FormHelper in Rails

Reference: http://api.rubyonrails.org/classes/ActionView/Helpers/FormHelper.html

Form helpers are designed to make working with resources much easier. Forms for models are created with `form_for`. The form builder is able to generate default values for input fields that correspond to model attributes, and also convenient names, IDs, endpoints, etc.

As we said above, in addition to manually configuring the form_for call, you can rely on automated resource identification, which will use the conventions and named routes of that approach. This is the preferred way to use form_for nowadays.

For example, if User is an existing model and @user is a record you want to edit, a simple form with two fields can be written as

```
<%= form_for @user do |f| %>
  <%= f.label :first_name %>:
  <%= f.text_field :first_name %><br />

  <%= f.label :last_name %>:
  <%= f.text_field :last_name %><br />

  <%= f.submit %>
<% end %>
```

Note that *first_name* and *last_name* are attributes defined in the User model.

The HTML generated for this would be (modulus formatting):

```
<form action="/users" class="new_user" id="new_user" method="post">
  <div style="margin:0;padding:0;display:inline">
    <input name="authenticity_token" type="hidden"
           value="NrOp5bsjoLRuK8IW5+dQEYjKGUJDe7TQoZVvq95Wteg=" />
  </div>

  <label for=”user_first_name">First name</label>:
  <input id="user_first_name" name="user[first_name]" size="30"
                                      type="text" /><br />

  <label for="user_last_name">Last name</label>:
  <input id="user_last_name" name="user[last_name]" size="30"
                           type="text" /><br />

  <input name="commit" type="submit" value="Create User" />
</form>
```

As you see, the HTML reflects knowledge about the resource in several spots, like the path the form should be submitted to, or the names of the input fields.

Check Boxes:

```
# Let's say that @user.member is 1:

<%= f.check_box(:member, options = {}, checked_value = "1",
                                        unchecked_value = "0") %>


# => <input name="user[member]" type="hidden" value="0" />
#    <input type="checkbox" id="user_member" name="user[member]"
                                        value="1" />
```

Text fields

```
<%= f.text_field(:name, :size => 20, :class=> 'class_def') %>

CREATES HTML TAG:

<input type="text" id="user_name" name="user[name]" size="20"
                            value="#{@user.name}" class="class_def" />
```

Radio buttons

```
<%= f.radio_button(:receive_newsletter, "yes") %>
<%= f.radio_button(:receive_newsletter, "no", :checked=>true) %>

CREATES HTML TAGS
# => <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" />
```

Text area

```
<%= f.text_area(:comment, :cols => 20, :rows => 40) %>

CREATES HTML TAGS:
# => <textarea cols="20" rows="40" id="user_comment"
name="user[comment]">
#      #{@user.comment}
#    </textarea>Select boxes
```

Drop down box, Selection list

```
<%= f.select(:education, [['High School', 1], ['Bachelor', 2],
                                          ['PostGraduate', 3]]) %>
CREATES HTML TAGS

<select name="user[education]" id="user_education">
```

```
  <option value="1">High School</option>
  <option value="2">Bachelor</option>
  <option value="3">PostGraduate</option>
</select>
```

DropDown box for dates: Date_select

```
<%= f.date_select :birth_date  %>

CREATES HTML TAGS

<select id="user_birth_date_1i" name="user[birth_date(1i)]"> ...
</select>

<select id="user_birth_date_2i" name="user[birth_date(2i)]"> ...
</select>

<select id="user_birth_date_3i" name="user[birth_date(3i)]"> ...
</select>
```

which results in a params hash like

```
{:user => {'birth_date(1i)' => '2008', 'birth_date(2i)' => '11',
'birth_date(3i)' => '22'}}
```

When this is passed to User.new (or update_attributes), Active Record spots that these parameters should all be used to construct the birth_date attribute and uses the suffixed information to determine in which order it should pass these parameters to functions such as Date.civil.

To change the years displayed in a dropdown box for specifying a date, use the :start_year and the :end_year options. For instance to change the starting and ending year in the dropbox, use the following code:

```
<%= f.date_select :birth_date, :start_year => 1960, :end_year => 2010 %>
```

To customize the order in which the date fields are shown, use the :order option. The following code displays the date fields with the order: day, month and year:

```
<%= f.date_select :birth_date, :start_year => 1960, :end_year => 2010,
                                    :order => [:day, :month, :year] %>
```

See this link for more information on date_select method:
http://api.rubyonrails.org/classes/ActionView/Helpers/DateHelper.html#method-i-date_select

Params

The two basic structures are arrays and hashes. Hashes mirror the syntax used for accessing the value in params. For example if a form contains

```
<input id="user_name" name="user[name]" type="text" value="Henry"/>
```

the params hash will contain

```
{'user' => {'name' => 'Henry'}}
```

and params[:user][:name] will retrieve the submitted value in the controller.