

## Newt, the second prototype

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

Caltech Library, Digital Library Development

2024-04-19



Goal: Answer a question.

Is Newt and “off the shelf” enough?

## How I am proceeding

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

## Off the shelf (no coding)

- ▶ Postgres or PostgREST
- ▶ Solr or OpenSearch
- ▶ Newt Mustache => Transform JSON into web pages
- ▶ Newt Router, ties it all together

## Assembling it with YAML (less coding)

- ▶ GitHub YAML issue template syntax described data models
- ▶ YAML describes configuration, routes, pipelines
- ▶ Template language is now Mustache
- ▶ Code generation, “look Mom, no AI!”

## Second prototype status

- ▶ A work in progress (April 2024)
- ▶ Hope to have a working prototype by June 2024
- ▶ Internal applications will serve as test bed

Is there a Demo I can run?

Not yet, hopefully soon.



## What's working, what's not?

- ☒ Router is implemented and working
- ☒ Mustache template engine is working
- ☐ Generator development, in progress

## How do I think things will work?

1. Generate our app YAML
2. Designing our data models
3. Generate SQL, PostgREST config and templates
4. Run generated SQL
5. Run Newt and Test

## How is the data model is described?

- ▶ GitHub YAML Issue Template Syntax
  - ▶ describes HTML
  - ▶ implies SQL

## Step one create our YAML file

```
newt init app.yaml  
Interactively generate app.yaml
```

## Step two define our data models

```
newt modeler app.yaml
```

*Interactively model your data*

## Step three, generate our code

- ▶ Use `newt generate`
  - ▶ SQL
  - ▶ PostgREST config
  - ▶ Templates
- ▶ Edit files if needed

## Step three, generate our SQL files, config

```
newt generate app.yaml postgres setup >setup.sql  
newt generate app.yaml postgres models >models.sql  
newt generate app.yaml postgres >postgrest.conf
```

## Step three, generate templates

```
newt generate app.yaml mustache \  
    create_form app >create_app_form.tpl  
newt generate app.yaml mustache \  
    create_response app >create_app_response.tpl
```



## Step three, generate templates ...

```
newt generate app.yaml mustache \  
    update_form app >update_app_form.tmpl  
newt generate app.yaml mustache \  
    update_response app >update_app_response.tmpl
```

## Step three, generate templates ...

```
newt generate app.yaml mustache \  
    delete_form app >delete_app_form.tmpl  
newt generate app.yaml mustache \  
    delete_response app >delete_app_response.tmpl
```

## Step three, generate templates ... finally

```
newt generate app.yaml mustache read app >read_app.tmpl  
newt generate app.yaml mustache list app >list_app.tmpl
```

## Step three, generate templates

*code generation should be fully automated*

## Step four, run our SQL

```
createdb app
```

```
psql app -c '\i setup.sql'
```

```
psql app -c '\i models.sql'
```

```
psql app -c '\dt'
```

*this should be automated with the newt command*

## Step five, run newt and test

```
newt run app.yaml
```

- ▶ fire up newt, test and debug
- ▶ web browser

## Insights from prototypes 1 & 2

- ▶ “Off the shelf” is simpler
- ▶ Large YAML structures benefit from code generation
- ▶ SQL turns people off, use a code generator
- ▶ Mustache/HTML needs a code generator
- ▶ Automatic “wiring up” of routes and templates is helpful

## Lessons learned, so far

- ▶ Managing routes and pipelines has a cognitive price
- ▶ Keep your pipelines short
- ▶ Web services need a “developer” mode for debugging
- ▶ Lots of typing discourages use



## What's next to wrap up prototype 2?

- ▶ Finish/improve the code generator
- ▶ Implement a data modeler
- ▶ Less typing!

## Newt's challenges

- ▶ Newt is **a work in progress** (April 2024)
- ▶ Newt is missing file upload support

# Unanswered Questions

- ▶ What is the minimum knowledge needed to use Newt?
- ▶ What should come out of the box with Newt?
  - ▶ GUI tools?
  - ▶ Web components?
  - ▶ Ready made apps?

## My wish list . . .

- ▶ SQLite 3 database support
- ▶ Visually programming would be easier than writing YAML files
- ▶ Web components for gallery, library, archive and museum metadata types
- ▶ A simple S3 protocol web service that implements storing object using OCFL

## Related resources

- ▶ Newt <https://github.com/caltechlibrary/newt>
- ▶ Postgres <https://postgres.org> + PostgREST <https://postgrest.org>
- ▶ [Mustache](#) programming languages support
- ▶ Go 1.22, pattern language in HTTP handlers, see <https://pkg.go.dev/net/http#hdr-Patterns>

# Thank you!

- ▶ This Presentation
  - ▶ pdf: <https://caltechlibrary.github.io/newt/presentation2/newt-p2.pdf>
  - ▶ pptx: <https://caltechlibrary.github.io/newt/presentation2/newt-p2.pptx>
- ▶ Newt Documentation <https://caltechlibrary.github.io/newt>
- ▶ Source Code: <https://github.com/caltechlibrary/newt>
- ▶ Email: [rsdoiel@caltech.edu](mailto:rsdoiel@caltech.edu)