Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator nr. 3 la Programarea în Rețea

Tema: Comunicarea în Web: protocol și aplicație client HTTP

Efectuat st. gr. TI-142:

Chicu Roman.

Verificat lect. asistent.:

Ostapenco Stepan.

Scopul lucrării:

Înțelegerea protocolului HTTP și rolul acestuia în comunicarea în Web, studiul componentelor C# utile în aplicarea protocolului HTTP.

Obiectiv:

Obectivul specific lucrări constînd în crearea unui aplicații client C# care ar interacțiuna cu serverul Web prin intermediul metodelor HTTP studiate.

Link la repozitoriu: https://github.com/logan11116/lab3

Secure Hyper Text Transfer Protocol (sau HyperText Transfer Protocol/Secure, abreviat HTTPS) reprezintă protocolul HTTP încapsulat într-un flux SSL/TLS care criptează datele transmise de la un browser web la un server web, cu scopul de a se oferi o identificare criptată și sigură la server. Conexiunile HTTPS sunt folosite în mare parte pentru efectuarea de operațiuni de plată pe World Wide Web și pentru operațiunile "sensibile" din sistemele de informații corporative. HTTPS nu trebuie confundat cu Secure HTTP (S-HTTP) specificat în RFC 2660

Metodele sînt de fapt operatiile care pot fi aplicate obiectelor constituite de resursele din retea, în acceptiunea protocolului HTTP. Metoda va trebui sa fie totdeauna primul element dintr-o linie de cerere. Metodele prevazute în versiunea 1.1 sînt urmatoarele: OPTIONS, GET, HEAO, POST, PUT, PATCH, COPY, MOVE, DELETE, LINK, UNLINK, TRACE, WRAPPED.

OPTIONS semnifica o cerere relativa la informatiile ce definesc optiunile de comunicare disponibile pe conexiunea catre URI-ul specificat în cerere. Metoda permite determinarea optiunilor si/sau posibilitatilor unui server, fara sa determine o actiune din partea resursei adresate.
si metoda are nevoie de parametri, nu numai resursa, iar în HTTP termenul consacrat pentru parametrii metodelor este "***header field***" sau "***antet de cîmp***". Definite în cadrul protocolului pentru fiecare metoda, antetele de cîmp pot avea valori care la rîndul lor sînt definite (dar nu limitate, extensiile fiind în principiu totdeauna posibile).

Exemplu:

O cerere de tipul

OPTIONS www.xxx.ro HTTP1/1 CRLF Accept: audio/*; q=0.2, audio/basic CRLF

reprezinta o cerere de definire a optiunilor catre serverul www.xxx.ro, în care clientul solicitant spune ca prefera audio/basic, dar accepta orice tip pentru date audio în cazul în care calitatea reprezentarii nu scade sub 20%.

Indentificatorul uniform al resursei , abreviat URI- Uniform Resource Identifier, oferă protocolui posibilitatea de a localiza resursa pe orice calculator al rețelei globale.În același timp este important de tratat separate indentificatoarele resursei și conținutul informational al acesteia, care paote varia. Identificatorul resursei poate lua forma URL (UNIFORM Resource Locator) sau URN (Uniform Resource Name). Structura simplificativă a unui URL este prezentată în figura 1.

