

Django

TomerBu

נושאים להיום:

path segments

עבודה עם מסד נתונים

זרעה של מסד הנתונים

Templates

דברים שכדי לזכור:

התקינה של ספריות:
`pip install django`

`django-admin startproject lec1`

`python manage.py startapp students`

`INSTALLED_APPS+=['students']`

מוסיפים פונקציות ל`views.py`
(request/response)

יצרים קובץ `urls.py` באפליקציה הגדירה שם
ומגדירים שם

בקובץ `urls.py` של האפליקציה הראשית:
עושים `url` של האפליקציה `include`

`python manage.py runserver`

עורכים את קובץ
מוסיפים מודול

`python3 manage.py makemigrations studentes`

`python3 manage.py migrate`

`python3 manage.py createsuperuser`

עורכים את הקובץ `admin`
להוספה מודלים שלנו לממשק ניהול

גישה לממשק ניהול:
`localhost:8000/admin`

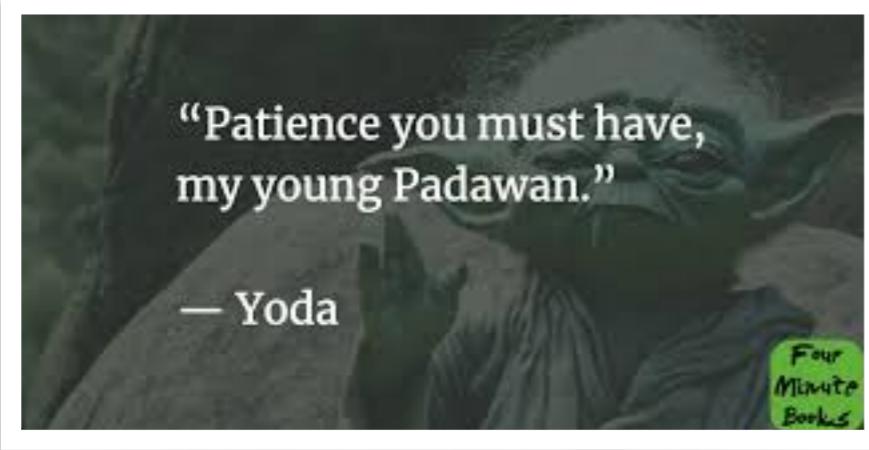
יצירת סביבה וירטואלית:
`python -m venv .venv`

שימוש בסביבה וירטואלית:
`./.venv/script/activate`

שיעור בית:

- 1) צרו פרויקט חדש בערת הפקודה `python3 django-admin startproject lec1_practice`
- 2) צרו 2 אפליקציות בתחום הפרויקט
יש לקרוא לאפליקציה הראשונה `quotes`
ולאפליקציה השנייה לקרוא `forum`
- (לא לשכוח לחבר את האפליקציות לפרויקט הראשי בקובץ `INSTALLED_APPS` בקובץ `py.settings` של האפליקציה הראשית)
- 3) באפליקציה `quotes` יש להגדיר בקובץ `views.py`
את הפעולות הבאות:
 - תחזר טקסט מתאים `quotes`
 - תחזר טקסט מתאים `about`
- 4) באפליקציה `forum` יש להגדיר בקובץ `views.py`
את הפעולות הבאות:
 - תחזר טקסט מתאים `topics`
 - תחזר טקסט מתאים `users`
- 5) יש להגדיר קובץ `urls.py` עבור כל אחת מהאפליקציות
- 6) יש לחבר את הקובץ `urls.py` לאפליקציה הראשית
ע"י עריכה של הקובץ `py.urls` של האפליקציה הראשית.
(זכרו לכל אפליקציה יש `urls.py` משלה).

שיעור בית:



7) באפליקציה quotes יש ליצור רשימה של quotes ולהציג את הרשימה ללקוח בתגובה לבקשת GET /quotes/list/

8) באפליקציה quotes יש ליצור view בשם add_quote שיאפשר להשתמש להוספה quote לרשימה quotes

9) באפליקציה quotes יש להציג את הדסוט quotes בHTML

10) באפליקציה quotes יש לאפשר הוספת quote בעזרת טופס HTML

ריעונות לquotes:

