Dot Net Library User Documentation

Author: Alan Savage

Date: 23rd July 2008

Contents

Bulk SMS Library

To use this library, you will need to add a reference to the ***BulkSmsLibrary.dll*** in your project.

Add a using (c#)/imports (vb) to Savage.DotNetLibrary.BulkSmsLibrary

C#:

```
using Savage.DotNetLibrary.DatabaseLibrary;
```

VB:

```
Imports Savage.DotNetLibrary.DatabaseLibrary
```

Notes:

- The user must have an account setup with Bulk SMS and have some credit in their account (I think you get one free text message when you register an account). Visit http://www.bulksms.co.uk/ for more details and to register your account.

How to send SMS Text Messages to mobile devices

Sends a text message to a mobile device.

Examples:

The following example sends a text message to a mobile device, the mobile number in the demonstration is a UK mobile and would normally be referred to as 07912345678, so the leading zero is dropped and replace with the international dialling code for the UK (44).

In both examples the console will display IN_PROGRESS if the text message was successfully received and processed by Bulk SMS and queued for sending (normally received within 5 seconds).

C#:

```
SendSmsInfo sms = new SendSmsInfo();
sms.UserName = "userName";
sms.Password = "secrets";
sms.Recipients.Add("447912345678");
sms.Message = "Test message from C# application";

SendSmsStatus status = BulkSmsHttpApi.SendSms(sms);
Console.Out.WriteLine(status.Description);
```

VB:

```
Dim sms As SendSmsInfo = New SendSmsInfo()
sms.UserName = "userName"
sms.Password = " secrets”
sms.Recipients.Add("447912345678")
sms.Message = "Test message from VB application"

Dim status As SendSmsStatus = BulkSmsHttpApi.SendSms(sms)
Console.Out.WriteLine(status.Description)
```

SendSmsInfo Properties

Required Properties:

Property	Type	Description
UserName	String	Your Bulk SMS user name.
Password	String	Your Bulk SMS password.
Recipients*	List<String>	A collection of phone numbers to send the message to.
GroupIds*	List<int>	A collection of group id's to send the message to.
Message	String	The text message that you wish to send.

*Either Recipients or GroupIds properties must contain values.

Optional Properties:

Property	Type	Description
Sender	String	Change the sender's address (available after credits have been purchased. You need to request a sender id from the User Account webpage).
MessageClass	MessageClassOption	Flash, Normal or Default
DataCodingAlphabet	DataCodingAlphabetOption	Dca16Bit, Dca8Bit, Dca7Bit, Default
WantReport	Boolean	True or False if you want a delivery report.
RoutingGroup	RoutingGroupOption	Default, Economy, Premium, Standard
SourceId	String	Your own string (maybe for account purposes).
Repliable	Boolean	Can receipents reply to your message?
StripDuplicateRecipients	Boolean	Removes any duplicate recipients from this instance. Bulk SMS recommend you do not rely on this but validate this.
StopDupId	Integer	Used to prevent duplicate sending, set a unique Integer value.
SendTime	DateTime	The date and time when the message should be sent. If not supplied then message is sent straight away.
SchedulingDescription	String	Provide a description of the scheduling for your own use.
AllowConcatenation	Boolean	Enable sending messages longer than 160 characters.
MaxConcatenatedMessageParts	Int	Maximum number of messages to use when concatenating messages.

SendSmsStatus Properties

BatchId: Type: Integer. This contains the ID for the sending so you can track this.

Description: Type: String. A text description that details the status returned from calling the Send method of the BulkSmsHttpApi.

Status Code: Type Integer. See list below for codes.

Possible values for status_code are:

0: In progress (a normal message submission, with no error encountered so far).
1: Scheduled (see *Scheduling* below).
22: Internal fatal error
23: Authentication failure
24: Data validation failed
25: You do not have sufficient credits
26: Upstream credits not available
27: You have exceeded your daily quota
28: Upstream quota exceeded
40: Temporarily unavailable

You should never depend on the value of status_description - only depend on status_code, which is a constant. However, status_description can contain useful information about the nature of failures when you are developing your initial application. batch_id is guaranteed to be a positive integer > 0, if present.
Note: you should attempt to resend if you receive status code 40. You *could* also do so if you receive 26 or 28, which *might* have been resolved after several resend attempts. All other errors should be considered fatal.
Any **HTTP status code** other than 200 should be considered transient, i.e. you should attempt to resend after some interval.

Please see http://www.bulksms.co.uk/docs/eapi/submission/send_sms/ for more detailed information on the properties.

Database Library

To use this library, you will need to add a reference to the ***DatabaseLibrary.dll*** in your project.

Add a using (c#)/imports (vb) to Savage.DotNetLibrary.DatabaseLibrary

C#:

```
using Savage.DotNetLibrary.DatabaseLibrary;
```

VB:

```
Imports Savage.DotNetLibrary.DatabaseLibrary
```

Notes:

- The user must have the rights to perform the relevant action on the SQL Server.
  CreateDatabase
  Requires CREATE DATABASE, CREATE ANY DATABASE, or ALTER ANY DATABASE permission.

  DropDatabase
  To execute DROP DATABASE, at a minimum, a user must have CONTROL permission on the database.

  GetDatabases
  User must have SELECT permissions on the view sys.sysdatabases

- A Database is not required to be specified in the connection string.
- If the method could not be completed due to an exception then the exception is thrown and the client will have to handle this.

