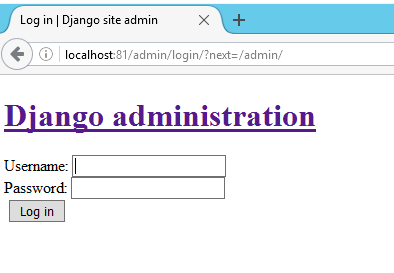
Set up Static File for Django website in IIS for Production:

* **Configure Django and IIS to Serve Static Files**

Now we have things more or less working but we aren't quite done. If you browse to the Django admin page (http://localhost:81/admin), you'll see something like this:



The reason things look a little strange is because IIS doesn't know where things like the CSS and image files for the Django application are located. In the Django world these assets are referred to as "static files," and we need to perform a couple of additional steps for IIS to be able to serve these files. This isn't an issue during development since Django has a static file server built into it, but in production we'll want IIS to handle serving these files.

How Django handles static files is a bit of a topic in and of itself, so for further information please consult the Django documentation (<https://docs.djangoproject.com/en/1.9/howto/static-files/>). In this guide we'll focus on a specific set of steps to get things working without getting into too many of the behind-the-scenes technical details, the different philosophies on where static files can or should be placed, or the various and sundry ways in which this aspect of Django applications can be addressed.

### Add a STATIC\_ROOT Setting to the Django Settings File

First, your Django application's settings file needs to include a STATIC\_ROOTsetting, and this setting is not in Django applications by default. The STATIC\_ROOT setting is used when Django's collectstatic management command is run to determine where to place the collected static files.

In our sample foo application, we'll tell Django to collect the static files in a static directory located in the same directory as the settings.py file. (You can ultimately put your static directory wherever you wish, but this is where we'll be putting it for the purposes of this example.)

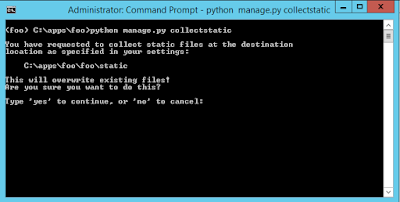
To accomplish this, do the following:

1. Using a plain text editor such as Notepad, or Idle if that was installed when you installed Python, open the file C:\apps\foo\foo\settings.py
2. Scroll to the bottom of the file, or use the find feature of your text editor, and find the STATIC\_URL setting
3. Above the STATIC\_URL setting, add the following setting:  
   STATIC\_ROOT = os.path.abspath(os.path.join(BASE\_DIR, 'foo', 'static'))
   1. The placement of the STATIC\_ROOT setting in the settings.py file doesn't matter, but putting it right next to the STATIC\_URL setting is typical and keeps all the settings related to static files in one place.
   2. The STATIC\_ROOT setting we're using as our example will put the static files in C:\apps\foo\foo\static
4. Save the settings.py file

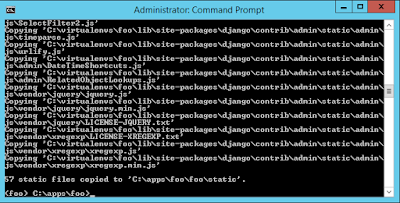
### Run the collectstatic Management Command

Now that Django knows where to put the static files, we can run the collectstatic management command. This command takes all the static files for your project, including the static files used by the Django admin and other packages in your virtual environment, and puts them under one staticdirectory, the location of which is the STATIC\_ROOT setting we added above, so they can be served by the web server.

1. Open a Command Prompt
2. Activate the application's virtual environment:  
   C:\virtualenvs\foo\Scripts\activate.bat [then press Enter]
3. Navigate to the application's root directory:  
   cd C:\apps\foo [then press Enter]
4. Run Django's collectstatic management command:  
   python manage.py collectstatic [then press Enter]
5. Django will then tell you the directory into which it's going to copy the static files, and ask you to confirm since this process overwrites any existing files in the directory. Type yes at this prompt and press Enter.



1. The static files will then be copied to the directory indicated by the STATIC\_ROOT setting.



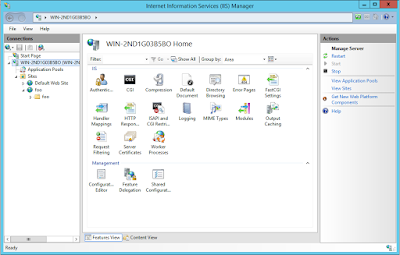
### Add a static Virtual Directory to the IIS Web Site

Now we have all our static files in one place, but we still haven't told IIS where they're located. We'll accomplish this by adding a static virtual directory to the IIS web site for our application.

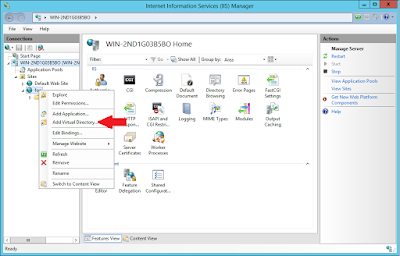
Note that the name of the virtual directory in IIS *must match* the value of the STATIC\_URL setting in the Django application's settings.py file, absent the beginning and trailing slashes. For our sample application we're using a value of /static/ for the STATIC\_URL setting, so the name of the virtual directory in IIS will be static.

