

DIALOG

Elsinore 2023

D08 - Using Packages

Morten Kromberg



ToDo List

```
:If ~WorldDestroyed  
  ReadMail  
:EndIf
```



Is the World Destroyed?

```
▽ r←WorldDestroyed
[1] r←HttpCommand.Get
    'http://www.hasthelargehadroncolliderdestroyedtheworldyet.com/atom.xml'
[2] r←XML r.Data
[3] r←'NOPE.'≠>r[r[;1 2]i2 'content';3]
▽
```



Read Mail

```
    ▽ r←ReadMail;server;user;pass;etc;etc;z  
[1...7] A Set up user id, passwords, server address, etc etc  
[8]     client←NEW MailKit.Net.Pop3.Pop3Client  
[9]     starttls←MailKit.Security.SecureSocketOptions.StartTls  
[10]    client.Connect server 110 starttls ct  
[11]    client.Authenticate user pass ct  
[12]    'You have ',(n←client.Count),' message(s)'  
[13]    ...more stuff...
```



HttpCommand APL Package from Tatin

Dependencies

```
▽ r←WorldDestroyed
[1] → r←HttpCommand.Get
      'http://www.hasthelargehadroncollider
      destroyedtheworldyet.com/atom.xml'
[2]   r←⊂XML r.Data
[3]   r←'NOPE.'≠>r[r[;1 2]⊔2 'content';3]
▽
```

```
▽ r←ReadMail;server;user;pass;etc;etc;z
[1...7]
[8]   client←⊂NEW MailKit.Net.Pop3.Pop3Client
[9]   starttls←MailKit.Security.SecureSocketOptions.StartTls
[10]  client.Connect server 110 starttls ct
[11]  client.Authenticate user pass ct
[12]  'You have ',(⊃n←client.Count),' message(s)'
[13]  ...more stuff...
```

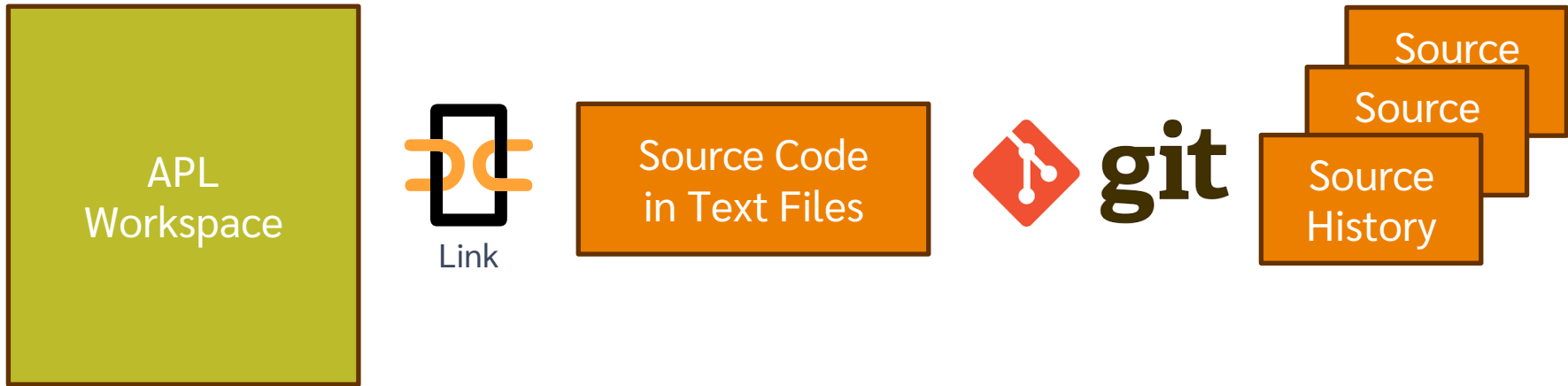
MailKit
C# Package
from NuGet



Demo – Run It



Then There was Link (and git/svn etc)

