

# Export DataGrid in Excel

<http://escher07.altervista.org>

## Introduzione

Esportare il contenuto di un DataGrid in Excel è una cosa relativamente banale se nell'esportazione sono presenti solo campi stringa o interi. Si trovano in rete svariati esempi in proposito.

I problemi ci possono essere quando si devono esportare campi data o numeri con virgola, a causa di conflitti vari fra impostazioni internazionali di server e client (per aspetti come punti, virgole, formati dd/mm/yyyy o mm/dd/yyyy etc...).

Per risolvere questi problemi l'unica soluzione che mi ha funzionato è quella che impiega i CSS, con qualche elemento relativamente "oscuro" ovvero legato alle API (proprietarie) di Microsoft Excel, di cui si può avere qualche lume leggendo la documentazione Microsoft nota come Office Connect Kit di cui fra i link, qui di seguito.

## Meccanismo

La prima cosa necessaria da capire è che un DataGrid renderizzato (ovvero a livello di client) è niente di meno o niente di più che una <table> HTML, e come tale può essere formattata via CSS.

La seconda cosa è che Excel ha delle funzioni interne per importare delle <table> HTML. Di questo è facile rendersene conto utilizzando questa funzione dalla main toolbar: Dati/Da Web. Quello che succede è che viene richiesto un URL e nella corrispondente pagina vengono individuate tutte i possibili oggetti che possono contenere dati, fra cui ovviamente le tabelle. Per ciascuna di queste al momento in cui si decide di importarla Excel crea una "query". Questa query (lo possiamo vedere salvandola, cosa che di default verrà fatta in ../Dati\_Applicazioni/Microsoft/Queries) è un file di testo con estensione .iqy e sarà fatta ad esempio così:

```
WEB
1
http://your\_url (es. http://mywebsite.mycompany.com/index.aspx)

Selection=your\_object (es. DataGrid1)
Formatting=None
PreFormattedTextToColumns=True
ConsecutiveDelimitersAsOne=True
SingleBlockTextImport=False
DisableDateRecognition=False
DisableRedirections=False
```

Sintassi e significati completi si possono trovare come detto sul Microsoft Office Connectivity Kit.

Si capisce pertanto che per fare la nostra esportazione da programma dovremo:

1. dire che l'applicazione con cui processare questo codice HTML è Excel
2. imporgli la formattazione desiderata
3. generare il codice HTML corrispondente al datagrid

Excel farà il resto, ovvero lo processerà con le sue routines in pratica creandosi una query iqy generica runtime e processandovi l'HTML. Attenzione al fatto della genericità: è proprio perché in questa non ci sono indicazioni di formato che dobbiamo procedere via CSS.

Aggiungere il CSS: sì ma come? In pratica:

1. Scorrendosi tutte le celle del dataset
2. A ciascuna cella aggiungendo un attributo style=[nome\_dello\_stile\_voluto]
3. Aggiungendo nella response il codice CSS che descrive lo stile sopra referenziato

Si arriva quindi a questo codice:

```
public void ExportToExcel(ref DataGrid DataGridToExport, string strFileName)
{
    // scritto direttamente nella pagina HttpContext.Current Sparisce
    HttpContext.Current.Response.Clear();
    HttpContext.Current.Response.Buffer = true;
    HttpContext.Current.Response.ContentType = "application/vnd.xls";
    HttpContext.Current.Response.ContentEncoding =
System.Text.Encoding.UTF8;
    HttpContext.Current.Response.AddHeader("Content-Disposition",
"attachment; filename=" + strFileName);
    HttpContext.Current.Response.Charset = "";

    HttpContext.Current.Response.Cache.SetCacheability(System.Web.HttpCacheability.P
ublic);

    DataGridToExport.EnableViewState = false;
    System.IO.StringWriter stringWrite = new System.IO.StringWriter();
    System.Web.UI.HtmlTextWriter htmlWrite = new
HtmlTextWriter(stringWrite);
    string style = @"<style> .text { mso-number-format:\@; } </style> ";
    for (int i=0;i<DataGridToExport.Items.Count;i++)
    {
        for (int j=0;j<DataGridToExport.Items[i].Cells.Count;j++)
        {
            DataGridToExport.Items[i].Cells[j].Attributes.Add("class", "text");
        }
    }
    DataGridToExport.RenderControl(htmlWrite);
    HttpContext.Current.Response.Write(style);
    HttpContext.Current.Response.Write(htmlWrite.InnerWriter);
    HttpContext.Current.Response.End();
}
```

```
}
```