Create a virtual directory in IIS as follows:

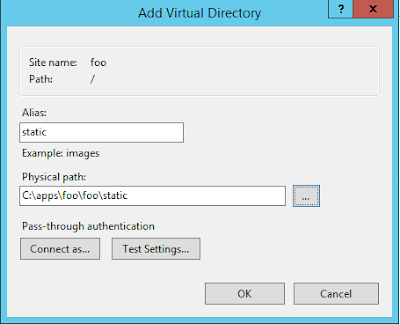
1. Open IIS Manager
2. On the left-hand side under "Connections," expand the server's tree
3. Expand the "Sites" folder
4. Expand the foo web site. At this point, things should look like this:



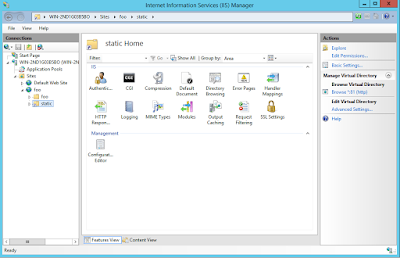
1. Right-click the foo web site and click "Add Virtual Directory"



1. In the Add Virtual Directory dialog, enter the following values:  
   Alias: static  
   Physical path: C:\apps\foo\foo\static



1. Click "OK." You'll see the static virtual directory appear under the web site.

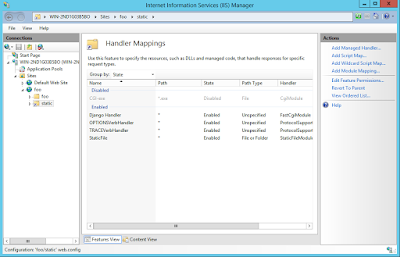


### Configure Handler Mappings for the static Virtual Directory

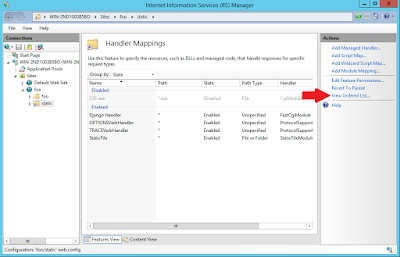
The last step is to tell IIS which handler to use to serve files out of the staticvirtual directory. Since at this point our Django Handler is set to serve all files for our application, even with the static virtual directory in place the Django Handler will still be attempting to serve these files.

IIS already has a static file handler active in the web site, but it's down further in the list of handlers than the global Django Handler we configured, so to get IIS to serve the static files under the static virtual directory we'll move the static file handler to the top of the handler list on the static virtual directory.

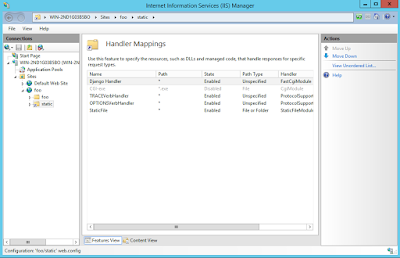
1. Open IIS Manager
2. Expand the server tree
3. Expand the foo web site
4. Click on the static virtual directory
5. Double-click the "Handler Mappings" icon. You'll see the following list of handler mappings:



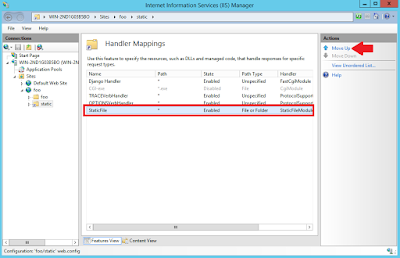
1. On the right-hand side under "Actions" click on "View Ordered List ..."



1. You'll now see a list of the handler mappings in the order in which they're run, with the Django Handler at the top.

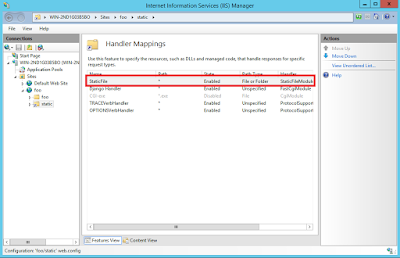


1. Click on the StaticFile handler at the bottom of the list, then click "Move Up" under "Actions" on the right-hand side.



* 1. Note at this point you may receive a warning that by moving handlers you are detaching this virtual directory from the inheritance of the parent's handler mappings. This is what we want to do, so if you get this warning click "Yes."

1. Continue to click "Move Up" until the StaticFile handler is at the top of the list of Handler Mappings.



With the StaticFile handler at the top of the list of Handler Mappings, requests that come into the static directory will be handled by the StaticFile handler as opposed to being processed by the Django Handler, so IIS will serve the static files now.

Confirm that everything is working by going to http://localhost:81/admin in a browser. You should now see the CSS being applied to the page.

