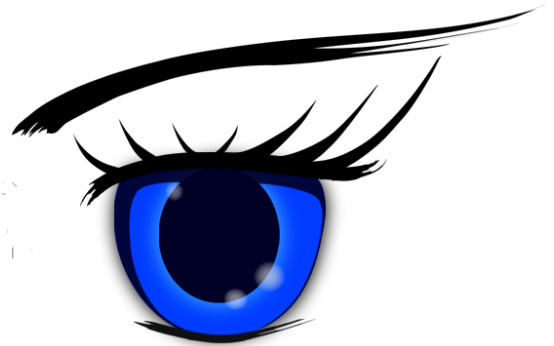


EYECATCH



# My First Umbraco Usercontrol

*For a long time, I have been able to develop for Umbraco only by using XSLT. However, recently I got a task that could not be solved by XSLT alone. I have never done a usercontrol before, so this was sure to be an interesting experience. The task is to find a way to generate a CSV representation of any Umbraco page. I decided to write a series of blog posts about my progress, to hopefully aid other people in understanding usercontrols.*

Ove Andersen  
[Velg dato]

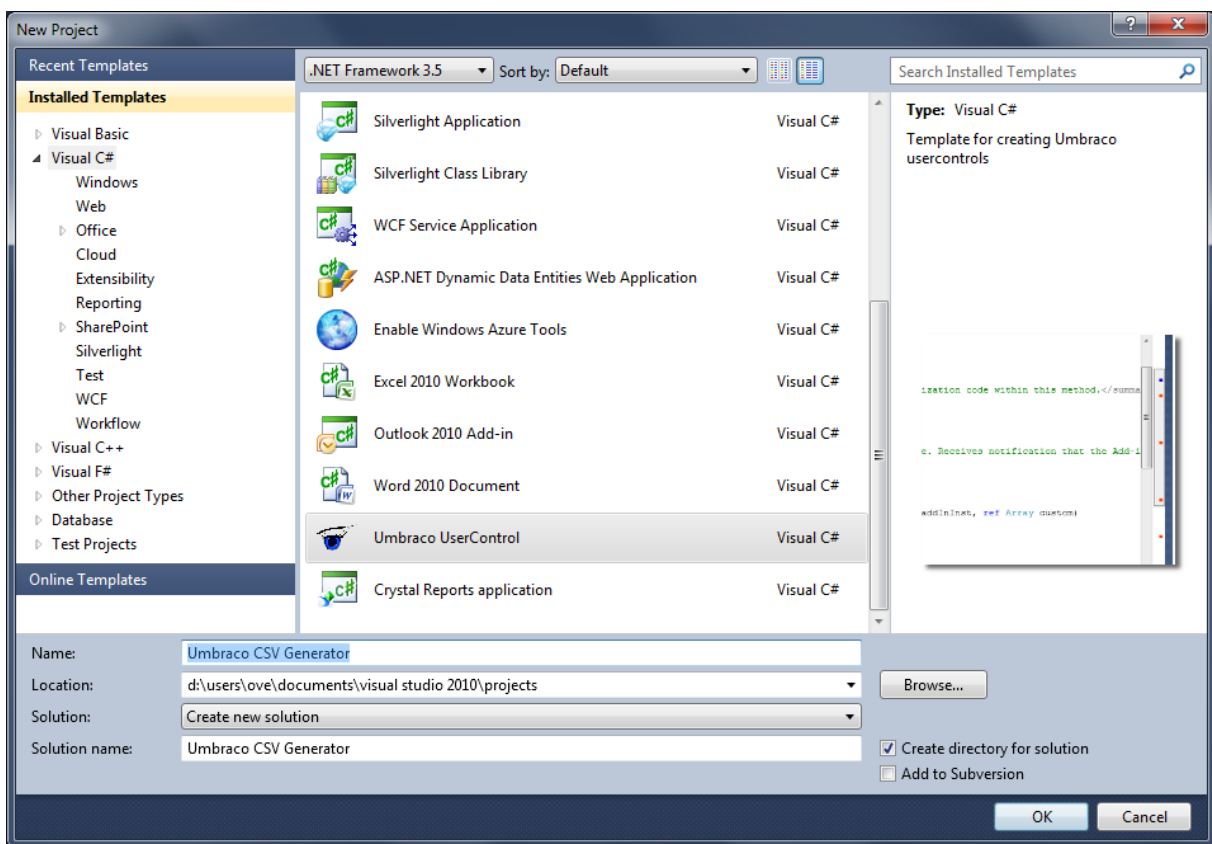


