# Görünümler ve Şablonlar

**Gerekli tahmini süre:** 40 dakika

Bu laboratuvar çalışmasında, görünümler ve şablonlar oluşturmayı, nesneleri güncellemek için görünümleri kullanmayı ve kullanıcıları diğer sayfalara yönlendirmeyi, ayrıca şablonlarınızı stilize etmek için CSS kullanmayı öğreneceksiniz.

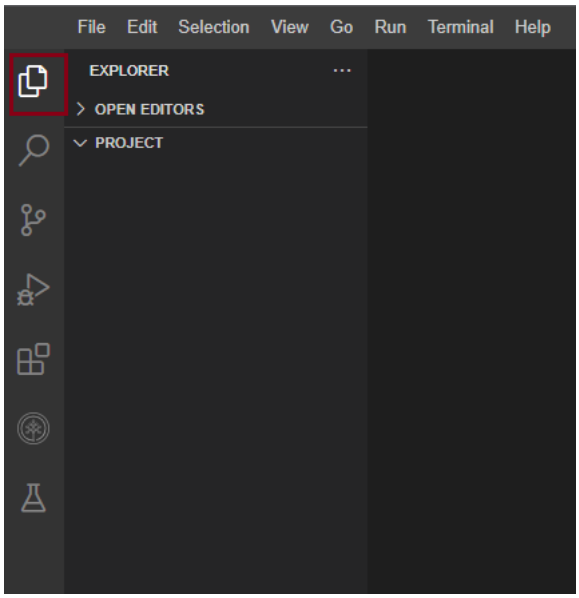
## Öğrenme Hedefleri

- HTTP isteklerini işlemek ve HTTP yanıtları döndürmek için işlev tabanlı görünümler oluşturun
- HTML sayfalarını oluşturmak için şablonlar oluşturun

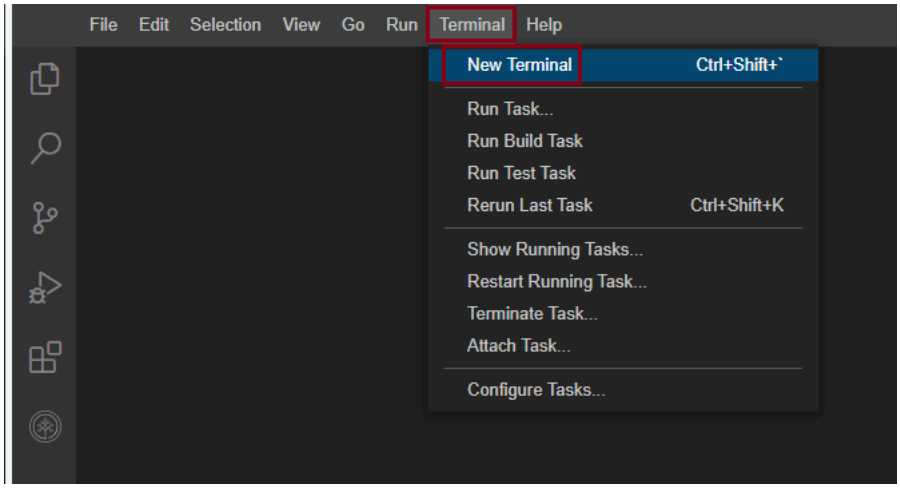
# Cloud IDE'de dosyalarla çalışma

Cloud IDE'ye yeniyseniz, bu bölüm projenizin bir parçası olan dosyaları nasıl oluşturacağınızı ve düzenleyeceğinizi gösterecektir.

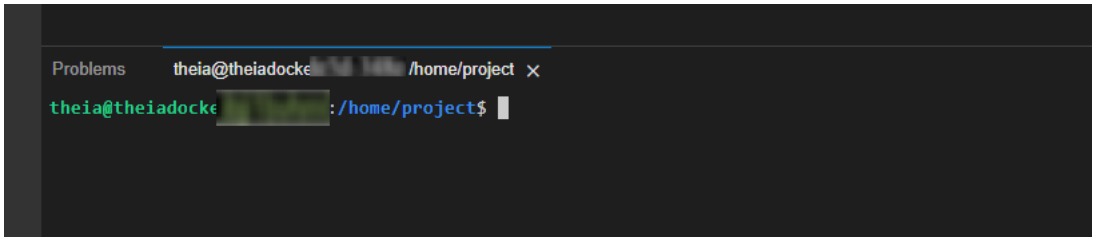
Cloud IDE içinde dosyalarınızı ve dizinlerinizi görüntülemek için bu dosya simgesine tıklayın.




