



UNIVERZITET U NOVOM SADU  
FAKULTET TEHNIČKIH NAUKA  
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

# Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2022/2023.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

# django

Šta je Django? → Veb razvojni okvir za perfekcionista sa rokovima.

💡 Django je Python okvir za razvoj veb aplikacija visokog nivoa koji podržava brz razvoj i čist, pragmatičan dizajn. Django je razvijen 2003. godine od strane iskusnih programera te je napravljen sa ciljem da se sam okvir pobrine za veb deo razvoja, a na programeru je da se posveti poslovnoj logici aplikacije.

Kompanije koje koriste Django:



## Važne karakteristike

- Python web framework
- Server Side Framework
- besplatan i otvorenog koda
- Batteries included
- Prati M-V-T (model-views-template) arhitekturni stil
- Django dokumentacija: <https://www.djangoproject.com/>

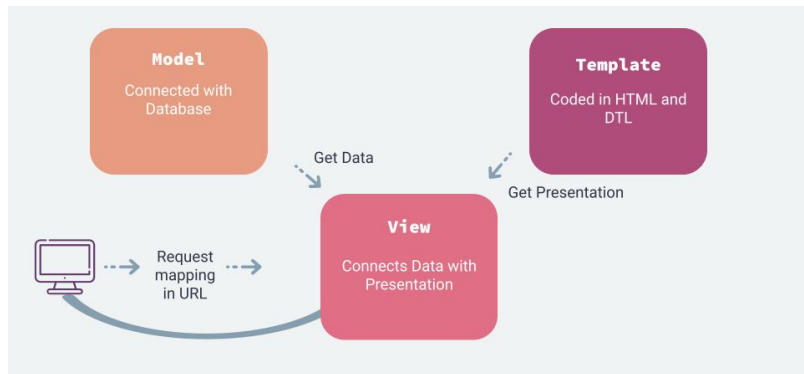
## M-V-T arhitekturni šablon

Šablon sličan MVC-u, ali za razliku od MVC-a kod MVT-a za deo sa kontrolerom (eng. Controller) se pobrinuo razvojni okvir.

**Model** - zadužen je za podatke i logiku. Povezan je sa bazom podataka.

**View** - izvršava poslovnu logiku, interaguje sa modelom i prikazuje templejte. Prihvata HTTP zahteve i vraća HTTP odgovore.

**Template** - sloj prezentacije koji u suštini podrazumeva HTML kod koji prikazuje podatke. Može biti statički ili dinamički.



# Prvi Django projekat

Potrebno je instalirati:

- Python
- Django
- + Poželjno je koristiti virtuelno okruženje za instalaciju

# Prvi Django projekat

\$ **django-admin** <command> [options]

- **django-admin** je djangov uslužni program komandne linije za administrativne zadatke
- unošenjem django-admin komande u terminal izlistavaju se sve moguće podkomande koje su dozvoljene

```
helena@helena-LIFE800K-A555-G:~$ django-admin
Type 'django-admin help <subcommand>' for help on a specific subcommand.

Available subcommands:

[django]
  check
  compilemessages
  createcachetable
  dbshell
  diffsettings
  dumpdata
  flush
  inspectdb
  loaddata
  makemessages
  makemigrations
  migrate
  optimizemigration
  runserver
  sendtestemail
  shell
  showmigrations
```

# Django-admin

\$ django-admin **startproject** **mysite**

- **startproject** je jedna od podkomandi kojom se kreira novi django projekat
- **mysite** je naziv projekta i direktorijuma koji se kreira izvršavanjem date komande (u pitanju je proizvoljni naziv)
- kreirani projekat mysite se sastoji iz sledećih delova:

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    asgi.py  
    wsgi.py
```

# Prvi Django projekat

Kreirani projekat se sastoji iz sledećih delova:

- spoljašnji **mysite/** je korenski direktorijum koji predstavlja kontejner za projekat.
- **manage.py** je uslužni program komandne linije koji omogućava interakciju sa kreiranim django projektom na različite načine.
- unutrašnji **mysite/** direktorijum je stvarni python paket za projekat.
- **mysite/\_\_init\_\_.py** je prazan fajl koji govori pythonu da ovaj direktorijum treba da se smatra python paketom
- **mysite/settings.py** predstavlja podešavanja/konfiguraciju za ovaj django projekat
- **mysite/urls.py** su URL deklaracije za django projekat, nešto poput tabele sadržaja.
- **mysite/asgi.py** je ulazna tačka za ASGI-kompatibilne veb servere.
- **mysite/wsgi.py** je ulazna tačka za WSGI-kompatibilne veb servere.

