

www.bytescout.com

How to display xls as html with spreadsheet sdk in ASP.NET C# with ByteScout Barcode Suite

Continuous learning is a crucial part of computer science and this tutorial shows how to display xls as html with spreadsheet sdk in ASP.NET C#

The coding instructions are formulated to help you to try-out the features without the requirement to write your own code. ByteScout Barcode Suite can display xls as html with spreadsheet sdk. It can be applied from ASP.NET C#. ByteScout Barcode Suite is the set that includes three different SDK products to generate barcodes, read barcodes and read and write spreadsheets: Barcode SDK, Barcode Reader SDK and Spreadsheet SDK.

Want to save time? You will save a lot of time on writing and testing code as you may just take the ASP.NET C# code from ByteScout Barcode Suite for display xls as html with spreadsheet sdk below and use it in your application. If you want to implement the functionality, just copy and paste this code for ASP.NET C# below into your code editor with your app, compile and run your application. Want to see how it works with your data then code testing will allow the function to be tested and work properly.

If you want to try other source code samples then the free trial version of ByteScout Barcode Suite is available for download from our website. Just try other source code samples for ASP.NET C#.

FOR MORE INFORMATION AND FREE TRIAL:

[Download Free Trial SDK \(on-premise version\)](#)

[Read more about ByteScout Barcode Suite](#)

[Explore API Documentation](#)

[Get Free Training for ByteScout Barcode Suite](#)

[Get Free API key for Web API](#)

[visit www.ByteScout.com](#)

Source Code Files:

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="DisplayXlsAsHtml.Default"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Width="311px"></asp:Label><br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="Select worksheet to display: "></asp:Label>
            <asp:DropDownList ID="DropDownList1" runat="server"
                Width="233px">
            </asp:DropDownList>
            <asp:Button ID="ButtonGo" runat="server" Text="Go" Width="45px" OnClick="ButtonGo_Click" />
        </form>
    </body>
</html>
```

Default.aspx.cs

```
using System;
using Bytescout.Spreadsheet;

namespace DisplayXlsAsHtml
{
    public partial class _Default : System.Web.UI.Page
    {

        /*
        IF YOU SEE TEMPORARY FOLDER ACCESS ERRORS:

        Temporary folder access is required for web application when you use Bytescout Spreadsheet API.
        If you are getting errors related to the access to temporary folder like "Temporary folder access is denied" or "Temporary folder does not exist", please make sure that your web application has sufficient permissions to access temporary folder.

        SOLUTION:

        If your IIS Application Pool has "Load User Profile" option enabled then it will have sufficient permissions to access temporary folder.

        If you are running Web Application under an impersonated account or IIS AppPool, then you need to grant necessary permissions to the temporary folder.
        */

    }
}
```

```

In this case
- check the User or User Group your web application is running under
- then add permissions for this User or User Group to read and write in
- restart your web application and try again

*/



Spreadsheet _document = null;

protected void Page_Load(object sender, EventArgs e)
{
    String inputXlsFile = Server.MapPath("example.xls");

    // Open spreadsheet
    _document = new Spreadsheet();
    _document.LoadFromFile(inputXlsFile);

    Label1.Text = "\"Example.xls\" loaded";

    for (int i = 0; i < _document.Worksheets.Count ; i++)
    {
        DropDownList1.Items.Add(_document.Worksheets[i].Name);
    }
}

protected void ButtonGo_Click(object sender, EventArgs e)
{
    String sheet = DropDownList1.SelectedItem.Text;

    if (!String.IsNullOrEmpty(sheet))
    {
        // Clear HTTP output
        Response.Clear();
        // Set the content type to HTML
        Response.ContentType = "text/HTML";
        // Save selected worksheet to output stream as HTML
        _document.Worksheets[sheet].SaveAsHTML(Response.OutputStream);

        Response.End();
    }
}
}

```

Default.aspx.designer.cs

```

//-----
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:2.0.50727.4927

```

```
//  
// Changes to this file may cause incorrect behavior and will be lost if  
// the code is regenerated.  
// </auto-generated>  
//-----  
  
namespace DisplayXlsAsHtml {  
  
    /// <summary>  
    /// _Default class.  
    /// </summary>  
    /// <remarks>  
    /// Auto-generated class.  
    /// </remarks>  
    public partial class _Default {  
  
