Week 3 Lecture 7

Applied

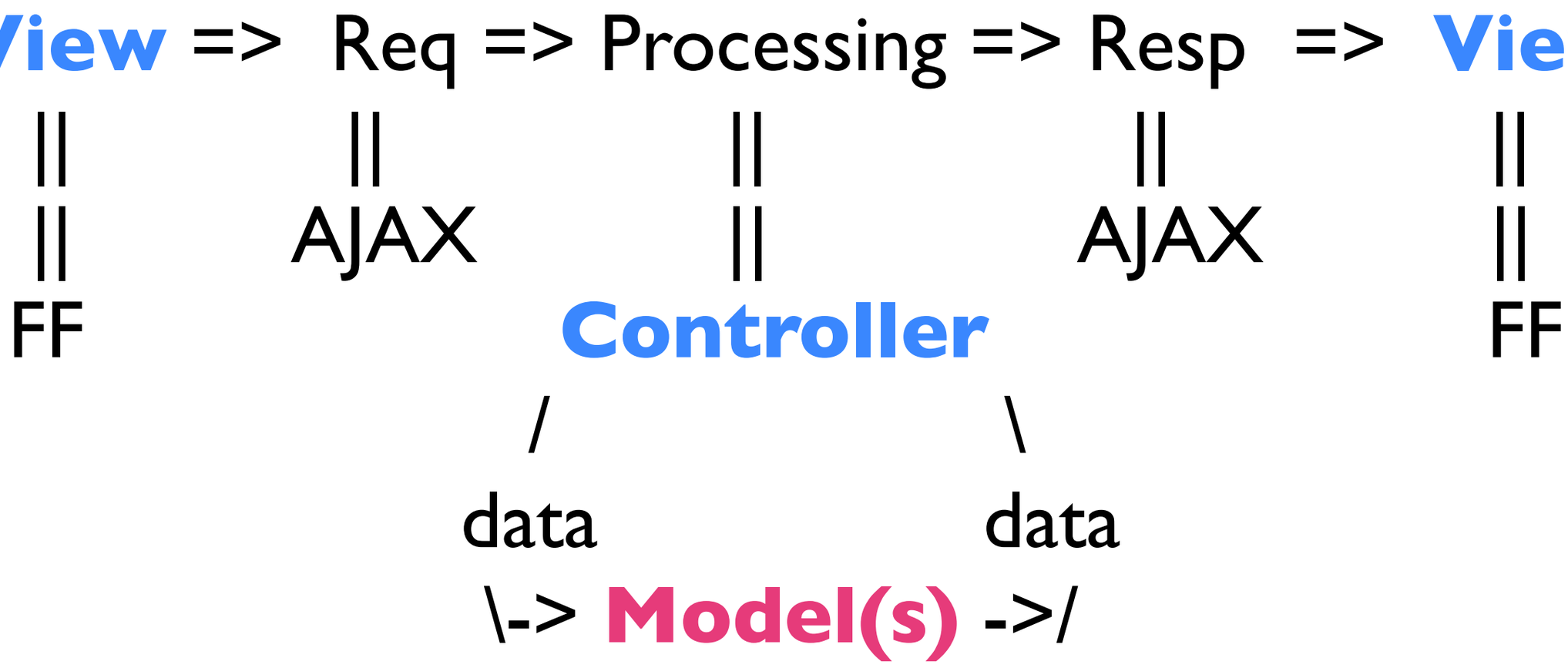
What's in this lecture?

- Introduction to Rails Controllers
- More AJAX!

Quick MVC:

- Data is sent to **Controller**
- **Controller** does ‘stuff’ with passed data
- Typically manipulating a data **Model**
- Passes data to the **View**
- **View** gets rendered on page

MVC Data Flow:



OK. Why?

- Gives us the ability to process input data
- Can now access our persistent data
- Interaction becomes easily customizable
- Centralized! Reusable! Extendable!

Use some analogies!

- Before we had built:
  Sandwich vending machine (just give me what you got)
- After: Full service deli (give me your order, and I'll make a great sub!)

How to get around

- Request is made... but how does the data know where to go?