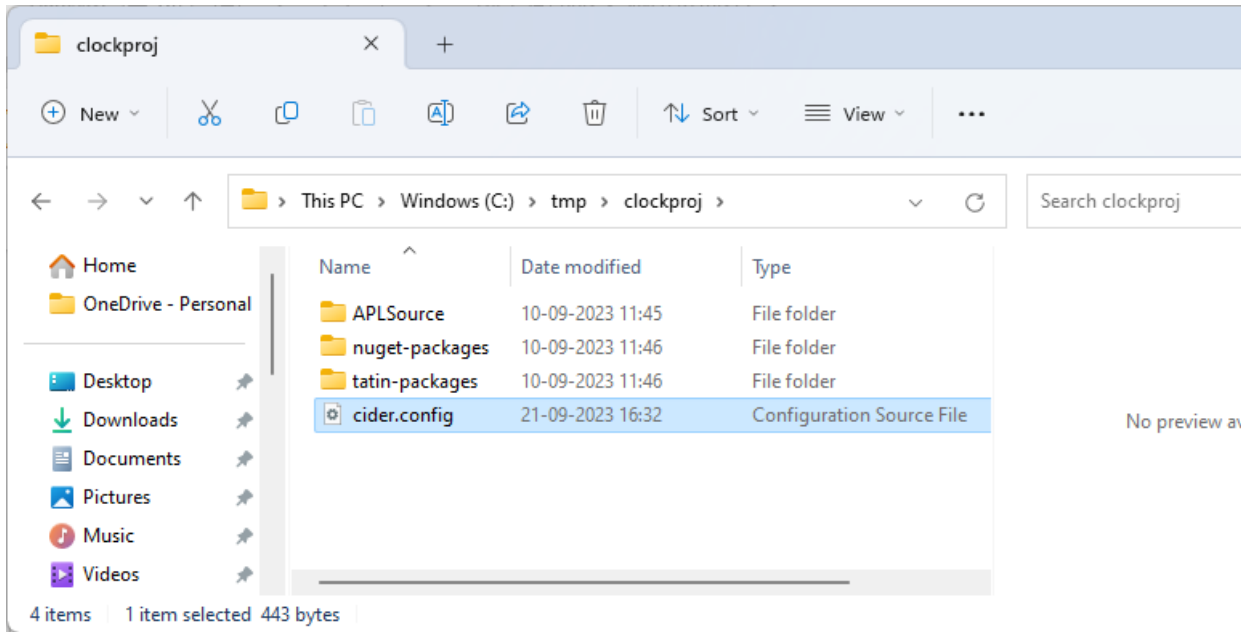


Next Step: Cider!

- Load other code that we **depend** on
- Run some code on **opening** the project
- Run a **build** function
- Decide **where** to load the code
- Run **tests**
- Set **Link options** to be used when loading the source code
- Set **IO, ML**

```
cider.config
x + - □
File Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages"
      tatin: "tatin-packages"
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj"
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```


A Cider Project



```
ciderr.config
File Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

So... What is a Package?



(From Longman Dictionary of Contemporary English)



A Project is...

Source Code +

- ◆ Dependencies (packages) loaded from a package manager
- ◆ Environment configuration
- ◆ Development tools and processes
- ◆ Can be opened and "set up" by a Project Manager (Cider)

A Package is...

A "build" of a project...

- ◆ In a standard format
- ◆ Can be **found, downloaded** and **installed** by a "Package Manager"
- ◆ Cider supports the development of Tatin Packages
- ◆ Cider can load Tatin + NuGet Packages



Tatin



Package manager for Dyalog APL

A tasty way to package APLs

48 Packages

NuGet



Package manager for .NET

Related to "Chocolatey"

~~361,905~~ 374,297 Packages



```
]z←tatin.listPackages
{α,≠ω}⊔{(-1+ωι'-')↑ω}¨3↓z[;1]
aplteam 42
davin 4
dyalog 2
```

```
¨2↑z
dyalog-HttpCommand 1
dyalog-Jarvis 1
```



Finding Packages – www.tatin.dev



List of packages

Package name	Description	Major Versions	Project URL	OS	UC	Tags
aplteam-ADOC	Automated generation od documentation	1	github.com	Lin, Mac, Win	Yes	documentation
aplteam-APLGit2	Git interface from Dyalog APL via Git Bash	1	github.com	Lin, Mac, Win	Yes	apl-git-interface
aplteam-APLGUI	Collection of GUI utilities	1	github.com	Win		gui-tools,gui
aplteam-APLProcess	Start an APL process from within Dyalog APL	1	github.com	Lin, Mac, Win		process
aplteam-APLTreeUtils2	General utilities required by most members of the APLTree library	1	github.com	Lin, Mac, Win		tools,utilities
aplteam-Cider	A project manager for Dyalog APL that cooperates with Tatin	1	github.com	Lin, Mac, Win	Yes	project-management
aplteam-CodeBrowser	Tool useful for code reviews	1	github.com	Lin, Mac, Win	Yes	code-browsing,code-reviews
aplteam-CodeCoverage	Monitors which parts of an application got actually executed	1	github.com	Lin, Mac, Win		code-coverage,test-framework,unit-tests
aplteam-CommTools	Communication tools for interactions in the session: YesOrNo and Select	1	github.com	Lin, Mac, Win		communication-tools,yes-or-no,select-tool
aplteam-Compare	Allows comparing and merging objects in the workspace with a file or a file with another file	3	github.com	Lin, Mac, Win		comparison-tool,merge-tool
aplteam-CompareFiles	Cover for comparison utilities	1	github.com	Lin, Mac, Win	Yes	file-comparison,comparison-utilities

Finding Packages



Tatin Registry

List of packages

Package name	Description	Major Versions	Project URL	OS	UC	Tags
aplteam-DotNetZip	Zippping and unzipping with .NET Core on all major platforms	2	github.com	Win		zip-tools
aplteam-SevenZip	Zip files with the Open Source tool 7Zip	1	github.com	Win		zip-tools
aplteam-ZipArchive	Zippping and unzipping with .NET on Windows and zip/unzip on other platforms	2	github.com	Lin, Mac, Win		zip-tools

Created by Tatin version 0.100.1+1627 from 2023-08-28 under Linux-64 18.2.45645.0 S Runtime — Bugs, questions, problems: info@tatin.dev



Finding Packages



Tatin Registry

List of packages

process

Package name	Description	Major Versions	Project URL	OS	UC	Tags
aplteam-APLProcess	Start an APL process from within Dyalog APL	1	github.com	Lin, Mac, Win		process
aplteam-Execute	Start a process from within APL	1	github.com	Win		process,run-applications

48 packages is enough to (sometimes) make it difficult to decide which one to use (and "dyalog-APLProcess" is yet to come 😊)





Tatin Registry

Home page of group "dyalog"

The dyalog group

This group contains packages published and supported by Dyalog, LTD.

