Generating PDFs From HTML With Rails

by Pedro Alonso 23 Aug 2016

Difficulty: Intermediate Length: Medium Languages: English

Ruby Ruby on Rails CSS HTML PDF Web Apps

There are many ways to generate PDFs in Ruby and Rails. Chances are that you are already familiar with HTML and CSS, so we are going to use PDFKit to generate PDF files using HTML from standard Rails view and style code.

Introduction to PDFKit

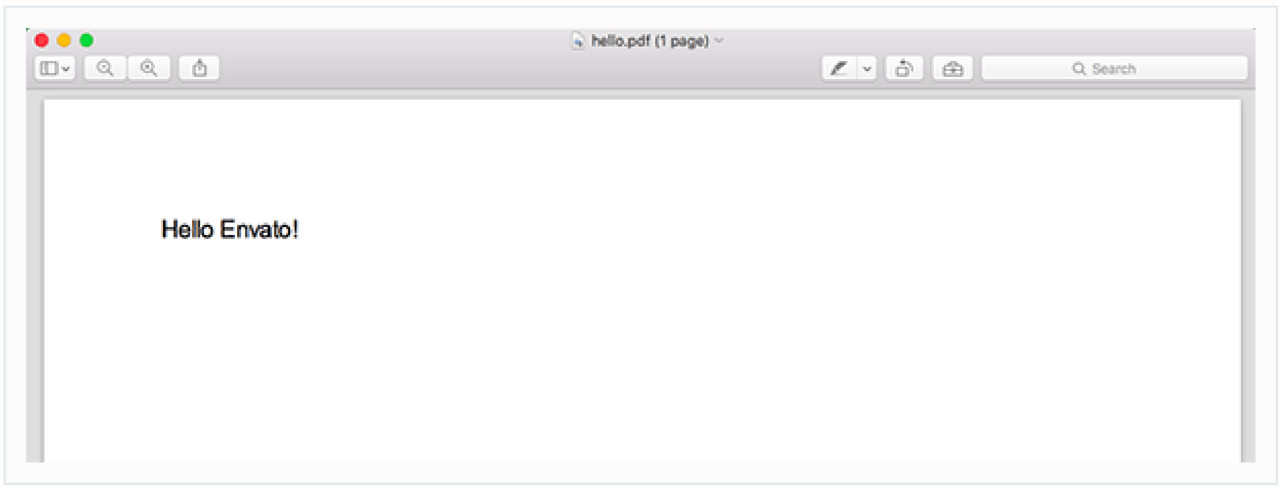
Internally, PDFKit uses wkhtmltopdf (WebKit HTML to PDF), an engine that will take HTML and CSS, render it using WebKit, and output it as a PDF with high quality.

To start, install wkhtmltopdf on your computer. You candownload the binary or install from Brew on Mac, or your preferred Linux repository.

You also need to install the `pdfkit gem`, and then run the following bit of Ruby to generate a PDF with the text "Hello Envato!"

```
require "pdfkit"

kit = PDFKit.new(<<-HTML)
  <p>Hello Envato!</p>
HTML

kit.to_file("hello.pdf")
```

You should have a new file calledhello.pdf with the text at the top.

