Getting Data and Code into the Workspace feat. Get Rich Park
Overview

]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
Overview

]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get
EXPERIMENTAL.Get

“get pretty much anything from pretty much anywhere”
EXPERIMENTAL Get

“get pretty much anything from pretty much anywhere”

- One-stop utility for quickly getting bringing resources in.
EXPERIMENTAL.Get

“get pretty much anything from pretty much anywhere”

- One-stop utility for quickly getting bringing resources in.
- Do not use at run time, as exact results may vary.
]EXPERIMENTAL.Get

Syntax

]get something
EXPERIMENTAL.Get

Syntax

]get something
#.something
Plain name heuristics

```bash
]get dfns #.dfns
]get HttpCommand #.HttpCommand
]get tools #.tools
```
EXPERIMENTAL.Get

File extension detection

]get MyFn.aplf Link-type source files
#.MyFn

]get myVar.apla APL Array Notation files
#.myVar

]get text.charmat Acre-type source files
#.text
EXPERIMENTAL.Get

Data format conversion

- `get obj.json` JS, JSON, JSON5, config files
- `get table.csv` Comma Separated Values
- `get data.xml` eXtensible Markup Language
- `get data.json`
EXPERIMENTAL.Get

Retrieval

]get https://example.com/index.html
#.index
][get https://github.com/abrudz/Kbd
#.Kbd
EXPERIMENTAL.Get

Protocols

]get https://github.com/abrudz/Kbd #Kbd
]get github.com/abrudz/Kbd #Kbd
]EXPERIMENTAL.Get

Examples: unpacking

]get main.zip
#.main

]get github.com/user/repo/tree/main/sub
#.sub
Examples: unpacking

```markdown
]get dfns -only=dxb,dab
#.dfns
    ]map
#
  ·  dfns
  ·  ·  ∇ dab dxb
```
 Examples: unpacking

```bash
)clear

)clear ws

]get dfns -only=dxb,dab -unpack

#.dxb #.dab

]map

#

· ▽ dab dxb
```
EXPERIMENTAL.Get

Examples: options

]get obj.json:txt
#.obj
10↑obj
{"myKey":1
]get thing -target=⎕SE.there
#.SE.there.thing
EXPERIMENTAL.Get

Examples: options

]get /tmp/repo -sync
#.repo
]link.status

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Directory</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>#.repo</td>
<td>c:/tmp/repo</td>
<td>14</td>
</tr>
</tbody>
</table>
[OUTPUT.View]
[TOOLS.APLCart]
[WS.Names]
[OUTPUT.Repr]
[EXPERIMENTAL.Get]

DIGITAL

processing

INPUT

DATA

HTTP

CR

XML

JSON

ENGLISH

printing

OUTPUT

[IMPORT]

[PROCESS]

[NGET]

HttpCommand

[SH 'curl ...']

Link.Import

[FIX]

[CSV]

[XML]

[JSON]
Vector of characters: \( \text{\textbackslash \text{nGET} filename} \)

Hello, World!

Vector of character vectors: \( \text{\textbackslash \text{nGET} filename 1} \)

Hello, World!
Getting Data and Code into the Workspace feat. JGet

]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get

IMPORT
SH 'curl ...'
HttpCommand
Link.Import

FIX
CSV
XML
JSON
HttpCommand

- Fetch HttpCommand:
  ⎕SE.SALT.Load'HttpCommand'

- Vector of characters:
  resp←HttpCommand.Get url
  resp.HttpStatus
  200
  resp.Data
HttpCommand

- Object oriented usage:
  \[hc ← SE.SALT.New'HttpCommand'\]

- One-off usage:
  \[((SE.SALT.New'HttpCommand').Do 'GET' url).Data\]
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get

\[NGET
HttpCommand
\]
\[SH 'curl ...'
Link.Import
\]
\[FIX
CSV
\]
\[XML
JSON
\]
SH 'curl ...'

