fancy Development

Software Development and Consulting

Inside .NET Core and DNX

Daniel Murrmann
daniel.murrmann@fancy-development.net
www.fancy-development.net

Microsoft CERTIFIED Technology Specialist

iSAQB Certified Professional for Software Architecture

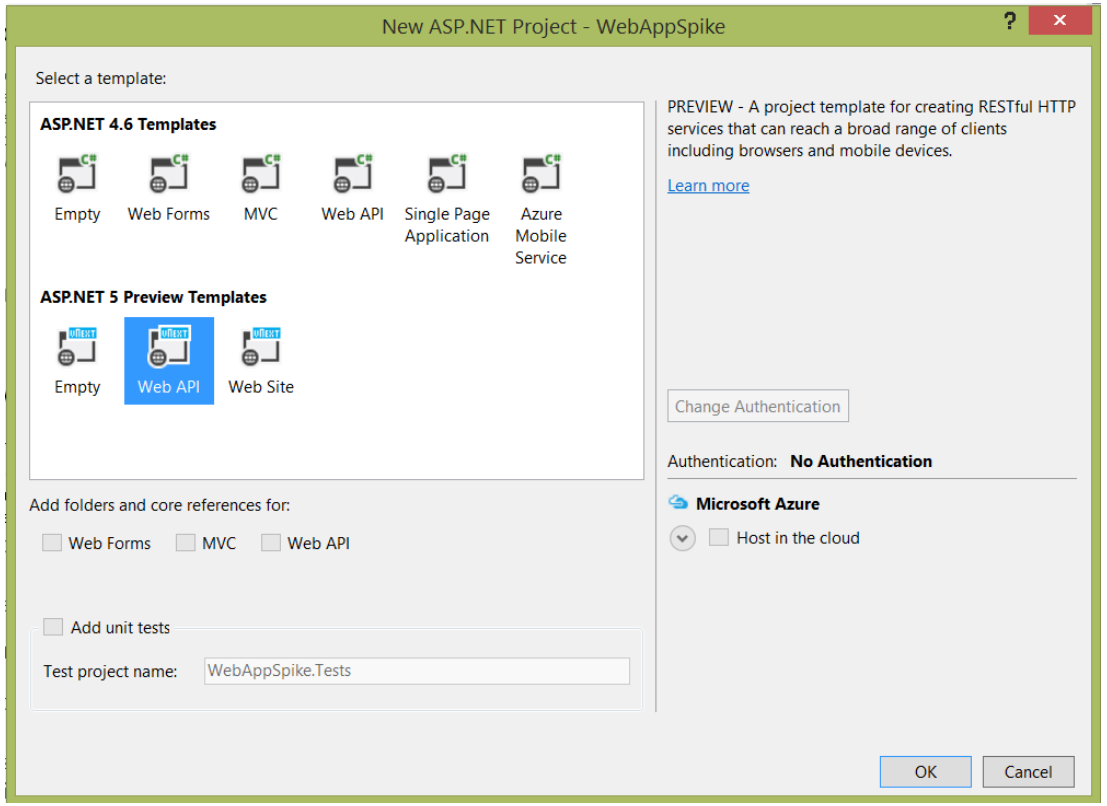
Contents

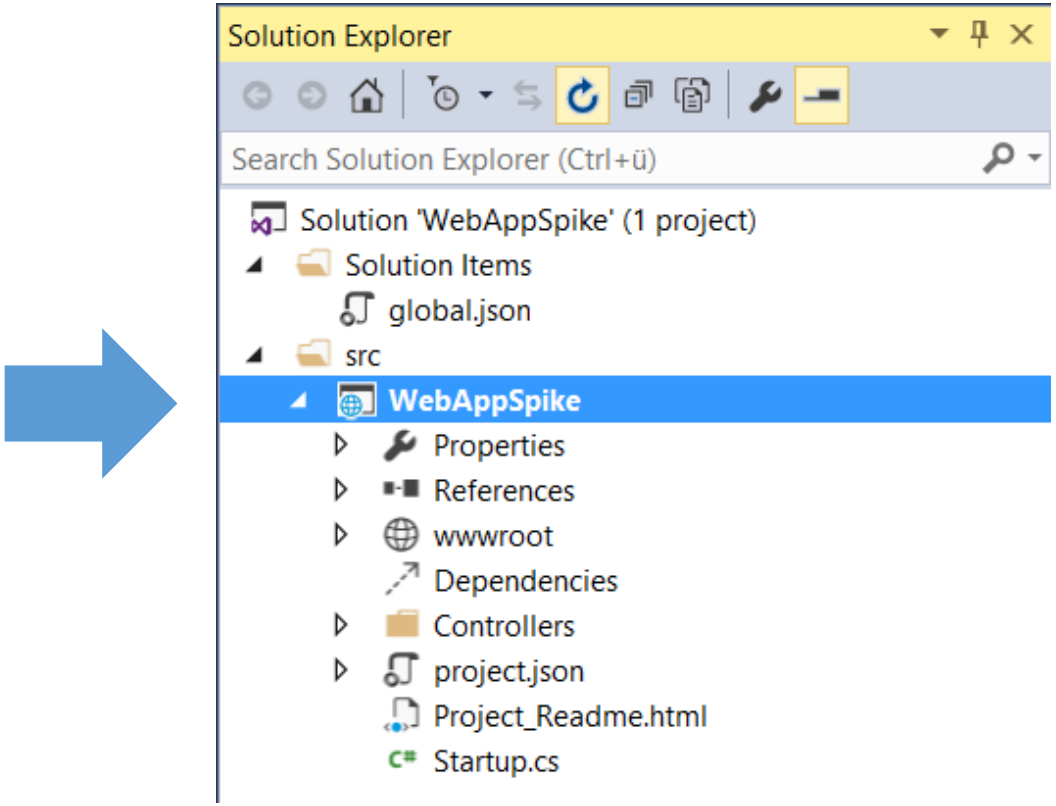
.NET Core and DNX

1. Background
2. Live-Demos
3. Summary

Inside .NET Core and DNX
Background

Notice the Changes at Web Projects





.NET Framework Retrospective

Years ago .NET Framework was a great thing because …

integrated all in one strategy

central runtime and framework

strong coupling with operating system

managed languages

Things have changed (1)