How to Create a Database

Creates a database with the specified name on the SQL server specified in the connection string. (NOTE: The user must have the rights to create databases on the SQL Server).

Examples:

The following examples demonstrate how to create a database named **MyDatabase** on the instance of SQL Server on the local machine named SQLEXPRESS.

C#:

```
string connectionString = @"Server=.\SQLEXPRESS;Trusted_Connection=true;";
DatabaseManager dm = new DatabaseManager(connectionString);

dm.SqlServerInstance.CreateDatabase("MyDatabase");
```

VB:

```
Dim connectionString As String = "Server=.\SQLExpress;Trusted_Connection=true;"
Dim dm As DatabaseManager = New DatabaseManager(connectionString)

dm.SqlServerInstance.CreateDatabase("MyDatabase")
```

How to List the Names of all the Databases on a SQL Server

Returns a list of the database names that have been created on the server.

Examples:

The following examples demonstrate how to display a list of the database names on an instance of SQL Server:

C#:

```
string connectionString = @"Server=.\SQLEXPRESS;Trusted_Connection=true;";
DatabaseManager dm = new DatabaseManager(connectionString);

string[] databases = dm.SqlServerInstance.GetDatabases();

foreach (string db in databases)
{
	Console.Out.WriteLine(db);
}
```

VB:

```
Dim connectionString As String = "Server=.\SQLExpress;Trusted_Connection=true;"
Dim dm As DatabaseManager = New DatabaseManager(connectionString)

Dim databases As String() = dm.SqlServerInstance.GetDatabases()

For Each db As String In databases
	Console.Out.WriteLine(db)
Next
```

How to Delete a Database

Deletes the database with the specified name on the SQL server specified in the connection string.

Warning!

Use this method with care, once a database is deleted the only way to restore the database is from backups (you do have backups, right?)

Example:

The following examples demonstrate how to delete a database named MyDatabase on the instance of SQLEXPRESS installed on the local machine:

C#:

```
string connectionString = @"Server=.\SQLEXPRESS;Trusted_Connection=true;";
DatabaseManager dm = new DatabaseManager(connectionString);

dm.SqlServerInstance.DropDatabase("MyDatabase");
```

VB:

```
Dim connectionString As String = "Server=.\SQLExpress;Trusted_Connection=true;"
Dim dm As DatabaseManager = New DatabaseManager(connectionString)

dm.SqlServerInstance.DropDatabase("MyDatabase")
```

Image Library

To use this library, you will need to add a reference to the ***ImageLibrary.dll*** in your project.

Add a using (c#)/imports (vb) to Savage.DotNetLibrary.ImageLibrary

C#:

```
using Savage.DotNetLibrary.ImageLibrary;
```

VB:

```
Imports Savage.DotNetLibrary.ImageLibrary
```

Notes:

- The client must have access to the file to thumbnail and have write access to the directory if saving the thumbnail.
- If the CreateThumbnail method is called without specifying a value for the InterpolationMode parameter the InterpolationMode.Default is used.
- If the method could not be completed due to an exception then the exception is thrown and the client will have to handle this.
- Not all Exif properties will contain a value, this will depend on the capabilities of the device that captured the image.

How to Create a Thumbnail of a Picture

To thumbnail a picture you use the CreateThumbnail method in the ImageUtility class. The method has several overloaded methods depending on what you require.

The image produced is scaled correctly to avoid making the thumbnail look distorted. For example, if the original image has a size of (2032 pixels in width and 1524 pixels in height (2032x1524)) and we call one of the CreateThumbnail methods in the utility class and provide the value 100 for the width and height properties, then the resulting thumbnail will be 100 pixels in width and 75 pixels in height. This is calculated using following method:

The width of 2032 pixels is larger than the height (1524) so CreateThumbnail calculates (because the width has more pixels than the height), the ratio that the width needs to shrink by doing 2032 divide 100 which is 20.32. CreateThumbnail then divides the height (1524 pixels) by this same ratio value, 20.32 to determine the height of the thumbnail, so 1524/20.32 is 75.

CreateThumbnail(sourceFilePath, width, height)

This method overload creates a thumbnail of the image file at *sourceFilePath* and the thumbnail will be scaled down to have a maximum width and height specified in the *width* and *height* properties. The method returns a System.Drawing.Bitmap object that contains the thumbnail image.

Example:

The following examples demonstrate how to create a thumbnail of an image named **london.jpg** with a maximum width of 100 and a maximum height of 100.

C#:

```
Bitmap thumbnail = ImageUtility.CreateThumbnail("london.jpg", 100, 100);
```

VB:

```
Dim thumbnail As Bitmap = ImageUtility.CreateThumbnail("london.jpg", 100, 100)
```

CreateThumbnail(sourceFilePath, width, height, interpolationMode)

This method overload creates a thumbnail of the image file at *sourceFilePath* and the thumbnail will be scaled down to have a maximum width and height specified in the *width* and *height* properties. The interpolation mode determines the quality of the thumbnail that is produced. The method returns a System.Drawing.Bitmap object that contains the thumbnail image.

Example:

The following examples demonstrate how to create a high quality thumbnail of an image named **london.jpg** with a maximum width of 250 and a maximum height of 250.

