CS 683 Emerging Technologies
Fall Semester, 2006
Doc 22 Rails 6 URLS, Forms, AJAX
Nov 9, 2006

Copyright ©, All rights reserved. 2006 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

References

Agile Web Development with Rails 2nd Ed Bl.16 October 25, Thomas & Hanson, The Pragmatic Bookshelf, PDF

Rails API, http://api.rubyonrails.org/

Url Mapping

config/routes.rb

Contains mapping Rules

First matching rules applies

:controller => which controller to call
:action => which method to call in controller
:id => value in params

```
ActionController::Routing::Routes.draw do |map|
  map.connect ':controller/service.wsdl', :action => 'wsdl'

  map.connect ':controller/:action/:id'
end
```

Default Example

rule: map.connect ':controller/:action/:id'

Request: http://127.0.0.1:3000/books/list/

:controller = books
:action = list
:id not defined

Request: http://127.0.0.1:3000/books/list/a

:controller = books
:action = list
:id = a

Request: http://127.0.0.1:3000/books/list/a/b

No matching rule

New Rule

```
map.connect   'store/index.html',
              :controller => 'books',
              :action => 'list'
```

http://127.0.0.1:3000/store/index.html

calls list method in BooksController

Does not work

```
map.connect   'index.html',
              :controller => 'books',
              :action => 'list'
```

http://127.0.0.1:3000/index.html

results in Rails Welcome page

How do we turn this off?

Default Values

```
map.connect 'cat/:a/:b/:c',
            :controller => 'foo',
            :action => 'bar',
            :a => 'at',
            :b => 'bat',
            :c => 'cat'
```

```
class FooController < ActionController::Base
def bar
    puts 'made it'
    a = @params[:a]
    b = @params[:b]
    c = @params[:c]
    render :text => "a = #{a}<br/>b = #{b}<br/>c = #{c}"
end
```

Extra text in URL

```
map.connect 'store/:controller/buy/:id',
      :action => 'buy'

http://0.0.0.0:3000/store/computer/buy/xbox

@param = {:controller => 'computer', :id => 'xbox'
```

Specifying Formats

```
map.connect 'whitney/:year/:month/:day',
        :controller => 'blog',
        :action    => "show_date",
        :requirements      => {:year => /(19|20)\d\d/,
                               :month => /[01]?\d/,
                               :day => /[0-3]?\d/},
        :day               => nil
```

http://0.0.0.0:3000/whitney/2005/01/30

```
@param = {:controller => 'blog', :action => 'show_date',
            :year            => '2005',
            :month   => '01',
            :day       => '30'}
```

map.connect ':controller/:action/:id'

Is it a good idea to let end users call methods directly?

Forms

Forms on Models

migration file

```
class CreateBooks < ActiveRecord::Migration
  def self.up
    create_table :books, :force => true do |t|
      t.column :title, :string
      t.column :author, :string
      t.column :date, :date
    end

    def self.down
      drop_table :books
    end
  end
```

Controller

books_controller.rb

```
class BooksController < ApplicationController
  def sampleForm
    @book = Book.new

    if request.post?
        newBook = Book.new(params[:book])
        if newBook.save
            #code to handle success
    else
        #code to handle failure
    end
end
```

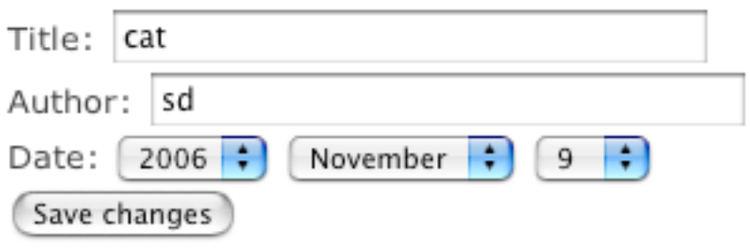
Form

views/books/sampleForm.rhtml

```
<% form_for :book do |form| %>
    Title: <%= form.text_field :title, :size => 30 %> <br/>
    Author: <%= form.text_field :author, :size => 30 %> <br/>
    Date: <%= form.date_select :date, :start_year => 1960 %> <br/>
     <%= submit_tag %>
<% end %>
```

Hidden assumption:

instance variable name = class name



Explicit Variable Name

```
class BooksController < ApplicationController
  def sampleForm
    @foo = Book.new
   end
```

```
<% form_for :book, @foo do |form| %>
      Title: <%= form.text_field :title, :size => 30 %> <br/>
      Author: <%= form.text_field :author, :size => 30 %> <br/>
      Date: <%= form.date_select :date, :start_year => 1960 %> <br/>
       <%= submit_tag %>
<% end %>
```

