



URLS & ROUTING

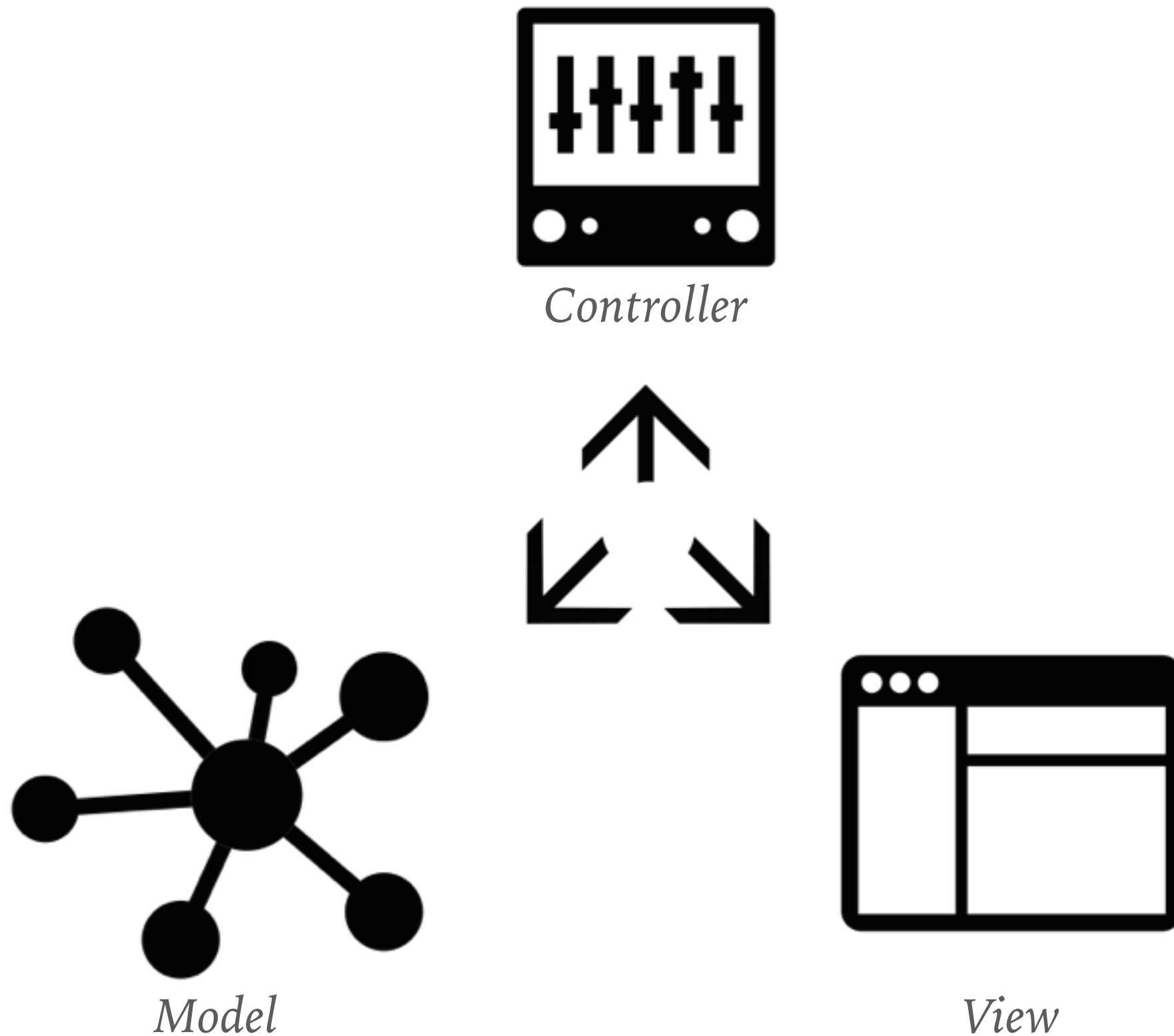
Code 301


CLASS FEEDBACK REVIEW: A PREVIOUS 301

- From a previous 301 (not our PDX-301d3):
 - “Monday...I felt like I wasn't properly equipped to handle the lab assignment.”
 - “I hear everyone bashing on jQuery like it is the drunk inbred cousin of computer programming. He’s invited to all the parties, but everyone makes fun of him! He passes out on the lawn, drooling, right next to PHP.”

ROUTING & CONTROLLERS

MVC FLOW





The controller in a web app is a bit more complicated, **because it has two parts**. The first part is the web server (such as a servlet container) that **maps incoming HTTP URL requests to a particular handler** for that request. The second part is those handlers themselves, which are in fact often called “controllers” [or actions].

So the C in a web app MVC includes both the web server "overlord" that routes requests to handlers and the logic of those handlers themselves, which **pull the data from the database and push it into the template**.

-Terence Parr

ROUTES: YOUR PUBLIC API

- What resources does your app offer?
- Abstract away the details of html files
 - <https://www.codefellows.org/blog.html>
 - <https://www.codefellows.org/blog/>
 - Same page!

ROUTES: YOUR PUBLIC API

- Making files is slow
- We want programmatic control of what our app can do
- GET: /
- GET: /about
- GET: /articles/42
- GET: /articles/42/edit
- PUT: /articles/42

CLIENT-SIDE ROUTING

- Client-side, the route can tell us a few things:
 - What resource the user wants:
 - `hipmunk.com/flights`
 - Additional info:
 - `hipmunk.com/flights#f=SEA;t=HNL;d=2015-12-22`
- JavaScript can interpret this route, and...
 - take apart the pieces
 - call the proper function to handle it all
 - ...all from a single index file: NO RELOAD!

CLIENT-SIDE CONTROLLER

- Our controller will handle the user request
- The controller converts a route into displayed content...
- with the proper data load.
- Simply a list of functions (aka: “actions”), waiting to be called
- One controller per resource:
 - ArticlesController
 - FlightsController
 - UsersController

CLIENT-SIDE ROUTING: HELPFUL LIBRARIES

- `page.js`
 - Connect routes with handling function:
 - `page('/', user.list)`
 - `page('/', index);`
 - `page('/about', about);`
 - `page('/contact', contact);`
 - Many many more examples:
 - <https://github.com/visionmedia/page.js>
 - Install: copy/paste `page.js` file into your project

CLIENT-SIDE ROUTING: HELPFUL LIBRARIES

- `pushstate-server`
 - Sends all requests to: `index.html`
 - Passes through static files:
 - `/scripts/blogArticles.json`
 - `/templates/article.html`
 - Requires full-path URLs, with starting slash
 - More details: <https://github.com/scottcorgan/pushstate-server>
 - Install: `npm install -g pushstate-server`

ROUTING: REVIEWS & DEMOS

- Pushstate API
- Page.js
- miniPage app
- Prep-work

TOASTER DEBUGGING



TOASTER DEBUGGING

- Scour sandbox. DON'T add code. First check each file: name, size, content, link
- Did you create the right filenames? Are all relative paths correct?
- Is server serving the right app? Is browser showing the right app?
- Does editor show the correct files? Did you save *each* file?
- Build in **very** small increments. TEST **every** step. Are you testing often enough?
- Guerrilla [(guh- ril -uh)] warfare: “Wars fought with hit-and-run tactics by small groups against an invader or against an established government.”

LAB READMES

STARTER CODE

RECAP

- URLs power the browser
- JavaScript can leverage the URL to control the app, without going back and forth with a server.