Working with SSL at Development Time is easier with IISExpress

April 21, 2011 Comment on this post [40] Posted in ASP.NET | ASP.NET MVC | IIS | VS2010

One of the demos in my Mix 11 talk "An Overview of the MS Web Stack of Love" was showing how IIS Express and Visual Studio SP1 (as well as WebMatrix) can make working with SSL (Secure Sockets Layer) a heck of a lot easier.

If you've used Cassini before (that's the little built in Visual Web Developer Server) you've likely noticed that I doesn't support SSL. This makes working with real world sites a little challenging. If you want your Login pages and Account Management pages to use secure sockets, you'd typically have to do all your work with the full version of IIS, either installed on your own machine or using a shared server.

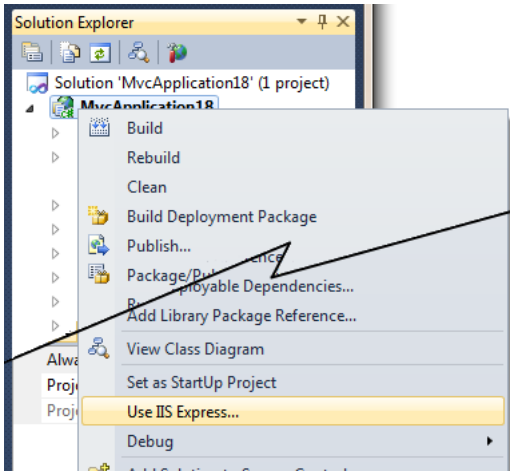
Here's a few ways to enable SSL. The first is **new** in Visual Studio 2010 SP1 and will allow you to use SSL on local host over ports 44300 and higher. This means you'll be able to test and develop how your site will work over SSL, but not over port 443 proper. I'll show you that in the final step.

If you watch the Mix video, you'll see towards the end where Damian Edwards educates me on this new SSL feature in VS2010SP1. I didn't know that VS2010SP1 (WebMatrix does also) installs some self-signed certificates and includes and option for turning on their use. However, as I pointed out in the video, that's only for high "strange" ports like 44300+, so my more complex example still has value if you want standard port numbers.

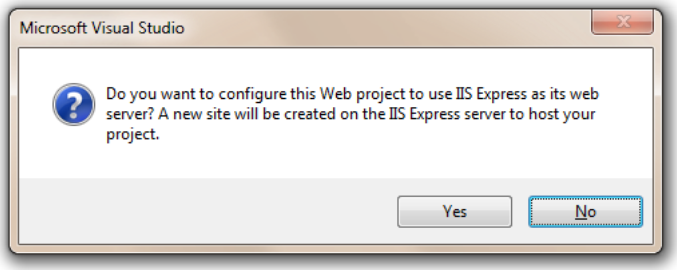
The Easy Way - Local SSL with IIS Express and VS2010 or WebMatrix

If you have IIS Express and VS2010SP1, you can do this now and follow along. Make a new ASP.NET Site in Visual Studio.

Right click on the Project in Solution and select **Use IIS Express**. You can also set IIS Express as the default from the **Tools | Options | Projects and Solutions | Web Projects**.



Next, click Yes, and VS will "make a new site" on IIS Express. **What does that mean?**



Click yes and let's find out.

Remember that IIS Express is really IIS. It's just "local personal not-a-service" IIS. That means that IISExpress puts its config files in **C:\Users\YOU\Documents\IISExpress\config** rather than in some machine-wide location.

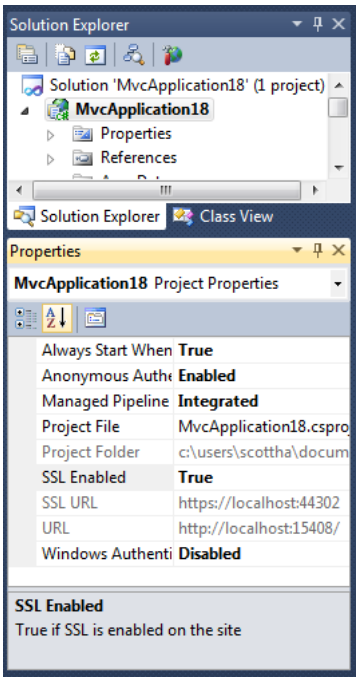
My project is called "MvcApplication18" so I can open up my ApplicationHost.config and look for "MvcApplication18." You can too. There's my site, right there, in IISExpress's applicationHost.config:

```
<site name="MvcApplication18" id="39">
    <application path="/" applicationPool="Clr4IntegratedAppPool">
        <virtualDirectory path="/" physicalPath="c:\users\scottha\documents\visual studio 2010\Projects\MvcApplication18\MvcApplication18" />
    </application>
    <bindings>
        <binding protocol="http" bindingInformation="*:15408:localhost" />
    </bindings>
</site>
```

Note the binding section. I can see that my site will show up on http://localhost:15408.