Specifying post method

```
<% form_for :book, @foo, :url => {:action => :create } do |form| %>
    Title: <%= form.text_field :title, :size => 30 %> <br/>
    Author: <%= form.text_field :author, :size => 30 %> <br/>
    Date: <%= form.date_select :date, :start_year => 1960 %> <br/>
     <%= submit_tag %>
<% end %>
```

Some Form Helpers

- check_box
- file_field
- hidden_field
- password_field
- radio_button
- text_area
- text_field

- collection_select
- country_options_for_select
- country_select
- option_groups_from_collection_for_select
- options_for_select
- options_from_collection_for_select
- select
- time_zone_options_for_select
- time_zone_select

http://api.rubyonrails.com/classes/ActionView/Helpers/FormHelper.html

http://api.rubyonrails.com/classes/ActionView/Helpers/FormOptionsHelper.html

http://api.rubyonrails.com/classes/ActionView/Helpers/DateHelper.html

Multiple Models in one Form

Publishers Table

```
create_table :publishers do |t|
  t.column :name, :string
```

Books Table

```
create_table :books, :force => true do |t|
  t.column :title, :string
  t.column :author, :string
  t.column :date, :date
  t.column :publisher_id, integer
```

Controller

```
class BooksController < ApplicationController
  def sampleForm
    @foo = Book.new
    @publisher = Publisher.new
    if request.post?
      newBook = Book.new(params[:book])
      publisher = Publisher.new(params[:publisher])
          # code to handle post here
  end

end
```

sampleForm.rhtml

```
<% form_for :book, @foo do |form| %>
    Title: <%= form.text_field :title, :size => 30 %> <br/>
    Author: <%= form.text_field :author, :size => 30 %> <br/>
    Date: <%= form.date_select :date, :start_year => 1960 %> <br/>

    <% fields_for :publisher do |pub| %>
        Publisher: <%= pub.text_field :name %><br/>
    <% end %>

        <%= submit_tag %>
<% end %>
```

Of course one really does not want to create a new publisher with each book

Showing the List

```
class BooksController < ApplicationController
  def sampleForm
    @foo = Book.new
    @publishers = Publisher.find(:all).collect {|p| [p.name, p.id]}
    if request.post?
        newBook = Book.new(params[:book])

        if newBook.save
            puts 'saved'
        else
            puts 'not saved'
        end
  end

end
```

