The great thing about URL encodings is that there are so many to choose from

devblogs.microsoft.com/oldnewthing/20100331-00

March 31, 2010

Raymond Chen

The phrase *URL encoding* appears to mean different things to different people.

First, Tim Berners-Lee says that URLs are encoded by using `%xx` to encode "dangerous" characters, or to suppress the special meaning that would normally be assigned to characters such as `/` or `?`. For example, the URL `http://server/why%3F/?q=bother` is a request to the server `server` with the path `/why?/` and with the query string `q=bother`. Notice that by escaping the question mark, we prevent it from being interpreted as the start of the query portion of the URL.

Now, it so happens that when a form is submitted via `GET`, then the contents of the form are encoded (by default) into the query according to a set of rules laid out in the HTML 4.01 specification: The query string takes the basic form of `var=value&var=value&...`. If a variable name or a value contains a "dangerous" character or a special character like `=` or `&`, then it must be %-escaped. For example, `co=AT%26T` says that the variable `co` has the value `AT&T`. Encoding the ampersand prevents it from being interpreted as a separator.

And here is the special additional rule that confuses a lot of people: When submitting a form via `GET`, the form data is encoded into the query portion of a URL, and under the default encoding, the character U+0020 (space) is encoded as U+002B (plus sign). This special use of the plus sign applies only to the query portion of the URL. Sometimes people get confused and think that it applies to URLs in general.

Example:

```
http://example.com/embedded%20space.html?key=apple+pie#result%20panel
```

The base URL and fragment (colored in blue) use the `%20` sequence to encode the embedded space, whereas the query (colored in green) uses the plus sign.

You'd think that would be the end of the story, but in fact it's just the beginning, because now we get to throw in all sorts of nonstandard URL encoders.

The PHP function `urlencode` treats the entire string as if it were a value (or variable name) in a query string, encoding spaces as a plus sign and being careful to escape all other punctuation. Not to be confused with `rawurlencode` which encodes everything (even characters like `/`).

JScript comes with a whole bucketload of functions for URL encoding. There's `escape()`, which encodes almost everything but leaves the slash and—bafflingly—the plus sign unencoded. And then there's the `encodeURI()` function which leaves a few more characters unencoded (including the colon (U+003A), and question mark (U+003F)). But wait, there's also `encodeURIComponent()` which goes to the effort of encoding slashes too. It's a total mess, but this site tries to make some sense out of the whole thing.

The ASP.Net function Server.UrlEncode behaves the same way as the PHP `urlencode` function.

There are probably a dozen other functions which purport to perform some form of URL encoding. You have to read the documentation on each one carefully to see whether it does the type of encoding you want.

But wait, you're not done yet. There are URL encodings which are built on top of the basic URL encoding.

The punycode encoding is used to encode Unicode characters in domain names, which have an even more limited character set than URLs.

When auto-generating a URL from a string, different Web sites use different algorithms. This isn't really an encoding in the URL encoding sense; it's just a convention for generating names for Web pages. The result of these conversion algorithms still need to be URL encoded.

For example, Wikipedia's URL auto-generation algorithm changes spaces to underscores. It leaves most punctuation marks unchanged, which means that once you've gone through Wikipedia's auto-generation algorithm, you still have to go back and escape all the characters which require escaping according to RFC3986.

As another example, it is popular with many blog software packages to change spaces to hyphens when auto-generating a URL from the title of a blog post. The handling of special characters varies. Some packages simply omit them; others try to encode them, resulting in a double-encoded string if the encoding uses characters for which RFC3986 requires encodings!

So if somebody asks a question about URL encoding, before you answer, make sure you understand what sense of the phrase "URL encoding" is being used.

Raymond Chen

Follow