Applicability

This whitepaper was written against 11.4. We're going to be revisiting it to get it updated for 2026.2, but in the meantime you may need to transpose versions a bit

The sample apps we're linking are for DotImage 11.5.0.10 which should translate with little trouble. In places where the screenshot and text disagrees here, please defer to the text.

.NET FRAMEWORK

Before we start, a quick note; this article covers WingScan web scanning using .NET Framework (our 4.6.2 DLLs targeting .NET 4.6.2 through 4.8. However, it does NOT cover .NET 5 or up (sometimes called .NET Core)

For a tutorial about .NET 8 please see

INFO: WDV (and WebCapture) In .NET 8 Whitepaper - Getting Started

For .NET 10, please see:

INFO: WDV (and WebCapture) In .NET 10 Whitepaper - Getting Started

GETTING STARTED WITH WEB SCANNING

PLEASE NOTE: This is the version of this whitepaper that applies to 11.4 of DotImage. It was updated from the 11.3 version as there was virtually no change except one script inclusion and a couple paths.. we will note in this updated for 11.4 paper the differences for 11.2 / 11.3 / 11.4. However, the original was written for 11.0 and there are significant enough changes that the main article was rewritten.

Please DO NOT attempt to use any version < 11.4.0.4 - Multiple browser changes have made previous versions unusable.

WingScan is a web-based TWAIN scanning SDK that allows developers to add scanning support to existing or new web applications. WingScan can be purchased as a stand-alone

SDK. This level of licensing allows developers to add our Web Scanning components to a web site and allows LIMITED use of the WebDocumentViewer (Using only a WingScan license, WebDocumentViewer will allow viewing only of documents scanned in the current session., and does not allow any custom handling. Viewing, annotation, rotation, page reordering (via drag/drop on the thumbnail viewer), and saving are supported via defaults.

Any custom handling or even viewing of documents from previous sessions require a license to DotImage Document Imaging to fully unlock the WebDocumentViewer)

In our getting started section, we're going to start out with a pure scanning with no viewing in order to show you the absolute minimum WingScan setup possible. This is useful for situations where you are just adding scanning to an existing app and already have viewing in place or do not require a viewing component.

We'll then move on to show using WingScan with the WebDocumentViewer and WebDocumentThumbnailer controls under the WingScan only license.

More in-depth discussion of the full WebDocumentViewer and WebDocumentThumbnailer controls are provided in our WebDocumentViewer guide.

The idea here is once you have a sense of using WingScan, you can see how to add it to any application.

WINGSCAN HISTORY

Originally, this paper was going to include a history of WingScan web scanning, but for the sake of brevity we'll just link you here:

INFO: A Brief History of Web Based Scanning at Atalasoft

DOCUMENT STANDARDS AND ASSUMPTIONS

The technical portions of this document assume that you are a developer with access to MS Visual Studio 2013 or later. We will be providing examples in C#. Our SDK works with VB.NET as well, but for the sake of simplicity any .NET code will be provided in C# You can use a converter tool such as Telerik Code converter to convert examples to VB.NET

INFO: WingScan Whitepaper - Getting Started with Web Scanning (in .NET Framework)

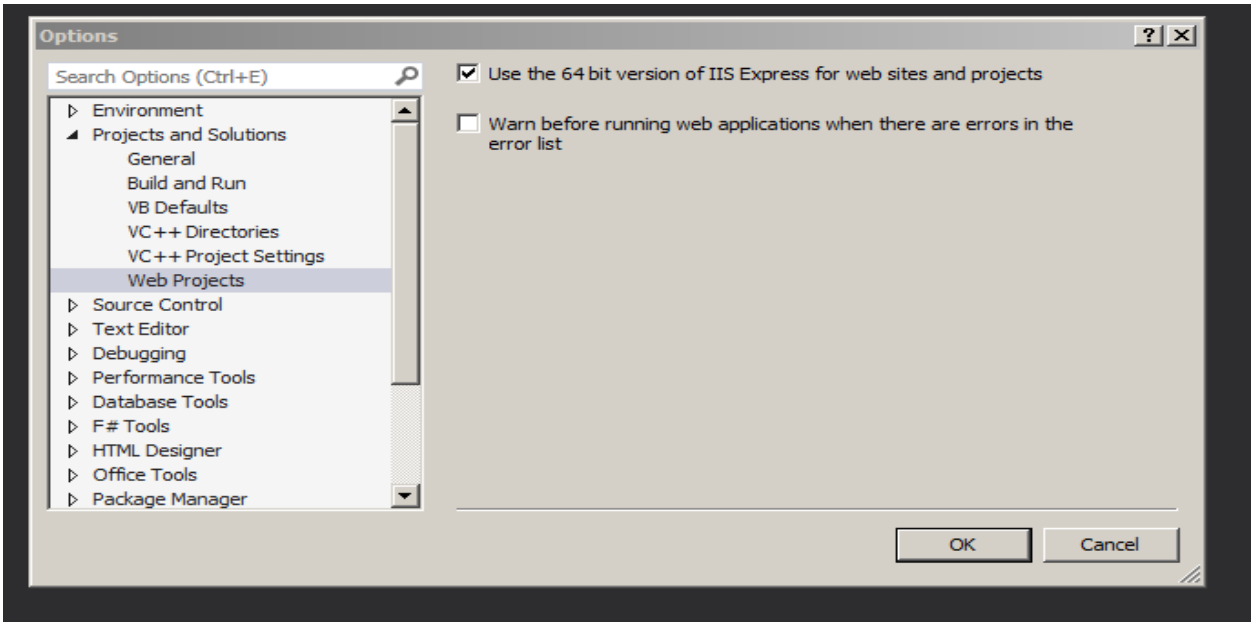
This document and our SDK assume that you're a developer familiar with basic use of Visual Studio, IIS Express or IIS, and general web application development practices. The samples will be using an HTML 5 approach in a non-MVC application targeting .NET framework 4.5.2 in a 64 bit application.

Please note that DotImage 11.0 and up support .NET 3.5 (using our 3.5 dlls) or .NET 4.5.2 or greater (using our 4.5.2 dlls) .. support is not possible for .NET 4.0, 4.5 or 4.5.1 using DotImage 11.0 or up. If you have an existing app you want to add DotImage to that targets an unsupported .NET framework, you will need to update it.

Also note that this document covers a standard ASP.NET web app, not ASP.NET Core. We do have support for ASP.NET core targeting .NET framework (not pure .NET Core), but that is beyond the scope of this document. The fundamental concepts of this document apply whether you are going to use ASP.NET or ASP.NET Core the client-side code is the same - the difference is in the handler (used for ASP.NET) versus the startup middleware (used by ASP.NET Core).

A brief set of instructions for downloading our SDK and activating a license will be given, but the rest of the document assumes you've installed the latest DotImage SDK (currently 11.4.0.14 as of March 2024)

Examples herein will use IIS Express (built in web server for Visual Studio) set to run in x64 mode



We do provide NuGet packages for our components, but this example will assume you've installed and activated our SDK locally.

If you're a licensed developer, or are actively evaluating our SDK and run into problems/questions, you are welcome to contact support. You can make a support case in writing in our support portal

You may also call in to support during our normal business hours

Support Hours: M - F from 8AM - 5PM EDT (New York Time)

Call us: 1-781-743-2119

STANDARD TROUBLESHOOTING

Web applications with WingScan have a lot of "moving parts" and troubleshooting can seem a daunting task.

There are two tools (one ours, one a free third party tool) that can greatly enhance troubleshooting and assist you in reporting issues to support

If you are having issues with WingScan

Please download and run the diagnostic logging tool

Use it to collect a log while you reproduce the issue.

These diagnostic logs will often show the root cause of an issue. Although they may be a bit daunting to look at at first, rest-assured, they're a valuable diagnostic tool. So, if you're contacting support about a WingScan issue, please take a moment to collect this log then save it to a text file, then zip up the log file and attach it to your support case

In addition to the WingScan log, problems related to the client-side web components are best diagnosed by looking at a log of the network traffic. Most commonly a faulting application will clearly show a 500 internal server error being thrown and the full response body of that error will offer a great deal of insight into the root cause (licensing issues, and errors in the WebCaptureHandler and WebDocumentRequestHandler will most often show up

in the 500 error response body)

We recommend Fiddler if you need to collect a log and report it to support.

