Transform title into dashed URL-friendly string

Ask Question

19

2

I would like to write a C# method that would transform any title into a URL friendly string, similar to what stackoverflow does:

- replace spaces with dashes
- remove parenthesis
- etc.

I'm thinking of removing Reserved characters as per RFC 3986 standard (from Wikipedia) but I don't know if that would be enough? It would make links workable, but does anyone know what other characters are being replaced here at stackoverflow? I don't want to end up with %-s in my URLs...

Current implementation

```
string result = Regex.Replace(value.Trim(), @"[!*'""`();:@&+=$,/\\?%#\[\]<>«»{}_]");
return Regex.Replace(result.Trim(), @"[\s*[\-–—\s]\s*]", "-");
```

My questions

1. Which characters should I remove?
2. Should I limit the maximum length of resulting string?
3. Anyone know which rules are applied on titles here on SO?

A sub-question
Should I move this question to meta even though it's programming related?

c# replace

edited Jan 29 '10 at 13:01

asked Jan 29 '10 at 11:51

Robert Koritnik
77.8k 42 243 367

FWIW, don't think it's meta, you're just using SO as an example. – T.J. Crowder Jan 29 '10 at 11:53

7 Answers

38

Rather than looking for things to replace, the list of *unreserved chars is so short*, it'll make for a nice clear regex.

```
return Regex.Replace(value, @"[^A-Za-z0-9_\.~]+", "-");
```

(Note that I didn't include the dash in the list of allowed chars; that's so it gets gobbled up by the "1 or more" operator [`+`] so that multiple dashes (in the original or generated or a combination) are collapsed, as per Dominic Rodger's excellent point.)

You may also want to remove common words ("the", "an", "a", etc.), although doing so can slightly change the meaning of a sentence. Probably want to remove any trailing dashes and periods as well.

Also strongly recommend you do what SO and others do, and include a unique identifier *other* than the title, and then only use that unique ID when processing the URL. So `http://example.com/articles/1234567/is-the-pop-catholic` (note the missing 'e') and `http://example.com/articles/1234567/is-the-pope-catholic` resolve to the same resource.

edited Jan 29 '10 at 14:41

answered Jan 29 '10 at 11:57

T.J. Crowder
700k 124 1244
1341

1 The whitelist approach, however, does prevent Unicode characters (in IRI) from going through. – bobince Jan 29 '10 at 12:26

@Bobince: Exactly. I have to provide our language related characters as well (from eastern european charset) – Robert Koritnik Jan 29 '10 at 13:00

@Robert: IRI (RFC3987; ietf.org/rfc/rfc3987.txt) changes the game. If it's important, you might want to mention it in your question. It doesn't look like it's hard to add the supported IRI values to the whitelist. For strong look-alikes, you may want to pre-filter them. – T.J. Crowder Jan 29 '10 at 13:06

@TJCrowder: shouldn't . (dot) be escaped in your regular expression pattern? – Robert Koritnik Jan 29 '10 at 14:14

@Robert: I don't *think* so, but frankly I'm not sure and would have to check. It's harmless to do so and I've edited the answer accordingly. The usual meaning of dot (any character here) doesn't make any sense within a character class construct. You have to escape `` ` `` (obviously), `-` (since it creates a range within the construct), and `]` (which closes it), but I don't *think* you have to escape most others. – T.J. Crowder Jan 29 '10 at 14:45

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

Teams
Q&A for work

Learn More

0

This works for me

```
string output = Uri.UnescapeDataString(input);
```

edited Dec 28 '14 at 13:37

answered Sep 28 '14 at 15:42
user4012980

0

I use this one...

```
    public static string ToUrlFriendlyString(this string value)
    {
        value = (value ?? "").Trim().ToLower();

        var url = new StringBuilder();

        foreach (char ch in value)
        {
            switch (ch)
            {
                case ' ':
                    url.Append('-');
                    break;
                default:
                    url.Append(Regex.Replace(ch.ToString(), @"[^A-Za
9'()\*\\+_~\:\/\?\-\.,;=#\[\]@!$&]", ""));
                    break;
            }
        }

        return url.ToString();
    }
```

answered Apr 20 '13 at 21:35

Keith Beller
747 9 20

1

this is how I currently slug words.

```
        public static string Slug(this string value)
    {
        if (value.HasValue())
        {
            var builder = new StringBuilder();
            var slug = value.Trim().ToLowerInvariant();

            foreach (var c in slug)
            {
                switch (c)
                {
                    case ' ':
                        builder.Append("-");
                        break;
                    case '&':
                        builder.Append("and");
                        break;
                    default:

                        if ((c >= '0' && c <= '9') || (c >= 'a' && c
')
                        {
                            builder.Append(c);
                        }

                        break;
                }
            }

            return builder.ToString();
        }

        return string.Empty;
    }
```

answered Jan 29 '10 at 16:38

Mike Geise
740 6 14

Sorry but I rather use regular expressions. Your many lines of code could easily be replaced by two regular expressions at most. – Robert Koritnik Jan 30 '10 at 15:43

ya but at what cost will it be to you when that regular expression is very complicated and hard to understand. I would rather have maintainability then 2 cryptic regular expressions :) – Mike Geise Jan 30 '10 at 18:03

1

How about this:

```
string FriendlyURLTitle(string pTitle)
{
    pTitle = pTitle.Replace(" ", "-");
    pTitle = HttpUtility.UrlEncode(pTitle);
    return Regex.Replace(pTitle, "\%[0-9A-Fa-f]{2}", "");
}
```

answered Jan 29 '10 at 12:09

hannasm
960 10 22

2

I would be doing:

```
string url = title;
url = Regex.Replace(url, @"^\W+|\W+$", "");
url = Regex.Replace(url, @"'\"", "");
url = Regex.Replace(url, @"_", "-");
url = Regex.Replace(url, @"\W+", "-");
```

Basically what this is doing is it:

- strips non-word characters from the beginning and end of the title;
- removes single and double quotes (mainly to get rid of apostrophes in the middle of words);
- replaces underscores with hyphens (underscores are technically a word character along with digits and letters); and
- replaces all groups of non-word characters with a single hyphen.

answered Jan 29 '10 at 11:57

cletus
513k 141 844 915

I'm wondering why this didn't get more votes? Very simple to understand and well explained. I also wonder whether String.Replace might be quicker for steps 2 and 3 which are litteral character replacements, but that's here nor there really. I'm also interested in how the ^\W non-word character performs with unicode and non-latin languages? If it handles those even slightly well I'd say this is the best answer of the lot. Final suggestion - I do like the earlier suggestion of replacing '&' with 'and'. – Dave Amphlett Sep 8 '12 at 12:56

Oh - final thing, I think in the 3rd line (the second .Replace) the @"'\"" should actually be @"'""" – Dave Amphlett Sep 8 '12 at 13:01

1

Most "sluggifiers" (methods for converting to friendly-url type names) tend to do the following:

1. Strip everything except whitespace, dashes, underscores, and alphanumerics.

2. (Optional) Remove "common words" (the, a, an, of, et cetera).
3. Replace spaces and underscores with dashes.
4. (Optional) Convert to lowercase.

As far as I know, StackOverflow's sluggifier does #1, #3, and #4, but not #2.

answered Jan 29 '10 at 11:55

Amber
361k 61 535 491

So my replacements do the same ones: 1, 3 and 4. I just had to add underscore to the first reg ex pattern. – Robert Koritnik Jan 29 '10 at 13:01