`Terminal`'e tıklayın, ardından `Yeni Terminal`'e tıklayın.




Bu, komutlarınızı çalıştırabileceğiniz yeni bir terminal açacaktır.



# Laboratuvarda Ele Alınan Kavramlar

1. `SQLite`: Yapılandırılmış verileri saklamak, yönetmek ve almak için SQL tabanlı bir arayüz sunan hafif bir ilişkisel veritabanı yönetim sistemi (RDBMS).
2. `List view`: Birden fazla kaydı tablo veya liste formatında almak ve sunmak için kullanışlı bir yol sağlayan sınıf tabanlı bir görünüm.
3. `HttpResponseRedirect`: Bir isteği işledikten sonra kullanıcıları farklı bir sayfaya veya URL'ye yönlendirmek için kullanılan bir sınıf.
4. `reverse()`: Django tarafından sağlanan, bir URL deseninin adını argüman olarak alan ve karşılık gelen URL'yi bir dize olarak döndüren bir yardımcı fonksiyon.
5. `CSS container class`: Web sayfası içeriği için duyarlı sabit genişlikte bir konteyner oluşturur ve farklı ekran boyutları ve cihazlar arasında içeriğiniz için tutarlı bir düzen ve boşluk sağlar.
6. `CSS card class`: Bilgileri, resimleri ve eylemleri kompakt ve düzenli bir şekilde sunmak için "kart" adı verilen esnek ve stilize bir konteyner oluşturur.

# Bir `onlinecourse` Uygulama Şablonu ve Veritabanı İçe Aktarma

Eğer terminal açık değilse, `Terminal > Yeni Terminal` yolunu izleyin ve mevcut Theia dizininizin `/home/project` olduğundan emin olun.

- Bu laboratuvar için bir kod şablonu indirmek üzere aşağıdaki komut satırlarını çalıştırın.

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m4_django_app/lab3_template.zip"
unzip lab3_template.zip
rm lab3_template.zip
```

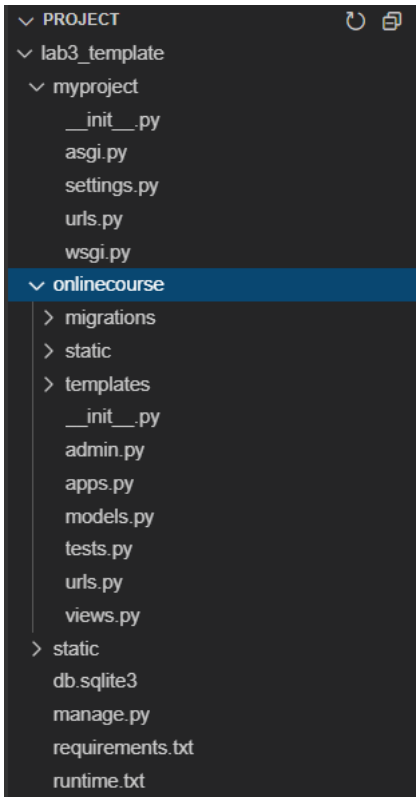
- `cd` proje klasörüne

```
cd lab3_template
```

- Sonra, bu laboratuvar için gerekli paketleri yükleyin:

```
pip install --upgrade distro-info
pip3 install --upgrade pip==23.2.1
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
pip install -U -r requirements.txt
```

Django projeniz aşağıdaki gibi görünmelidir:




`myproject/settings.py` dosyasını açın ve `DATABASES` bölümünü bulun. Bu laboratuvar için SQLite kullandığımızı göreceksiniz.

**SQLite, bazı kurs verileri önceden yüklenmiş dosya tabanlı bir gömülü veritabanıdır. Bu nedenle verileri kendiniz doldurmak için Model API'lerini veya yönetim sitesini kullanmanıza gerek yoktur ve görünüm ve şablon oluşturmaya odaklanabilirsiniz.**

Sonraki adımda `onlinecourse` uygulaması için modelleri etkinleştirin.

- Gerekli tabloları oluşturmak için ilk kez çalıştırma sırasında göç işlemleri gerçekleştirmeniz gerekiyor.