Email address:

Edit

Packages owned by "dyalog"

Package name	Description	OS	Tags
dyalog-HttpCommand	Utility to execute HTTP requests	Lin, Mac, Win	
dyalog-Jarvis	JSON and REST Web Service Framework	Lin, Mac, Win	



Details of <aplteam-CodeCoverage-0.10.0>

```
{
  api: "CodeCoverage",
  assets: "",
  date: 20230323.182755,
  description: "Monitors which parts of an application got actually executed",
  documentation: "",
  files: "",
  group: "aplteam"
  io: 1,
  license: "MIT",
  lx: "",
  maintainer: "kai@aplteam.com",
  minimumApiVersion: "18.0",
  ml: 1,
  name: "CodeCoverage",
  os_lin: 1,
  os_mac: 1,
  os_win: 1,
  project_url: "https://github.com/aplteam/CodeCoverage",
  source: "APLSource/CodeCoverage.aplc",
  tags: "code-coverage,test-framework,unit-tests",
  userCommandScript: "",
  version: "0.10.0+55",
}
```



Tatin Registry

Dependencies of "aplteam-CodeCoverage-0.10.0"

Dependencies

[aplteam-APLTreeUtils2-1.1.3](#)

[aplteam-Tester2-3.3.1](#)

[aplteam-CommTools-1.3.0](#)



]Tatin.ListPackages

```
]Tatin.ListPackages -group=dyalog
```

```
Registry: https://tatin.dev
```

Group & Name	# major versions
-----	-----
dyalog-HttpCommand	1
dyalog-Jarvis	1

```
]Tatin.ListPackages -tag=crypto
```

```
Registry: https://tatin.dev
```

Group & Name	# major versions
-----	-----
aplteam-HashPasswords	1

```
]tatin.listtags
tags from https://tatin.dev
-----
apl-git-interface
build
calculations
chm
code-browsing
code-coverage
code-reviews
command-generation
communication-tools
comparison-tool
comparison-utilities
components
config-files
converter
copy
cryptography
date
dates
...
...
utilities
validation
webservice
windows-event-log
windows-registry
wincsp-interface
write
yes-or-no
zip-tools
```



Adding a Tatin Dependency

- Example: I use `HttpCommand` in just about every new project
- To add it to our Cider project:

```
]Cider.AddTatinDependencies HttpCommand  
1 Tatin dependency added:  
  dyalog-HttpCommand-5.2.0
```

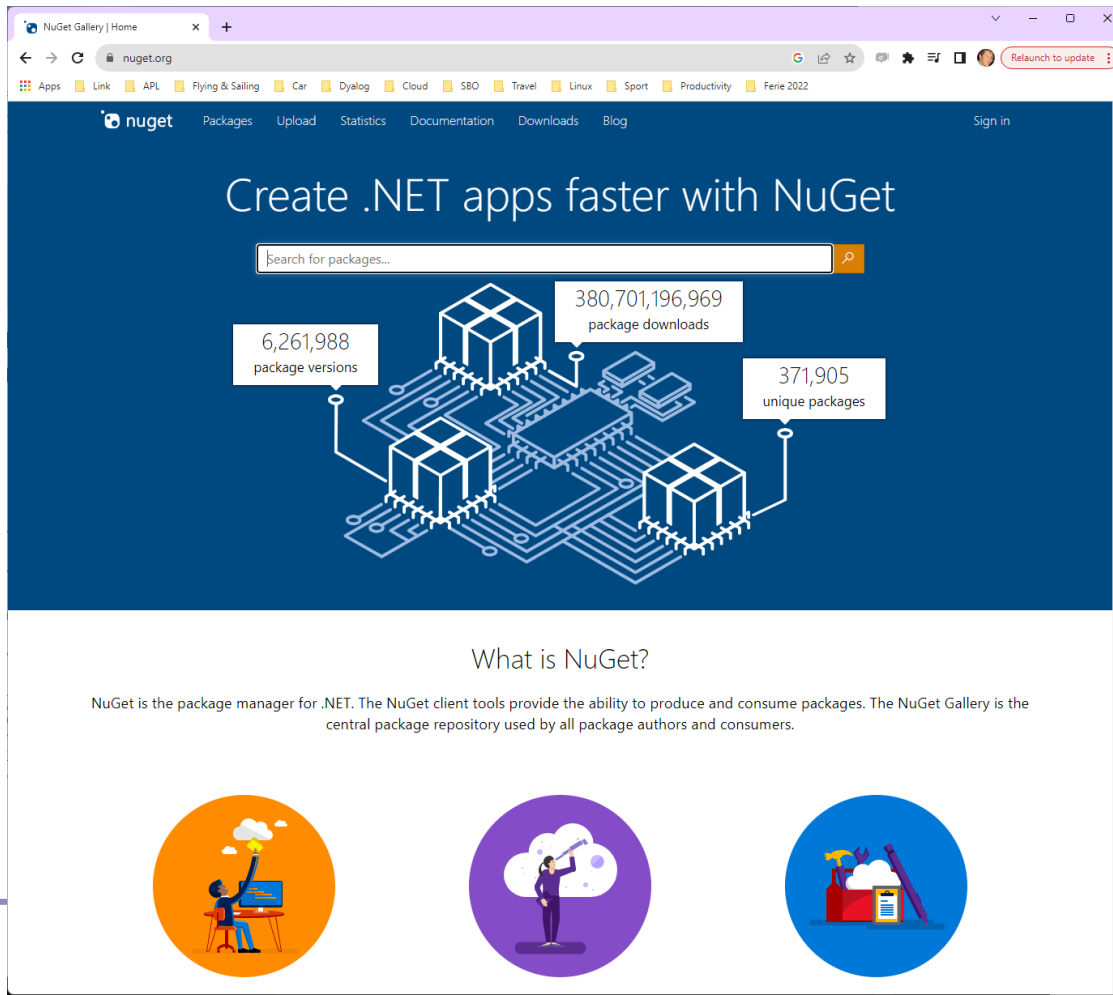
- Since we did not specify a version, we get the latest.
- A reference is created to the loaded package within our project space:

```
D08.HttpCommand.Get 'www.dyalog.com'  
[rc: 0 | msg: | HTTP Status: 200 "OK" | #Data: 22580]
```