## Part 1a: Setting up Visual Studio 2010

First, we need to create a new Web Application. To ease the process of setting up the usercontrol in Visual Studio 2010, I created a template which you can download [here](#).

First, install the template you just downloaded by double-clicking on it. If the file has a .zip extension, rename it to .vsix.

Then create a new Web Application by clicking “File” -> “New Project” -> “Visual C#” -> “Umbraco UserControl”. You can call the usercontrol whatever you like. My project is called “EyeCatchCSVGenerator”.



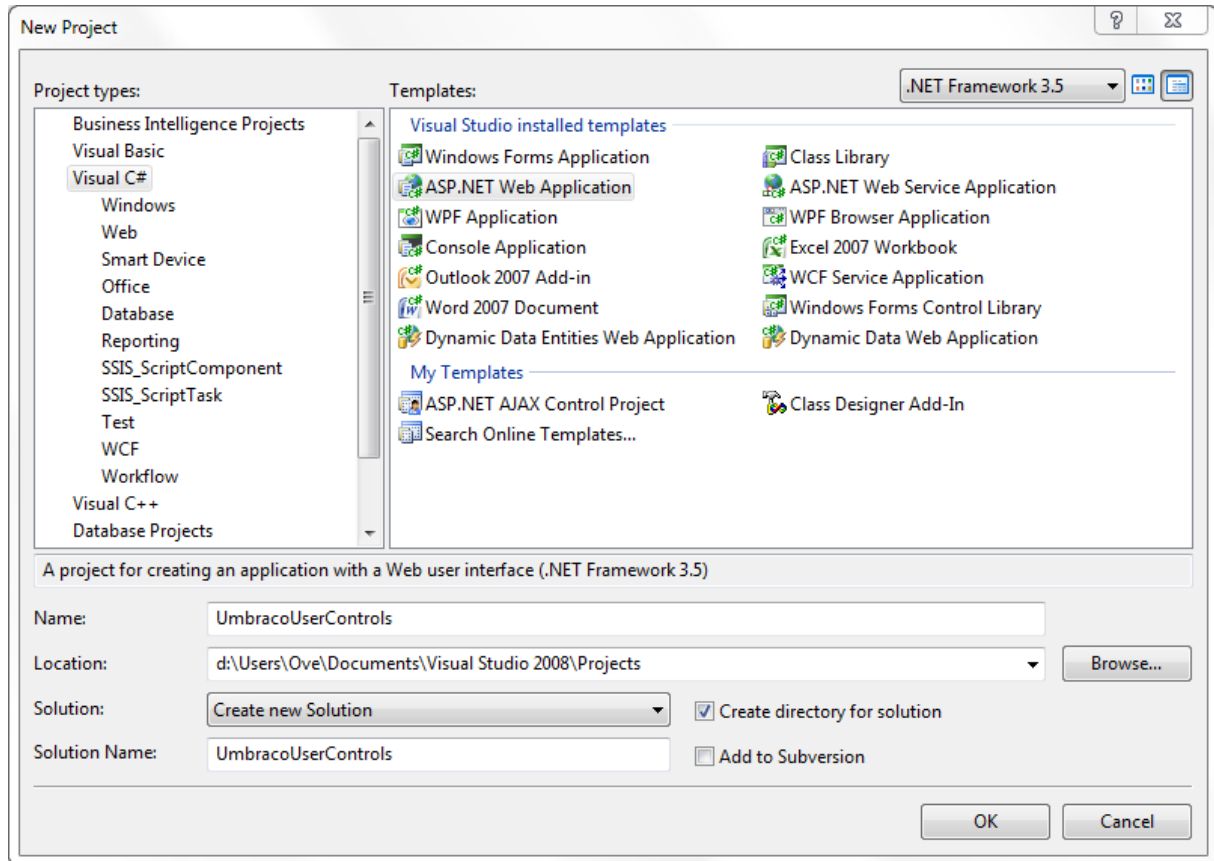
2

Now you are ready to create your first Umbraco usercontrol!



