Build the hash table of HTML entity references and encode Url

```
//
// System.Web.HttpUtility
//
// Authors:
// Patrik Torstensson (Patrik.Torstensson@labs2.com)
// Wictor Wilén (decode/encode functions) (wictor@ibizkit.se)
// Tim Coleman (tim@timcoleman.com)
// Gonzalo Paniagua Javier (gonzalo@ximian.com)
//
// Copyright (C) 2005 Novell, Inc (http://www.novell.com)
//
// Permission is hereby granted, free of charge, to any person obtaining
// a copy of this software and associated documentation files (the
// “Software”), to deal in the Software without restriction, including
// without limitation the rights to use, copy, modify, merge, publish,
// distribute, sublicense, and/or sell copies of the Software, and to
// permit persons to whom the Software is furnished to do so, subject to
// the following conditions:
//
// The above copyright notice and this permission notice shall be
// included in all copies or substantial portions of the Software.
//
```

```
// THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND,
// EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
// NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
// LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
// OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
// WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//

using System;
using System.Collections;
using System.Collections.Specialized;
using System.Globalization;
using System.IO;
using System.Security.Permissions;
using System.Text;

#if NETCF

namespace System.Web
{
internal sealed class HttpUtility
{

#region Fields

static Hashtable entities;
static object lock_ = new object();

#endregion // Fields

static Hashtable Entities
{
```

```
get
{
lock (lock_)
{
if (entities == null)
InitEntities();

return entities;
}
}
}

#region Constructors

static void InitEntities()
{
// Build the hash table of HTML entity references. This list
comes
// from the HTML 4.01 W3C recommendation.
entities = new Hashtable();
entities.Add(“nbsp”, 'u00A0');
entities.Add(“iexcl”, 'u00A1');
entities.Add(“cent”, 'u00A2');
entities.Add(“pound”, 'u00A3');
entities.Add(“curren”, 'u00A4');
entities.Add(“yen”, 'u00A5');
entities.Add(“brvbar”, 'u00A6');
entities.Add(“sect”, 'u00A7');
entities.Add(“uml”, 'u00A8');
entities.Add(“copy”, 'u00A9');
entities.Add(“ordf”, 'u00AA');
entities.Add(“laquo”, 'u00AB');
entities.Add(“not”, 'u00AC');
entities.Add(“shy”, 'u00AD');
entities.Add(“reg”, 'u00AE');
entities.Add(“macr”, 'u00AF');
entities.Add(“deg”, 'u00B0');
entities.Add(“plusmn”, 'u00B1');
```

```
entities.Add(“sup2”, 'u00B2');
entities.Add(“sup3”, 'u00B3');
entities.Add(“acute”, 'u00B4');
entities.Add(“micro”, 'u00B5');
entities.Add(“para”, 'u00B6');
entities.Add(“middot”, 'u00B7');
entities.Add(“cedil”, 'u00B8');
entities.Add(“sup1”, 'u00B9');
entities.Add(“ordm”, 'u00BA');
entities.Add(“raquo”, 'u00BB');
entities.Add(“frac14”, 'u00BC');
entities.Add(“frac12”, 'u00BD');
entities.Add(“frac34”, 'u00BE');
entities.Add(“iquest”, 'u00BF');
entities.Add(“Agrave”, 'u00C0');
entities.Add(“Aacute”, 'u00C1');
entities.Add(“Acirc”, 'u00C2');
entities.Add(“Atilde”, 'u00C3');
entities.Add(“Auml”, 'u00C4');
entities.Add(“Aring”, 'u00C5');
entities.Add(“AElig”, 'u00C6');
entities.Add(“Ccedil”, 'u00C7');
entities.Add(“Egrave”, 'u00C8');
entities.Add(“Eacute”, 'u00C9');
entities.Add(“Ecirc”, 'u00CA');
entities.Add(“Euml”, 'u00CB');
entities.Add(“Igrave”, 'u00CC');
entities.Add(“Iacute”, 'u00CD');