```
mysite/  
  manage.py  
  mysite/  
    __init__.py  
    settings.py  
    urls.py  
    asgi.py  
    wsgi.py
```

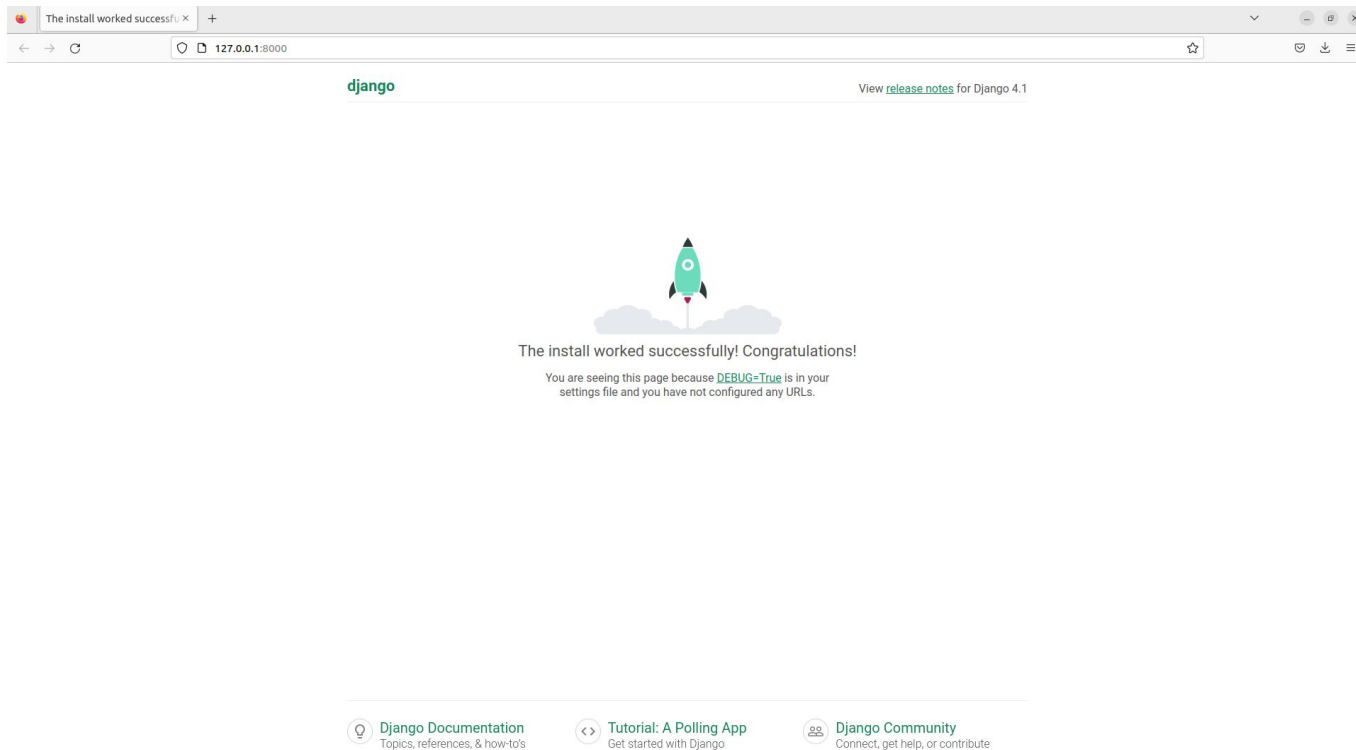


## Pokretanje servera

\$ python manage.py runserver

- izvršavanjem ove komande pokreće se server na localhost-u sa kreiranim projektom
- podrazumevani port je 8080, ali se on može izmeniti unošenjem porta kao dodatnog argumenta u komandnoj liniji na sledeći način: \$ python manage.py runserver 8081
- server se automatski prilagođava svim izmenama u kodu, te nije potrebno gasiti i ponovo paliti server kako bi se načinjene izmene primenile

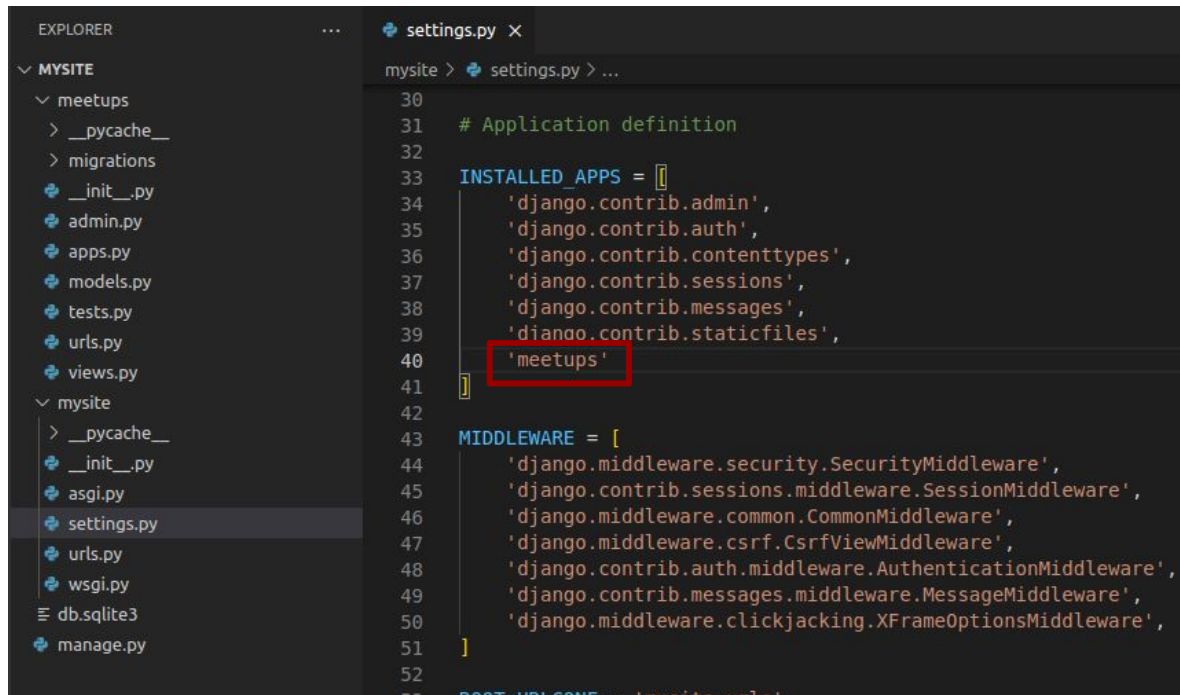
# Računarstvo u oblaku 2022/2023.



## Django aplikacije

- Django projekat se sastoji iz više malih aplikacija.
- Za projekat se može reći da je kolekcija konfiguracija i aplikacija za određenu veb aplikaciju.
- Jedna aplikacija može da se koristi u više projekata.
- Primer: online prodavnica bi mogla imati posebne aplikacije za korpu, pregled proizvoda, pregled istorije kupovine,...
- Komanda za kreiranje aplikacije: `$ python manage.py startapp meetups`
- Nakon kreiranja aplikacije potrebno je registrovati datu aplikaciju u settings.py fajl projekta

## Registrowanje nove aplikacije



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'mysite'. The tree includes a 'meetups' directory and a 'mysite' directory. The 'mysite' directory contains files like '\_\_pycache\_\_', '\_\_init\_\_.py', 'asgi.py', 'settings.py' (which is selected and highlighted), 'urls.py', 'wsgi.py', 'db.sqlite3', and 'manage.py'. The main editor area shows the 'settings.py' file. The 'INSTALLED\_APPS' list is expanded, and the string 'meetups' is being added to the list, highlighted with a red rectangular box. The code is as follows:

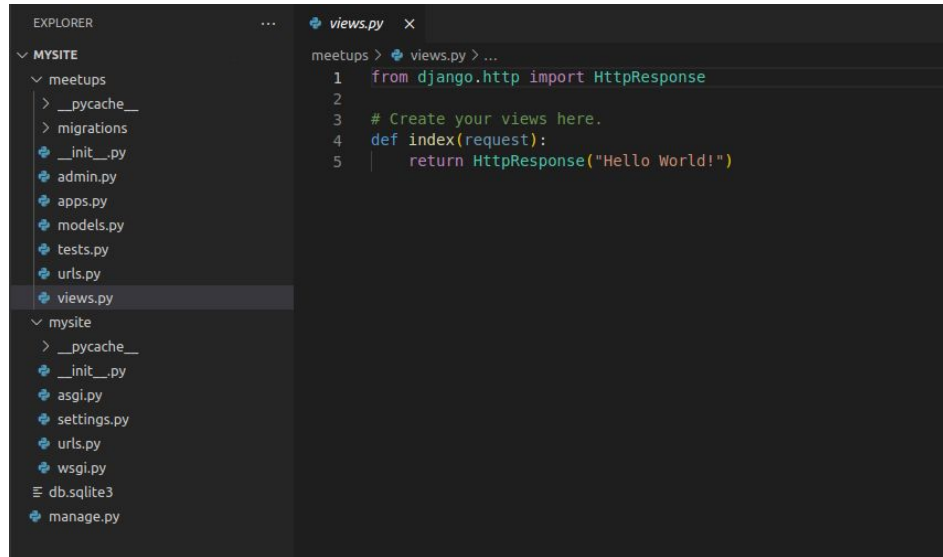
```
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'meetups'
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'mysite.urls'
```

## Http request/response

- Da bismo napravili jedan funkcionalan http request/response, potrebno je uraditi sledeće korake:
  1. napisati u okviru view.py fajla funkciju koja prima zahtev i vraća odgovor
  2. definisati url u urls.py fajlu koji namenjen datom zahtevu (urlpatterns)
  3. obezbediti mapiranje aplikacije u okviru projekta

# Http request/response

1.



The screenshot shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a file tree. The tree is expanded to show the 'mysite' directory, which contains files like '\_\_pycache\_\_', '\_\_init\_\_.py', 'asgi.py', 'settings.py', 'urls.py', 'wsgi.py', 'db.sqlite3', and 'manage.py'. The 'views.py' file is selected. The main editor area shows the code for 'views.py', which includes an import for 'HttpResponse' from 'django.http' and a function 'index' that returns 'HttpResponse("Hello World!")'.

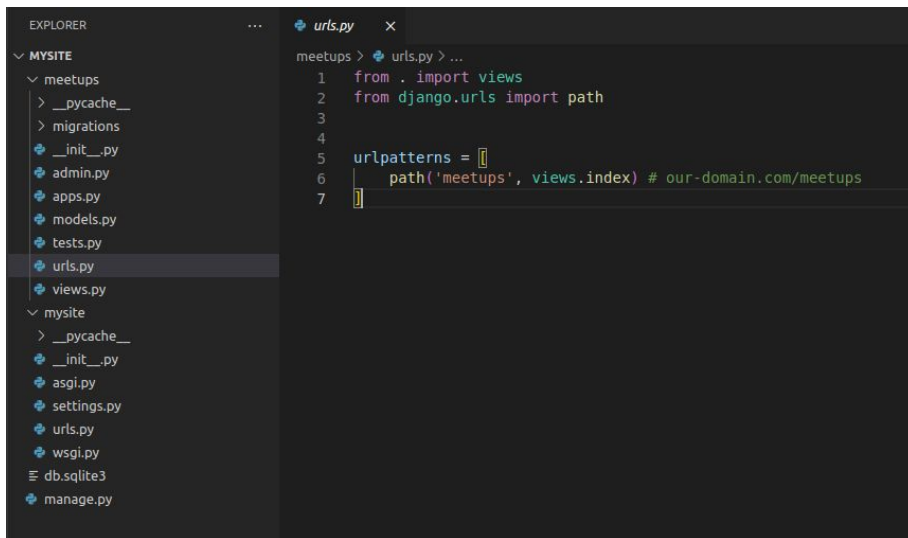
```
EXPLORER
...
views.py x

