using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using HtmlAgilityPack;

namespace Eyespy\_Core\_v1

{

/// <summary>

/// HTTPReader will be used to extract URL and metadata from a specific HTML dump.

/// </summary>

/// <remarks></remarks>

class HTTPReader

{

/// <summary>

/// Algorithm as pseudocoded within the Design document.

/// This function will extract the URL from the HTML data.

/// </summary>

/// <param name="data">The HTML dump from which to extract the URL. </param>

/// <returns>Returns the URL.</returns>

/// <remarks></remarks>

public string getURL(string data) // use the getBetween algorithm

{

return getBetween(data, "Host: ", "User-Agent:"); // the URL will be sandwiched between these 2 strings.

}

/// <summary>

/// Get the metadata-keywords from the provided URL.

/// </summary>

/// <param name="url">URL from which to extract the keywords.</param>

/// <returns>Returns the first 3 keywords if present, and NULL-string if not.</returns>

/// <remarks></remarks>

public string getMetadata(string url)

{

var webGet = new HtmlWeb(); // HTMLAgilityPack class used to create a virtual web browser within which we can authenticate with a website to get metadata...

string txtDesc = null; // buffer to hold the keywords

var document = webGet.Load(url); // load the whole HTML content of the URL provided as parameter to this variable

var metaTags = document.DocumentNode.SelectNodes("//meta"); // get all the meta variables

if (metaTags != null) // ensure there IS metadata present (not always the case...)

{

foreach (var tag in metaTags) // loop through each META-node until we find the one that yeild "keywords", then pass

// the content (assigned to attributes[].Value) into txtDesc

{

if (tag.Attributes["name"] != null && tag.Attributes["content"] != null && tag.Attributes["name"].Value == "keywords")

{

txtDesc = tag.Attributes["content"].Value; // keywords are yeilded to this variable in the following way: "keyword1, keyword2, keyword3, ..."

string[] keywords = txtDesc.Split(','); // 'explode' the string of keywords into a string array using the comma as a delimited (as illustrated above)

return (string)keywords[0] + ", " + keywords[1] + ", " + keywords[2]; // return the first 3 keywords.

}

}

}

else return ""; // if no meta tags are returned, return an empty string.

return ""; // to avoid the compiler from complaining that "not all paths have a return value" - though this return statement should never be reached.

}

/// <summary>

/// Used by the getURL function, this is the underlying algorithm which extracts data between two strings within a source text.

/// </summary>

/// <param name="strSource">The source text.</param>

/// <param name="strStart">The start string</param>

/// <param name="strEnd">The end string</param>

/// <returns>The data between the start and end string.</returns>

/// <remarks></remarks>

public string getBetween(string strSource, string strStart, string strEnd) // getBetween algorithm as pseudocoded within the Design document - credits: Unknown

{

int Start, End;

if (strSource.Contains(strStart) && strSource.Contains(strEnd)) // make sure the 'between' sources exist with the strig

{

Start = strSource.IndexOf(strStart, 0) + strStart.Length; // index of start position

End = strSource.IndexOf(strEnd, Start); // index of end position

return strSource.Substring(Start, End - Start); // return the resultant trucated string

}

else

{

return ""; // return null string if position is not found.

}

} }}