C#:

```
Bitmap thumbnail = ImageUtility.CreateThumbnail("london.jpg", 250, 250, System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic);
```

VB:

```
Dim thumbnail As Bitmap = ImageUtility.CreateThumbnail("london.jpg", 250, 250, Drawing2D.InterpolationMode.HighQualityBicubic)
```

CreateThumbnail(sourceFilePath, thumbnailFilePath, width, height)

This method overload creates a thumbnail of the image file at *sourceFilePath* and the thumbnail will be scaled down to have a maximum width and height specified in the *width* and *height* properties. The thumbnail will be saved as *thumbnailFilePath*. This method does not return anything.

Example:

The following examples demonstrate how to create a thumbnail of an image named **london.jpg** with a maximum width of 100 and a maximum height of 100 and save the thumbnail as **myThumbnail.jpg**.

C#:

```
ImageUtility.CreateThumbnail("london.jpg", "myThumbnail.jpg", 100, 100);
```

VB:

```
ImageUtility.CreateThumbnail("london.jpg", "myThumbnail.jpg", 100, 100)
```

CreateThumbnail(sourceFilePath, thumbnailFilePath, width, height, interpolationMode)

This method overload creates a thumbnail of the image file at *sourceFilePath* and the thumbnail will be scaled down to have a maximum width and height specified in the *width* and *height* properties. The interpolation mode determines the quality of the thumbnail that is produced. The thumbnail will be saved as *thumbnailFilePath*. This method does not return anything.

Example:

The following examples demonstrate how to create a high quality thumbnail of an image named **london.jpg** with a maximum width of 250 and a maximum height of 250 and save the thumbnail as **myThumbnail.jpg**.

C#:

```
ImageUtility.CreateThumbnail("london.jpg", "myThumbnail.jpg", 250, 250,
System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic);
```

VB:

```
ImageUtility.CreateThumbnail("london.jpg", "myThumbnail.jpg", 250, 250,
Drawing2D.InterpolationMode.HighQualityBicubic)
```

See the section **InterpolationMode enumeration** in the appendix for more details.

How to View Exif Tags for a Picture

Each time a photo is taken a multitude of data is also embedded into the photograph and this data can be very useful to applications, such as the date and time the image was taken.

Examples:

The following examples demonstrate how to display the date and time that the photograph was taken.

C#:

```
Bitmap photo = new Bitmap("london.jpg");
Exif exif = new Exif(photo);
Console.Out.WriteLine(exif.DateTimeOriginal.Value.ToString());
```

VB:

```
Dim photo As Bitmap = New Bitmap("london.jpg")
Dim exif As Exif = New Exif(photo)
Console.Out.WriteLine(exif.DateTimeOriginal.Value.ToString())
```

TIFF Tags

ImageWidth The number of columns of image data, equal to the number of pixels per row. In JPEG compressed data a JPEG marker is used instead of this tag.

ImageHeight The number of rows of image data. In JPEG compressed data a JPEG marker is used instead of this tag.

BitsPerSample The number of bits per image component. In this standard each component of the image is 8 bits, so the value for this tag is 8. See also SamplesPerPixel. In JPEG compressed data a JPEG marker is used instead of this tag.

Compression The compression scheme used for the image data. When a primary image is JPEG compressed, this designation is not necessary and is omitted.

PhotometricInterpolation The pixel composition. In JPEG compressed data a JPEG marker is used instead of this tag.

Orientation The image orientation viewed in terms of rows and columns. Returns the position of row 0 and column 0.

SamplesPerPixel The number of components per pixel. Since this standard applies to RGB and YCbCr images, the value set for this tag is 3. In JPEG compressed data a JPEG marker is used instead of this tag.

PlanarConfiguration Indicates whether pixel components are recorded in chunky or planar format. In JPEG compressed files a JPEG marker is used instead of this tag. If this field does not exist, the TIFF default of 1 (chunky) is assumed.

YCbCrSubSampling The sampling ratio of chrominance components in relation to the luminance component. In JPEG compressed data a JPEG marker is used instead of this tag.

YCbCrPositioning The position of chrominance components in relation to the luminance component.

XResolution The number of pixels per ResolutionUnit in the ImageWidth direction. When the image resolution is unknown, 72 [dpi] is designated.

YResolution The number of pixels per ResolutionUnit in the ImageLength direction. The same value as XResolution is designated.

ResolutionUnit The unit for measuring XResolution and YResolution (inches or centimetres). The same unit is used for both XResolution and YResolution.

StripOffsets For each strip, the byte offset of that strip. It is recommended that this be selected so the number of strip bytes does not exceed 64 Kbytes. With JPEG compressed data this designation is not needed and is omitted. See also RowsPerStrip and StripByteCounts.

RowsPerStrip The number of rows per strip. This is the number of rows in the image of one strip when an image is divided into strips. With JPEG compressed data this designation is not needed and is omitted. See also RowsPerStrip and StripByteCounts.

StripByteCounts The total number of bytes in each strip. With JPEG compressed data this designation is not needed and is omitted.

JpegInterchangeFormat The offset to the start byte (SOI) of JPEG compressed thumbnail data. This is not used for primary image JPEG data.

JpegInterchangeFormatLength The number of bytes of JPEG compressed thumbnail data. This is not used for primary image JPEG data.

FileChangeDateTime The date and time of image creation. In this standard it is the date and time the file was changed.

ImageDescription A character string giving the title of the image. It may be a comment such as "1988 company picnic" or the like.

Make The manufacturer of the recording equipment. This is the manufacturer of the DSC, scanner, video digitizer or other equipment that generated the image. When the field is left blank, it is treated as unknown.

Model The model name or model number of the equipment. This is the model name of number of the DSC, scanner, video digitizer or other equipment that generated the image. When the field is left blank, it is treated as unknown.

