Countdown To ASP 5

Building a Sample App

http://ammonsonline.com

Who Am I?

- Jeff Ammons
  - Principal Architect
  - Sage Software

- President
  - GGMUG
  - Gwinnett, Georgia, Microsoft User Group
  - 2nd Thursday
  - Gwinnett Technical College
  - http://GGMUG.com

http://ammonsonline.com

Why is ASP Changing?

- Market Share/Mind Share
  - Web developers choose:
    - Mac for development
    - Linux for deployment
  - Dynamic languages popular
- Cross Platform
  - See above…
- Technical Deficits
  - .Net built for desktop apps
  - Not optimized for server apps

http://ammonsonline.com

Cynical History of ASP

- ASP (classic)
  - Hey, look, we have a web stack!
  - Better than Perl/CGI!
- ASP.Net
  - Pretend the web is Windows!
- ASP.Net MVC
  - We can do what Ruby on Rails does!
- ASP.Net Web API
  - We have to use WCF, damnit!
- Where'd everybody go?
- ASP 5
  - We can do what Node does!
  - We'll let you work on Mac!
  - We'll let you deploy to Linux!
  - You don't *have* to buy Visual Studio!

http://ammonsonline.com

What's Staying The Same

- MVC
  - Controllers
  - Views
  - Models
  - Routes*
  - Razor*
- Web API
  - Controllers
  - Routes*

http://ammonsonline.com

* A few changes to Routes and Razor

What's New?

- Open Source
- Cross Platform
- Modular
- OPT*

- *Other People's Tools*

http://ammonsonline.com

What's New?

- .Net Core
  - Deployable CLR
- Nuget is first class citizen
- Dependency Injection by default
- wwwroot
  - Lock down files that will be served
- Node.js
  - NPM: Node Package Manager
  - Grunt/Gulp for automation
  - Bower for *client* package management

http://ammonsonline.com

What's Different?

- Project file
  - project.json instead of web.config
- Configuration
  - config.json instead of web.config
  - Environment vars instead of web.confg
- Minification and Bundling
  - grunt/gulp instead of ASP.net
  - Bundle/minify at *compile time* not runtime

http://ammonsonline.com

Current Timeline

- Beta 6: July 27, 2015
- Beta 7: September 2, 2015
- Beta 8: October 15, 2015
- RC: November (Go Live License?)
- RTM: Q1 2016

http://ammonsonline.com

ASP 5 Cheat Sheet

- project.json
  - Replaces web.config
  - Define webroot directory
  - Register dependencies
    - Nuget packages
  - Register commands
  - Define which frameworks are valid (.Net full, .Net Core)
  - Exclude files & directories from project
- wwwroot
  - Web accessible files ~ think content dir in older mvc
    - Images, scripts, stylesheets, etc.
  - Easier to isolate the files you want to serve
- config.json
  - Replaces web.config for appsettings & connection strings
  - Default option, you can also use xml, ini, or env vars
  - Expected path: config.json for local dev, env vars for servers
- bower.json
  - Manages *client* side dependencies
  - Like Nuget for javascript libs
- gulpfile.js
  - Define build and publish tasks
    - Copy files, minify css & javascript, etc.
  - Task Runner Explorer
- Startup.cs
  - Entry point for app
  - Add “Middleware” to pipeline
  - Dependency Injection
  - Configure Routes

http://ammonsonline.com

Resources

- https://github.com/aspnet/Home
- http://docs.asp.net/en/latest/index.html
- https://www.microsoftvirtualacademy.com/en-us/training-courses/what-s-new-with-asp-net-5-8478
- https://www.microsoftvirtualacademy.com/en-us/training-courses/introduction-to-aspnet-5-13786?l=PvSZtxoXB_5101937557
- https://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-deploy-aspnet-mvc-app-membership-oauth-sql-database/#add-a-database-to-the-application
- http://ef.readthedocs.org/en/latest/getting-started/aspnet5.html
- http://www.codeproject.com/Tips/988763/Database-Migration-in-Entity-Framework
- http://www.bricelam.net/2014/09/14/migrations-on-k.html
- http://stephenwalther.com/archive/2015/02/24/top-10-changes-in-asp-net-5-and-mvc-6
- http://www.pluralsight.com/courses/aspdotnet-5-ef7-bootstrap-angular-web-app

http://ammonsonline.com

Contact Info

- Twitter: jeffa00
- Google Plus: jeffa00
- Linked-In: jeffammons
- Blog: http://ammonsonline.com
- YouTube: http://youtube.com/ammonsonline
- SciFi/Humor: http://galacticbeacon.com
- User Group: http://ggmug.com