  ```
  python3 manage.py makemigrations
  ```

- Ve `onlinecourse` uygulaması için modelleri etkinleştirmek amacıyla göç işlemini çalıştırın.

  ```
  python3 manage.py migrate
  ```

# Popüler Kurs Listesi Görünümü Oluştur

Artık veritabanından veri okumak için modellerdeki yöntemleri kullanarak görünümler yazmaya ve dinamik HTML sayfalarını oluşturmak için verilere şablonlar eklemeye başlayabiliriz.

Bu adımda, `onlinecourse` uygulamasının indeks veya ana sayfası olarak bir popüler kurs listesi görünümü oluşturacağız.

Öncelikle bir kurs listesi şablonu oluşturalım.

- `onlinecourse/templates/onlinecourse/course_list.html` dosyasını açın ve aşağıdaki kod parçasını `<!-- Add your template there -->` kısmının altına yapıştırın.

  ```
  {% if course_list %}
      <ul>
      {% for course in course_list %}
  ```

```
            <div>
                <img src="{{ MEDIA_URL }}/{{ course.image }}" width="360" height="360" >
                <h1><b>{{ course.name }}</b></h1>
            </div>
        {% endfor %}
        </ul>
    {% else %}
    <p>No courses are available.</p>
    {% endif %}
```

Yukarıdaki kod parçasında, önce `{% if course_list %}` ifadesini ekleyerek `course_list`'in index görünümünden gönderilen bağlamda mevcut olup olmadığını kontrol ediyoruz.

Eğer mevcutsa, ardından `{% for course in course_list %}` ekleyerek kurs listesini döngüye alıyor ve `course`'un `image`, `name` gibi alanlarını gösteriyoruz.

Sonra, `Course` modelini kullanarak şablona bir kurs listesi sağlamak için bir görünüm oluşturalım.

- `onlinecourse/views.py` dosyasını açın, modellerden en popüler 10 kursu almak için bir görünüm ekleyin.
  Kurs nesneleri `total_enrollment`'a göre azalan sırayla sıralanır ve ilk on nesne en popüler 10 kurs olarak kesilir,

```
Course.objects.order_by('-total_enrollment')[:10]
```

- Ardından, bir `context` sözlük nesnesi oluşturur ve `course_list`'i `context`'e ekleriz.

```
def popular_course_list(request):
    context = {}
    # If the request method is GET
    if request.method == 'GET':
        # Using the objects model manage to read all course list
        # and sort them by total_enrollment descending
        course_list = Course.objects.order_by('-total_enrollment')[:10]
        # Appen the course list as an entry of context dict
        context['course_list'] = course_list
        return render(request, 'onlinecourse/course_list.html', context)
```

Sonra, `popular_course_list` görünümü için bir rota yolu ekleyin.

- `onlinecourse/urls.py` dosyasını açın, `urlpatterns` içinde bir yol girişi ekleyin:

```
path(route='', view=views.popular_course_list, name='popular_course_list'),
```

ve `urlpatterns` listeniz aşağıdaki gibi görünmelidir:

```
urlpatterns = [
    # Add path here
    path(route='', view=views.popular_course_list, name='popular_course_list'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)\
 + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