<https://dummyjson.com/quotes/random/10>

שיעור בית:

```
django-admin startproject lec2
python manage.py startapp quotes
python manage.py runserver
```

lec2/quotes/views.py

```
from django.http import HttpRequest, HttpResponse, HttpResponseRedirect
from django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt

# Create your views here.
quotes = [
    'Quote 1',
    'Quote 2',
]

def list(request:HttpRequest):
    return HttpResponse(", ".join(quotes))

@csrf_exempt
def add(request:HttpRequest):
    if request.method == 'POST':
        quote = request.POST.get('quote')
        if quote:
            quotes.append(quote)
            return HttpResponse('Quote added')
        else:
            return HttpResponseRedirect('Quote is empty')
    else:
        return HttpResponseRedirect('POST Only')
```

שיעור בית:

lec2/quotes/urls.py

```
from django.urls import path
from .views import quotes_list, add

urlpatterns = [
    path('list/', quotes_list, name='quotes_list'),
    path('add/', add, name='add'),
]

# the name attribute
# can be used in html template
# {% url 'quotes_list' %}
```

lec2/lec2/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('quotes/', include('quotes.urls')),
]
```

שיעור בית:

`http://127.0.0.1:8000/quotes/list/`

The screenshot shows a POST request in the Postman application. The URL is `http://127.0.0.1:8000/quotes/add/`. The method is set to POST. The body is configured with 'form-data' and contains two fields: 'name' (Text: Sonny) and 'quote' (Text: Quote 4). The 'Headers' tab shows 8 headers. The 'Test Results' tab displays the response: '1 Quote added'.

Overview | **POST** http://127.0.0.1:8000/q | GET http://127.0.0.1:8000/qu | +

HTTP `http://127.0.0.1:8000/quotes/add/`

POST | http://127.0.0.1:8000/quotes/add/ | **http://127.0.0.1:8000/quotes/add/**

Params Authorization Headers (8) **Body** • Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

| | | | |
|-------------------------------------|-------|------|---------|
| <input type="text"/> | name | Text | Sonny |
| <input checked="" type="checkbox"/> | quote | Text | Quote 4 |

Body Cookies Headers (8) Test Results |

HTML | Preview | Visualize |

1 Quote added

שיעור בית:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'quotes'  
]
```

Django ORM:

ORM = Object Relational Mapper

מערכת שיצרת מיפוי של אובייקטים בOOD
לSQL

מетодות: save/get/all וכו'

הMETHODS יבצעו פעולה SQL

כשעובדים עם ORM קל להחילך שרט SQL

Django ORM:

המחשה של עבודה עם מסד נתונים:

lec2/quotes/models.py

```
from django.db import models

# Create your models here.
class Quote(models.Model):
    # props?
    author = models.CharField(max_length=40)
    quote = models.TextField()
    year = models.IntegerField()

    # administrative time stamps
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

נודיע למערכת שיצרנו מודל חדש:

python manage.py makemigrations quotes

הרצתה של השינויים מול מסד הנתונים:

python manage.py migrate

Django ORM:

המחשה של עבודה עם מסד נתונים:

The screenshot shows a SQLite database viewer interface with three tabs at the top: `models.py`, `db.sqlite3`, and `admin.py`. The `db.sqlite3` tab is active, displaying the contents of the `db.sqlite3` database. On the left, a sidebar lists the tables in the database, including `auth_group`, `auth_group_permissions`, `auth_permission`, `auth_user`, `auth_user_groups`, `auth_user_user_permissions`, `django_admin_log`, `django_content_type`, `django_migrations`, `django_session`, and `quotes_quote`. The `quotes_quote` table is currently selected and highlighted in blue. The main area shows a table with the following data:

| | id | author | quote | year | created_at |
|--|-----------|---------------|----------------------|-------------|-------------------|
| | 1 | Itamar | scrippppppppptttttt! | 2024 | 2025-02-16 08:05: |
| | 2 | | | | |

Below the table, there is a large empty white space.

Django ORM:

הוספת המודל החדש לממשק ניהול של Django

```
from django.contrib import admin

# Register your models here.
from .models import Quote

admin.site.register([Quote])

# python manage.py createsuperuser
# python manage.py runserver
```

תבניות HTML

quotes/templates/quotes/list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>List</h1>
    <table>
        <tr>
            <th>Quote</th>
            <th>Author</th>
            <th>Year</th>
        </tr>

        {% for quote in quotes %}
        <tr>
            <td>{{quote.quote}}</td>
            <td>{{quote.author}}</td>
            <td>{{quote.year}}</td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```

שאילתות ORDER BY

```
from .models import Quote
```

ORDER BY year ASC

```
def quotes_list(request:HttpRequest):  
    quotes = Quote.objects.all().order_by('year')  
    return render(request, 'quotes/list.html', {'quotes': quotes})
```

ORDER BY year DESC

