NuGet beyond Hello World

Maarten Balliauw
@maartenballiauw

Who am I?

Maarten Balliauw
Antwerp, Belgium
Developer Advocate, JetBrains
Founder, MyGet
AZUG
Focus on web
- ASP.NET MVC, Azure, SignalR, …
- Former MVP Azure & ASPInsider

Big passion: Azure
http://blog.maartenballiauw.be
@maartenballiauw

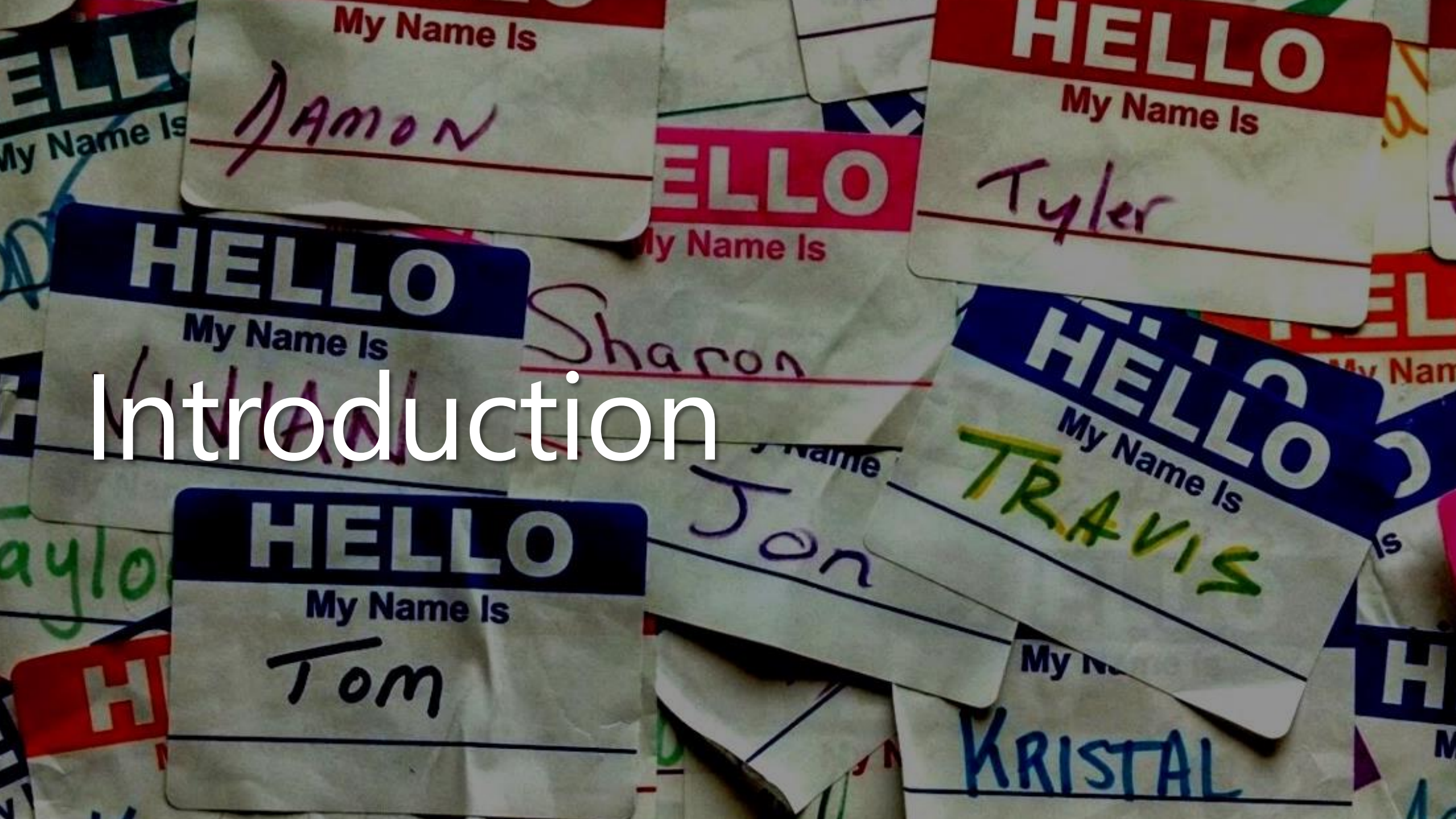
Agenda

Introduction

Maturing NuGet (conventions)

Beyond package dependencies

- Extending tools
- Using client SDK
- Server API



Introduction

NuGet?