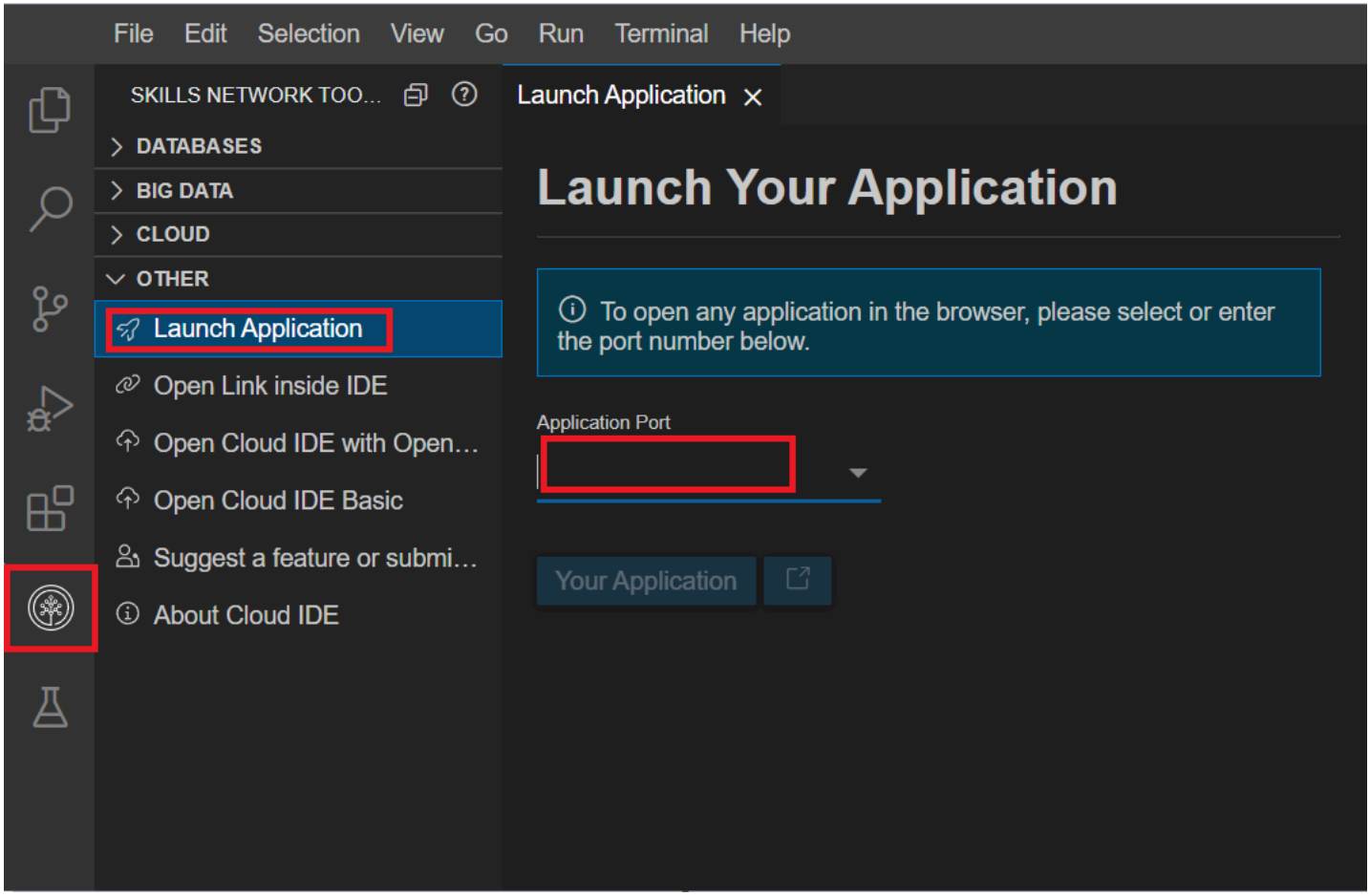
Şimdi en popüler 10 kursu görüntülemek için bir görünüm ve bir şablon oluşturduk.

Görünümü ve şablonu test edelim.

- Sunucuyu başlatın

```
python3 manage.py runserver
```

- Sol taraftaki Beceriler Ağı butonuna tıklayın, bu **"Beceriler Ağı Araç Kutusu"**'nu açacaktır. Ardından **Diğer**'e, sonra da **Uygulamayı Başlat**'a tıklayın. Buradan `8000` portunu girmeli ve başlatmalısınız.




- Yeni bir tarayıcı sekmesi açıldığında, `/onlinecourse` yolunu ekleyin ve tam URL'niz aşağıdaki gibi görünmelidir

```
https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse
```

- Ve `Django'ya Giriş` ve `Python'a Giriş` adlı iki kursla bir kurs listesi görmelisiniz.

Yeni oluşturduğumuz şablon, herhangi bir stil veya biçimlendirme olmadan basit bir HTML şablonudur. Görünümünü sonraki adımlarda iyileştireceğiz.