```
from .models import Quote  
def quotes_list(request:HttpRequest):  
    quotes = Quote.objects.all().order_by('-year')  
    return render(request, 'quotes/list.html', {'quotes': quotes})
```

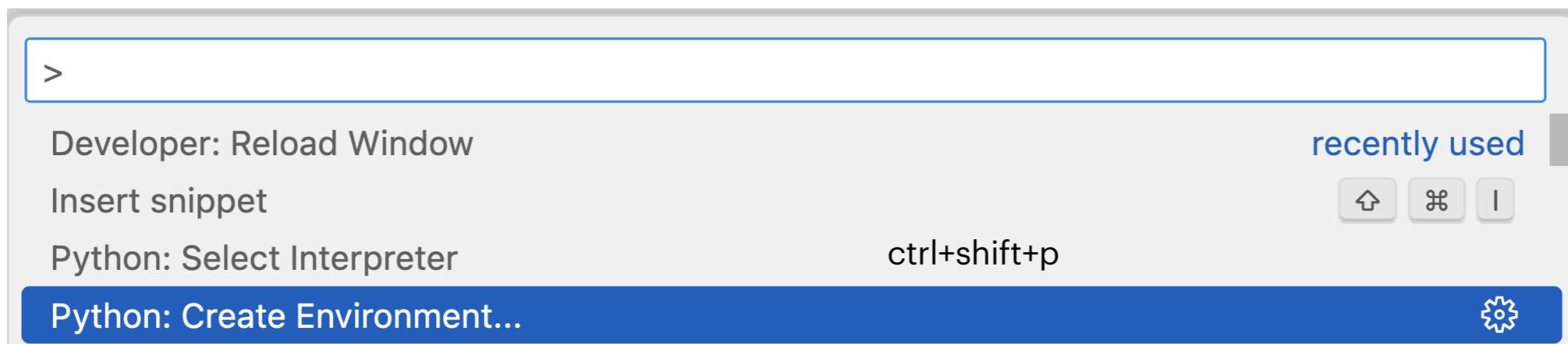
Virtual Environment:

כשעובדים על פרויקט - חשוב שכלל את כל הספריות בפרויקט בתוך התיקיה של הפרויקט
(חסרו - התיקיה תהיה כבדה יותר)
(יתרונו - הפרויקט מכיל את כל התלויות)

`python -m venv .venv`

שם התיקיה שתיווצר
בתוך תיקית הפרויקט
übero הספריות

אפשרות נוחה ליצירת venv:



Virtual Environment:

הפעלה של `venv`:

הרצה של קובץ `activate` בתיקיה `.venv/scripts`:

חלונות:

`.venv/scripts/activate`

מак/לינוקס:

`source .venv/bin/activate`

אפשרות נוחה להפעלה של `venv`:

(i) We noticed a new environment has been created. Do you want to select it for the workspace folder?  

Source: Python

Yes

No

Don't show again

Virtual Environment:

```
pip install django
```

```
pip install django-bootstrap-v5
```

```
pip install django-bootstrap-icons
```

כל הספריות שモתקנות בעט - יותקנו לתוך הפרויקט עצמו:



עובדת עם Bootstrap

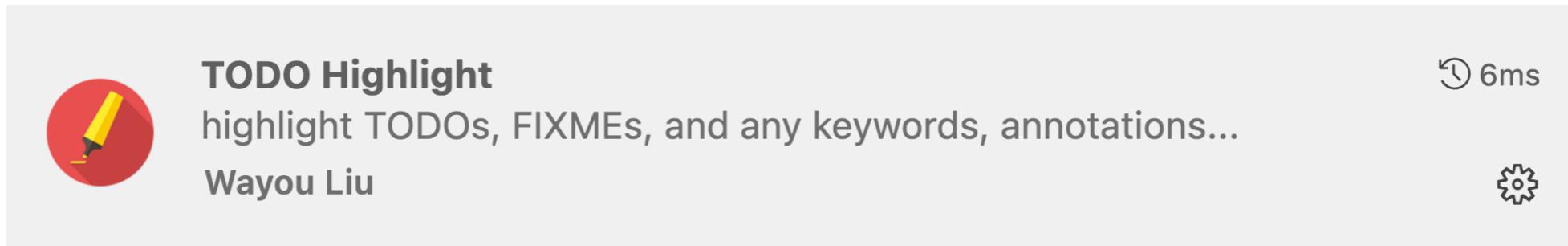
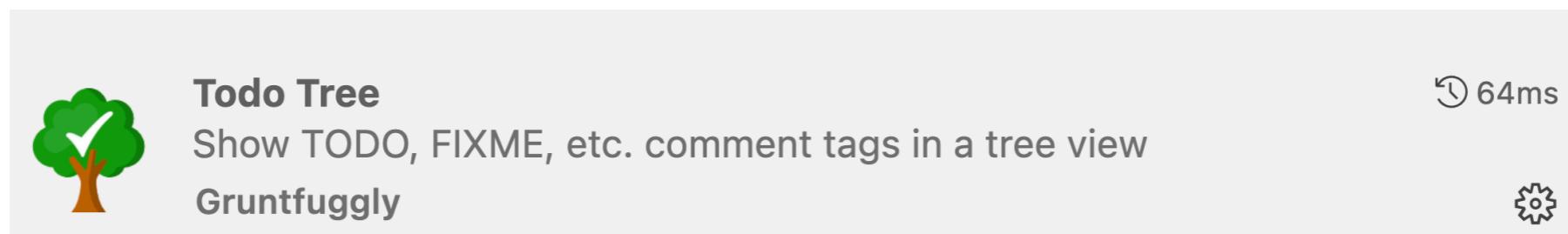
```
pip install django-bootstrap-v5
```

