

## Re: ToUpper() Better solution

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2007-03/msg00087.html>

---

- *From:* "Chris Mullins [MVP]" <[cmullins@xxxxxxxxxx](mailto:cmullins@xxxxxxxxxx)>
  - *Date:* Fri, 2 Mar 2007 14:26:57 -0800
- 

Opps. Definatly KD!

We want to do the decomposition & make our changes. For KC would decompose & then perform a canonical recompose – which would defeat the purpose!

I've never used (or even seen) the DecoderReplacementFallback – that's another good idea. By now the original poster has probably given up and will never try any of these solutions, but I think they would very cleanly do the trick.

—  
Chris Mullins, MCSD.NET, MCPD:Enterprise, Microsoft C# MVP  
<http://www.coversant.com/blogs/cmullins>

"JR" <[NoMail@xxxxxxxxxx](mailto:NoMail@xxxxxxxxxx)> wrote in message  
[news:%23%234RUTRXHHA.4252@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:%23%234RUTRXHHA.4252@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

You meant NormalizationForm.FormKD.

Looking into it, I see a simpler method: After normalization, use ASCIIEncoding with DecoderReplacementFallback replacing invalid ASCII characters (which will be the accents) with the empty string.

JR

"Chris Mullins" <[cmullins@xxxxxxxxxx](mailto:cmullins@xxxxxxxxxx)> ???  
?????:1172862185.347445.89000@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
I hadn't thought of that, but it's certainly an option.

Doing the normalization in .Net 2.0 is easy enough:

```
string s = "test";  
string normalized = s.Normalize(NormalizationForm.FormKC);
```

Then you can iterate over the normalized string looking for (and removing) the accents.

Re: ToUpper() Better solution

Chris Mullins

On Mar 1, 11:15 pm, "JR" <NoM...@xxxxxxxxxx> wrote:

In a more general way:

There is a Unicode database at

<http://www.unicode.org/Public/UNIDATA/>

You could do what you want in two steps: decompose the string to base characters followed by accent (NFKD normalization), then remove the accents.

JR

"Chris Mullins [MVP]" <cmull...@xxxxxxxxxx> üá  
ääãäöä:usdBDIGXHHA.3...@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

The closest thing that comes to mind is an RFC called stringprep. There are a wide variety of stringprep profiles, and while they don't quite do what you're looking for, they're close. Included in stringprep is a set of mapping tables for Uppder->Lower case conversions. These are (in that context) called case-foldings, are found in table B.2. Unfortunately, they're Upper->Lower, not the other way around.

Stringprep:  
<http://www.faqs.org/rfcs/rfc3454.html>

There are a number of profiles:  
[Profile for Internaional Domain Names]  
<http://www.rfc-editor.org/rfc/rfc3491.txt>

[Profile for iSCSI names]  
<http://tools.ietf.org/html/draft-ietf-ips-iscsi-string-prep-01>

Re: ToUpper() Better solution

[Profile for SASL UserNames & Passwords]  
<http://www.ietf.org/rfc/rfc4013.txt>

[Profile for XMPP Resources]  
<http://www.xmpp.org/internet-drafts/attic/draft-ietf-xmpp-resourcepre...>

There's a C# implementation of this RFC that's part of the libidn library.  
There's also a C++ & Java version.  
<http://www.gnu.org/software/libidn/>

We've actually got a full implementation of stringprep as well – it's much more .Net 2.0 ish than the libidn one, which is just a native C++ app that was then ported to Java & .Net. It's found in our open-source SoapBox Framework.

--  
Chris Mullins, MCSD.NET, MCPD:Enterprise, Microsoft C# MVP  
<http://www.coversant.com/blogs/cmullins>

"Jon Skeet [C# MVP]" <s...@xxxxxxxx> wrote in message <news:MPG.2050e52a399075e398d906@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Ornette <abstrait...nospam...@xxxxxxx>  
wrote:

So how would you do ?

The mapping table idea you had before looked best to me, although I

## Re: ToUpper() Better solution

wouldn't quite implement it the same way.  
I'd have a look up table for  
every possible character, where it defaults to  
the Unicode character,  
but for all the accented characters you care  
about, you specify the  
non-accented version.

You'd then call ToCharArray() on the string  
in question, go through  
each character replacing the original with the  
mapped character, and  
then create a new string with the char array.

It does require you to manually map all the  
accented characters you  
care about though.

My guess is that there are libraries around to  
do this somewhere, but  
I  
don't know of any myself.

--

Jon Skeet - <s...@xxxxxxxx>

<http://www.pobox.com/~skeet>

Blog: <http://www.msmpvs.com/jon.skeet>

If replying to the group, please do not mail  
me too