NuGet

- ◆ NuGet is the .NET package manager
- ◆ You can use NuGet packages from Dyalog APL, with .NET 6.0 or later



The screenshot shows the NuGet Gallery website. The header includes the NuGet logo and navigation links: Packages, Upload, Statistics, Documentation, Downloads, and Blog. A search bar is prominently displayed with the text "search for packages...". Below the search bar, three statistics are presented with callout boxes: "6,261,988 package versions", "380,701,196,969 package downloads", and "371,905 unique packages". The background features a stylized illustration of a circuit board with three 3D package icons. Below the statistics, the heading "What is NuGet?" is followed by a paragraph: "NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers." At the bottom, there are three circular icons: an orange circle with a person at a computer, a purple circle with a person holding a telescope, and a blue circle with a person at a workbench.

NuGet Gallery | Home

nuget.org

Apps Link APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity Ferie 2022

nuget Packages Upload Statistics Documentation Downloads Blog Sign in

Create .NET apps faster with NuGet

search for packages...




6,261,988 package versions

380,701,196,969 package downloads

371,905 unique packages

What is NuGet?

NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.



[Microsoft].NET History

- .NET has been around for 20+ years. The old "Framework" is being replaced by an open source, cross-platform .NET, initially known as ".NET Core".

Name	Platforms	Version Numbers
Microsoft.NET Framework	Windows	1 2 3 4.0 4.8.1
".NET Core"	Windows Linux macOS	1 2 3
↳ ".NET"	Windows Linux macOS	5.0 6.0 7.0 8.0

- Dyalog v18.0 added a bridge to .NET Core 3, to complement the 20 year old bridge to the .NET Framework.
- v18.2 was tested with "Core" 3.1 but works with 5.0 and later
- v19.0 will target 8.0 (Long Term Support version due on Nov 8th 2023)



Finding NuGet Packages (HARD!!!)

The screenshot shows a web browser window with the address bar at `nuget.org/packages?q=fun`. The page header includes the NuGet logo, navigation links for Packages, Upload, Statistics, Documentation, Downloads, and Blog, and a Sign in button. A search bar contains the text 'fun'. Below the search bar, the page displays '11,706 packages returned for fun' and a 'Sort by' dropdown menu set to 'Relevance'. On the left, there are filter sections for Frameworks (listing .NET, .NET Core, .NET Standard, and .NET Framework) and Package type (listing All types, Dependency, .NET tool, and Template). The main content area shows two search results:

- Microsoft.NET.Sdk.Functions** by: Microsoft nugetazurefunctions
85,667,952 total downloads | last updated 5 months ago | Latest version: 4.2.0
azurefunctions
Build SDK for Azure Functions
- Microsoft.Azure.Functions.Analyzers** by: azure-sdk Microsoft
33,538,913 total downloads | last updated 21/05/2021 | Latest version: 1.0.0
Azure Functions analyzers
This package provides development time code analysis for C# Azure Functions.