## Part 1b: Setting up Visual Studio 2008

First, create a new Web Application (You can find it by clicking “New Project” -> “ASP.NET Web Application”). You can call it whatever you like. My project is called “EyeCatchCSVGenerator”.



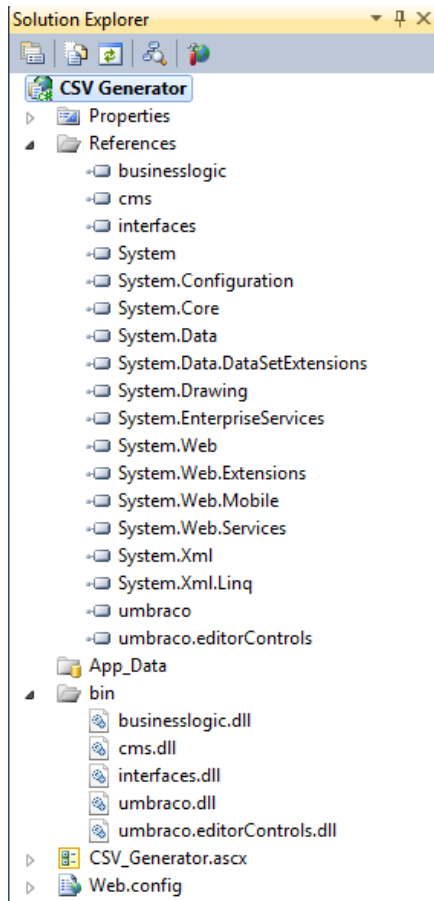
3

Then you need to copy some files from Umbraco to your “bin” folder. Not all of the following files are necessary, but I included them just in case. [You can find a zip with the files here.](#)

- umbraco.dll
- umbraco.editorControls.dll
- cms.dll
- interfaces.dll
- businesslogic.dll

You also need to add references to these files. You can do this by right-clicking the “References” folder and press “Add Reference”. Then you go to the “Browse” pane, and locate the .dll files you just copied.

Finally, your project should look like this:



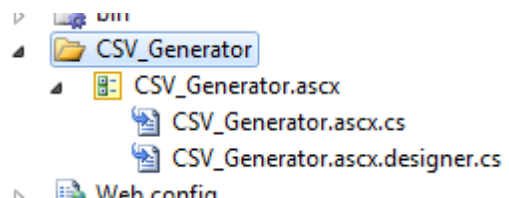
4

Now, you can choose if you want to create a separate Visual Studio project for every usercontrol or just make one that contains them all.

### A. One Project, One UserControl (My preference)

In this case you create one project per usercontrol you develop. I think it is a much cleaner way of development.

The setup here is simple, just right -click your project and click "Add New Item", then choose "Web User Control". Give your usercontrol a proper name.



### B. One Project, Multiple UserControls

In this case you create a separate folder for every usercontrol inside the project.

Right -click your folder and click "Add New Item", then choose "Web User Control". Give your usercontrol a proper name.



## Part 2: Basic Functionality Test

I needed to make sure that the usercontrol was working. The easiest way for me to do this, was to make the usercontrol print out a parameter sent to the usercontrol from the macro.

