Newt, the second prototype

R. S. Doiel, rsdoiel@caltech.edu

Caltech Library, Digital Library Development

What is Newt?

- A rapid application develop tool
  - for applications that curate metadata
- Audience: Libraries, Archives, Gallaries 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
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 our simplification efforts?
- How do I ensure that large YAML files remaining human managable?
- Mustache template language is a little too simple, what should replace it?

Goal of Prototype 3: Answer these questions three.

1. Is generating TypeScript programs a suitable way to solve the validator problem?
2. Is HandlebarsJS via TypeScript a good fit for managing data views?
3. What default JSON data sources should be supported? (e.g. Postgres+PostgREST, dataset+datasetd)

High level Concepts (remaing 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)

- data sources
- data models
- routing requests through data pipelines

Themes (remaing the same)

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

Stack changes

- render JSON responses via HandlebarJS template engine
- generated validation code via TypeScript running in Deno

Off the shelf (no coding)

- Data Sources
  - Dataset + datasetd
  - Postgres + PostgREST
  - Solr ??? OpenSearch ???
- Newt Handlebars => Transform JSON into web pages
- Newt Router, ties it all together

Assemble app from YAML (less coding)

- The application you want is described in YAML
- Create the initial Newt YAML through a conversational TUI
- Newt generates the code you need
- Customize by editing the generated code and managing your pipelines

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. Setup a primary data source
5. Run our app with Newt

Steps one and two are interactive

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

Step three, generate our code

```
newt generate app.yaml
```

Created a dataset collection and datasetd YAML file or render SQL, PostgREST conf Render Handlebars templates

Step four, setup primary JSON data source

Dataset collection

Collection generation is done "automagically" by `newt generate app.yaml`
datasetd YAML file gets generated so Newt can run the datasetd JSON API

Step four, setup the primary JSON data source

Postgres+PostgREST

1. Use the generated SQL and configuration
2. Setup and check via `createdb` and `psql`

Step four, setup Postgres and PostgREST

```
createdb app
psql app -c '\i setup.sql'
psql app -c '\i models.sql'
psql app -c '\dt'
```

should this be automated too?

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)
- ☐ Handlebars template engine (planning and design)
- ☐ Generator development (paused)
- ☐ Modeler (design stage)

Insights from prototypes 1 & 2

- “Off the shelf” is simpler
- Lots of typing discourages use

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?

- Implement new template engine, using HandlebarsJS, TypeScript and Deno
- Debug and improve the code generator
- Continue to implement a data modeler
- Generate validation layer written in TypeScript and run by Deno

Out of the box

- Newt (development tool)
- Newt Router
- 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 or integration
- A S3 protocol web service implementing object storage using OCFL
- Web components for library, archive and museum metadata types
- Extend Newt through WASI+WASM run time modules and expose to use in pipelines

Related resources

- Newt https://github.com/caltechlibrary/newt
- Dataset + datasetd https://github.com/caltechlibrary/dataset
- Postgres https://postgres.org + PostgREST https://postgrest.org
- HandlebarsJS 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