Please download, install, and run Fiddler web logging

Use it to collect a log while you reproduce the issue, then save the log as a native .saz file and and attach it to your support case as a file attachment/ (please do not save as CSV or text log. The native SAZ file gives us much better access to the tools we need to help diagnose your issue)

PLEASE NOTE: you need to capture a session while NOT using SSL/HTTPS. Fiddler logs cannot see into HTTPS without enabling a special certificate which we do not recommend.. if your capture is of an HTTPS session we will not get useful diagnostic information from the log

INSTALLATION AND LICENSING

If you have not done so already, please download the latest version of Atalasoft DotImage SDK

NOTE: The current version is 11.4.0.14 as of the creation of this document (March 2024) updates of the SDK happen with regularity. We work hard not to make breaking changes so this guide should be compatible with future releases. We will update the specific technical details as needed as changes arise

You can not use < 11.4.0.4 in Google Chrome or Microsoft Edge due to changes in Chromium engine

If you have not done so, please create an account with Atalasoft so that you can download and activate our SDK

- Download and install the Atalasoft DotImage SDK
- Run the activation wizard
- If you have SDK serials for DotImage and/or WingScan use the "Activate an SDK serial for use on this machine or a server license" option and activate your serials for version

11.4
- If you do not yet have licenses, you can select the "request 30 day evaluation"
- If you run into issues activating or require assistance, please contact support

NOTE for IIS users: The licensing above should be sufficient if you're using the built in web server in Visual Studio (IIS Express) but if you are using a local copy of IIS, then the IIS process does not run as your user name so it will not pick up your licenses. You'll need to take the extra step of copying your licenses into the application bin directory.

Assuming you have an app named WingScanTestApp hosted in IIS under:

c:\inetpub\wwwroot\WingScanTestApp\

Then you'd copy your licenses from:

C:\users\YOUR_USERNAME\AppData\Local\Atalasoft\DotImage 11.4\

to

c:\inetpub\wwwroot\WingScanTestApp\bin\

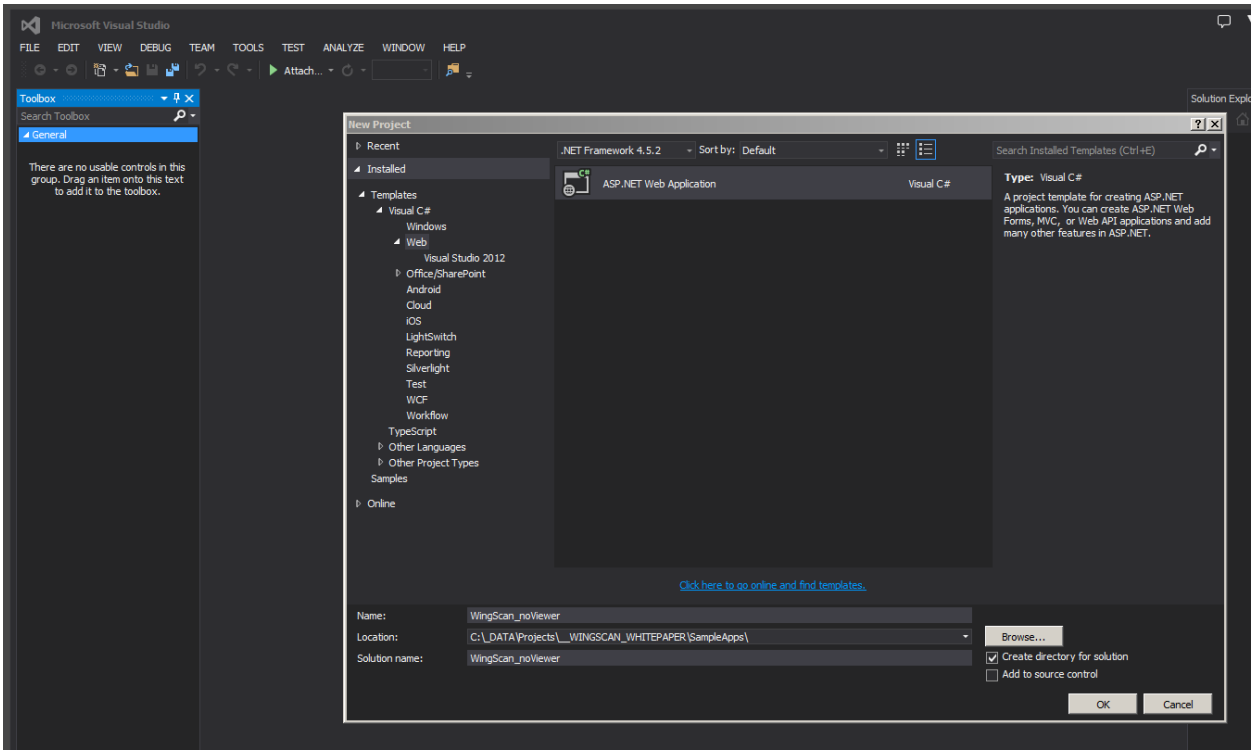
EXAMPLE 1

BUILDING A MINIMUM WINGSCAN APP

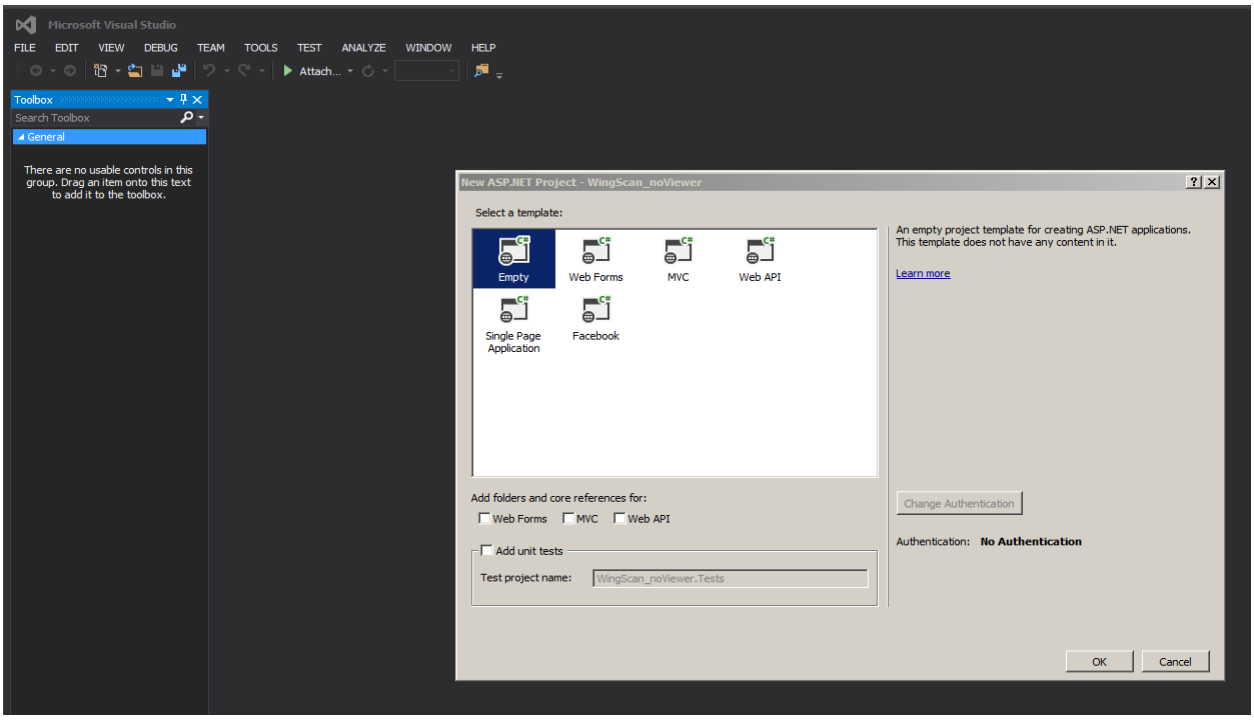
We're going to be creating the minimum possible WingScan application. It's just going to scan and upload.

While this may seem not too useful, it's the fundamental minimum you need to add scanning. Your use case may not involve using our viewers (we'd love you to, but maybe you have another viewer, or maybe your app is meant to be fairly utilitarian and viewing isn't needed. Either way, this minimalist approach should be easily adaptable to any application where you want to add scanning. Don't worry, we will get fancier in a bit. For now, we're going to start with the basics.

