web frameworks

Daniel Jackson
basic server side architecture

Apache → Rails → MySQL

request

response
what a framework does

basic areas of built-in functionality
› routing
› request data
› data mapping
› templates
› validation
› session state
routing

separation of concerns
› client’s name for op vs. developer’s name for method

example
› request       GET /photos/17
› routes.rb     match 'photos/:id' => 'photos#show'
› call          class PhotosController ...
                 def show
                 ... params[:id] ...
                 end

for more, see [http://guides.rubyonrails.org/routing.html](http://guides.rubyonrails.org/routing.html)
› bidirectional
› RESTful default routing
request data

two kinds of request data
› query strings from GETs
› form data from POSTs

read in the same way
› params[:field]

class ClientsController < ActionController::Base
  # sample URL: /clients?status=activated
  def index
    if params[:status] == "activated"
      @clients = Client.activated
    else
      @clients = Client.unactivated
    end
  end
web app = data conversion

update request

tipster: share tips with friends

update request

request.form

id    2
rating 5
content yummy!

database table

response

<table>
<thead>
<tr>
<th>id</th>
<th>by</th>
<th>content</th>
<th>rating</th>
<th>about</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>yummy!</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>neat</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

response

<table>
<thead>
<tr>
<th>id</th>
<th>rating</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>yummy!</td>
</tr>
</tbody>
</table>
data mapping

basic idea
› object in heap = row in table

object relational mapper
› generates database schema from class defs
› backs up object methods with SQL updates/queries
Rails ActiveRecord

class Client < ActiveRecord::Base
    has_one :address
    has_many :orders
    has_and_belongs_to_many :roles
end

class Address < ActiveRecord::Base
    belongs_to :client
end

address = client.address

# Find the client with primary key (id) 10
client = Client.find(10)

SELECT * FROM clients WHERE (clients.id = 10) LIMIT 1
**Listing Books**

<table>
<thead>
<tr>
<th>Title</th>
<th>Summary</th>
<th>Show</th>
<th>Edit</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;%= book.title %&gt;</td>
<td>&lt;%= book.content %&gt;</td>
<td>&lt;%= link_to 'Show', book %&gt;</td>
<td>&lt;%= link_to 'Edit', edit_book_path(book) %&gt;</td>
<td>&lt;%= link_to 'Remove', book, :confirm =&gt; 'Are you sure?', :method =&gt; :delete %&gt;</td>
</tr>
</tbody>
</table>

Also
- layouts, partials, flashing
validation

built in functions to check

```ruby
class Person < ActiveRecord::Base
  validates :name, :length => { :minimum => 2 }
  validates :bio, :length => { :maximum => 500 }
  validates :password, :length => { :in => 6..20 }
  validates :registration_number, :length => { :is => 6 }
end
```

messages to be displayed in flash

```ruby
class Coffee < ActiveRecord::Base
  validates :size, :inclusion => { :in => %w(small medium large), :message => '%{value} is not a valid size' }
end
```
session state

session variable holds session state

```ruby
session[:user_id] = @current_user.id
User.find(session[:user_id])
```

where’s session state stored?
› in a (hidden) database table; cookie just holds id
› or entirely in cookie

what’s the tradeoff?