```
pip install django-bootstrap-icons
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'bootstrap5',  
    'django_bootstrap_icons',  
    'quotes'  
]
```

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Document</title>  
    {% load bootstrap5 %}  
    {% bootstrap_css %}  
    {% bootstrap_javascript %}  
  
    {% load bootstrap_icons %}  
  
</head>  
<body>  
    <h1>List</h1>  
    <table>  
        <tr>  
            <th>Quote</th>  
            <th>Author</th>  
            <th>Year</th>  
        </tr>  
  
        {% for quote in quotes %}  
            <tr>  
                <td>{{quote.quote}}</td>  
                <td>{{quote.author}}</td>  
                <td>{{quote.year}}</td>  
            </tr>  
        {% endfor %}  
    </table>  
</body>  
</html>
```

תוספָּה לשמירה של Code Snippets ב-VSCode



תוספָּה לשמירה של Code Snippets ב-VSCode

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  {%load bootstrap_icons%}
  <title>Document</title>
  {% load bootstrap5 %}
  {% bootstrap_css %}
  {% bootstrap_javascript %}
</head>
<body>
  <h1>List</h1>
  <table class="table table-striped">
    <tr>
      <th>Quote</th>
      <th>Author</th>
      <th>Year</th>
    </tr>

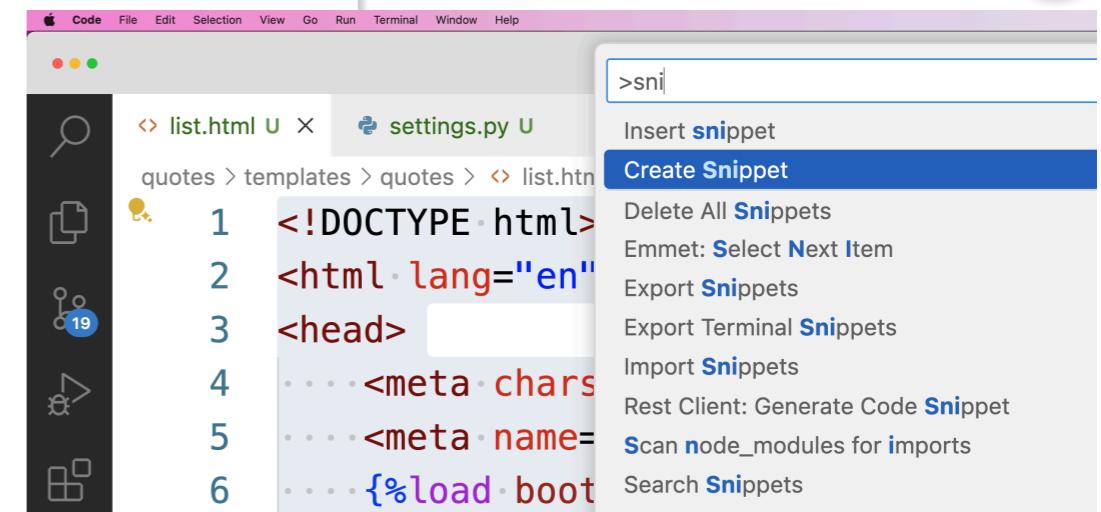
    {% for quote in quotes %}
      <tr>
        <td>{{quote.quote}}</td>
        <td>{{quote.author}}</td>
        <td>{{quote.year}}</td>
      </tr>
    {% endfor %}
  </table>
</body>
</html>
```

נסמן קוד שרוצים לשימור

1

ctrl+shift+p -> create snippet

2



שם הסnippet: dbs5

3

חייבים למלא את התגיות

שימוש ב-snippet : dbs5

4

תבנית לשימוש חוזר ב-HTML

templates/quotes/base.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>
      {% block title %}
        Quotes
      {% endblock title %}
    </title>

    {% block bootstrap %}
      {% load bootstrap_icons %}
      {% load bootstrap5 %}
      {% bootstrap_css %}
      {% bootstrap_javascript %}
    {% endblock %}

  </head>
  <body>
    <header>Site Header</header>
    <nav>Site Navigation</nav>
    <main>
      {% block content %}{% endblock content %}
    </main>
    <footer>Site Footer</footer>
  </body>
</html>
```