- Open Visual Studio 2022 (or newer)
- Create a new project a C# ASP.NET web application targeting .NET framework 4.6.2 (or higher if you wish, but NOT .NET Core)



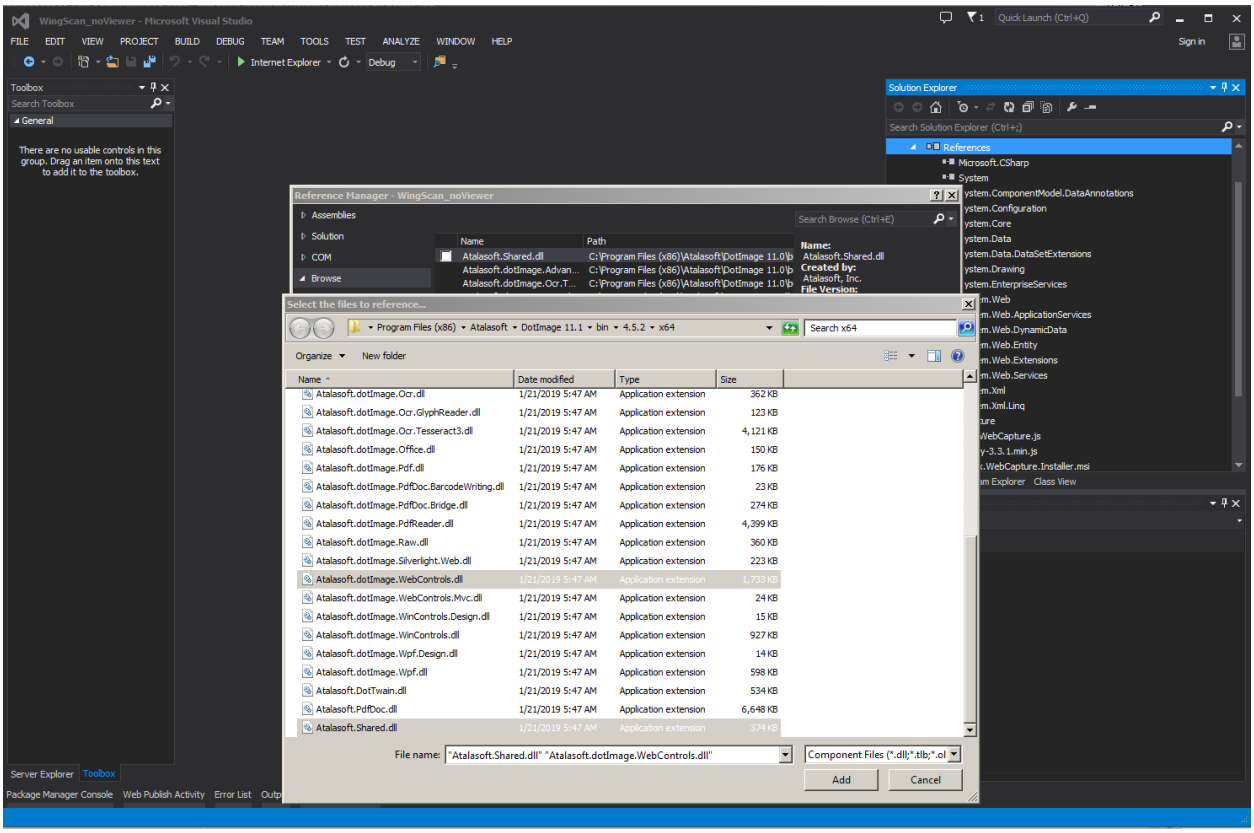
- Name the project/solution WinScan_noViewer
- We will be setting it up for EMPTY project



- Now, add references to the following Atalasoft components:

  Atalasoft.dotImage.WebControls.dll

  Atalasoft.Shared.dll



- Make sure you're using the 64 bit .NET 4.2.2 versions from:

  C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\4.2.2\x64\

- In Solution explorer, add the following two new folders at the root of the project:

  atala-capture-upload

  WebCapture

- Copy the contents of:

  C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\WebResources\WebCapture\

  into the WebCapture folder

- Copy jquery-3.5.1.min.js from:

  C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\WebResources\WebDocViewer\

into the WebCapture folder

- Add a new aspx page (Default.aspx)

  go to Add New item

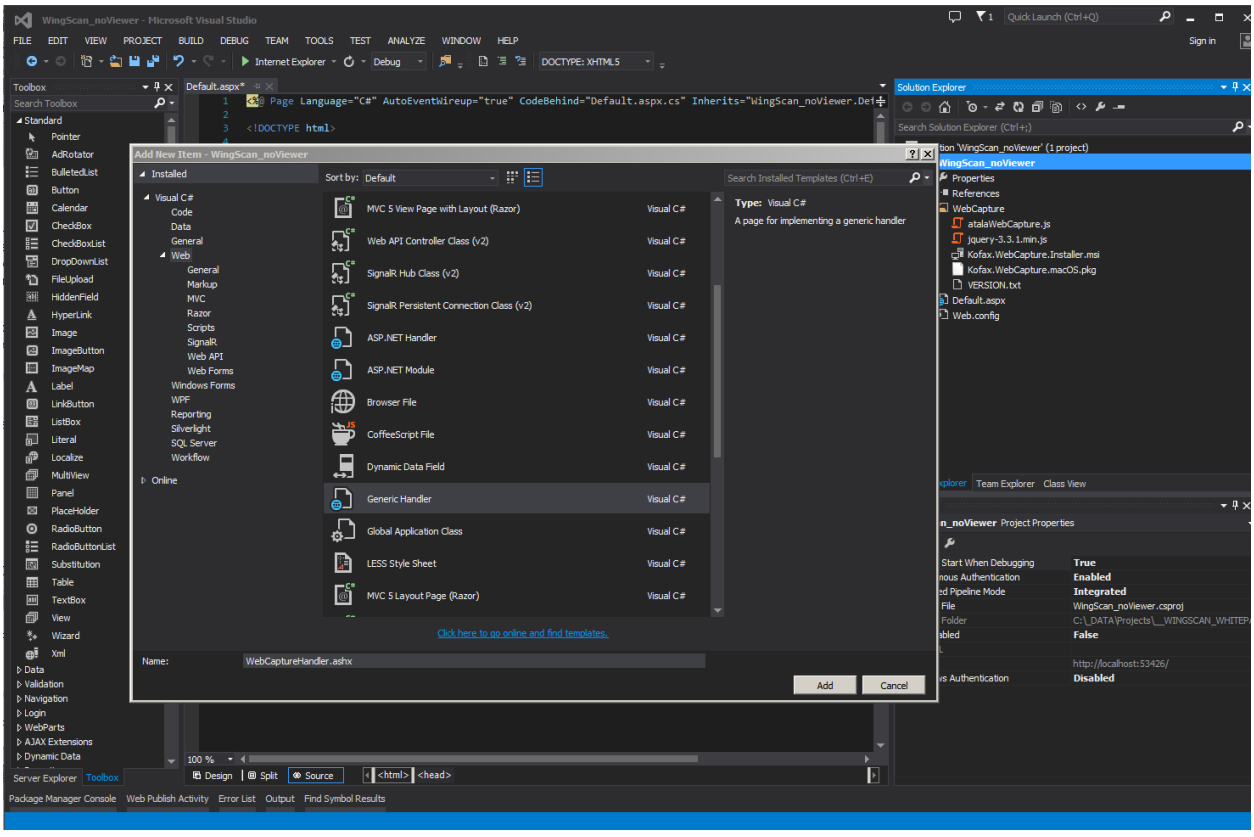
  WebForm

  name it Default.aspx
- Edit the Default.aspx page

  in the <header> section, add the following:

```
<script src="/WebCapture/jquery-3.5.1.min.js" type="text/javascript"></script> <script src="/WebCapture/atalaWebCapture.js" type="text/javascript"></script>
```

- Now, we'll need to set up the server side component. The ASHX handler that will be responsible for the back end server side calls.
- Go to the solution explorer, and right click and add new item of type Generic Handler



- Name it WebCaptureHandler.ashx
- That handler will have some default content. You now need to edit the

WebCaptureHandler.ashx and wipe out its content and replace it all with this:

```
using System; using System.Collections.Generic; using System.Linq; using System.Web;
using Atalasoft.Imaging.WebControls.Capture; using System.IO; namespace WingScan_noViewer
{ public class WebCaptureHandler : WebCaptureRequestHandler { public WebCaptureHandler()
{ } } }
```

- Ok, you've got the prerequisites all set. Now we need to initialize the scanning component
- Go back to default.aspx and in the header and just after the call to the scripts we included above, we need to add a manual script block

```
<script type="text/javascript"> // Initialize Web Scanning. This will run at startup
$(function () { try { Atalasoft.Controls.Capture.WebScanning.initialize({ handlerUrl:
'WebCaptureHandler.ashx', onUploadCompleted: function (eventName, eventObj) { if
(eventObj.success) { // for this demo we are just going to notify the user of the
filename uploaded // in your app you may want to take additional action.
eventObj.documentFilename will have the name of the file that got uploaded
document.getElementById('resultDiv').innerHTML = "Status: Upload Success... " +
eventObj.documentFilename; } }, onScanError: scanErrorHandler, scanningOptions: {
applyVRS: false, pixelType: 0, showScannerUI: true } }); } catch (error) { alert('Thrown
error: ' + error.description); Atalasoft.Controls.Capture.WebScanning.dispose(); } });
function scanWithSelectedScanner() {
Atalasoft.Controls.Capture.WebScanning.scanningOptions.scanner =
$('.atala-scanner-list').val(); Atalasoft.Controls.Capture.WebScanning.scan(); } // this
is a really simplified error handler for now. We will expand on this later function
scanErrorHandler(msg, params) { alert(msg); } </script>
```