Here is the code:

### EyeCatchCSVGenerator.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeBehind="EyeCatchCSVGenerator.ascx.cs"
Inherits="EyeCatchCSVGenerator.EyeCatchCSVGenerator" %>
```

```
<h4>If everything works, the source node should show up below</h4>
<table id="sourcenode" runat="server"></table>
```

### EyeCatchCSVGenerator.ascx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using umbraco.presentation.nodeFactory;

namespace EyeCatchCSVGenerator
{
    public partial class EyeCatchCSVGenerator : System.Web.UI.UserControl
    {
        public int Source { get; set; }

        protected void Page_Load(object sender, EventArgs e)
        {
            Node source = new Node(Source);

            HtmlTableRow r = new HtmlTableRow();
            HtmlTableCell c1 = new HtmlTableCell();
            HtmlTableCell c2 = new HtmlTableCell();

            c1.InnerHtml = "Name: " + source.Name;
            c2.InnerHtml = "ID:" + source.Id;

            r.Cells.Add(c1);
            r.Cells.Add(c2);

            sourcenode.Rows.Add(r);
        }
    }
}
```

5

Build your project. It should build without problems.

20.04.2010

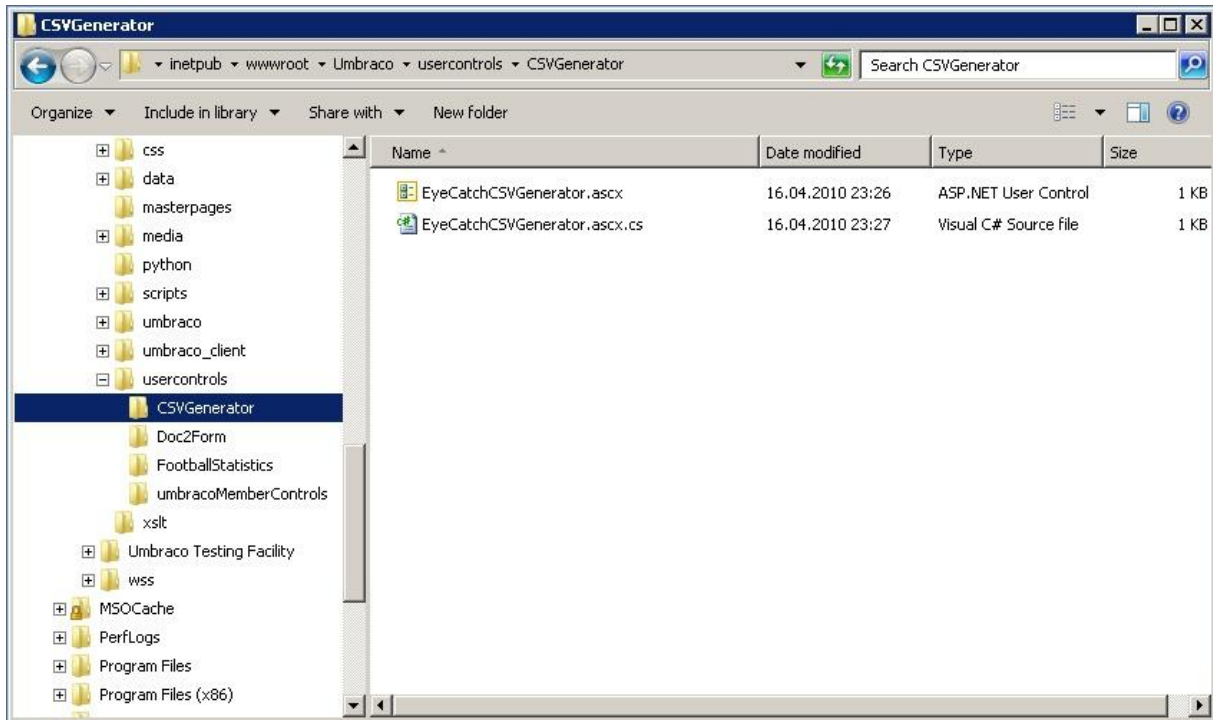
Org. No: NO 991 157 328 MVA  
Phone: +47 99 01 66 10

Address: Straume Teknologisenter  
v/ 3D Design Media  
Trollhaugmyra 15  
5353 Straume  
Norway



