

## Forum:[Hack Forums](#)

Thread: [BrowserHistoryGatherer | Chrome, Firefox, IE, Safari](#)

**chrizator** - February 24, 2018, 9:15 am

I'm currently working on a project that needed access to the history entries of major browsers. I decided to make a lil' library and share it, since some people might be searching something similar.

Usage and installation is explained in the README.md file on github.

In order to demonstrate what you can do with it (in case it's not clear yet ? )

I made a lil' program that basically collects the history of specified browsers and displays them in a ListView.

Screenshot

Spoiler ([Click to View](#))

BrowserHistoryGatherer (library): <https://github.com/chrizator/BrowserHistoryGatherer>

BrowserHistoryViewer Source (lil' program):

<https://mega.nz/#!sVtAIDZa!9H9KiYdSJlIKzqoHPoNuF0HcP7LQnuBbUR-dyeK7aNm>

Have a good one! ?

**gigajew** - February 24, 2018, 6:57 pm

This is really nice, good job. I'm working with a ListView control right now and I'm getting some bad lag. Not quite as interesting as your project though.

**0xDEAD10CC** - February 24, 2018, 7:25 pm

At least the idea is a bit more unique.

- If your UrlHistoryWrapperClass class provides a Dispose() method why doesn't it implement IDisposable? You should review proper dispose pattern for this, especially since you're dealing with unmanaged stuff here.

- Your gatherer's are not truly an example of the singleton pattern either because you still allow for public instantiation of the object. That stuff could've been abstracted though if this is what you were going for.

- I would've marked the Browser enum with the FlagsAttribute as well so they could be used in various combinations in a single function call

Code [Flags]

public enum Browsers <-- edit: made this plural too

```
{
    None = 0,
    InternetExplorer = 1,
    Firefox = 2,
    Chrome = 4,
    Safari = 8
```

All = 15

}

Then you wouldn't have to code each individual combination and then manually concatenate the results for your ALL value, and you would also gain the ability to specify any 2 browser combinations too if they want to only exclude 1. I also don't think you would be supporting anywhere near at most 8 or 16 browsers here so this could definitely be an advantageous approach.

I also think this is kind of silly:

Code if (sortByDate)

historyList.Sort((x, y) => DateTime.Compare(y.LastVisitTime, x.LastVisitTime));

Reason: What if you wanted to sort the collection in other ways or choose asc/desc? That shouldn't be part of this method.

Also not sure why you do this:

Code IEnumerable<HistoryEntry> historyEntries = new List<HistoryEntry>();

Manipulate the data, then create a list from the results with ToList()? Why not just use it as a list and return the same?

I'd send you a pull request if I had some spare time.

(Feb 24, 2018 07:57 PM)gigajew Wrote: [\[CITE\]](#) This is really nice, good job. I'm working with a ListView control right now and I'm getting some bad lag. Not quite as interesting as your project though.

Let me guess you're not sorting with a ListViewItemSorter (if you support item sorting), and you're calling Add() to add large ranges of elements at a time constantly?

**gigajew** - February 24, 2018, 7:57 pm

Yeah I'm calling Add a bunch, I googled it real quick and found a source saying to inherit from listview and add double buffering and remove the unnecessary painting of the background that's built it and it works a lot smoother now

**0xDEAD10CC** - February 24, 2018, 8:14 pm

(Feb 24, 2018 08:57 PM)gigajew Wrote: [\[CITE\]](#) Yeah I'm calling Add a bunch, I googled it real quick and found a source saying to inherit from listview and add double buffering and remove the unnecessary painting of the background that's built it and it works a lot smoother now  
You can also AddRange() or BeginUpdate()/EndUpdate().

@OP: You're also using TryParse() wrong:

Code int outVal;

visitCount = int.TryParse(row["visit\_count"].ToString(), out outVal)

? (int?)outVal

: null;

HistoryEntry entry = new HistoryEntry(uri, title, lastVisit, visitCount, Browsers.Firefox);

visitCount is being assigned a boolean, and you're using that for the count and not using outVal.

There's a lot of things to fix here.

