O'REILLY®

Software Architecture

ENGINEERING THE FUTURE OF SOFTWARE

softwarearchitecturecon.com

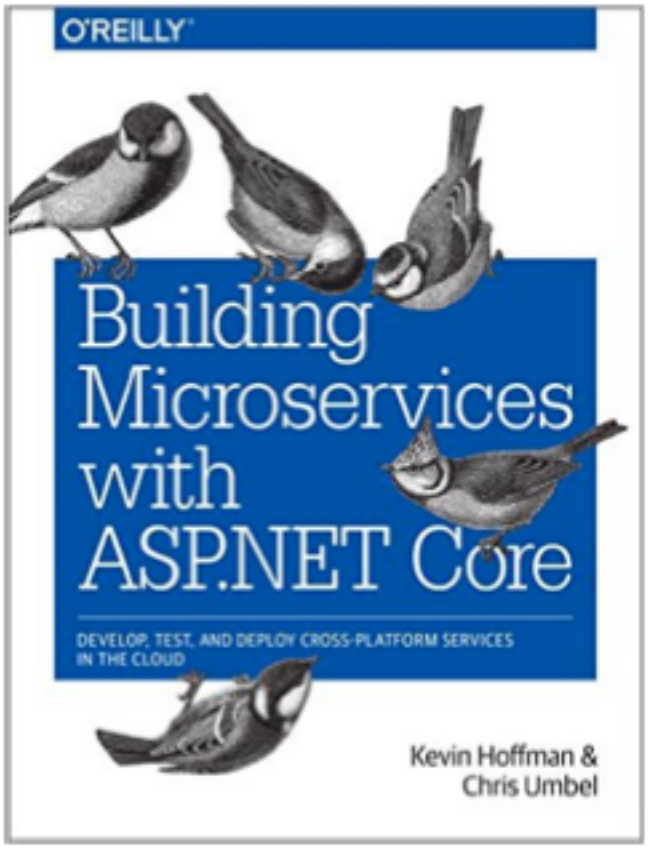
#OReillySACon

Microservices with ASP.NET Core

Kevin Hoffman
and
Chris Umbel

About Us

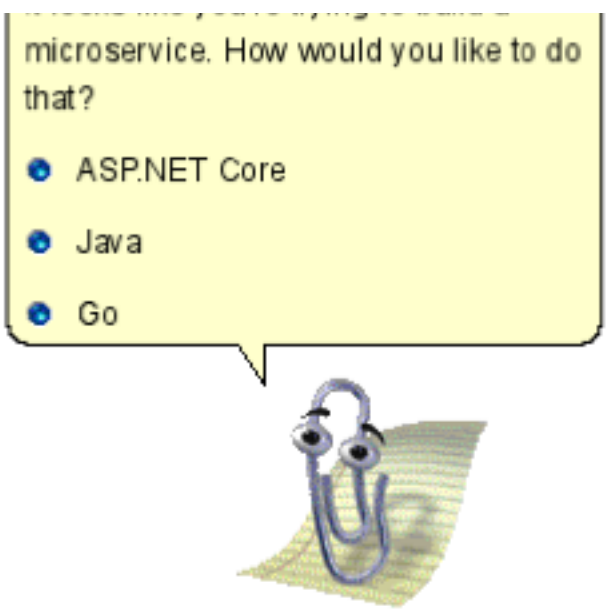
@KevinHoffman @ChrisUmbel



Agenda

It looks like you're trying to build a

- Create .NET Core Project
- Add middleware
- Add controller
- Inject a repository
- Communicate with a Database
- Consume Spring Cloud Configuration Server
- Participate in Service Discovery
- All on a Mac
  - *No Windows were harmed for any of these demos!* 😮



ASP.NET Web and Service Development Today

- Maintaining Windows servers carrying pre-cloud baggage
- Traditional MSI-based Windows Server Application Installs
  - Production deploy with RDP, run MSI!
  - Deploy via Active Directory Group Policy
- Maintain and Configure IIS
- Servers are long-lived, overfed pets
- 3 Rs (Rotate, Repave, Repair) Security very difficult
- Tightly coupled, closed ecosystem
- Mostly closed source
- All-or-nothing monolithic framework

The Future is Core

- Modern, immutable, dependency-vendored artifact deploy

- Modular - import only what you need!
- Easier to maintain Linux servers
- NO IIS.
  - That's right, *no IIS*.
- Linux servers are *cloud ready*
  - Easier to treat as disposable out of the box
- 100% Open Source, from runtime to frameworks
- Did we mention that there's no IIS?
- Core on Windows is ideal stepping-stone from legacy to pure Core
- Core is Microsoft's stated future direction for .NET

What's in ASP.NET Core?