Adding a NuGet Package

- Example: NuGet contains a very simple package called "Clock".
- We can add it to our Cider project (by default, we get the latest version):

```
]Cider.AddNuGetDependencies Clock  
Clock 1.0.3
```

- A reference to a namespace hosting the .NET package is created:

```
#.clockproj.Clock.UtcNow.(Hour Minute)  
14 43
```

- In fact, the namespace is empty except for `using`:

```
clockproj.Clock.using  
,c:/tmp/clockproj/nuget-packages/published/Clock.dll
```

NuGet support currently requires .NET 6.0, 7.0 or 8.0

Support for "Framework" packages MAY follow



Demo – Build It



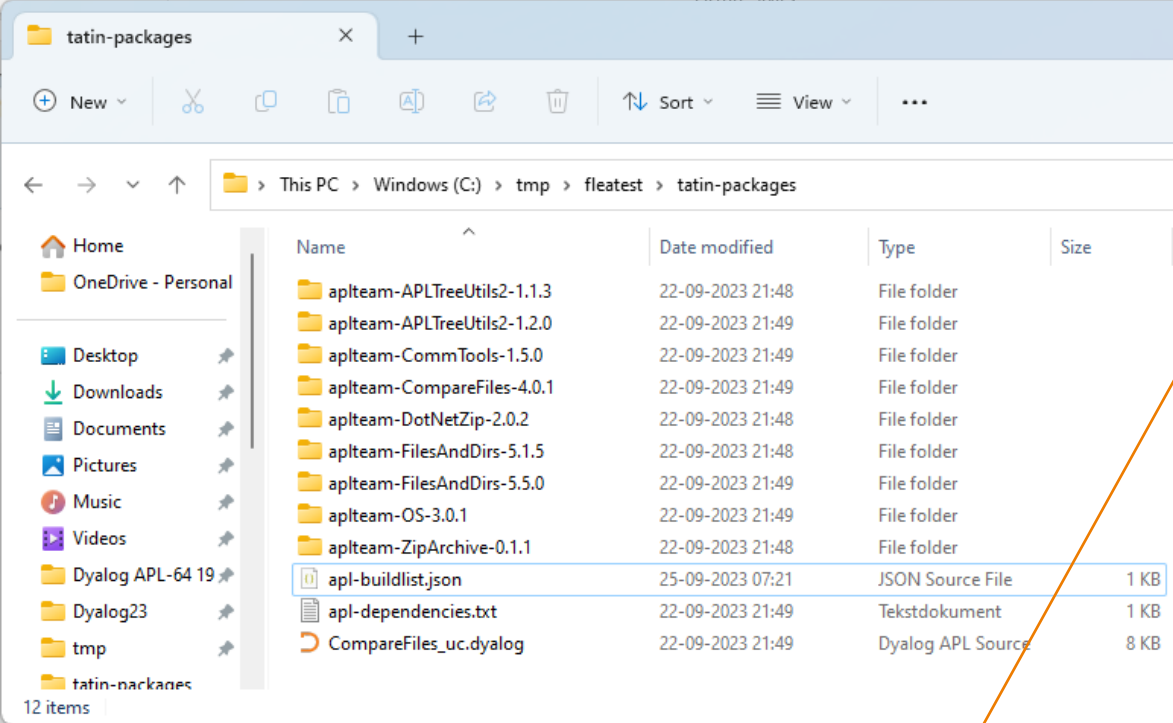
Dependencies of Dependencies

Great fleas have little fleas upon their backs to bite 'em,
And little fleas have lesser fleas, and so *ad infinitum*.

Both Tatin and NuGet will automatically load such dependencies

Augustus De Morgan was a British mathematician and logician. He formulated De Morgan's laws and introduced the term mathematical induction, making its idea rigorous.