- Ok, now, you've initialized the scanning and set it up for use

  Inside the Form section of the page, paste in this code

```
<div id="scanDiv"> <h1>Basic WingScan Web Capture Demo</h1> <h3>Using WingScan and NO
Viewer</h3> <p>Select your scanner and click the scan button... the file will be uploaded
to /atala-capture-upload/ with a unique filename (the filename will appear in the status
below)</p> <p> </p> <p>Select Scanner: <select class="atala-scanner-list"
disabled="disabled" name="scannerList" style="width:22em"> <option
selected="selected">(no scanners available)</option> </select>   <!-- DO NOT USE <input
type="button" class="atala-scan-button" value="Scan" /> --> <input type="button"
id="scanNOW" value="Scan" onclick="scanWithSelectedScanner(); return false;" /> </p>
</div> <div id="resultDiv">Status: NoUploads</div>
```

- Ok, now if you run the project it will likely throw you an error message

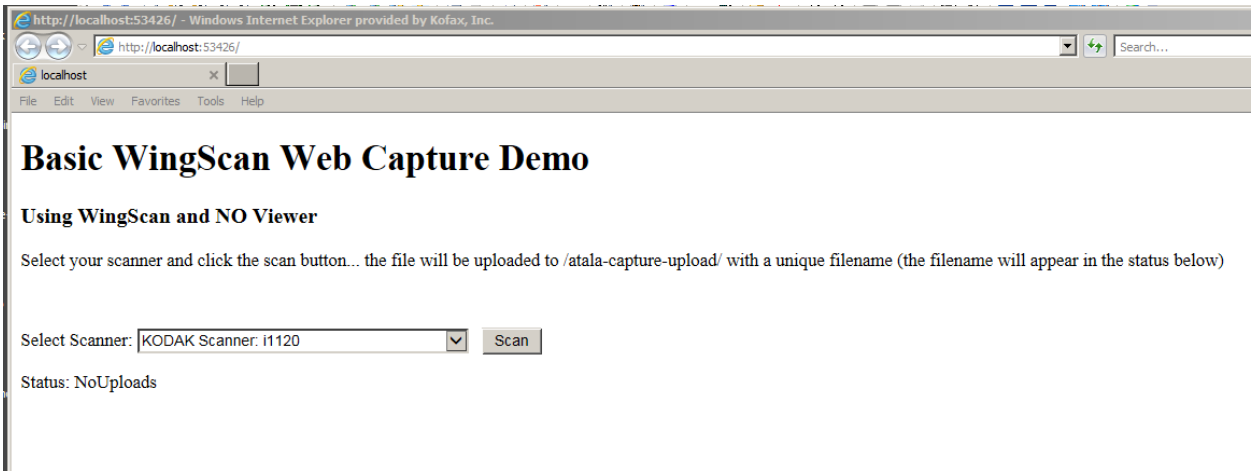
  saying the scanning service is not installed/available.

  This is because you do not have the scanning service installed on your local machine.
- In later sections we'll talk about gracefully handling this type of error message, but for now, just go ahead and navigate to:

C:\Program Files (x86)\Atalasoft\DotImage 11.4\bin\WebResources\WebCapture\

or the WebCapture folder of your app and run Kofax.WebCapture.Installer.msi

- Now, go ahead and run the app in Visual Studio
- If the app starts up and gives you our rather ugly page that shows a list of scanners and a scan button, congratulations!



- Pick a scanner and scan
- If you get any errors in the msg window, read them for a hint as to the root cause. You must have at least one TWAIN scanner installed on your system to use the application
- Assuming you have a scanner, select it and hit scan.
- The scan should start and the status will update with the name of the file uploaded
- If you look in the atala-capture-upload folder (remember we created this earlier?) that file will be there
- Congratulations! you've built the most basic working scanning app with WingScan.

MAKING IT BETTER

Ok, so the interface is really minimal and all it does is scans and uploads, but when it encounters errors, it really isn't al that helpful.

That's Ok we can make it better

One of the first things I like to do is to handle the "no plugin/ out of date plugin" errors. This is because in a real app, you would probably rather have your customer be prompted to download/install the viewer directly.

Ok, so let's flesh out the error handling.

We're going to be greatly expanding upon the scanerrorHandler we set up earlier.

```
// this is a really simplified error handler for now... we can expand on this later function
scanErrorHandler(msg, params) { alert(msg); }
```

If you note: there are two components, the message (msg) and the parameters (params). When you get an error, there will always be a message, but parameters are only present for specific errors where they are needed.

The WingScan components have a set of predefined messages.. this is handy both for translations to other languages as well as using in a switch/case statement to allow for custom messaging.

First, here's a list of the errors and their meanings:

Error Code	Defa
ajax	Could not create an XMLHttpReq
badBrowser	Scanning requires Chrome, FireF version 8 and with Compatibility Windows.
badVrsLicense	VRS license is missing or invalid.
batchFieldsError	The server could not retrieve the class.
batchFieldValidationError	Batch field validation failed.
brokenConnection	Connection to the scanning servi the page to restore it.
contentDescError	The server could not retrieve the
contentTypesError	The server could not retrieve cor
docClassIndexFieldError	The server could not retrieve the
doorOpen	Scanner reports cover open.
doubleFeed	Scanner reports a double-feed.

driverCrash	Fatal exception in scanner driver
dsmFail	Failure in TWAIN Manager.
dsOpen	Failed to open scanner.
fieldValidationError	Batch, or index field validation fa
fileFail	File not found or cannot be writte
fileLocked	File locked by other application.
helperDll	Failed to load scanning service h
importError	The server could not import the
indexFieldValidationError	IndexField validation failed.
internalError	Unexpected internal error.
licensingError	The server did not return valid li
lowIntegrityAccessDenied	Access to the Web Capture Servi has insufficient rights. This can h with the untrusted certificate or from internet. Contact your adm
noPlugin	The scanning service needs to be to be enabled. In the browser loc
noTwain	The TWAIN Manager needs to be
oldPlugin	The scanning service needs to be
oldWindowsService	The scanning service needs to be your administrator for assistance
outOfMemory	Insufficient memory to complete Service may help.
paperJam	Scanner reports a paper jam.
scanFail	Unable to start scan.
serverNotResponding	The server is not responding.
tooManyImages	Too many images for Scanning s
unsupportedFormatWarning	Warning: Only the following imag BMP, GIF, JPG/JPEG, PDF, TIFF. will not be imported.
uploadError	The server reported an error whi

userCancel	
vrsBadCommand	eVRS command string is invalid.
vrsBadDpiWarning	Warning: One or more images w requires image DPI > 25
webServiceMissed	Web Capture local service is not
xferFail	Image transfer from scanner fail

We could custom handle any/all of these.. but realistically, our default displaying of the message will do.. so we can just concentrate on ones where we want some special handling.

The ones needing our attention are:

- noPlugin
- oldPlugin
- oldWindowsService
- webServiceMissed
- noTwain

noPlugin and oldPlugin are errors that happen when a local user account has either not installed the web capture service, or has an out of date version of it respectively. The fix for both situations is to provide them with a link and install instructions.

webServiceMissed and oldWindowsService are similar, but are related to running in a Terminal Server / Citrix environment where the administrator must install the WCS as a system service

noTwain is important because the default message isn't instructive enough.

These examples will show you the pattern of how to handle such errors so you can add your own custom error handling for any/all events.

```
// This is an updated handler with more sophisticated handling of certain errors function
scanErrorHandler(msg, params) { switch (msg) { case
Atalasoft.Controls.Capture.Errors.noTwain: alert("This web capture application requires that
you have at least one " + "valid TWAIN device installed and configured. Please ensure you
install " + " the TWAIN driver for your scanner device(s) then re-visit this site."); break;
case Atalasoft.Controls.Capture.Errors.oldWindowsService: alert("The version of Kofax
```

```
WebCapture Service running on the multiuser " + "system you're using is out of date. Please contact your system administrator."); break; case Atalasoft.Controls.Capture.Errors.webServiceMissed: alert("The Kofax WebCapture Service on the multiuser system you're using " + "is not running. Please contact your system administrator."); break; case Atalasoft.Controls.Capture.Errors.noPlugin: var installerUrl = window.location.protocol + "//" + window.location.host + "/WebCapture/" + params.filename; window.open(installerUrl, '_downloadService'); alert( "The WingScan Web Scanning service is not available. \n\n" + "If the download doesn't start automatically, please download from " + installerUrl + "\n\n" + "If you are not prompted to install, the service may be installed but not running. \n\n" + "enable it by running from START->All Programs->Kofax Web Capture Service->Kofax Web Capture Service \n\n" + "Refresh your browser when completed."); break; case Atalasoft.Controls.Capture.Errors.oldPlugin: var installerUrl = window.location.protocol + "//" + window.location.host + "/WebCapture/" + params.filename; window.open(installerUrl, '_downloadService'); alert( "The WingScan Web Scanning service is out of date.<br />" + "If the download does not begin automatically, please go to " + params.filename); break; default: alert(msg); break; } }
```

