**TextParsing Example**

// Course Material "Software Service Engineering"

// (c) 2013 by Distributed and Self-organizing Systems Group, TUC

using System;

using System.Text.RegularExpressions;

namespace Vsr.Teaching.SSE.Sample

{

/// <summary>

/// A sample program that demonstrates the parsing of text.

/// </summary>

class TextParsingExample

{

/// <summary>

/// The main program routine.

/// </summary>

static void Main(string[] args)

{

string myText = "This is a sample sentence.";

// working with character positions

int pos = myText.IndexOf(" a ");

Console.WriteLine(myText.Substring(0, pos));

Console.WriteLine(Environment.NewLine);

// splitting text at certain characters

string[] words = myText.Split(' ');

foreach (string word in words)

{

Console.WriteLine(word);

}

Console.WriteLine(Environment.NewLine);

// matching with regular expressions

Match match = Regex.Match(myText, @"^(?<a>[^s]\*s)(?<b>[^p]\*p)(?<c>.\*)$");

Console.WriteLine(match.Groups["a"].Value); // This

Console.WriteLine(match.Groups["b"].Value); // is a samp

Console.WriteLine(match.Groups["c"].Value); // le sentence.

Console.WriteLine(Environment.NewLine);

// parsing numbers

myText = "41";

int myNumber = Int32.Parse(myText);

Console.WriteLine(myNumber + 1);

Console.WriteLine(Environment.NewLine);

// convert between numbers and hexadecimal strings

string hex = Convert.ToByte(42).ToString("x");

int number = Int32.Parse(hex, System.Globalization.NumberStyles.HexNumber);

Console.WriteLine("hex: {0}, dec: {1}", hex, number);

Console.WriteLine(Environment.NewLine);

Console.ReadLine();

}

}

}

**TASK 3**

**URL**

using System;

using System.Text.RegularExpressions;

namespace Task3

{

/// <summary>

/// A class for generating and parsing HTTP-URIs.

/// </summary>

public class Url

{

const string VALID\_CHARACTERS = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789$-\_.~";

public string Scheme = "";

public string Host = "";

public int Port = 80;

public string Path = "";

public string Query = "";

public string FragmentId = "";

/// <summary>

/// Constructor for parsing URLs.

/// </summary>

public Url(string urlStr)

{

Match match = Regex.Match(urlStr, @"

^(?:

(?<scheme>[^:]\*)

\:\/\/(?<host>[^:^/^?^#]\*)

(?:\:(?<port>\d\*))?

)?

(?<path>\/[^?^#]\*)?

(?:\?(?<query>[^#]\*))?

(?:\#(?<fragmentid>.\*))?$

", RegexOptions.IgnorePatternWhitespace);

if (match.Success)

{

this.Scheme = match.Groups["scheme"].Value.ToLower();

this.Host = Decode(match.Groups["host"].Value);

if (match.Groups["port"].Value != "")

this.Port = Convert.ToInt32(match.Groups["port"].Value);

this.Path = Decode(match.Groups["path"].Value);

this.Query = Decode(match.Groups["query"].Value);

this.FragmentId = Decode(match.Groups["fragmentid"].Value);

}

else

{

throw new FormatException("Could not parse URL: " + urlStr);

}

}

/// <summary>

/// Constructor for building URLs from their components.

/// </summary>

public Url(string scheme, string host, int port, string path, string query, string fragmentId)

{

this.Scheme = scheme;

this.Host = host;

this.Port = port;

this.Path = path;

this.Query = query;

this.FragmentId = fragmentId;

}

/// <summary>

/// Returns the string representation of the URL.

/// </summary>

public override string ToString()

{

string url = Scheme + "://" + Host;

if (Port != 80) url += ":" + Port;

if (Path != "") url += "/" + Encode(Path.Substring(1));

if (Query != "")

{

var queryParts = Query.Split('&');

var qEncoded = "";

foreach ( var queryPart in queryParts )

{

var nameValue = queryPart.Split( '=' );

qEncoded += "&" + Encode( nameValue[0] ) + "=" + Encode( nameValue[1] );

}

url += "?" + qEncoded.Substring( 1 );

}

if (FragmentId != "") url += "#" + Encode(FragmentId);

return url;

}

/// <summary>

/// Encodes any special characters in the URL with an escaping sequence.

/// </summary>

public static string Encode(string s)

{

string result = "";

for (int i = 0; i < s.Length; i++)

{

if (VALID\_CHARACTERS.Contains(s[i].ToString()))

{

// allowed character

result += s[i];

}

else

{

// character has to be encoded as "%" + HexDigit + HexDigit

result += "%" + Convert.ToByte(s[i]).ToString("X");

}

}

return result;

}

/// <summary>

/// Decodes any escaping sequence in the URL with the corresponding characters.

/// </summary>

public static string Decode(string s)

{

while (s.Contains("%"))

{

int pos = s.IndexOf("%");

byte b = byte.Parse(s.Substring(pos + 1, 2), System.Globalization.NumberStyles.HexNumber);

s = s.Substring(0, pos) + Convert.ToChar(b) + s.Substring(pos + 3);

}

return s;

}

}

}

**URL TEST**

using Xunit;

namespace Task3

{

public class UrlTests

{

[Fact]

public void TestUrls()

{

string s = "http://www.tu-chemnitz.de:8080/ein%20test?my-name=my-value&a=1#id";

Url url = new Url(s);

Assert.Equal("http", url.Scheme);

Assert.Equal("www.tu-chemnitz.de", url.Host);

Assert.Equal(8080, url.Port);

Assert.Equal("/ein test", url.Path);

Assert.Equal("my-name=my-value&a=1", url.Query);

Assert.Equal("id", url.FragmentId);

Assert.Equal(s, url.ToString());

}

}

}