Workshop

- Windows Phone Spotter App
  - Log every time you see a Windows Phone in the Wild!
  - Goal: Highlight the familiar, introduce a bit of the new
    - Show that lots of the new changes are optional
- Warning:
  - BETA BETA BETA
    - Sometimes feels more like ALPHA ALPHA ALPHA
    - Getting better constantly
  - Some things can be
    - Broken Broken Broken
    - YMMV: Your Mileage May Vary. Wildly.
  - **Let's keep things simple!**

http://ammonsonline.com

Workshop

- I'll build directly from this recipe
- You can watch now
  - Then build from it yourself later
- Feel free to skip the optional Prequel sections if you don't want to use Git or Azure
- My Samples:
  - https://github.com/jeffa00/WinPhoneSpotter01
  - http://winphonespotter01.azurewebsites.net/
    - Note: They will likely change in the future

http://ammonsonline.com

User Stories

- Prequel:
  - Create project
  - Initial setup
  - Optional Bits
    - GitHub
    - Azure
- Story 1: User wants to log in
- Story 2: User wants to see list of sightings
- Story 3: User wants to log a phone spotting

http://ammonsonline.com

Prequel: Create Project

- File
  - New
  - Project…
  - Web
  - ASP.NET Web Application
  - ASP.NET 5 Preview Templates
  - Web Sites

  - Wait…

http://ammonsonline.com

Prequel: Change Site Title

- Open appsettings.json
- Add AppSettings: SiteTitle, set value to
  - Windows Phone Spotter Dev

```
{
  "AppSettings": {
    "SiteTitle": "Windows Phone Spotter Dev"
  },
  "Data": {
    "DefaultConnection": {
      "ConnectionString":
         "Server=(localdb)\\mssqllocaldb;Database=
           aspnet5-WebApplication3-0b0afda1-9370-4bde-b0a5-1b58f4c08ca9;
           Trusted_Connection=True;MultipleActiveResultSets=true"
    }
  }
}
```

http://ammonsonline.com

Prequel: Create Options Model

- In Models folder
- Add class ApplicationOptions.cs
- Add property SiteTitle

Prequel: Map Settings Config

- Open Startup.cs
- Find ConfigureServices(IServiceCollection services)
- Add
```
services.Configure<ApplicationOptions>(Configuration.GetSection("AppSettings"));
```

Prequel: Inject ApplicationOptions

- Open HomeController.cs
- Add private property ApplicationOptions

```
private ApplicationOptions Options { get; set; }
```

- Add Constructor

```
 public HomeController(IOptions<ApplicationOptions> options)
{
  Options = options.Value;
}
```

Prequel: Add SitleTitle to Page

- Open HomeController.cs
- In public IActionResult Index()
- Add

```
ViewData["SiteTitle"] = Options.SiteTitle;
```

- Open View Shared/_Layout.cshtml
- Change

```
<a asp-controller="Home" asp-action="Index" class="navbar-brand">AspBeta8Test01</a>
```

- to

```
<a asp-controller="Home" asp-action="Index" class="navbar-brand">@ViewData["SiteTitle"]</a>
```

Prequel: Test Title on Home Page

- Does the home page show the new SiteTitle in the upper left hand corner?
- Try the About page.
- Does it show the SiteTitle?
- Fix it!
- Create LoadViewData method

```
 public void LoadViewData()
{
   ViewData["SiteTitle"] = Options.SiteTitle ?? “Site Title Missing”;
}
```

- Call LoadViewData in each IActionResult method

Prequel: Learn More About Configuration Info

- Watch this segment of the Introduction To ASP.NET 5 course on Microsoft Virtual Academy
- https://www.microsoftvirtualacademy.com/en-us/training-courses/introduction-to-aspnet-5-13786?l=8eIeQ5pXB_3201937557