Software This tag records the name and version of the software or firmware of the camera or image input device used to generate the image. When the field is left blank, it is treated as unknown.

Artist This tag records the name of the camera owner, photographer or image creator. When the field is left blank, it is treated as unknown.

Copyright Copyright information. In this standard the tag is used to indicate both the photographer and editor copyrights. It is the copyright notice of the person or organization claiming rights to the image.

Exif Tags

ExifVersion The version of this standard supported. Nonexistence of this field is taken to mean nonconformance to the standard. Conformance to this standard is indicated by recording "0220" as 4-byte ASCII.

FlashPixVersion The Flashpix format version supported by a FPXR file. If the FPXR function supports Flashpix format Ver. 1.0, this is indicated similarly to ExifVersion by recording "0100" as 4-byte ASCII.

ColorSpace The color space information tag (ColorSpace) is always recorded as the color space specifier.

PixelXDimension Information specific to compressed data. When a compressed file is recorded, the valid width of the meaningful image shall be recorded in this tag, whether or not there is padding data or a restart marker. This tag should not exist in an uncompressed file.

PixelYDimension Information specific to compressed data. When a compressed file is recorded, the valid height of the meaningful image shall be recorded in this tag, whether or not there is padding data or a restart marker. This tag should not exist in an uncompressed file. Since data padding is unnecessary in the vertical direction, the number of lines recorded in this valid image height tag will in fact be the same as that recorded in the SOF.

MakerNote A tag for manufacturers of Exif writers to record any desired information. The contents are up to the manufacturer, but this tag should not be used for any other than its intended purpose.

UserComment A tag for Exif users to write keywords or comments on the image besides those in ImageDescription, and without the character code limitations of the ImageDescription tag.

DateTimeOriginal The date and time when the original image data was generated. For a DSC the date and time the picture was taken are recorded.

DateTimeDigitized The date and time when the image was stored as digital data. If, for example, an image was captured by DSC and at the same time the file was recorded, then the DateTimeOriginal and DateTimeDigitized will have the same contents.

SubsecTime A tag used to record fractions of seconds for the DateTime tag.

SubsecOriginal A tag used to record fractions of seconds for the DateTimeOriginal tag.

SubsecDigitized A tag used to record fractions of seconds for the DateTimeDigitized tag.

ExposureProgram The class of the program used by the camera to set exposure when the picture is taken.

ImageUniqueId This tag indicates an identifier assigned uniquely to each image. It is recorded as an ASCII string equivalent to hexadecimal notation and 128-bit fixed length.

GPS Tags

VersionId Indicates the version of GPSInfoIFD. The version is given as 2.2.0.0. This tag is mandatory when GPSInfo tag is present.

Latitude Indicates the latitude.

Longitude Indicates the longitude.

AltitudeRef Indicates the altitude used as the reference altitude. If the altitude is below sea level, the altitude is indicated as an absolute value in the GPSAltitude tag.

Altitude Indicates the altitude based on the reference in GPSAltitudeRef. The reference unit is meters.

UtcDateTimeStamp Indicates the date and time as UTC (Coordinated Universal Time).

Satellites Indicates the GPS satellites used for measurements. This tag can be used to describe the number of satellites, their ID number, angle of elevation, azimuth, SNR and other information.

Status Indicates the status of the GPS receiver when the image is recorded.

MeasureMode Indicates the GPS measurement mode.

MapDatum Indicates the geodetic survey data used by the GPS receiver. If the survey data is restricted to Japan, the value of this tag is 'TOKYO' or 'WGS-84'.

Protocol Library

To use this library, you will need to add a reference to the ***ProtocolLibrary.dll*** in your project.

Add a using (c#)/imports (vb) to Savage.DotNetLibrary.ProtocolLibrary

C#:

```
using Savage.DotNetLibrary.ProtocolLibrary;
```

VB:

```
Imports Savage.DotNetLibrary.ProtocolLibrary
```

Notes:

- The client must have access to the uri specified.
- If the method could not be completed due to an exception then the exception is thrown and the client will have to handle this (such as the uri being unavailable).
- Each instance of the HttpWrapper class exposes a property named WebRequest and this allows the client to access the WebRequest object and edit the values of any properties before carrying out a GET or POST request.

How to perform a HTTP GET request

When you browse to a web page by entering the web address into your browser your browser performs a GET request, this request is normally dealt with by the web server sending the client a document. This document will likely contain HTML markup, that our browsers then use to render into content and display in our browser window. The **HttpWrapper** in the **ProtocolLibrary.dll** allows clients to capture the HTML markup contained in the returned document.

Examples:

The following examples demonstrate how to return the HTML content from the CodePlex webpage.

C#:

```
Uri uri = new Uri("http://www.codeplex.com");
HttpWrapper http = new HttpWrapper(uri);
string response = http.Get();

Console.Out.WriteLine(response);
```

VB:

```
Dim uri As Uri = New Uri("http://www.codeplex.com")
Dim http As HttpWrapper = New HttpWrapper(uri)
Dim response As String = http.Get()

Console.Out.WriteLine(response)
```

How to perform a HTTP POST request

A Http POST request is similar to a GET request; howver normally some data is included in along with a HTTP POST request that the web server will process and may return a document based on the information that is posted back.

Examples:

The following examples demonstrate how to perform a POST to retrieve the weather for the next 5 days from the BBC's weather website.