mysite
  > __pycache__
  > migrations
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  urls.py
  views.py
mysite
  > __pycache__
  __init__.py
  asgi.py
  settings.py
  urls.py
  wsgi.py
  db.sqlite3
  manage.py

meetups > views.py > ...
1 from django.http import HttpResponse
2
3 # Create your views here.
4 def index(request):
5     return HttpResponse("Hello World!")
```

# Http request/response

2.

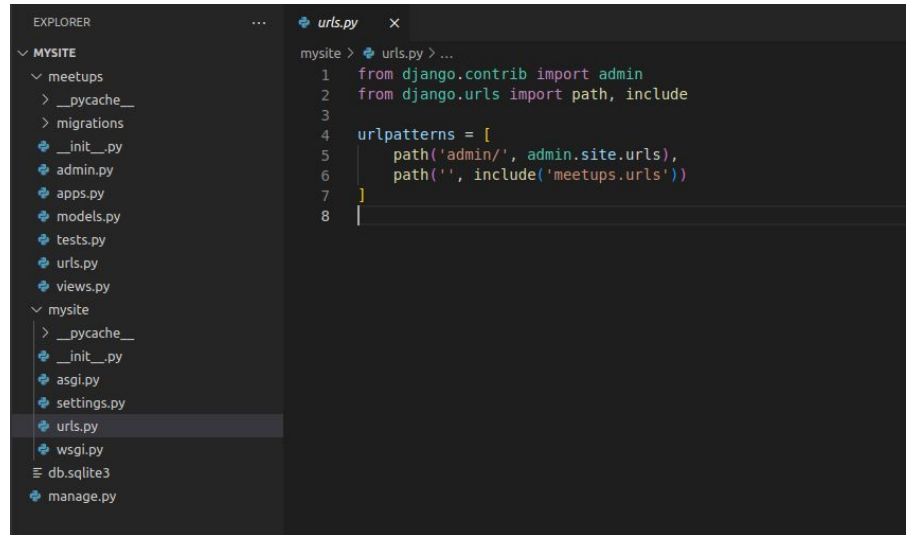


The screenshot shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a project structure. The 'MYSITE' folder is expanded, showing subfolders 'meetups' and 'mysite'. The 'meetups' folder is further expanded, listing files: '\_\_pycache\_\_', 'migrations', '\_\_init\_\_.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py' (which is selected and highlighted), and 'views.py'. The 'mysite' folder is also expanded, listing files: '\_\_pycache\_\_', '\_\_init\_\_.py', 'asgi.py', 'settings.py', 'urls.py', 'wsgi.py', 'db.sqlite3', and 'manage.py'. The main editor area shows the content of 'urls.py' with the following code:

```
meetups > urls.py > ...
1  from . import views
2  from django.urls import path
3
4
5  urlpatterns = []
6  |   path('meetups', views.index) # our-domain.com/meetups
7  |
```

# Http request/response

3.



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'MYSITE'. The tree includes folders for 'meetups' and 'mysite', and various Python files. The 'urls.py' file in the 'mysite' folder is selected and highlighted. The main editor area shows the content of 'urls.py', which includes imports for 'admin' and 'path' from 'django.contrib' and 'django.urls' respectively. It also defines a 'urlpatterns' list with two entries: a path for the admin site and a path that includes the 'meetups.urls'.

```
EXPLORER
...
MYSITE
  meetups
    __pycache__
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  mysite
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
    db.sqlite3
    manage.py

mysite > urls.py > ...
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('', include('meetups.urls'))
7 ]
8
```

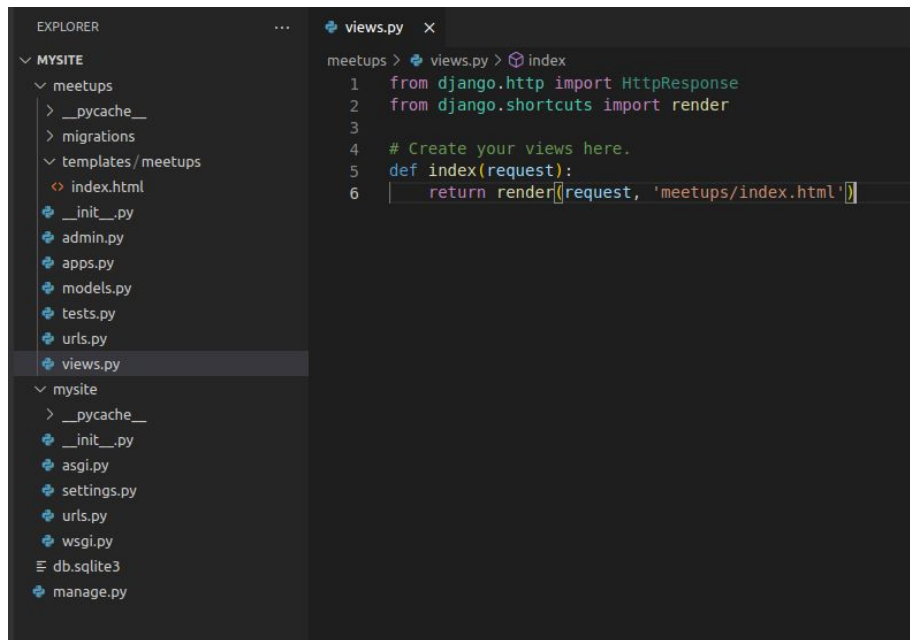


# Http request/response



# HTML fajl kao odgovor na zahtev

- `render()` funkcija vraća templejt kao http response



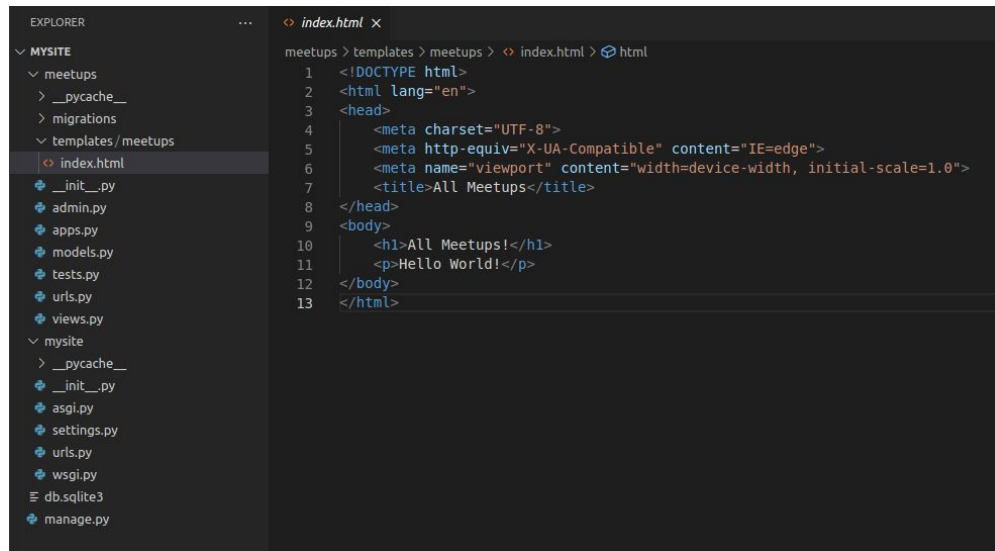
The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'MYSITE'. The tree includes folders like 'meetups' and 'templates/meetups', and various Python files such as 'views.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py', 'asgi.py', 'settings.py', 'wsgi.py', 'db.sqlite3', and 'manage.py'. The 'views.py' file in the 'meetups' folder is selected. On the right, the editor shows the content of 'views.py', which includes imports for 'HttpResponse' and 'render' from Django, a comment '# Create your views here.', and a function 'def index(request):' that returns 'render(request, 'meetups/index.html')'.