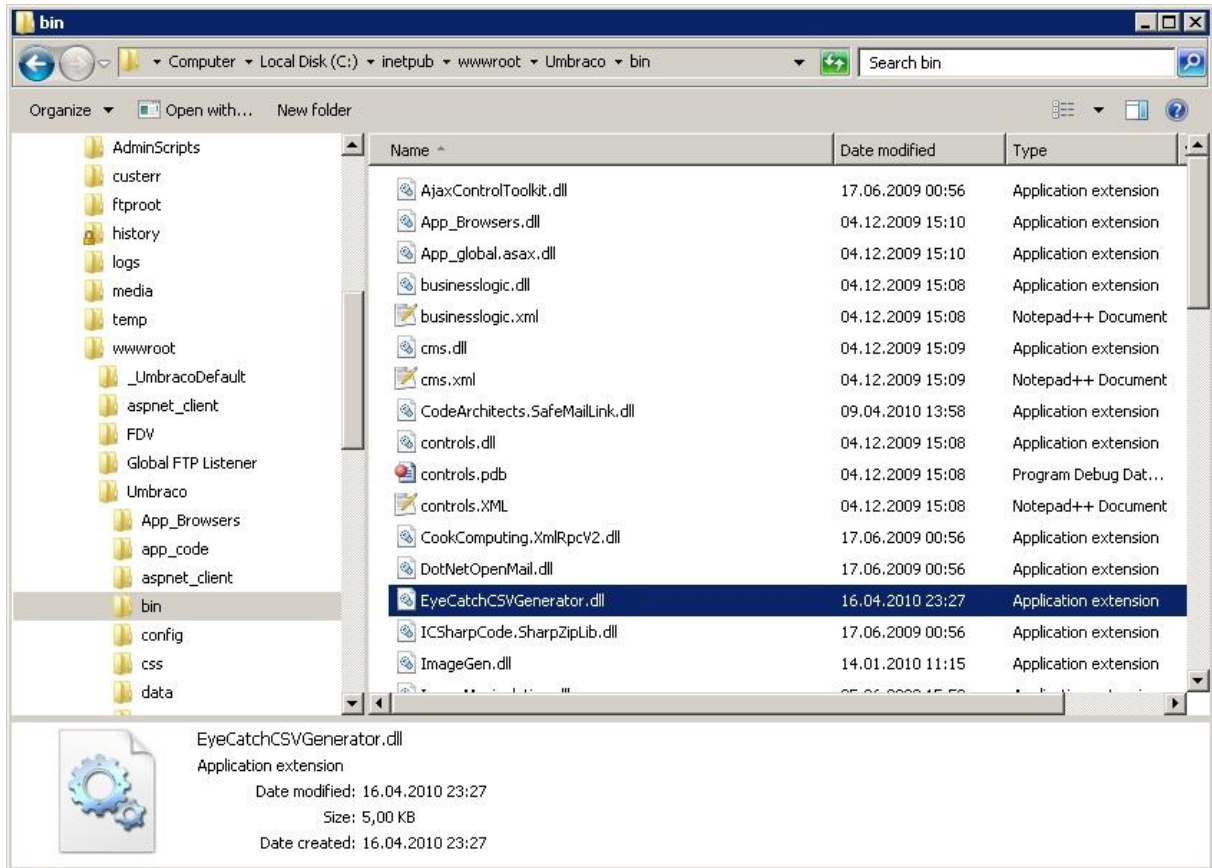
Now, create a folder under “usercontrols” in your Umbraco directory with a proper name.

Copy your .ascx file and your .ascx.cs into the newly created folder.



6

Then copy your .dll file from the bin folder on your PC to the bin folder on the server:



7

Now you need to create a macro to use on your template.

Create a macro (don't create parameters yet), and choose your .ascx file. Then click "Browse properties" and then "Save properties" in the modal window.