נרצה לכלול בכל העמודים את אותו סרגל

מה צריך להשתנות:
title, content

תבנית לשימוש חוזר בHTML

list.html

```
{% extends "quotes/base.html" %}

{% block title %}
    Quotes List
{% endblock title %}

{% block content %}
    <table class="table table-striped">
        <tr>
            <th>Quote</th>
            <th>Author</th>
            <th>Year</th>
        </tr>

        {% for quote in quotes %}
            <tr>
                <td>{{ quote.quote }}</td>
                <td>{{ quote.author }}</td>
                <td>{{ quote.year }}</td>
            </tr>
        {% endfor %}
    </table>
{% endblock content %}
```

קובץ שיורש מHTML

פעולות delete, get

views.py

```
def edit_quote(request:HttpRequest, id:int):
    if request.method == 'GET':
        quote = Quote.objects.get(id=id)
        return render(request, 'quotes/edit.html', {'quote': quote})
    else:
        pass ←———— נישם בשקפים הבאים

def delete_quote(request:HttpRequest, id:int):
    quote = Quote.objects.get(id=id)
    quote.delete()
    return HttpResponseRedirect('/quotes')
```

urls.py

```
from django.urls import path
from .views import quotes_list, add, edit_quote, delete_quote

urlpatterns = [
    path('', quotes_list, name='quotes_list'),
    path('edit/<id:int>/', edit_quote, name='edit_quote'),
    path('delete/<id:int>/', delete_quote, name='delete_quote'),
    path('add/', add, name='add'),
]
```

תגיות a - קישורים

list.html

```
{% extends "quotes/base.html" %}

{% block title %}
    Quotes List
{% endblock title %}

{% block content %}
    <table class="table table-striped">
        <tr>
            <th>Quote</th>
            <th>Author</th>
            <th>Year</th>
            <th>Actions</th>
        </tr>

        {% for quote in quotes %}
            <tr>
                <td>{{ quote.quote }}</td>
                <td>{{ quote.author }}</td>
                <td>{{ quote.year }}</td>
                <td>
                    <a href="{% url 'edit_quote' quote.id %}">
                        Edit
                    </a>
                    <a href="{% url 'delete_quote' quote.id %}">
                        Delete
                    </a>
                </td>
            </tr>
        {% endfor %}
    </table>
{% endblock content %}
```

זה ה-name
מתוך הקובץ
urls.py

פעולות edit

edit.html

```
{% extends "quotes/base.html" %}

{% block title %}
    Edit Quote
{% endblock title %}

{% block content %}
    <h1>Edit Quote</h1>
    <form method="post">
        {% csrf_token %}
        <label for="author">Author</label>
        <input type="text" name="author" id="author" value="{{ quote.author }}>

        <label for="quote">quote</label>
        <input type="text" name="quote" id="quote" value="{{ quote.quote }}>

        <label for="year">year</label>
        <input type="number" name="year" id="year" value="{{ quote.year }}>

        <input type="hidden" name="id" value="{{ quote.id }}>
        <input type="submit" value="Save">
    </form>
{% endblock content %}
```

פערות edit שימוש בעולת העריכה

```
def edit_quote(request:HttpRequest, id:int):
    quote = Quote.objects.get(id=id)

    if request.method == 'GET':
        return render(request, 'quotes/edit.html', {'quote': quote})
    else:
        author = request.POST.get('author')
        if not author:
            return render(request, 'quotes/edit.html', {'quote': quote, 'err': "No Author"})
        quote.author = author
        quote.quote = request.POST.get('quote', quote.quote)
        quote.year = request.POST.get('year', quote.year)

        quote.save()
```

איך ניתן לטרוף?