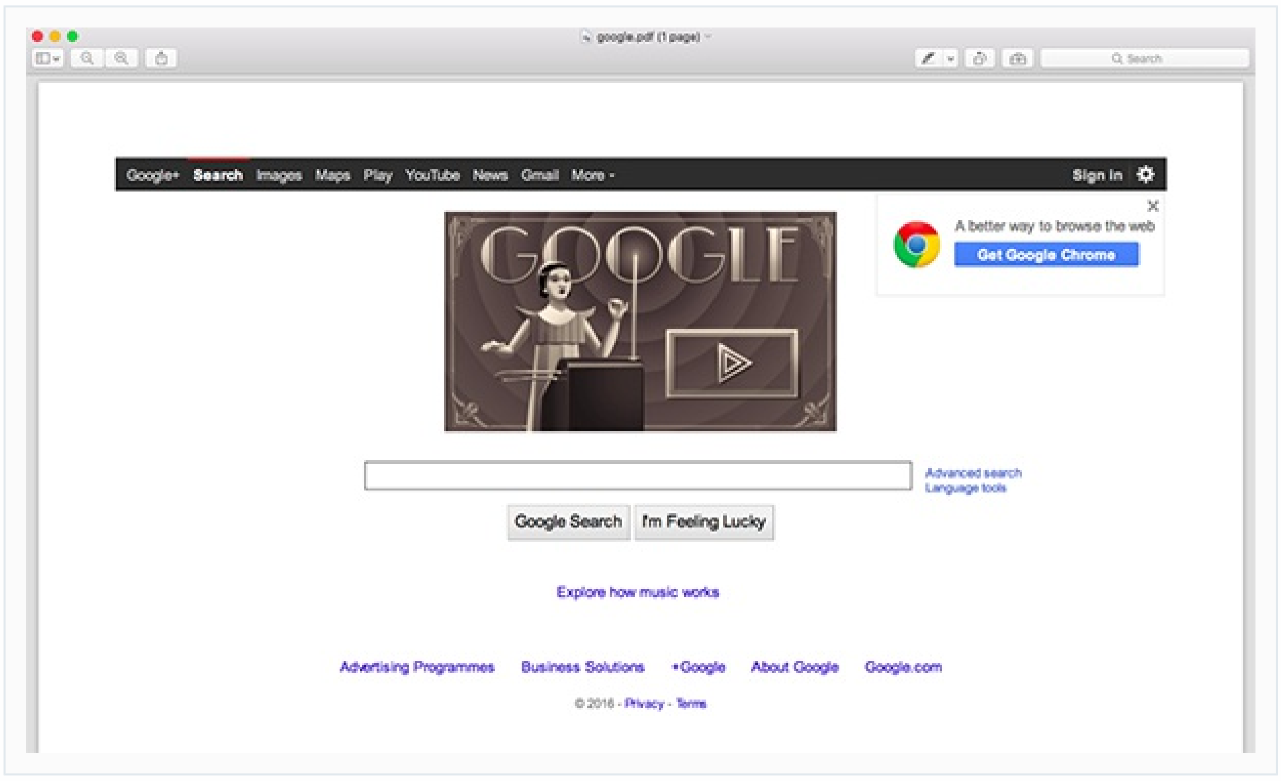


Hello Envato!

PDFKit also allows you to generate a PDF from a URL. If you want to generate a PDF from the Google homepage, you can run:

```ruby
require "pdfkit"

PDFKit.new('https://www.google.com', :page_size => 'A3').to_file('google.pdf')
```

As you can see, I'm specifying the `page_size` —by default, A4 is used. You can see a full list of optionshere.

