Uygulama

Hedef: MVC tabanlı Restful mimariye sahip, Bootstrap/Jquery entegre edilmiş bir uygulama gerçekleştirmek. Uygulamada hedef, kitap adı, özeti ve resmi girilebilen bir kitap uygulaması ara yüzü ve yazar girilebilen bir yazar ara yüzü oluşturmak. Uygulama ya devise gem kullanarak kullanıcı yönetimi ekleyeceğiz ve bir sonraki hafta Yazar-User arasında 1-1, Yazar-Kitap arasında ve User-Kitap arasında 1-Many ilişki kuracağız.

Ana adımlar

1) Uygulama template'i oluşturma
2) Scaffold kullanarak kitap kaynağı için MVC dosyalarını üretmek
3) Root sayfası verme
4) Alias verme
5) Bootstrap/jquery entegre etme
6) Paperclip gem ile kitap uygulamasına resim eklenmesi
7) Uygulamaya ara yüz oluşturma

Bölüm sonunda: Tamamen farklı bir uygulama için (örneğin stok takibi için ürün adı, miktarı, fiyatı, resmi girilen bir uygulamayı baştan sona yapabilmeniz gerekmektedir)

Adımlar.

1) Uygulama şablonu oluşturma

   rails new uygulama
   cd uygulama

2) Scaffold ile çalışan bir /kitaps restful mvc uygulaması üretmek (Bu scaffold ile aynı zamanda Kitap isimli bir tablo oluştuğunu hatırlayalım)

   rails g scaffold kitap isim ozet:text sayfa:integer

   rake db:migrate

   rails s

3) Root (açılış sayfası verme)
   #config/routes.rb #ROR ortamında tüm pathler bu dosyadan ayarlanır
   #aşağıdaki satırı üstteki dosyaya ekleyerek kitaps controllerinin (app/controller/kitaps_controller) index #methodunu açılış sayfası yap demiş oluyoruz

   root "kitaps#index"

4) Alias verme (Linkleri takma isimle yönlendirmek)

   #config/routes.rb içinde ki resources :kitaps satırı aşağıdaki gibi düzenlenecek

   resources :kitaps, :path=>"kitaplar"

Diğer adımlara başlamadan önce herhangi bir gem ne işe yarar, nasıl kurulur/konfigure edilir, tüm gemler için bir arama motoru ve yönetim sistemi olan rubygems.org adresinden bakabilirsiniz.

5) Uygulamaya bootstrap ve jquery desteği eklenmesi

#Gemfile a aşağıdaki kütüphaneler eklenecek

gem 'bootstrap'

gem 'jquery-rails'

#Komut satırında, uygulama klasörü içinde aşağıdaki komut çalıştırılacak (Gemfile eklenen yeni kütüphanelerin indirilme ve kurma işlemi gerçekleştirilir)

bundle

#app/assets/stylesheets/application.css dosyası

#application.scss olarak değiştirilir

#ardından application.scss dosyasında en alta aşağıdaki satır eklenir

@import "bootstrap";

#app/assets/javascripts/application.js

#dosyasına aşağıdaki iki satır eklenir //=require . satırından önce olacak şekilde eklenecek

//= require jquery

//= require bootstrap

Not 1: Bootstrap/jquery kurulumundan sonra Puma sunucusu (rails s) açık ise Ctrl+C ile kapatarak yeniden başlatmanız gerekmektedir.

Not 2: Bootstrap/Jquery çalıştığını denetlemek için herhangi bir bootstrap öğesi (grid, jumbotron vb) sayfaya koyarak deneyiniz.

