

# ASP.NET Core Deep Dive

Daniel Roth  
Senior Program Manager  
ASP.NET



A deep dive into . . .

Hosting  
Request handling  
Routing  
MVC extensibility



## Get Started with ASP.NET Core 1.0

Go to <http://dot.net>

Docs: <https://docs.asp.net>

Samples and code: <https://github.com/aspnet>

## ASP.NET Core hosting

ASP.NET Core is just a bunch of libraries

Host in your own process (ex. a console app)

# Setting up ASP.NET Core hosting

## Build the host

- Setup host services
- Configure a server
- Identity startup logic
- Build the app RequestDelegate

## Run the host

- Start the server for the hosted app
- Handle requests using the app RequestDelegate

# Configuring hosting properties

Server

Server URLs

Startup

Content root

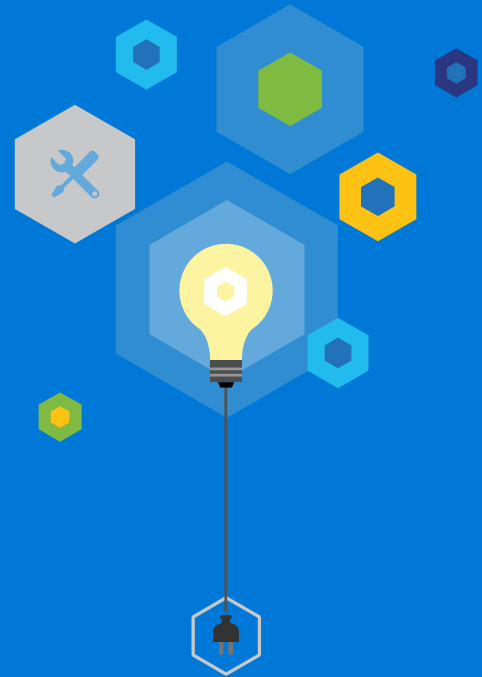
Web root

Environment

Application name

# Hosting

Daniel Roth

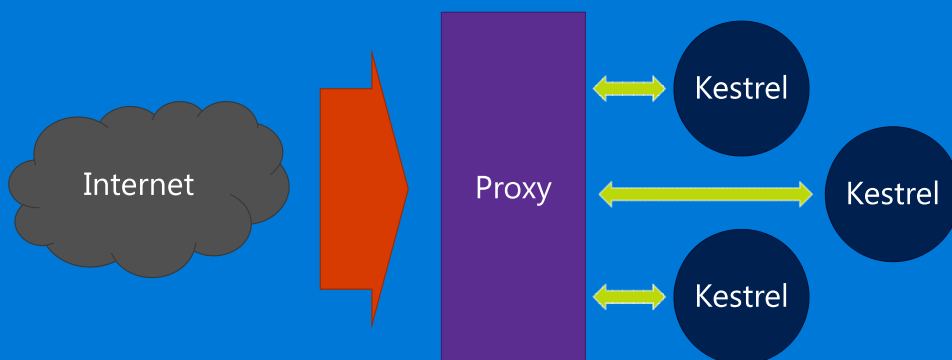


## Hosting in production

Kestrel is not yet hardened for internet facing traffic

Run Kestrel behind a mature reverse proxy

Ex IIS, Nginx, Apache, etc.



## Publishing to IIS

### Install the new ASP.NET Core Module

Native IIS module that forwards request to your application

Supports forwarding client certs, Windows auth, IIS sub applications, app offline

### Setup module in web.config

New iis-publish tool will handle this for you

### Point IIS at your app content root

Static files will only be served from the web root

### Setup IIS integration for your ASP.NET Core host

```
webHostBuilder.UseIISIntegration()
```

## ASP.NET Core Module (ACM)

1. IIS receives a request
2. ACM starts the app in a separate process (if it's not already running)  
The app no longer runs as w3wp.exe but as dotnet.exe or myapp.exe
3. ACM forwards the request to the application
4. The app processes the request and sends the response back to ACM
5. IIS forwards the response to the client