Go back to Visual Studio, click on your project and press F4 to bring up the properties dialog. You can also press Ctrl-W, then P, or select View | Property Window.

Since I'm using IIS Express and I have VS2010 SP1 installed, I have a new option, "SSL Enabled." If I click it, a new "SSL URL" shows up with a new port number chosen from that pool of ports I mentioned before.

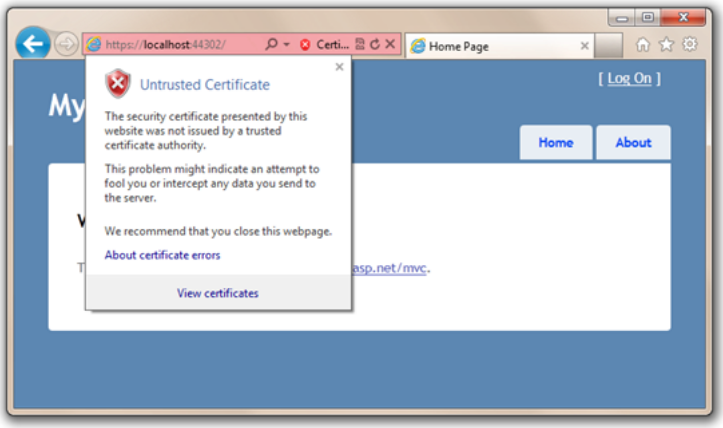


Go back over to your ApplicationHost.config if you want to see what really happened.

```
<site name="MvcApplication18" id="39">
<application path="/" applicationPool="Clr4IntegratedAppPool">
    <virtualDirectory path="/" physicalPath="c:\users\scottha\documents\visual studio 2010\Projects\MvcApplication18\MvcApplication18" />
</application>
<bindings>
    <binding protocol="http" bindingInformation="*:15408:localhost" />
    <binding protocol="https" bindingInformation="*:44302:localhost" />
</bindings>
```

See that new binding? That was created for us when we clicked SSL Enabled = True.

Run your site. Visit it with and without SSL. Don't forget the port number! You're now running under SSL locally, but you're reminded you are a bad person because this certificate is not trusted. Still, create an app, check a box and you've got local SSL.



Ok, how can we get this running in a slightly better way? I want:

- **A friendly machine name, not localhost.**
- **People to be able to talk to my instance of IIS Express from the outside.**
- **Actual SSL over port 443.**
- **My ASP.NET application to switch between SSL and not automatically when I'm logging in.**
- **My self-signed certificate to be trusted so I don't get warnings.**
- **To use PowerShell at some point for no reason at all because that's bad-ass.**

Here we go.

The ~~Hard~~ Ninja Way - Local SSL over 443 with IIS Express and the Gracious Manatee that is The Command Line

These steps may seem a little scary, but it's useful to know that they are happening (or have happened) already to make the Easy Way work for you. I'll show you how to do it yourself, then I'll show you an undocumented way to make *part* of The Hard Way even easier. It's important to know what's happening though and why when you start running random commands from an Administrator Command Prompt, right?

1. Getting IIS Express to serve externally over Port 80

My machine is called HANSELMAN-W500, so I'll use that name. You could update your hosts file and use a friendly name. To start, use your computer name. if you don't know the name of your computer, you're silly. Go to the command prompt and type "HOSTNAME" to find out.

First, we need to tell HTTP.SYS at the kernel level that it's OK to let everyone talk to this URL by making an "Url Reservation." From an administrative command prompt:

```
netsh http add urlacl url=http://hanselman-w500:80/ user=everyone
```

Next, as I want to be able to talk to IIS Express from outside (folks on my network, etc. Not just localhost) then I need to allow IIS Express through the Windows Firewall. I can do that graphically from Windows, or type:

```
netsh firewall add portopening TCP 80 IISExpressWeb enable ALL
```

Finally, I need to make sure that my project will use Port 80. I can do that one of two ways. I can either edit the applicationHost.config manually and add the binding (my recommended way):

```
<site name="MvcApplication18" id="39">
    <application path="/" applicationPool="Clr4IntegratedAppPool">
        <virtualDirectory path="/" physicalPath="c:\users\scottha\documents\visual studio 2010\Projects\MvcApplication18\MvcApplication18" />
    </application>
    <bindings>
        <binding protocol="http" bindingInformation="*:15408:localhost" />
        <binding protocol="https" bindingInformation="*:44302:localhost" />
        <binding protocol="http" bindingInformation="*:80:hanselman-w500" />
    </bindings>
</site>
```