מחזירים שוב את הטופס
עם הודעת שגיאה:

```
{% extends "quotes/base.html" %}

{% block title %}
    Edit Quote
{% endblock title %}

{% block content %}
    <h1>Edit Quote</h1>

    {% if err %}
        <p class="alert alert-danger alert-dismissible fade show">{{ err }}</p>
    {% endif %}

    <form method="post">
```

פעולות edit

```
def edit_quote(request:HttpRequest, id:int):
    quote = Quote.objects.get(id=id)
    errors = {}
    if request.method == 'GET':
        return render(request, 'quotes/edit.html', {'quote': quote})
    else:
        author = request.POST.get('author')
        if not author:
            errors['author'] = 'No Author'
            return render(request, 'quotes/edit.html', {'quote': quote, 'errors': errors})
        quote.author = author
        quote.quote = request.POST.get('quote', quote.quote)
        quote.year = request.POST.get('year', quote.year)

    quote.save()
```

במקום הודעת שגיאה כללית
אפשר לשיק את השגיאה
לשדה מסוים - author

```
<form method="post">
    {% csrf_token %}
    <label for="author">Author</label>
    <input type="text" name="author" id="author" value="{{ quote.author }}" />
    {% if errors.author %}
        <p class="text text-danger">{{ errors.author }}</p>
    {% endif %}
    <label for="quote">quote</label>
    <input type="text" name="quote" id="quote" value="{{ quote.quote }}" />

    <label for="year">year</label>
    <input type="number" name="year" id="year" value="{{ quote.year }}" />

    <input type="hidden" name="id" value="{{ quote.id }}" />
    <input type="submit" value="Save" />
</form>
```

טיפוסים למדלים:

טיפוסים מספריים:

| סוג שדה | הסבר | פרמטרים חשובים |
|---------------------------|--------------------------------------|----------------------------|
| AutoField | מזהה ייחודי שנוצר אוטומטית (Integer) | primary_key=True |
| BigAutoField | מזהה ייחודי גדול יותר (BigInteger) | primary_key=True |
| BigIntegerField | שדה מספר שלם גדול במיוחד | |
| IntegerField | שדה מספר שלם | |
| PositiveIntegerField | שדה למספרים שלמים חיוביים בלבד | |
| PositiveBigIntegerField | שדה למספרים חיוביים גדולים בלבד | |
| SmallIntegerField | שדה למספר שלם קטן | |
| PositiveSmallIntegerField | שדה למספרים שלמים קטנים חיוביים בלבד | |
| DecimalField | שדה למספרים עשרוניים מדויקים | max_digits, decimal_places |
| FloatField | שדה למספרים ממשיים (float) | |

טיפוסים למדלים:

טיפוסים למחוזות:

| סוג שדה | הסבר | פרמטרים חשובים |
|------------|---|---------------------------|
| CharField | שדה טקסט עם אורך מוגבל | max_length |
| TextField | שדה טקסט ללא מגבלת אורך | |
| SlugField | טקסט ייחודי ל-URL (אותיות, מספרים, מקפירים) | max_length, allow_unicode |
| EmailField | טקסט המאמת כתובות דוא"ל | |
| URLField | טקסט המאמת כתובות URL | |
| UUIDField | מזהה ייחודי בפורמט UUID | |
| JSONField | שדה לאחסון נתונים JSON | |

טיפוסים למודלים:

טיפוסים לתאריכים:

| סוג שדה | הסבר | פרמטרים חשובים |
|---------------|-----------------------------|------------------------|
| DateField | שדה לתאריך בלבד | auto_now, auto_now_add |
| DateTimeField | שדה לתאריך ושעה | auto_now, auto_now_add |
| TimeField | שדה לשעה בלבד | |
| DurationField | שדה למרווחי זמן (timedelta) | |

```
created_at = models.DateTimeField(auto_now_add=True)  
updated_at = models.DateTimeField(auto_now=True)
```

טיפוסים למודלים:

טיפוסים ליחסים גומלי:

| סוג קשר | פרמטרים חשובים | יצוג במסד נתונים | הסבר |
|-----------------|---|--|----------------------------------|
| ForeignKey | on_delete, related_name, related_query_name, to_field | עמודת מפתח זר בטבלה, המקשרת לטבלת היעד. | קשר *רבים-לאחד* בין שני מודלים. |
| ManyToManyField | related_name, related_query_name, through, symmetrical | טבלה ביןים (junction table) המקשרת בין שתי הטבלאות. | קשר *רבים-לרבים* בין שני מודלים. |
| OneToOneField | on_delete, related_name, related_query_name, to_field | עמודת מפתח ייחודי עם מפתח זר בטבלה. | קשר *אחד-לאחד* בין שני מודלים. |