Project started in 2011 to make dependency management easier & better

Several things

- Repositories
  - *Public repository at www.nuget.org*
  - *Private repositories everywhere (www.myget.org ☺)*
- Clients & tools
  - *NuGet.exe, Visual Studio, Rider, Xamarin Studio, Paket, …*
  - *dotnet CLI tools*
- Conventions

NuGet evolution

Others have been doing package management for decades

- Perl (CPAN.org)
- Linux (RPM/YUM/APT-GET/…)
- PHP (PEAR, Composer)
- Node (npm)
- Ruby (Gems)

We can learn a great deal from these! (and we did, with .NET Core)

Learnings

- Every project is a package
  - Clearly identifyable
  - Proper Semantic Versioning
  - Tooling makes it easy to package
- Every project becomes discoverable
  - Nice description, release notes, …
  - Add it to a private feed so developers can find it
- Dependencies stay out of source control
- Dependencies are versioned


Maturing NuGet

Package producers are responsible for consumers.

- Maarten Balliauw

It's easy to break the world…

NPM and left-pad http://www.haneycodes.net/npm-left-pad-have-we-forgotten-how-to-program/

- Package removed…

Switch to .NET 4.6.1, can't start application https://github.com/dotnet/announcements/issues/6

- Because of change in behaviour (new `Exception` type thrown)
- Public method signature changed

Conventions!

Don’t break the world!

Never delete a package (unless it’s unused)

https://docs.microsoft.com/en-us/nuget/policies/deleting-packages

Versioning = specify intent

- Detect public API changes
- Use semantic versioning

Targeting = specify what is supported

Metadata and symbols = troubleshooting help

Versioning

Use semantic versioning to state intent

`<major>.<minor>.<patch>-<prerelease>+<metadata>`

Increment **`major`** on breaking change

- Increment **`minor`** on new feature
- Increment **`patch`** for bugfixes
- **`Prerelease`** means it's not stable yet
- **`Metadata`** can annotate build info (not supported on NuGet.org yet)

http://semver.org/

State intent & acceptable changes

Semantic versioning states intent

- Did we break something?
- Do you need this update? (feature/bugfix)
- Are we on scary territory? (prerelease)

Helps consumers specify dependency ranges

```
1.0         = 1.0 ≤ x
(,1.0]      = x ≤ 1.0
(,1.0)      = x < 1.0
[1.0]       = x == 1.0
(1.0,)      = 1.0 < x
(1.0,2.0)   = 1.0 < x < 2.0
[1.0,2.0]   = 1.0 ≤ x ≤ 2.0
1.0-*       = give me all prereleases in 1.0 ≤ x ≤ 1.1
```

Use the tools

API Comparer

- Compare public API surface and detect breaking API's
- https://github.com/ParticularLabs/APIComparer
- Example: http://apicomparer.particular.net/compare/rabbitmq.client/3.4.3...3.5.0

GitVersion

- Auto-generate package version number based on Git info
- https://github.com/GitTools/GitVersion

Targeting

Frameworks and versions

.NET has so many frameworks!

- .NET framework, .NET Core, Xamarin, Tizen, Micro Framework
- Windows Phone 
- Nice list in NuGet repo: FrameworkConstants.cs

Our NuGet package may not (want to) support them all

- Be specific about targeting – state platform support
- Target one or multiple

Multi-targeting

demo

.NET Standard

Multi-targeting can be exhausting (add and test all TFM configurations)

Just select one TFM: a .NET Standard version

.NET Standard to the rescue! https://github.com/dotnet/standard

- Defines API's a platform should implement as a minimum
- Specification describing supported features (not platforms)

- Higher version: more fx features exposed to your library
- Lower version: less features but more platforms supported
- https://docs.microsoft.com/en-us/dotnet/articles/standard/library

