Building Cross Platform Web Services with Microsoft Technologies Using .NET Core

Sergey Barskiy, sergey@barskiy.com

.NET Core

Cross platform incarnation of .NET Framework
.NET Standard is the way to unify API across platforms and frameworks
Applications we can build with available tools

.NET Core

- Cross platform
- Performant
- Each app deploys its version
- Highly modular
- Carries over the best part of C# and VB, such as async, generics, Linq
- Open source

Unified Api

- .NET Standard 2.0

Tooling is important

- Visual Studio
- Visual Studio for Mac
- Visual Studio Code

Understaning CLI Tools

DotNet.exe and its Purposes
DotNet.exe Invocation Structure
Using Key Commands Installed with DotNet.exe
https://docs.microsoft.com/en-us/dotnet/articles/core/tools/

Applications You Can Build

- Web Apps and Web Services
- Mobile Applications
- IoT Applications

EF Core Capabilities and Architecture

Exposes data as set of objects, using DbContext and DbSet
Can update database structure via Migrations
Uses provider architecture

DbContext

Maintains state for changes
Converts state changes into queries
Provides access to RDBMS views, procedures and functions

DbSet<T>

Table Abstraction
Supports queries via Linq
Supports additions and deletions

Migrations

C# or VB.NET Code

Validation

Data Annotations
DbContext

Performance And Scalability

Async Code

Web Api

Builds on top of Web Api
Controllers with methods are services and actions
Inheriting from Controller is not required, but is helpful
Dependency injection everywhere, include EF

Web Api (cont.)

Action conventions and templates
Routes attributes are unified

Code

Setup new project
Add necessary references
Run to test
Create Entity Framework Code
Add Tests

Continue to learn

https://docs.microsoft.com/en-us/dotnet/core/