ActionView Helpers

Sang Shin
Michèle Garoche
www.javapassion.com
“Learn with Passion!”

Topics

- Types of ActionView Helpers
- FormHelper's
- FormTagHelper's

What Are ActionView Helpers?

Types of ActionView Helpers

- FormHelper's
  - Specifically work with model attributes
- FormTagHelper's
  - Do not work with model attributes

FormHelper's

What Are FormHelpers For?

- Form helpers are designed to make working with models much easier
  - by providing a set of methods for creating forms based on your models.
- These helpers generate HTML for forms, providing a method for each type of input (e.g., text, password, select, and so on).
- When the form is submitted (i.e., when the user hits the submit button or form.submit is called via JavaScript), the form inputs will be bundled into the *params* object and passed back to the controller.

FormHelpers

- form_for
- text_field, text_area
- password_field
- hidden_field
- check_box
- radio_button
- fields_for
- file_field
- label

form_for Helper

- *form_for(record_or_name_or_array, *args, &proc)*
- The *form_for* gives you the ability to create a form for a model instance

form_for Helper Example

- You have a model *Person* and want to create a new instance of it

```
# Note: a @person variable will have been created in the controller.
# For example: @person = Person.new
<% form_for :person, @person, :url => { :action => "create" } do |f| %>
  <%= f.text_field :first_name %>
  <%= f.text_field :last_name %>
  <%= submit_tag 'Create' %>
<% end %>
```

- *The HTML generated for this would be:*

```
<form action="/persons/create" method="post">
  <input id="person_first_name" name="person[first_name]" size="30"
type="text" />
  <input id="person_last_name" name="person[last_name]" size="30"
type="text" />
  <input name="commit" type="submit" value="Create" />
</form>
```

text_field Helper

- *text_field(object_name, method, options = {})*
- Returns an input tag of the "text" type tailored for accessing a specified attribute (identified by method) on an object assigned to the template (identified by object).
- Additional options on the input tag can be passed as a hash with options.
  - These options will be tagged onto the HTML as an HTML element attribute.

text_field Helper Examples

```
text_field(:post, :title, :size => 20)
# => <input type="text" id="post_title" name="post[title]"
  value="#{@post.title}"  size="20" />

text_field(:post, :title, :class => "create_input")
# => <input type="text" id="post_title" name="post[title]"
  value="#{@post.title}" class="create_input" />

text_field(:session, :user, :onchange => "if $('session[user]').value == 'admin'
  { alert('Your login can not be admin!'); }")
# => <input type="text" id="session_user" name="session[user]"
  value="#{@session.user}" onchange = "if $('session[user]').value == 'admin'
  { alert('Your login can not be admin!'); }"/>

text_field(:snippet, :code, :size => 20, :class => 'code_input')
# => <input type="text" id="snippet_code" name="snippet[code]" size="20"
  value="#{@snippet.code}" class="code_input" />
```

text_area Helper

- *text_area(object_name, method, options = {})*
- Returns a textarea opening and closing tag set tailored for accessing a specified attribute (identified by method) on an object assigned to the template (identified by object).
- Additional options on the input tag can be passed as a hash with options.

text_area Helper Examples

```
text_area(:post, :body, :cols => 20, :rows => 40)
# => <textarea cols="20" rows="40" id="post_body" name="post[body]">
#      #{@post.body}
#    </textarea>

text_area(:comment, :text, :size => "20x30")
# => <textarea cols="20" rows="30" id="comment_text"
  name="comment[text]">
#      #{@comment.text}
#    </textarea>
```

text_area Helper Examples (Cont)

```
text_area(:application, :notes, :cols => 40, :rows => 15, :class => 'app_input')
# => <textarea cols="40" rows="15" id="application_notes"
  name="application[notes]" class="app_input">
#      #{@application.notes}
#    </textarea>

text_area(:entry, :body, :size => "20x20", :disabled => 'disabled')
# => <textarea cols="20" rows="20" id="entry_body" name="entry[body]"
  disabled="disabled">
#      #{@entry.body}
#    </textarea>
```

check_box Helper

- *check_box(object_name, method, options = {}, checked_value = "1", unchecked_value = "0")*
- Returns a checkbox tag tailored for accessing a specified attribute (identified by method) on an object assigned to the template (identified by object).
- It‘s intended that method returns an integer and if that integer is above zero, then the checkbox is checked.
- The checked_value defaults to 1 while the default unchecked_value is set to 0 which is convenient for boolean values. Since HTTP standards say that unchecked checkboxes don‘t post anything, we add a hidden value with the same name as the checkbox as a work around.