The Form

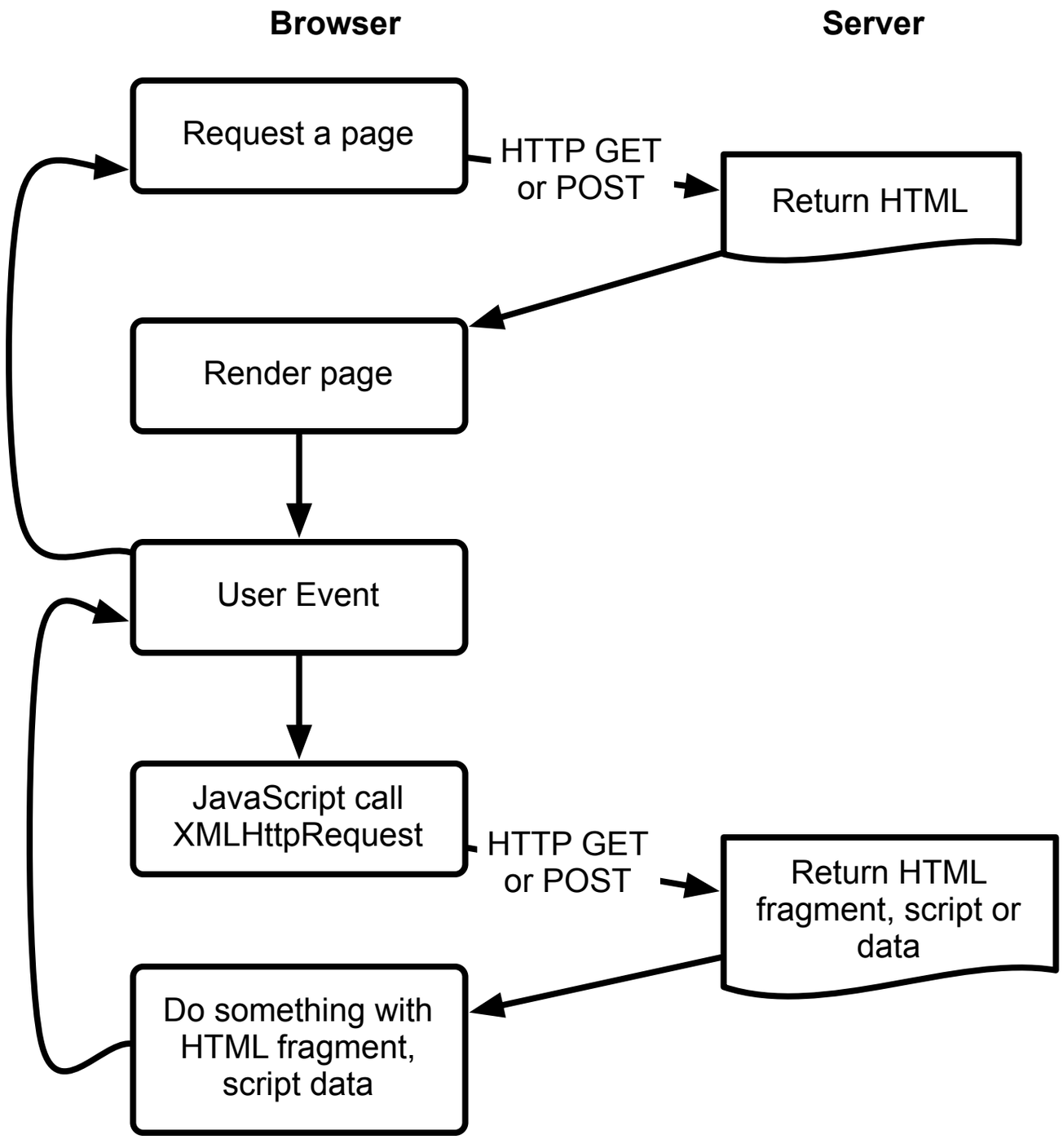
```
<% form_for :book, @foo do |form| %>
    Title: <%= form.text_field :title, :size => 30 %> <br/>
    Author: <%= form.text_field :author, :size => 30 %> <br/>
    Date: <%= form.date_select :date, :start_year => 1990 %> <br/>
    Publisher: <%= form.select :publisher_id, @publishers %> <br/>

     <%= submit_tag %>
<% end %>
```

http://api.rubyonrails.com/classes/ActionView/Helpers/FormOptionsHelper.html#M000399

Ajax Intro

Ajax & Web 2.0

AJAX - Asynchronous JavaScript and XML



Some Web 2.0 sites

Google maps/local http://maps.google.com/

Housing Maps - Graiglist using Google maps, http://www.housingmaps.com/

flickr, photos & slide shows, http://www.flickr.com/

37 signals http://www.37signals.com/

- Basecamp - project management, http://www.basecamphq.com/
- Backpack - organize information, http://www.backpackit.com/
- Writeboard - collaborative writing, http://www.writeboard.com/
- Ta-da List - to-do lists, http://www.tadalist.com/

Jakob Nielsen on Ajax

For new or inexperienced Web designers -
Just say no to Ajax

Web page

User's view of information
Unit of navigation
Textual address used to retrieve information
Unit of information on server

Ajax

Breaks this model of a web page
Back button does not work
Printing problems
Authoring problems
Search Problems

First Example

app/controllers/ajax_controller

```
class AjaxController < ApplicationController
    def time_now
      render :layout => false
    end
    def remoteLink

    end
end
```

Views

app/views/layout/ajax.rhtml

```
<html>
<head>
  <title>Ajax Examples</title>
  <%= javascript_include_tag "prototype" %>
</head>
<body>
        <%= @content_for_layout %>
</body>
</html>
```

app/views/ajax/remoteLink.rhtml

```
    <%= link_to_remote( "Click on me",
      :update => 'changeDiv',
      :url => { :action => :time_now}
        )%>
    <div id="changeDiv">Hi mom</div>
```

app/views/ajax/time_now.rhtml

```
Hello Ajax
<p>The time is now <%= Time.now%>
</p>
<p>Session ID is <%= session.session_id %>
</p>
```

Screen Shots

Click on me
Hi mom

Click on me
Hello Ajax

The time is now Thu Nov 09 16:02:02 PST 2006

Session ID is f3c3847b5d45679eb8ac268ecb6c0823