# Kodlama Pratiği: Kursa Daha Fazla Alan Ekle

Aşağıdaki şablonu `course_list.html` tamamlayarak bir kursa `total_enrollment` ve `description` alanlarını ekleyin.

```
{% if course_list %}
    <ul>
    {% for course in course_list %}
        <div>
            <img src="{{ MEDIA_URL }}/{{ course.image }}" width="360" height="360" >
            <h1><b>{{ course.name }}</b></h1>
            <p><!--<HINT> add a `total_enrollment` variable here --> enrolled</p>
            <p><!--<HINT> add a `description` variable here --></p>
        </div>
    {% endfor %}
    </ul>
{% else %}
<p>No courses are available.</p>
{% endif %}
```

▼ Çözümü görmek için buraya tıklayın

```
{% if course_list %}
    <ul>
    {% for course in course_list %}
        <div>
            <img src="{{ MEDIA_URL }}/{{ course.image }}" width="360" height="360" >
            <h1><b>{{ course.name }}</b></h1>
            <p>{{course.total_enrollment}} enrolled</p>
            <p>{{course.description}}</p>
        </div>
    {% endfor %}
    </ul>
{% else %}
<p>No courses are available.</p>
{% endif %}
```

▼ Details

::page{title=”Bir Kurs Kaydı Görünümü Oluştur”}

Şimdi, bir kursun `total_enrollment` alanını güncellemek için bir görünüm oluşturarak basitleştirilmiş bir kullanıcı kayıt süreci oluşturalım.

- `course_list.html` dosyasını güncelleyerek bir kayıt talebi göndermek için bir form ekleyin.

Bu, `{% url 'onlinecourse:enroll' course.id %}` etiketiyle eşleşen bir güncelleme görünümüne `POST` isteği gönderir.

```
{% if course_list %}
    <ul>
    {% for course in course_list %}
        <div>
            <img src="{{ MEDIA_URL }}/{{ course.image }}" width="360" height="360" >
            <h1><b>{{ course.name }}</b></h1>
            <p>{{course.total_enrollment}} enrolled</p>
            <p>{{ course.description }}</p>
            <form action="{% url 'onlinecourse:enroll' course.id %}" method="post">
                {% csrf_token %}
                <input type="submit" value="Enroll">
            </form>
        </div>
    {% endfor %}
    </ul>
{% else %}
<p>No courses are available.</p>
{% endif %}
```

Bir `enroll` görünümü oluşturun ve `total_enrollment` alanına bir ekleyin

- `onlinecourse/views.py` dosyasını açın, bir `enroll` görünümü ekleyin:

```
def enroll(request, course_id):
    # If request method is POST
    if request.method == 'POST':
        # First try to read the course object
        # If could be found, raise a 404 exception
        course = get_object_or_404(Course, pk=course_id)
        # Increase the enrollment by 1
        course.total_enrollment += 1
        course.save()
        # Return a HTTP response redirecting user to course list view
        return HttpResponseRedirect(reverse(viewname='onlinecourse:popular_course_list'))
```

1. `enroll` görünümü önce `course_id` argümanına göre bir kurs nesnesini okur.

2. Eğer kurs veritabanında yoksa, `404 Not Found` hata durumu kodu ile bir HTTP yanıtı döner.
3. Sonra toplam kaydı bir artırır ve nesneyi günceller.
4. POST verileri ile başarılı bir şekilde işlem yaptıktan sonra her zaman bir `HttpResponseRedirect` döndürmelisiniz.
5. `HttpResponseRedirect`, kullanıcının yönlendirileceği bir URL argümanı alır.
6. Burada `popular_course_list` görünümü için URL'yi oluşturmak üzere `HttpResponseRedirect` içinde `reverse()` fonksiyonunu kullanıyoruz (örn. `https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse/`).

Şu an için, kullanıcıyı `viewname='onlinecourse:popular_course_list'` ile aynı sayfaya yönlendiriyoruz.

- `onlinecourse/urls.py` dosyasını açın ve `enroll` görünümüne bir rota oluşturmak için `path('course/<int:course_id>/enroll/', views.enroll, name='enroll'),` ekleyin.

  ```
  path('course/<int:course_id>/enroll/', views.enroll, name='enroll'),
  ```

- Tüm güncellenmiş dosyaları kaydedin ve kurs listesi sayfasını yenileyin, ardından kurs listesi sayfasında her kursun altında bir `Enroll` butonu görmelisiniz. `Enroll` butonuna tıkladığınızda, sayı 1 artacaktır.

::page{title="Bir Kursun Detay Görünümünü Şablon ile Oluşturma"}

Önceki adımda, bir kullanıcı bir kursa kaydolduktan sonra kurs listesi sayfasına yönlendirilecektir. Bu pratik bir senaryo değil çünkü kullanıcılar kursun detaylarına yönlendirilmelidir, böylece öğrenim materyallerini okuyabilir ve sınavlara girebilirler.

