Dizzy Making business beautiful.

# Cheatsheet * FormHelper

## parsing form data

When a form is submitted to a Rails application, the parameters are automatically translated by Rails into the `params` object which is accessible as a hash structure.

- Key/value pairs of your form's input fields are stored simply as key/value pairs in the `params` hash, such as the id which is extracted by routing from the URL:

| | | |
|---|---|---|
| `/customers/1` | `id=1` | `{ :id => "1" }` |

- Square brackets `[]` are used to build more complex, nested structures:

| | | |
|---|---|---|
| `text_field :user, :name` | `user[name]=David` | `{ :user => { :name => "David" }` |
| `text_field "user[address]", :city` | `user[address][city]=London` | `{ :user => { :address => { :city => "London" }}}` |
| `text_field "user[address]", :street` | `user[address][street]=Road` | `{ :user => { :address => { :street => "Road" }}}` |

- Using empty square brackets `[]` after the name of a model object, such as `address[]`, will insert the `id` of the record you are editing into the input field, useful for editing multiple records on one form

| | | |
|---|---|---|
| `text_field "address[]", :country` | `address[4][country]=England` | `{ :address => { 4 => { :country => "England" }}}` |
| `text_field "address[]", :town` | `address[4][town]=London` | `{ :address => { 4 => { :town => "London" }}` |

- If the record is new and has no `id`, then upon submitting the form, Rails will convert the fields into an array of hashes in order of appearance:

| | | |
|---|---|---|
| `text_field "address[]", :country` | `address[][country]=England` | `{ :address => [` |
| `text_field "address[]", :town` | `address[][town]=London` | `{ :country => "England", :town => "London" },` |
| `text_field "address[]", :country` | `address[][country]=Australia` | `{ :country => "Australia", :town => "Sydney" } ]` |
| `text_field "address[]", :town` | `address[][town]=Sydney` | `}` |

## input field helpers

```
f.error_messages
f.check_box :terms, { :class => 'check' }, "yes", "no"
f.file_field :image
f.hidden_field :id
f.label :customer, "Text for label"
f.password_field :password
f.radio_button :language, "French"
f.text_area :comment, :size => "20x30", :disabled => "disabled"
f.text_field :age, :size => "20", :class => "age_box"
```

## Example

**Controller**

```
def new
    @customer = Customer.new
    3.times do
        @customer.addresses.build
    end
end
```

**View**

```
<% form_for(@customer) do |f| %>
    <%= f.text_field :name %>
    <%= f.text_field :email %>
    <% @customer.addresses.each do |address| %>
        <% fields_for "customer[addresses][]", address do |fields| %>
        <%= fields.text_field :number %>
        <%= fields.text_field :street %>
    <% end %>
<% end %>
```

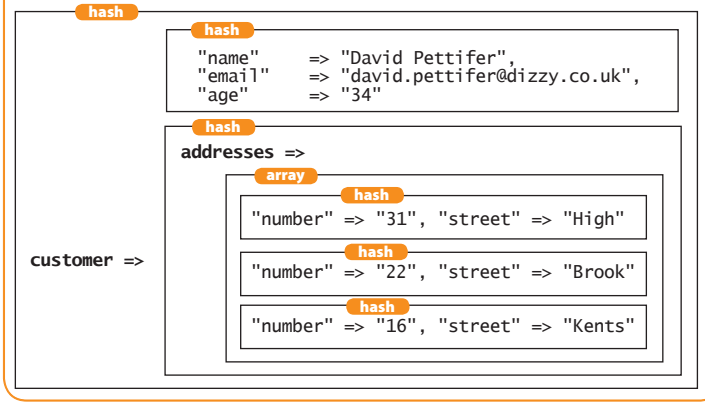
**HTML**

```
<form id="new_customer" class="new_customer" method="post" action="/customers">
<input type="text" size="30" name="customer[name]"/>
<input type="text" size="30" name="customer[email]"/>
<input type="text" size="30" name="customer[addresses][][number]"/>
<input type="text" size="30" name="customer[addresses][][street]"/>
<input type="text" size="30" name="customer[addresses][][number]"/>
<input type="text" size="30" name="customer[addresses][][street]"/>
<input type="text" size="30" name="customer[addresses][][number]"/>
<input type="text" size="30" name="customer[addresses][][street]"/>
<input type="submit" value="Create" name="commit"/>
</form>
```

**params[]**

```
params = {
    "customer" => {   "name"=>"David Pettifer",
                      "email"=>"david.p@dizzy.co.uk",
                      "addresses"=> [
                          {"number"=>"31", "street"=>"High"},
                          {"number"=>"22", "street"=>"Brook"},
                          {"number"=>"16", "street"=>"Kents"}    ]
                  } }
```

**Visual representation of params**




## form_for

`form_for` is used to easily manipulate HTML forms which are based upon ActiveRecord model objects:

```
<%= form_for(:customer, @customer, :url => { :controller => "customers", :action => "create" }, :html => { :multipart => true, :method => :put }) do |f| %>
    <%= f.text_field :age %>
    <%= text_field "customer", :age %>
<%= submit_tag %>
<% end %>
```

**Parameters**

| | | |
|---|---|---|
| `:customer` ***required*** | *:symbol* or *"string"* | The name of the model object for all the fields in the form. All input fields will be prefixed with this. Rails will also look for an `@instance_variable` with the same name which should contain an instance of an *existing* or *new* ActiveRecord model object |
| `@customer` ***optional*** | *ActiveRecord model object* | If the `@instance_variable` containing the model object is named differently, you can pass a variable containing the actual model object here |
| `:url` ***optional*** | *"string"* or *{hash}* | The URL to post the form to. Can take an explicit url as a string, or a hash in the same format as `url_for` |
| `:html` ***optional*** | *{hash}* | A {hash} of HTML attributes which will be added to the HTML `<form>` tag. |
| `:method` ***optional*** | *:symbol* | Pass as part of the {hash} of HTML attributes. Can be `:put`, `:post`, `:get` or `:delete` |

## fields_for

`fields_for` creates a scope around a specific model object like `form_for`, but doesn't create the form tags themselves, making `fields_for` suitable for specifying additional model objects in the same form. See the example on the left.

## RESTful form_for

When standard routes are used in a RESTful context, Rails will reflect upon the object passed to it and automatically build a form with the relevant RESTful URL depending on whether the form is wrapping a new record (`create`) or an existing record (`update`). Nested routes will require you to be more verbose.

| **Standard routes** | *new record?* | *method* | *URL* |
|---|---|---|---|
| `form_for(@customer)` | ✓ | POST | `/customers` |
| `form_for(@customer)` | × | PUT | `/customers/1` |
| **Nested routes** | | | |
| `form_for(@address, :url => customer_addresses_path(@customer))` | ✓ | POST | `/customers/1/addresses` |
| `form_for(@address, :url => customer_addresses_path(@customer))` | × | PUT | `/customers/1/addresses/24` |

## Multipart form

**View**

```
<% form_for(@customer, :html => { :multipart => true }) do |f| %>
    <%= f.file_field :image_file %>
    <%= submit_tag %>
<% end %>
```

**Model**

```
class Customer < ActiveRecord::Base
    def image_file=(uploaded_data)
        self.filename     = uploaded_data.original_filename
        self.image_data   = uploaded_data.read
        self.size         = uploaded_data.size
        self.content_type = uploaded_data.content_type
    end
end
```