Prequel (Optional): Commit in Git

- Team Explorer
- Changes
- Enter commit message: “initial commit”
- Click “Commit”

http://ammonsonline.com

Prequel (Optional): Add to Git Server

- GitHub with 2015 GitHub explorer
- Sync
- Make sure you are connected to your GitHub acct
- Either:
  - New Repo
    - Enter new Repo Name
    - Publish
  - OR
  - Existing Repo
    - Enter URL (https version) of existing Repo
    - Publish

http://ammonsonline.com

Prequel (Optional): Create Azure WebApp

- If you plan to go this route I'd do it early in the process so you are testing the fewest of *your* changes
- NOTE: You will have to work through the EF migrations if you test this way.

- Using Azure Portal
- Create a new Web App
- Link to your GitHub repo
- …
- Profit!

http://ammonsonline.com

Prequel (Optional): Change Azure Site Title

- Application Settings
- App Settings
  - Key: AppSettings:SiteTitle
  - Value: Windows Phone Spotter

http://ammonsonline.com

Prequel (Optional): Create Azure SQL Db

- Use Portal to create a new DB
- Or use an existing one

http://ammonsonline.com

Prequel (Optional): Add AzureSQL Connection String

- Get connection string from portal
- Go to Web App in Portal
  - Application Settings
    - Show Connection Strings
    - Add
      - Key: DefaultConnection
      - Value: Paste in conn string

http://ammonsonline.com

Prequel (Optional): Test Login

- At this point if things are swinging your way:
  - Register
  - Add email address and pwd
  - Should create DB/tables/add new user
  - …
  - More Profit!

http://ammonsonline.com

Story 1: User wants to log in

- For simplicity we'll use the default identity
- We'll also take the simple route for dev/prod
  - Dev/local in config.json
    - Don't change it – leave it local and trusted/Never check in credentials…
      - Windows security, not SQL Server Login: No user/pwd
  - Prod from environment variable
- SecretManager is nearly usable
  - Works sometimes
  - For now I'd stick with the local/ trusted_connection for dev

http://ammonsonline.com

Story 2:
User wants to see list of sightings

- Create basic Model Objects
  - PhoneSpotting
    - int PhoneSpottingId
      - Annotate with
        - `[Key]`
        - `[DatabaseGenerated(DatabaseGeneratedOption.Identity)]`
    - DateTime SpotTime
    - string PhoneManufacturer
    - string PhoneModel
    - double Latitude
    - double Longitude
    - string Notes
  - PhoneSpottingsViewModel
    - List<PhoneSpotting> PhoneSpottings

http://ammonsonline.com

Story 2:
User wants to see list of sightings

- Add PhoneSpottingDbContext

```
public class PhoneSpottingDbContext : DbContext
{
    public DbSet<PhoneSpotting> Spottings { get; set; }
    private static bool _created;
}
```

- Register in Startup.cs
  - In ConfigureServices after ApplicationDbContext

```
services.AddEntityFramework()
  .AddDbContext<PhoneSpottingContext>(options =>
    options.UseSqlServer(Configuration["Data:DefaultConnection:ConnectionString"]));
```

- Generate EF Migration
  - Command line (likely added to VS in the future)
  - cd to src directory
  - Run

```
dnx ef migrations add PhoneSpotterCreate –c PhoneSpottingDbContext
dnx ef database update –c PhoneSpottingDbContext
```

http://ammonsonline.com

Story 2:
Learn More About Entity Framework

- Watch this segment of the Introduction To ASP.NET 5 course on Microsoft Virtual Academy
- https://www.microsoftvirtualacademy.com/en-us/training-courses/introduction-to-aspnet-5-13786?l=pWAaw6pXB_9401937557

- **Note: Some things have changed since the video was made!**
  - **Was: dnx ef migration apply**
  - **Now: dnx ef database update**