Or, I can do that from the command line too! Although it's a little scary. I can confirm my changes in ApplicationHost.config though if I mess up.

```
"c:\Program Files (x86)\IIS Express\appcmd.exe" set site /site.name:MvcApplication18 /+bindings.[protocol='http',bindingInformation='*:80:hanselman-w5
```

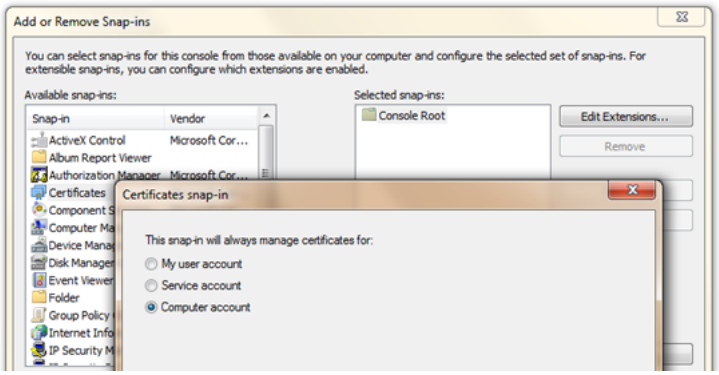
Notice that I'm using the appcmd.exe that came with IIS Express. I don't want to mess up my actual IIS installation if I have one.

2. Making an SSL Cert, hooking it up to IIS Express and making it Trusted

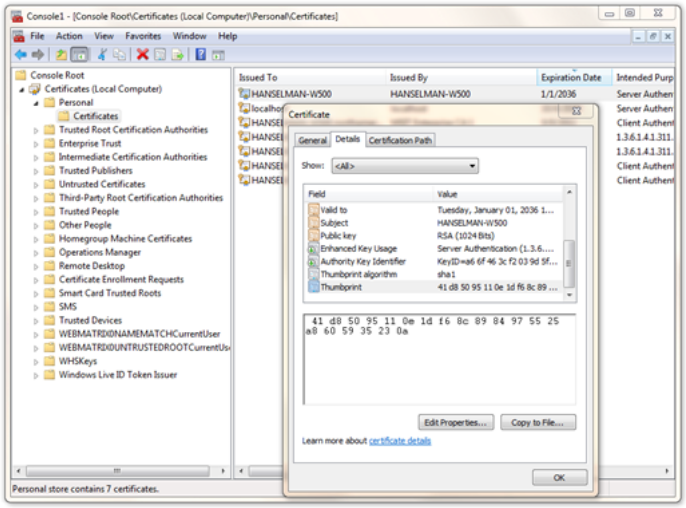
Let's make a SSL certificate of our own. Note the CN=. I'm making it my Computer Name, but you could make it nerddinner.com or whatever makes you happy. It should line up with whatever name you've been using so far.

```
makecert -r -pe -n "CN=HANSELMAN-W500" -b 01/01/2000 -e 01/01/2036 -eku 1.3.6.1.5.5.7.3.1 -ss my -sr localMachine -sky exchange -sp "Microsoft RSA SCh
```

Now, a tricky part. Go find this certificate in the Certificate Manager. Run MMC.exe, go File | Add/Remove Snap In, then select Certificates. Pick the Computer Account. (This is why you can't just run certmgr.msc) and add it.



It'll likely be the certificate with an expiration data of 1/1/2036 under Personal Certificates. Double click on your certificate. Go to Details, and scroll down to Thumbprint. Copy that into the clipboard, as that identifies our new certificate.



Remove all the spaces from that Thumbprint hash. You can remove those spaces with Notepad if you're Phil Haack, or in PowerShell if you're awesome:

```
C:\>"41 d8 50 95 11 0e 1d f6 8c 89 84 97 55 25 a8 60 59 35 23 0a" -replace " "
41d85095110e1df68c8984975525a8605935230a
```

Take the hash and plug it in to the end of THIS command:

```
netsh http add sslcert ipport=0.0.0.0:443 appid={214124cd-d05b-4309-9af9-9caa44b2b74a} certhash=YOURCERTHASHHERE
```

The AppId doesn't really matter, its just a GUID. This tells HTTP.SYS that we're using that certificate. Leave the Certificate Manager MMC running.