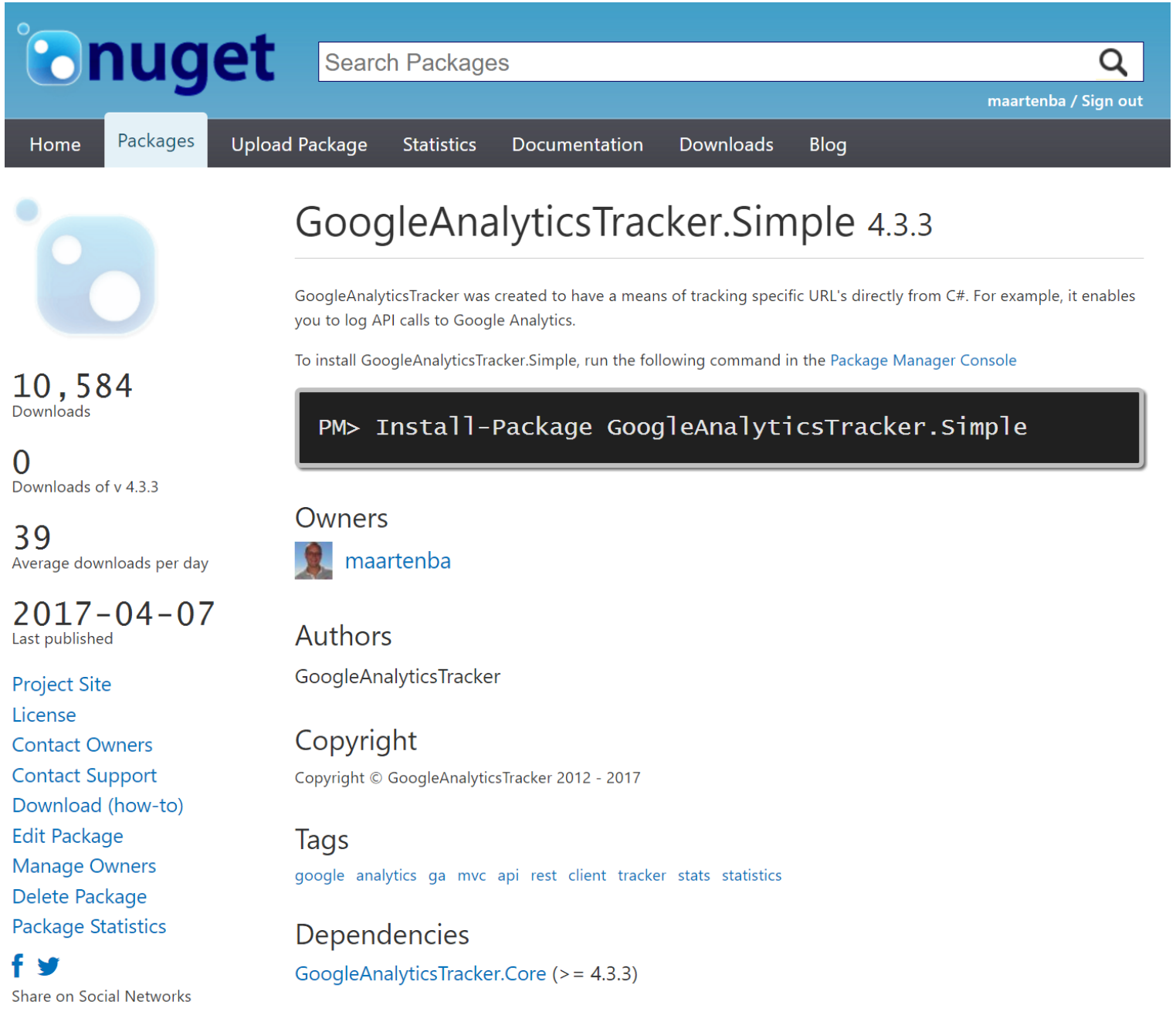
Metadata & symbols

Metadata

Make it easy for people to find:

- Description
- Release notes
- Project site
- License info
- …

(also shown in IDE)



Symbols

Help people troubleshoot!

- Their own code by understanding yours
- Your code (better issues logged / PR)

Publish package symbols

- Symbols packages – **`IncludeSource`** & **`IncludeSymbols`**
  - *Needs symbol server like SymbolSource.org / MyGet.org / …*
- Portable PDB (symbols + sources embedded) – tools like GitLink / SourceLink
  - *Just works™ - at least in theory*

Metadata and symbols

demo


Beyond project
dependencies

NuGet

- More than just a package manager
  - Extensibility (**`dotnet`** CLI)
- NuGet is… a protocol!
  - API for transferring metadata and binaries
    - *Server API*
    - *Client-side libraries*
  - Versioning system built-in

Extending dotnet CLI

Extending dotnet CLI

Why?

- Examples:
  - **`dotnet nuget`** – *the NuGet command line, integrated*
  - **`dotnet ef`** – *Entity Framework command line tools*
- Create tools your team/customers can use
  - **`dotnet protobuf`** – *Companion tool for creating protobuf models*
- After this talk, go and build for me ☺
  - **`dotnet outdated`** – *Scan and show outdated package references*
  - **`dotnet mirror`** – *Mirror referenced packages to a local folder*
  - **`dotnet nuke`** – *Clear all NuGet cache locations*

Extending dotnet CLI

How?

- Create a console application
- Name it **`dotnet-<something>`**
- Package it with **`<PackageType>DotnetCliTool</PackageType>`**
- Install as a CLI tool (or in **`%PATH%`**)

https://github.com/dotnet/docs/blob/master/docs/core/tools/extensibility.md

Extending dotnet CLI

demo

https://github.com/maartenba/dotnetcli-init

Client SDK

Consuming NuGet programatically

Why?

- Find package metadata
- Download and extract package
- Custom build task, custom tool used in CI, …
- Distribute applications / plugins as NuGet packages
  - *Transport and versioning come free!*
  - *NuGet Package Explorer, R# extension manager, Chocolatey, Octopus Deploy, Seq, CMS like EPIserver, SiteCore, Orchard, …*

Consuming NuGet programatically

How?