new ways to deploy applications ▶ sandbox/container deployment

no MSIs, no complex changes on operating system (registry, etc.)

new ways to deliver a library or framework ▶ modularized and via NuGet

no local installation of SDKs or development tools

Smartphone, Smartwatch, IoT Devices, etc. ▶ different feature set, self contained apps

no full .NET Framework available

highly distributed systems ▶ (SOAP), REST/JSON, Microservices

local communication not sufficient anymore

Things have changed (2)

standardized processes and configuration ▶ convention over configuration

no individual ways to accomplish things on same platform

Cloud, Server, Smart-*, Tablet, Desktop, IoT, etc ▶ same software needs to run everywhere

cross platform is a must have

Changes in the .NET Ecosystem

Modularized Frameworks
frameworks are modularized and delivered via NuGet packages (e.g. Web API, Mvc)

Roslyn
a new cross platform C# and VB compiler

.NET Core
a new true cross platform .NET Runtime

DNX
a cross platform .NET development and execution environment using Roslyn and .NET Core and decoupling .NET development from Visual Studio and Windows

.NET Core

What is it?

New .NET Runtime with Small Footprint
alternative to full .NET Framework and Mono

Operating System Independent
runs on Windows, Linux and MacOS

Cloud Optimized
in terms of size, deployment, etc.

.NET Framework vs. .NET Core

.NET Framework

.NET Core

DNX - .NET Execution Environment

What is it?

Framework
small set of libraries and tools to manage projects

Runtime / Bootstrapper
.NET Runtime independent application launcher

Project System
manage projects without development environment

Advantages…

… of Modularized Frameworks, Roslyn, .NET Core and DNX

less dependencies

easier to update and deliver

easier to port to different platforms

app decides which version to use

get exactly what you want, nothing else

Inside .NET Core and DNX
Live Demos

.NET Core (without DNX)

How to …

Get .NET Core
use a NuGet package feed listing .NET Core as package and download it

Use a native Bootstrapper to Start a .NET Application using .NET Core
use *Fancy.CoreClrHost* to load .NET Core and an application into a Operation System process

Debug a Process running .NET Core
use Visual Studio Attach to Process

Demo
.NET Core (without DNX)

DNX Installation

How to …

Install DNX Version Manager on Local System
use download script for your Operating System to get *dnvm*

Install latest Version of DNX
use *dnvm* to download and set up latest version

Switch between .NET Framework and .NET Core for Execution
use *dnvm* to switch between versions of DNX

Demo
DNX Installation

Hello World using DNX

How to …

Set up a Project using DNX
create folder and *project.json*

Implement Code and Specify Dependencies
use the editor of your choice or Visual Studio and edit plain text files

Run and Debug the Application
use *dnu* to resotore packages / use *dnx* to run / use the Visual Studio debugger to debug

Demo
Hello World using DNX

WebApp with DNX

How to …

Set up a Web Project using DNX
create folder, *project.json* and add dependencies to MVC 6

Configure the MVC 6 pipeline and add controllers
implement *Startup* class, add mvc modules to ioc and pipeline and implement a *Controller*

Set up the Project to listen to HTTP Requests
use self host or use Helios to host inside IIS

Demo
WebApp with DNX

DNX on Linux

How to …

Install and use DNX on Linux
use Bash shell script to install DNX on Linux, use same command line tools as on Windows

Create and Edit a Project
use the editor of your choice to edit plain text files of a DNX project

Run and Debug a Project on Linux
use command *dnu* and *dnx* to restore and start project

Demo
DNX on Linux

Solution of Multiple DNX Projects

How to …

Set up a Solution Consisting of Multiple DNX Projects
set up proper folder structure and create *global.json*

Set up *global.json* with the Corrent Paths and Dependencies in *project.json* Files
list all directories containing DNX projects belonging to the solution in *global.json*
add references to other projects in *project.json*

Run Startup Project
use *dnx* to run startup project, DNX automatically finds dependencies and compiles it

Demo

Solution of Multiple DNX Projects

Commands

How to …

Use Existing Commands
add the NuGet package containing the command to the dependencies and set up command alias

Define Custom Commands
implement a *Main* method as entry point for your command / reference the project / set up alias

Demo
Custom Commands

Build App and Create Package

How to …

Explicitly Build an App
use *dnu build* to build a project and create assembly files

Prepare an App for deployment/delivery
use *dnu publish* to publish an app into a folder with all its dependencies

Create .NET Independent App Package
use *dnu publish* and the *--runtime* switch to include .NET Core into the app package

Demo
Build App and Create Package

Inside .NET Core and DNX
Summary

Summary

DNX makes .NET Projects independent of…

.NET Runtime
use .NET Framework, Mono or .NET Core

Operating System
run a DNX Project on different Platforms without changes (Windows, Linux, MacOS)

Development Environment
use Visual Studio or your favorite editor