        /// <summary>  
        /// form1 control.  
        /// </summary>  
        /// <remarks>  
        /// Auto-generated field.  
        /// To modify move field declaration from designer file to code-behind file.  
        /// </remarks>  
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;  
  
        /// <summary>  
        /// Label1 control.  
        /// </summary>  
        /// <remarks>  
        /// Auto-generated field.  
        /// To modify move field declaration from designer file to code-behind file.  
        /// </remarks>  
        protected global::System.Web.UI.WebControls.Label Label1;  
  
        /// <summary>  
        /// Label2 control.  
        /// </summary>  
        /// <remarks>  
        /// Auto-generated field.  
        /// To modify move field declaration from designer file to code-behind file.  
        /// </remarks>  
        protected global::System.Web.UI.WebControls.Label Label2;  
  
        /// <summary>  
        /// DropDownList1 control.  
        /// </summary>  
        /// <remarks>  
        /// Auto-generated field.  
        /// To modify move field declaration from designer file to code-behind file.  
        /// </remarks>  
        protected global::System.Web.UI.WebControls.DropDownList DropDownList1;  
  
        /// <summary>  
        /// ButtonGo control.  
        /// </summary>  
        /// <remarks>  
        /// Auto-generated field.  
        /// To modify move field declaration from designer file to code-behind file.  
        /// </remarks>
```

```
    protected global::System.Web.UI.WebControls.Button ButtonGo;
}

}
```

DisplayXlsAsHtml.sln

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 2013
VisualStudioVersion = 12.0.40629.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "DisplayXlsAsHtml", "DisplayXlsAsHtml"
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Any CPU = Debug|Any CPU
        Release|Any CPU = Release|Any CPU
    EndGlobalSection
    GlobalSection(ProjectConfigurationPlatforms) = postSolution
        {2B34C366-1868-492C-AF61-F47FCC7BCE2C}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
        {2B34C366-1868-492C-AF61-F47FCC7BCE2C}.Debug|Any CPU.Build.0 = Debug|Any CPU
        {2B34C366-1868-492C-AF61-F47FCC7BCE2C}.Release|Any CPU.ActiveCfg = Release|Any CPU
        {2B34C366-1868-492C-AF61-F47FCC7BCE2C}.Release|Any CPU.Build.0 = Release|Any CPU
    EndGlobalSection
    GlobalSection(SolutionProperties) = preSolution
        HideSolutionNode = FALSE
    EndGlobalSection
EndGlobal
```

Web.config

```
<?xml version="1.0"?>

<configuration>

    <appSettings/>
    <connectionStrings/>

    <system.web>
        <!--
```

```
Set compilation debug="true" to insert debugging
symbols into the compiled page. Because this
affects performance, set this value to true only
during development.

-->
<compilation debug="true" />
<!--
The <authentication> section enables configuration
of the security authentication mode used by
ASP.NET to identify an incoming user.
-->
<authentication mode="Windows" />
<!--
The <customErrors> section enables configuration
of what to do if/when an unhandled error occurs
during the execution of a request. Specifically,
it enables developers to configure html error pages
to be displayed in place of a error stack trace.

<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
    <error statusCode="403" redirect="NoAccess.htm" />
    <error statusCode="404" redirect="NotFound.htm" />
</customErrors>
-->
</system.web>
</configuration>
```

VIDEO

<https://www.youtube.com/watch?v=NEwNs2b9YN8>

ON-PREMISE OFFLINE SDK

[60 Day Free Trial](#) or [Visit ByteScout Barcode Suite Home Page](#)
[Explore ByteScout Barcode Suite Documentation](#)
[Explore Samples](#)
[Sign Up for ByteScout Barcode Suite Online Training](#)

ON-DEMAND REST WEB API

[Get Your API Key](#)
[Explore Web API Docs](#)
[Explore Web API Samples](#)

[visit www.ByteScout.com](http://www.ByteScout.com)

[visit www.PDF.co](http://www.PDF.co)

www.bytescout.com