```
EXPLORER
MYSITE
├── meetups
│   ├── __pycache__
│   ├── migrations
│   └── templates/meetups
│       └── index.html
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── tests.py
├── urls.py
├── views.py
├── mysite
│   ├── __pycache__
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   ├── db.sqlite3
│   └── manage.py
└── ...

views.py
meetups > views.py > index
1 from django.http import HttpResponse
2 from django.shortcuts import render
3
4 # Create your views here.
5 def index(request):
6     return render(request, 'meetups/index.html')
```

# HTML fajl kao odgovor na zahtev

- Templejti moraju biti u određenom direktorijumu sa tačnim nazivom da bi radili



```
EXPLORER
...
index.html x

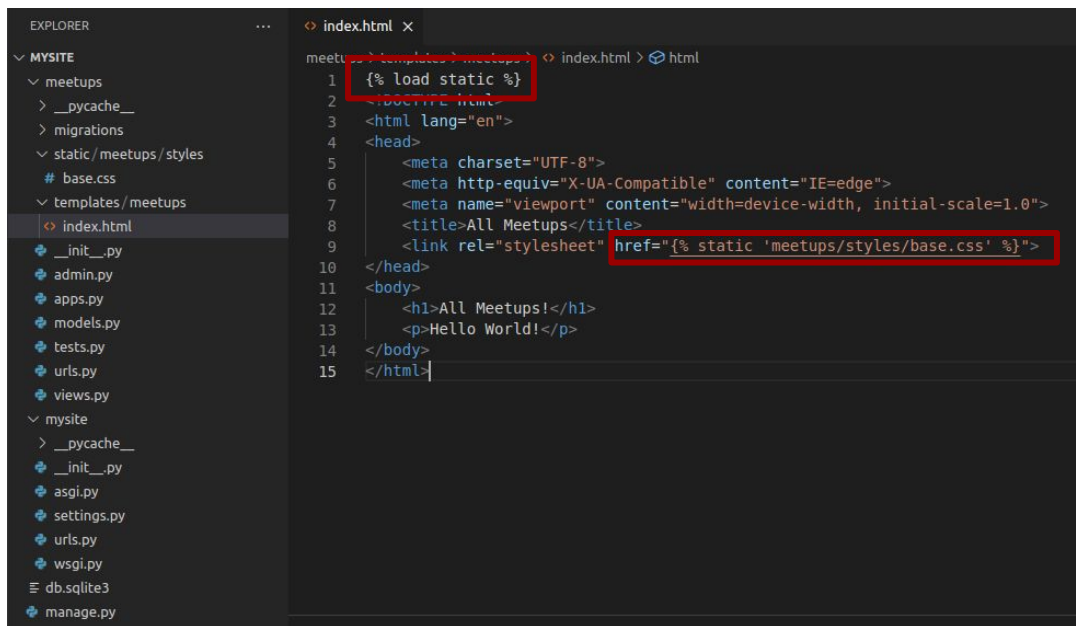
MYSITE
├── meetups
│   ├── __pycache__
│   ├── migrations
│   └── templates/
│       └── meetups
│           └── index.html
├── __init__.py
├── admin.py
├── apps.py
├── models.py
├── tests.py
├── urls.py
├── views.py
└── mysite
    ├── __pycache__
    ├── __init__.py
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    ├── wsgi.py
    ├── db.sqlite3
    └── manage.py

meetups > templates > meetups > index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>All Meetups</title>
8  </head>
9  <body>
10     <h1>All Meetups!</h1>
11     <p>Hello World!</p>
12 </body>
13 </html>
```

# HTML fajl kao odgovor na zahtev



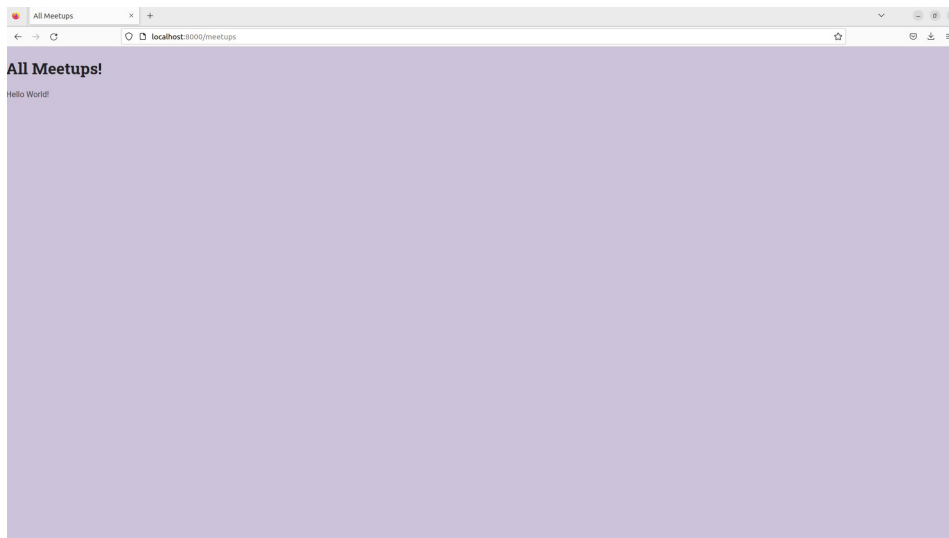
## Rad sa statičkim fajlovima



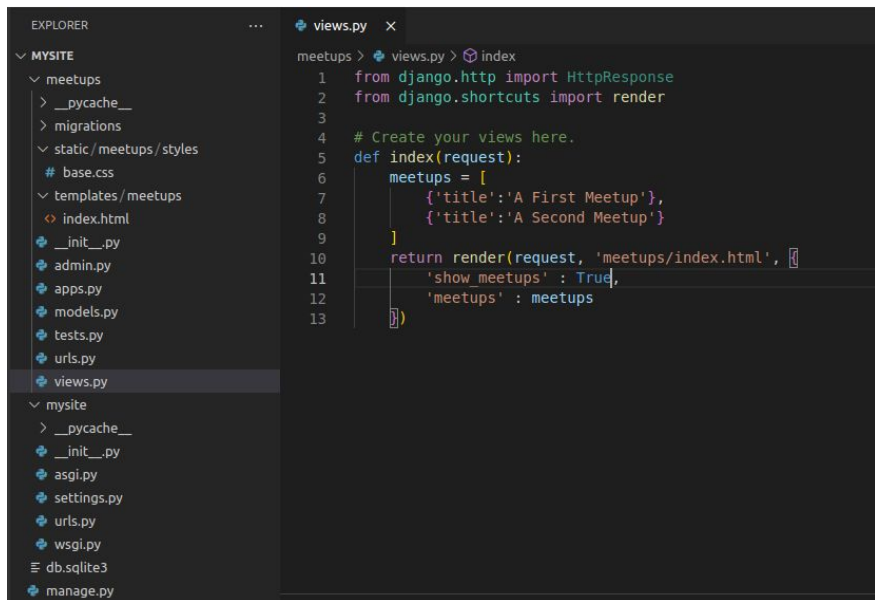
```
EXPLORER
MYSITE
  meetups
    __pycache__
    migrations
    static/meetups/styles
      base.css
    templates/meetups
      index.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  mysite
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
    db.sqlite3
    manage.py

index.html x
meetups > templates > meetups > index.html > html
1 {% load static %}
2 <doctype html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>All Meetups</title>
9   <link rel="stylesheet" href="{% static 'meetups/styles/base.css' %}">
10 </head>
11 <body>
12   <h1>All Meetups!</h1>
13   <p>Hello World!</p>
14 </body>
15 </html>
```

# Rad sa statičkim fajlovima



# Render funkcija

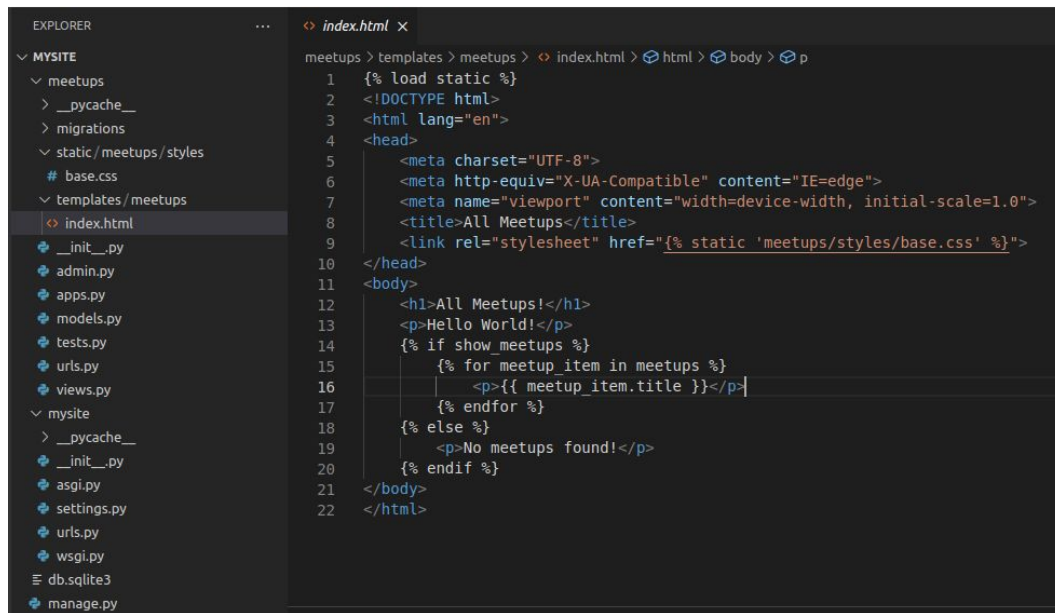


The image shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a project structure. The 'meetups' directory is expanded, showing files like `__pycache__`, `migrations`, `static/meetups/styles`, `base.css`, `templates/meetups`, `index.html`, `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py`, and `views.py`. The `views.py` file is selected. On the right, the code for `views.py` is displayed. It shows imports for `HttpResponse` and `render` from Django, a comment to create views here, and a function `index(request)` that returns a `render` call with the request, the template path `'meetups/index.html'`, and a context dictionary containing `'show_meetups': True` and `'meetups': meetups`.