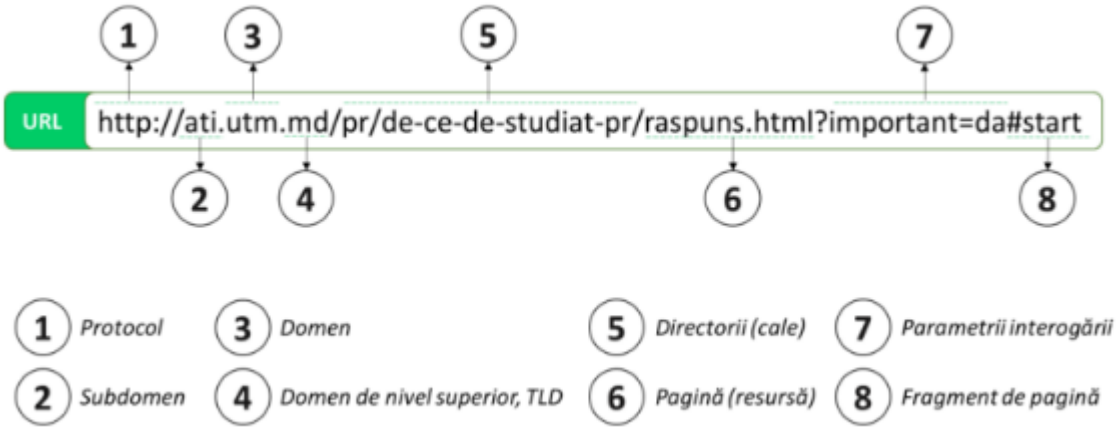

Figura 1- Structura URL

În conformitate cu modelul de intracțiune descris de protocol o conexiune este inițiată de agentul utilizatorului, care va genera și trimite cererea. Serverul HTTP primește crerea, identifică resursa și împachetînd în răspuns conținutul informațional (document hipertext, imagine, etc.) îl trimite spre client (figura 2).

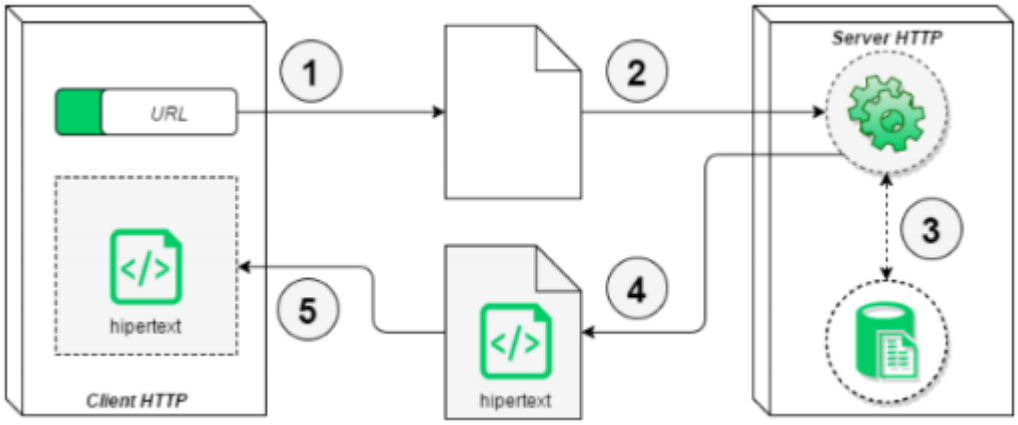

Figura 2 – Interacțiunea HTTP cerere-răspuns

O cerere HTTP cuprinde trei elemente esențiale (figura 3) linia de interogare, grupul de antete și corpul mesajului , care în cazul unor cereri poate lipsi. Linia de inerogare definește metoda de interacțiune, resursa țintă și versiunea protocolui, conform căruia se va desfășura schimbul de mesaje. În cazul răspunsului HTTP , mesajuk este compus din linia de stare, antetele și corpul mesajului care conține resursa cerută.

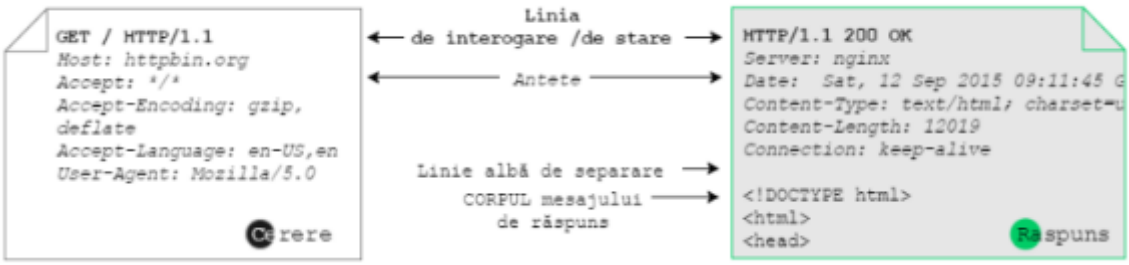

Figura 3. Mesaje de cerere și răspuns HTTP

Metodele HTTP definesc operațiile de efectuat la server asupra resursei la primirea cereii. Acestea fiind gîndite într-un context mai larg decît “cererea ;I primireaunui pagine Web” au permis în tim interacțiuni mai complexe, ce stau la baza unor sisteme Web distribuite. Numele metodei este indicate în prima linie de interogare și este dependent de registru. În table 4. Sunt przentate unele metode propuse de protocol.

