Djargon on Azure

Saturday, September 10, 2016 3:14 PM

Building Djargon Documentation Buttons on Azure

Introduction

This page describes and screen-caps adding a new button to this documentation website http://djargon.azurewebsites.net which is built using the Bottle web framework. This site is deployed on the Azure cloud using Visual Studio and it has buttons that link to static pdf files. For me this is a reasonably simple and fast way of publishing documentation from goals, screen captures and narratives. The various pdf documentation files concern common tasks on the cloud (AWS or Azure) and some details from implementation projects I am involved with.

What is a Web Framework?

Notice at the top of the djargon web page is a top bar including a bunch of links. One of these links is something like 'Web Frame'. Click that to read some documentation on building and deploying off of a Web Framework in Azure. If you'd like to skip that for now just keep reading here.

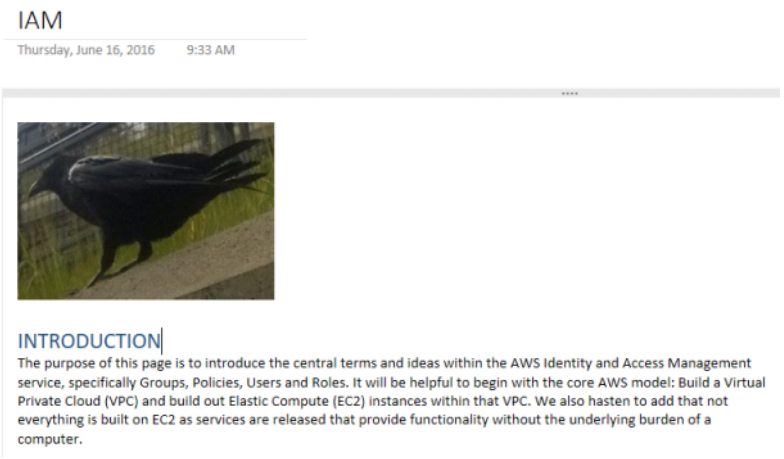
A Web Framework is a web application structure that takes care of a lot of the necessary structure so you don't have to. Since at UW we are big fans of Python we regularly use the Python web frameworks Bottle, Flask and Django. Django is the complicated (and powerful) one. Bottle is the one we use here. This document does not cover setting up the Bottle framework; we assume that is already done and that the standard files are in place. This narrative picks up with "ok now let's add a new button"; so first let's see what the buttons look like and create the content that will be connected to a new button.

Start by Creating a pdf

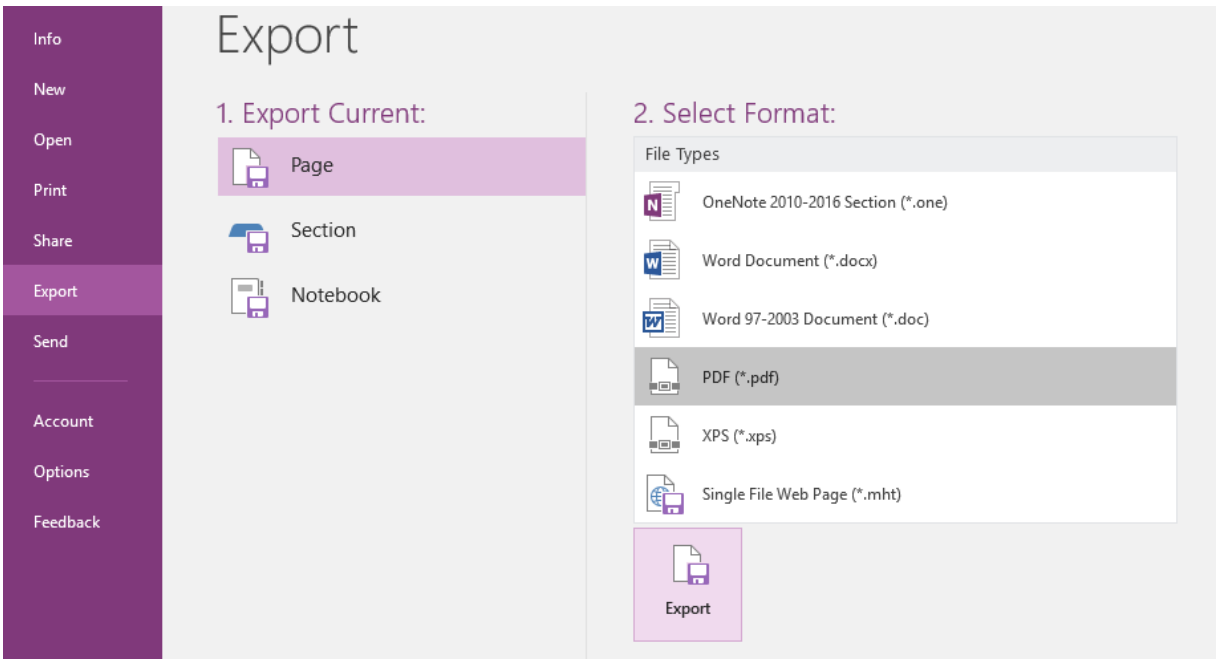
Here is what the AWS section looks like at the web page http://djargon.azurewebsites.net prior to adding a new button:

Let's add a button that will read simple 'IAM' which is short for Identity and Access Management. First we go through a process on the AWS Console that involves using IAM. I write this up in OneNote with a

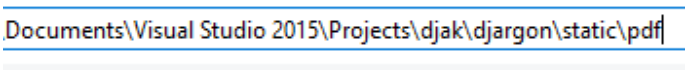
lot of screen captures. Here in turn is a screen capture of the OneNote page:



Suppose I have this page in OneNote in a state where it is ready to be published to the djargon website. Great: Let's save it as a pdf into the Visual Studio project file system. It goes into a folder of static content, sub-folder 'pdf'.

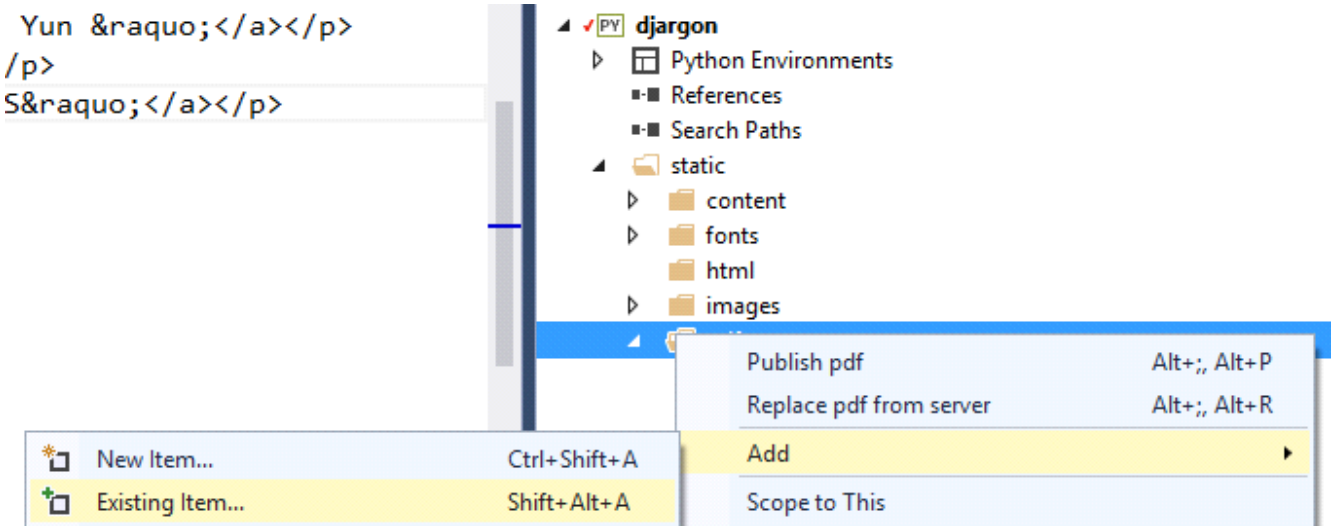


Here is the path where I save this file:

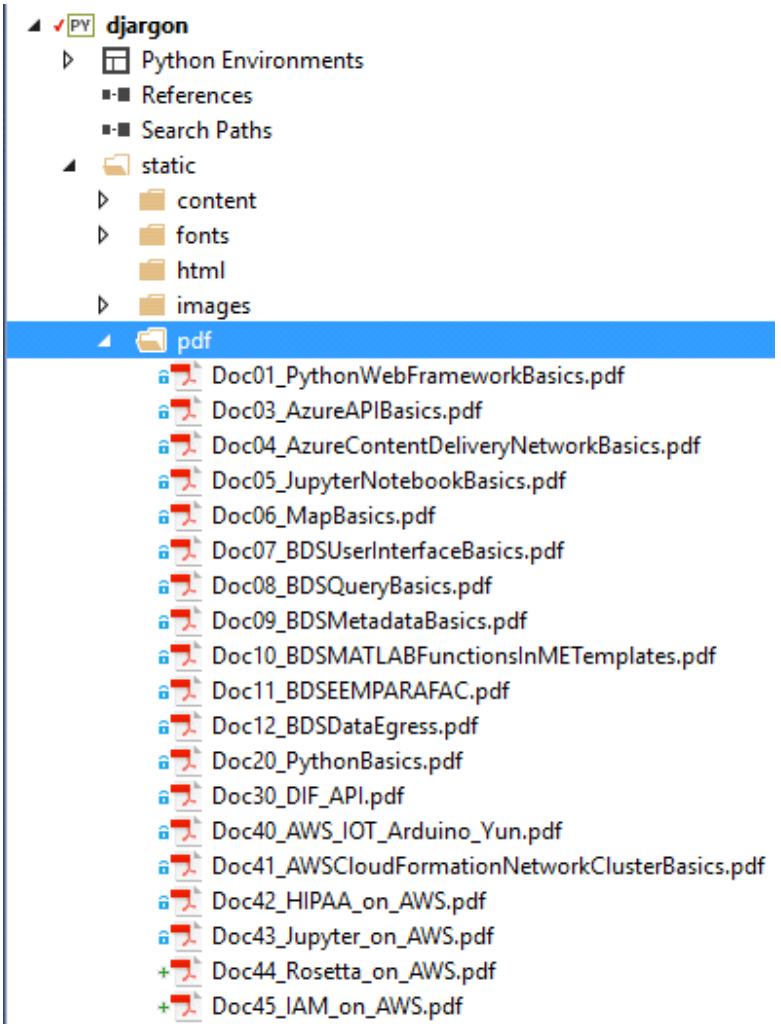


So 'djargon' is a Web Framework application within the 'djak' solution. In Visual Studio one solution can include multiple Projects and I think we would say that 'djargon' is such a project.

Now once a file is saved into the static/pdf folder of the Visual Studio project it is staged… but it is not yet part of the Solution. That is: Visual Studio must be instructed to include the file as part of the Solution; and this is done in the Solution Explorer: Right-click on 'static/pdf' folder >> Add >> Existing Item…



After this file is added you can see the file structure in Visual Studio.



Notice that each pdf file starts with a simple identifier to keep these files in order.

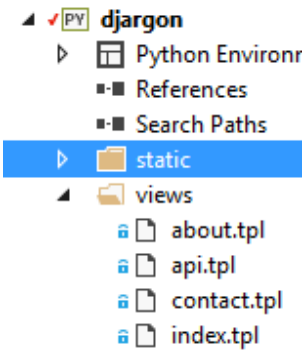
Intermezzo

So what is going on here is that content for the website has been created; and now we are moving it on to the web in three stages. The first stage is to move it into the Project file structure and 'register' it with the Visual Studio project. That is done above. What follows is creating code in two files that will cause

the button to appear and link to that pdf file. The third and final part is to Publish the project to the website.