Kullanıcı senaryosunu daha pratik hale getirmek için, bir kurs detay sayfası oluşturacak ve kullanıcıları kaydolduktan sonra detay sayfasına yönlendireceğiz.

Bir kurs detay şablonu oluşturmaya başlayalım.

- `onlinecourse/templates/onlinecourse/course_detail.html` dosyasını açın ve aşağıdaki kod parçasını ekleyin.

  ```
  <div>
       <h2>{{ course.name }}</h2>
       <h5>{{ course.description }}</h5>
  </div>
  ```

Yukarıdaki kod parçası iki değişkeni `{{course.name}}` ve `{{course.description}}` ekler. Bunlar, bir kurs kimliğiyle verilen bir kursun detaylarını temsil etmek için iki HTML başlığına sarılmıştır.

Sonraki adımda, istenen kursu okumak ve `course_detail.html` şablonuna bağlam göndermek için bir görünüm ekleyelim.

- `onlinecourse/views.py` dosyasını açın, bir `course_details` görünümü ekleyin.

  ```
  def course_details(request, course_id):
      context = {}
      if request.method == 'GET':
          try:
              course = Course.objects.get(pk=course_id)
              context['course'] = course
              # Use render() method to generate HTML page by combining
              # template and context
              return render(request, 'onlinecourse/course_detail.html', context)
          except Course.DoesNotExist:
              # If course does not exist, throw a Http404 error
              raise Http404("No course matches the given id.")
  ```

Yukarıdaki kod parçası, belirtilen kurs kimliğine göre belirli bir kursu almaya çalışır ve bunu HTML sayfasını oluşturmak için kullanılacak bağlama ekler.

Eğer kurs bulunamazsa, bir `Http404` istisnası oluşturulacaktır.

Sonraki adımda, `course_details` sayfası için rotayı yapılandırabiliriz.

- `onlinecourse/urls.py` dosyasını açın, `path('course/<int:course_id>/', views.course_details, name='course_details'),` yolunu ekleyin.
- Ardından, `onlinecourse/views.py` dosyasına geri dönün ve `enroll` görünüm fonksiyonunu, kullanıcıyı `course_details` görünümü tarafından oluşturulan web sayfasına yönlendirecek şekilde güncelleyin.

```
return HttpResponseRedirect(reverse(viewname='onlinecourse:course_details', args=(course.id,)))
```

Not edin ki, yönlendirme URL'sinin şöyle görünebilmesi için URL'ye bir `course.id` argümanı ekledik:

```
https://userid-8000.theiadocker-1.proxy.cognitiveclass.ai/onlinecourse/course/1/
```

Kurs detayları için görünüm ve şablon oluşturduk. Şimdi bunları test edelim.

- Kurs listesi sayfasını yenileyebilir ve `enroll` butonuna tıklayabilirsiniz, ardından uygulama, az önce oluşturduğumuz görünüm ve şablon ile oluşturulan kurs detayları sayfasını gösterecektir.

# Kodlama Pratiği: Kurs Detayları Sayfasına Ders Ekleyin

Her kursun `Lesson` ile Birden-Çok ilişkisi vardır. Ön yükleme veritabanında, bir kurs için zaten birkaç ders oluşturduk (daha fazla ders eklemek için admin sitesini kullanabilirsiniz).

- Kurs detayları sayfasında dersleri göstermek için `course_details.html` dosyasına `{{course.description }}` altına aşağıdaki HTML kod parçasını tamamlayıp ekleyin.

```
<h2>Lessons: </h2>
{% for lesson in course.lesson_set.all %}
     <div>
          <h5>Lesson <!--<HINT>add `order` variable  --> : <!--<HINT>add `title` variable  --></h5>
          <p><!--<HINT>add `content` variable  --></p>
     </div>
{% endfor %}
```

▼ Çözümü görmek için buraya tıklayın

`course_detail.html` dosyasını güncelleyin ve aşağıdaki kod parçasını ekleyin.

```
<h2>Lessons: </h2>
{% for lesson in course.lesson_set.all %}
     <div>
         <h5>Lesson {{lesson.order}} : {{lesson.title}}</h5>
         <p>{{lesson.content}}</p>
     </div>
{% endfor %}
```

▼ Details

::page{title="Şablonlara CSS Ekle"}

Oluşturduğumuz kurs listesi ve kurs detayları sayfaları, tamamen HTML sayfaları oldukları için çok ilkel görünüyordu ve herhangi bir CSS stiline sahip değildi.