6) Paperclip gem ile kitap modeline resim eklemek

#Gemfile a aşağıdaki satır eklenecek ardından terminal ortamında uygulama klasörü içinde terminal ortamında bundle yapılacak

gem "paperclip"

bundle

Aşağıdaki satır (terminal ortamında) paperclip kullanarak Kitap modeline resim isminde bir alan ekler

rails g paperclip kitap resim

rake db:migrate #veritabanında ki değişikliklerin gerçekleştirilmesi

#app/controllers/kitaps_controller permit listesi güncellenecek

#kitap uygulaması en başta scaffold ile oluşturulmuştu ve permit (izin) listesinde sadece isim ve özet alanları vardı

params.require(:kitap).permit(:isim, :ozet, :resim)

#app/views/kitaps/_form.erb.html dosyasına dosya yüklenmesi için gerekli field eklenecek

** **

<%= form.label :resim %>

<%= form.file_field :resim %>

** **

#app/models/kitap.rb dosyasına resim dosyalarını eklemesi için

has_attached_file :resim, styles: { medium: "300x300>", thumb: "100x100>" }, default_url: "/images/:style/missing.png"

validates_attachment_content_type :resim, content_type: /\Aimage\/.*\z/

#resimleri göstermek için görünüm dosyaları editlenecek

#app/views/kitaps/index.html.erb

<%= image_tag kitap.resim.url(:thumb) if kitap.resim? %>

#app/views/kitaps/show.html.erb

<%= image_tag @kitap.resim.url(:medium) if @kitap.resim? %>

7) Arayüz için bootsnipp ya da w3schools gibi kaynaklardan ücretsiz şablonlar seçebilirsiniz, şablon seçiminde kullanılan bootstrap ve jquery versiyonlarının sizin uygulamanız ile uyumlu çalışması ve farklı tarayıcılarda denenmesi gerekmektedir.

Biz 5.adımda bootstrap kurarken Gemfile a gem 'bootstrap' satırı ile ekleme yaptık, burada versiyon belirtmediğimiz için default olarak bootstrap 4 versiyonu yüklenir. Bu yüzden template bootstrap 4 yada uyumlu bir versiyon olması gerekmektedir.

Kullanılan örnek template linki https://bootsnipp.com/snippets/1eyWZ

Template nasıl uygulamaya eklenir.

Bootsnipp sitesindeki şablonlar ***Preview, Html, CSS, JS*** kodlarını içermektedir.

1) Html kodları uygulama layout dosyasına ***app/views/layouts/application.html.erb*** kopyalanır. Yalnız üstteki meta taglar kopyalanmasına gerek yoktur. Yani yukarıdaki örnek için 6.satırından itibaren bizim application.html.erb dosyasındaki body içine kopyalıyoruz.

   Not: application.html.erb dosyasındaki **<%=yield%>** satırı silinmemelidir.

2) CSS kodları ***app/assets/stylesheets/application.scss*** dosyasına eklenir
3) JS kodları ***app/assets/javascripts/application.js*** dosyasına eklenir
4) Kullanılan şablon da ki html içerikler düzenlemek (gereksiz divleri kaldırmak, İngilizce içerikleri temizlemek/değiştirmek vb…) ve uygulamaya uygun hale getirmek

Üstteki 7 adımı tamamlayarak, aşağıdaki gibi ekran görüntülerinde olduğu gibi grid sistemide entegre ederek uygulamayı çalışır hale getirin.

7.1) Navbar ekleme, aşağıdaki kodlar için **app/views altında _navbar.html.erb** dosyası oluşturarak kaydedin, burası bizim menümüz, linkleri kendimize göre değiştireceğiz.

```
      <div id="wrapper" class="animate">
      <nav class="navbar header-top fixed-top navbar-expand-lg navbar-dark bg-dark">
        <span class="navbar-toggler-icon leftmenutrigger"></span>
        <a class="navbar-brand" href="#">LOGO</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarText" aria-controls="navbarText"
          aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarText">
          <ul class="navbar-nav animate side-nav">
            <li class="nav-item">
              <a class="nav-link" href="#">Home
                <span class="sr-only">(current)</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Side Menu Items</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Pricing</a>
            </li>
          </ul>
          <ul class="navbar-nav ml-md-auto d-md-flex">
            <li class="nav-item">
              <a class="nav-link" href="#">Home
                <span class="sr-only">(current)</span>
              </a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Top Menu Items</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Pricing</a>
            </li>
          </ul>
        </div>
      </nav>
```