entities.Add(“Icirc”, 'u00CE');
entities.Add(“Iuml”, 'u00CF');
entities.Add(“ETH”, 'u00D0');
entities.Add(“Ntilde”, 'u00D1');
entities.Add(“Ograve”, 'u00D2');
entities.Add(“Oacute”, 'u00D3');
entities.Add(“Ocirc”, 'u00D4');
entities.Add(“Otilde”, 'u00D5');
entities.Add(“Ouml”, 'u00D6');
```

```
entities.Add(“times”, 'u00D7');
entities.Add(“Oslash”, 'u00D8');
entities.Add(“Ugrave”, 'u00D9');
entities.Add(“Uacute”, 'u00DA');
entities.Add(“Ucirc”, 'u00DB');
entities.Add(“Uuml”, 'u00DC');
entities.Add(“Yacute”, 'u00DD');
entities.Add(“THORN”, 'u00DE');
entities.Add(“szlig”, 'u00DF');
entities.Add(“agrave”, 'u00E0');
entities.Add(“aacute”, 'u00E1');
entities.Add(“acirc”, 'u00E2');
entities.Add(“atilde”, 'u00E3');
entities.Add(“auml”, 'u00E4');
entities.Add(“aring”, 'u00E5');
entities.Add(“aelig”, 'u00E6');
entities.Add(“ccedil”, 'u00E7');
entities.Add(“egrave”, 'u00E8');
entities.Add(“eacute”, 'u00E9');
entities.Add(“ecirc”, 'u00EA');
entities.Add(“euml”, 'u00EB');
entities.Add(“igrave”, 'u00EC');
entities.Add(“iacute”, 'u00ED');
entities.Add(“icirc”, 'u00EE');
entities.Add(“iuml”, 'u00EF');
entities.Add(“eth”, 'u00F0');
entities.Add(“ntilde”, 'u00F1');
entities.Add(“ograve”, 'u00F2');
entities.Add(“oacute”, 'u00F3');
entities.Add(“ocirc”, 'u00F4');
entities.Add(“otilde”, 'u00F5');
entities.Add(“ouml”, 'u00F6');
entities.Add(“divide”, 'u00F7');
entities.Add(“oslash”, 'u00F8');
entities.Add(“ugrave”, 'u00F9');
entities.Add(“uacute”, 'u00FA');
entities.Add(“ucirc”, 'u00FB');
```

```
entities.Add(“uuml”, 'u00FC');
entities.Add(“yacute”, 'u00FD');
entities.Add(“thorn”, 'u00FE');
entities.Add(“yuml”, 'u00FF');
entities.Add(“fnof”, 'u0192');
entities.Add(“Alpha”, 'u0391');
entities.Add(“Beta”, 'u0392');
entities.Add(“Gamma”, 'u0393');
entities.Add(“Delta”, 'u0394');
entities.Add(“Epsilon”, 'u0395');
entities.Add(“Zeta”, 'u0396');
entities.Add(“Eta”, 'u0397');
entities.Add(“Theta”, 'u0398');
entities.Add(“Iota”, 'u0399');
entities.Add(“Kappa”, 'u039A');
entities.Add(“Lambda”, 'u039B');
entities.Add(“Mu”, 'u039C');
entities.Add(“Nu”, 'u039D');
entities.Add(“Xi”, 'u039E');
entities.Add(“Omicron”, 'u039F');
entities.Add(“Pi”, 'u03A0');
entities.Add(“Rho”, 'u03A1');
entities.Add(“Sigma”, 'u03A3');
entities.Add(“Tau”, 'u03A4');
entities.Add(“Upsilon”, 'u03A5');
entities.Add(“Phi”, 'u03A6');
entities.Add(“Chi”, 'u03A7');
entities.Add(“Psi”, 'u03A8');
entities.Add(“Omega”, 'u03A9');
entities.Add(“alpha”, 'u03B1');
entities.Add(“beta”, 'u03B2');
entities.Add(“gamma”, 'u03B3');
entities.Add(“delta”, 'u03B4');
entities.Add(“epsilon”, 'u03B5');
entities.Add(“zeta”, 'u03B6');
entities.Add(“eta”, 'u03B7');
entities.Add(“theta”, 'u03B8');
```

```
entities.Add(“iota”, 'u03B9');
entities.Add(“kappa”, 'u03BA');
entities.Add(“lambda”, 'u03BB');
entities.Add(“mu”, 'u03BC');
entities.Add(“nu”, 'u03BD');
entities.Add(“xi”, 'u03BE');
entities.Add(“omicron”, 'u03BF');
entities.Add(“pi”, 'u03C0');
entities.Add(“rho”, 'u03C1');
entities.Add(“sigmaf”, 'u03C2');
entities.Add(“sigma”, 'u03C3');
entities.Add(“tau”, 'u03C4');
entities.Add(“upsilon”, 'u03C5');
entities.Add(“phi”, 'u03C6');
entities.Add(“chi”, 'u03C7');
entities.Add(“psi”, 'u03C8');
entities.Add(“omega”, 'u03C9');
entities.Add(“thetasym”, 'u03D1');
entities.Add(“upsih”, 'u03D2');
entities.Add(“piv”, 'u03D6');
entities.Add(“bull”, 'u2022');
entities.Add(“hellip”, 'u2026');
entities.Add(“prime”, 'u2032');
entities.Add(“Prime”, 'u2033');
entities.Add(“oline”, 'u203E');
entities.Add(“frasl”, 'u2044');