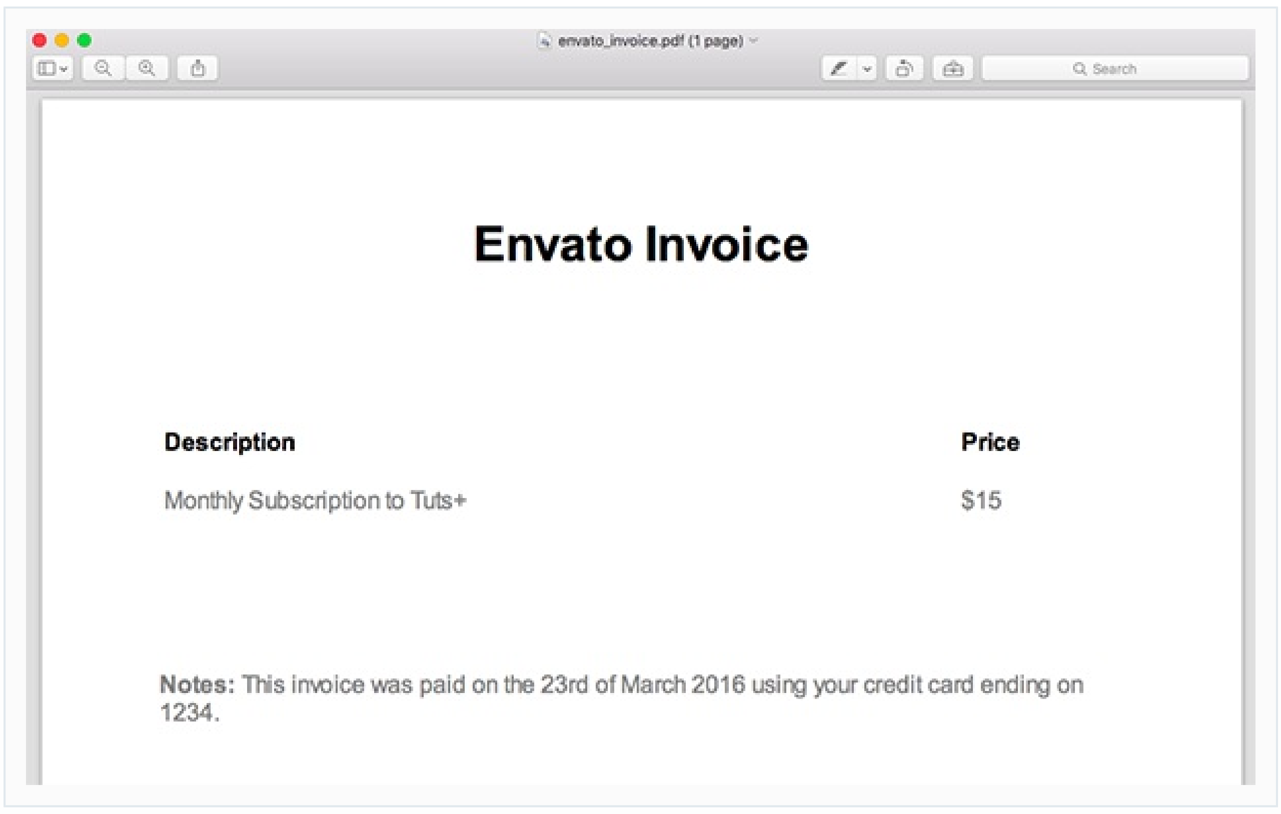


Styling Your PDF Using CSS

Earlier I mentioned that we are going to generate PDF files using HTML and CSS. In this sample, I have added a bit of CSS to style the HTML for a sample invoice, as you can see:

```
require "pdfkit"

kit = PDFKit.new(<<-HTML)
  <style>
    * {
      color: grey;
    }
    h1 {
      text-align: center;
      color: black;
      margin-bottom: 100px;
    }
    .notes {
      margin-top: 100px;
    }

    table {
      width: 100%;
    }
    th {
      text-align: left;
      color: black;
      padding-bottom: 15px;
    }
  </style>

  <h1>Envato Invoice</h1>

  <table>
    <thead>
        <tr>
          <th>Description</th>
          <th>Price</th>
        </tr>
      </thead>
      <tbody>
          <tr>
            <td>Monthly Subscription to Tuts+</td>
            <td>$15</td>
          </tr>
      </tbody>
  </table>

  <div class="notes">
    <p><strong>Notes:</strong> This invoice was paid on the 23rd of March 2016 using your credit card end
  </div>
HTML

kit.to_file("envato_invoice.pdf")
```

If you run this script, the file envato_invoice.pdf will be generated. This photo shows the result of the sample invoice:



Envato Invoice

Description	Price
Monthly Subscription to Tuts+	$15

Notes: This invoice was paid on the 23rd of March 2016 using your credit card ending on 1234.

As you can see, PDFKit is very easy to use, if you are already familiar with HTML and CSS. You can continue customising or styling this document as you like.