Story 2:
User wants to see list of sightings

- Create PhoneSpottingController
- Submit list of spottings to View in Index

```
public class PhoneSpottingController : Controller
 {
    private PhoneSpottingDbContext _spotContext;

    public PhoneSpottingController(PhoneSpottingDbContext context)
    {
        _spotContext = context;
    }

    public IActionResult Index()
    {
         return View(new PhoneSpottingsViewModel { PhoneSpottings = _spotContext.Spottings.ToList() });
   }
}
```

http://ammonsonline.com

Story 2:
User wants to see list of sightings

- Create View
  - PhoneSpotting/Index.cshtml

```
@*
   For more information on enabling MVC for empty projects, visit http://go.microsoft.com/fwlink/?LinkID=397860
*@
@using AspBeta8Test01.ViewModels.PhoneSpottings
@model PhoneSpottingsViewModel
@{
   ViewData["Title"] = "Windows Phone Spotting List";
}

<h1>Windows Phone Spottings</h1>

@foreach (var spotting in Model.PhoneSpottings)
{
   <div>
      <h2>@spotting.PhoneManufacturer @spotting.PhoneModel</h2>
      <div class="spotting-date">@spotting.SpotTime</div>
      <div class="spotting-location">
         <span class="coordinate-name">Latitude:</span> <span class="coordinate">@spotting.Latitude</span>
         <span class="coordinate-name">Longitude:</span> <span class="coordinate">@spotting.Longitude</span>
         <div>Spotting Notes:</div>
         <div>@spotting.Notes</div>
      </div>
   </div>
}
```

http://ammonsonline.com

Story 2:
User wants to see list of sightings

- Add Link to PhoneSpotting in _Layout.cshtml

Story 3:
User wants to log a phone spotting

- Create Create method(s)

```
// GET: /<controller>/Create
        [Authorize]
        public IActionResult Create()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize]
        public IActionResult Create(PhoneSpotting newSpotting)
        {
            if (ModelState.IsValid)
            {
                _spottingContext.Spottings.Add(newSpotting);
                _spottingContext.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(newSpotting);
        }
```

http://ammonsonline.com

Story 3:
User wants to log a phone spotting

- Create View
  - Try out some tag-helpers

```
@*
    For more information on enabling MVC for empty projects, visit http://go.microsoft.com/fwlink/?LinkID=397860
*@
@using AspBeta8Test01.ViewModels.PhoneSpottings
@model PhoneSpotting
@{
    ViewData["Title"] = "Create New Windows Phone Spotting";
}

<h1>Log Windows Phone Spotting!</h1>

<form asp-controller="PhoneSpotting" asp-action="Create" method="post" class="form-horizontal">
    <div class="form-group">
        <label asp-for="PhoneManufacturer" class="control-label col-xs-2">Manufacturer:</label>
        <div class="col-xs-10">
            <input asp-for="PhoneManufacturer" class="form-control" />
        </div>
    </div>

    <div class="form-group">
        <label asp-for="PhoneModel" class="control-label col-xs-2">Model:</label>
        <div class="col-xs-10">
            <input asp-for="PhoneModel" class="form-control" />
        </div>
    </div>
```

Story 3:
User wants to log a phone spotting

```
    <div class="form-group">
        <label asp-for="Latitude" class="control-label col-xs-2">Latitude:</label>
        <div class="col-xs-10">
            <input asp-for="Latitude" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Longitude" class="control-label col-xs-2">Longitude:</label>
        <div class="col-xs-10">
            <input asp-for="Longitude" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="SpotTime" class="control-label col-xs-2">Date &amp;amp; Time:</label>
        <div class="col-xs-10">
            <input type="text" asp-for="SpotTime" class="form-control" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Notes" class="control-label col-xs-2">Notes:</label>
        <div class="col-xs-10">
            <textarea asp-for="Notes" class="form-control"></textarea>
        </div>
    </div>
    <div class="form-group">
        <div class="col-xs-offset-2 col-xs-10">
            <button type="submit" class="btn btn-primary">Login</button>
        </div>
    </div>
</form>

@section Scripts {
    <script src="~/js/phoneSpotter.js"></script>
}
```

Story 3:
User wants to log a phone spotting

- For fun you can prefill lat/long from javascript
  - The crux of it is this:

```
$(document).ready(function () {
    if ("geolocation" in navigator) {
        navigator.geolocation.getCurrentPosition(function (position) {
            $("#Longitude").val(position.coords.longitude);
            $("#Latitude").val(position.coords.latitude);

            var currentTime = new Date();
            $('#SpotTime').val(currentTime.toLocaleString());
        });
    } else {
        alert("Geolocate NOT found.");
    }
});
```