entities.Add(“weierp”, 'u2118');
entities.Add(“image”, 'u2111');
entities.Add(“real”, 'u211C');
entities.Add(“trade”, 'u2122');
entities.Add(“alefsym”, 'u2135');
entities.Add(“larr”, 'u2190');
entities.Add(“uarr”, 'u2191');
entities.Add(“rarr”, 'u2192');
entities.Add(“darr”, 'u2193');
entities.Add(“harr”, 'u2194');
entities.Add(“crarr”, 'u21B5');
```

```
entities.Add(“lArr”, 'u21D0');
entities.Add(“uArr”, 'u21D1');
entities.Add(“rArr”, 'u21D2');
entities.Add(“dArr”, 'u21D3');
entities.Add(“hArr”, 'u21D4');
entities.Add(“forall”, 'u2200');
entities.Add(“part”, 'u2202');
entities.Add(“exist”, 'u2203');
entities.Add(“empty”, 'u2205');
entities.Add(“nabla”, 'u2207');
entities.Add(“isin”, 'u2208');
entities.Add(“notin”, 'u2209');
entities.Add(“ni”, 'u220B');
entities.Add(“prod”, 'u220F');
entities.Add(“sum”, 'u2211');
entities.Add(“minus”, 'u2212');
entities.Add(“lowast”, 'u2217');
entities.Add(“radic”, 'u221A');
entities.Add(“prop”, 'u221D');
entities.Add(“infin”, 'u221E');
entities.Add(“ang”, 'u2220');
entities.Add(“and”, 'u2227');
entities.Add(“or”, 'u2228');
entities.Add(“cap”, 'u2229');
entities.Add(“cup”, 'u222A');
entities.Add(“int”, 'u222B');
entities.Add(“there4”, 'u2234');
entities.Add(“sim”, 'u223C');
entities.Add(“cong”, 'u2245');
entities.Add(“asymp”, 'u2248');
entities.Add(“ne”, 'u2260');
entities.Add(“equiv”, 'u2261');
entities.Add(“le”, 'u2264');
entities.Add(“ge”, 'u2265');
entities.Add(“sub”, 'u2282');
entities.Add(“sup”, 'u2283');
entities.Add(“nsub”, 'u2284');
```

```
entities.Add(“sube”, 'u2286');
entities.Add(“supe”, 'u2287');
entities.Add(“oplus”, 'u2295');
entities.Add(“otimes”, 'u2297');
entities.Add(“perp”, 'u22A5');
entities.Add(“sdot”, 'u22C5');
entities.Add(“lceil”, 'u2308');
entities.Add(“rceil”, 'u2309');
entities.Add(“lfloor”, 'u230A');
entities.Add(“rfloor”, 'u230B');
entities.Add(“lang”, 'u2329');
entities.Add(“rang”, 'u232A');
entities.Add(“loz”, 'u25CA');
entities.Add(“spades”, 'u2660');
entities.Add(“clubs”, 'u2663');
entities.Add(“hearts”, 'u2665');
entities.Add(“diams”, 'u2666');
entities.Add(“quot”, 'u0022');
entities.Add(“amp”, 'u0026');
entities.Add(“lt”, 'u003C');
entities.Add(“gt”, 'u003E');
entities.Add(“OElig”, 'u0152');
entities.Add(“oelig”, 'u0153');
entities.Add(“Scaron”, 'u0160');
entities.Add(“scaron”, 'u0161');
entities.Add(“Yuml”, 'u0178');
entities.Add(“circ”, 'u02C6');
entities.Add(“tilde”, 'u02DC');
entities.Add(“ensp”, 'u2002');
entities.Add(“emsp”, 'u2003');
entities.Add(“thinsp”, 'u2009');
entities.Add(“zwnj”, 'u200C');
entities.Add(“zwj”, 'u200D');
entities.Add(“lrm”, 'u200E');
entities.Add(“rlm”, 'u200F');
entities.Add(“ndash”, 'u2013');
entities.Add(“mdash”, 'u2014');
```

```
entities.Add(“lsquo”, 'u2018');
entities.Add(“rsquo”, 'u2019');
entities.Add(“sbquo”, 'u201A');
entities.Add(“ldquo”, 'u201C');
entities.Add(“rdquo”, 'u201D');
entities.Add(“bdquo”, 'u201E');
entities.Add(“dagger”, 'u2020');
entities.Add(“Dagger”, 'u2021');
entities.Add(“permil”, 'u2030');
entities.Add(“lsaquo”, 'u2039');
entities.Add(“rsaquo”, 'u203A');
entities.Add(“euro”, 'u20AC');
}

public HttpUtility()
{
}

#endregion // Constructors

#region Methods

private static int GetInt(byte b)
{
char c = (char)b;