קשרי גומלין

```
# User is a built-in model                                     אחד לאחד:  
  
from django.contrib.auth.models import User  
  
class Student(models.Model):  
    # int filed, text field, date field  
    # relational fields  
    birth_date = models.DateField()  
    # student -> user (one-to-one)  
    user = models.OneToOneField(User, on_delete=models.CASCADE)  
  
    def __str__(self):  
        return f'{self.user.username} - {self.birth_date}'
```

מודלים עם וlidציה:

```
import django.core.validators as v
class Course(models.Model):
    name = models.CharField(max_length=100, unique=True, validators=[
        v.MinLengthValidator(2),
        v.MaxLengthValidator(100)
    ])
    description = models.TextField(blank=False, null=False)
    credits = models.PositiveSmallIntegerField(default=3,
                                                validators=[
            v.MinValueValidator(1),
            v.MaxValueValidator(5)
        ])

```

רבים לרבים:

```
#student.courses
#course.student_set
class Student(models.Model):
    courses = models.ManyToManyField(Course)
    # int filed, text field, date field
    # relational fields
    birth_date = models.DateField()
    # student -> user (one-to-one)
    user = models.OneToOneField(User, on_delete=models.CASCADE)

    def __str__(self):
        return f'{self.user.username} - {self.birth_date}'
```

לכל סטודנט יש תכונה של courses

לכל קורס יש תכונה של student_set

רבים לרבים:

:related_name

```
class Student(models.Model):
    courses = models.ManyToManyField(Course, related_name="students")
    # int filed, text field, date field
    # relational fields
    birth_date = models.DateField()
    # student -> user (one-to-one)
    user = models.OneToOneField(User, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.user.username} - {self.birth_date}"
```

לכל סטודנט יש תכונה של courses

לכל קורס יש תכונה של students

רבים לרבים: Many to many with extra fields:



id = 1, name = dave



id = 1, name = python



student_id = 1
course_id = 1
grade = 100

במקרה זה עליינו ליצור מפורשות את הטבלה המחברת

(מיימוש בעמוד הבא)

רבים לרבים: Many to many with extra fields:

```
class Enrollment(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE, related_name='enrollments')
    course = models.ForeignKey(Course, on_delete=models.CASCADE, related_name='enrollments')
    grade = models.IntegerField(null=True,
                                validators=[
                                    v.MinValueValidator(0),
                                    v.MaxValueValidator(100)
                                ])
```

הוספת המודלים למשק הניהול:

```
from django.contrib import admin

# Register your models here.
from .models import Quote, Student, Course, Enrollment

admin.site.register([Quote, Student, Course, Enrollment])
```

ביצוע השינויים במאסן הנתונים:

```
python manage.py makemigrations quotes
python manage.py migrate
```

זרעה של מסד-נתונים:

1

פקודת זרעה כללית שאפשר להריץ בכל רגע נתון:

במקרה שמסד הנתונים ריק ומעוניינים למלא אותו
נרצה להריץ Command שמלא את מסד הנתונים

2

מיגרציה לזרעה של מסד הנתונים:
בעת ייצור הדטה-בייס יתווסף הנתונים

זרעה של מסד נתונים:

```
└─ quotes
    └─ management/commands
        └─ seed_db.py
```

lec2/quotes/management/commands/seed_db.py

```
from django.core.management.base import BaseCommand

from django.contrib.auth.models import User, Group, Permission
from quotes.models import Quote, Student, Course, Enrollment
class Command(BaseCommand):
    help = 'Seed the database with initial data'

    def handle(self, *args, **options):
        self.stdout.write('Seeding the database...')
        quote = Quote(quote = "Scrippppptttt", author = "Itamar", year = 2024)
        quote.save()
```

הרצה של הסקריפט מהCMD

```
python manage.py seed_db
```

בעיה עם הסקריפט:

គותב את אותו מידע בכל הריצה
(נשפר את הסקריפט בעמוד הבא)