Metodă	*Descriere*
GET	Cere o resursă, care va fi împachetată în răspuns în conformitate cu convenția MIME
HEAD	Cere doar antetul resursei indicate
PUT	Transmite spre server o resursă în vederea stocării acesteia
PATCH	Modifică parțial resursa, datele fiind transmise în corpul cererii
POST	Transmite spre server date în vederea atașării acestora resursei indicate
DELETE	Cere ştergerea resursei de la server
OPTIONS	Obține informații despre server (de exemplu despre metodele pe care le suportă)
TRACE	Cerere de tip *loop-back*, utilă în caz dacă se dorește a vedea ce date ajung la server de la un client

Figura 4. Metode al protocolului HTTP

Antetele HTTP este mijlocul prin care se definesc detaliile interacțiunii: informațiile despre clinet și serverele implicate , condiționări de formate, volumul și tipul datelor transmise , diverse meta-informații despre conținut și interacțiunea propriu-zisă. Antetele, zise și cîmpuri HTTP, pot specifice cererii sau răspunsului, dar comune ambelor tipuri de mesaje (figura 5).

Antet	**Tip**	**Descriere**
User-Agent	Cerere	Informație despre client şi platformă
Accept	Cerere	Tipul resursei acceptate, tip MIME
Host	Cerere	Numele DNS al serverului
Date	Ambele	Data generării mesajului
Cookie	Cerere	Trimite valoarea *cookie* înapoi serverului
Referer	Cerere	Pagina care s-a referit spre resursa cerută
Upgrade	Ambele	Protocolul spre care se dorește trecerea
Content-Type	Ambele	Tipul conținutului, tip MIME
Content-Length	Ambele	Mărirea datelor transmise, *bytes*
Server	Răspuns	Informație despre server
Last-Modified	Răspuns	Data ultimei modificări a resursei
Set-Cookie	Răspuns	Valoarea *cookie* după care serverul poate identifica un client într-o ulterioară cerere
Cache-Control	Ambele	Cere conținut localizat pe serverul original
Connection	Ambele	Cere păstrarea conexiunii

Figura 5 – Antete ale protocolui HTTP

Mersul lucrării:

Scopul lucrării de laborator presupune înțelegerea protocolului și aplicarea acestuia. Astfel sarcinile tip variază între un client simplu Http la aplicații-utilitare de colectare a informației în Web:

a) Client simplu Http apt să trimită și să primească răspunsuri la cereri de tip Get, Head și Post;

b) Aplicație care validează hiperlăgăturile prezente pe pagina resursei specificate;

c) Aplicație crawler de parcurgere conform unui algoritm a unui șir de resurse Web;

d) Aplicație de colectare a unor informații specifice pe diverse resurse Web (tip date, cuvinte cheie, etc.);

Metoda GET. Cea mai răspândită metodă a protocolului, este invocată prin comanda **curl httpbin.org/ip**, care va obține și va afișa în întregime conținutul resursei, adresa URL a căreia este indicată ca parametru. În mod obișnuit cURL afișează doar conținutul, deși primește de la server și antetele HTTP. Pentru a vizualiza antetele la comanda curl se va adăuga **--inlcude** (sau **–i).** Prin urmare pentru **curl -i httpbin.org/ip** se obține și se afișează următorul răspuns al serviciului:
Metoda Get cu aplicație în C# utilizată sub forma următaore (figura 6)

```
if (!textBox1.Text.StartsWith("http://") &&
!textBox1.Text.StartsWith("ftp:/") &&
!textBox1.Text.StartsWith("https://")) textBox1.Text = "http://" + textBox1.Text;

if (comboBox1.SelectedIndex == 0)
{
    try
    {
        WebRequest request = WebRequest.Create(textBox1.Text);
        request.Credentials = CredentialCache.DefaultCredentials;
        WebResponse response = request.GetResponse();
        richTextBox1.AppendText("Status " + ((HttpWebResponse)response).StatusDescription + "\n\n");
        Stream dataStream = response.GetResponseStream();
        StreamReader reader = new StreamReader(dataStream);
        richTextBox1.AppendText(reader.ReadToEnd());
        reader.Close();
        response.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

Figura 6 – Metoda Get in C#

b) Metoda POST. Trimiterea datelor spre server prin metoda POST se realizează prin **curl –data "a=1&b=2" httpbin.org/post.** Variabilele a și b pot corespunde într-un formular HTML unor câmpuri, valorile cărora sunt determinate de utilizator. Răspunsul serviciului pentru cererea formulată conține în format JSON datele transmise de tip **application/x-www-form- urlencoded**, precum și antetele cererii HTTP:

```
postData = textBox2.Text;
WebRequest request = WebRequest.Create(postUrl);
request.Method = "POST";
byte[] byteArray = Encoding.UTF8.GetBytes(postData);
request.ContentType = "application/x-www-form-urlencoded"; //multipart/form-data
request.ContentLength = byteArray.Length;
Stream dataStream = request.GetRequestStream();
dataStream.Write(byteArray, 0, byteArray.Length);
dataStream.Close();
WebResponse response = request.GetResponse();
richTextBox1.AppendText("Status " + ((HttpWebResponse)response).StatusDescription + "\n\n");
dataStream = response.GetResponseStream();
StreamReader reader = new StreamReader(dataStream);
richTextBox1.AppendText(reader.ReadToEnd());
reader.Close();
dataStream.Close();
response.Close();
```

Figura 7 – Metoda Post în protocolul HTTP

c) **Metoda HEAD.** În cazul cînd se dorește obținerea doar a cîmpurilor HTTP opțiunea –head (sau –I) impune curl să trimită cererea de tip Head. Astfel cererea curl –I httpbin.org/ip obține doar rîndurile 1-8 precedentului răspuns.

```
WebRequest request = WebRequest.Create(textBox1.Text);
request.Credentials = CredentialCache.DefaultCredentials;
request.Method = "HEAD";
WebResponse response = request.GetResponse();
richTextBox1.AppendText("Status " + ((HttpWebResponse)response).StatusDescription + "\n\n");
richTextBox1.AppendText(response.Headers.ToString());
response.Close();
```

Figura 8 – Metoda Head în protocolul HTTP

Mesajele de cerere și de răspuns ale protocolului HTTP reprezintă de fapt linii de text în format ASCII. Astfel interacțiunile client-server conforme protocolului pot fi ușor urmărite inclusiv prin utilitare de linie de comandă. Printre aplicațiile de acest tip se menționează cURL, care permite operațiuni variate conform schemelor URL. În vederea studiului protocolului în cadrul lucrărilor de laborator se va utiliza serverul httpbin.org. Acest serviciu HTTP acceptă cereri de test ale majorității metodelor de interacțiune ale protocolului. În cazul cererilor în care se transmit date, serviciul va întoarce datele formatate JSON și împachetate în răspuns. Testarea metodelor se va realiza în condițiile descărcării prealabile a utilitarului. Astfel am reazlizat o aplicație ce olosesște respectiv metodele Get, Head, Post (figura 9-11). Deasemenea căutarea expresilor regulate in masiv (figura 11,12).

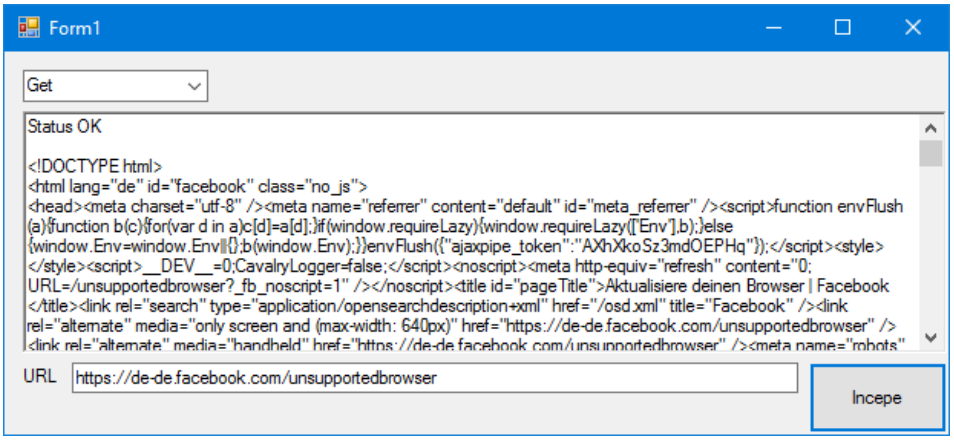

Figura 9 – Metoda Get

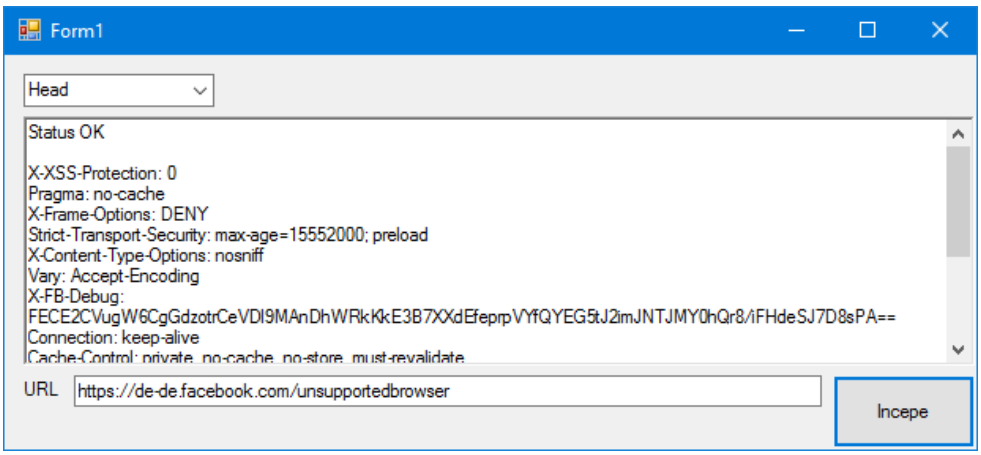

Figura 10. Metoda Head

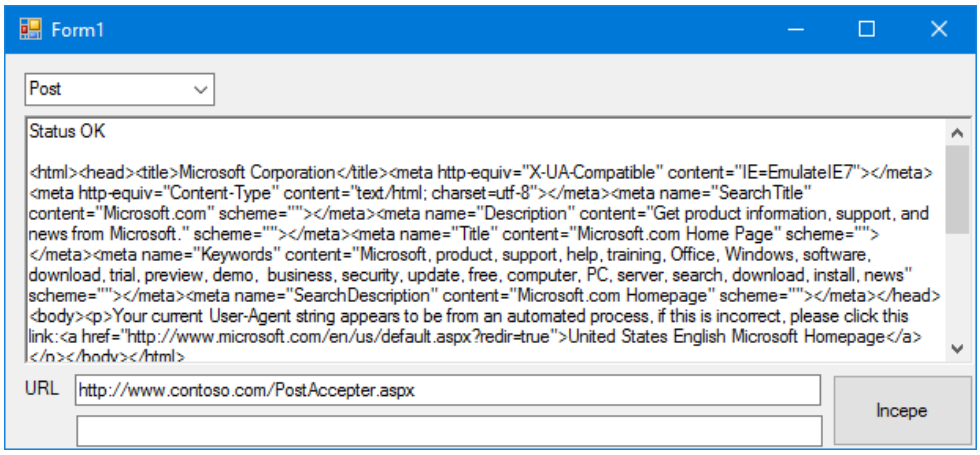

Figura 11. Metoda Post

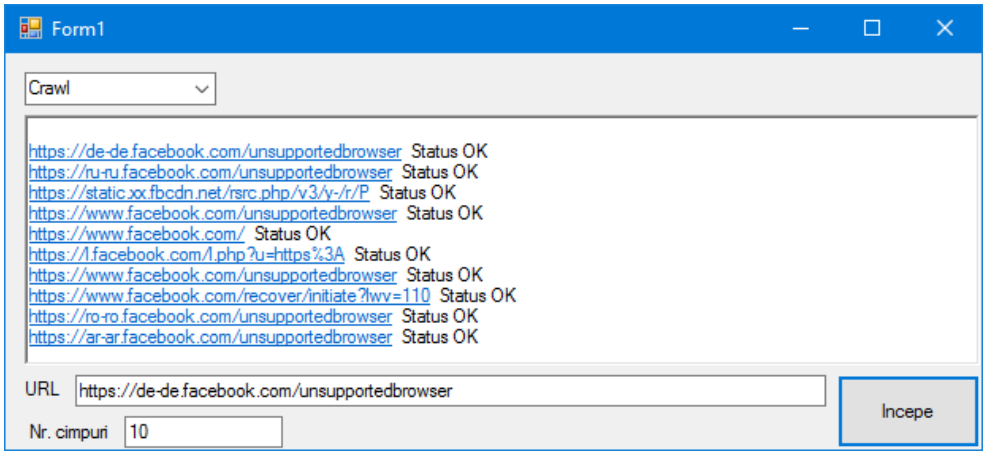

Figura 12. Expresii regulate

Concluzie

În urma acestei lucrări de laborator au fost obținute abilități de lucru cu protocolul HTTP in mediul C#. Au fost verificate mai multe metode get, head, post pe care le pune la dipoziție protocolul http. Deasemenea au fost utilizate metodele regulate pentru a găsi cuvintele cheie dintr-un masiv de date.

Bibliografie

1. https://drive.google.com/file/d/0B0vf11XUnLc2Q0NrLTk3czlOTWM/view