```
EXPLORER
...
MYSITE
  meetups
    __pycache__
    migrations
    static/meetups/styles
      base.css
    templates/meetups
      index.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  mysite
    __pycache__
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
    db.sqlite3
    manage.py

views.py
meetups > views.py > index
1 from django.http import HttpResponse
2 from django.shortcuts import render
3
4 # Create your views here.
5 def index(request):
6     meetups = [
7         {'title': 'A First Meetup'},
8         {'title': 'A Second Meetup'}
9     ]
10    return render(request, 'meetups/index.html', {
11        'show_meetups': True,
12        'meetups': meetups
13    })
```

# Templejt jezik



The screenshot shows a code editor interface. On the left is the 'EXPLORER' sidebar showing a project structure for 'MYSITE'. The 'templates/meetups' directory is expanded, and 'index.html' is selected. The main editor area displays the content of 'index.html', which is a Django template. The code includes a {% load static %} directive, a <!DOCTYPE html> declaration, a <html lang='en'> tag, and a <head> section with meta tags for charset, http-equiv, and viewport, a title 'All Meetups', and a link to a static CSS file. The <body> section contains a <h1>All Meetups!</h1>, a <p>Hello World!</p>, and a conditional block that iterates over 'meetups' to display each item's title, or a message if no meetups are found.

```
meetups > templates > meetups > index.html > html > body > p
1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>All Meetups</title>
9      <link rel="stylesheet" href="{% static 'meetups/styles/base.css' %}">
10 </head>
11 <body>
12     <h1>All Meetups!</h1>
13     <p>Hello World!</p>
14     {% if show_meetups %}
15         {% for meetup_item in meetups %}
16             <p>{{ meetup_item.title }}</p>
17         {% endfor %}
18     {% else %}
19         <p>No meetups found!</p>
20     {% endif %}
21 </body>
22 </html>
```



# Templejt jezik

- `{% if <uslov> %}`    `#naredba grananje`  
  
`{% else <uslov> %}`  
  
`{% endif %}`
- `{% for <jedna_stavka> in <kolekcija_stavki> %}`    `#for petlja`  
  
`{% endfor %}`
- `{{ <promenljiva_koja_salje_u_render_funkciji> }}`    `#interpolacija`

# **Dovršavanje index.html stranice**

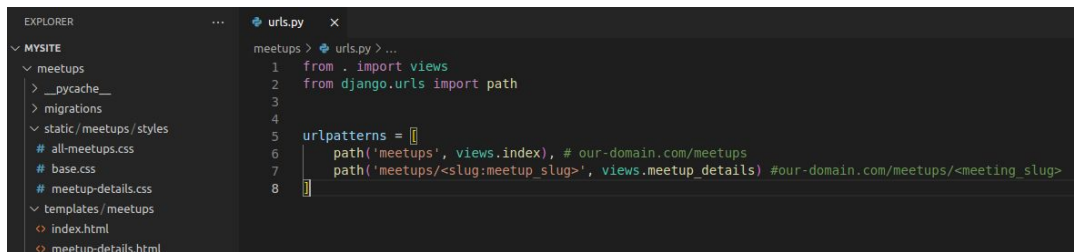
## Dinamičke putanje

- Za kreiranje dinamičkih putanja koriste se templejti poput:

```
path('<int:question_id>/', views.detail)
```

```
path('<int:question_id>/results/', views.results)
```

- `question_id` je deo stringa koji definiše naziv koji će se koristiti da se identifikuje šablon
- `int` deo služi za validaciju i konverziju

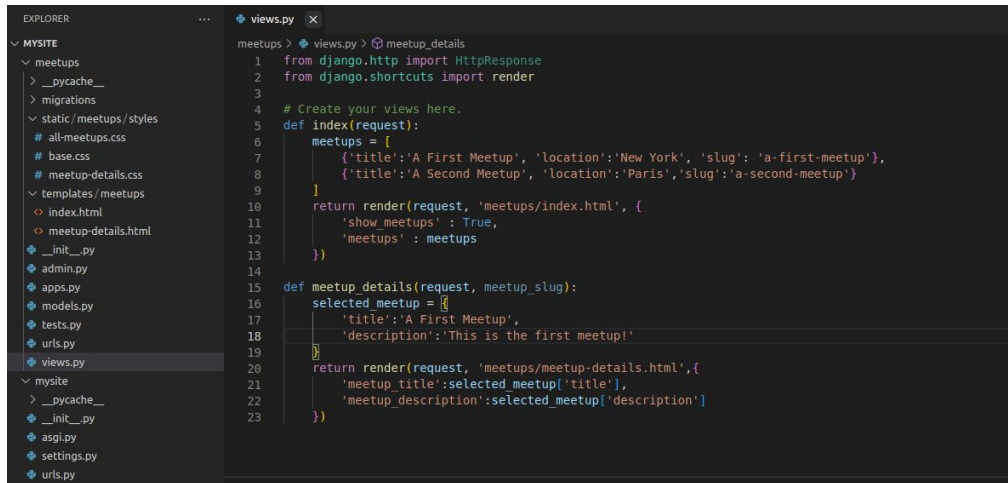


The screenshot shows a code editor with two panels. The left panel, titled 'EXPLORER', displays a file tree for a project named 'MYSITE'. The tree includes folders for 'meetups', '\_\_pycache\_\_', 'migrations', and 'static/meetups/styles', along with files like 'all-meetups.css', 'base.css', 'meetup-details.css', 'templates/meetups/index.html', and 'meetup-details.html'. The right panel, titled 'urls.py', shows the content of the file. It imports 'views' and 'path' from 'django.urls', and defines a list of URL patterns. The first pattern is 'meetups', which points to 'views.index'. The second pattern is 'meetups/<slug:meeting\_slug>', which points to 'views.meetup\_details'.