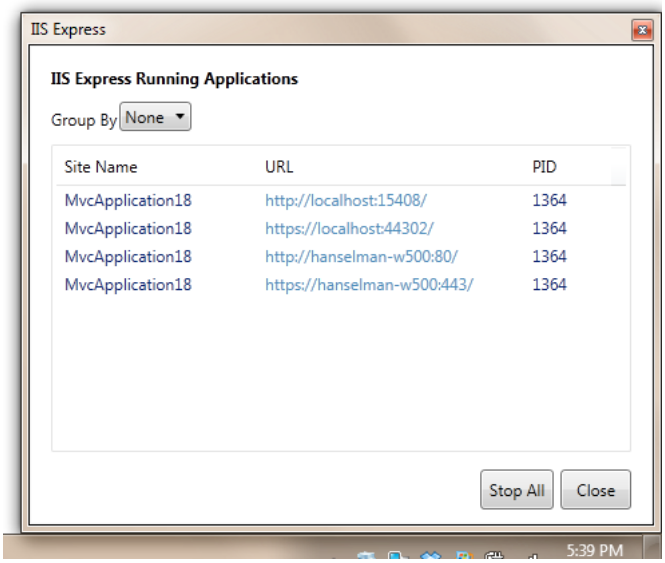
Now, tell HTTP.SYS that we're cool with port 443 also (we told it that 80 was cool a minute ago, remember?):

```
netsh http add urlacl url=https://hanselman-w500:443/ user=Everyone
```

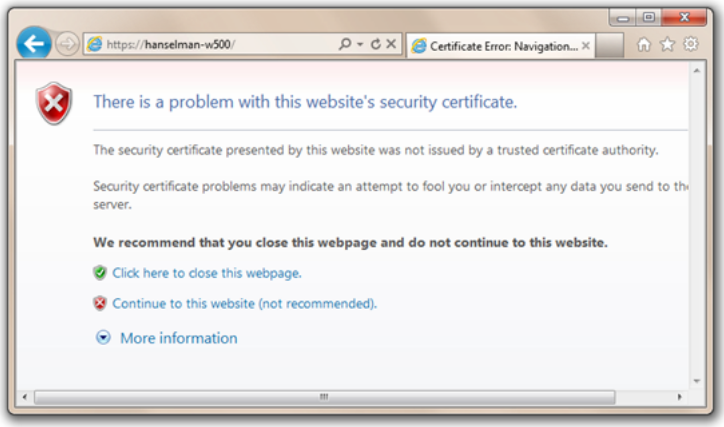
Return to your applicationHost.config and add the 443 binding for your site:

```
<site name="MvcApplication18" id="39">
  <application path="/" applicationPool="Clr4IntegratedAppPool">
     <virtualDirectory path="/" physicalPath="c:\users\scottha\documents\visual studio 2010\Projects\MvcApplication18\MvcApplication18" />
  </application>
  <bindings>
    <binding protocol="http" bindingInformation="*:15408:localhost" />
    <binding protocol="https" bindingInformation="*:44302:localhost" />
    <binding protocol="http" bindingInformation="*:80:hanselman-w500" />
    <binding protocol="https" bindingInformation="*:443:hanselman-w500" />
  </bindings>
</site>
```

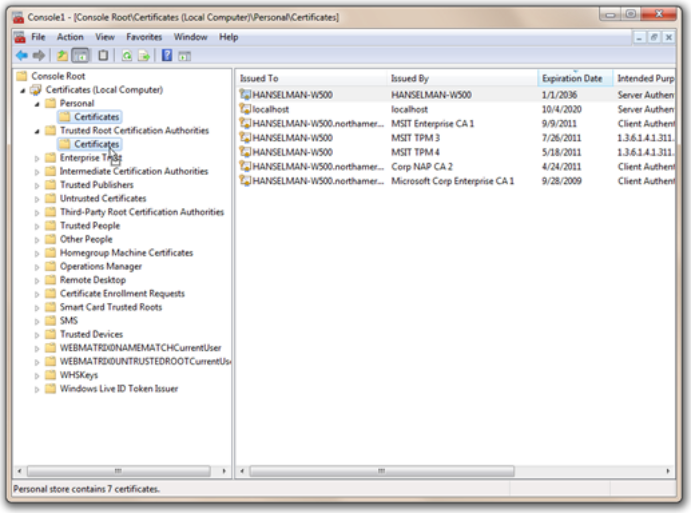
If I say "Show All Sites" from the IIS Express tray icon, I'll see my site(s) and the URLs they are bound to.



