GemBox Support Center

Export GridView and/or DataGrid control to Excel file

Mario - GemBox - 2017-02-10 - 0 Comments - in GemBox.Spreadsheet

Currently GemBox.Spreadsheet lacks an API support for direct exporting GridView and/or DataGrid controls to an Excel file.

However, we do have an API for importing and exporting a DataTable to an Excel file which can be used to export GridView's or DataGrid's DataSource property. The following link is a demonstration sample of InsertDataTable method usage:

https://www.gemboxsoftware.com/spreadsheet/examples/c-sharp-export-datatable-to-excel/501

With the above approach we lose style and formatting that was used on the control, so as an alternative what we can do is use the control's HTML representation and convert it to an Excel file.

For example, here is a sample "Default.aspx" which contains GridView and Button controls:

Now, in order to export the GridView content to an excel file on the "Export GridView" button's click event, we can do the following:

C# code

```csharp
public partial class _Default : Page
{
    protected void btnGridViewExport_Click(object sender, EventArgs e)
    {
        StringWriter sw = new StringWriter();
        HtmlTextWriter htw = new HtmlTextWriter(sw);
        GridView1.RenderControl(htw);

        ExcelFile excel;
        var options = LoadOptions.HtmlDefault;
```

```csharp
        using (var htmlStream = new MemoryStream(options.Encoding.GetBytes(sw.ToString()
)))
            excel = ExcelFile.Load(htmlStream, options);

        excel.Save(this.Response, "GridView.xlsx");
    }

    // Note, it is important to override the form verification in order to successfully call a Grid
View1.RenderControl method.
    public override void VerifyRenderingInServerForm(Control control)
    {

    }
}
```

VB.NET code

```vbnet
Public Class _Default
    Inherits Page

    Protected Sub btnGridViewExport_Click(sender As Object, e As EventArgs) Handles btnGri
dViewExport.Click
        Dim sw As New StringWriter()
        Dim htw As New HtmlTextWriter(sw)
        GridView1.RenderControl(htw)

        Dim excel As ExcelFile
        Dim options = LoadOptions.HtmlDefault
        Using htmlStream = New MemoryStream(options.Encoding.GetBytes(sw.ToString()))
            excel = ExcelFile.Load(htmlStream, options)
        End Using

        excel.Save(Me.Response, "GridView.xlsx")
    End Sub

    ' Note, it is important to override the form verification in order to successfully call a GridV
iew1.RenderControl method.
    Public Overrides Sub VerifyRenderingInServerForm(control As Control)

    End Sub

End Class
```

And here is the resulting "GridView.xlsx" file:

	A	B	C	D	E	F	G	H	I
1	COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5	COLUMN 6	COLUMN 7	COLUMN 8	
2	Cell[1, 1]	Cell[2, 1]	Cell[3, 1]	Cell[4, 1]	Cell[5, 1]	Cell[6, 1]	Cell[7, 1]	Cell[8, 1]	
3	Cell[1, 2]	Cell[2, 2]	Cell[3, 2]	Cell[4, 2]	Cell[5, 2]	Cell[6, 2]	Cell[7, 2]	Cell[8, 2]	
4	Cell[1, 3]	Cell[2, 3]	Cell[3, 3]	Cell[4, 3]	Cell[5, 3]	Cell[6, 3]	Cell[7, 3]	Cell[8, 3]	
5	Cell[1, 4]	Cell[2, 4]	Cell[3, 4]	Cell[4, 4]	Cell[5, 4]	Cell[6, 4]	Cell[7, 4]	Cell[8, 4]	
6	Cell[1, 5]	Cell[2, 5]	Cell[3, 5]	Cell[4, 5]	Cell[5, 5]	Cell[6, 5]	Cell[7, 5]	Cell[8, 5]	
7	Cell[1, 6]	Cell[2, 6]	Cell[3, 6]	Cell[4, 6]	Cell[5, 6]	Cell[6, 6]	Cell[7, 6]	Cell[8, 6]	
8	Cell[1, 7]	Cell[2, 7]	Cell[3, 7]	Cell[4, 7]	Cell[5, 7]	Cell[6, 7]	Cell[7, 7]	Cell[8, 7]	
9	Cell[1, 8]	Cell[2, 8]	Cell[3, 8]	Cell[4, 8]	Cell[5, 8]	Cell[6, 8]	Cell[7, 8]	Cell[8, 8]	
10									
11									