```
meetups > urls.py > ...
1  from . import views
2  from django.urls import path
3
4
5  urlpatterns = [
6      path('meetups', views.index), # our-domain.com/meetups
7      path('meetups/<slug:meeting_slug>', views.meetup_details) #our-domain.com/meetups/<meeting_slug>
8  ]
```

# Dinamičke putanje

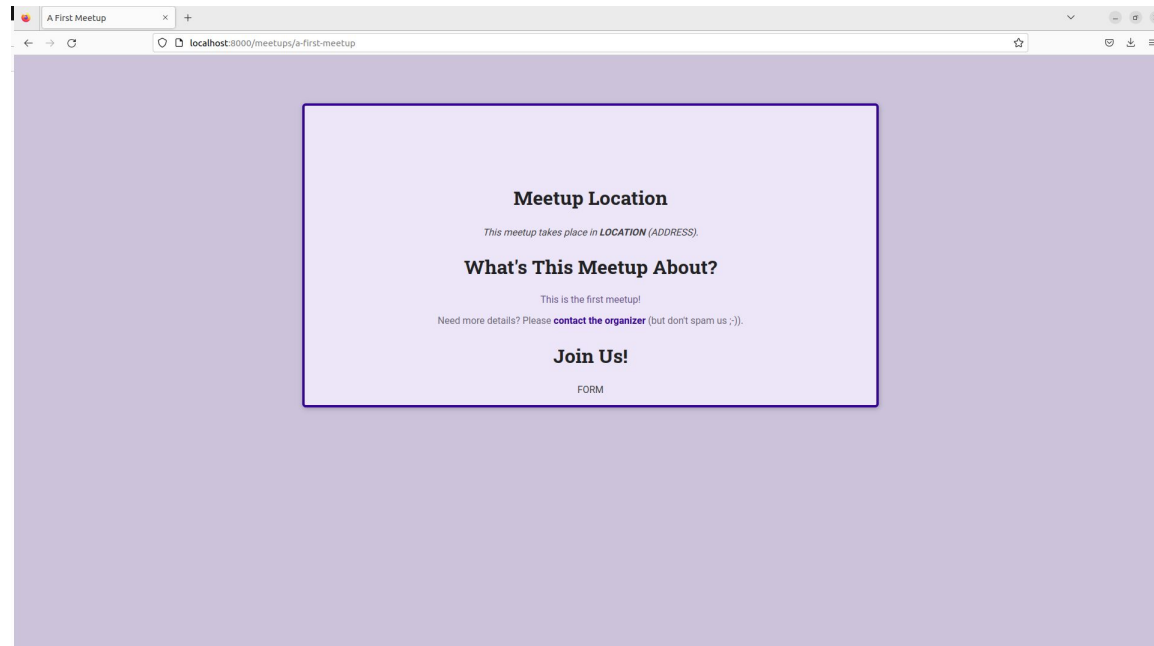
- Svaki funkcija u view je zadužena ili da vrati HttpResponse objekat ili da izazove izuzetak kao što je Http404
- Ako je putanja dinamička, tada kao drugi parametar funkcije treba da bude naziv koji se koristio kao deo šablona u URL-u (čak iako se ne upotrebljava)



```
EXPLORER
mysite
  __pycache__
  migrations
  static/meetups/styles
    all-meetups.css
    base.css
    meetup-details.css
  templates/meetups
    index.html
    meetup-details.html
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  urls.py
  views.py
mysite
  __pycache__
  __init__.py
  asgi.py
  settings.py
  urls.py

views.py
meetups > views.py > meetup_details
1 from django.http import HttpResponse
2 from django.shortcuts import render
3
4 # Create your views here.
5 def index(request):
6     meetups = [
7         {'title': 'A First Meetup', 'location': 'New York', 'slug': 'a-first-meetup'},
8         {'title': 'A Second Meetup', 'location': 'Paris', 'slug': 'a-second-meetup'}
9     ]
10    return render(request, 'meetups/index.html', {
11        'show_meetups': True,
12        'meetups': meetups
13    })
14
15 def meetup_details(request, meetup_slug):
16     selected_meetup = {}
17     'title': 'A First Meetup',
18     'description': 'This is the first meetup!'
19     }
20    return render(request, 'meetups/meetup-details.html', {
21        'meetup_title': selected_meetup['title'],
22        'meetup_description': selected_meetup['description']
23    })
```

# Dinamičke putanje - rezultat



# Dinamički URL-ovi u templejtima

- Interpolacija je moguća i u linkovima - cilj je izbegavanje hardkodovanih URL-ova

```
index.html x
meetups > templates > meetups > index.html > html > body > main > section > ol > li.meetup-item > article > div.meetup-action
22     {% for meetup_item in meetups %}
23     <li class="meetup-item">
24         <article>
25             <div class="meetup-summary">
26                 <div class="meetup-image">
27                     <img src="" alt="">
28                 </div>
29                 <div class="meetup-details">
30                     <h3>{{meetup_item.title}}</h3>
31                     <address>{{meetup_item.location}}</address>
32                 </div>
33             </div>
34             <div class="meetup-actions">
35                 <a href="/meetups/{{ meetup_item.slug }}" class="btn">More Details</a>
36             </div>
37         </article>
38     </li>
39     {% endfor %}
40 </ol>
41 </section>
42 </main>
43 </body>
44 </html>
```

## Dinamički URL-ovi u templejtima

- `{% url 'naziv_jednak_nazivu_urlpatterna' %}` - u primeru `meetup-details`

```
32         </div>
33     </div>
34     <div class="meetup-actions">
35         <a href="{% url 'meetup-details' meetup_item.slug %}" class="btn">More Details</a>
36     </div>
37 </article>
38 </li>
```

```
< Index.html  urls.py  X
meetups > urls.py > ...
1  from . import views
2  from django.urls import path
3
4
5  urlpatterns = [
6      path('meetups', views.index, name='all-meetups'), # our-domain.com/meetups
7      path('meetups/<slug:meetup_slug>', views.meetup_details, name='meetup-details') #our-domain.com/meetups/<meeting_slug>
8  ]
```

# Nasleđivanje templejta



# Includes

## Rad sa bazom podataka

- Podrazumevano konfiguracija koristi SQLite
- Za velike projekte poželjno je koristiti skalabilnije baze podataka kao što je PostgreSQL
- **Model** je jedinstveni izvor informacija o podacima u aplikaciji. Model sadrži najvažnija polja i ponašanja u vezi podataka koje skladištimo.
- Svako polje u modelu reprezentovano je jednom instancom klase Field: CharField, DateTimeField, TextField, SlugField,... Ovo govori Django koji tip podataka će se nalaziti u polju
- Mogu se definisati i dodatna ograničenja poput maksimalne dužine(max\_length) i jedinstvenosti(unique)

```
models.py x
meetups > models.py > ...
1  from enum import unique
2  from pydoc import describe
3  from django.db import models
4
5  # Create your models here.
6
7  class Meetup(models.Model):
8      title = models.CharField(max_length=200)
9      slug = models.SlugField(unique=True)
10     description = models.TextField()
11
```

## Aktivacija modela

- Kako bi Django kreirao šemu baze podataka na osnovu definisanih modela, potrebno je:
  1. Kreirati migraciju: `$ python3 manage.py makemigrations`
  2. Primeniti migraciju: `$ python3 manage.py migrate`
- **makemigrations** govori Django da postoje izmene u modelu i da je potrebno te izmene skladištiti kao migracije. Migracije su način na koji Django skladišti izmene na modelima (samim tim i šemi baze podataka) - to su fajlovi na disku.
- **migrate** komanda uzima sve migracije koje još nisu primenjene i primenjuje ih na bazu podataka - suštinski, sinhronizuje modele sa šemom baze podataka.
- Migracije su moćan alat koji omogućava izmene na modelima tokom vremena, kako se razvija projekat - bez potrebe da se briše baza podataka ili tabele i da se prave nove.

# Admin panel

\$python3 manage.py **createsuperuser**

- Kako bi se koristio admin panel, potrebno je registrovati jednog korisnika koji će imati admin privilegije.
- Nakon pokretanje servera na sledećem URL-u se može pristupiti admin panelu:  
**localhost:8000/admin/**
- Kako bi se iz admin panela moglo rukovati modelima potrebno je registrovati odgovarajući model u **admin.py** fajlu.

