

## Newt, the third prototype

R. S. Doiel, [rsdoiel@caltech.edu](mailto:rsdoiel@caltech.edu)

Caltech Library, Digital Library Development



# What is Newt?

- ▶ A rapid application develop tool
  - ▶ for applications that curate metadata
- ▶ Audience: Libraries, Archives, Galleries and Museums

## Findings from Prototype 2:

*Is Newt and “off the shelf” software enough to create metadata curation applications?*

Short answer is **yes**. Longer answer is more nuanced.

## Findings from Prototype 2:

*Is Newt and “off the shelf” software enough to create metadata curation applications?*

1. Newt's YAML file can grow very large for applications with many data models
2. Model vetting and validation should happen early in the data pipeline, ideally as a generated program and browser side
3. Postgres+PostgREST is a powerful combination but it'd be nice to have something simpler
4. Managing the YAML file can be done conversationally

## Questions raised by Prototype 2:

- ▶ Where do I focus my simplification efforts?
- ▶ How do I ensure that large YAML files remaining human manageable?
- ▶ Mustache template language is a little too simple, what should replace it?

## High level Concepts (remain the same)

- ▶ describe the application you want
- ▶ generate the application you described
- ▶ running the application using a service oriented architecture

## Implementation Concepts (remaining the same)

- ▶ JSON data sources
- ▶ data modeled in YAML
- ▶ routing requests through data pipelines
- ▶ simple template engine renders JSON to HTML



## Themes (remains the same)

- ▶ Pick Simple = (No coding) + (Less coding)
- ▶ Compose applications using data pipelines
- ▶ Avoid inventing new things

## Goal of Prototype 3: Questions to explore

1. What should the default JSON data source be? (dataset+datasetd vs. Postgres+PostgREST)
2. Is generated TypeScript middleware the right fit? (e.g. validation service)
3. Is Handlebars a good fit for managing data views and rendering HTML?

## Changes from last prototype

- ▶ Removed some Go cli (e.g. ws, mustache, newtmustache)
- ▶ Renamed newtrouter -> ndr (Newt Data Router)
- ▶ Added nte (Newt Template Engine) supporting Handlebars templates
- ▶ Generating collection and YAML for dataset+datasetd
- ▶ Generating Handlebars templates
- ▶ Generating TypeScript validator as middleware run via Deno

## Off the shelf (no coding)

- ▶ JSON Data Source
  - ▶ Dataset + datasetd
  - ▶ Postgres + PostgREST
- ▶ newt, ndr, and nte
- ▶ Deno to run generated TypeScript middleware

## Assemble app from YAML (less coding)

- ▶ Create the initial Newt YAML through a conversational TUI
- ▶ Data modeling via a conversational TUI

## How are data models described?

- ▶ A model is a set of HTML form input types
- ▶ Expressed using GitHub YAML Issue Template Syntax
- ▶ Model describes HTML and implies SQL

## How do I think things will work?

1. Interactively generate our application's YAML file
2. Interactively define data models
3. Generate our application code
4. Run `newt generate ...` for primary data source
5. Run `newt run ...` to run the application

Steps one and two are interactive

```
newt init app.yaml  
newt model app.yaml
```



## Step three, generate our code

```
newt generate app.yaml
```

*Create a dataset collection and datasetd YAML file Generate Handlebars templates Wires up routes and template mappings*

## Step four, setup primary JSON data source

### Dataset collection

*Collection generation is done “auto magically” by `newt generate app.yaml`  
`datasetd` YAML file gets generated so Newt can run the `datasetd` JSON API*

Step five, run your application and test

```
newt run app.yaml
```

*Point your web browser at <http://localhost:8010> to test*

Can I run a demo?

Not yet, hopefully in early December 2024.

## Third prototype Status

- ▶ A work in progress (continuing through 2024)
- ▶ Working prototype target date June 2025
- ▶ Using internal applications as test bed

## How much is built?

- ☒ Newt developer tool
- ☒ Router is implemented and working
- ☒ ~~Mustache template engine is working~~ (removed)
- ☒ Newt template engine (supporting Handlebars templates)
- ☐ Modeler (design stage)
- ☐ Generator development (refactor, design stage)

## Insights from prototypes 1 & 2

- ▶ “Off the shelf” is simpler
- ▶ Lots of typing discourages use
- ▶ Explore conversational coding

## Insights from prototypes 1 & 2

- ▶ SQL turns people off, use a code generator
- ▶ Hand typing templates is a turn off, use a code generator
- ▶ Large YAML structures benefit from code generation
- ▶ Automatic “wiring up” of routes and templates very helpful



## What's next to wrap up prototype 3?

- ▶ Refine and simplify Newt YAML syntax
- ▶ Refine data router
- ▶ Retarget, debug and improve the code generator
- ▶ Design and replace template engine

## Out of the box

- ▶ newt the Newt development tool
- ▶ ndr the Newt data router
- ▶ nte the Newt Template Engine

# Unanswered Questions

- ▶ What is the minimum knowledge required to use Newt effectively?
- ▶ Who is in the target audience?

## Someday, maybe ideas

- ▶ A visual programming approach could be easier than editing YAML files
- ▶ Direct SQLite 3 database support and integration
- ▶ Web components for library, archive and museum metadata types
- ▶ A S3 protocol web service implementing object storage using OCFL
- ▶ Generate code which can compile stack into a single binary application

## Related resources

- ▶ Newt <https://github.com/caltechlibrary/newt>
- ▶ Dataset + datasetd <https://github.com/caltechlibrary/dataset>
- ▶ [Handlebars](#) programming languages support

# Thank you!

- ▶ This Presentation
  - ▶ pdf: <https://caltechlibrary.github.io/newt/presentation3/newt-p3.pdf>
  - ▶ pptx: <https://caltechlibrary.github.io/newt/presentation3/newt-p3.pptx>
- ▶ Newt Documentation <https://caltechlibrary.github.io/newt>
- ▶ Source Code: <https://github.com/caltechlibrary/newt>
- ▶ Email: [rsdoiel@caltech.edu](mailto:rsdoiel@caltech.edu)