## ASP.NET Core Module configuration

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.webServer>
    <handlers>
      <add name="aspNetCore" path="*" verb="*"
          modules="AspNetCoreModule" resourceType="Unspecified"/>
    </handlers>
    <aspNetCore processPath="dotnet" arguments=".\HelloWorld.dll"
        stdoutLogEnabled="false" stdoutLogFile=".\logs\stdout" />
  </system.webServer>
</configuration>
```

## Env vars set by ACM for the host process

ASPNETCORE\_PORT

Port to forward the request to

ASPNETCORE\_APPL\_PATH

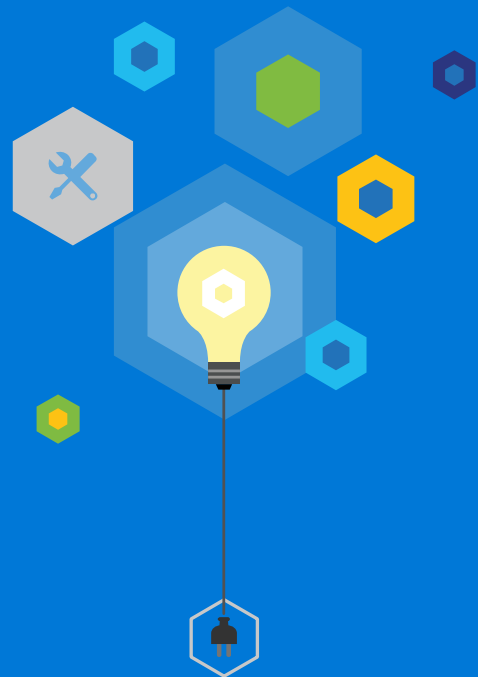
Application path to forward the request to

ASPNETCORE\_TOKEN

Pairing token to ensure only local requests from IIS get received

# Hosting in IIS

Daniel Roth



## Request handling

Request handling pipeline defined in startup

Use middleware to process requests

Built RequestDelegate used to handle all requests

Lots of built-in middleware

Routing, authentication, static files, diagnostics, error handling, session, CORS, localization, custom

## RequestDelegate

```
public delegate Task RequestDelegate(HttpContext context);
```

## Middleware

```
app.Use(Func<RequestDelegate, RequestDelegate> middleware);
```



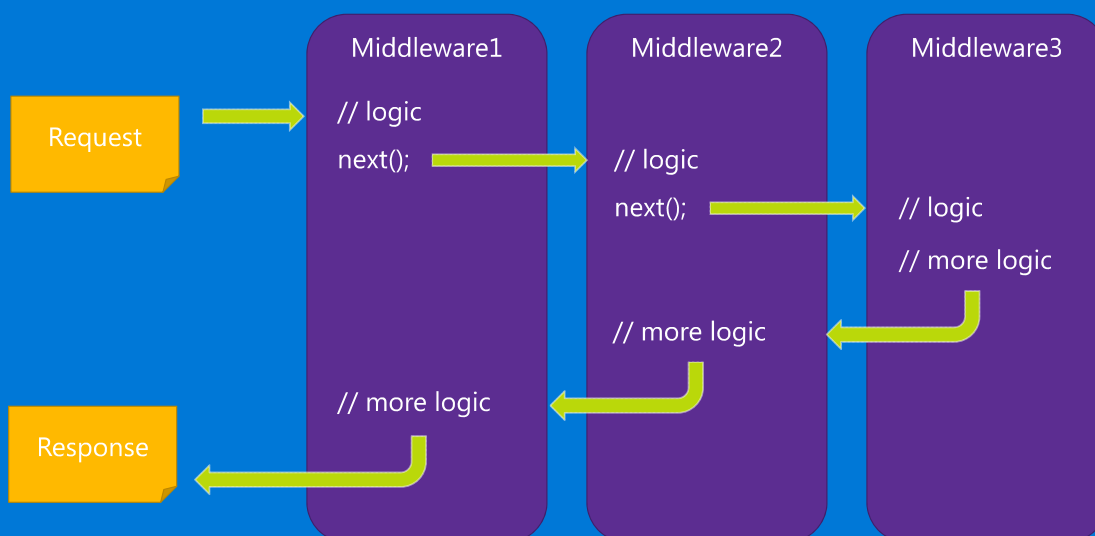
# Middleware

```
public class MyMiddleware
{
    public MyMiddleware(RequestDelegate next, ...) { ... }