**cocaine\_monkey** - February 25, 2018, 3:02 am

(Feb 24, 2018 07:57 PM)gigajew Wrote: [\[CITE\]](#) This is really nice, good job. I'm working with a ListView control right now and I'm getting some bad lag. Not quite as interesting as your project though.

how its possible after those runpes etc you released your listview is getting Bad lag?

**Ninja** - February 25, 2018, 5:47 am

(Feb 24, 2018 07:57 PM)gigajew Wrote: [\[CITE\]](#) This is really nice, good job. I'm working with a ListView control right now and I'm getting some bad lag. Not quite as interesting as your project though.

You might want to enable double buffering.

(Feb 24, 2018 08:57 PM)gigajew Wrote: [\[CITE\]](#) Yeah I'm calling Add a bunch, I googled it real quick and found a source saying to inherit from listview and add double buffering and remove the unnecessary painting of the background that's built it and it works a lot smoother now

I suggest staying away from the ListView and using DataGridView. You can make it look like a ListView anyways, but it is much easier to use. I don't remember exactly why I switched, but the ListView had limitations that were bothering me.

Also, use DataBinding instead of calling Add/AddRange.

**chrizator** - February 25, 2018, 5:56 am

Quote:- I would've marked the Browser enum with the FlagsAttribute as well so they could be used in various combinations in a single function call

Then you wouldn't have to code each individual combination and then manually concatenate the results for your ALL value, and you would also gain the ability to specify any 2 browser combinations too if they want to only exclude 1. I also don't think you would be supporting anywhere near at most 8 or 16 browsers here so this could definitely be an advantageous approach.

That's interesting, haven't really dealt with attributes before. Going to have a lookt at it.

Quote:Also not sure why you do this:

Manipulate the data, then create a list from the results with ToList()? Why not just use it as a list and return the same?

I was doing it because LINQ's Concat method needs an IEnumerable. But yeah, there's probably a better approach.

I'm really thankful for your post, learned a lot out of it. It's cool to have more experienced users sacrificing their spare time to help others. ?

**0xDEAD10CC** - February 25, 2018, 9:33 pm

(Feb 25, 2018 06:56 AM)chrizator Wrote: [\[CITE\]](#) That's interesting, haven't really dealt with attributes before. Going to have a lookt at it.

I was doing it because LINQ's Concat method needs an IEnumerable. But yeah, there's probably a better approach.

I'm really thankful for your post, learned a lot out of it. It's cool to have more experienced users sacrificing their spare time to help others. ?

Why use LINQ to begin with though? You already have an instantiated List<T> object where you can just call AddRange() with new elements. Also:

Code List<T> -> IList<T> : ICollection<T>

ICollection<T> : IEnumerable<T>

Anything above that in the chain inherits IEnumerable<T> anyways implicitly.

```
Code public static IList<HistoryEntry> GetHistory(Browsers browsers, DateTime? startTime = null,
DateTime? endTime = null)
{
    var historyEntries = new List<HistoryEntry>();

    if ((browsers & Browsers.InternetExplorer) == Browsers.InternetExplorer)
        historyEntries.AddRange(IEGatherer.Instance.GetBrowserHistory(startTime, endTime));

    if ((browsers & Browsers.Firefox) == Browsers.Firefox)
        historyEntries.AddRange(FirefoxGatherer.Instance.GetBrowserHistory(startTime, endTime));

    if ((browsers & Browsers.Chrome) == Browsers.Chrome)
        historyEntries.AddRange(ChromeGatherer.Instance.GetBrowserHistory(startTime, endTime));

    if ((browsers & Browsers.Safari) == Browsers.Safari)
        historyEntries.AddRange(SafariGatherer.Instance.GetBrowserHistory(startTime, endTime));

    return historyEntries;
}
```

I know this could've used HasFlag() too but I don't like the way it boxes.

You say you've never dealt with attributes before but you have [Flags] elsewhere in your code which makes use of FlagsAttribute already. Attribute based programming is really useful however for some things, and more often than not if you're using a marker interface, it's a good indication that you can make a better design using a custom attribute instead.