2. https://msdn.microsoft.com/en-us/library/btky721f.aspx
3. https://moodle.ati.utm.md/course/view.php?id=90

Anexă

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.IO;
using System.Text.RegularExpressions;

namespace lab3
{
    public partial class Form1 : Form

    {
        string Url = "http://facebook.com";
        string postUrl = "http://www.contoso.com/PostAccepter.aspx ";
        string postData = "This is a test that posts this string to a Web server.";


        public Form1()
        {
            InitializeComponent();
            comboBox1.SelectedIndex = 1;
            textBox1.Text = "http://facebook.md/";
          //  textBox3.Visible = false;
            textBox3.Text = "10";
          //  label3.Visible = false;
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            richTextBox1.Text = "";
            if (comboBox1.SelectedIndex == 2)
            {
                Url = textBox1.Text;
                textBox1.Text = postUrl;
                textBox2.Text = postData;
                label1.Visible = true;
                textBox2.Visible = true;
                textBox3.Visible = false;
                label3.Visible = false;
            }
            else if (comboBox1.SelectedIndex == 0)
            {

                    textBox1.Text = Url;
                    label2.Visible = false;
                    textBox2.Visible = false;
                    textBox3.Visible = false;
                    label3.Visible = false;

            }
            else if (comboBox1.SelectedIndex == 4)
            {

                    textBox1.Text = Url;
```

```
                textBox3.Visible = true;
                label3.Visible = true;
            label2.Visible = false;
            textBox2.Visible = false;


        }
        else if (comboBox1.SelectedIndex == 3)
        {

            textBox1.Text = Url;
            textBox3.Visible = false;
            label3.Visible = false;
            textBox2.Visible = false;
            label2.Visible = false;

        }
        else if (comboBox1.SelectedIndex == 1)
        {

            textBox1.Text = Url;
            textBox3.Visible = false;
            label3.Visible = false;
            textBox2.Visible = false;
            label2.Visible = false;

        }

    }

    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar)) e.Handled = true;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (!textBox1.Text.StartsWith("http://") &&
!textBox1.Text.StartsWith("ftp:/") && !textBox1.Text.StartsWith("https://"))
textBox1.Text = "http://" + textBox1.Text;

        if (comboBox1.SelectedIndex == 0)
        {
            try
            {
                WebRequest request = WebRequest.Create(textBox1.Text);
                request.Credentials = CredentialCache.DefaultCredentials;
                WebResponse response = request.GetResponse();
                richTextBox1.AppendText("Status " +
((HttpWebResponse)response).StatusDescription + "\n\n");
