Controller Naming Convention

Methods and Actions

Parameters

Hash and Array Parameters

GET /clients?ids[]=1&ids[]=2&ids[]=3

JSON parameters

Routing Parameters

default_url_options

Strong Parameters

Permitted Scalar Values (permit!)

Nested Parameters

```
params.permit(:name, { emails: [] },
              friends: [ :name,
                         { family: [ :name ], hobbies: [] }])
```

Fetch Params

```
params.fetch(:blog, {}).permit(:title, :author)
```

Session

```
ActionDispatch::Session::CookieStore - Stores everything on the client.
ActionDispatch::Session::CacheStore - Stores the data in the Rails cache.
ActionDispatch::Session::ActiveRecordStore - Stores the data in a database
using Active Record. (require activerecord-session_store gem).
ActionDispatch::Session::MemCacheStore - Stores the data in a memcached
cluster (this is a legacy implementation; consider using CacheStore
instead).
```

Accessing the Session

Sessions are lazily loaded reset_session

The Flash

```
flash[:notice] = "You have successfully logged out."

redirect_to root_url, notice: "You have successfully logged out."
redirect_to root_url, alert: "You're stuck here!"
redirect_to root_url, flash: { referral_code: 1234 }

 <% flash.each do |name, msg| -%>
   <%= content_tag :div, msg, class: name %>
 <% end -%>

flash.now[:error] = "Could not save client"
```

Cookies

```
        cookies[:commenter_name] = @comment.author
```

Rendering xml and json data

```
respond_to do |format|
      format.html # index.html.erb
      format.xml  { render xml: @users}
      format.json { render json: @users}
end
```

Filter

```
class ApplicationController < ActionController::Base
  before_action :require_login

  private

  def require_login
    unless logged_in?
      flash[:error] = "You must be logged in to access this section"
      redirect_to new_login_url # halts request cycle
    end
  end

  skip_before_action :require_login, only: [:new, :create]
```

After Filters and Around Filters

```
class ChangesController < ApplicationController
  around_action :wrap_in_transaction, only: :show

  private

  def wrap_in_transaction
    ActiveRecord::Base.transaction do
      begin
        yield
      ensure
        raise ActiveRecord::Rollback
      end
    end
  end
end

class ApplicationController < ActionController::Base
  before_action do |controller|
    redirect_to new_login_url unless controller.send(:logged_in?)
  end
end
```

Request Forgery Protection

The Request and Response Objects

Headers

```
response.headers["Content-Type"] = "application/pdf"
```

HTTP Basic Authentication

```
class AdminsController < ApplicationController
  http_basic_authenticate_with name: "humbaba", password: "5baa61e4"
end
```

HTTP Digest Authentication

```
class AdminsController < ApplicationController
  USERS = { "lifo" => "world" }

  before_action :authenticate

  private

  def authenticate
    authenticate_or_request_with_http_digest do |username|
      USERS[username]
    end
  end
end
```

Streaming and File Downloads

```
send_data generate_pdf(client),
              filename: "#{client.name}.pdf",
              type: "application/pdf"
```

Send file

```
send_file("#{Rails.root}/files/clients/#{client.id}.pdf",
              filename: "#{client.name}.pdf",
              type: "application/pdf")
```

Parameters Filtering

config.filter_parameters << :password

Redirects Filtering

config.filter_redirect << 's3.amazonaws.com' or regexp

The Default 500 and 404 Templates

404.html and 500.html

rescue_from

```
class ApplicationController < ActionController::Base
  rescue_from ActiveRecord::RecordNotFound, with: :record_not_found

  private

  def record_not_found
    render text: "404 Not Found", status: 404
  end
end
```

Force HTTPS protocol

```
class DinnerController
  force_ssl only: :cheeseburger
  # or
  force_ssl except: :cheeseburger
end
```