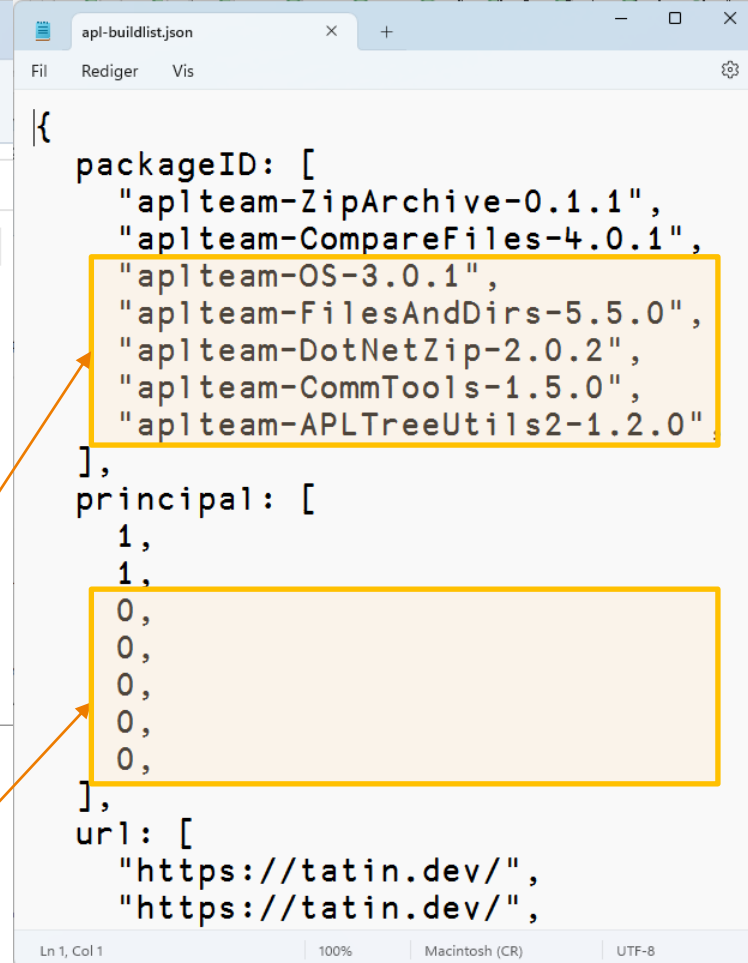
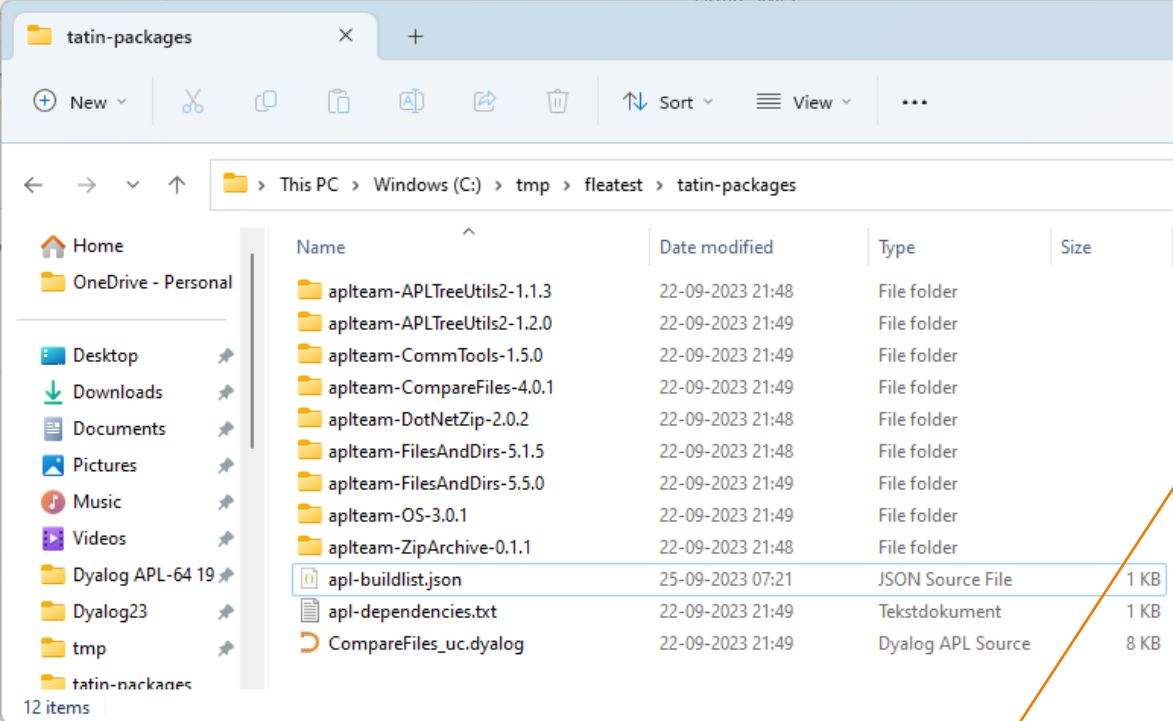




```
apl-buildlist.json
{
  packageID: [
    "aplteam-ZipArchive-0.1.1",
    "aplteam-CompareFiles-4.0.1",
    "aplteam-OS-3.0.1",
    "aplteam-FilesAndDirs-5.5.0",
    "aplteam-DotNetZip-2.0.2",
    "aplteam-CommTools-1.5.0",
    "aplteam-APLTreeUtils2-1.2.0",
  ],
  principal: [
    1,
    1,
    0,
    0,
    0,
    0,
    0,
  ],
  url: [
    "https://tatin.dev/",
    "https://tatin.dev/",
  ]
}
```

"Principal" dependencies (that we added)





"Lesser" fleas



"Principal" dependencies (that we added)

The screenshot shows a Windows File Explorer window with the following details:

- Address bar: This PC > Windows (C:) > tmp > D08Complete > nuget-packages > published
- Left sidebar: Home, OneDrive - Personal, Desktop, Downloads, Documents, Pictures, Music, Videos
- Main pane table:

Name	Date modified	Type	Size
Folder	13-10-2023 12:12	File folder	
BouncyCastle.Cryptography.dll	21-04-2023 14:06	Application extens...	6.903 KB
MailKit.dll	02-09-2023 19:55	Application extens...	863 KB
MimeKit.dll	02-09-2023 19:21	Application extens...	1.163 KB
nuget-packages.deps.json	06-10-2023 16:09	JSON Source File	5 KB
nuget-packages.dll	06-10-2023 16:09	Application extens...	4 KB
nuget-packages.pdb	06-10-2023 16:09	PDB File	11 KB
System.Security.Cryptography.Pkcs.dll	24-05-2023 16:39	Application extens...	256 KB