Modifying Necessary Files

The file index.tpl is the first of two to modify. The ***.tpl*** extension just means 'template' (because they couldn't use '.tmp' of course). In VS Solution Explorer:



In the editor I simply took the line of code for 'Jupyter on AWS' and copied it twice. I am making two new buttons: One for IAM and another for something called 'Rosetta'. While I am here I am going to clean up the button labels a little bit; but for this tutorial all you care about is copy-paste-edit the entry for Jupyter to suit the new button. Notice that the button name is changed and the file reference is changed. My two new buttons will be labelled 'Rosetta' and 'IAM'.

```
    <div class="col-md-4">
        <h2>AWS Procedurals</h2>
        <p>Doing stuff in the Amazon cloud.</p>
        <p><a class="btn btn-default" href="pdf/Doc41_AWSCloudFormationNetworkClusterBasics">CFN Cluster &raquo;</a></p>
        <p><a class="btn btn-default" href="pdf/Doc40_AWS_IOT_Arduino_Yun">AWS IOT + Arduino Yun &raquo;</a></p>
        <p><a class="btn btn-default" href="pdf/Doc42_HIPAA_on_AWS">HIPAA on AWS&raquo;</a></p>
        <p><a class="btn btn-default" href="pdf/Doc43_Jupyter_on_AWS">Jupyter notebook on AWS&raquo;</a></p>
        <p><a class="btn btn-default" href="pdf/Doc44_Rosetta_on_AWS">Rosetta&raquo;</a></p>
        <p><a class="btn btn-default" href="pdf/Doc45_IAM_on_AWS">IAM&raquo;</a></p>
```

Blow up of important text:

```
href="pdf/Doc44_Rosetta_on_AWS">Rosetta&raquo;</
href="pdf/Doc45_IAM_on_AWS">IAM&raquo;</a></p>
```

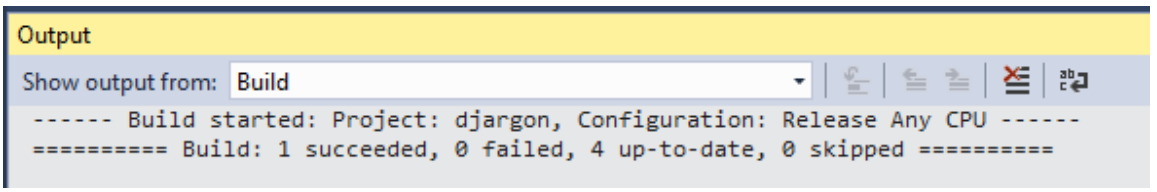
Now on to the second file, routes.py. This is a Python file that is used to connect URL details to files, in this case to our static pdf files. The block of code looks like this (notice it was written by me using my nom-de-code 'kilroy'):

```
# kilroy added these routes to bind to static pdf files
#   The better way to do this is with a /pdf/Name approach but I'm in a hurry at the moment.
@route('/pdf/<static_pdf_file>')
def pdf_chooser(static_pdf_file):
    theText = str(static_pdf_file);
    theBase = theText.split('.')[0]
    allowedBases = ['Doc01_PythonWebFrameworkBasics', \
                    'Doc03_AzureAPIBasics', \
                    'Doc04_AzureContentDeliveryNetworkBasics', \
                    'Doc05_JupyterNotebookBasics', \
                    'Doc06_MapBasics', \
                    'Doc07_BDSUserInterfaceBasics', \
                    'Doc08_BDSQueryBasics', \
                    'Doc09_BDSMetadataBasics', \
                    'Doc10_BDSMATLABFunctionsInMETemplates', \
                    'Doc11_BDSEEMPARAFAC', \
                    'Doc12_BDSDataEgress', \
                    'Doc20_PythonBasics', \
                    'Doc30_DIF_API',\
                    'Doc40_AWS_IOT_Arduino_Yun',\
                    'Doc41_AWSCloudFormationNetworkClusterBasics',\
                    'Doc42_HIPAA_on_AWS',\
                    'Doc43_Jupyter_on_AWS'\
                    ]
    if theBase in allowedBases:
        return static_file(theBase + '.pdf', root='./static/pdf/')
```