- .NET Core
  - Cross-platform, open-source runtime

- Cross-platform, open-source runtime
  - Bare-bones, minimum required to start an app
  - Application types: Console, Web (ASP.NET Core), UWP
- Implementation of lessons learned since ASP.NET 1.0 released 14 years go
  - Yes, we're ***THAT OLD***. 👴
- MVC Framework (optional module, like everything else)
  - Server-side rendering/templating
  - Routing
  - Microservices

Hello Core World

http://www.microsoft.com/net/co

- dotnet new
- dotnet restore

- dotnet run

O'REILLY
Software
Architecture

Architecture

ENGINEERING THE FUTURE OF SOFTWARE

softwarearchitecturecon.com

#OReillySACon

DEMO

Hello, World!

ASP.NET Middleware

- Components added to request pipeline to handle requests & responses
- Build pipeline with Run, Map, and Use methods.
- Common middleware:
  - Static files

- Static files
- Error handling
- Logging
- Authentication
- MVC (Routing and Processing)

- Code and defer, branch, or terminate pipeline

O'REILLY
Software
Architecture
ENGINEERING THE FUTURE OF SOFTWARE

softwarearchitecturecon.com
#OReillySACon

DEMO

Adding Middleware

RESTful Routes

- Immediately familiar to developers w/experience with Web API
- Route pattern at class level
  - `[Route("api/[controller]")`
- Route pattern on HTTP methods
  - `[HttpGet("{id}")]`

- Automatic deserialization
  - `[FromBody]Monster monster`
- Return `ActionResult`, supports async keyword for async actions
- Explicit Routing
  - Can add global context roots (useful in PCF)
  - http://bit.ly/2ap5gGF
- Mix-and-Match

Dependency Injection

- Scoped services
- Global services
- Configuration system
- Middleware
- Default IoC container provided, can customize/replace

- Autofac

softwarearchitecturecon.com
#OReillySACon

DEMO

Dependency Injection
and
Controllers

Delivering Continuously

- "Always be deploying"
- Every commit could end up in production *today*
- CI Flow
  - Check in
  - Build Code
  - Run Unit Tests
  - Run UI/JavaScript Tests (if applicable)

- Run UI/JavaScript Tests (if applicable)
- Run Integrations
- Deploy to Dockerhub
- Push Docker Image to target cloud environment (e.g. PCF, Pivotal Web Services, etc.)

More CI

- Docker Images + Cloud CI tool:
  - Spin up backing services in containers
  - Run integration tests against isolated backing services
  - Higher degree of confidence in builds
- *Tested* artifact is the *deployed* artifact, with **no changes**.
- Backing services can be 3rd party products or *your own* services
- Acceptance test environments spun up on demand

- Choose which services are real and which are simulators
- “Battlefield simulator”

DEMO

softwarearchitecturecon.com
#OReillySACon

Introducing Steeltoe

- http://steeltoe.io
- Spring Cloud Clients in ASP.NET and ASP.NET Core
  - Spring Cloud Configuration Server
  - Service Discovery with Eureka
  - Spring Cloud Connectors
  - Initial Release Sept. 2016
- 100% Open Source
  - https://github.com/steeltoeoss/

External Configuration

- application.json file can provide local defaults
- `config.AddEnvironmentVariables()` adds raw environment vars to config
- `config.AddCloudFoundry()` adds bound service metadata to config
- Add Configuration Server client to get config from SCCS.
- All configuration is last-added-wins - you choose override order

Connectors

- SQL Server + Entity Framework 6 (Windows)
- mySQL*
- Redis
- RabbitMQ

Data Services in Core

- Database Clients
  - SQL Server
  - Postgres
  - MongoDB
  - RESTful DBs
- Not available
  - DB2
  - Oracle (yet)
- Entity Framework Core 1.0 (not to be confused with EF6)

O'REILLY
Software Architecture
ENGINEERING THE FUTURE OF SOFTWARE

DEMO

Database Backing Services via External Configuration

Service Discovery with Eureka

- Create DiscoveryHttpClientHandler instance
- Pass as parameter to HttpClient
- Make a simple HTTP request
- Steeltoe takes care of URL changes
- Application code is never aware of real location of backing service

http://grabs/api/grabs

O'REILLY
Software Architecture
ENGINEERING THE FUTURE OF SOFTWARE

DEMO

Service Discovery and Consumption

Summary and Take-Aways

- Small gap from empty project to working service
  - No more giant monolith framework
  - “Micro” is the default way of doing things
- Build modern microservices in C# on any device (Mac, Linux, Windows) and deploy to modern PaaS targets.
- .NET and Java apps can interoperate in the same microservices ecosystem.
- We can stop using IIS.