    public async Task Invoke(HttpContext httpContext, ...) { ... }
}

app.UseMiddleware<MyMiddleware>();
```

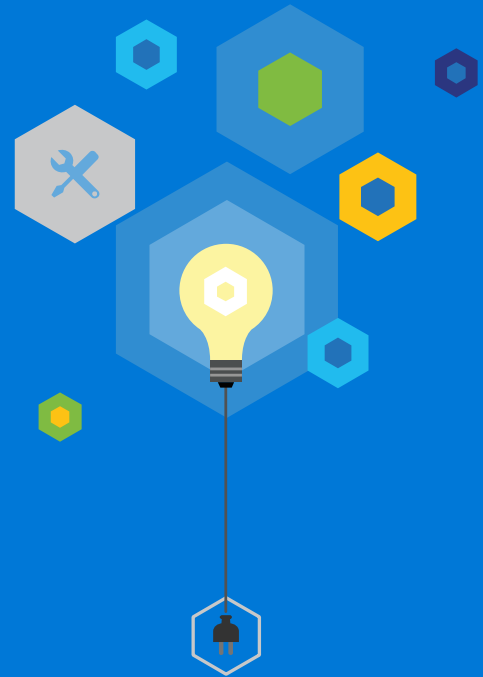
# Middleware



MICROSOFT CONFIDENTIAL – INTERNAL ONLY

# Request handling

Daniel Roth



## Routing

Routing middleware calls into IRouter

IRouter creates a RequestDelegate for handled requests

RouteCollection iterates through a list of IRouters and stops when one handles the request

Route implements route templates and calls a target IRouter when the request matches

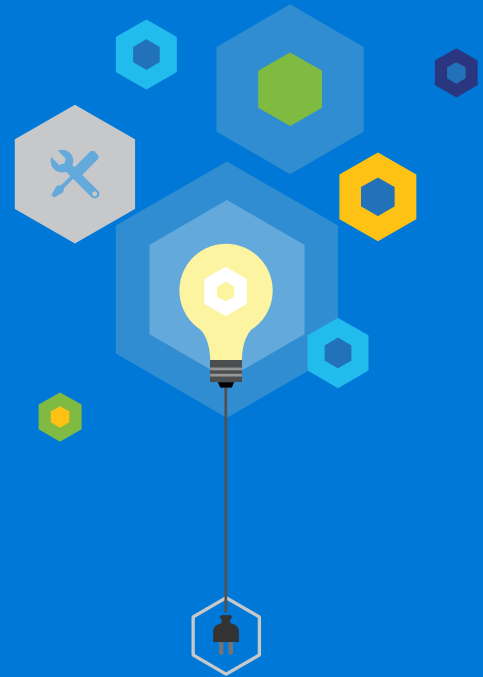
RouteBuilder creates a RouteCollection using a default IRouter for the target of added routes

MVC sets itself up as the default IRouter on the RouteBuilder

Supports link generation too

# Routing

Daniel Roth



## MVC: Application Parts

ApplicationPartManager is the one-stop shop for discovery logic in MVC

Controllers, View Components, Tag Helpers, Razor compilation references

Application parts are resources from which to discover features (ex assemblies)

Feature providers populate features from the application parts

# MVC: Application Model Conventions

Application model is a representation of your MVC app that you can read and manipulate

Controller names, action names, action parameters, attribute routes, filters

Modify the application model with custom conventions

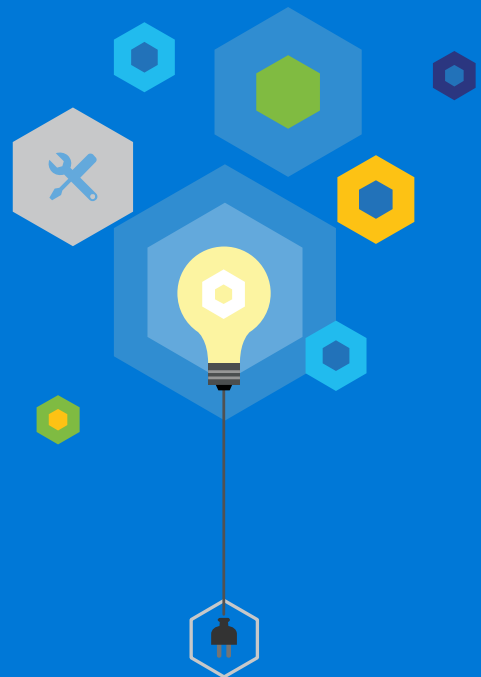
Ex Web API compat shim

Explore your app

Ex. Swagger

## Application parts & conventions

Daniel Roth

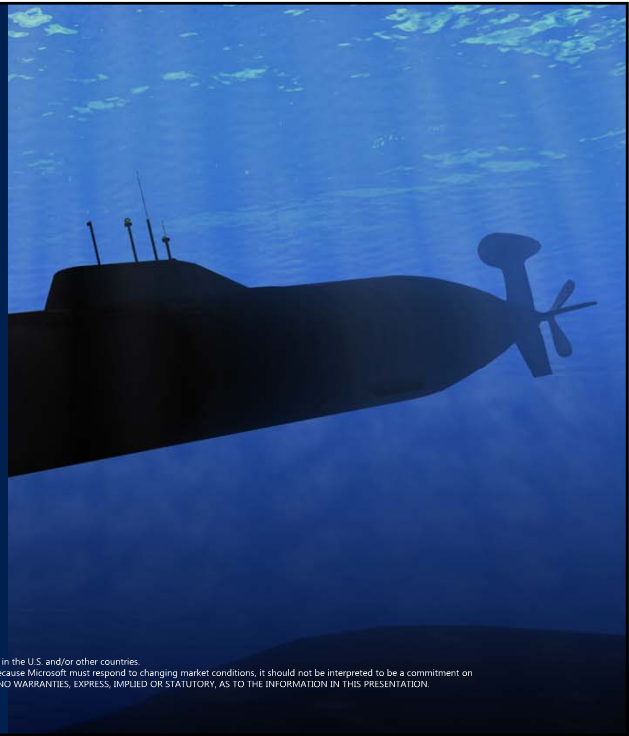


Thanks for watching!  
Questions?



Microsoft

© 2016 Microsoft Corporation. All rights reserved. Microsoft, Windows, and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.



## TH17 - Deep Dive into ASP.NET Core - Daniel Roth

© 2015 Microsoft Corporation. All rights reserved. Microsoft, Windows, and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.