Şablonlardaki HTML öğelerine bazı CSS stil sınıfları ekleyerek şablonları stilize etmeye çalışalım.

`course.css` adında bir örnek css dosyası sağladık, sadece bunu şablonlarınıza dahil etmeniz ve stil sınıflarını kullanmanız gerekiyor.

- `course_list.html` dosyasını açın, `<head>` öğesine `course.css` statik dosyasının bağlantısını ekleyin:

  ```
  {% load static %}
  <link rel="stylesheet" type="text/css" href="{% static 'onlinecourse/course.css' %}">
  ```

- Ardından, kurs listesini `<div>`'yi ( `{% for course in course_list %}` etiketinin altında) bazı dolgu ekleyen `.container` sınıfı ile stilize edin.

  ```
  <div class="container">
  ...
  </div>
  ```

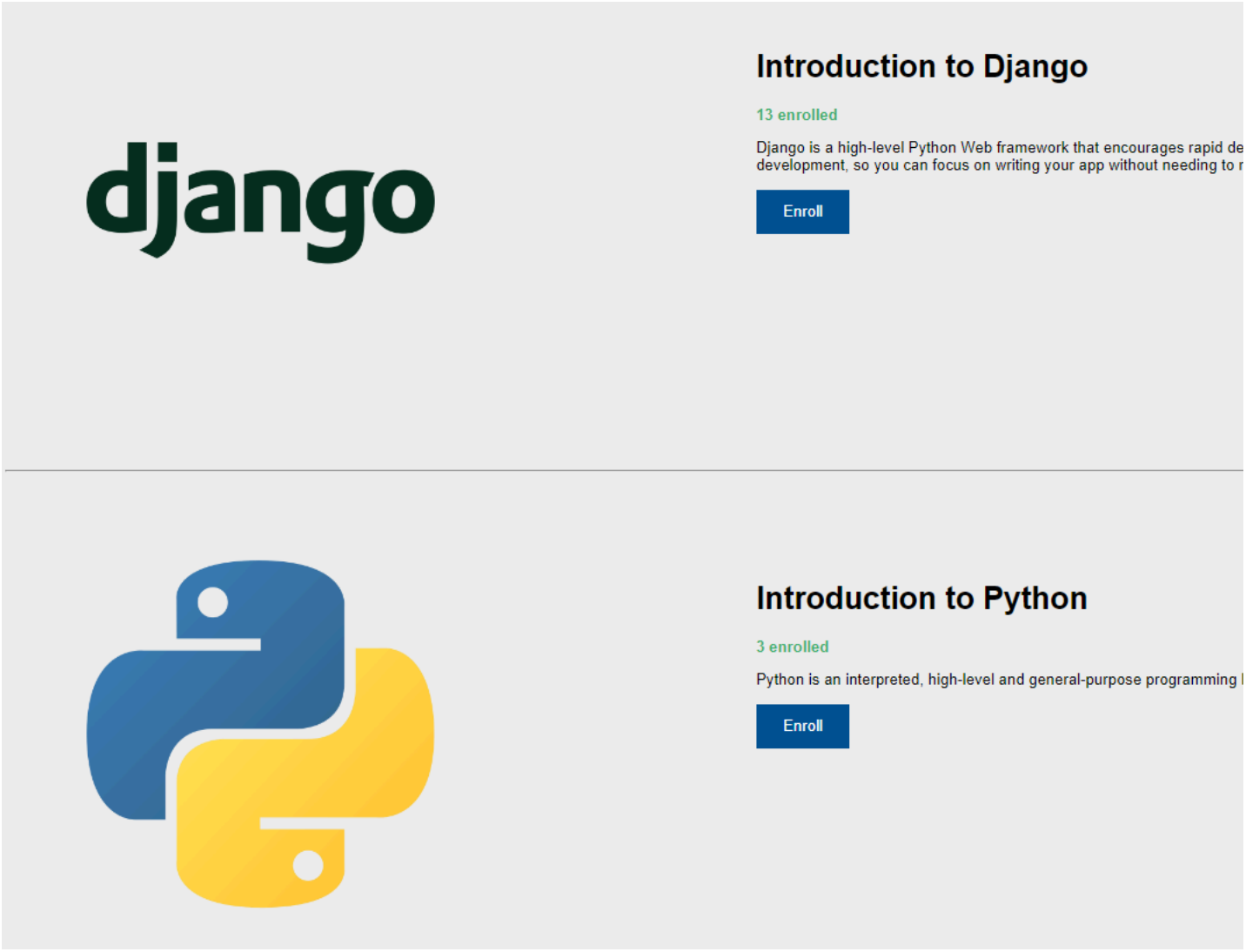
- Kurs alanlarını bir tablo satırı olarak sarmak için bir `<div class="row">` eklemeyi deneyin.
- Kurs resmini `33%` genişlik kaplayacak şekilde bir sütun olarak sarmak için bir `<div class="column-33">` kullanın.
- `name`, `total_enrollment`, `description` ve kayıt formunu sarmak için bir `<div class="column-66">` kullanın.