- Fetch raw vector of character vectors (bytes):
  ```
  raw←SH 'curl -L ',url
  unicode←'UTF-8'∘⎕UCS∘⎕UCS¨raw
  ```

- Fetch raw file:
  ```
  SH 'curl -L -o ',outfile,' ',url
  :NDDELETE outfile
  ```
Fetch zip file:
```bash
SH 'curl -L -o ',outfile,',',url

NDELETE outfile
```
Fetch zip file:

```bash
⎕SH 'curl -L -o ',outfile,' ',url
⎕MKDIR dir
:If ⊃'Windows'∊# ⎕WG 'APLVersion'
  ⎕SH 'tar -xf ',outfile,' -C ',dir
:Else
  ⎕SH 'unzip ',outfile,' -d ',dir
:EndIf
⎕NDELETE outfile
```
SH 'curl ...'

- Fetch zip file:
  - SH 'curl -L -o ', outfile, ',url
  - MKDIR tmp,'/',',dir
  - If 'Windows'∊# ⎕WG 'APLVersion'
    - SH 'tar -xf ', outfile, ' -C ', dir
  - Else
    - SH 'unzip ', outfile, ' -d ', dir
  - EndIf
- NDELETE outfile
Getting Data and Code into the Workspace feat. }Get

]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get

}NGET
HttpCommand
}SH 'curl ...'
Link.Import

}FIX
}CSV

}XML
}JSON

IMPORT

PROCESS
Link.Import

□SE.Link.Import '#.repo' '/tmp/repo'
Imported: #.repo ← c:\tmp\repo
EXPERIMENTAL.Get

TOOLS.APLCart

[OUTPUT.View

[WS.Names

[OUTPUT.Repr

Getting Data and Code into the Workspace feat. Get
Get data and code into the workspace feat. 

- **From array:** `←2 ⎕FIX 'foo' 'bar'

- **From file:** `←2 ⎕FIX 'file://foo.aplf'`
Getting Data and Code into the Workspace feat. \texttt{Get}

\texttt{IMPORT:}
- \texttt{NGET}
- \texttt{HttpCommand}
- \texttt{SH 'curl ...'}
- \texttt{Link.Import}

\texttt{PROCESS:}
- \texttt{FIX}
- \texttt{CSV}
- \texttt{XML}
- \texttt{JSON}

\texttt{OUTPUT:}
- \texttt{View}
- \texttt{TOOLS.APLCart}
- \texttt{WS.Names}
- \texttt{OUTPUT.Repr}
- \texttt{EXPERIMENTAL.Get}
CSV

- From array: ⍪CSV ('1,2,3' '4,5,6') θ 4

- From file: ⍪CSV 'foo.csv' θ 4

- For more techniques: is.gd/dyalog_csv
XML

XML '=root><sub>cont</sub>outer</root>'

0 root
1 sub cont
1 outer
]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get

[\text{\texttt{\textasciitilde NGET}}]
[\text{\texttt{\textasciitilde SH 'curl ...'}}]
[\text{\texttt{\textasciitilde Link.Import}}]
[\text{\texttt{\textasciitilde FIX}}]
[\text{\texttt{\textasciitilde CSV}}]
[\text{\texttt{\textasciitilde XML}}]
[\text{\texttt{\textasciitilde JSON}}]
```python
r ← ⎕JSON '{"Abe":31,"Bob":27}'
]names r
r.Abe  r.Bob
      r.Abe
31
```
JSON

`⎕JSON 'M'⊢ '{"Abe":31,"Bob":27}'`
JSON

r←⎕JSON ' {"Abe":31,"Bob":27}'
r←⎕JSON '{Abe:31, /*Bob:27*/}'

DOMAIN ERROR: JSON import: invalid name at offset 1

r←⎕JSON '{Abe:31, /*Bob:27*/}'

^
r←⎕JSON '{(Abe:31,/*Bob:27*/)}'
Getting Data and Code into the Workspace feat. Get

```
r←⎕JSON⍠'Dialect' 'JSON5'⊢'{Abe:31,/*Bob:27*/}'
```

**DOMAIN ERROR: JSON import: invalid name at offset 1**
Overview

]OUTPUT.View
]TOOLS.APLCart
]WS.Names
]OUTPUT.Repr
]EXPERIMENTAL.Get

INTRO

EXPERIMENTAL

IMPORT

PROCESS

]NGET
HttpCommand
]SH 'curl ...
Link.Import

]FIX
]CSV

]XML
]JSON
Next Webinar

June 9: TBA

Remember: BAA webinars every other week
britishaplassociation.org/webinar-schedule-2022
May 19th; June 2nd, 16th, 30th; etc.

More at: apl.wiki/activities