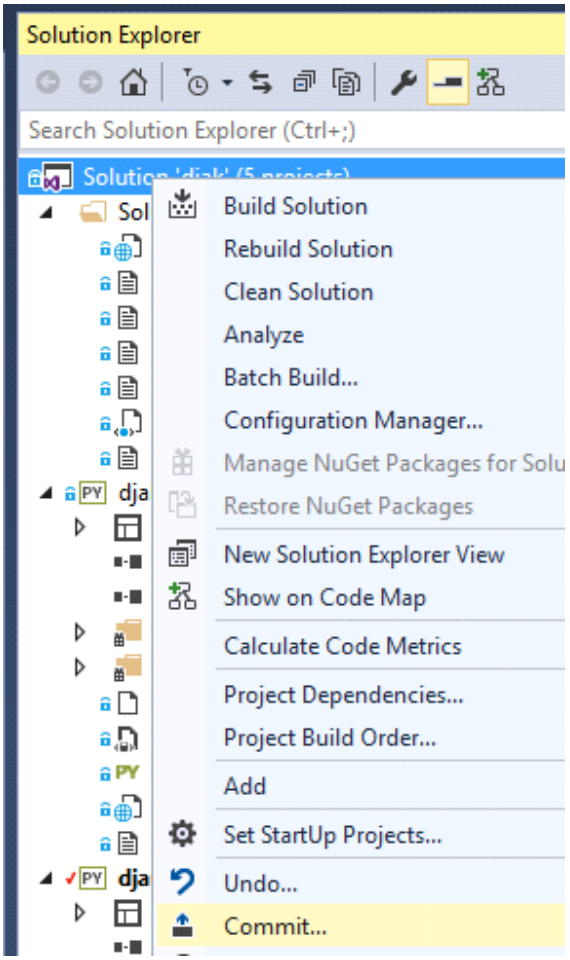
Clearly here we have a sequence of entries including the 'Doc43_Jupyter_on_AWS' entry. Each is followed by a trailing comma except for that last line; so I will need to be careful about adding new commas. When I am done creating entries for Rosetta and IAM the code looks like this:

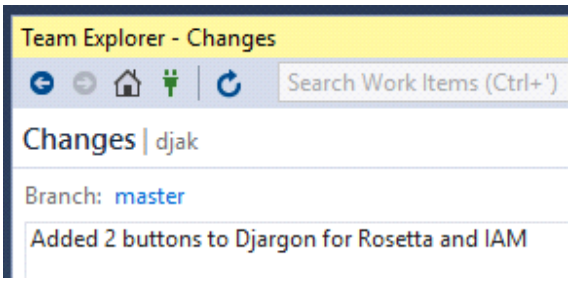
```
        'Doc42_HIPAA_on_AWS',\
        'Doc43_Jupyter_on_AWS',\
        'Doc44_Rosetta_on_AWS',\
        'Doc45_IAM_on_AWS'\
        ]
```

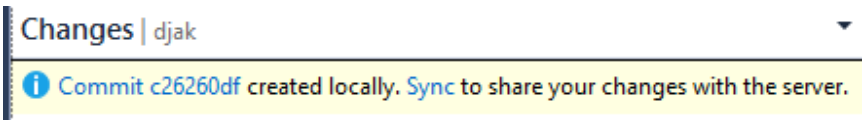
And that is all there is to it; just two lines of code. Now I will Build this Project… and that seems to go ok:

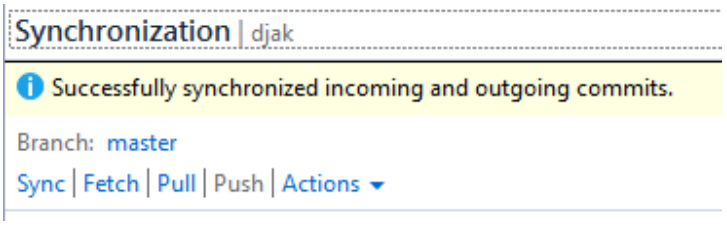


Now I will do a Commit of this code and Synch with Github before I forget to do this… I Commit from the Solution level rather than the Project level.