Using PDFKit From a Rails Application

Now let's take a look at how to use PDFKit in the context of a Rails application, so we can dynamically generate PDF files using the data from our models. In this section we're going to build a simple rails application to generate the previous "Envato Invoice" dynamically. Start by creating a new rails app and adding three models:

```
$ rails new envato_invoices
$ cd envato_invoices

$ rails generate model invoice date:date client notes
$ rails generate model line_item description price:float invoice:references

$ rake db:migrate
```

Now, we have to add some sample data to the database. Add this code snippet to `db/seeds.rb`.

```
line_items = LineItem.create([
    { description: 'Tuts+ Subscription April 2016', price: 15.0 },
    { description: 'Ruby eBook', price: 9.90} ])
Invoice.create(
    client: 'Pedro Alonso',
    total: 24.90,
    line_items: line_items,
    date: Date.new(2016, 4, 1))
```

Run `rake db:seed` in your terminal to add the sample invoice to the database.

We are also interested in generating a list of invoices and the detail of one invoice in our app, so using rails generators, run `rails generate controller Invoices index show` to create the controller and views.

app/controllers/invoices_controller.rb

```
class InvoicesController < ApplicationController
  def index
    @invoices = Invoice.all
  end

  def show
    @invoice = Invoice.find(params[:id])
  end
end
```

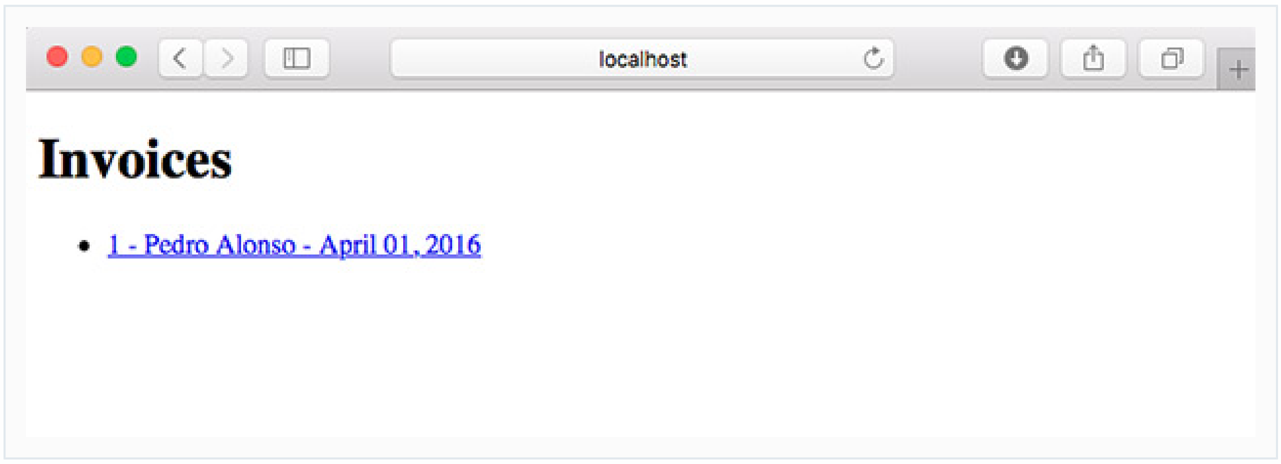
app/views/invoices/index.html.erb

```
<h1>Invoices</h1>
<ul>
  <% @invoices.each do |invoice| %>
  <li>
    <%= link_to "#{invoice.id} - #{invoice.client} - #{invoice.date.strftime("%B %d, %Y")} ", invoice_path(
  </li>
  <% end %>
</ul>
```

We need to modify rails routes to redirect to `InvoicesController` by default, so edit `config/routes.rb`:

```
Rails.application.routes.draw do
  root to: 'invoices#index'

  resources :invoices, only: [:index, :show]
end
```

Start your `rails server` and navigate to localhost:3000 to see the list of invoices:



Invoices

- 1 - Pedro Alonso - April 01, 2016

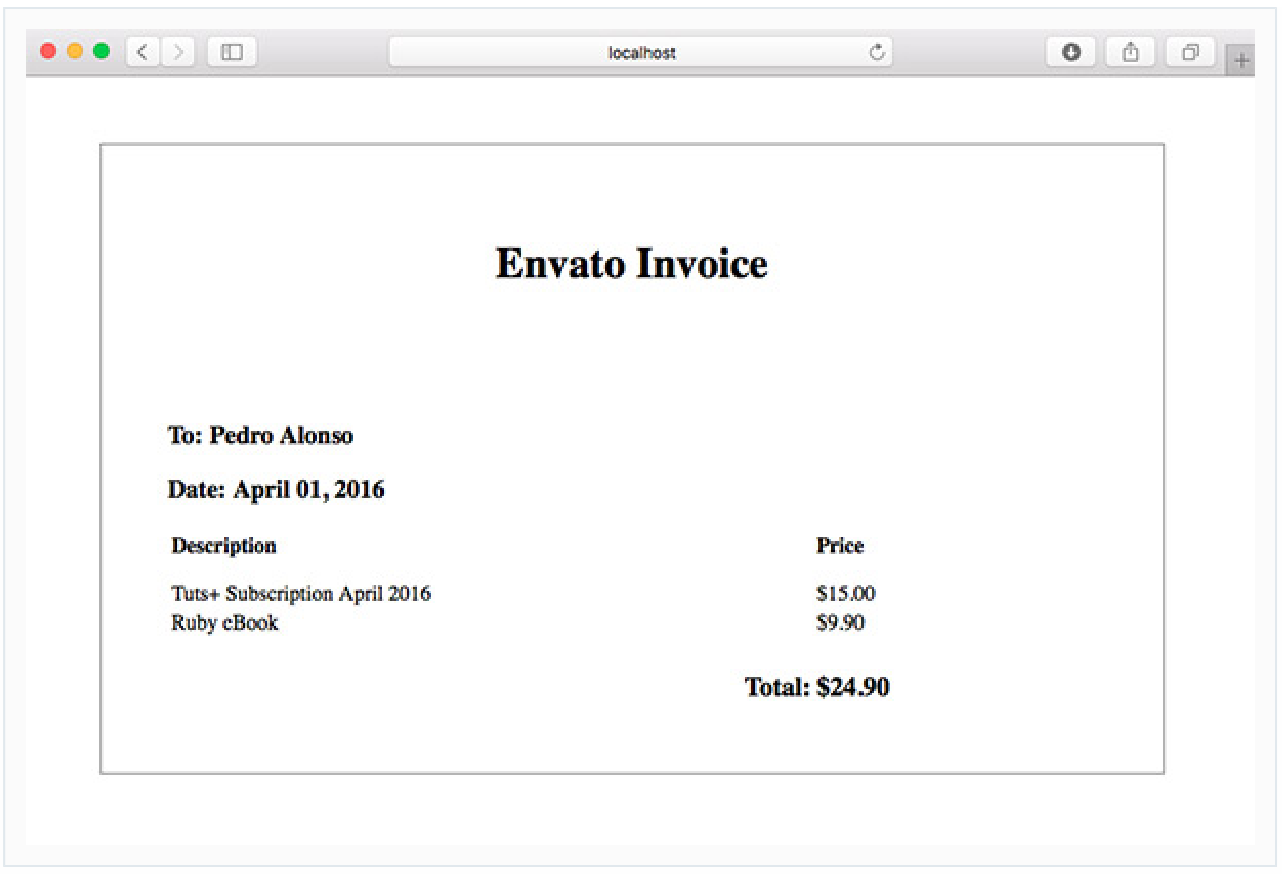
app/views/invoices/show.html.erb

```
<div class="invoice">
  <h1>Envato Invoice</h1>

  <h3>To: <%= @invoice.client %></h3>
  <h3>Date: <%= @invoice.date.strftime("%B %d, %Y") %></h3>

  <table>
    <thead>
        <tr>
          <th>Description</th>
          <th>Price</th>
        </tr>
      </thead>
      <tbody>
        <% @invoice.line_items.each do |line_item| %>
          <tr>
            <td><%= line_item.description %></td>
            <td><%= number_to_currency(line_item.price) %></td>
          </tr>
        <% end %>
        <tr class="total">
          <td style="text-align: right">Total: </td>
          <td><%= number_to_currency(@invoice.total) %></span></td>
        </tr>
      </tbody>
  </table>

  <% if @invoice.notes %>
  <div class="notes">
    <p><strong>Notes:</strong> <%= @invoice.notes %></p>
  </div>
  <% end %>
</div>
```