C#

```
Uri uri = new Uri("http://www.bbc.co.uk/cgi-perl/weather/search/new_search.pl");
HttpWrapper httpWrapper = new HttpWrapper(uri);

string postData = "search_query=exeter";

string response = httpWrapper.Post(postData);

Console.Out.WriteLine(response);
```

VB:

```
Dim uri As Uri = New Uri("http://www.bbc.co.uk/cgi-perl/weather/search/new_search.pl")
Dim http As HttpWrapper = New HttpWrapper(uri)

Dim postData As String = "search_query=exeter"

Dim response As String = http.Post(postData)

Console.Out.WriteLine(response)
```

Web Library

To use this library, you will need to add a reference to the ***WebLibrary.dll*** in your project.

Add a using (c#)/imports (vb) to Savage.DotNetLibrary.WebLibrary

C#:

```
using Savage.DotNetLibrary.WebLibrary;
```

VB:

```
Imports Savage.DotNetLibrary.WebLibrary
```

Notes:

How to Generate Machine Keys

Machine keys are used by ASP.NET for encrypting data for applications, and this is normally set in the machine.config file when the .NET Framework is installed. However, there are some scenarios when you may need to define your own machine key for your application (such as running an ASP.NET website on a cluster of web servers-the machine key must be identical on all the web servers in the cluster running your application).

Example:

To generate a new machine key all you need to do is instantiate a new MachineKeyGenerator object.

C#:

```
MachineKeyGenerator mkg = new MachineKeyGenerator();
Console.Out.WriteLine("Validation Key: " + mkg.ValidationKey);
Console.Out.WriteLine("Decryption Key: " + mkg.DecryptionKey);
```

VB:

```
Dim mkg As MachineKeyGenerator = New MachineKeyGenerator()
Console.Out.WriteLine("Validation Key: " & mkg.ValidationKey)
Console.Out.WriteLine("Decryption Key: " & mkg.DecryptionKey)
```

Example:

The WebLibrary also contains a method that can not only generate the machine key for you but also save this to the web.config:

C#:

```
string                              applicationVirtualPath                              =
System.Web.Hosting.HostingEnvironment.ApplicationVirtualPath;


WebConfiguration.AddMachineKey(applicationVirtualPath);
```

VB:

```
Dim                                 applicationVirtualPath                              =
System.Web.Hosting.HostingEnvironment.ApplicationVirtualPath

WebConfiguration.AddMachineKey(applicationVirtualPath)
```

How to use Version Control on web.config files for Developers

A problem I have had to address is how do you handle web.config files for different developers and keep the web.config file under source control. Take the following scenario:

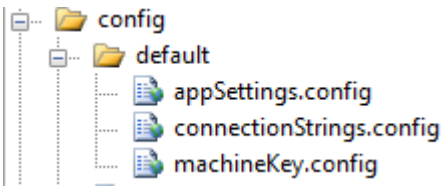
Another Company has a team of 10 developers, all working on the same ASP.NET web application. Each developer may have different settings in their web.config, for example Bob might have his SQL Server database installed as the default instance, while Ted might be using .\SQLEXPRESS. A new developer starts and he checks out from your code repository the latest version of the web application, which web.config file should he receive.

A solution I have found to be practical is to use the feature introduced in .NET 2.0 where you can add the configSource attribute to the main config to point to other external web.config files, and this can be done for all nodes.

The first step is to edit the main web.config file for the project to look like the following:

```
<appSettings configSource="config\appSettings.config"/>
<connectionStrings configSource="config\connectionStrings.config"/>
<system.web>
    <machineKey configSource="config\machineKey.config"/>
    ...
</system.web>
```

You then create a directory called config, and inside that directory create a subdirectory called default:



The default files contain the default settings for each section of the web.config, for example the appSettings.config file may look like this:

```
<?xml version="1.0"?>
<appSettings>
      <add key="Photographs" value="c:\Photos"/>
</appSettings>
```

All these files are then kept under version control, however the web.config file is not actually pointing to the default files, it is expecting these in the **config** directory, so we need to copy these from the **default** directory to the **config** directory.

The WebLibrary.dll component provides a method that will perform this copying, this could be called when in the application_started event:

C#:

```
string defaultConfigDirectory = Server.MapPath("~/config/default");
string configDirectory = Server.MapPath("~/config");

string                          applicationVirtualPath                          =
System.Web.Hosting.HostingEnvironment.ApplicationVirtualPath;

WebConfiguration.ConfigureApplication(defaultConfigDirectory,
configDirectory, applicationVirtualPath, false);
```

VB:

```
Dim defaultConfigDirectory As String = Server.MapPath("~/Config/default")
Dim configDirectory As String = Server.MapPath("~config")
Dim                             applicationVirtualPath                          =
System.Web.Hosting.HostingEnvironment.ApplicationVirtualPath

WebConfiguration.ConfigureApplication(defaultConfigDirectory,      configDirectory,
applicationVirtualPath, False)
```

This will then have copied the all the *.config files into the config directory from the default directory if they are not currently located in the config directory. All the *.config files in the config directory should then be ignored by the version control system.

The fourth parameter is a boolean and is set to true of you want a machine key to be generated and added to the web.config file if their is not a fixed machine key currently set (i.e if the current machine key is set to Auto).