Where Do Dependencies Go?

```
]Cider.OpenProject C:\tmp\fleatest  
Project successfully loaded and established in "#.fleatest"
```

```
)cs fleatest
```

```
#.fleatest
```

```
  [NL -9
```

```
CiderConfig CompareFiles ZipArchive
```

Our Dependencies

```
  CompareFiles
```

```
  #._tatin.aplteam_CompareFiles_4_0_1.API
```

```
    ;#._tatin.[NL -9
```

```
    aplteam_APLTreeUtils2_1_2_0
```

```
    aplteam_CommTools_1_5_0
```

```
    aplteam_CompareFiles_4_0_1
```

```
    aplteam_DotNetZip_2_0_2
```

```
    aplteam_FilesAndDirs_5_5_0
```

```
    aplteam_OS_3_0_1
```

```
    aplteam_ZipArchive_0_1_1
```

Lesser Fleas

```
  #._tatin.aplteam_CompareFiles_4_0_1.[NL -9
```

```
  API APLTreeUtils2 Admin CommTools ComparisonTools FilesAndDirs TatinVars
```



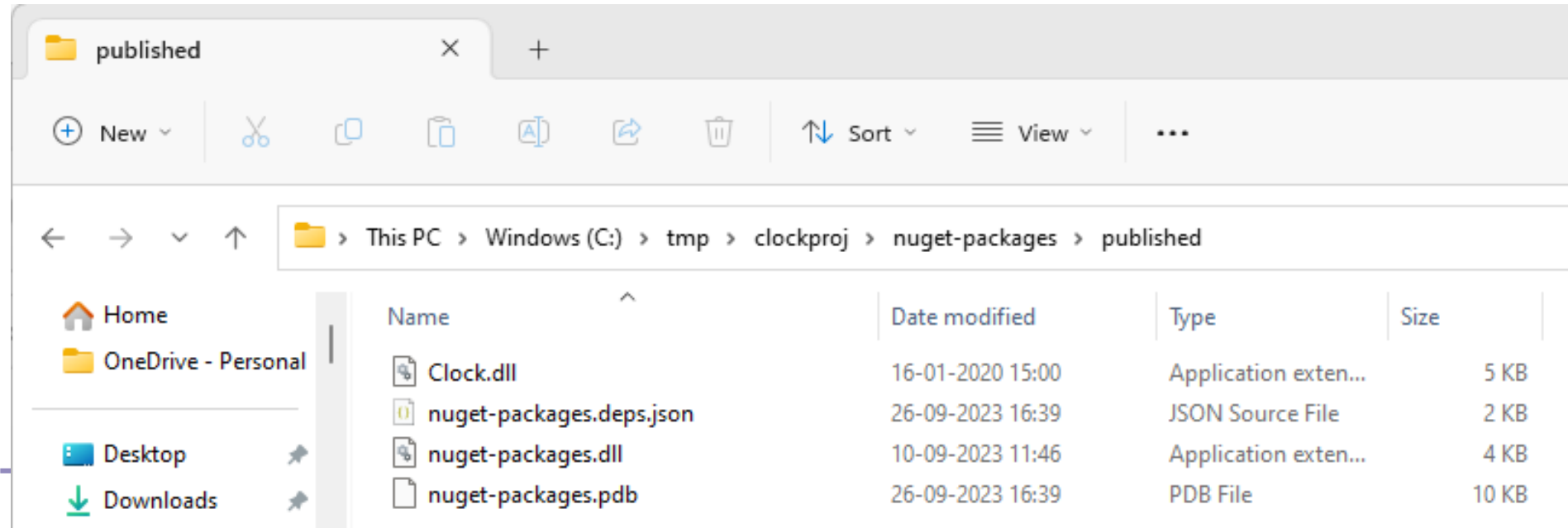
dotnet command-line tool

- ◆ Under Windows, Linux and macOS, .NET provides a "dotnet" command which:
 - ◆ Creates .NET projects that we use to define and manage dependencies (complete with a C# class that we never use)
 - ◆ Adds Dependencies
 - ◆ "Publishes" collections of DLLs that implement packages
- ◆ Dyalog's NuGet support depends heavily on this
 - ◆ We just set `□USING` to point to the published DLLs
 - ◆ The alternative is to try to replicate poorly documented .NET behaviours



