ASP.NET Core with the New MSBuild Based Tooling

CREATING WEB PROJECTS FROM TEMPLATES WITH DOTNET NEW

Wes Higbee
THE DEMYSTIFIER

@g0t4 www.weshigbee.com

project.json   console.csproj

```json
{
  "version": "1.0.0-*",
  "buildOptions": {
    "debugType": "portable",
    "emitEntryPoint": true
  },
  "dependencies": {},
  "frameworks": {
    "netcoreapp1.0": {
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.0.1"
        }
      },
      "imports": "dnxcore50"
    }
  }
}
```

```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp1.1</TargetFramework>
  </PropertyGroup>
<PackageReference Include="Microsoft.NETCore.App" Version="1.0.1" />
</Project>
```

.NET Framework

MyWeb
.csproj

XML

Assembly
Info.cs

C#

packages
.config

XML

MyWeb
.nuspec

XML

.NET Core 1.0

project
.json

JSON

.NET Core Tools 1.0

MyWeb
.csproj

XML

Desired
Package

Desired Package

Desired Package

Benefits of project.json

Clean, json project file

Explicit -> Implicit project files

Package references in project file

Modify project without Visual Studio

Create NuGet package from project file

Cross compiling a single project

- Target multiple frameworks

Cross-platform

Transitive package dependencies

Enhancements

Project to project refs to any .NET project

MSBuild goodness: Composable

- Sdk “Rug”
- Implicit meta-package references

XML

- vs. JSON
- Cleaner project files?

Meaningful element naming

- PackageTargetFallback vs imports
- OutputType vs emitEntryPoint
- VersionPrefix/VersionSuffix vs 1.0.0-*
- ProjectReference (relative path)

Enhancements

Enhancements to all .NET project types

- PackageReference default

Key Takeaways

.NET Core plays well with all .NET project types

Project.json benefits brought back to MSBuild

And further project system enhancements

New templating engine for dotnet new