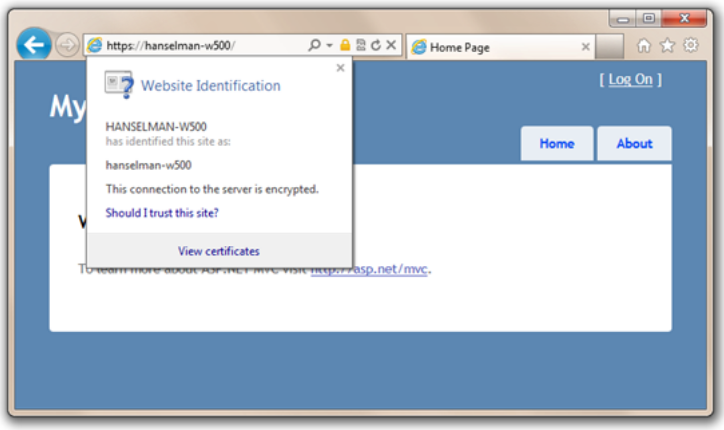
Now I can visit https://hanselman-w500, but again I get a certificate error.



Go back to the CertMgr MMC, and drag your self-signed SSL Certificate from Personal into Trusted Root Certificates.



Suddenly my local SSD site is legit! Very cool.



3. Getting ASP.NET to force SSL with an URL Rewrite Rule

One of the things Cassini (Visual Studio Web Developer Server) can't do is UrlRewriting. I want my app to force SSL when I hit /account/logon or /account/register. I'll add this to the first node of system.webServer in my app's web.config:

```
<rewrite>
  <rules>
    <rule name="Redirect to HTTPS" stopProcessing="true">
      <match url="^account/logon$|^account/register$" />
      <conditions>
        <add input="{HTTPS}" pattern="^OFF$" />
      </conditions>
      <action type="Redirect" url="https://{HTTP_HOST}/{R:0}" redirectType="Permanent" />
    </rule>
  </rules>
</rewrite>
```

I could also use the RequireHttps attribute on my controllers if I like.

Appendix Z: A totally undocumented way to make part of this easier that you use at your own risk

There's a command line helper deep inside of the IIS Express directory that I never mentioned to you. We never spoke! I don't know you. Who is this? Stop calling! ;)

```
C:\Program Files (x86)\IIS Express>IisExpressAdminCmd.exe
Usage: iisexpressadmincmd.exe <command> <parameters>
Supported commands:
      setupFriendlyHostnameUrl -url:<url>
      deleteFriendlyHostnameUrl -url:<url>
      setupUrl -url:<url>
      deleteUrl -url:<url>
      setupSslUrl -url:<url> -CertHash:<value>
      setupSslUrl -url:<url> -UseSelfSigned
      deleteSslUrl -url:<url>

Examples:
1) Configure "http.sys" and "hosts" file for friendly hostname "contoso":
iisexpressadmincmd setupFriendlyHostnameUrl -url:http://contoso:80/
2) Remove "http.sys" configuration and "hosts" file entry for the friendly
hostname "contoso":
iisexpressadmincmd deleteFriendlyHostnameUrl -url:http://contoso:80/
```

From the command line with this utility, I can quickly setup my hosts file and my HTTP.SYS Url ACLs with one command:

```
C:\Program Files (x86)\IIS Express>IisExpressAdminCmd.exe setupFriendlyHostnameU
rl -url:http://daddyisawesome:80/
```

```
Command 'setupFriendlyHostnameUrl' completed.
```

And remove them:

```
C:\Program Files (x86)\IIS Express>IisExpressAdminCmd.exe deleteFriendlyHostname
Url -url:http://daddyisawesome:80/
Command 'deleteFriendlyHostnameUrl' completed.
```

At this point you just need to update the IISExpress applicationHost.config with the correct binding. You can also use IISExpressAdminCmd setupSslUrl with SSL ports that are already reserved. However, I really think The Hard Way is best because you can really see what's going on, and you have more control.

Make It Stop!

How do I undo it all? Delete the Certificate in CertMgr, and from an Administration Console:

```
netsh http delete sslcert ipport=0.0.0.0:443
netsh http delete urlacl url=http://hanselman-w500:80/
netsh http delete urlacl url=https://hanselman-w500:443/
```

If you have existing SSLCerts registered with HTTP.sys, the adjust these commands.

Enjoy! Thanks to CarlosAG for his help with this post.

About Scott

Scott Hanselman is a former professor, former Chief Architect in finance, now speaker, consultant, father, diabetic, and Microsoft employee. He is a failed stand-up comic, a cornrower, and a book author.

About Newsletter

Disclaimer: The opinions expressed herein are my own personal opinions and do not represent my employer's view in any way.