The CSS for this invoice details page has been moved toapp/assets/stylesheets/application.scss

```
.invoice {
  width: 700px;
  max-width: 700px;
  border: 1px solid grey;
  margin: 50px;
  padding: 50px;

  h1 {
    text-align: center;
    margin-bottom: 100px;
  }
  .notes {
    margin-top: 100px;
  }

  table {
    width: 90%;
    text-align: left;
  }
  th {
    padding-bottom: 15px;
  }

  .total td {
    font-size: 20px;
    font-weight: bold;
    padding-top: 25px;
  }
}
```

Then when you click on an invoice in the main listing page, you'll see the details:



Envato Invoice

To: Pedro Alonso

Date: April 01, 2016

Description	Price
Tuts+ Subscription April 2016	$15.00
Ruby eBook	$9.90

Total: $24.90

At this point, we are ready to add the functionality to our rails application to view or download the invoices in PDF.

InvoicePdf Class to Handle PDF Rendering

In order to render invoices from our rails app to PDF, we need to add three gems to the Gemfile: PDFKit, render_anywhere, and wkhtmltopdf-binary. By default, rails only allows you to render templates from a controller, but by using `render_anywhere`, we can render a template from a model or background job.

```
gem 'pdfkit'
gem 'render_anywhere'
gem 'wkhtmltopdf-binary'
```

In order not to pollute our controllers with too much logic, I'm going to create a new `InvoicePdf` class inside the `app/models` folder to wrap the logic to generate the PDF.

```
require "render_anywhere"

class InvoicePdf
  include RenderAnywhere

  def initialize(invoice)
    @invoice = invoice
  end

  def to_pdf
    kit = PDFKit.new(as_html, page_size: 'A4')
    kit.to_file("#{Rails.root}/public/invoice.pdf")
  end

  def filename
    "Invoice #{invoice.id}.pdf"
  end

  private

    attr_reader :invoice

    def as_html
      render template: "invoices/pdf", layout: "invoice_pdf", locals: { invoice: invoice }
    end
end
```

This class is just taking the invoice to render as a parameter on the class constructor. The private methoc `as_html`

is reading the view template `invoices/pdf` and `layout_pdf` that we are using to generate the HTML that we need to render as PDF. Lastly, the method `to_pdf` is using PDFKit to save the PDF file in the rails public folder.

Possibly you want to generate a dynamic name in your real application so the PDF file doesn't get overwritten by accident. You might want to store the file on AWS S3 or a private folder too, but that is outside of the scope of this tutorial.

/app/views/invoices/pdf.html.erb

```
<div class="invoice">
  <h1>Envato Invoice</h1>

  <h3>To: <%= invoice.client %></h3>
  <h3>Date: <%= invoice.date.strftime("%B %d, %Y") %></h3>

  <table>
    <thead>
        <tr>
          <th>Description</th>
          <th>Price</th>
        </tr>
      </thead>
      <tbody>
        <% invoice.line_items.each do |line_item| %>
          <tr>
            <td><%= line_item.description %></td>
            <td><%= number_to_currency(line_item.price) %></td>
          </tr>
        <% end %>
        <tr class="total">
          <td style="text-align: right">Total: </td>
          <td><%= number_to_currency(invoice.total) %></span></td>
        </tr>
      </tbody>
  </table>

  <% if invoice.notes %>
  <div class="notes">
    <p><strong>Notes:</strong> <%= invoice.notes %></p>
  </div>
  <% end %>
</div>
```