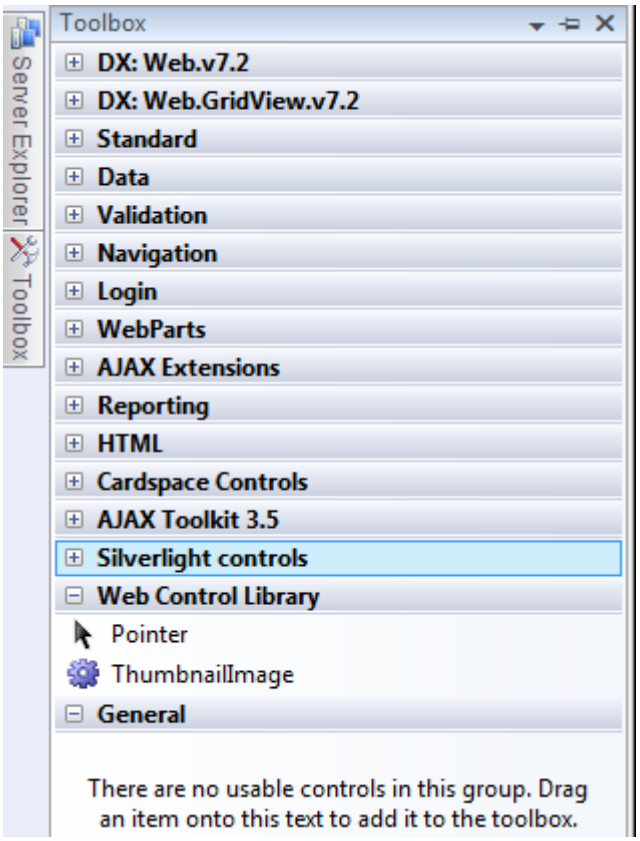
Each developer is then free to edit their web.config settings for their dev or testing environments without causing issues for other developers.

Web Control Library

This project will contain commonly used controls that have use in multiple ASP.NET applications. To begin with the library contains a web Thumbnail control that uses the Image Library components.

To use simply add a reference to **Savage.DotNetLibrary.WebControlLibrary.dll** in the ToolBox section in Visual Studio:

1. Right-click in the Toolbox and select **Add Tab**. Enter a name such as **Web Control Library**.
2. Inside the new tab you created right-click and select **Choose Items**.
3. After a short while a window opens named Choose Toolbox Items, in this window click the **Browse...** button.
4. Browse to the **Savage.DotNetLibrary.WebControlLibrary.dll** file.
5. Click OK and then you will be able to drag and drop the controls onto your web page.



ThumbnailImage

The ThumbnailImage control will produce a correctly scaled image on your web page. The original source image does not even have to be in a directory that is accessible from the website. So you could have a high resolution image stored on your web server, and then use this control to only display a thumbnail of the image.

Properties

SourceImagePath ***string*** ***null***

Gets or sets the physical path to the image that should be displayed as a thumbnail. Example value could be "c:\Images\Holiday2008\Beach.jpg"

NOTE: The application has to have security access to the file.

ThumbnailRootDirectory ***string*** ***thumbnails***

Gets or sets the name of the root directory that the thumbnails will be saved into. For a file to be viewed on a web page the thumbnail has to be created in a directory in your website. This defaults to "thumbnails". If the directory does not exist the directory will be created when the thumbnail is created.

ThumbnailFilename ***string*** ***null***

Gets or sets the filename for the thumbnail. This allows you to set the name of the thumbnail. If left null or as an empty string then depending on the value in the *UseOriginalFilename* property will determine the name the thumbnail will be given-if this is true then this will use the original filename, if this is set to false then a new filename is generated. The filename does not need to be specified since this will be the same as the source image. Example value could be "sandcastle".

UseOriginalFilename ***bool*** ***true***

Gets or sets a value that indicates if the original filename should be used for the thumbnail.

ThumbnailWidth ***int*** ***100***

Gets or sets the maximum width of the thumbnail. The thumbnail may have a lower width depending on how the image scales.

ThumbnailHeight ***int*** ***100***

Gets or sets the maximum height of the thumbnail.

InterpolationMode ***InterpolationMode*** ***InterpolationMode.Default***

Gets or sets the quality of the thumbnail that is created. See the section **InterpolationMode enumeration** in the appendix for more details.

Appendix

How to add a reference to your project in Visual Studio

1. Open your project in Visual Studio.
2. In the Solution Explorer window, right-click the Project you wish to add the reference to and select **Add Reference** (figure 1).