check_box Helper Examples

```
# Let's say that @post.validated? is 1:

check_box("post", "validated")
# => <input type="checkbox" id="post_validated"
  name="post[validated]" value="1" />
#    <input name="post[validated]" type="hidden" value="0" />

# Let's say that @puppy.gooddog is "no":
check_box("puppy", "gooddog", {}, "yes", "no")
# => <input type="checkbox" id="puppy_gooddog"
  name="puppy[gooddog]" value="yes" />
#    <input name="puppy[gooddog]" type="hidden" value="no" />

check_box("eula", "accepted", { :class => 'eula_check' }, "yes", "no")
# => <input type="checkbox" class="eula_check" id="eula_accepted"
  name="eula[accepted]" value="yes" />
#    <input name="eula[accepted]" type="hidden" value="no" />
```

radio_button Helper

- *radio_button(object_name, method, tag_value, options = {})*
- Returns a radio button tag for accessing a specified attribute (identified by method) on an object assigned to the template (identified by object).
- If the current value of method is tag_value the radio button will be checked.
- Additional options on the input tag can be passed as a hash with options.

radio_button Helper Examples

```
# Let's say that @post.category returns "rails":
radio_button("post", "category", "rails")
radio_button("post", "category", "java")
# => <input type="radio" id="post_category_rails"
 name="post[category]" value="rails" checked="checked" />
#    <input type="radio" id="post_category_java"
 name="post[category]" value="java" />

radio_button("user", "receive_newsletter", "yes")
radio_button("user", "receive_newsletter", "no")
# => <input type="radio" id="user_receive_newsletter_yes"
 name="user[receive_newsletter]" value="yes" />
#    <input type="radio" id="user_receive_newsletter_no"
 name="user[receive_newsletter]" value="no" checked="checked" />
```

FormTagHelper's

What Are FormTagHelpers For?

- Provides a number of methods for creating form tags that doesn‘t rely on an Active Record object assigned to the template like FormHelper does. Instead, you provide the names and values manually.

FormTagHelper's

- check_box_tag
- field_set_tag
- file_field_tag
- form_tag
- hidden_field_tag
- image_submit_tag
- label_tag
- password_field_tag
- radio_buttoon_tag
- select_tag
- submit_tag
- text_area_tag
- text_field_tag

text_field_tag

- *text_field_tag(name, value = nil, options = {})*
- Creates a standard text field; use these text fields to input smaller chunks of text like a username or a search query
- Options
  - :disabled - If set to true, the user will not be able to use this input.
  - :size - The number of visible characters that will fit in the input.
  - :maxlength - The maximum number of characters that the browser will allow the user to enter.
  - Any other key creates standard HTML attributes for the tag.

text_field_tag Helper Examples

```
text_field_tag 'name'
 # => <input id="name" name="name" type="text" />

 text_field_tag 'query', 'Enter your search query here'
 # => <input id="query" name="query" type="text" value="Enter your
   search query here" />

 text_field_tag 'request', nil, :class => 'special_input'
 # => <input class="special_input" id="request" name="request"
   type="text" />

 text_field_tag 'address', '', :size => 75
 # => <input id="address" name="address" size="75" type="text"
   value="" />
```

text_field_tag Helper Examples (Cont)

```
text_field_tag 'zip', nil, :maxlength => 5
# => <input id="zip" maxlength="5" name="zip" type="text" />

text_field_tag 'payment_amount', '$0.00', :disabled => true
# => <input disabled="disabled" id="payment_amount"
  name="payment_amount" type="text" value="$0.00" />
```

Thank you!

We do Instructor-led Codecamps!
http://www.javapassion.com/codecamps