LEVERAGING NUGET IN YOUR PROJECTS

Low Friction Sharing of Common Code Across Projects

Alien Arc Technologies, LLC

www.alienarc.com
@AlienArcLLC

Duane Newman

DuaneNewman.net
@DuaneNewman

DUANE NEWMAN

ALIEN ARC TECHNOLOGIES

Kansas City .net USER GROUP KCDNUG.ORG

MVP Microsoft® Most Valuable Professional

DUANE & LISA'S DIVING ADVENTURES EST. 2015

DO YOU HAVE SOLUTIONS WITH?

- Code duplication between projects and solutions
- Solutions that are a mess to navigate
- Painful manual build process to update support libraries before you can develop
- A massive code repository because of multiple solutions depending on a few libraries
- Git repositories full of submodules to introduce dependencies
- Problems with projects breaking because you updated the code for a project it depends on, but don't actually need the changes for your solution to work

NUGET PACKAGES CAN HELP!

- Reduce code duplication between projects and solutions
- Provide cleaner solutions that just have the logic specific to that need
- Allow code to easily be separated from each other in different repositories
- Avoid messy scenarios like git submodules
- Eliminate the need for strange developer build steps to get external libraries built before development can proceed.
- Lock projects to a specific version of a dependency and avoid accidental breakage by updating a support project.

GREAT, BUT HOW DO I GET THERE?

- Separate Concerns
  - Identify common libraries
  - Move each library to it's own repository
- NuGet Packages
  - Add support for NuGet package generation
  - Build NuGet package(s)
- Deploy NuGet package(s) to somewhere accessible by dev team and build process
  - Nuget.org
  - Private Nuget Server
  - Windows File Share
- Reference the NuGet Package from your project and enjoy!

ADDING NUGET PACKAGE CREATION

- There are two ways to do this, depending on your library's needs
  - The new simpler way using Visual Studio 2017
    - Ideal for libraries that just target a single "platform"
  - The old (now advanced) way using nupkg files
    - Ideal for cross-platform libraries and more advanced scenarios

USING VISUAL STUDIO 2017

- Project Properties -> Package
  - Fill out all fields
  - Package Id needs to be unique
    - It is typical to follow your assembly naming, but not required
- Build package
  - Either check "Generate NuGet package on build" and build, or
  - Run msbuild from the command line with the /t:pack switch

USING A NUSPEC FILE

```xml
<?xml version="1.0"?>
<package>
    <metadata>
        <id>$id$</id>
        <version>$version$</version>
        <title>NuSpec Demo Package</title>
        <authors>Duane Newman</authors>
        <owners>Duane Newman</owners>
        <requireLicenseAcceptance>false</requireLicenseAcceptance>
        <description>A simple NuSpec Demo</description>
        <releaseNotes>Anything special</releaseNotes>
        <copyright>Copyright 2017</copyright>
        <tags>nuget nuspec demo</tags>
    </metadata>
</package>
```

OTHER COOL THINGS YOU CAN DO

- Run PowerShell Scripts
  - You can even interact with the Visual Studio DTE
- Add files to projects with support for transforms
  - Config transforms: App.config.transform
  - Source Transforms: Filename.ext.pp
- Add Native DLLs (not .net references)
  - <file src="..\ThirdParty\x64\ThirdParty.dll" target="binaries\ ThirdParty.dll" />
- Include custom *.targets files that can be referenced by your project and integrated into the build process
- Make an “App Starter”
  - Contains no, or minimal, functionality, but adds references by dependency on other packages
  - Allows quick creation of new projects with all your standard references
  - Could include boiler plate/template files, and more

THANK YOU!

Duane Newman

Alien Arc Technologies, LLC

@DuaneNewman

DuaneNewman.net

duane@alienarc.com

Github.com/duanenewman/talks