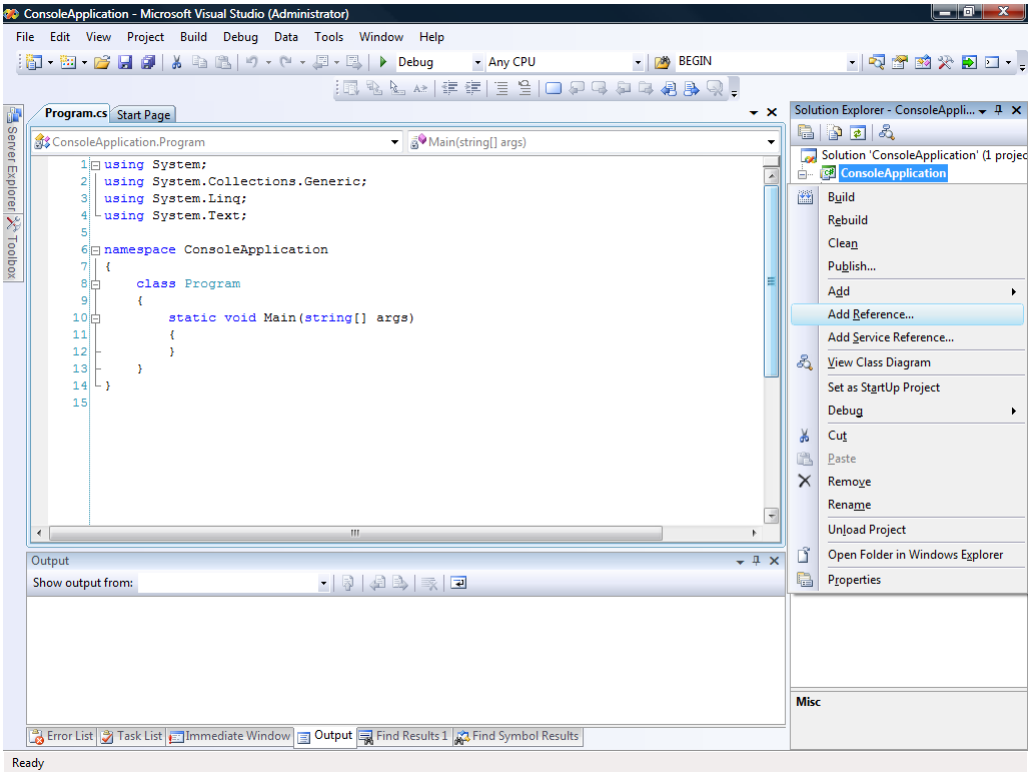


Figure 1

NOTE: If the Solution Explorer window is not visible this can be found under the **View Menu**, named **Solution Explorer**:

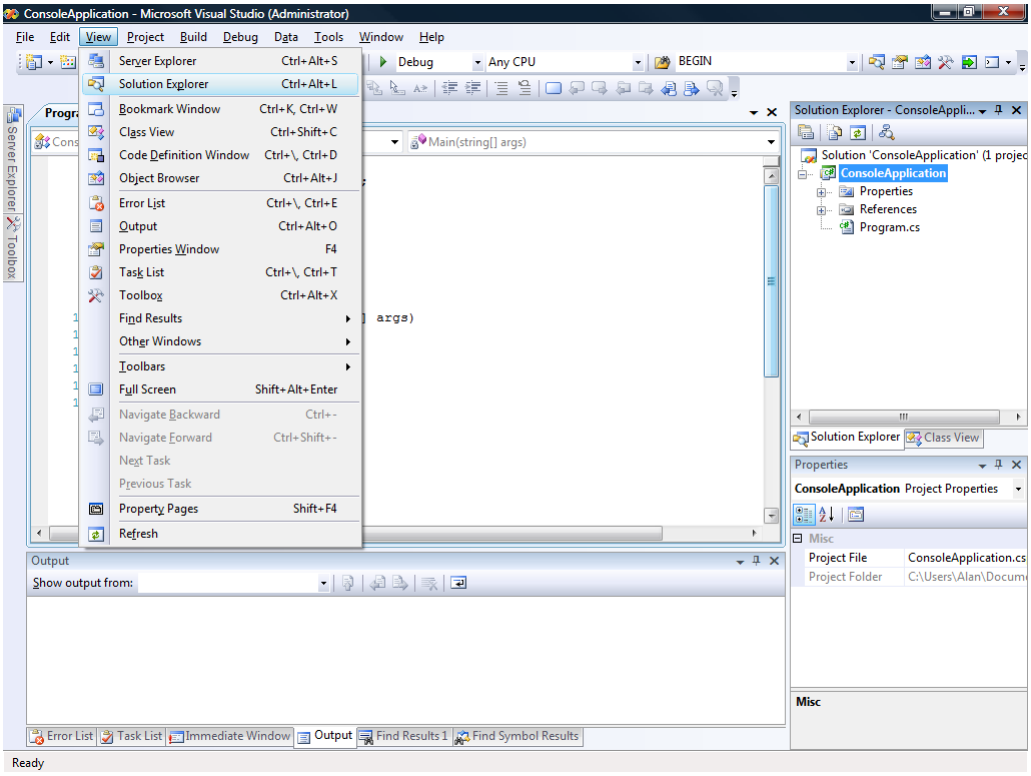


Figure 2

3. After a short delay a new window will open named **Add Reference**, you will need to select the **Browse** tab and then navigate to the file that you need to reference (figure 3).

