

Simple Contact Form using a Table-less Model

By Brian Hogan

This simple guide will take you through developing a contact form that will leverage the ActiveRecord validations but will not be bound to the database.

Create a Table-less model

Create a new file under **models/** called **tableless.rb**

Paste this code into the file:

```
class Tableless < ActiveRecord::Base

  def self.columns() @columns ||= []; end

  def self.column(name, sql_type = nil, default = nil, null = true)
    columns << ActiveRecord::ConnectionAdapters::Column.new(name.to_s, default,
      sql_type.to_s, null)
  end

  # override the save method to prevent exceptions.
  def save(validate = true)
    validate ? valid? : true
  end

end
```

Create the Contact model

Create a new file under **models/** called **contact.rb**

Paste this code into the file:

```
class Contact < Tableless
  column :name,           :string
  column :city,           :string
  column :state,          :string
  column :phone,          :string
  column :email_address, :string
  column :address,        :string
  column :zip,            :string
  column :message,        :text

  validates_presence_of :name, :email_address, :message
end
```

Notice that this new class extends the Tableless class you just made.

Also, notice the explicit calls to **column**; this is how you add new attributes to the model. You now have the ability to use this class just like an ActiveRecord class including use of all of the validation methods you're used to using.

Creating a Mailer model

Generate a new Mailer by going to your project root and typing

Simple Contact Form using a Table-less Model

By Brian Hogan

```
ruby script/generate mailer Mailer
```

This will create
models/mailer.rb and
views/mailer

Configuring ActionMailer

Place this code in your **config/environment.rb** file, towards the bottom of the file

```
c = YAML::load(File.open("#{RAILS_ROOT}/config/config.yml"))
ActionMailer::Base.server_settings = {
  :address => c[RAILS_ENV]['email']['server'],
  :port => c[RAILS_ENV]['email']['port'],
  :domain => c[RAILS_ENV]['email']['domain'],
  :authentication => c[RAILS_ENV]['email']['authentication'],
  :user_name => c[RAILS_ENV]['email']['username'],
  :password => c[RAILS_ENV]['email']['password']
}
CONTACT_RECIPIENT = c[RAILS_ENV]['email']['contact_recipient']
```

Create a config file

Create a new file under **config/** called config.yml

Modify the following data and then paste it into that file

```
development:
  email:
    server: your_development_server
    port: 25
    domain: your_domain
    authentication: none
    username:
    password:
    contact_recipient: your_email@server.com
production:
  email:
    server: your_production_server
    port: 25
    domain: bphogan.railsplayground.com
    authentication: login
    username: username
    password: your_password
    contact_recipient: your_email@server.com
```

Code a Delivery Handler

Open up **models/mailer.rb**

Add the following method

Simple Contact Form using a Table-less Model

By Brian Hogan

```
def contact_message(contact)
  @recipients = CONTACT_RECIPIENT
  @from = contact.email_address
  @subject = '[Scheppke.com] Contact Form Request'
  @body['name'] = contact.name
  @body['city'] = contact.city
  @body['state'] = contact.state
  @body['zip'] = contact.zip
  @body['phone'] = contact.phone
  @body['email'] = contact.email_address
  @body['address'] = contact.address
  @body['message'] = contact.message
end
```

Code your Response View

Create a new file under **app/views/mailer** called **contact_message.rhtml**

Paste the following code into the file:

```
A user has chosen to contact you via your web site.
=====
User:      <%=@name %>
Email:     <%=@email %>
Address:   <%=@address %>
City:      <%=@city %>
State:     <%=@state %>
Zip:       <%=@zip %>
Phone:     <%=@phone %>

Message:
-----
<%=@message %>

-----
Generated at <%=Time.now.to_s %>
```

Notice that you stored the variables in **@body[]** but you reference them directly? Another "strange" feature of ActionMailer that takes a little getting used to.

Code the controller actions

Let's assume we have a controller called **home** which is located at **app/controllers/home_controller.rb**

First, let's code the action to show the contact form. This is pretty easy.

```
def contact
  @contact = Contact.new
end
```

Next, we'll code the action that will receive the form data and mail the response.

Simple Contact Form using a Table-less Model

By Brian Hogan

```
def send_contact_request
  @contact = Contact.new(params['contact'])
  if @contact.save
    begin
      Mailer::deliver_contact_message(@contact)
      flash[:notice] = "Success!"
      redirect_to :action=>"contact"
    rescue
      flash[:notice] = "It broke!"
      render :action=>"contact"
    end
  else
    render :action=>"contact"
  end
end
```

In this method, we create an instance of a Contact model just like any other model. The important thing to notice here is that we are still calling the **save** method on our Contact object. If the save is successful then we invoke a special class method of our Mailer model.

```
Mailer::deliver_contact_message(@contact)
```

We don't directly call the **contact_message** method; instead, we call it indirectly.

Code the Contact Form

Lastly, to make this work, you'll need a view for your contact form so that users can actually enter some data.

Create a new file under views/controller_name/ called **contact.rhtml**

```
<%=error_messages_for "contact" %>

<%= start_form_tag :controller=>"home", :action=>"send_contact_request" %>
  <p>Name <%= text_field("contact", "name") %></p>
  <p>Email <%= text_field("contact", "email_address") %></p>
  <p>Address <%= text_field("contact", "address") %></p>
  <p>City <%= text_field("contact", "city") %>
    State <%= text_field("contact", "state", {"size"=>"5"}) %>
    Zip: <%= text_field("contact", "zip") %></p>
  <p>Phone <%= text_field("contact", "phone") %></p>
  <p>Message<br> <%= text_area("contact", "message") %></p>
  <%= submit_tag "Send" %>
<%= end_form_tag %>
```

That should just about do what you need. Test it out!

Acknowledgements

Tableless model from Rick Olson - http://rails.technoweenie.net/tip/2005/11/19/validate_your_forms_with_a_table_less_model