if (c >= '0' && c <= '9') return c - '0'; if (c >= 'a' && c <= 'f') return c - 'a' + 10; if (c >= 'A' && c <= 'F') return c - 'A' + 10; return -1; } public static string UrlEncode(string str) { return UrlEncode(str, Encoding.UTF8); } public static string UrlEncode(string s, Encoding Enc) { if (s == null) return null; if (s == "") return ""; byte[] bytes = Enc.GetBytes(s);                                    return Encoding.ASCII.GetString(UrlEncodeToBytes(bytes,     0, bytes.Length), 0, bytes.Length); } public static string UrlEncode(byte[] bytes) { if (bytes == null) return null; if (bytes.Length    ==    0)    return    "";    return Encoding.ASCII.GetString(UrlEncodeToBytes(bytes,     0, bytes.Length), 0, bytes.Length); } public static string
```

```
UrlEncode(byte[] bytes, int offset, int count) { if (bytes == null) return null; if (bytes.Length == 0) return ""; return Encoding.ASCII.GetString(UrlEncodeToBytes(bytes, offset, count), offset, count); } public static byte[] UrlEncodeToBytes(string str) { return UrlEncodeToBytes(str, Encoding.UTF8); } public static byte[] UrlEncodeToBytes(string str, Encoding e) { if (str == null) return null; if (str == "") return new byte[0]; byte[] bytes = e.GetBytes(str); return UrlEncodeToBytes(bytes, 0, bytes.Length); } public static byte[] UrlEncodeToBytes(byte[] bytes) { if (bytes == null) return null; if (bytes.Length == 0) return new byte[0]; return UrlEncodeToBytes(bytes, 0, bytes.Length); } static char[] hexChars = "0123456789abcdef".ToCharArray(); const string notEncoded = "!'()*-._"; static void UrlEncodeChar(char c, Stream result, bool isUnicode) { if (c > 255)
{
//FIXME: what happens when there is an internal error?
//if (!isUnicode)
// throw new ArgumentOutOfRangeException (“c”, c, “c must be less than 256”);
int idx;
int i = (int)c;

result.WriteByte((byte)'%');
result.WriteByte((byte)'u');
idx = i >> 12;
result.WriteByte((byte)hexChars[idx]);
idx = (i >> 8) & 0x0F;
result.WriteByte((byte)hexChars[idx]);
idx = (i >> 4) & 0x0F;
result.WriteByte((byte)hexChars[idx]);
idx = i & 0x0F;
result.WriteByte((byte)hexChars[idx]);
return;
}

if (c > ' ' && notEncoded.IndexOf(c) != -1)
```

```
{
result.WriteByte((byte)c);
return;
}
if (c == ' ')
{
result.WriteByte((byte)'+');
return;
}
if ((c < '0') || (c < 'A' && c > '9') ||
(c > 'Z' && c < 'a') || (c > 'z'))
{
if (isUnicode && c > 127)
{
result.WriteByte((byte)'%');
result.WriteByte((byte)'u');
result.WriteByte((byte)'0');
result.WriteByte((byte)'0');
}
else
result.WriteByte((byte)'%');

int idx = ((int)c) >> 4;
result.WriteByte((byte)hexChars[idx]);
idx = ((int)c) & 0x0F;
result.WriteByte((byte)hexChars[idx]);
}
else
result.WriteByte((byte)c);
}

public static byte[] UrlEncodeToBytes(byte[] bytes, int
offset, int count)
{
if (bytes == null)
return null;

int len = bytes.Length;
```

```
if (len == 0)
return new byte[0];

if (offset < 0 || offset >= len)
throw new ArgumentOutOfRangeException(“offset”);

if (count < 0 || count > len – offset)
throw new ArgumentOutOfRangeException(“count”);

MemoryStream result = new MemoryStream(count);
int end = offset + count;
for (int i = offset; i < end; i++) UrlEncodeChar((char)bytes[i], result, false); return result.ToArray(); } public static string UrlEncodeUnicode(string str) { if (str == null) return null; byte [] bytes = Encoding.ASCII.GetBytes (str); bytes = UrlEncodeToBytes (bytes, 0, bytes.Length); return Encoding.ASCII.GetString(bytes, 0, bytes.Length); } public static byte[] UrlEncodeUnicodeToBytes(string str) { if (str == null) return null; if (str == "") return new byte[0]; MemoryStream result = new MemoryStream(str.Length); foreach (char c in str) { UrlEncodeChar(c, result, true); } return result.ToArray(); } #endregion // Methods } } #endif [/csharp]
```