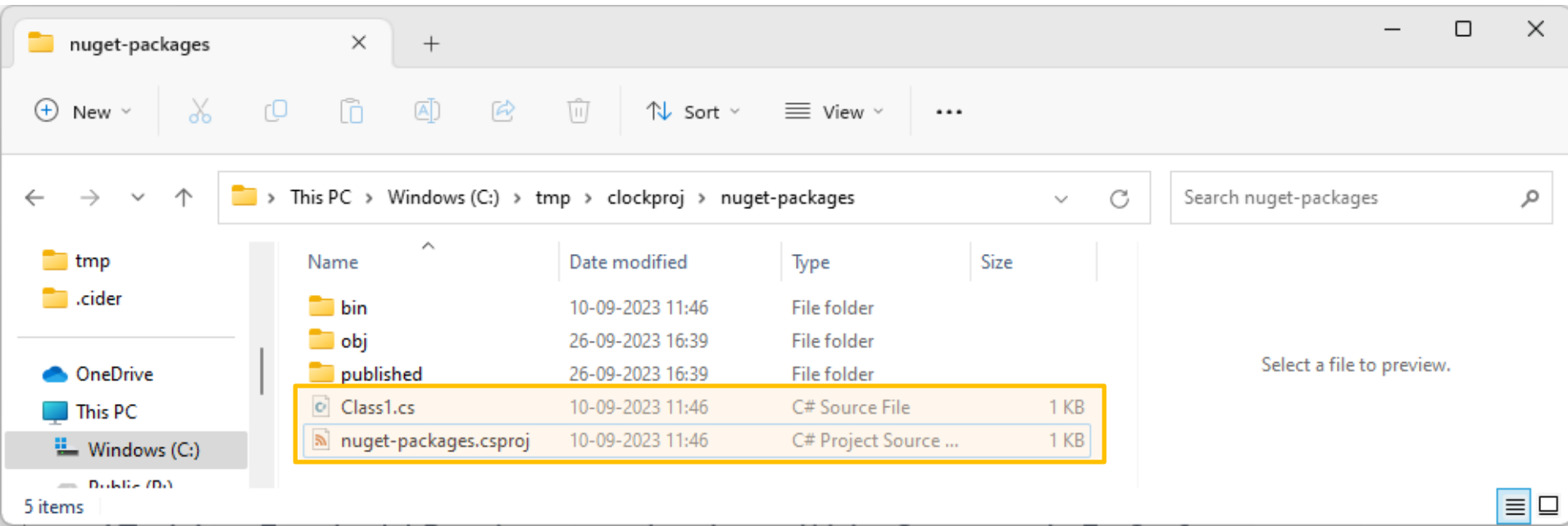
NuGet Packages – Under the Covers

- ◆ NuGet DLL's go in a folder called "published"



NuGet Packages – Under the Covers

- The dotnet command line tool has created some C# code which "pretends" to use the NuGet packages



Same Same but Different

Tatin

```
apl-buildlist.json
{
  packageID: [
    "dyalog-HttpCommand-5.2.0",
  ],
  principal: [
    1,
  ],
  url: [
    "https://tatin.dev/",
  ],
}
```

#.projectSpace.HttpCommand

NuGet

```
nuget-packages.csproj
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <RootNamespace>nuget_packages</RootNamespace>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <LangVersion>Latest</LangVersion>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Clock" Version="1.0.3" />
  </ItemGroup>
</Project>
```

#.projectSpace.Clock



The Cast, in order of appearance



Link Synchronises Source Files and Workspace
The workspace and source files are "Linked"



Cider is a Project Manager
A Project is a linked source folder,
a config file, plus optional dependencies



Tatin is the APL Package Manager
A Package is a project wrapped up for consumption by others



NuGet is the .NET Package Manager
The Dyalog.NET Bridge allows APL to use .NET libraries



History

- ◆ **Tatin** development started in 2020 using the **Acre** project management system
 - ◆ We decided that we needed a "more agnostic" / "less opinionated" project management system to base Tatin development upon
- ◆ **Cider** was born in 2021
 - ◆ Initially as an internal tool for Tatin development
 - ◆ Support for NuGet packages added in 2023
- ◆ **Tatin** is now close to v1.0, we think
- ◆ **Cider** still a prototype (v0.36)
 - ◆ Cider is likely to evolve significantly in next year or two
- ◆ **Cider** is based on **Link**, which is now at v4.0



Cider and Tatin "To Do" lists

- ◆ Review of Names & Messages
 - ◆ Dyalog to help with Documentation
- ◆ Shell-callable API for installation
- ◆ Ability to manage Local / Intermediate package stores within an organisation
- ◆ Import part of a package (e.g. dfns cmpx)?
- ◆ Actually running tests and builds for you



Link Road Map

Link v4.0 Highlights

- ◆ Configuration Files (incl "Global" config)
- ◆ Link single Class or Namespace file
- ◆ Create/Export/Import default to current namespace if none supplied
- ◆ Support for character vectors, matrices and vec-of-vecs in simple text files
- ◆ Link now being used by APL interpreter to load user code at startup

Link 5 & 6

- ◆ Crawler which will periodically compare workspace to source folders
 - ◆ Postponed from 3.0 and 4.0
 - ◆ And again from 4.0 to 5.0
- ◆ Create a proper API
 - ◆ Likely to be 6.0, after the Crawler is done



Dyalog APL Road Map Items

In the interpreter itself...

- Ability launch the APL interpreter on a Cider project and have it open & run
- Create virtual environments a la Python
 - Isolate an APL environment including packages
- Possibly add an extension to `⎕USING` that allows easier references to loaded modules



Dyalog: Making APL more Enjoyable



Tatin



Cider



Dyalog: Making APL more Enjoyable

