, use Encode.UTF8.GetString to overwrite in a derived class, decoding a byte sequence which is the bit initialized above into a string. Assign the string s value to the textbox txtView.Text to display the result on the screen.

+ For the Delete button:

```
1 reference
private void btnDelete_Click(object sender, EventArgs e)
{
    textURL.Text = fileSaveDes.Text = txtView.Text = null;
}
```

- Functions to delete all entered and displayed data so that new data can be entered.

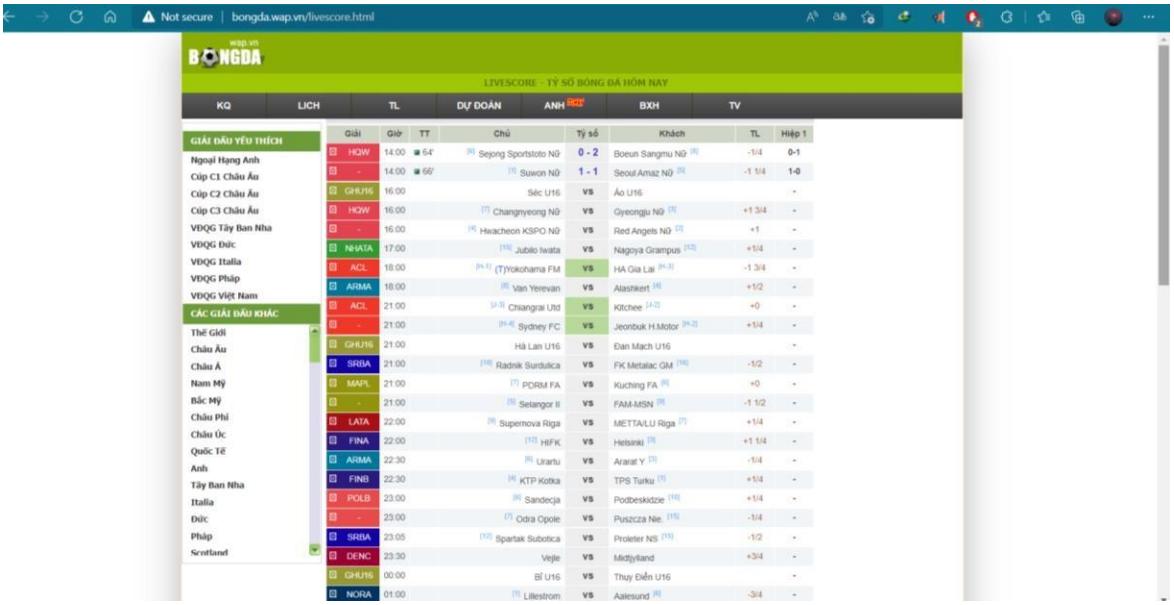
+ For the Exit button:

```
1 reference
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
```

- Used to close the form and exit the program.

– Step 4: Proceed with the demo

+ The website used to retrieve data ( http://bongda.wap.vn/livescore.htmll )




+ After entering the path of the .html file into the textbox textURL.Text and storing the data from the webclient:


- After the file is downloaded successfully, a MessageBox will display "File downloaded successfully", meaning the file has been downloaded successfully. The html source code of the web downloaded will display on the textbox txtView.Text.
- The downloaded file will save the web's path as html in the pre-created empty .html file in the folder, click on that .html file to navigate to the web page that has downloaded the html source code.

## 4. Scenario 04:

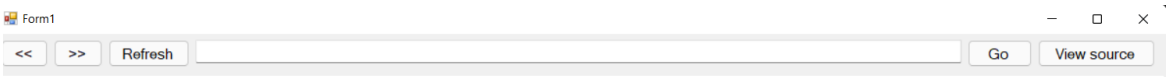
- Resource: Attached compressed File

- Objective: Write a program that operates like a basic Web Browser allowing the implementation of the following features:
  - View Website Content
  - Download HTML File
  - View Source

- Steps to implement:

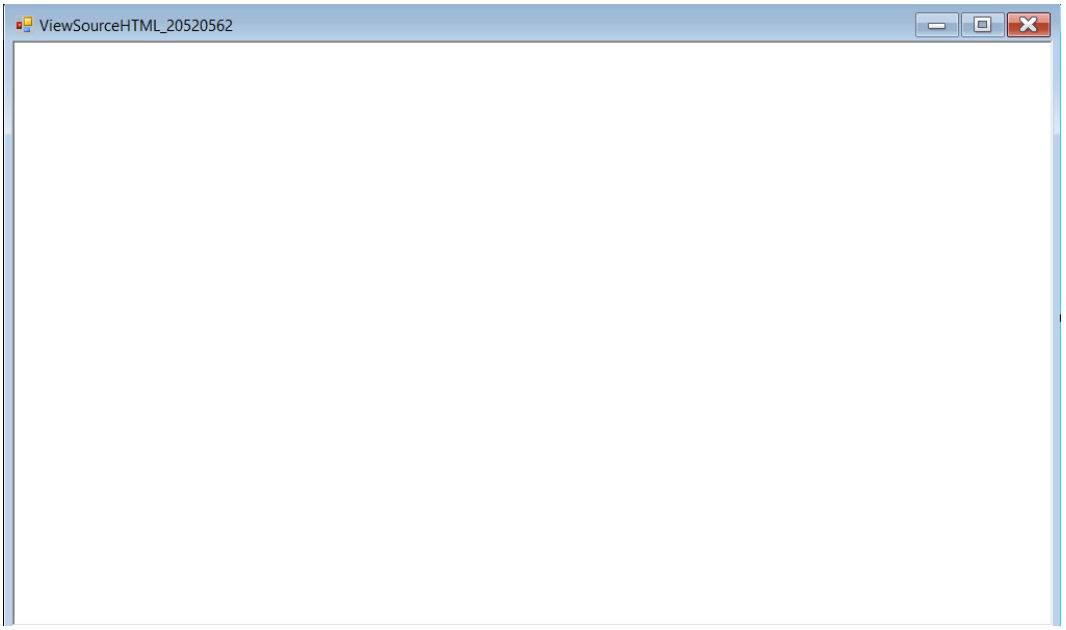
– Step 1: Design the interface

+ For the main form: the form will include 2 tableLayoutPanels to contain the buttons and textboxes needed, 1 webbrowser to display the content of the Website, 1 textbox txt_webUrl to enter the web address, 5 buttons: btnBack,




btnNext, btnRefresh, btnGo, and btnViewSource each with their own distinct function..

+ For the ViewSource form: the form is designed to be minimalistic, only including 1 richTextBox rc_ViewSource to display the html source code of the web browser that was entered from the main form.




– Bước 2: Begin coding:

+ For the btnBack button :

```
1 reference
private void btnBack_Click(object sender, EventArgs e)
{
    webbrowser.GoBack();
}

// Disables the Back button at the beginning of the navigation history.
0 references
private void webbrowser_CanGoBackChanged(object sender, EventArgs e)
{
    btnBack.Enabled = webbrowser.CanGoBack;
}
```

- The .GoBack() function allows the WebBrowser control to navigate to the previous page in the navigation history, if available.
- Assign the value of btnBack.Enabled to the webbrowser's CanGoBack function. Use the CanGoBack property to determine if the navigation history is available and contains a previous page. Handle the CanGoBackChanged event to be notified when the value of the CanGoBack property changes.

+ For the btnNext button:

```
1 reference
private void btnNext_Click(object sender, EventArgs e)
{
    webbrowser.GoForward();
}
// Disables the Forward button at the end of navigation history.
0 references
private void webbrowser_CanGoForwardChanged(object sender, EventArgs e)
{
    btnNext.Enabled = webbrowser.CanGoForward;
}
```

- The .GoForward() function is used to navigate the WebBrowser control to the next page in the navigation history, if available.
- Assign the value of btnNext.Enabled to the webbrowser's CanGoForward function. Use the CanGoForward property to determine if the navigation history is available and contains a page following the current page. Handle the CanGoForwardChanged event to be notified when the value of the CanGoForward property changes.

+ For the btnRefresh button:

```
1 reference
private void btnRefresh_Click(object sender, EventArgs e)
{
    webbrowser.Refresh();
}
```

- The .Refresh() function reloads the document displayed in the webbrowser, or reloads the web page currently displayed on the form

+ For the btnGo button:

```
public string resource = string.Empty;
```

- Declare a string variable resource and set its value to string.Empty, meaning the string variable is understood as an empty string.

```
1 reference
private void btnGo_Click(object sender, EventArgs e)
{
    if (txt_webUrl.Text != string.Empty)
    {
        webbrowser.Navigate(txt_webUrl.Text.Trim());
        WebClient wc = new WebClient();
        resource = wc.DownloadString(txt_webUrl.Text.Trim());

    }
    else
        MessageBox.Show("Error !, Please input URL !!!");
}
```

- Use the webbrowser's .Navigate function for the textbox txt_webUrl.Text to load the document at the specified location into the webBrowser.
- Initialize a WebClient and set the value of resource to wc.DownloadString for textbox txt_webUrl.Text. The .DownloadString function is used to download the requested resource as a string. The resource to download is specified as a string containing the URL. In this case, the resource to download is entered from textbox txt_webUrl.Text.
- Use an if-else to check whether a URL has been entered or not, if not then display a MessageBox notifying of the error and requesting the URL to be entered.

