Here learn how to make a custom error_messages partial,
remove the field title from the message.
reflect on validations,
clean up complex validations in a model,
use of locales,

Lets make a new rails project ‘custom-err-msg-demo’

```
manish@manish-Vostro-1450:~/rails_projects$ rails new custom-err-msg-demo -T
      create
      create  README.rdoc
      create  Rakefile
      create  config.ru
      create  .gitignore
      create  Gemfile
      create  app
      create  app/assets/javascripts/application.js
      create  app/assets/stylesheets/application.css
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
      create  app/views/layouts/application.html.erb
      create  app/assets/images/.keep
      create  app/mailers/.keep
      create  app/models/.keep
      create  app/controllers/concerns/.keep
      create  app/models/concerns/.keep
      create  bin
      create  bin/bundle
      create  bin/rails
      create  bin/rake
      create  config
      create  config/routes.rb
      create  config/application.rb
      create  config/environment.rb
      create  config/environments
      create  config/environments/development.rb
      create  config/environments/production.rb
      create  config/environments/test.rb
      create  config/initializers
      create  config/initializers/backtrace_silencers.rb
      create  config/initializers/filter_parameter_logging.rb
      create  config/initializers/inflections.rb
      create  config/initializers/mime_types.rb
      create  config/initializers/secret_token.rb
      create  config/initializers/session_store.rb
      create  config/initializers/wrap_parameters.rb
```

cd custom-err-msg-demo/
and generate a scaffold for ‘user’ with following attributes
name:string, email:string, mobile_no:integer
rails g scaffold user name:string email:string mobile_no:integer

```
manish@manish-Vostro-1450:~/rails_projects$ cd custom-err-msg-demo/
manish@manish-Vostro-1450:~/rails_projects/custom-err-msg-demo$ rails g scaffold user name:string email:string mobile_no:integer
```

rake db:migrate

in app/models/user.rb

---

include ActiveModel::Validations

```
class User < ActiveRecord::Base
  include ActiveModel::Validations
  attr_accessor :name, :email

  validates :name, :presence =>  {:message=>"Please Enter name"}, :uniqueness => true, :length => {
:maximum => 25 }
  validates :email, :format => { :with => /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/i, :on => :create,
:message=>'Plz input valid email id' }
  validates :mobile_no,:presence => true, :numericality => { :message => " should be a number" }
end
```

make a file in app/views/shared/_error_messages.html.erb

```
<% if target.errors.any? %>
   <div id="error_explanation">
     <h2><%= pluralize(target.errors.count, "error") %> prohibited this recaord from being saved:</h2>
            <ul>
                <% target.errors.full_messages.each do |msg| %>
                  <li><%= msg %></li>
                <% end %>
            </ul>
     </div>
 <% end %>
```

in file users/_form.html.erb

```
<%= form_for(@user) do |f| %>
<%= render 'shared/error_messages', :target=>@user %>
  <div class="field">
            :
            :
            :
            :
<% end %>
```

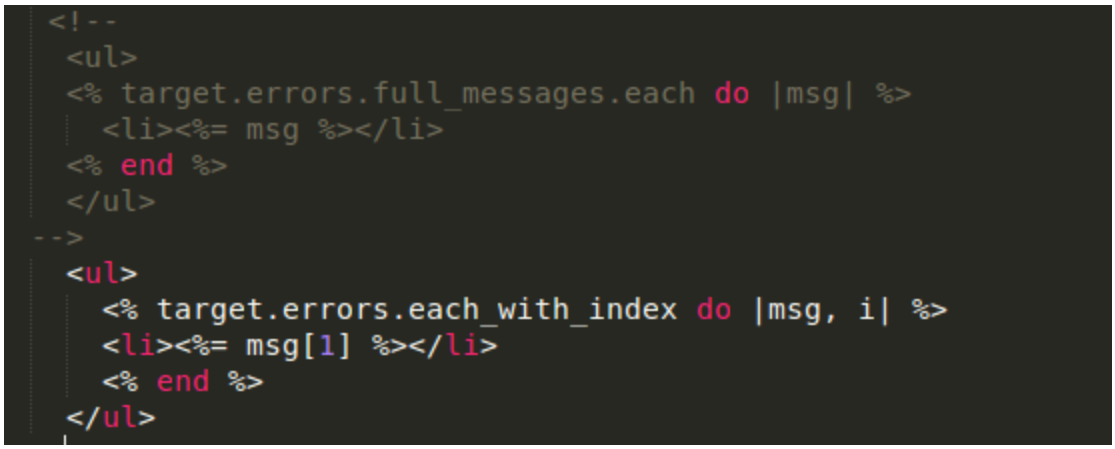
Result:

- Name Please Enter name
- Email can't be blank
- Email is not an email message from locales
- Mobile no can't be blank
- Mobile no should be a number

:::: To remove 'field title' from the 'messages' ::::

replace code like this in app/views/shared/_error_messages.html.erb

```
<!--
  <ul>
  <% target.errors.full_messages.each do |msg| %>
    <li><%= msg %></li>
  <% end %>
  </ul>
-->
  <ul>
    <% target.errors.each_with_index do |msg, i| %>
    <li><%= msg[1] %></li>
    <% end %>
  </ul>
```



It will result like this:

- Please Enter name
- can't be blank
- is not an email message from locales
- can't be blank
- should be a number

::: To place a Mark * on required fields :::

Name*

Email*

Mobile no*

Create User

in app/view/users/_form.html.erb

Use

```
<%= mark_required(@user, :name) %>
```

```
<%= form_for(@user) do |f| %>
<%= render 'shared/error_messages', :target=>@user %>
  <div class="field">
    <%= f.label :name %><%= mark_required(@user, :name) %><br>
    <%= f.text_field :name %>
  </div>
  <div class="field">
    <%= f.label :email %><%= mark_required(@user, :email) %><br>
    <%= f.text_field :email %>
  </div>
  <div class="field">
    <%= f.label :mobile_no %><%= mark_required(@user, :mobile_no) %><br>
    <%= f.number_field :mobile_no %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```

in app/helpers/application_helper.rb

```
def mark_required(object, attribute)
  "*" if object.class.validators_on(attribute).map(&:class).include? ActiveRecord::Validations::PresenceValidator
end
```

```
2.0.0p247 :014 > @user=User.first
  User Load (0.4ms)  SELECT "users".* FROM "users" ORDER BY "users"."id" ASC LIMIT 1
 => #<User id: 1, name: "", email: "", mobile_no: nil, created_at: "2013-12-28 11:31:18", updated_at: "2013-12-28 11:31:18">
2.0.0p247 :015 > @user.class.validators_on(:mobile_no)
 => [#<ActiveRecord::Validations::PresenceValidator:0xa472688 @attributes=[:mobile_no], @options={}>, #<ActiveModel::Validations::NumericalityValidato
r:0xa471ef4 @attributes=[:mobile_no], @options={:message=>" should be a number"}>]
```

To check for validation class present on object , u can try like this:

@user=User.first

@user.class.validators_on(:name).map(&:class)
=> [ActiveRecord::Validations::PresenceValidator]

@user.class.validators_on(:name).map(&:class).include? ActiveRecord::Validations::PresenceValidator
=> true
@user.class.validators_on(:mobile_no)
=> [#<ActiveRecord::Validations::PresenceValidator:0xa472688 @attributes=[:mobile_no], @options={}>, #<ActiveModel::Validations::NumericalityValidator:0xa471ef4 @attributes=[:mobile_no], @options={:message=>" should be a number"}>]

:::: clean up complex validations in a model :::

```
class User < ActiveRecord::Base
  include ActiveModel::Validations
  attr_accessor :name, :email


  validates :name, :presence =>  {:message=>"Please Enter name"}, :uniqueness => true, :length => { :maximum => 25 }
  validates :email, :presence => true, :email_format => true
  validates :mobile_no, :presence => true, :numericality => { :message => " should be a number" }
end
```

validates :email, :presence => true, :email_format => true

:email_format => true will call EmailFormatValidator and add your message in error object if its value doesn't satisfies required conditions. In present scenario we are using ':emailmsg' I18n to display error message.

lib/email_format_validator.rb

```
class EmailFormatValidator < ActiveModel::EachValidator
  def validate_each(record, attribute, value)
    record.errors.add(attribute, options[:message] || :emailmsg) unless
      value =~ /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i
  end
end
```

class EmailFormatValidator < ActiveModel::EachValidator
 def validate_each(record, attribute, value)
  record.errors.add(attribute, options[:message] || :emailmsg) unless
   value =~ /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i
 end
end

You can use your own message in place of ‘:emailmsg’

:::: Use of locales ::::

To use a different locale, set it with `I18n.locale`
in config/locales/en.yml

```
en:
  hello: "Hello world"

  activerecord:
    errors:
      messages:
        emailmsg: "is not an email message from locales"
```