RUBY ON RAILS - FILE UPLOADING

You may have a requirement in which you want your site visitors to upload a file on your server. Rails makes it very easy to handle this requirement. Now we will proceed with a simple and small Rails project.

As usual, let's start off with a new Rails application called **testfile**. Let's create the basic structure of the application by using simple rails command.

```
tp> rails new testfile
```

Before starting application development, we should install gem files as shown below –

```
gem install carrierwave
gem install bootstrap-sass
```

Open up your gemfile and add the following two gems at the bottom as shown in the following image –

```
42
43  # Windows does not include zoneinfo files, so bundle the tzinfo-data gem
44  gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]
45
46  gem 'carrierwave', '~> 0.9'
47  gem 'bootstrap-sass', '~> 2.3.2'
```

After adding gems in the gem file, we need to run the following command on the console –

```
bundle install
```

Creating the Model

We need to create a model with two strings as name and attachment as shown below –

```
rails g model Resume name:string attachment:string
```

We need to create the database migration as shown below –

```
rake db:migrate
```

We need to generate the controller as shown below –

```
rails g controller Resumes index new create destroy
```

Great! Now we have the basic structure set up. Now we need to create an uploader. An Uploader came from carrierwave gem and it tells to carrierwave how to handle the files. In short, it contained all file processing functionalities. Run the command to create an uploader as shown below

```
rails g uploader attachment
```

Now open the resume model and call the uploader as shown below. Resume model has placed at app/models/resume.rb –

```
class Resume < ActiveRecord::Base
   mount_uploader :attachment, AttachmentUploader # Tells rails to use this uploader for this
model.
   validates :name, presence: true # Make sure the owner's name is present.
end
```

Before working on controller, we need to modify our config/routes.db as shown below –

```
CarrierWaveExample::Application.routes.draw do
   resources :resumes, only: [:index, :new, :create, :destroy]
   root "resumes#index"
end
```

Lets us edit the controller as shown below.

```
class ResumesController < ApplicationController
   def index
      @resumes = Resume.all
   end

   def new
      @resume = Resume.new
   end

   def create
      @resume = Resume.new(resume_params)

      if @resume.save
         redirect_to resumes_path, notice: "The resume #{@resume.name} has been uploaded."
      else
         render "new"
      end

   end

   def destroy
      @resume = Resume.find(params[:id])
      @resume.destroy
      redirect_to resumes_path, notice:  "The resume #{@resume.name} has been deleted."
   end

   private
      def resume_params
      params.require(:resume).permit(:name, :attachment)
   end

end
```

Let's add bootstrap implementation in css file.css file could be in app/assets/stylesheets/resumes.css.scss

```
@import "bootstrap";
```

Now open up app/views/layouts/application.html.erb and add codes as shown below –

```
<!DOCTYPE html>
<html>

   <head>
      <title>Tutorialspoint</title>
      <%= stylesheet_link_tag "application", media: "all", "data-turbolinks-track" => true %>
      <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
      <%= csrf_meta_tags %>
   </head>

   <body>
      <div class = "container" style = "padding-top:20px;">
         <%= yield %>
      </div>
   </body>

</html>
```

Now we need to set up index views as shown below –

```
<% if !flash[:notice].blank? %>
   <div class = "alert alert-info">
      <%= flash[:notice] %>
   </div>
<% end %>

<br />

<%= link_to "New Resume", new_resume_path, class: "btn btn-primary" %>
<br />
<br />

<table class = "table table-bordered table-striped">
   <thead>.
      <tr>
         <th>Name</th>
         <th>Download Link</th>
         <th> </th>
      </tr>
   </thead>

   <tbody>
      <% @resumes.each do |resume| %>

         <tr>
            <td><%= resume.name %></td>
            <td><%= link_to "Download Resume", resume.attachment_url %></td>
            <td><%= button_to "Delete",  resume, method: :delete, class: "btn btn-danger",
confirm: "Are you sure that you wish to delete #{resume.name}?" %></td>
         </tr>

      <% end %>
   </tbody>

</table>
```

Now, lets edit new.html.erb and add our form code.

