The Microsoft Anti-XSS Library.

V4.0 Changes - 15 November, 2014

Abstract

The AntiXSS Library is licensed under the Microsoft Public License, an open source license the text of which can be read at http://www.microsoft.com/opensource/licenses.mspx.

Contents

Code Listings

It is often necessary to split code listings over multiple lines in this document when, in fact, they are a single line. The ⮰ symbol indicates where this split takes place. When entering code containing ⮰ symbols please ensure you put the code on a single line.

Introduction to the AntiXSS Libary

Getting Help

If you have a question or comment please post it on the discussion forum on the WPL CodePlex site.

Changes in V4.0

Return values

If you pass a null as the value to be encoding to an encoding function the function will now return null. Previous behavior was to return `String.Empty`.

Medium Trust Support

The HTML Sanitization methods, `GetSafeHtml()` and `GetSafeHtmlFragment()` have been moved to a separate assembly. This enables the AntiXssLibrary assembly to run in medium trust environments, a common user request. If you wish to use the Html Sanitization library you must now include the HtmlSanitizationLibrary assembly. This assembly requires full trust and the ability to run unsafe code.

Adjustable safe-listing for HTML/XML Encoding

The safe list for HTML and XML encoding is now adjustable. The `MarkAsSafe(LowerCodeCharts, LowerMidCodeCharts, MidCodeCharts, UpperMidCodeCharts, UpperCodeCharts)` method allows to you choose from the Unicode Code Charts which languages your web application normally accepts. Safe-listing a language code chart leaves the defined characters in their native form during encoding, which increases readability in the HTML/XML document and speeds up encoding. Certain dangerous characters will also be encoded. The language code charts are defined in the `Microsoft.Security.Application.LowerCodeCharts`, `Microsoft.Security.Application.LowerMidCodeCharts`, `Microsoft.Security.Application.MidCodeCharts`, `Microsoft.Security.Application.UpperMidCodeCharts` and `Microsoft.Security.Application.UpperCodeCharts` enumerations.

It is suggested you safe list your acceptable languages during your application initialization.

Invalid Unicode character detection

If any of the HTML, XML or CSS encoding methods encounters a character with a character code of 0xFFFE or 0xFFFF, the characters used to detect byte order at the beginning of files an `InvalidUnicodeValueException` will be thrown.

HTML 4.01 Named Entity Support

A new overload of the `HtmlEncode` method,

```
HtmlEncode(string input, bool useNamedEntities)
```

allows you to specify if the named entities from the HTML 4.01 specification should be used in preference to `&#xxxx;` encoding when a named entity exists. For example if `useNamedEntities` is set to true © would be encoded as `&copy;`.

Surrogate Character Support in HTML and XML encoding

Support for surrogate character pairs for Unicode characters outside the basic multilingual plane has been improved. Such character pairs are now combined and encoded as their &xxxxx; value.

If a high surrogate pair character is encountered which is not followed by a low surrogate pair character, or a low surrogate pair character is encountered which is not preceded by a high surrogate pair character an `InvalidSurrogatePairException` is thrown.

HtmlFormUrlEncode

A new encoding type suitable for using in encoding Html POST form submissions is now available via `HtmlFormUrlEncode(string input)`. This encodes according to the W3C specifications for `application/x-www-form-urlencoded` MIME type.

LDAP Encoding changes

The `LdapEncode` function has been deprecated in favor of two new functions, `LdapFilterEncode` and `LdapDistinguishedNameEncode`.

`LdapFilterEncode` encodes input according to RFC4515 where unsafe values are converted to \XX where XX is the representation of the unsafe character. For example

Input	Output
`Parens R Us (for all your parenthetical needs)`	`Parens R Us \28for all your parenthetical needs\29`
`*`	`\2A`
`C:\MyFile`	`C:\5CMyFile`
`Lučić`	`Lu\C4\8Di\C4\87`

`LdapDistinguishedNameEncode` encodes input according to RFC 2253 where unsafe characters are converted to #XX where XX is the representation of the unsafe character and the comma, plus, quote, slash, less than and great than signs are escaped using slash notation (\X). In addition to this a space or octothorpe (#) at the beginning of the input string is \ escaped as is a space at the end of a string.

Input	Output
`, + \ " \ < >`	`\, \+ \" \\ \< \>`
` Hello`	`\ Hello`
`Hello `	`Hello\ `
`#Hello`	`\#Hello`
`Lučić`	`Lu#C4#8Di#C4#87`

An override

```
LdapDistinguishedNameEncode(
  string input,
  bool useInitialCharacterRules,
  bool useFinalCharacterRule)
```

is also provided so you may turn off the initial or final character escaping rules, for example if you are concatenating the escaped distinguished name fragment into the midst of a complete distinguished name.
In addition to the RFC mandated escaping the safe list excludes the characters listed at http://projects.webappsec.org/LDAP-Injection.

MarkOutput

The ability to mark output using an `HtmlEncode` overload and query string parameter has been removed.