+ For the btnViewSource button:

```
1 reference
private void btnViewSource_Click(object sender, EventArgs e)
{
    ViewSource source = new ViewSource(this);
    source.Show();
}
```

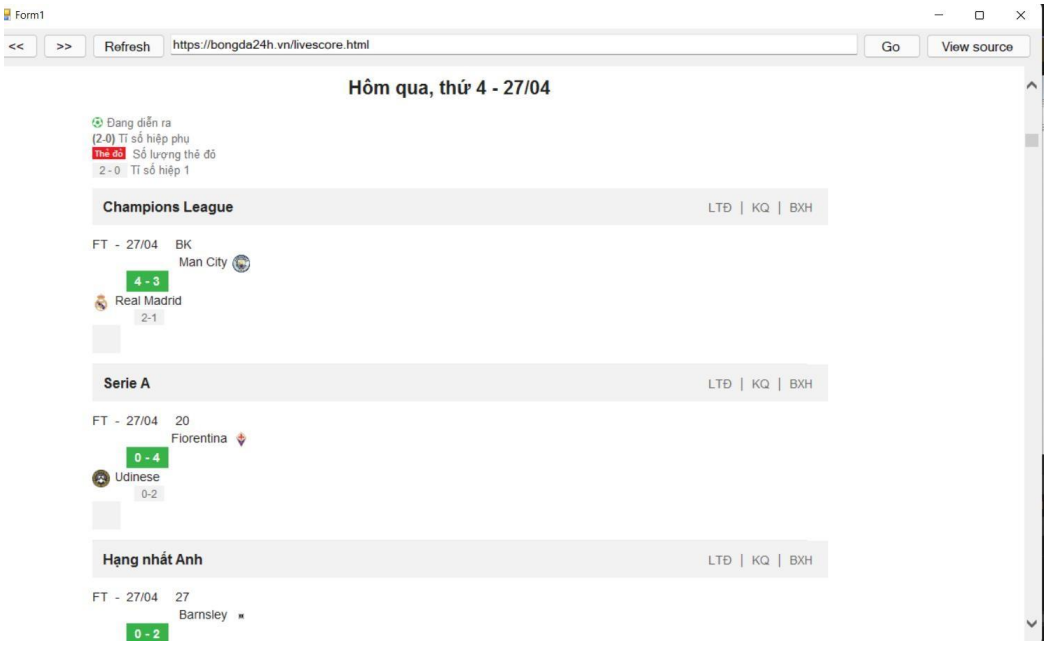
- Declare the name of the form used to view the html source code after downloading, use the pointer this for easier form navigation.
- Use the .Show() function for the value source assigned to the form ViewSource to display the form ViewSource.

+ For the ViewSourceHTML form:

```
4 references
public partial class ViewSource : Form
{
    Bai4 bai4;
    1 reference
    public ViewSource(Bai4 sourceForm)
    {
        InitializeComponent();
        this.bai4 = sourceForm;
        rc_ViewSource.Text = bai4.resource;
    }
}
```

- Declare the name of the main form Bai4 and name the variable for the form as bai4
- Initialize the ViewSource form with the parameter value being Bai4 sourceForm
- Use the pointer this to point to the main form and set it equal to the Bai4's sourceForm value, then set the value for richTextBox rc_ViewSource.Text to the value of the bai4.resource of the Main Form. Various methods can be used to parse data from this form to another such as delegates, constructors but this method is the most optimal and simple.

– Step 4: Proceed with the Demo




+ Enter the Url to the desired website (in this Demo case it's a football score website with the path as in the picture) and press the btnGo button to start loading the website's document. And the webpage will be displayed on the webbrowser of the main form as shown.

+ We can customise pressing the buttons btnBack, btnForward, btnRefresh to perform actions like going back to the previous page, going to the next page, or reloading the webpage.


+ When pressing the ViewSource button, it will redirect to the ViewSource form to display the html sourcecode of that website, use the mouse to click and drag down to view the entire source code of the webpage.

END