                Stream dataStream = response.GetResponseStream();
                StreamReader reader = new StreamReader(dataStream);
                richTextBox1.AppendText(reader.ReadToEnd());
                reader.Close();
                response.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }

        }
        else if (comboBox1.SelectedIndex == 1)
        {
            try
```

```
                {
                    WebRequest request = WebRequest.Create(textBox1.Text);
                    request.Credentials = CredentialCache.DefaultCredentials;
                    request.Method = "HEAD";
                    WebResponse response = request.GetResponse();
                    richTextBox1.AppendText("Status " +
((HttpWebResponse)response).StatusDescription + "\n\n");
                    richTextBox1.AppendText(response.Headers.ToString());
                    response.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: " + ex.Message);
                }

            }
            else if (comboBox1.SelectedIndex == 2)
            {
                try
                {
                    postData = textBox2.Text;
                    WebRequest request = WebRequest.Create(postUrl);
                    request.Method = "POST";
                    byte[] byteArray = Encoding.UTF8.GetBytes(postData);
                    request.ContentType = "application/x-www-form-urlencoded";
//multipart/form-data
                    request.ContentLength = byteArray.Length;
                    Stream dataStream = request.GetRequestStream();
                    dataStream.Write(byteArray, 0, byteArray.Length);
                    dataStream.Close();
                    WebResponse response = request.GetResponse();
                    richTextBox1.AppendText("Status " +
((HttpWebResponse)response).StatusDescription + "\n\n");
                    dataStream = response.GetResponseStream();
                    StreamReader reader = new StreamReader(dataStream);
                    richTextBox1.AppendText(reader.ReadToEnd());
                    reader.Close();
                    dataStream.Close();
                    response.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: " + ex.Message);
                }
            }

            else if (comboBox1.SelectedIndex == 3)
            {
                try
                {
                    if (richTextBox1.Text != "") richTextBox1.Text = "";
                    if (!textBox1.Text.StartsWith("http://") &&
!textBox1.Text.StartsWith("ftp:/") && !textBox1.Text.StartsWith("https://"))
textBox1.Text = "http://" + textBox1.Text;
                    if (label2.Text != "Result:") label2.Text = "Result:";
                    WebRequest request = WebRequest.Create(textBox1.Text);
                    WebResponse response = request.GetResponse();
                    Stream stream = response.GetResponseStream();
                    StreamReader reader = new StreamReader(stream);