```
EXPLORER
...
admin.py x
meetups > admin.py
1 from django.contrib import admin
2
3 from .models import Meetup
4
5 # Register your models here.
6 admin.site.register(Meetup)
```

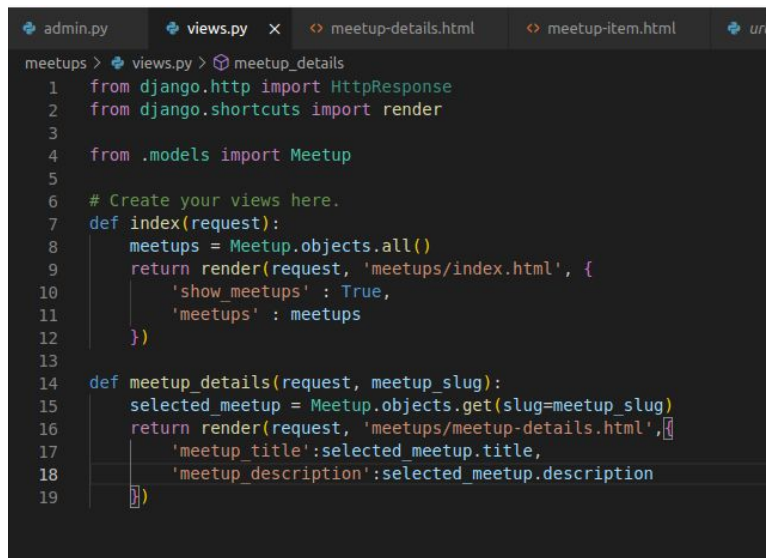
## Modifikacija admin panela

- Redefinisanjem povratne vrednosti za metodu `__str__` u okviru klase modela, modifikuje se identifikacija objekta u okviru admin panela
- Primer: umesto `Location object(1)` piše Novi Sad
- Moguće modifikacije su i prikaz u vidu tabele sa odgovarajućim poljima iz modela kao naslov kolone:

```
class LocationAdmin(admin.ModelAdmin):  
    list_display = ('name', 'address')  
  
# Register your models here.  
admin.site.register(Meetup)  
admin.site.register(Location, LocationAdmin)  
admin.site.register(Participant)
```

## Dobavljanje objekata iz baze

- `Model.objects.all()`
- `Model.objects.get(id=1)`
- `Model.objects.all().filter()`
- `Model.objects.all().orderby()`



The screenshot shows a code editor with several tabs: `admin.py`, `views.py` (active), `meetup-details.html`, `meetup-item.html`, and `urls.py`. The `views.py` file contains the following Python code:

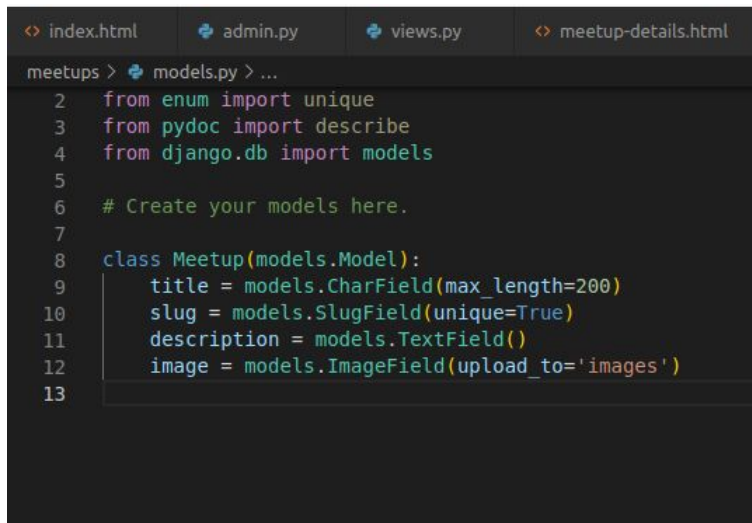
```
meetups > views.py > meetup_details
1 from django.http import HttpResponse
2 from django.shortcuts import render
3
4 from .models import Meetup
5
6 # Create your views here.
7 def index(request):
8     meetups = Meetup.objects.all()
9     return render(request, 'meetups/index.html', {
10         'show_meetups': True,
11         'meetups': meetups
12     })
13
14 def meetup_details(request, meetup_slug):
15     selected_meetup = Meetup.objects.get(slug=meetup_slug)
16     return render(request, 'meetups/meetup-details.html', {
17         'meetup_title': selected_meetup.title,
18         'meetup_description': selected_meetup.description
19     })
```

# try-except

```
admin.py  views.py  meetup-details.html  meetup-item.html  urls.py
meetups > views.py > meetup_details
4  from .models import Meetup
5
6  # Create your views here.
7  def index(request):
8      meetups = Meetup.objects.all()
9      return render(request, 'meetups/index.html', {
10          'show_meetups': True,
11          'meetups': meetups
12      })
13
14  def meetup_details(request, meetup_slug):
15      try:
16          selected_meetup = Meetup.objects.get(slug=meetup_slug)
17          return render(request, 'meetups/meetup-details.html', {
18              'meetup_found': True,
19              'meetup_title': selected_meetup.title,
20              'meetup_description': selected_meetup.description
21          })
22      except Exception as exc:
23          return render(request, 'meetups/meetup-details.html', {
24              'meetup_found': False
25          })
```

```
admin.py  views.py  meetup-details.html  meetup-item.html  urls.py  index.html
meetups > templates > meetups > meetup-details.html > p
17  {% block body %}
18
19  {% if meetup_found %}
20  <article>
21      <img src="" alt="">
22
23      <section id="location">
24          <h2>Meetup Location</h2>
25          <address>This meetup takes place in <span>LOCATION</span> (ADDRESS).</address>
26      </section>
27
28      <section id="details">
29          <h2>What's This Meetup About?</h2>
30          <p>{{ meetup_description }}</p>
31          <footer>
32              <p>Need more details? Please <a href="">contact the organizer</a> (but don't spam us ;-)).</p>
33          </footer>
34      </section>
35
36      <section id="registration">
37          <h2>Join Us!</h2>
38          FORM
39      </section>
40  </article>
41  {% else %}
42  <p>No meetup found for this slug, sorry!</p>
43  {% endif %}
44
45
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
You have Docker installed on your system. Do
install the recommended extensions for it?
```

# Upload slike



```
index.html  admin.py  views.py  meetup-details.html
meetups > models.py > ...
2  from enum import unique
3  from pydoc import describe
4  from django.db import models
5
6  # Create your models here.
7
8  class Meetup(models.Model):
9      title = models.CharField(max_length=200)
10     slug = models.SlugField(unique=True)
11     description = models.TextField()
12     image = models.ImageField(upload_to='images')
13
```



## Hostovanje statičkih fajlova

```
index.html  admin.py  views.py  meetup-details.html  meetup-item.html
mysite > urls.py > ...
1  from django.contrib import admin
2  from django.urls import path, include
3  from django.conf.urls.static import static
4  from django.conf import settings
5
6  urlpatterns = [
7      path('admin/', admin.site.urls),
8      path('', include('meetups.urls'))
9  ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
10
```

```
ml  admin.py  views.py  meetup-details.html  meetup-item.html  models.py
mysite > settings.py > ...
115
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.1/howto/static-files/
118
119 STATIC_URL = 'static/'
120
121 # Default primary key field type
122 # https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
123
124 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
125
126 MEDIA_ROOT = BASE_DIR / 'uploads'
127 MEDIA_URL = '/files/'
128
```

# One-To-Many relacija

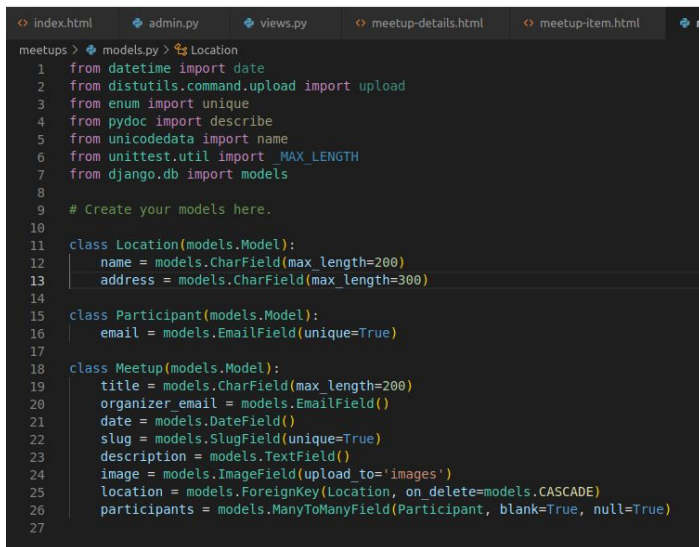
- Definisanje polja lokacija kao **ForeignKey**

```
meetups > admin.py
1 from django.contrib import admin
2
3 from .models import Meetup, Location
4
5 # Register your models here.
6 admin.site.register(Meetup)
7 admin.site.register(Location)
```