EXAMPLE 2

WINGSCAN WITH VIEWER

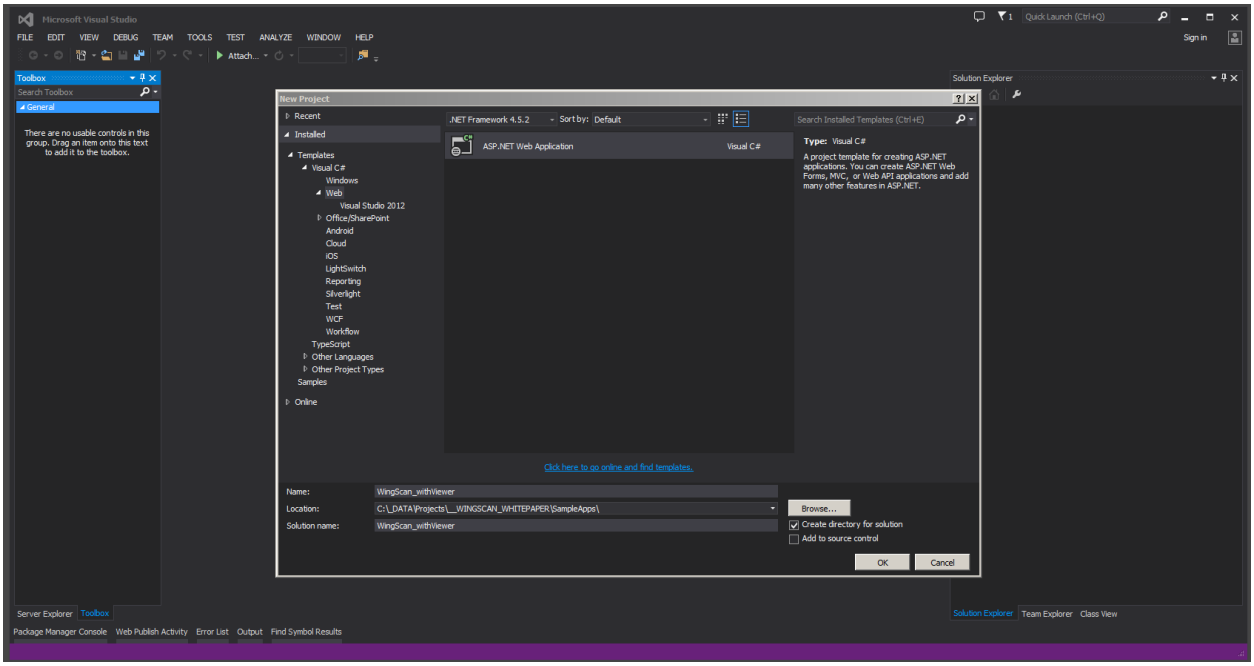
Up to this point, the whole setup has been bare minimum, but now let's say you want to have users view the document they scanned.

WingScan licensing on its own (without DotImage Document Imaging) gives you limited use of the WebDocumentViewer and WebDocumentThumbnailer controls. They are not able to be used to open or modify images from other sources, but they can be used to view images just captured in the current session.

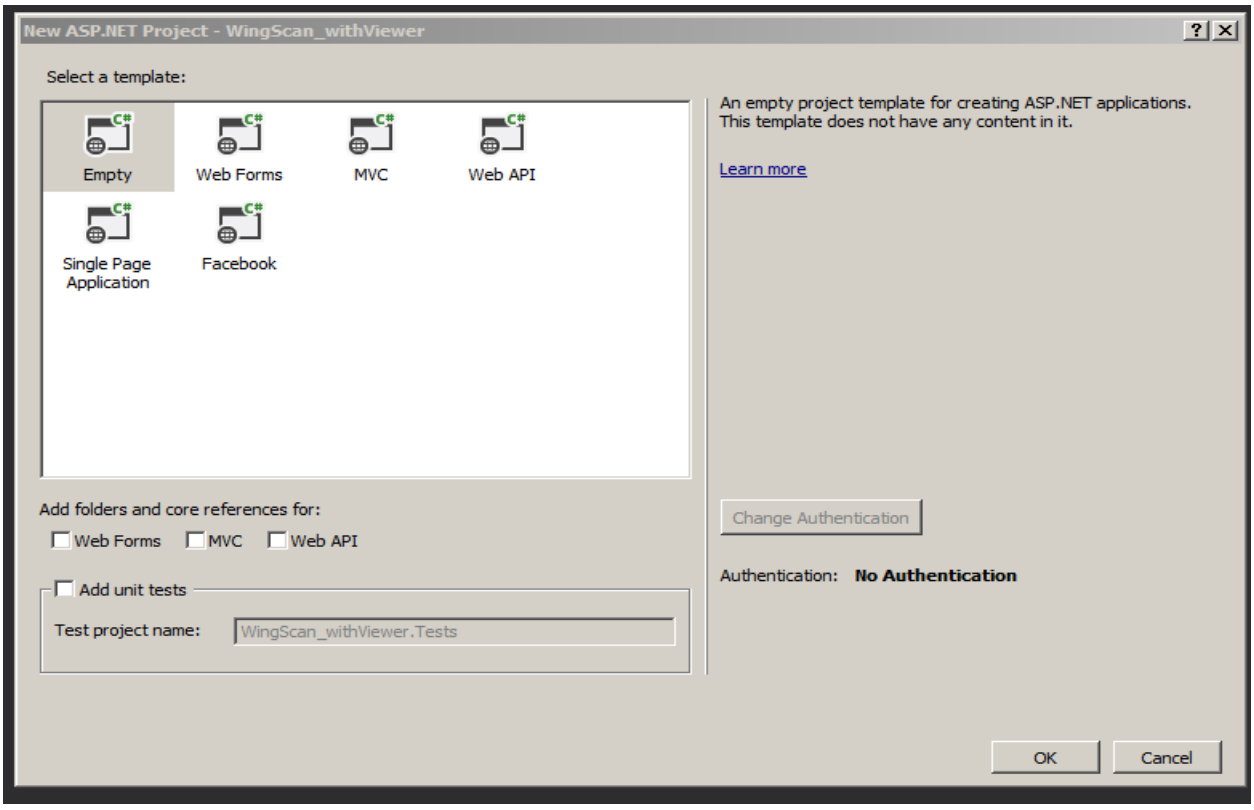
This paper will explain the basics of hooking up the WDV and WDT to the WebCapture scanning. For more advanced approaches to our web viewing controls unlocked with a full DotImage Document Imaging license see WebDocumentViewer Whitepaper.

The getting started steps are going to be nearly identical to the Wingscan_noviewer sample app, but there are several significant differences, so you probably will run into issues if you just try and modify the noViewer sample.

- Open Visual Studio 2022
- Create a new project a C# ASP.NET web application targeting .NET framework 4.6.2 (or higher if you wish, but NOT .NET Core)



