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

Goal of Prototype 2: Answer the question.

Is Newt and "off the shelf" software enough to create metadata curation applications?

Prototype 2: findings

1. YAML file can grow very large for a "real world" applications with multiple models of objects
2. Model vetting should happen via generated code but probably not Golang code. TypeScript/JavaScript via Deno seems promising
3. Postgres+PostgREST is a powerful combination but it'd be nice to have something simpler. Dataset collections running behind datasetd seem promising.
4. Managing the YAML file should be more conversational

Goal of Prototype 3: Answer these questions three.

1. Is generating TypeScript programs a suitable way to solve the validator problemw while leaving room for customization?
2. Is TypeScript suitable for generated code to common external services? (e.g. ORCID, ROR)
3. Is HandlebarsJS via TypeScript a good fit for managing data views?
4. What default JSON data sources should be supported? (e.g. Postgres+PostgREST, dataset collection hosted with datasetd)

High level Concepts (remaing the same)

- describe the application you want
- generate the application you described

Implementation Concepts

- data sources
- data models
- routing requests through data pipelines
- rendering JSON responses via template engine (now HandlebarsJS)
- generated validation and proxy code (e.g. TypeScript+Deno)

Themes

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

Off the shelf (no coding)

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

Office the shelf (other data sources)

- ArchivesSpace, RDM -> JSON API
- ORCID, ROR, CrossRef, DataCite -> JSON API
- What about intergrating external services?

Assemble app from YAML (less coding)

- The application you want is described in YAML
- Newt generates the code you need
- Customize by editing the generated code

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 Postgres and PostgREST
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
```

Renders SQL, PostgREST conf, Mustache templates

Step four, setup Postgres and 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.

Second prototype Status

- A work in progress (continuing in 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 (but replaced)
- ☐ Handlebars template engine (in progress)
- ☐ Generator development (in progress)
- ☐ 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?

- Debug and improve the code generator
- Continue to implement a data modeler

Unanswered Questions

- What should be the minimum knowledge needed to use Newt?
- What should come out of the box with Newt?
  - GUI tools via TypeScript+Deno?
  - Web components?
  - Ready made apps?

Someday, maybe ideas

- Direct SQLite 3 database support
- A S3 protocol web service implementing object storage using OCFL
- Web components for library, archive and museum metadata types
- Visual programming would be easier than editing YAML files

Related resources

- Newt https://github.com/caltechlibrary/newt
- 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