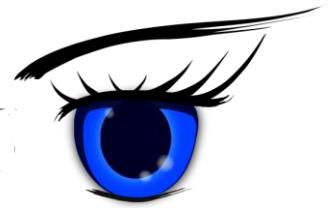
*Macro:*

Name	<input type="text" value="[EyeCatch]CSV Generator"/>
Alias	<input type="text" value="[EyeCatch]CSVGenerator"/>
Use XSLT file	<input type="text"/> Browse xslt files on server...
or .NET User Control	<input type="text" value="/usercontrols/CSVGenerator/EyeCatchCSV"/> Browse usercontrols on server... <input type="button" value="Browse properties"/>
or .NET Custom Control	<input type="text"/> (Assembly) <input type="text"/> (Type)
or python file	<input type="text"/> Browse python files on server...

20.04.2010

Org. No: NO 991 157 328 MVA  
Phone: +47 99 01 66 10

Address: Straume Teknologisenter  
v/ 3D Design Media  
Trollhaugmyra 15  
5353 Straume  
Norway



Now we need to create a template for testing our usercontrol and insert our macro in it.

*Template:*

```
<%@ Master Language="C#" MasterPageFile="/masterpages/default.master"
AutoEventWireup="true" %>

<asp:Content ContentPlaceHolderID="ContentPlaceHolderDefault" runat="server">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>EyeCatch CSV Generator Demo Page</title>
    </head>
    <body>
      <!-- Insert Macro Here! -->
    </body>
  </html>
</asp:Content>
```

Apply your template to one of your nodes, and try to view the page in a browser.

The result should be something like this:

**If everything works, the source node should show up below**

Name: My First Umbraco Usercontrol - Part 1: Setting up Visual Studio ID:1758

8

We have now established that the usercontrol works with Umbraco and are now ready to create the real functionality.



## Part 3: The Real Deal

Today we will create the CSV functionality. Since we have already set up a working usercontrol, the only thing we need to do is to change the code in Visual Studio, and upload the files again.

I will not go into detail about the code and why I chose to code it the way I did. Let the comments in the code speak for itself.

### EyeCatchCSVGenerator.ascx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="EyeCatchCSVGenerator.ascx.cs" Inherits="EyeCatchCSVGenerator.EyeCatchCSVGenerator" %>
```

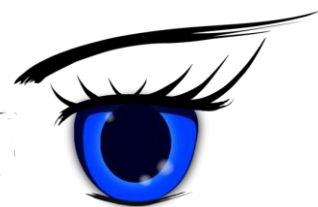
### EyeCatchCSVGenerator.ascx.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using umbraco.presentation.nodeFactory;

namespace EyeCatchCSVGenerator
{
    public partial class EyeCatchCSVGenerator : System.Web.UI.UserControl
    {
        public int Source { get; set; }
        public Boolean getChildrenOfSource { get; set; }
        public string documentTypeAlias { get; set; }
        public Boolean encodeHtmlTags { get; set; }
        public Boolean stripHtmlTags { get; set; }
        public string Delimiter { get; set; }

        /// <summary>
        /// Handles the Load event of the Page control.
        /// </summary>
        /// <param name="sender">The source of the event.</param>
        /// <param name="e">The <see cref="System.EventArgs"/> instance containing the
event data.</param>
        protected void Page_Load(object sender, EventArgs e)
        {
            // By default, set src to current page
            Node src = Node.GetCurrent();

            // Set a value for src if the parameterer was passed
            if(Source!=0)
            {
                src = new Node(Source);
            }
        }
    }
}
```



```

if(src.Id != 0)
{
    // Set a value for Delimiter if none exists
    if (string.IsNullOrEmpty(Delimiter))
    {
        Delimiter = ",";
    }

    // Set a value for documentTypeAlias if none exists
    if (string.IsNullOrEmpty(documentTypeAlias))
    {
        documentTypeAlias = src.NodeTypeAlias;
    }

    Response.Clear();
    Response.ContentType = "text/csv";
    Response.AddHeader("content-disposition", "attachment; filename=\"" +
src.Name + ".csv\"");
    Response.Buffer = true;

    // If getChildrenOfSource and documentTypeAlias is specified
    if (getChildrenOfSource && !documentTypeAlias.Equals(""))
    {
        // LINQ expression that extracts all nodes that matches the specified
documentTypeAlias
        List<Node> nodeCollection = (from Node n in src.Children where
n.NodeTypeAlias.Equals(documentTypeAlias) select n).ToList();

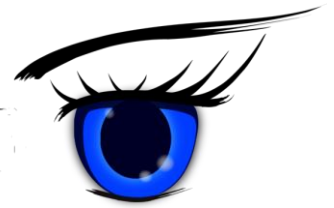
        // Print headers from first node
        WritePropertyHeaders((Node)nodeCollection[0]);

        // Print values for all nodes
        foreach (Node n in nodeCollection)
        {
            WritePropertyValues(n);
        }
    }

    // Default action
    else
    {
        WritePropertyHeaders(src);
        WritePropertyValues(src);
    }

    Response.End();
}
else
{
    throw new NullReferenceException("Something went wrong with the Source!
\nIs the referenced node published?");
}
}