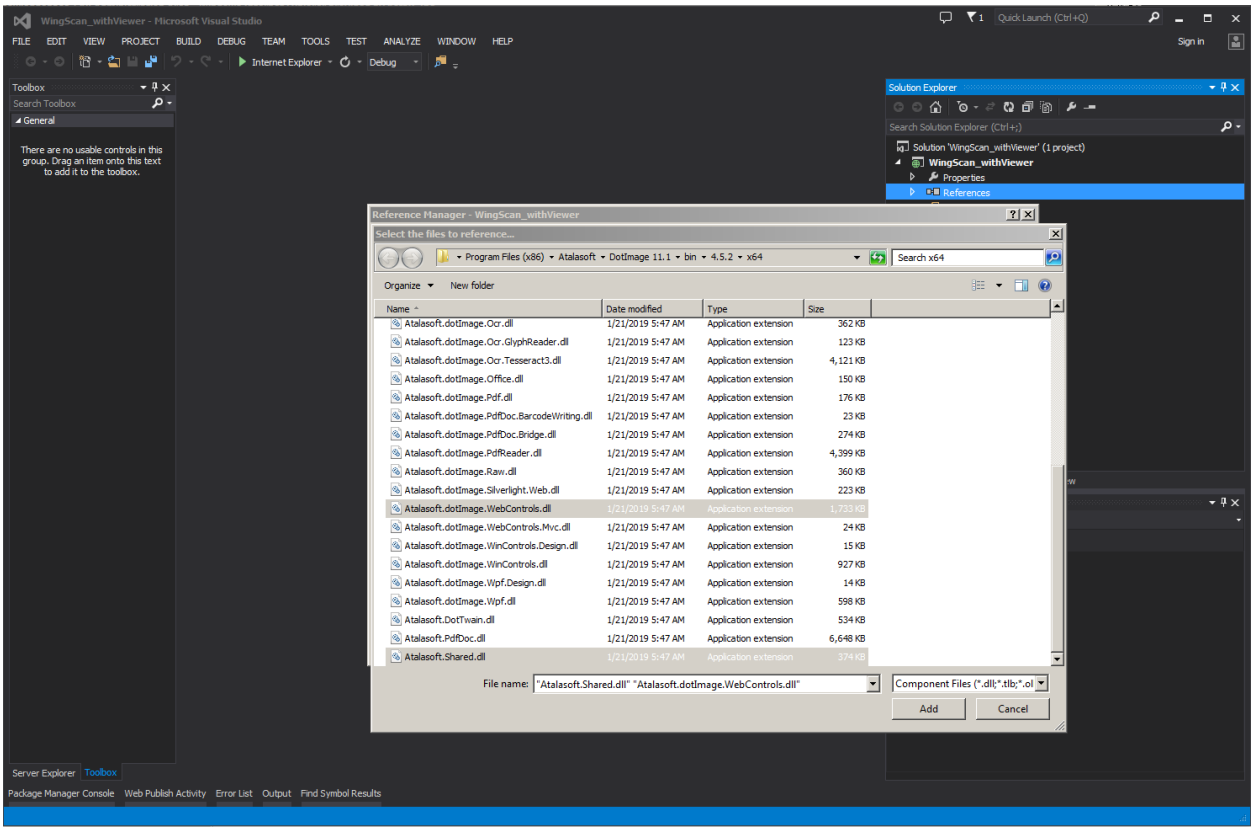
- Name the project/solution WinScan_withViewer
- We will be setting it up for EMPTY project



- Now, add references to the following Atalasoft components:

  Atalasoft.dotImage.WebControls.dll

  Atalasoft.Shared.dll



- Make sure you're using the 64 bit .NET 4.6.2 versions from:

  C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\4.6.2\x64\

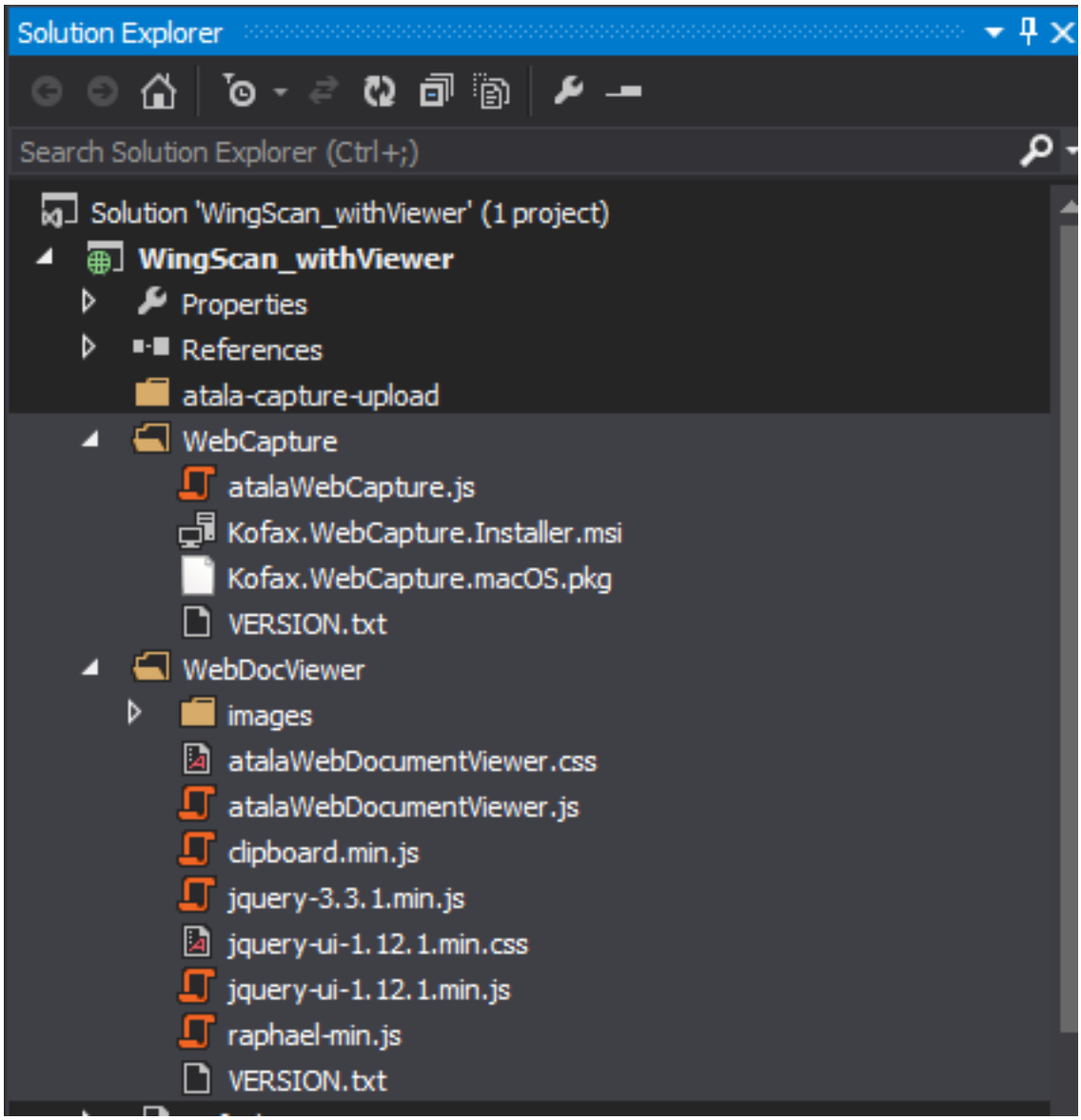
- In Solution explorer, add the following three new folders at the root of the project:

  atala-capture-upload

  WebCapture

  WebDocViewer

  (PLEASE NOTE: Screenshot was made for slightly older version so things like Jquery will be 3.5.1 and JqueryUI will be 1.14.0 in current versions - Always use the latest version of DotImage and use the specific files that ship with them)



- Copy the contents of:

  C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\WebResources\WebCapture\

  into the WebCapture folder
- Copy the contents of:

  C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\WebResources\WebDocViewer\

  into the WebDocViewer folder.
- Add a new aspx page (Default.aspx)