Kayıt sayılarını yeşil yapmak gibi daha fazla stil eklemekten çekinmeyin.

- Stilize edilmiş kurs listesi aşağıdaki gibi görünmelidir:

```
{% if course_list %}
   <ul>
   {% for course in course_list %}
       <div class="container">
          <div class="row">
            <div class="column-33">
                <img src="{{MEDIA_URL}}/{{ course.image }}" width="360" height="360" >
            </div>
            <div class="column-66">
                <h1 class="xlarge-font"><b>{{ course.name }}</b></h1>
                <p style="color:MediumSeaGreen;"><b>{{course.total_enrollment}} enrolled</b></p>
                <p> {{ course.description }}</p>
                <form action="{% url 'onlinecourse:enroll' course.id %}" method="post">
                {% csrf_token %}
                <input class="button"  type="submit"  value="Enroll">
              </form>
            </div>
          </div>
       </div>
       <hr>
   {% endfor %}
   </ul>
{% else %}
<p>No courses are available.</p>
{% endif %}
```

Kurs listesi sayfasını yenileyin ve sonucu kontrol edin. Kurs listesi sayfanız aşağıdaki gibi görünmelidir:




::page{title="Kodlama Pratiği: Kurs Detayları Sayfasını Stilize Etme"}

`onlinecourse/course.css` dosyasındaki diğer CSS sınıflarıyla oynamak, örneğin `.card` sınıfını kullanarak bir dersi kart gibi görünmesini sağlamak.

- `course_detail.html` dosyasını güncelleyin ve kurs adı, açıklaması ve kurstaki her bir dersi saran `<div>` elemanına `.card` sınıfını ekleyin:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    {% load static %}
    <link rel="stylesheet" type="text/css" href="{% static 'onlinecourse/course.css' %}">
</head>
<body>
    <div <!--<HINT> add `.card` class here -->>
        <h2>{{ course.name }}</h2>
        <h5>{{ course.description }}</h5>
    </div>
    <h2>Lessons: </h2>
    {% for lesson in course.lesson_set.all %}
    <div <!--<HINT> add `.card` class here -->>
        <h5>Lesson {{lesson.order}} : {{lesson.title}}</h5>
        <p>{{lesson.content}}</p>
    </div>
    {% endfor %}
</body>
</html>
```

▼ Çözümü görmek için buraya tıklayın

`course_detail.html` dosyasını güncelleyin ve aşağıdaki kod parçasını ekleyin.

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    {% load static %}
```

```
    <link rel="stylesheet" type="text/css" href="{% static 'onlinecourse/course.css' %}">
</head>
<body>
    <div class="card">
        <h2>{{ course.name }}</h2>
        <h5>{{ course.description }}</h5>
    </div>
    <h2>Lessons: </h2>
    {% for lesson in course.lesson_set.all %}
    <div class="card">
        <h5>Lesson {{lesson.order}} : {{lesson.title}}</h5>
        <p>{{lesson.content}}</p>
    </div>
    {% endfor %}
</body>
</html>
```

▼ Details

::page{title=”Özet”}

Bu laboratuvar çalışmasında, kursların ve kurs detaylarının listesini göstermek için nasıl görünüm ve şablon oluşturacağınızı öğrendiniz. Ayrıca, nesneleri güncellemek ve kullanıcıları diğer sayfalara yönlendirmek için görünümleri nasıl kullanacağınızı da öğrendiniz. Son olarak, şablonlarınızı stilize etmek için CSS kullanmayı öğrendiniz.

# Author(s)

**Yan Luo**