CS 142 Section 4

Forms and Sessions

Overview

1. Forms
2. Validation
3. Uploading
4. Virtual Attributes
5. Session
6. Flash
7. Hashing
8. Filters

Useful resources

- http://api.rubyonrails.org/classes/ActionView/Helpers/FormHelper.html
- http://guides.rubyonrails.org/active_record_validations_callbacks.html

form_tag

```
<%= form_tag("post_submit", :method => "post") do %>
    <%= label_tag(:text_val, "Enter something here:") %>
    <%= text_field_tag(:text_val) %>
    <%= submit_tag("Submit") %>
<% end %>
```

- Useful for simple forms not related to a specific model object (i.e. search bar, login, etc.). Access value in controller with params[:text_val].
- Use form_for if you want to work with a specific model, or want to use a model's validations (i.e. creating new user, modifying existing information)
- Many other tags for use in form_tag such as: check_box_tag, email_field_tag, file_field_tag, hidden_field_tag, password_field_tag, text_area_tag, radio_button_tag

Review: form_for

Name of model class (Student)

Name of variable containing data (@student)

```
<% form_for(:student, :url => {:action => :modify,
    :id =>@student.id}) do |form| %>
  <table class="form">
    <tr>
      <td class="label">Name:<td>
      <td><%= form.text_field(:name) %><td>
    </tr>
    <tr>
      <td class="label">Date of birth:<td>
      <td><%= form.text_field(:birth) %><td>
    </tr>
    ...
  <table>
  <%= submit_tag "Modify Student" %>
<% end %>
```

Initial value will be @student.name

Text to display in submit button

```
<input id="student_name" name="student[name]"
    size="30" type="text" value="Hernandez" />
```

Review: Post Action Method

Hash with all of form data

```
def modify
  @student = Student.find(params[:id])
  if @student.update_attributes(params[:student]) then
    redirect_to(:action => :show)
  else
    render(:action => :edit)
  end
end
```

Redirects on success

- update_attributes() modifies values and saves into database if valid
- Manually save into database using save()
- To access individual elements of params[:students], just access it as a nested hash (i.e. params[:students][:birth])

Differences between symbol and instance variable in form

By default, if you use the symbol :post, you get:

<form action="/posts" method="post">

if you use the instance @post where @post = Post.new:

<form action="/posts/create" class="new_account" id="new_account" method="post">

if @post = Post.find(1) you will get:

<form action="/posts/update" class="edit_account" id="edit_account_1" method="post">
<input name="_method" type="hidden" value="put">

Review: Validation

Custom validation method

```
class Student < ActiveRecord::Base
  def validate
    if (gpa < 0) || (gpa > 4.0) then
      errors.add(:gpa, "must be between 0.0 and 4.0")
    end
  end

  validates_format_of :birth,
    :with => /\d\d\d\d-\d\d-\d\d/,
    :message => "must have format YYYY-MM-DD"
end
```

Saves error info

Built-in validator

Review: Error Message Helper

```
<% @student.errors.full_messages.each do |msg| %>
   <p><%= msg %></p>
<% end %>
<
<% form_for(:student, :url => {:action => :modify,
    :id =>@student.id}) do |form| %>
  ...
  <%= form.text_field(:name) %>
  ...
<% end %>
```

.valid? method

```
class Person < ActiveRecord::Base
  validates :name, :presence => true
end

Person.create(:name => "John Doe").valid? # => true
Person.create(:name => nil).valid? # => false
```

Review: File Uploads with Rails

```
<% form_for(:student, :html=>{:multipart => true}
        :url => {...}) do |form| %>
  ...
  <%= form.file_field(:photo) %>
  ...
<% end %>
```

In form post method:

```
params[:student][:photo].read()
params[:student][:photo].original_filename
```

Virtual Attributes

What if your Model attributes are different than your form fields? Define virtual attributes with getters/setters.

```
class User < ActiveRecord::Base
    def full_name
        "#{first_name} #{last_name}"
    end

    def full_name=(name)
        first_name, last_name = name.split()
    end
end
```

Session

Session is a hash that is accessible during all requests from a particular browser. Session data is not meant to be permanent.

Use reset_session to clear this data and start a new session.

Session example

```
def controller_method
    ...
    session["foo"] = bar
    ...
end

def another_method
    ...
    reset_session
    ...
end
```

Flash

If you need to display a message to the user in the *next* page they view, use flash. This is useful if you're redirecting to a new page.

Flash Example

```
def logout
    redirect_to url, :notice => "You have been
logged out"
end
...
<% if flash[:notice] %>
  <div><%= flash[:notice] %>
<% end %>
```

Password Hashing

Storing passwords in plain text in your database is insecure. What if bad guys break into your server and steal your users database? Now they have the passwords of all your users, many of whom use the same password on every site

Solution: Use password hashing to store hashes, instead of passwords

Password Hashing

Hash a string using Digest::SHA1

Digest::SHA1.hexdigest("hello") =>
"aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d"

Filters

```
class MyController < ActionController::Base
    before_filter :check_params

    def check_params
        if params[:foo] == "bar"
            redirect_to "url", :flash => "message"
        end
    end
end
```