  go to Add New item

WebForm

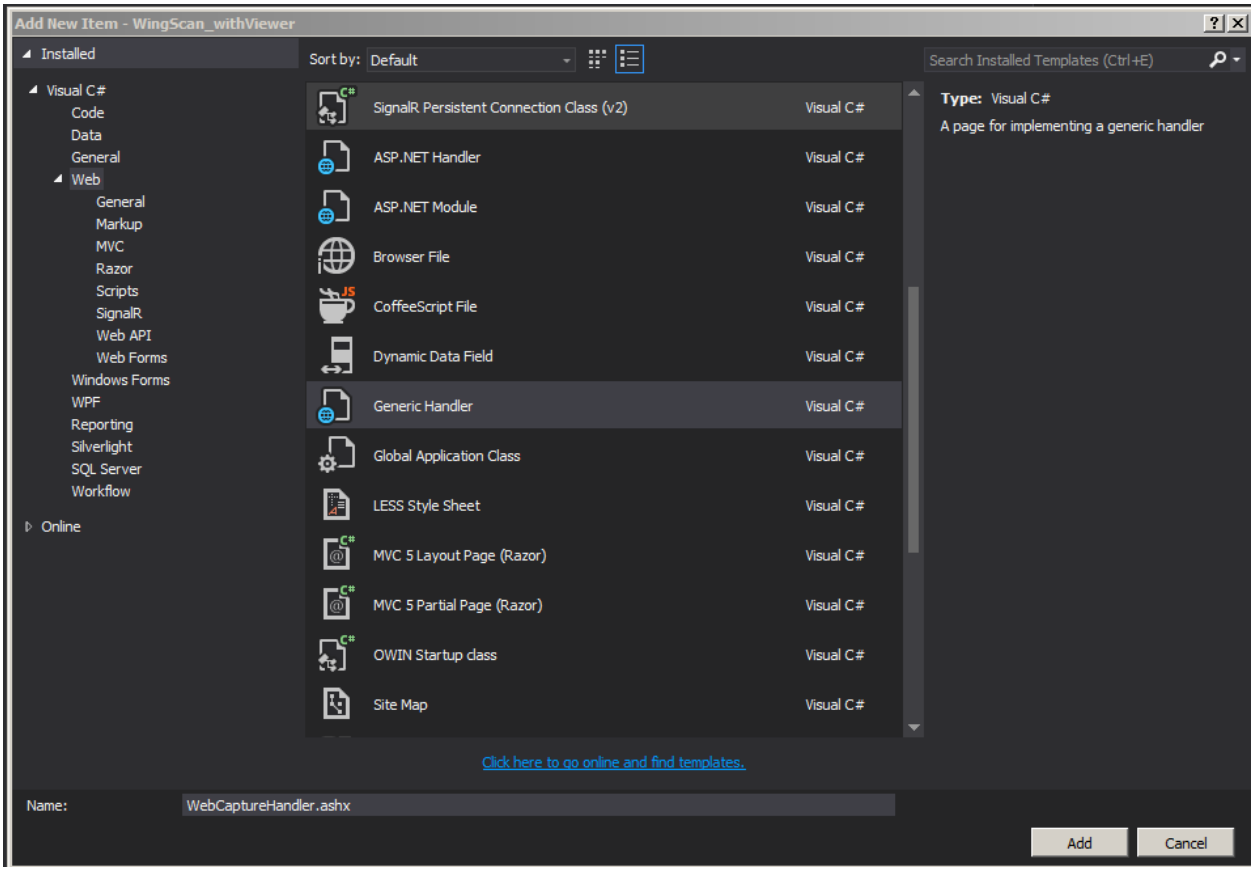
name it Default.aspx

- Edit the Default.aspx page

in the <header> section, add the following:

```
<!-- Script includes for Web Viewing --> <script src="/WebDocViewer/jquery-3.5.1.min.js"
type="text/javascript"></script> <script src="/WebDocViewer/jquery-ui-1.14.0.min.js"
type="text/javascript"></script> <script src="/WebDocViewer/raphael-min.js"
type="text/javascript"></script> <script src="/WebDocViewer/atalaWebDocumentViewer.js"
type="text/javascript"></script> <!-- Style for Web Viewer --> <link
href="/WebDocViewer/jquery-ui-1.14.0.min.css" rel="Stylesheet" type="text/css" /> <link
href="/WebDocViewer/atalaWebDocumentViewer.css" rel="Stylesheet" type="text/css" /> <!--
Script includes for Web Capture --> <script src="/WebCapture/atalaWebCapture.js"
type="text/javascript"></script>
```

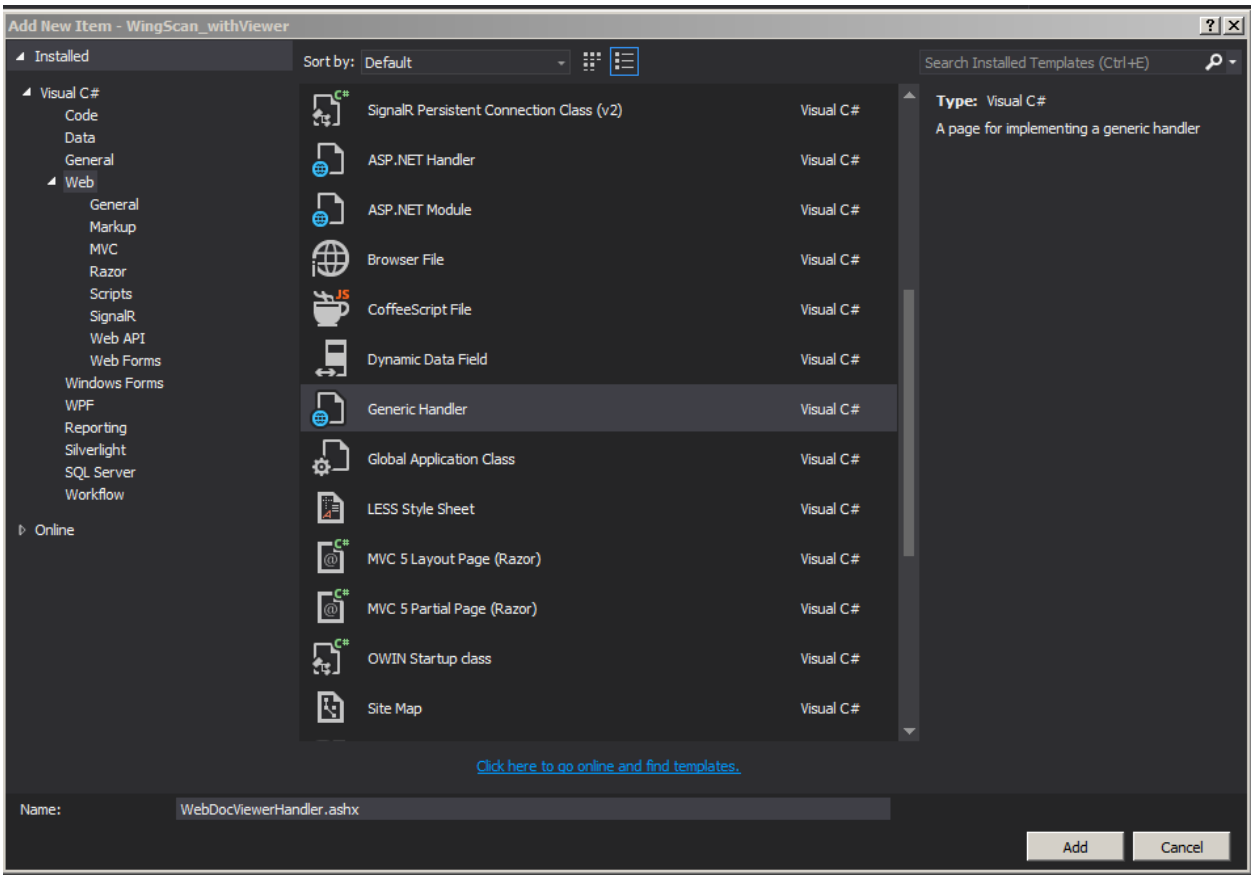
- Now, we'll need to set up the server side components. The ASHX handlers that will be responsible for the back end server side calls.
- Go to the solution explorer, and right click and add new item of type Generic Handler



- Name it WebCaptureHandler.ashx
- That handler will have some default content. You need to edit the WebCaptureHandler.ashx and wipe out its content and replace it all with this:

```
using System; using System.Collections.Generic; using System.Linq; using System.Web;
using Atalasoft.Imaging.WebControls.Capture; using System.IO; namespace
WingScan_withViewer { public class WebCaptureHandler : WebCaptureRequestHandler { public
WebCaptureHandler() { } } }
```

- Go to the solution explorer, and right click and add new item of type Generic Handler



- Name it WebDocViewerHandler.ashx
- That handler will have some default content. You need to edit the WebCaptureHandler.ashx and wipe out its content and replace it all with this:

```
    this.FileUpload += new FileUploadEventHandler(delegate(object sender,
FileUploadEventArgs e) { e.Cancel = true; });

} }
```

- Ok, you've got the prerequisites all set. Now we need to initialize the scanning component
- Go back to default.aspx and in the header and just after the call to the scripts we included above, we need to add a manual script block

```
<script type="text/javascript"> // note, we anchor the viewer objects to the root of the
dom here, but can't define these inline.. // they will be initialized inside a default
page load function below var _viewer = null; var _thumbs = null; // Initialize Web
Scanning and Web Viewing... this will run at startup $(function () { _viewer = new
Atalasoft.Controls.WebDocumentViewer({ parent: $('#_containerViewer'), toolbarparent:
$('#_toolbar1'), serverurl: '/WebDocViewerHandler.ashx', allowannotations: true,
showscrollbars: true, forcepagefit: true }); _thumbs = new
Atalasoft.Controls.WebDocumentThumbnailer({ parent: $('#_containerThumbs'), serverurl:
'/WebDocViewerHandler.ashx', allowannotations: true, viewer: _viewer, allowdragdrop:
true, showscrollbars: true }); try { Atalasoft.Controls.Capture.WebScanning.initialize({
handlerUrl: 'WebCaptureHandler.ashx', onUploadCompleted: function (eventName, eventObj) {
if (eventObj.success) { _thumbs.OpenUrl("atala-capture-upload/" +
eventObj.documentFilename); Atalasoft.Controls.Capture.CaptureService.documentFilename =
eventObj.documentFilename; } }, onScanError: scanErrorHandler, scanningOptions: {
applyVRS: false, pixelType: 0, showScannerUI: true } }); } catch (error) { alert('Thrown
error: ' + error.description); Atalasoft.Controls.Capture.WebScanning.dispose(); } });
function scanWithSelectedScanner() {
Atalasoft.Controls.Capture.WebScanning.scanningOptions.scanner =
$('.atala-scanner-list').val(); Atalasoft.Controls.Capture.WebScanning.scan(); } // This
is an updated handler with more sophisticated handling of certain errors function
scanErrorHandler(msg, params) { switch (msg) { case
Atalasoft.Controls.Capture.Errors.noTwain: alert("This web capture application requires
that you have at least one " + "valid TWAIN device installed and configured. Please
ensure you install " + " the TWAIN driver for your scanner device(s) then re-visit this
site."); break; case Atalasoft.Controls.Capture.Errors.oldWindowsService: alert("The
version of Kofax WebCapture Service running on the multiuser " + "system you're using is
out of date. Please contact your system administrator."); break; case
Atalasoft.Controls.Capture.Errors.webServiceMissed: alert("The Kofax WebCapture Service
on the multiuser system you're using " + "is not running. Please contact your system
administrator."); break; case Atalasoft.Controls.Capture.Errors.noPlugin: var
installerUrl = window.location.protocol + "//" + window.location.host + "/WebCapture/" +
params.filename; window.open(installerUrl, '_downloadService'); alert( "The WingScan Web
Scanning service is not available. \n\n" + "If the download doesn't start automatically,
please download from " + installerUrl + "\n\n" + "If you are not prompted to install, the
service may be installed but not running. \n\n" + "enable it by running from START->All
Programs->Kofax Web Capture Service->Kofax Web Capture Service \n\n" + "Refresh your
browser when completed."); break; case Atalasoft.Controls.Capture.Errors.oldPlugin: var
installerUrl = window.location.protocol + "//" + window.location.host + "/WebCapture/" +
params.filename; window.open(installerUrl, '_downloadService'); alert( "The WingScan Web
Scanning service is out of date.<br />" + "If the download does not begin automatically,
please go to " + params.filename); break; default: alert(msg); break; } } </script>
```