7.2) views/layout/application.html.erb son hali aşağıdaki gibi olacak

```
  <body>
    <%=render "shared/navbar"%>
    <%= yield %>
  </body>
```

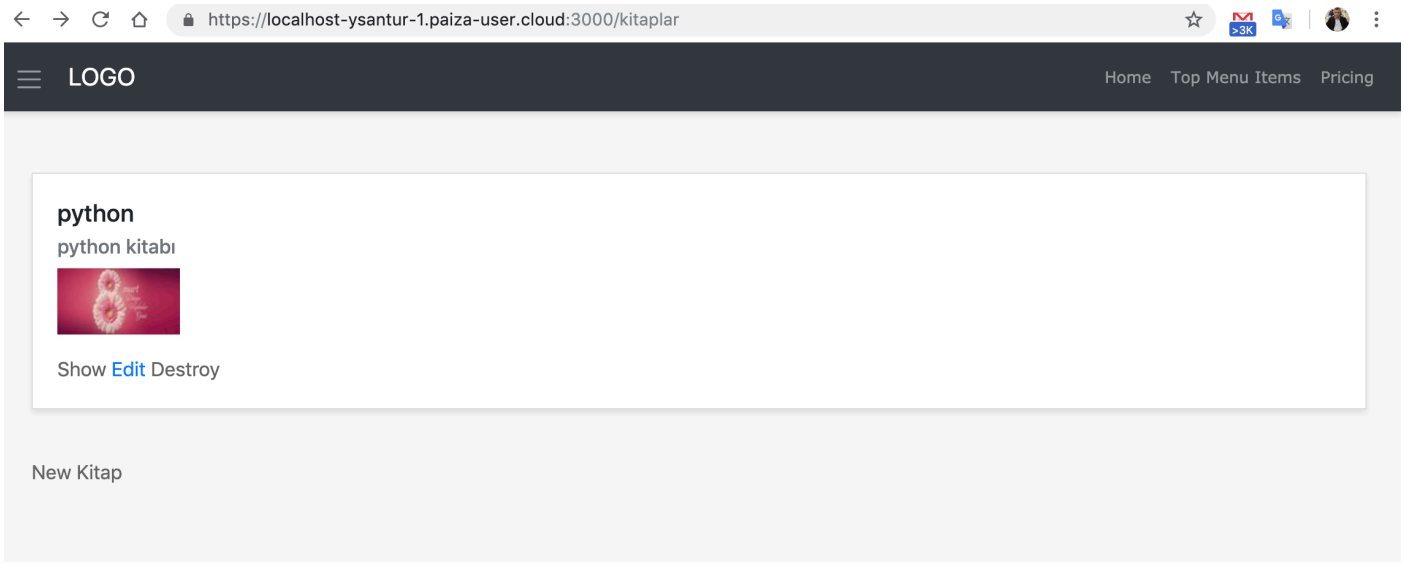
7.3) template i kendi uygulamamız olan kitapların index sayfası için uygun hale getirince **app/views/kitaps/index.html.erb** son hali aşağıdaki gibi olacak

```
<p id="notice"><%= notice %></p>

  <% @kitaps.each do |kitap| %>
    <div class="container-fluid">
         <div class="row">
    <div class="col">
      <div class="card">
        <div class="card-body">
          <h5 class="card-title"> <%= kitap.isim %> </h5>
          <h6 class="card-subtitle mb-2 text-muted"><%= kitap.ozet %></h6>
          <p class="card-text"><%= image_tag kitap.resim.url(:thumb) if kitap.resim? %></p>
          <%= link_to 'Show', kitap %>
          <%= link_to 'Edit', edit_kitap_path(kitap) %>
          <%= link_to 'Destroy', kitap, method: :delete, data: { confirm: 'Are you sure?' } %>
        </div>
      </div>
    </div>
  </div>
  <% end %>

<%= link_to 'New Kitap', new_kitap_path %>
```

Görsel 1 => Uygulama ana sayfa görünümü (Grid sistem yerleştirilmiş template uygulamaya uyarlanmış ve Türkçeleştirilmiş)



