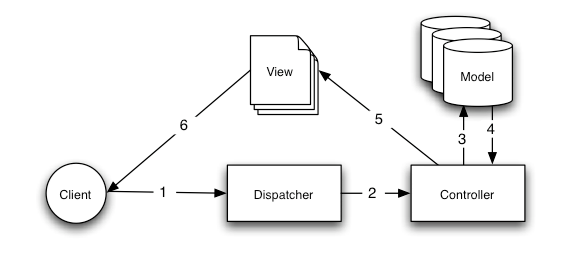
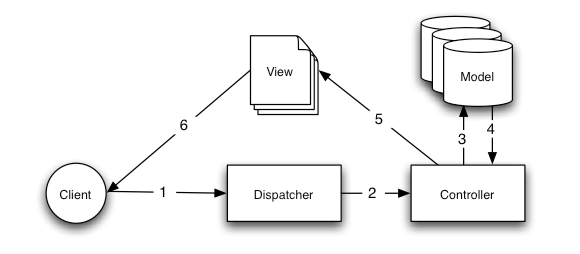
* **Static webpages**: content, presentation, interaction
* **Dynamic** **webpages**: has static webpage components, content changes on user interaction
  + database, logic, HTML/CSS/JavaScript
* **Ruby** is the language, **Rails** is the framework
  + ex) Twitter: users
    - user info => database => interacted with using models
    - following => logic
    - content => views
* **MVC**: Model View Control
* 
* **Goals**
* 1. Know general web-app structure
  + 2. Use Ruby on Rails to make web apps
* **Logistics**
  + Syllabus: bit.ly/web-syl
  + OH: bit.ly/dec-oh
* **HTTP Requests**
  + Client makes a request to Server
  + The four HTTP request we’re going to focus on: GET, POST, PATCH, DELETE
  + Today: GET
* **routes.rb**
  + Our Dispatcher in the MVC program
  + Found in config folder
  + Takes request from the client and figures out which controller function to call
* **Routes Syntax**
  + get ‘/name’, to: ‘controller#function’
  + To make a page a root directory: root ‘controller#function’
* **Contollers**
  + Takes dispatch from routes.rb
  + Gets info from the database and models
  + Performs logic needed before rendering view
  + For static pages we can simply have an empty function
* **Views**
  + Presents the content of the webpage
  + HTML, CSS
  + Embedded Ruby
    - <% ruby code %>
    - <%= ruby code and substitution %>
  + Finds resources in app/assets folder
* **Linking within Pages**
  + get ‘/path’, to: ‘controller#function’, as: ‘{path\_name}’
  + Creates a path which can be called by using {path\_name}\_path
  + Path can be used to link pages in the ERB of the View using link\_to
  + Root path canb e referenced by root\_path
* **Models in Rails**
  + Built-in class frameworks
    - Methods you don’t need to define
    - Interact with the database
    - class <classname> < ActiveRecord::Base
  + Why are they important?
    - Core of Rails apps: represents all info you need
      * Allows controllers to work with the model
  + What can you do with models?
    - Ex) Twitter: create, read, (show) update (edit), destroy
    - “CRUD”
    - Do this using controllers!



\* Focusing on Model/Controller interaction (3, 4)

**Databases**

* + Hold data
  + Don’t need to know much
  + Migrations are what keep models and the database in sync. Unfortunately, we must run these ourselves
* **Today: CREATE, READ, UPDATE**
* **Create**
  + Usually done through forms
  + Pass in values like:
    - Name: Isaiah
    - Age: 19
  + Create model instance, then add to database
* **Read (Show/Index)**
  + In the view controller can specify which model objects to show
  + Controller selects correct ones

**Update (Update/Edit)**

* + Also usually done through forms
  + Change the name (ex)
  + Controller updates model instance (database row)
* **Hands-On**
* $ rails generate model User age:integer name:string   
  $ rake db:migrate  
    
  $ rails c  
    
  # Create users!  
  > wonjun = User.new   
  > wonjun.age = 10  
  > [wonjun.name](http://wonjun.name/" \t "_blank) = "wonjun"  
  > wonjun.save  
    
  # See all the users!  
  > User.all  
  > quit  
    
  # Create controller!  
  $ rails generate controller Users index show edit  
    
  # in app/controllers/users\_controller.rb  
    
  def index  
   @users = User.all  
  end  
    
  # in app/views/users/index.html.erb  
    
  <% @users.each do |user| %>  
   <h1> <%= [user.name](http://user.name/" \t "_blank) %> <%= user.age %> </h1>  
   <br>  
  <% end %>  
    
  # Now examine magic  
  $ rails s  
  $ Now in browser go to localhost:3000/users  
    
  You should see "wonjun 10"

# To update fields

rails generate migration RemoveFieldFromTable field\_name:type

rake db:migrate

Then, add

* **CRUD:** Create, Read, Update, Destroy
* Rails create book\_library
* Rails generate model Book title:string description:text
* Rake db:migrate
* Rails generate controller books
* 1. Browser sends GET request
* 2. Rails server receives request
* 3. routes.rb specifies which controller action
* 4. Controller action renders the view
* **Gems**: libraries written for the Ruby language
  + Gems: Ruby, Modules: Python, Packages: JS
* Have unique names and versions
* Include:
  + Code (w/ tests)
  + Documentation
  + Gemspec (info about a gem)
* Gem for everything! Email, payments, easier developing, etc.
* Rails is: a gem!
* **Bundler**: provides consistent environment for Ruby projects by tracking and installing exact ems and versions needed
  + Without bundler: gem install pg
  + With:
    - All dependencies stored in Gemfile
    - bundle install installs all
    - Dependencies listed in Gemfile
    - bundler is a gem
* **Gemfile**
  + Require at least one gem sources, in the form of the URL for a Rubygems server (rubygems.org)
  + Git repositories also valid, as long as repo contains one or more valid gems
  + Dependencies can also be placed in groups
  + Gemfile.lock
    - List of all gem names and versions in a file
    - Ensures that same dependencies are called across diff environments
    - Think of it as a snapshot of all the gems and their versions
* **RVM:** tool that allows you to install multiple versions of Ruby on a machine, allows switching between them
  + Has gemsets: set of gems specific to given project
    - Why use gemsets?
      * Faster loading, because less gem-specs need to be parsed
      * Easy to distribute/share
* **Forms**: validating forms – ensures that only valid data saved onto database
  + Where to validate?
    - Model-level validations
    - Client-side validations
    - Controller-level validations
* **Error Messages**
  + Flash
    - Special part of the session that is cleared with each request
    - Represented as a hash
    - Useful for passing error messages