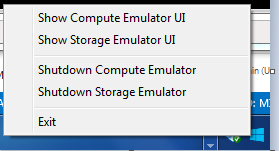
# Installation

1. If you don’t already have Visual Studio, MonoDevelop, or Xamarin Studio installed, install one of the following:
   1. F# Tools for Visual Studio Express 2012 for Web (and walk through of Type Providers) <http://blogs.msdn.com/b/fsharpteam/archive/2012/09/12/announcing-the-release-of-f-tools-for-visual-studio-express-2012-for-web.aspx>
   2. Xamarin Studio - <http://xamarin.com/studio>
   3. MonoDevelop - <http://monodevelop.com/>
2. Install the Azure SDK <http://www.windowsazure.com/en-us/downloads/>.

# Exercise 1 (Web API, OWIN, and Azure)

1. Run Visual Studio as Admin (this is needed in order to use the Azure emulator).
2. Open the solution at ..\Azure Lab\AzureFSharpOwinLabStarter
3. Ensure that the project (AzureFSharpOwin) builds and runs (though it doesn’t yet do much of anything). You may have to grant access to the Microsoft DFAgent if your Firewall blocks it. You can monitor what is going on by viewing the Compute Emulator UI (right click the Windows Azure icon in your tray and click Show Compute Emulator UI).



1. Open the Startup.fs file in the WebApiRole project and add the following content. This sets up the Web API service to run on OWIN and tells it to serve up with index.html file.

namespace WebApiRole

open Owin

open System

open System.IO

open System.Net

open System.Net.Http

open System.Web.Http

type Startup() =

member x.Configuration(app: IAppBuilder) =

// Host Web API

let config = new HttpConfiguration()

WebApiConfig.Register config

app.UseWebApi(config) |> ignore

let fileServerOptions = Microsoft.Owin.StaticFiles.FileServerOptions() in

fileServerOptions

.WithDefaultFileNames("index.html")

.WithPhysicalPath(AppDomain.CurrentDomain.BaseDirectory)

|> app.UseFileServer

|> ignore

1. We’ll also need to startup OWIN at a desired URI whenever the worker role starts up. To do that, open the WorkerRole.fs file and replace the text “(\* REPLACE WITH CONTENT FROM STEP 3 IN EXERCISE 1 \*)” with the following code:

let endpoint =

RoleEnvironment.CurrentRoleInstance.InstanceEndpoints.["Endpoint1"]

let baseUri = sprintf "%s://%A" endpoint.Protocol endpoint.IPEndpoint

log ("Starting OWIN at " + baseUri) "Information"

let options = StartOptions()

options.Urls.Add(baseUri)

webApp <- WebApp.Start<Startup>(options)

1. You’ve now officially used OWIN to self-host a simple web app on Azure. Navigate to <http://127.255.0.0:82/> and verify that the site comes up as expected. Add a few different guitar names and remove a few.

# Exercise 2 (Adding Fog)

1. In this exercise we will use a library named Fog to help us save/retrieve the list of guitars to Blob Storage. Fog sits on top of the Azure .NET API and makes it very easy to get started. Normally, the first step for getting started with Fog would be to install the Fog NuGet package. The AzureFSharpOwinLabStarter project already has this package installed for you, so you can skip that for now.
2. Fog optionally makes use of conventions to reduce the amount of code that you have to write to interact with various aspects of Azure. One of these conventions is related to configuration settings. For Blob Storage interaction, Fog expects a configuration setting with a name of BlobStorageConnectionString in the ServiceConfiguration.<environment>.cscfg files. Open these files and verify that this setting exists (it should definitely be there).
3. We’ll now replace the use of the text file to storage the list of guitars with Azure Blob Storage. We start by checking to see if a blob with a given name already exists and creating it if it doesn’t. Open the GuitarsApi.fs file, find (\* REPLACE with content from Step 3 of Exercise 2 \*) and replace everything in between the REPLACE coments with the following code:

open Fog.Storage.Blob

let containerName = "guitars"

let blobName = "guitarsblob"

let blob = GetBlobReference containerName blobName

try

blob.FetchAttributes()

with \_ -> UploadBlob containerName blobName "" |> ignore

1. We’ll need to make a few changes to any function that retrieves or saves the list. We’ll start with the getGuitars() function. Replace the body this function with the following code.

let guitars = DownloadBlob<string> containerName blobName

if not (String.IsNullOrEmpty guitars) then

guitars.Split(',') |> Array.map (fun x -> Guitar(Name = x))

else [||]

1. We only need to replace 1 line in the addGuitar function. Find the text (\* REPLACE with code from Step 5 of Exercise 2 \*) in the addGuitar function and replace the code between the comments with the following line:

UploadBlob containerName blobName result |> ignore

1. Lastly, we need to replace two lines in the removeGuitar function. Find the text (\* REPLACE with code from Step 6 of Exercise 2 \*) and replace the code between the comment lines with the following:

String.Join(",", data) |> UploadBlob containerName blobName |> ignore

1. That’s it. You should now be able to run your OWIN app and have the guitar list stored to Blob Storage. Go to <http://127.255.0.0:82/> in a browser to verify.

# Extra Credit Exercise

1. Use Fog to interact with Azure Service Bus <http://www.windowsazure.com/en-us/develop/net/how-to-guides/service-bus-queues/>, <http://dmohl.github.io/Fog/>, <https://github.com/dmohl/fs-web-cloud-mobile/tree/master/Ch%203/HackySackStore>.

# F# Cheat Sheets

DZone Reference Card: <http://cdn.dzone.com/sites/all/files/refcardz/rc081-010d-fsharp.pdf>

The F# Handout: <http://www.hamletdarcy.com/files/2008/TheFSharpHandout.pdf>