8) Scaffold ile çalışan bir /yazars restful mvc uygulaması üretmek (Bu scaffold ile aynı zamanda Kitap isimli bir tablo oluştuğunu hatırlayalım)

   rails g scaffold yazar isim

   rake db:migrate

   rails s

9) Yazars için Alias verme (Linkleri takma isimle yönlendirmek)

   #config/routes.rb içinde ki resources :yazars satırı aşağıdaki gibi düzenlenecek

   resources :yazars, :path=>"yazarlar"

Yazar uygulaması (/yazarlar) içinde ara yüzleri düzenleyin

DEVISE GEM İLE KULLANICI YÖNETİMİNİN UYGULAMAYA ENTEGRE EDİLMESİ

10) #Gemfile a aşağıdaki satır eklenecek ardından terminal ortamında uygulama klasörü içinde terminal ortamında bundle yapılacak

gem "devise"

bundle

11) Terminal ortamında, uygulama klasörü içinde sırası ile aşağıdaki komutlar çalıştırılarak devise geminin kurulumu yapılmış olacak. Bu işlem sonucunda email, password alanlarından oluşan **User** isimli bir modele sahip olmuş olacağız ***rails console*** ortamında bu tabloyu inceleyebilirsiniz.

rails g devise:install

rails g devise user

rails g devise:views

rake db:migrate

12) Kullanıcı yönetimi için navbarı bir **partial** olarak ekleyeceğiz. Sayfa html kodlarının tamamının ***application.html.erb*** içinde yazılmasından ziyade bir kısmı başka bir isimle kaydedilmesine partial adı verilir, partial isimleri alt çizgi (_) ile başlamak zorundadır!

12.1) Aşağıdaki kodları **app/views/shared/_navbar.html.erb** isimli menü olarak oluşturduğumuz dosyaya kullanıcı yönetimi için aşağıdaki kodu ekleyin

```
<p class="navbar-text pull-right"> <% if user_signed_in? %>

Logged in as <strong><%= current_user.email %></strong>.
<%= link_to 'Edit profile', edit_user_registration_path, :class => 'navbar-link' %> |
<%= link_to "Logout", destroy_user_session_path, method: :delete, :class => 'navbar-link' %>

<% else %>
<%= link_to "Sign up", new_user_registration_path, :class => 'navbar-link' %> | <%= link_to
"Login", new_user_session_path, :class => 'navbar-link' %>

<% end %> </p>
```

Notlar

current_user Sistemideki aktif kullanıcıyı simgeler

if user_signed_in? Sistemde giriş yapmış bir kullanıcı varmı True/False döner

Sign_up/login/Edit/Logout linklerine dikkat ediniz. (Ezberlemek zorunda değilsiniz terminal ortamında **rake routes** komutu size tüm linkleri verir)

13) Son olarak **app/controllers/kitaps_controller.rb** dosyası içinde aşağıdaki satırı yazarak controllere aksiyondan önce kullanıcı giriş şartı ara demiş oluyoruz.

before_action :authenticate_user!, only: [:new, :edit, :destroy] #yeni/güncelleme/silme için üyelik şartı ara

veya

before_action :authenticate_user!, except: [:index, :show] #index ve Show hariç üyelik şartı ara

Notlar

1) Devise gem i ile ilgili tüm görünümler (kayıt/login/doğrulama vb…) aşağıdaki dizindedir. Grid sistemi burada ki .html.erb dosyalarına da uygulamanız gerekmektedir.

app/views/devise

2) Devise ile ilgili ayarlar app/config/initializers/devise.rb dosyası içinde bulunur. Örneğin aşağıdaki satır şifrelerin en az, en çok kaç karakter olacağını belirler, 2.satır ise şifre politikası için regex paterni tanımlar

config.password_length = 6..128

config.email_regexp= /\A[^@\s]+@[^@\s]+\z/

14) User- kitap arasında 1-Many ilişki kurma (Not: İlişki kurmadan önce oluşturulan kitap tablosundaki kayıtlarda daha önce user bilgisi olmadığı için hata almamak adına kitapları silerek tabloyu boşaltın)

rails g migration AddUserToKitap user:references
rake db:migrate

belongs_to :user #app/models/kitap.rb dosyasına eklenecek
has_many :kitaps #app/models/user.rb dosyasına eklenecek, s takısına dikkat

14.1) Kitap oluştururken kullanıcı bilgisini çekebilmek için **app/controllers/kitaps_controller.rb** create methodu içinde @kitap = Kitap.new(kitap_params) satırından hemen sonraya aşağıdaki kodu eklememiz lazım!