Questo codice nel caso specifico l'ho messo in una libreria posizionata sotto la App\_Code, ovvero un file .cs siffatto:

```
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Security.Permissions;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Drawing;
using System.IO;
using System.Globalization;
using System.Text;

namespace SRVControls
{
    [
       AspNetHostingPermission(SecurityAction.Demand,
            Level = AspNetHostingPermissionLevel.Minimal),
        AspNetHostingPermission(SecurityAction.InheritanceDemand,
            Level = AspNetHostingPermissionLevel.Minimal),
        ToolboxData("<{0}:SRVControls runat=\"server\"> </{0}:SRVControls>")
    ]

    public class SRVControlNoGUI : Control // non ha elementi visivi
    {

    }
}
```

Dove chiaramente la ExportToExcel è uno dei metodi di SRVControlsNoGUI. Ho fatto questa scelta perché questa routine viene usata in molti posti dell'applicazione. Se fosse servita solo per una maschera niente vietava di mettere il codice della ExportToExcel nella form.cs corrispondente.

## **Precisazioni**

Come si capisce in questo caso si usa una soluzione drastica: si esporta sempre tutto come se fosse un testo. In questo caso ad esempio 12,23 e 01/06/2009 verranno esportati come "12,23" e "01/06/2009" ovvero in modo visivamente corretto (e non stravolti tipo 12.23.00 o 06/01/2009), ma che necessita di qualche piccolo workaround se su quei dati ci dobbiamo fare dei calcoli (es. sostituzione di "," con "." e simili).

Un miglioramento si potrebbe ottenere sapendo cosa si vuole esportare. Ovvero: se la routine è in grado anche di andarsi a vedere di che tipo sono le varie colonne, può utilizzare un formato più "preciso" per le medesime. Ovvero l'attributo aggiunto (alla cella è, dopo il rendering al <td>) sarà ad esempio:

- class=text

- class=float
- class=date

a seconda se il dato è un testo (o intero senza virgola) oppure un reale, oppure una data, col gli style che potranno essere dichiarati così:

```
string style = @"<style> .text { mso-number-format:\@; } </style> ";
```

dove nella parte mso-number-format possiamo utilizzare sintassi di questo tipo:

caso	Esempio di testo fra {}	Nota
testo	mso-number-format:\@	
numero reale	mso-number-format:"0\,000"	3 decimali
numero reale	mso-number-format:\#\, \#\#0\,000	Virgola come separatore e 3 decimali
data	mso-number-format:"mm\dd\yy"	Data, mese/giorno/anno a due cifre
data	mso-number-format:"d\\-mmm\\-yyyy"	Altro formato data
percentuale	mso-number-format:Percent	

dove mso sta per "Microsoft Office" e dove quelli nominati sono formati standard "comprensibili" alle API di Office, ovvero attraverso i quali – se usati in un codice HTML/CSS oggetto di importazione – Excel formatta le sue celle come gli viene detto programmaticamente.

**Links**

<http://agoric.com/sources/software/htmltoExcel>

<http://www.15seconds.com/issue/991021.htm>

<http://www.microsoft.com/downloads/details.aspx?FamilyID=221CE325-9D73-48C1-9081-034E4931BC87&displaylang=EN>