

Gets a property's parent object

/*

Copyright (c) 2010 [James Craig](#)

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.*/

```

#region Usings
using System;
using System.IO;
using System.Reflection;
using System.Text;
using System.Linq.Expressions;
using System.Collections;
#endregion

namespace Utilities
{
    ///
    /// Utility class that handles various
    /// functions dealing with reflection.
    ///
    public static class Reflection
    {
        ///
        /// Gets a property's parent object
        ///
        /// Source object /// Path of the property (ex:
        Prop1.Prop2.Prop3 would be
        /// the Prop1 of the source object, which then has a Prop2 on
        it, which in turn
        /// has a Prop3 on it.) /// Property info that is sent back
        /// The property's parent object
        public static object GetPropertyParent(object SourceObject,
        string PropertyPath, out PropertyInfo PropertyInfo)
        {
            try
            {
                if (SourceObject == null)
                {
                    PropertyInfo = null;
                    return null;
                }
            }
        }
    }
}

```

```
string[] Splitter = { "." };
string[] SourceProperties = PropertyPath.Split(Splitter,
StringSplitOptions.None);
object TempSourceProperty = SourceObject;
Type PropertyType = SourceObject.GetType();
PropertyInfo = PropertyType.GetProperty(SourceProperties[0]);
PropertyType = PropertyInfo.PropertyType;
for (int x = 1; x < SourceProperties.Length; ++x) { if
(TempSourceProperty != null) { TempSourceProperty =
PropertyInfo.GetValue(TempSourceProperty, null); }
PropertyInfo = PropertyType.GetProperty(SourceProperties[x]);
PropertyType = PropertyInfo.PropertyType; } return
TempSourceProperty; } catch { throw; } } }
```