MVC in MediaWiki

How to utilize SMW, ArrayFunctions and Scribunto to make your wiki Parsoid-compatible

Marijn van Wezel
April 20, 2023

EMWCon Spring 2023

wikibase solutions

- Parsoid does not support sequential processing of wikitext (T250963).
- This is a problem for extensions that have some side-effect, such as:
  - Extension:Arrays
  - Extension:Variables
  - Extension:HashTables
- Because the order in which things are parsed is non-deterministic, you can no longer rely on things being parsed before you use them.

```
{{#vardefine: foobar | Hello World!}}

{{#var: foobar}}
```

```
1 {{#vardefine: foobar | Hello World!}}

2 {{#var: foobar}} → Hello World!
```

```
2 {{#vardefine: foobar | Hello World!}}

1 {{#var: foobar}} ↯
```

- How do we keep using these technologies in the Parsoid-era?

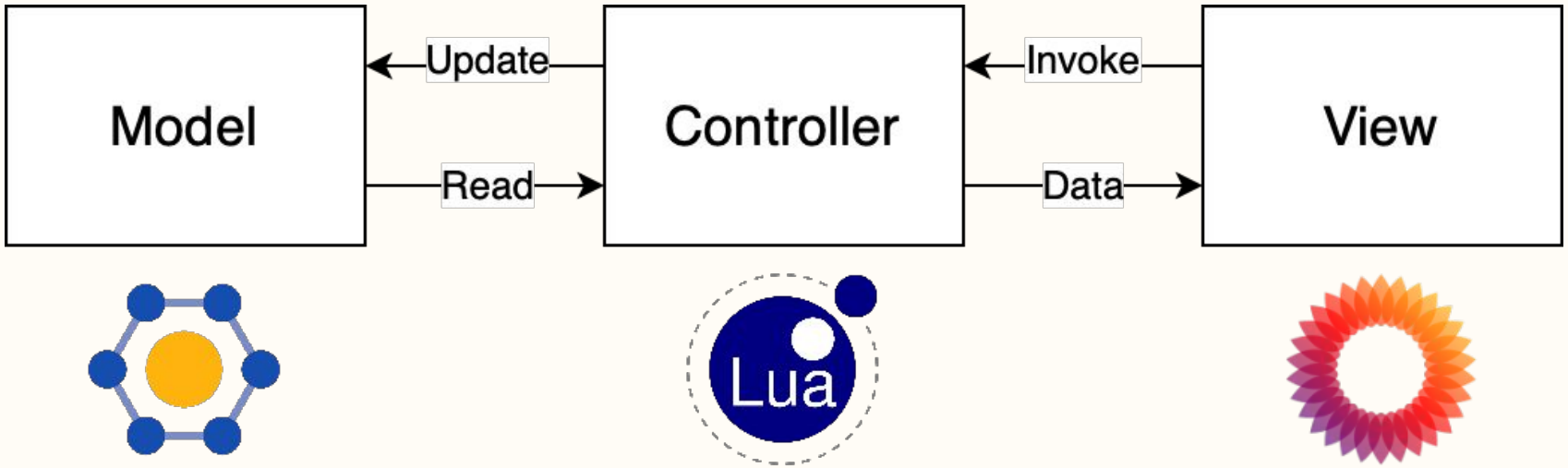
- *Idea*: Parser functions shouldn't have side effects at all: functional parser functions.
- Let parser functions take some arguments, do some computation, and return the result.
- Use function composition to create more complex functions.

ArrayFunctions uses this approach to be a Parsoid-compatible alternative to the Arrays extension:

```
{{#af_join:
  {{#af_map:
    {{#af_list: red | green | blue }}
    | color
    | * {{{color}}} is my favourite.
  }}
  | \n
}}
```

- red is my favourite.
- green is my favourite.
- blue is my favourite.

- But what about the Variables extension?
- *Observation*: Variables are mostly used to store intermediate values during some computation.
- *Idea*: We separate the computation from the wikitext.
  - By using Semantic MediaWiki (or Cargo) only for data storage (model).
  - By using wikitext only for presentation (view).
  - By using Scribunto only for computation (controller).



```
{{#vardefine: nl
| {{#show: The Netherlands | ?Population }}
}}

{{#vardefine: be
| {{#show: Belgium | ?Population }}
}}

{{#vardefine: lu
| {{#show: Luxembourg | ?Population }}
}}

{{#vardefine: bx
| {{#expr: {{#var: nl }} + {{#var: be }} + {{#var: lu }} }}
}}

Populations:
* The Netherlands: {{#var: nl }}
* Belgium: {{#var: be }}
* Luxembourg: {{#var: lu }}
* Benelux: {{#var: bx }}
```

[[Population]]

```
{{PopulationPresentation
| {{#invoke: PopulationController | main }}
}}
```

[[Template:PopulationPresentation]]

```
Populations:
* The Netherlands: {{#af_print: {{#af_get: {{{1}}} | nl }}
}}
* Belgium: {{#af_print: {{#af_get: {{{1}}} | be }} }}
* Luxembourg: {{#af_print: {{#af_get: {{{1}}} | lu }} }}
* Benelux: {{#af_print: {{#af_get: {{{1}}} | bx }} }}
```

[[Module:PopulationController]]

```
local nl = mw.smw.ask(
  "[[The Netherlands]]|?Population=|mainlabel=-"
)[1][1];

local be = mw.smw.ask(
  "[[Belgium]]|?Population=|mainlabel=-"
)[1][1];

local lu = mw.smw.ask(
  "[[Luxembourg]]|?Population=|mainlabel=-"
)[1][1];

return mw.af.export({
  ["nl"] = nl,
  ["be"] = be,
  ["lu"] = lu,
  ["bx"] = nl + be + lu
});
```


Example (cont.)

In short, to make a page Parsoid-compatible:

- Identify which parts are related to computation, and which to presentation.
- Move all computation and queries to a Scribunto module.
  - This module builds an array (table) that contains all *data* used for presenting the page.
- Invoke the module once, and pass the result to a template.
  - This template now has access to all data necessary to present the page.
- Use ArrayFunctions to read and present the data to the user.

Demo

Questions