

textutil.cm

```
/*  
  
    Copyright (c) 2012–2016 Seppo Laakko  
    http://sourceforge.net/projects/cmajor/  
  
    Distributed under the GNU General Public License, version 3 (GPLv3).  
    (See accompanying LICENSE.txt or http://www.gnu.org/licenses/gpl.html  
    )  
  
*/  
  
// Copyright (c) 1994  
// Hewlett-Packard Company  
// Copyright (c) 1996  
// Silicon Graphics Computer Systems, Inc.  
// Copyright (c) 2009 Alexander Stepanov and Paul McJones  
  
using System;  
  
namespace System.Text  
{  
    public nothrow string HexEscape(char c)  
    {  
        return "\\x" + ToHexString(cast<byte>(c));  
    }  
  
    public nothrow string HexEscape(wchar c)  
    {  
        return "\\x" + ToHexString(cast<ushort>(c));  
    }  
  
    public nothrow string HexEscape(uchar c)  
    {  
        return "\\x" + ToHexString(cast<uint>(c));  
    }  
  
    public nothrow string CharStr(char c)  
    {  
        switch (c)  
        {  
            case '\\': return "\\\\";  
            case '\"': return "\\\"";  
            case '\': return "\\\"";  
            case 'a': return "\\a";  
            case 'b': return "\\b";  
            case 'f': return "\\f";  
            case 'n': return "\\n";  
            case 'r': return "\\r";  
            case 't': return "\\t";  
        }  
    }  
}
```

```

        case '\\v': return "\\v";
        case '\\0': return "\\0";
        default:
        {
            if (IsPrintable(c))
            {
                return string(c);
            }
            else
            {
                return HexEscape(c);
            }
        }
    }
}

public nothrow string MakeCharLiteral(char c)
{
    if (c == '"' )
    {
        return string("'\"'");
    }
    return "\"" + CharStr(c) + "\"";
}

public nothrow string MakeWCharLiteral(wchar c)
{
    if (cast<ushort>(c) < 0x100u)
    {
        return MakeCharLiteral(cast<char>(c));
    }
    else
    {
        return "\"" + HexEscape(c) + "\"";
    }
}

public nothrow string MakeUCharLiteral(uchar c)
{
    if (cast<uint>(c) < 0x100u)
    {
        return MakeCharLiteral(cast<char>(c));
    }
    else
    {
        return "\"" + HexEscape(c) + "\"";
    }
}

public nothrow string StringStr(const string& s)
{
    string result;
    for (char c : s)

```

```

    {
        if (c == '\\')
        {
            result.Append(c);
        }
        else
        {
            result.Append(CharStr(c));
        }
    }
    return result;
}

public nothrow string MakeStringLiteral(const string& s)
{
    string result("\\");
    result.Append(StringStr(s));
    result.Append("\\");
    return result;
}

public string Trim(const string& s)
{
    int b = 0;
    while (b < s.Length() && IsSpace(s[b]))
    {
        ++b;
    }
    int e = s.Length() - 1;
    while (e >= b && IsSpace(s[e]))
    {
        --e;
    }
    return s.Substring(b, e - b + 1);
}

public string TrimAll(const string& s)
{
    string result;
    result.Reserve(s.Length());
    int state = 0;
    for (char c : s)
    {
        switch (state)
        {
            case 0: // skip starting spaces
            {
                if (!IsSpace(c))
                {
                    result.Append(c);
                    state = 1;
                }
                break;
            }
        }
    }
}

```

```

    }
    case 1: // collect non-space characters
    {
        if (IsSpace(c))
        {
            state = 2;
        }
        else
        {
            result.Append(c);
        }
        break;
    }
    case 2: // replace spaces in the middle with one space
           character
    {
        if (!IsSpace(c))
        {
            result.Append(' ');
            result.Append(c);
            state = 1;
        }
        break;
    }
}
return result;
}
}

```