```



```
///
/// Writes the property headers to page.
///
/// The node.
public void WritePropertyHeaders(Node elem)
{
    // Going through every property and lists the title
    foreach (Property p in elem.Properties)
    {
        Response.Write(p.Alias);
        Response.Write(Delimiter);
    }
}

/// <summary>
/// Writes the property headers to page.
/// </summary>
/// <param name="elem">The node.</param>
public void WritePropertyHeaders(Node elem)
{
    // Going through every property and lists the title
    foreach (Property p in elem.Properties)
    {
        Response.Write(p.Alias);
        Response.Write(Delimiter);
    }
}

/// <summary>
/// Writes the property values to page.
/// </summary>
/// <param name="elem">The node.</param>
public void WritePropertyValues(Node elem)
{
    Response.Write("\n");

    // Going through every property and lists the value
    foreach (Property p in elem.Properties)
    {
        string result = "";
        // Remove carriage returns (0x0D)
        result = p.Value.Replace("\r", " ");
        // Remove line feeds (0x0A)
        result = result.Replace("\n", string.Empty);

        // If macro is set to encode HTML tags
        if(encodeHtmlTags && !stripHtmlTags)
        {
            result = HttpUtility.HtmlEncode(result);
        }
    }
}
```



```

property) // If macro is set to remove HTML tags. (Ignores the encodeHtmlTags
    if(stripHtmlTags)
    {
        result = StripTagsCharArray(result);
    }

    Response.Write("\'" + result + "\'");
    Response.Write(Delimiter);
}

}

/// <summary>
/// Remove HTML tags from string using char array.
/// </summary>
/// <param name="source">The string containing HTML tags.</param>
public string StripTagsCharArray(string source)
{
    char[] array = new char[source.Length];
    int arrayIndex = 0;
    bool inside = false;

    for (int i = 0; i < source.Length; i++)
    {
        char let = source[i];
        if (let == '<')
        {
            inside = true;
            continue;
        }
        if (let == '>')
        {
            inside = false;
            continue;
        }
        if (!inside)
        {
            array[arrayIndex] = let;
            arrayIndex++;
        }
    }
    return new string(array, 0, arrayIndex);
}

}

}

```

Build the project and upload your .dll and .ascx files.



Since we have added some new parameters, we need to update the macro to reflect these changes. Go to your macro and click “Browse properties” again and then “Save properties”. You have now added the parameters from the code to your macro.

You also probably need to do some changes to your template. This is because it needs to be empty except for the CSV data this usercontrol produces.

Here is my template (remember to change the parameters to reflect your nodes):