זרעה של מסד נתונים:

```
from django.core.management.base import BaseCommand
from django.contrib.auth.models import User, Group, Permission
from quotes.models import Quote, Student, Course, Enrollment
```

```
class Command(BaseCommand):
    help = 'Seed the database with initial data'
```

```
def handle(self, *args, **options):
    self.stdout.write('Seeding the database...')
```

```
quote, is_created = Quote.objects.get_or_create(quote='Scripttt!', defaults={
    'author': 'Itamar',
    'year': 2024
})
```

```
if is_created:
    self.stdout.write('Quote created!')
else:
    self.stdout.write('Quote already exists!')
```

```
# run the command:
# python manage.py seed_db
```

methode קיצור לחיפוש אובייקט קיים
או יירה של אובייקט חדש אם הוא לא קיים:

היפosh לפי "scripts!!"

defaults:

אם האובייקט לא קיים - יוצר אובייקט חדש עם הערכים הבאים

זרעה של מסד- נתונים:

זרעה של משתמשים:

```
from django.core.management.base import BaseCommand

from django.contrib.auth.models import User, Group, Permission
from quotes.models import Quote, Student, Course, Enrollment

class Command(BaseCommand):
    help = 'Seed the database with initial data'

    def handle(self, *args, **options):
        self.stdout.write('Seeding the database...')

        quote, is_created = Quote.objects.get_or_create(quote='Scripttt!', defaults={
            'author': 'Itamar',
            'year': 2024
        })

        user, _ = User.objects.get_or_create(username = 'itamar', defaults={
            'first_name': 'Itamar',
        })
        user.set_password('123456')
        user.save()

        self.stdout.write(f'User {user.username} created with password {user.password}')
```

הmethodה
דווגת להצפנה של הסיסמא
set_password

זרעה של מסד נתונים:

זרעה של טבלה עם יחס גומלין:

```
from django.core.management.base import BaseCommand

from django.contrib.auth.models import User, Group, Permission
from quotes.models import Quote, Student, Course, Enrollment


class Command(BaseCommand):
    help = 'Seed the database with initial data'

    def handle(self, *args, **options):
        self.stdout.write('Seeding the database...')

        quote, is_created = Quote.objects.get_or_create(quote='Scripttt!', defaults={
            'author': 'Itamar',
            'year': 2024
        })

        user, _ = User.objects.get_or_create(username='itamar', defaults={
            'first_name': 'Itamar',
        })

        user.set_password('123456')
        user.save()

        student, _ = (
            Student.objects
            .get_or_create(
                user=user, defaults={'birth_date': '1980-01-01'}
            )
        )

# run the command:
# python manage.py seed_db
```

זרעה של מסד-נתונים:

מיגרציה לזרעה של מסד הנתונים:
בעת יצירת הדטה-בייס יתווסף הנתונים

```

< migrations
>   __pycache__
>   __init__.py
>   0001_initial.py
>   0002_course_a
>   0003_seed.py

```

lec2/quotes/migrations/0003_seed.py

Generated by Django 4.2.19 on 2025-02-16 13:39

```
from django.db import migrations
from django.core.management import call_command
```

```
def run_seed_command(apps, schema_editor):
    call_command('seed_db')
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [
        ('quotes', '0002_course_alter_quote_year_student_enrollment'),
    ]
```

```
    operations = [
        migrations.RunPython(run_seed_command)
    ]
```

ניצור מיגרציה חדשה וריקה:

```
python manage.py makemigrations quotes --empty --name seed
```

הפקודה תיצור עבורנו קובץ migration בשם seed בתיקיית migrations בטעינה קובץ migration בשם seed.py

1

2

ערוך את הקובץ שנוצר בשלב 1

הfonקציה חיבת לקבלת 2 פרמטרים

הפעלה של ה함ction מהשלב הקודם

מיגרציה שמריצה קוד בפייתון:

שיעור בית:

צרו את המודלים הבאים:

Category:
name

שם הקטגוריה - טקסט קצר
קטגוריות דוגמא:
תכנות, פיתון, LOLAOT, HTTP, וכו'...

Post:
title
content
created
modified
categories

קשר של רבים ל רבים
ManyToManyField

Author:
name
birthday
user

קשר של אחד לאחד

Comment:
author
text
created
post

קשר של אחד ל רבים:
models.ForeignKey
קשר של אחד ל רבים:
models.ForeignKey

זרעו את המודלים בסיס הנתונים
שימוש לב ליצור ולידציות

תרגיל רשות:

צרו עמוד HTML שמציג את כל הכתובות

צרו עמוד HTML ליצירת כתבות

צרו עמוד HTML שעורך כתבה

לכל עמוד יש ליצור:

url

view

קובץ html