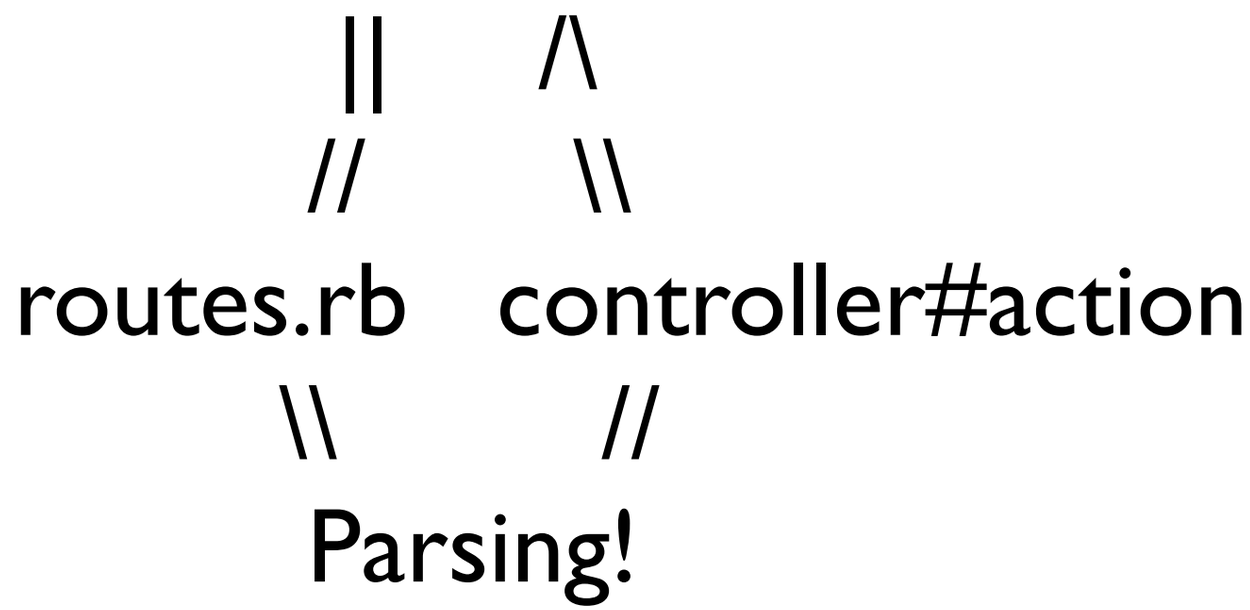
ROUTING!

Rails Routing Explained

- initiate: we tell it!
- padawan: rules map paths to actions
- knight: path resolution is actually regex parsing, so we write custom regexes using a provided DSL that map paths to actions
- master: tell it, we do

Routing Diagram

- URL => REQ => Processing => RESP => VIEW



Routing Example

- match 'person/foo' => 'person#foo'
- “Match the path ‘person/foo(.format)’ to the controller person and the action foo”

Controllers

- Controllers are just special files
- Define actions which operate based on input data
- How come we don't let the model do this?
- Answer: Models contain 'business logic', controllers contain 'application logic'

Controller: Example

- Model would be a bank account
- Controller would be bank teller

- Bank teller takes input from you, and knowing a specific set of steps, can perform debits, credits, and all transactions.

Controller Layout

```
class PersonController < ApplicationController

  def foo # the action
  end

end
```

Driving it home

- A request is made to a specific path
- Path is resolved (through a regex) to a controller and action
- Code within *only* the specified action is called and executed
- Data is sent to the view and back as resp

Goal: rewrite ‘Ajaxy’ using controller actions

Ajaxy: Routing

```
Ajaxy::Application.routes.draw do

  match 'person/foo' => 'person#foo'

end
```

Ajaxy: Controller Logic

```
class PersonController < ApplicationController

  def foo # the action
    person = {}
    person[:name] = "foo"
    person[:message] = "this is foo from the controller!"
    render :json => person
  end

end
```

Ajaxy: Client-side 1

```
function do_it() {
  var the_file = $("#a_select")[0].value;
if (the_file == '-') {
      $("#the_message")[0].innerHTML = "<h1>Initial!</h1><p>Select a value</p>";
  } else {
  $.getJSON("http://localhost:3000/person/" + the_file,
    function(data) {
      $("#the_message")[0].innerHTML = "<h1>" + data.name + "</h1><p>" + data.message + "</p>";
    });
}
}
```

Ajaxy: Client-side II

```
<form onsubmit="return false;">
<select id="a_select" onchange="do_it()">
  <option value="-">-</option>
   <option value="foo.json">foo</option>
 </select>
</form>
```

Exercise

- Update Ajaxy so ‘bar’ and ‘baz’ respond to controller actions