                    string status = ((HttpWebResponse)response).StatusDescription;
                    richTextBox1.Text = "Checking>.. " + textBox1.Text + " Status " +
status + "\n";

                    string pattern = @"(((http|https|ftp)+\:/\/)[&#95;a-z0-
9\/&#95;:@=.+?,##%&~-]*[^.|\'|""|\# |!|\(|?|,| |>|<|;|\)])";
```

```
                    MatchCollection numberoflinks = Regex.Matches(reader.ReadToEnd(),
pattern, RegexOptions.Singleline);
                    label2.Text = "Results found " + numberoflinks;

                    int counter = 0;
                    foreach (Match m in numberoflinks)
                    {
                        string link = m.Groups[1].Value;
                        try
                        {
                            WebRequest newrequest = WebRequest.Create(link);
                            WebResponse newresponse = newrequest.GetResponse();
                            richTextBox1.AppendText("\n" + link + "  Status " +
((HttpWebResponse)newresponse).StatusCode);
                            newresponse.Close();
                            counter++;
                            label2.Text = "Results found " + counter;
                        }
                        catch (Exception ex)
                        {
                            richTextBox1.AppendText("\n" + link + " " + ex.Message);
                            counter++;
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error: " + ex.Message);
                }


            }

            else if (comboBox1.SelectedIndex == 4)
            {
              //  textBox3.Visible = true;
               // label3.Visible = true;
                if (!textBox1.Text.StartsWith("http://") &&
!textBox1.Text.StartsWith("ftp://") && !textBox1.Text.StartsWith("https://"))
textBox1.Text = "http://" + textBox1.Text;
                for (int i = 0; i < Convert.ToInt32(textBox3.Text); i++)
                {
                    WebRequest newRequest = WebRequest.Create(Url);
                    WebResponse newResponse = newRequest.GetResponse();
                    Stream newStream = newResponse.GetResponseStream();
                    StreamReader newReader = new StreamReader(newStream);
                    richTextBox1.AppendText("\n" + Url + "  Status " +
((HttpWebResponse)newResponse).StatusCode);

                    string pattern = @"(((http|https|ftp)+\:/\/)[&#95;a-z0-
9\/&#95;:@=.+?,##%&~-]*[^.|\'|""|\# |!|\(|?|,| |>|<|;|\)])";
                    MatchCollection links = Regex.Matches(newReader.ReadToEnd(), pattern,
RegexOptions.Singleline);

                index:
                    Random rand = new Random();
                    Url = links[rand.Next(links.Count)].Value;
                    if (((HttpWebResponse)newResponse).StatusCode.ToString() != "OK")
goto index;

                    newResponse.Close();
                    newReader.Close();
```

```
                }
            }
        }



        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```