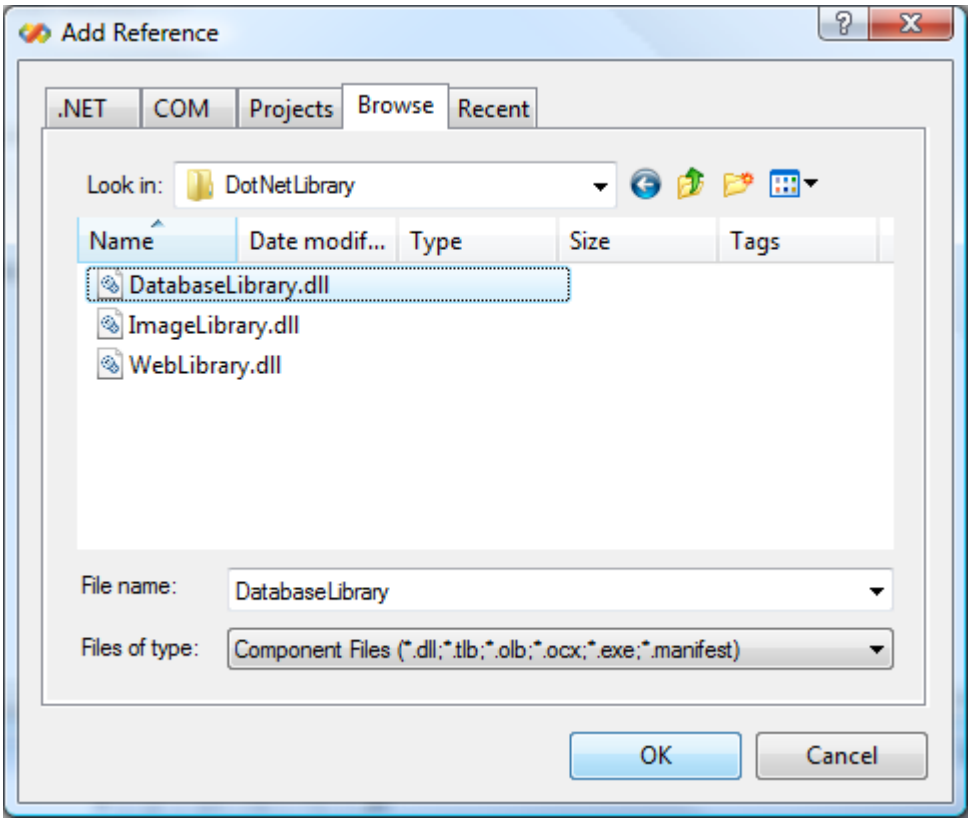


Figure 3

4. Click the **OK** button once to complete this process. If you have added a reference correctly you will see the component listed under the **References** branch inside Visual Studio (figure 4):

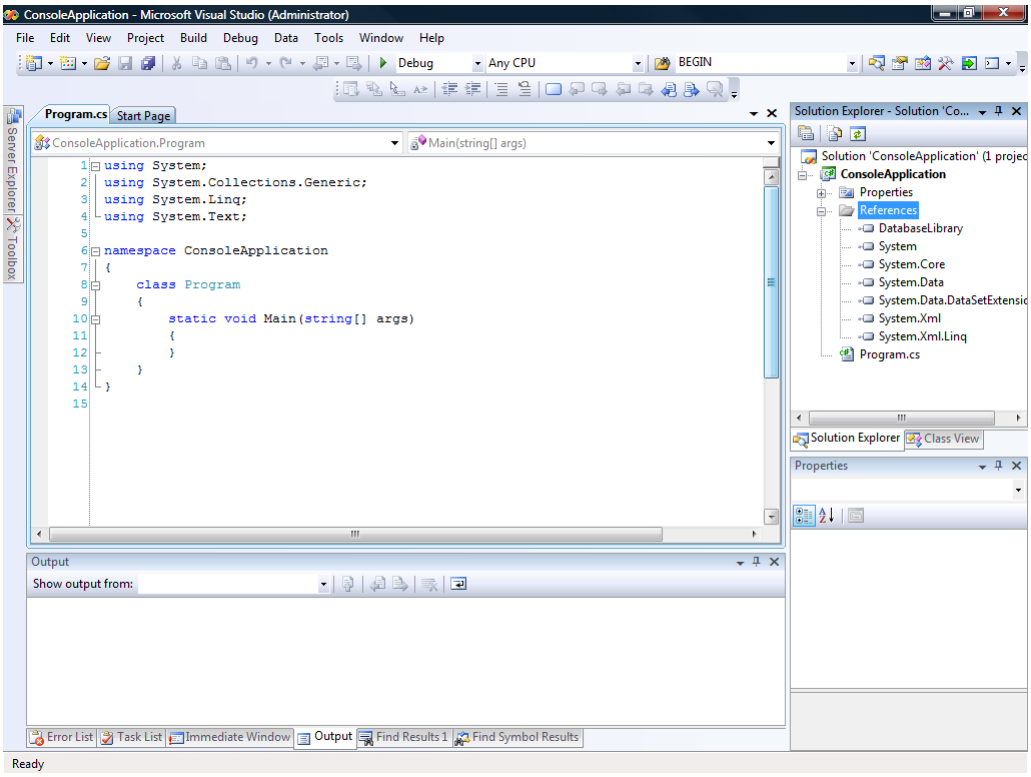


Figure 4

InterpolationMode Enumeration

The InterpolationMode enumeration specifies the algorithm that is used when images are scaled or rotated.

Member name	Description
Invalid	Equivalent to the Invalid element of the QualityMode enumeration.
Default	Specifies default mode.
Low	Specifies low quality interpolation.
High	Specifies high quality interpolation.
Bilinear	Specifies bilinear interpolation. No prefiltering is done. This mode is not suitable for shrinking an image below 50 percent of its original size.
Bicubic	Specifies bicubic interpolation. No prefiltering is done. This mode is not suitable for shrinking an image below 25 percent of its original size.
NearestNeighbor	Specifies nearest-neighbor interpolation.
HighQualityBilinear	Specifies high-quality, bilinear interpolation. Prefiltering is performed to ensure high-quality shrinking.
HighQualityBicubic	Specifies high-quality, bicubic interpolation. Prefiltering is performed to ensure high-quality shrinking. This mode produces the highest quality transformed images.

This enumeration is part of the .NET Framework Library and more documentation can be found at:

http://msdn.microsoft.com/en-us/library/system.drawing.drawing2d.interpolationmode.aspx