- OK, now, you've initialized the scanning and set it up for use

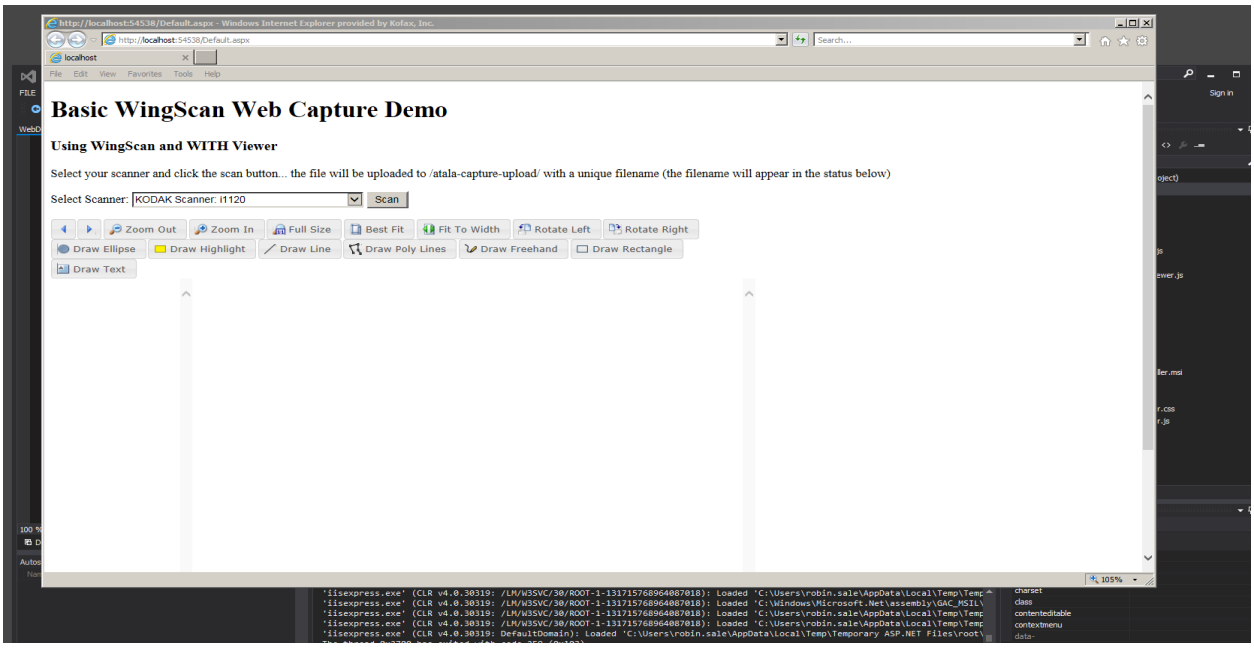
  Inside the Form section of the page, paste in this code

```
<div id="scanDiv"> <h1>Basic WingScan Web Capture Demo</h1> <h3>Using WingScan and WITH
Viewer</h3> <p>Select your scanner and click the scan button... the file will be uploaded
to /atala-capture-upload/ with a unique filename (the filename will appear in the status
below)</p> <p> </p> <p>Select Scanner: <select class="atala-scanner-list"
disabled="disabled" name="scannerList" style="width:22em"> <option
selected="selected">(no scanners available)</option> </select>   <!-- DO NOT USE <input
type="button" class="atala-scan-button" value="Scan" /> --> <input type="button"
id="scanNOW" value="Scan" onclick="scanWithSelectedScanner(); return false;" /> </p>
</div> <div style="width: 900px;"> <div id="_toolbar1"></div> <div id="_containerThumbs"
style="width: 180px; height: 600px;display: inline-block;"></div> <div
id="_containerViewer" style="width: 710px; height: 600px; display: inline-block;"></div>
</div>
```

- OK, now if you run the project it will likely throw you an error message saying the scanning service is not installed/available. This is because you do not have the scanning service installed on your local machine.
- In later sections we'll talk about gracefully handling this type of error message, but for now, just go ahead and navigate to:

  C:\Program Files (x86)\Atalasoft\DotImage 11.4\bin\WebResources\WebCapture\

  or the WebCapture folder of your app and run Kofax.WebCapture.Installer.msi
- Now, go ahead and run the app in Visual Studio
- If the app starts up and gives you our rather ugly page that shows a list of scanners and a scan button, congratulations!



- Pick a scanner and scan
- If you get any errors in the msg window, read them for a hint as to the root cause. You must have at least one TWAIN scanner installed on your system to use the application
- Assuming you have a scanner, select it and hit scan.
- The scan should start and when done the file will open in the viewer
- Congratulations! You've built a basic wing scan app with our WebDocumentViewer.

MAKING IT BETTER

So you can scan and view. The viewing components do not allow a lot of customization without a license for DotImage Document Imaging. However, one thing that you may want to to (and can do with this level of licensing) is enable saving. So, now, let's modify the app to enable the save button so you can annotate and rotate and re-order your thumbs and then save.

- Right click on the base project in solution explorer and add new folder. Name it "Saved"
- Edit the JavaScript in the Default.aspx page. In the section for the _viewer: add the config option

  savepath: 'Saved/',
- Also, for fun lets add a handler to notify on success/failure of save:

  _viewer.bind('documentsaved', function(e) { alert('Document Save: ' + e.success); });

  It will now look something like this:

```
_viewer = new Atalasoft.Controls.WebDocumentViewer({ parent: $('#_containerViewer'),
toolbarparent: $('#_toolbar1'), serverurl: '/WebDocViewerHandler.ashx', savepath:
'Saved/', allowannotations: true, showscrollbars: true, forcepagefit: true });
_viewer.bind('documentsaved', function(e) { alert('Document Save: ' + e.success); });
```

- Re-run the app and now when you scan, annotate, rotate etc... you can hit the save button and should be notified of successful saves
- The file will be saved to the Saved directory with its original filename. A file will also be saved alongside it with the same name but an .xmp extension. This .xmp file will contain the annotation data.

SAMPLE APP DOWNLOADS

It is our hope that you've been following along with the tutorial and have successfully built

your solution. However, if you've run into issues or if you want a working reference app, we've implemented both the WingScan_noViewer and WingScan_withViewer for you to download and run if needed. They also make great test harnesses for any experimental WingScan code you want to try out... letting you start from a known-working application.

- BasicWebCapture_noViewer (v11.5.0.10)
- BasicWebCapture_withViewer (v11.5.0.10)

CONCLUSION

It is possible to seriously customize the viewer and the saving, but you would need a license for DotImage Document Imaging and a bit more information to do that. The upcoming WebDocumentViewer Whitepaper will go into more details about the WebDocumentViewer. For now, we've shown you how to use the base WebCapture and WebDocumentViewer with WebDocumentThumbnailer controls using only a WingScan license.

Original Article

Q10462 - INFO: WingScan Whitepaper - Getting Started with Web Scanning (11.1 version)

Last Update

2026-11-11 - TD - (Moved Images)

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50039/INFO-WingScan-Whitepaper-Getting-Sta...