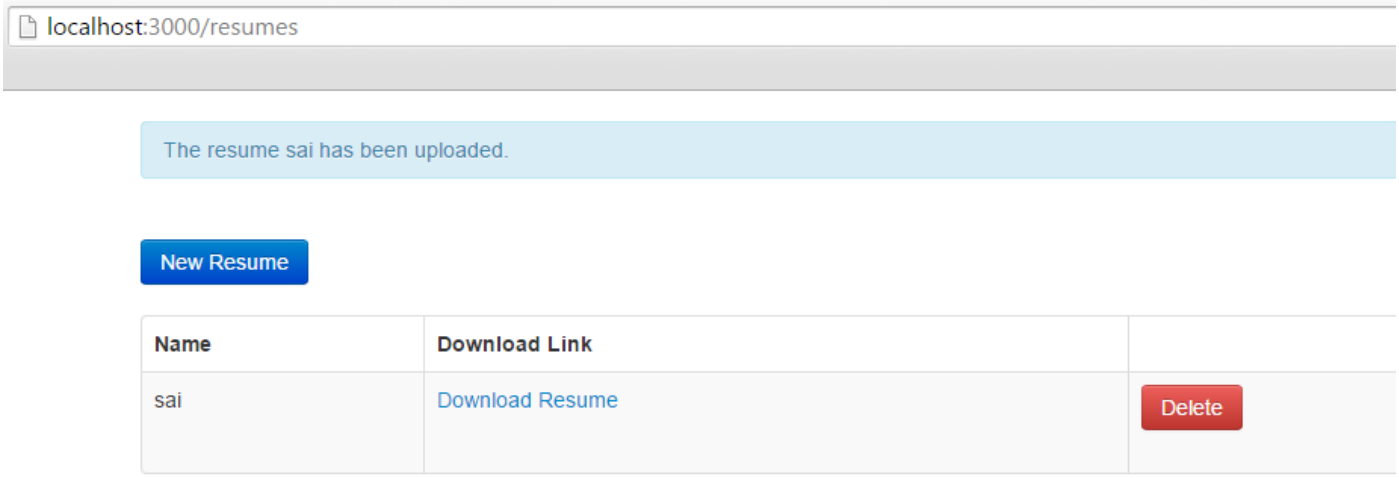
```
<% if !@resume.errors.empty? %>
   <div class = "alert alert-error">

      <ul>
         <% @resume.errors.full_messages.each do |msg| %>
            <li><%= msg %></li>
         <% end %>
      </ul>

   </div>
<% end %>

<div class = "well">
   <%= form_for @resume, html: { multipart: true } do |f| %>
      <%= f.label :name %>
      <%= f.text_field :name %>
      <%= f.label :attachment %>
      <%= f.file_field :attachment %>
      <%= f.submit "Save", class: "btn btn-primary" %>
   <% end %>
</div>
```

Now start the server and visit http://localhost:3000. It will produce a screen similar to as follows –



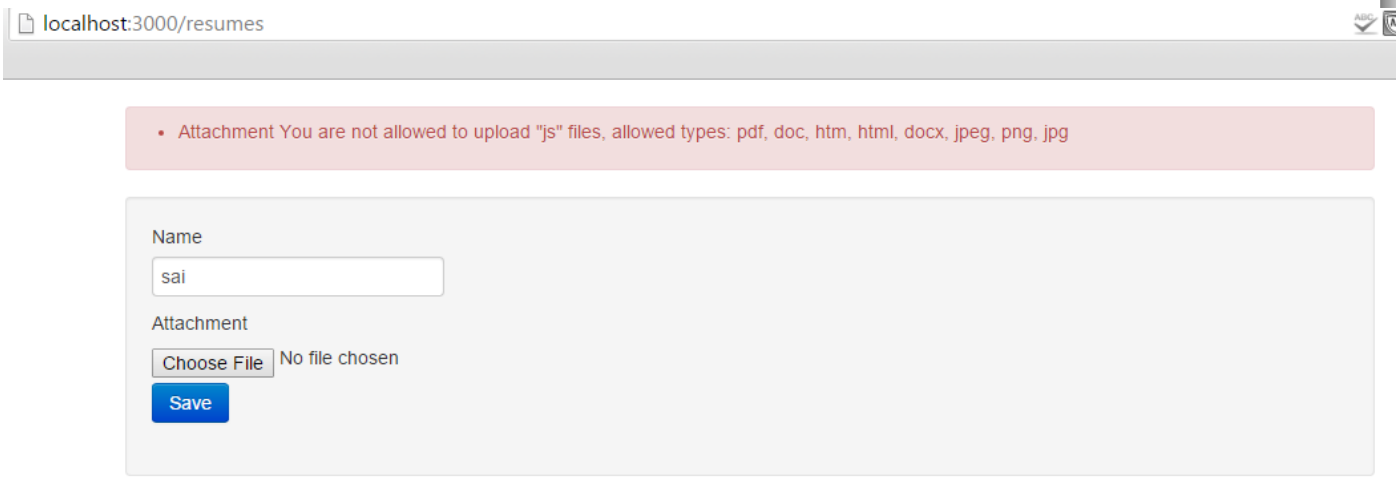
One last thing we need to do is filter the list of allowed filetypes. For that we need add simple code as shown below at app/uploaders/attachment_uploader.rb

```
class AttachmentUploader < CarrierWave::Uploader::Base
   storage :file

   def store_dir
      "uploads/#{model.class.to_s.underscore}/#{mounted_as}/#{model.id}"
   end
```

```
  def extension_white_list
     %w(pdf doc htm html docx)
  end
end
```

Now start the server and visit http://localhost:3000. Now input a wrong format; it will generate a wrong message as shown below –



For a complete detail on **File** object, you need to go through the **Ruby Reference Manual**.