- Old world: NuGet.Core
- New world: NuGet v3 client
  - *Series of packages, **`NuGet.PackageManagement`** being the main package*
  - *Open-source on GitHub*

- Dave Glick has a good series (part 1, part 2, part 3)

Packages to install

`NuGet.PackageManagement`
Overall management of packages, sources, …

`NuGet.Protocol.Core.v2`
V2 protocol client (OData)

`NuGet.Protocol.Core.v3`
V3 protocol client (JSON and randomness)

`NuGet.ProjectManagement`
Project system (inside IDE and inside file system)

`NuGet.Packaging`
Read and write .nuspec and .nupkg

`NuGet.Versioning`
Implements working with versions and ranges

`NuGet.Commands (optional)`
Command-line commens (spec, pack, publish, …) implemented as static classes mostly

NuGet, via C#

demo

A few things to know…

All operations performed via resources (e.g. `PackageSearchResource`, `ListCommandResource`, …)

All operations performed against a project

- Folder (note: only supports installing as there is no tracking)
- PackagesConfig
- MSBuild
- Your own implementation

Package manager orchestrates all operations

Package repository holds packages (can be remote or local)

SearchPortal

demo

Server API

NuGet talks to a repository

Can be on disk/network share

Or remote over HTTP(S)

2* API's

- V2 – OData based (used by pretty much all NuGet servers out there)
- V3 – JSON based (available on NuGet.org and MyGet.org)

** This is a lie…*

V2 Protocol

OData, started as "OData-to-LINQ-to-Entities" (V1 protocol)

Optimizations added to reduce # of random DB queries (VS2013+ & NuGet 2.x)

- **`Search`** – Package manager list/search
- **`FindPackagesById`** – Package restore (Does it exist? Where to download?)
- **`GetUpdates`** – Package manager updates

https://www.nuget.org/api/v2 (code in https://github.com/NuGet/NuGetGallery)

V3 Protocol

JSON based

More of a "resource provider" (see client SDK) of various endpoints per purpose

- Catalog (NuGet.org only) – append-only event log
- Registrations – latest incarnation of a package
- Flat container - .NET Core package restore (and VS autocompletion)
- Report abuse URL template
- Statistics
- …

https://api.nuget.org/v3/index.json (code in https://github.com/NuGet/NuGet.Services.Metadata)

V3 Protocol is… interesting

- Service index points to resources
  - `@id`, `@type` (versioned), `comment`
- Client should load-balance on `@type` occurrences
  - NuGet.org does this to perform search etc.: `RetryingHttpClientWrapper` (also has a request queue)
- Many views on the same data
- Specific to a scenario in the client
  - Some compute based, some storage based
  - Search, autocomplete -> compute
  - Registrations, flatcontainer -> storage
  - Catalog -> NuGet.org internal use

Server API

demo

How does it fit together?

- User uploads to NuGet.org
  - Data added to database
  - Data added to catalog (append-only data stream)
  - Various jobs run over catalog using a cursor
    - *Registrations (last state of a package/version), reference catalog entry*
    - *Flatcontainer (fast restores)*
    - *Search index (search, autocomplete, NuGet Gallery search)*

How does it fit together?

User uploads to NuGet.org

- Data added to database
- Data added to catalog (append-only data stream)
- Various jobs run over catalog using a cursor
  - *Registrations (last state of a package/version), reference catalog entry*
  - *Flatcontainer (fast restores)*
  - *Search index (search, autocomplete, NuGet Gallery search)*

NuGet.org catalog

demo

Catalog data

- Updates (add/update)
- Deletes
- Chronological
  - Can continue where left off
  - Can restore NuGet.org to a given point in time
- Why use it?
  - Mirror locally
  - Send notifications when certain packages changed
  - Build own index, build own datastore based on data
  - E.g. https://packagesearch.azurewebsites.net/
  - …

Other extensibility points

Other extensibility points

NuGet Credential Provider
https://docs.microsoft.com/en-us/nuget/api/nuget-exe-credential-providers

NuGet package download
https://github.com/NuGet/Home/wiki/NuGet-Package-Download-Plugin


IN MEMORIAM
MEN OF THIS BANK WHO GAVE
THEIR LIVES IN THE GREAT WAR
MCMXIV
MCMXVIII

Conclusion

Conclusion

Go out and create packages!

Follow conventions, help your consumers, help mature NuGet
- Versioning
- Targeting
- Symbols

Consider NuGet not only as package dependencies but as a protocol
- Tooling extensibility
- Protocol for metadata and versioning

Thank you!

http://blog.maartenballiauw.be
@maartenballiauw

Need training, coaching, mentoring, performance analysis?
Hire me! https://blog.maartenballiauw.be/hire-me.html