@kitap.user=current_user

14.2) Kullanıcı bilgisini göstermek için **app/views/kitaps/index.html.erb** dosyasına eklenebilir.

<%=kitap.user.email%>

15) Yetki denetimi, kitabı oluşturan kullanıcı kim ise silme/değiştirme işlemini de o yapabilmeli!

15.1) Aşağıdaki gibi bir kontrol methodunu **app/controllers/kitaps_controller** dosyasına private kısmından sonra ekleyin

```
def kontrol
  if @kitap.user!=current_user
    redirect_to root_url, notice: 'Yetkiniz yok'
  end
end
```

15.2) Silme ve değiştirme aksiyonlarından önce bu methodun çağrılabilmesi için aşağıdaki filtre satırını aynı dosyanın başına eklememiz lazım.

```
before_action :kontrol, only: [:edit, :update, :destroy]
```

Not: kontrol methoduna dikkat ederseniz @kitap değişkeninin bilinmesi lazım bu yüzden **before_action :set_kitap** methoduna bu kontrol ünde eklenmesi lazım, yani controller ın filtre methodlarının son hali aşağıdaki gibi olacak.

```
before_action :set_kitap, only: [:show, :edit, :update, :destroy, :kontrol]
before_action :authenticate_user!, except: [:index, :show]
before_action :kontrol, only: [:edit, :update, :destroy]
```

16) Ajax entegrasyonu, öncelikle bir Like modeli oluşturalım bu model her kitaba ait beğeni sayısını tutsun aradaki ilişki 1-1 ilişkisi olacak (Not: öncelikle oluşturulan kitapları yine silin)

rails g model like kitap:references sayi:integer
rake db:migrate

```
has_one :like, :dependent => :destroy   #kitap modelinde
belongs_to :kitap #like modelinde
```

Not: depented kısmı kayıt silindiğinde ilişkili kayıtlar silinsin diye

16.1) Kitap oluşturulduğunda beğeni sayısını 0 yapmak için, **app/controllers/kitaps_controller.**rb create methoduna aşağıdaki kodu ekleyebiliriz.

```
Like.create(kitap: @kitap, sayi:0)
```

16.2) Kitabın beğeni sayısını show sayfasında göstermek için **app/views/kitaps/show.html.erb** dosyasına aşağıdaki kod eklenebilir.

```
<p>
  <strong>Beğeni sayısı:</strong>
  <%= link_to " Beğen ", kitap_like_path(@kitap), remote: :true, method: :put %>
</p>

  <div id="like">
    <%= @kitap.like.sayi %>
  </div>
```

16.3) Ajax ile kitabın beğeni sayısını 1 arttırmak için, önce **config/routes.rb** dosyasına ek methodun yolunu tanımlamamız gerekiyor.

```
resources :kitaps, :path =>"kitaplar" do
      put "like", to: "kitaps#like"
  end
```

16.4) Kitap Show sayfasında

```
<%= link_to " Beğen ", kitap_like_path(@kitap), remote: :true, method: :put %>
```

16.5) Beğeni sayısını arttırmak için app/controller/kitaps_controller.rb dosyasına like methodunu ekleyelim

```
def like
   @like=Like.find_by(kitap:params[:kitap_id])
   @like.sayi=@like.sayi+1
   @like.save
   respond_to do |format|
      format.js {@like}
      format.json
   end
end
```

16.6) Ajax cevabı görmek için app/views/kitaps/like.js.erb dosyası

```
$("#like").html("beğendin"+"<%=@like.sayi%>")
```