*The best way is probably to remove the macro line below and re-add it.*

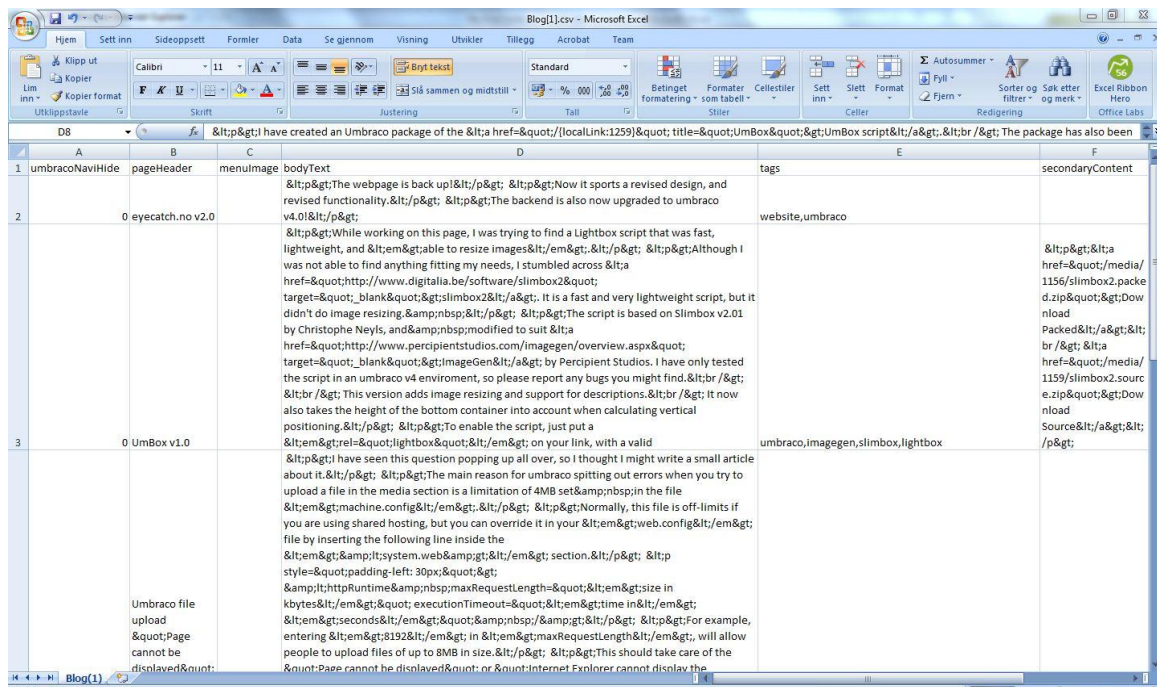
```
<%@ Master Language="C#" MasterPageFile="/masterpages/default.master"
AutoEventWireup="true" %>"><umbraco:Macro Source="1127" getChildrenOfSource="1"
documentTypeAlias="Article" stripHtmlTags="0" encodeHtmlTags="1" Delimiter=";"
Alias="[EyeCatch]CSVGenerator" runat="server"></umbraco:Macro>
```

**Note: There is no set standard for the delimiter character in CSV, so you can choose which one you want. Microsoft Excel uses the “;” character to delimit cells.**

When you have changed your template and added the macro, view the page from Part 2 in your browser. You should get a file download window, with your CSV data in it. Try to open it in Excel or a similar program and see if it can parse the file properly.

Mine looked like this:

13



If it doesn't work for you, please give me a notification.

20.04.2010

Org. No: NO 991 157 328 MVA  
Phone: +47 99 01 66 10

Address: Straume Teknologiserter  
v/ 3D Design Media  
Trollhaugmyra 15  
5353 Straume  
Norway



## Part 4: Now what?

I must admit, making usercontrols in Umbraco seemed like a much tougher task than it really was. For me, usercontrols were a mountain of trouble, but as it turned out it was relatively easy to do.

This last blog post is meant as a pointer to more information about creating Umbraco usercontrols.

And without further ado, here are some links to bring you further in the world of Umbraco:

### Umbraco Level 1/2 Courses

The [Umbraco courses](#) are the definitive resource on anything Umbraco, so I really recommend taking one (or all) of them.

### Umbraco programming in general:

- [API Documentation](#)
- [API Cheat Sheet](#)

### Specific subjects:

- [/Base](#)
- [404 Not found handlers](#)
- [Membership Providers](#)

### Blogs worth following:

- [Tim Geysens](#)
- [Dirk de Grave](#)
- [Lee Kelleher](#)
- [Hendy Racher](#)
- [Richard Soeterman](#)
- [Darren Ferguson](#)
- [Skiltz](#)
- [Cultiv](#)
- [Percipient Studios](#)