The Dyalog Project Project (DP2)

Morten Kromberg

DYALOG

Sicily 2015

The next best thing
to knowing something is
knowing where to find it.

Samuel Johnson

From “Why APL Programmers don’t use Libraries”
Morten Kromberg, April 2003 (Vector Vol 20, No 1)
http://archive.vector.org.uk/art10004590

The Dyalog Project Project (DP2)

[New] Dyalog Users need a ***COMMON*** way to describe software projects implemented in Dyalog APL. We need to:

- Ideally, start with a prefabricated sample
  - Console Application, ⎕WC GUI App, Web Service, …
- Manage the Source Code that we write (diff, blame, revert, etc)
- Locate Tools and Utilities
- Include and Manage Dependencies
  - Common Tools and Utilities as well as larger Modules
- Build and deploy Runtime Environments
  - Optionally obfuscating / encrypting the source
- Create and run Automated Tests
- On all supported platforms, of course!

Manage The Source Code

- A DP2 project will be a folder with a standard layout and some configuration files
- The configuration or “Project Description” files will themselves be Unicode Text files which can be managed along with the code
- The user decides whether to use SVN, Git[Hub], Mercurial or [the next cool thing] to manage the source
  - We are NOT building yet another source code management tool

- **Any other approach would completely undermine the project. Binary formats are NOT an option for source.**

Interpreter Support for Text Source

- We plan to add support for load/save (including auto-save-on-edit) of textual source to the v15.0 interpreter
  - Future versions of APL may be able to operate without the source code in the workspace (e.g. only “compiled” code)
  - We may be able to preserve source code exactly as entered by the user ☺
  - Support for saving the source of # and individual functions
- SALT will continue to exist and use this layer
  (it currently uses APL code for Unicode file processing)

Build Runtime Environments

- Wide variety of target environments
  - Workspaces and Bound Executables
  - Microsoft.Net Assemblies and COM Components
  - MiSites and Web Services
  - Component Files and External Workspaces
  - “Packages” that can be depended on by other DP2 projects
  - Create your own target using …
- Simple DSL to describe target environments
  - Think: lightweight version of IP Sharp’s LOGOS
- Goal: Support new projects immediately for new users, eventually also “legacy” runtime environments

Installers

- Eventually, we want to be able to build an installer for your runtime environment
  - Check for dependencies at install time
- A bridge too far for v1.0

Locate Tools and Utilities

- We will collect and organize Standard Libraries
  - files, strings, dates, xml, json, sql, parsing, e-mail, error logging
  - … we already have many of these in MiServer …
  - "Cross Platform" if at all possible
  - Searchable online
- These – and everything else related to DP2 – will be provided as open-source repositories (https://github.com/dyalog)
- The tool library should also be easy to use for projects not based on DP2

Manage Dependencies

- Perhaps the hardest piece of design: Declare dependencies on functions, modules or “packages”
  - We will research existing packages: npm, pip, cask, cargo and the GNU APL and Jsoftware package managers for inspiration
- **[Pre]Build:** Copy/download or link to specific versions of an external dependency
- **At Runtime:** Import something from the deployed runtime environment
  - Import entire classes or namespaces
  - Import individual functions from a namespace INTO something
- Indirect dependency via named “resources”, for easy substitution:
  - Run a test with v2.2 rather than v2.1 of a dependency
  - Substitute module “database” with “mockdb”
- Hooks to allow sophisticated user to intercept all file access

Test Automation

- We should include a tool for defining and running automated unit and integration tests

  (waves hands)

- Also a tricky piece of design to get the balance right.
- Possibly best to provide hooks and allow people to add test frameworks – Dyalog will provide one or two simple defaults

Pre-Fabricated Samples

We need to build a collection of sample applications:

- Console / Scriptable Application
- □WC GUI Application with menus, icons and a grid
- A WPF application with menus, routed events, etc
- A Web (MiSite) Site
- A Web Service
- Microsoft.Net Assemblies for various purposes
- And so on…

Utility Libraries

- Files
- Strings
- Dates
- XML
- JSON
- SQL
- Web Client Requests
- Parsing
- E-mail
- [error] logging
- Inverted Database (vecdb)
- WPF
- Win32 GUI (□WC)
- Configuration (INI/REG)
- Crypto
- Dictionaries
- Test Automation (e.g. Selenium)

Status

- Very early design stage
- Version 0.1 “in the spring”
- Expected to take several cycles to reach maturity

Why Will We Succeed This Time?

- Because we must
- Dyalog will now invest significant resources in it

- What can you do?
  - Contribute to design discussions
  - Demand things
  - Contribute libraries or sample applications

One part of knowledge consists in being ignorant of such things as are not worthy to be known.

Crates of Thebes

→ *"DP2" must be easy to use, and easy not to use (or even know about)*