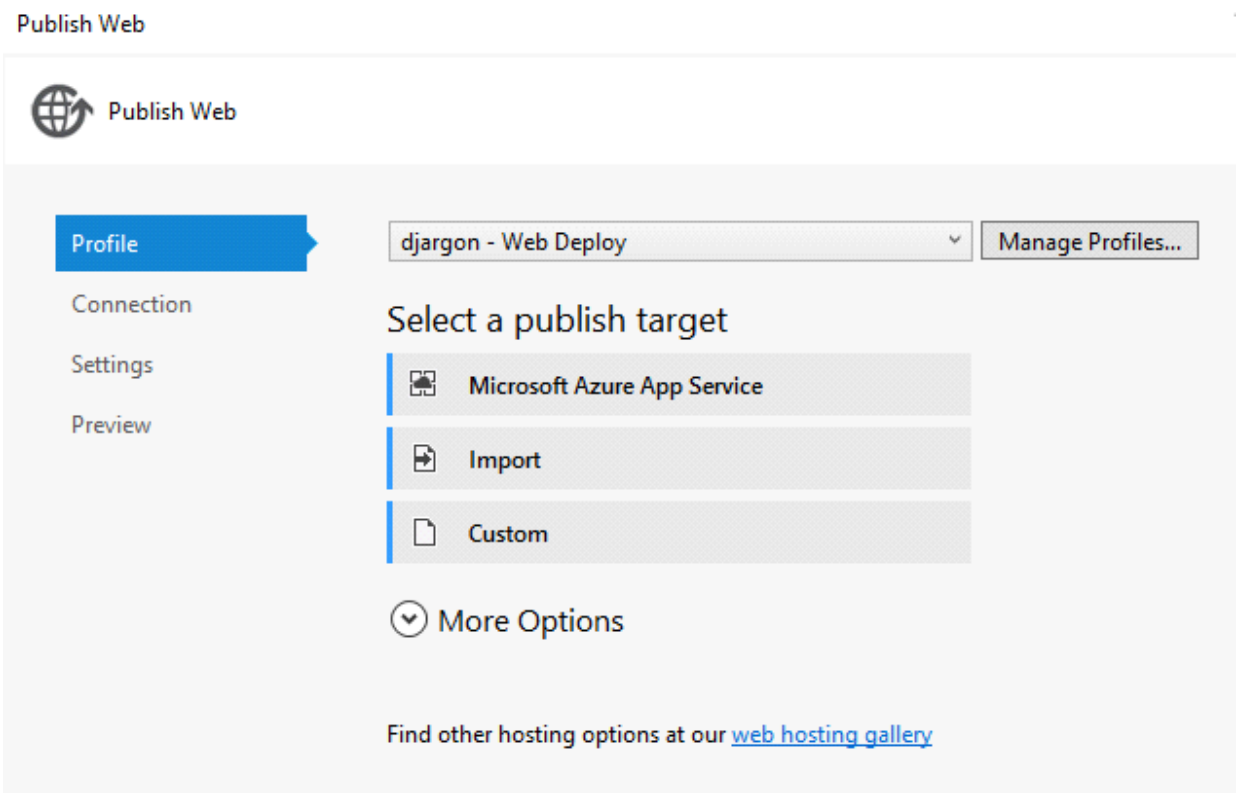


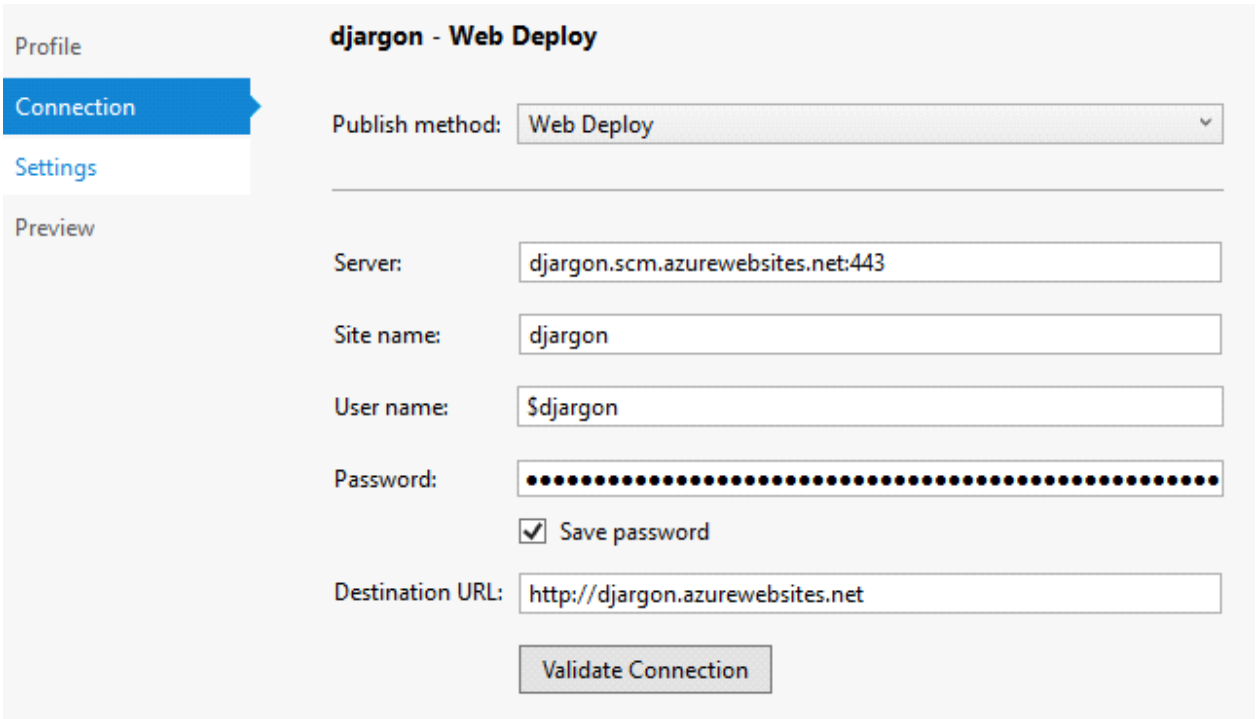


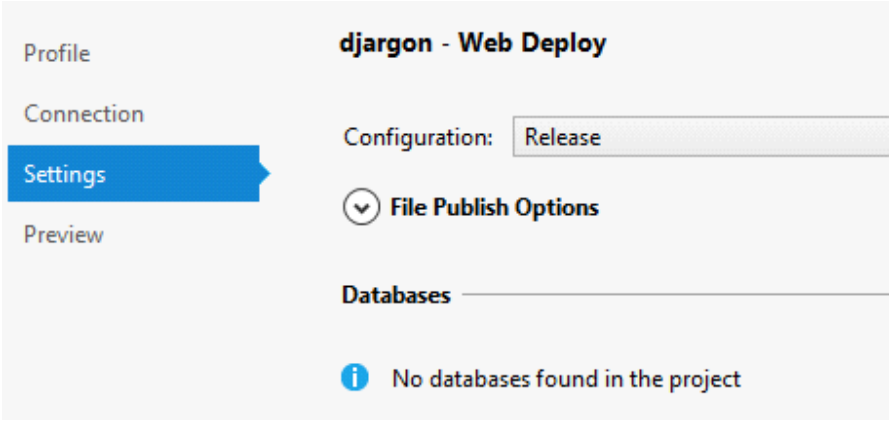


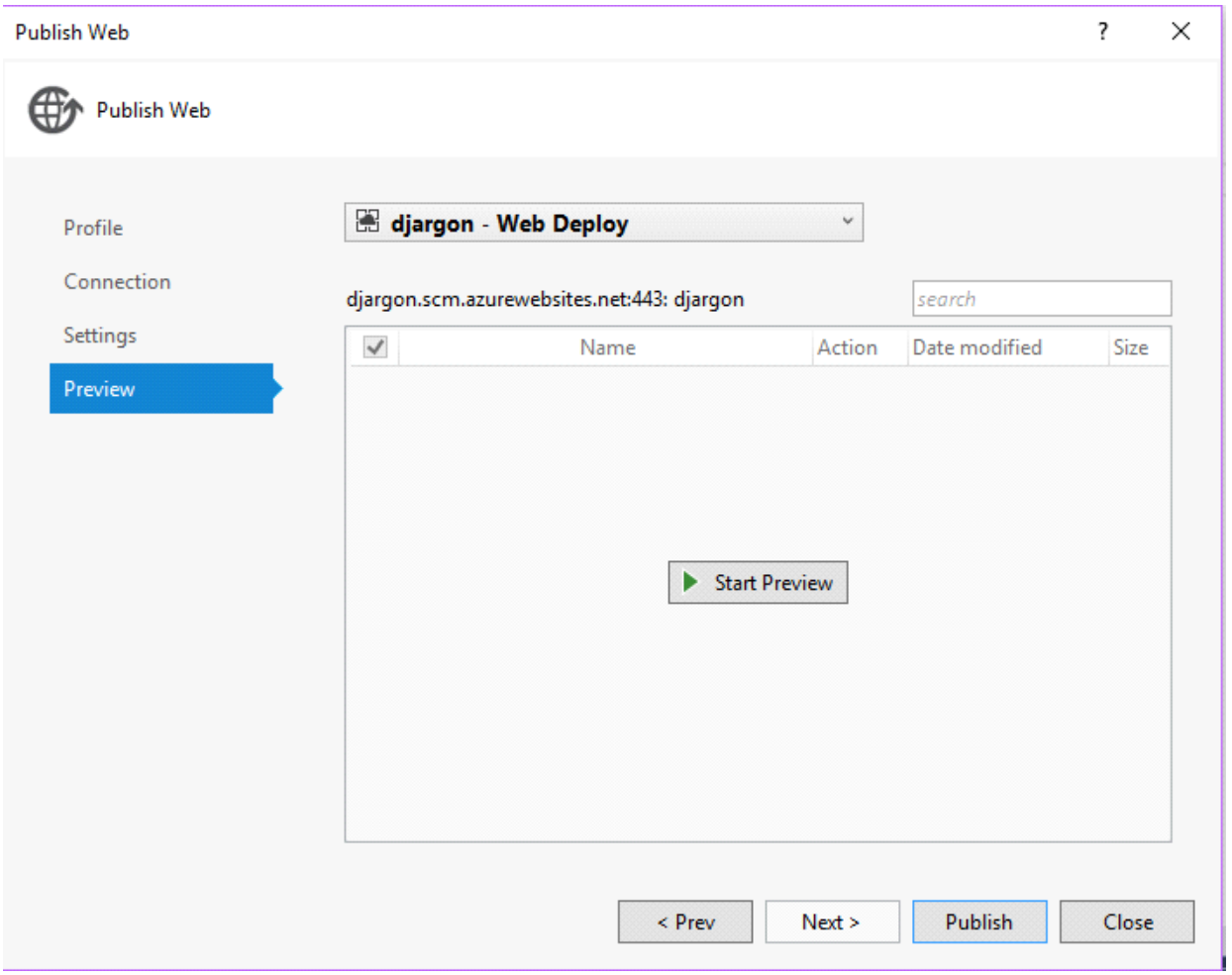


...and now finally let's Publish this and see if it shows up properly. I right click on 'djargon' in the Solution Explorer and select Publish... which brings up the Publish wizard. This has four tabs: Profile, Connection, Settings, Preview. I show these in sequence:

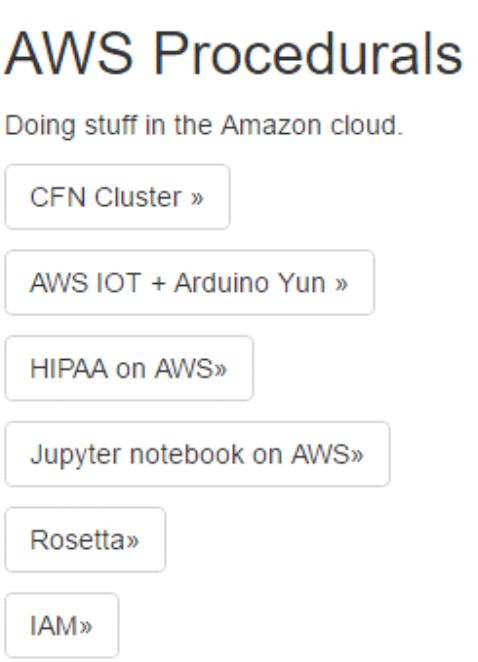

Publish Web
Publish Web
Profile
Connection
Settings
Preview
djargon - Web Deploy
Manage Profiles...
Select a publish target
Microsoft Azure App Service
Import
Custom
More Options
Find other hosting options at our web hosting gallery


Profile
Connection
Settings
Preview
djargon - Web Deploy
Publish method:
Web Deploy
Server:
djargon.scm.azurewebsites.net:443
Site name:
djargon
User name:
$djargon
Password:
Save password
Destination URL:
http://djargon.azurewebsites.net
Validate Connection





So I will cut to the chase and click Publish. This took five minutes or so… it was doing a lot of updates… and now the web page looks like this:



…and clicking on the IAM button…

So that is that.

Now to create an entry for this page.