/app/views/layouts/invoice_pdf.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>Envato Invoices</title>
  <style>
    <%= Rails.application.assets.find_asset('application.scss').to_s %>
  </style>
</head>
<body>
  <%= yield %>
</body>
</html>
```

One thing to notice in this layout file is that we are rendering the styles in the layout. WkHtmlToPdf does work better if we render the styles this way.

DownloadsController to Render the PDF Invoice

At this point we need a route and controller that call the class `InvoicePdf` to send the PDF file to the browser, so edit `config/routes.rb` to add a nested resource:

```ruby
Rails.application.routes.draw do
  root to: "invoices#index"

  resources :invoices, only: [:index, :show] do
    resource :download, only: [:show]
  end
end
```

If we run `rake routes`, we see the list of routes available in the application:

```
          Prefix Verb URI Pattern                                  Controller#Action
            root GET  /                                            invoices#index
invoice_download GET  /invoices/:invoice_id/download(.:format) downloads#show
        invoices GET  /invoices(.:format)                          invoices#index
         invoice GET  /invoices/:id(.:format)                      invoices#show
```

Add `app/controllers/downloads_controller.rb`:

```ruby
class DownloadsController < ApplicationController

  def show
    respond_to do |format|
      format.pdf { send_invoice_pdf }
    end
  end

  private

  def invoice_pdf
    invoice = Invoice.find(params[:invoice_id])
    InvoicePdf.new(invoice)
  end

  def send_invoice_pdf
    send_file invoice_pdf.to_pdf,
      filename: invoice_pdf.filename,
      type: "application/pdf",
      disposition: "inline"
  end
end
```

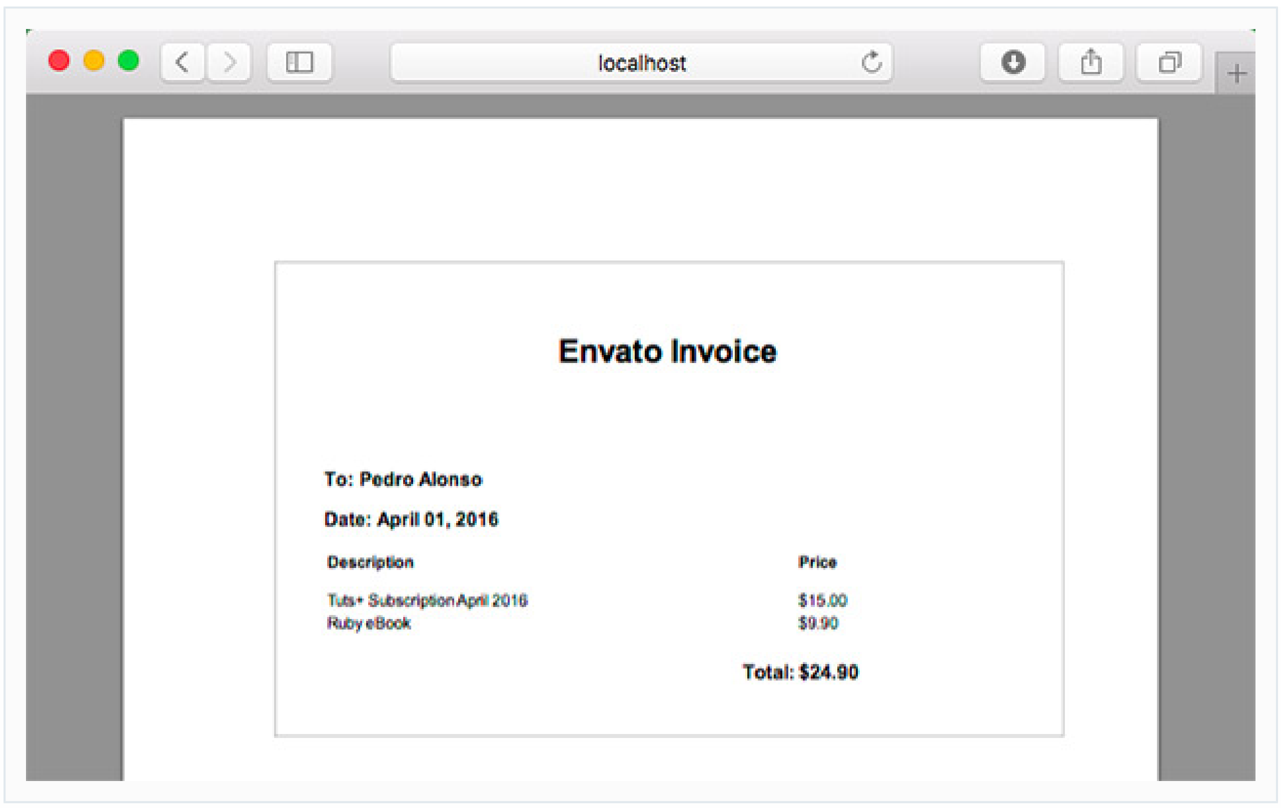
As you can see, when the request is asking for a PDF file, the method `send_invoice_pdf` is processing the request. The method `invoice_pdf` is just finding the invoice from the database by id, and creating an instance of InvoicePdf. Then `send_invoice_pdf` is just calling the method `to_pdf`, to send the generated PDF file to the browser.

One thing to note is that we are passing the parameter `disposition: "inline"` to `send_file`. This parameter is sending the file to the browser, and it will be displayed. If you want to force the file to be downloaded, then you'll need to pass `disposition: "attachment"` instead.

Add a download button to your invoice show template `app/views/invoices/show.html.erb`:

```erb
<%= link_to "Download PDF",
  invoice_download_path(@invoice, format: "pdf"),
  target: "_blank",
  class: "download" %>
```

Run the application, navigate to the invoice details, click on download, and a new tab will open displaying the PDF Invoice.



Envato Invoice

To: Pedro Alonso

Date: April 01, 2016

Description	Price
Tuts+ Subscription April 2016	$15.00
Ruby eBook	$9.90

Total: $24.90

Render PDF as HTML in Development

When you're working on the markup for your PDF, having to generate a PDF every time you want to test a change can be slow sometimes. For this reason, being able to view the HTML that will be converted to PDF as plain HTML can be really useful. We only need to edit `/app/controllers/downloads_controller.rb`.

```ruby
class DownloadsController < ApplicationController

  def show
    respond_to do |format|
      format.pdf { send_invoice_pdf }

      if Rails.env.development?
        format.html { render_sample_html }
      end
    end
  end

  private

  def invoice
    Invoice.find(params[:invoice_id])
  end

  def invoice_pdf
    InvoicePdf.new(invoice)
  end

  def send_invoice_pdf
    send_file invoice_pdf.to_pdf,
      filename: invoice_pdf.filename,
      type: "application/pdf",
      disposition: "inline"
  end

  def render_sample_html
    render template: "invoices/pdf", layout: "invoice_pdf", locals: { invoice: invoice }
  end
end
```

Now the `show` method is also responding for HTML requests in development mode. The route for a PDF invoice would be something like http://localhost:3000/invoices/1/download.pdf. If you change it to http://localhost:3000/invoices/1/download.html, you will see the invoice in HTML using the markup that is used to generate the PDF.

Given the code above, generating PDF files using Ruby on Rails is straightforward assuming you're familiar with the Ruby language and the Rails framework. Perhaps the nicest aspect of the entire process is that you don't have to learn any new markup languages or specifics about PDF generation.

I hope this tutorial has proved useful. Please leave any questions, comments, and feedback in the comments and I'll be happy to follow-up.

Pedro Alonso
Technology Consultant, London UK

Pedro Alonso is an experienced technology consultant. He enjoys travel, reading and photography in his free time.

pedropaf