```
meetups > models.py
1 from distutils.command.upload import upload
2 from enum import unique
3 from pydoc import describe
4 from unicodedata import name
5 from unittest.util import MAX_LENGTH
6 from django.db import models
7
8 # Create your models here.
9
10 class Location(models.Model):
11     name = models.CharField(max_length=200)
12     address = models.CharField(max_length=300)
13
14 class Meetup(models.Model):
15     title = models.CharField(max_length=200)
16     slug = models.SlugField(unique=True)
17     description = models.TextField()
18     image = models.ImageField(upload_to='images')
19     location = models.ForeignKey(Location, on_delete=models.CASCADE)
20
```

# Dodavanje Participant-a

Izmene u okviru meetup-item, meetup-details da prikaze pravilno



```
index.html  admin.py  views.py  meetup-details.html  meetup-item.html  m
meetups > models.py > Location
1  from datetime import date
2  from distutils.command.upload import upload
3  from enum import unique
4  from pydoc import describe
5  from unicodedata import name
6  from unittest.util import MAX_LENGTH
7  from django.db import models
8
9  # Create your models here.
10
11 class Location(models.Model):
12     name = models.CharField(max_length=200)
13     address = models.CharField(max_length=300)
14
15 class Participant(models.Model):
16     email = models.EmailField(unique=True)
17
18 class Meetup(models.Model):
19     title = models.CharField(max_length=200)
20     organizer_email = models.EmailField()
21     date = models.DateField()
22     slug = models.SlugField(unique=True)
23     description = models.TextField()
24     image = models.ImageField(upload_to='images')
25     location = models.ForeignKey(Location, on_delete=models.CASCADE)
26     participants = models.ManyToManyField(Participant, blank=True, null=True)
27
```

# Forma

```
registration-confirmation.css  meetup-details.html X  meetup-item.html  models.py  forms.py  U
meetups > templates > meetups > meetup-details.html > article > section#registration > form
30  <p>{{ meetup.description }}</p>
31  <footer>
32  |   <p>Need more details? Please <a href="">contact the organizer</a> (but don't spam
33  |   </p>
34  </section>
35
36  <section id="registration">
37  |   <h2>Join Us!</h2>
38  |   <form action="{% url 'meetup-details' meetup.slug %}" method="POST">
39  |   |   {% csrf_token %}
40  |   |   <ul>
41  |   |   |   {{form.as_ul}}
42  |   |   </ul>
43  |   |   <div id="registration-actions">
44  |   |   |   <button>Register</button>
45  |   |   </div>
46  |   </form>
47  </section>
48 </article>
49 {% else %}
50 <p>No meetup found for this slug, sorry!</p>
51 {% endif %}
52
53
54
55 {% endblock %}
```

```
registration-success.html  # registration-confirmation.css  > meet
meetups > forms.py > ...
1  from django import forms
2
3  from .models import Participant
4
5  class RegistrationForm(forms.ModelForm):
6  |   class Meta:
7  |   |   model = Participant
8  |   |   fields = ['email']
```

```
admin.py  views.py  urls.py  meetups  registration-success.html  # registration-confirmation

meetups > views.py > confirm_registration

15 def meetup_details(request, meetup_slug):
16     try:
17         selected_meetup = Meetup.objects.get(slug=meetup_slug)
18
19         if request.method == 'GET':
20             registration_form = RegistrationForm()
21
22         else:
23             registration_form = RegistrationForm(request.POST)
24             if registration_form.is_valid():
25                 participant = registration_form.save()
26                 selected_meetup.participants.add(participant)
27                 return redirect('confirm_registration')
28
29         return render(request, 'meetups/meetup-details.html', {
30             'meetup_found': True,
31             'meetup': selected_meetup,
32             'form': registration_form
33         })
34
35     except Exception as exc:
36         return render(request, 'meetups/meetup-details.html', {
37             'meetup_found': False
38         })
39
40
41 def confirm_registration(request):
42     return render(request, 'meetups/registration-success.html')
```

```
admin.py  views.py  urls.py  meetups  registration-success.html  # registration-confirmation

meetups > urls.py > ...

1 from . import views
2 from django.urls import path
3
4
5 urlpatterns = []
6 path('meetups', views.index, name='all-meetups'), # our-domain.com/meetups
7 path('meetups/success', views.confirm_registration, name='confirm_registration'),
8 path('meetups/<slug:meetup_slug>', views.meetup_details, name='meetup-details') #our
9
```

# Email

```
views.py  urls.py meetups X  registration-success.html  # registration-confirmation.css  meetup-details.html  meetup-ite

meetups > • urls.py > ...
1  from . import views
2  from django.urls import path
3
4
5  urlpatterns = [
6      path('meetups', views.index, name='all-meetups'), # our-domain.com/meetups
7      path('meetups/<slug:meetup_slug>/success', views.confirm_registration, name='confirm_registration'),
8      path('meetups/<slug:meetup_slug>', views.meetup_details, name='meetup-details') #our-domain.com/meetups
9  ]
```

```
42
43  def confirm_registration(request, meetup_slug):
44      meetup = Meetup.objects.get(slug=meetup_slug)
45      return render(request, 'meetups/registration-success.html', {
46          'organizer_email':meetup.organizer_email
47      })
```

meetups &gt; views.py &gt; meetup\_details

```
14     })
15
16 def meetup_details(request, meetup_slug):
17     try:
18         selected_meetup = Meetup.objects.get(slug=meetup_slug)
19
20         if request.method == 'GET':
21             registration_form = RegistrationForm()
22
23         else:
24             registration_form = RegistrationForm(request.POST)
25             if registration_form.is_valid():
26                 user_email = registration_form.cleaned_data['email']
27                 participant, _ = Participant.objects.get_or_create(email=user_email)
28                 selected_meetup.participants.add(participant)
29                 return redirect('confirm_registration', meetup_slug=meetup_slug)
30
31             return render(request, 'meetups/meetup-details.html', {
32                 'meetup_found': True,
33                 'meetup': selected_meetup,
34                 'form': registration_form
35             })
36
37     except Exception as exc:
38         return render(request, 'meetups/meetup-details.html', {
39             'meetup_found': False
```

## Form VS. ModelForm

```
registration-success.html  # registration-confirmation.css  <> meetup-details.htm

meetups > forms.py > RegistrationForm
1  from django import forms
2
3  from .models import Participant
4
5  class RegistrationForm(forms.Form):
6      email = forms.EmailField(label='Your email')
```



# Redirekcija

```
mysite > urls.py > ...
1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf.urls.static import static
4 from django.conf import settings
5 from django.views.generic.base import RedirectView
6
7 urlpatterns = [
8     path('admin/', admin.site.urls),
9     path('', RedirectView.as_view(url='/meetups/')),
10    path('meetups/', include('meetups.urls'))
11 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
12
```

```
meetups > urls.py > ...
1 from . import views
2 from django.urls import path
3
4
5 urlpatterns = [
6     path('', views.index, name='all-meetups'), # our-domain.com/meetups
7     path('<slug:meetup_slug>/success', views.confirm_registration, name='confirm_registration'),
8     path('<slug:meetup_slug>', views.meetup_details, name='meetup-details') #our-domain.com/meetups/<meeti
9 ]
```

## JSON serijalizacija

- JSON (JavaScript Object Notation) je najčešće upotrebljen format za razmenu podataka.
- Ugrađena JSON serijalizacija:

```
def get_meetups(request):  
    meetup = Meetup.objects.all().values()  
    meetup_list = list(meetup)  
    return JsonResponse(meetup_list, safe=False)
```

- Dobavljanje podataka iz baze podataka podrazumeva QuerySet tip podatka, dok JsonResponse prima samo listu objekata, te je potrebno izvršiti konverziju

## Kreiranje serijalizatora

- Na prethodnom primeru prikazana je opšta serijalizacija - svaki objekat se serijalizuje u odgovarajući JSON sa svim poljima u okviru objekta
- Ukoliko se kreiraju posebni serijalizatori za odgovarajući model, moguće je modifikovati JSON objekat koji nastaje serijalizacijom (prikazati samo određena polja u okviru objekta)
- `pip install djangorestframework + 'rest_framework' in INSTALLED_APPS`

```
def get_locations(request):  
    location = Location.objects.all()  
    location_list = LocationSerializer(location, many=True)  
    return JsonResponse(location_list.data, safe=False)
```

```
serializers.py  settings.py  urls.py  meetups  
serializers.py > LocationSerializer > Meta  
from rest_framework import serializers  
  
from meetups.models import Location  
  
class LocationSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Location  
        fields = ['name', 'address']
```

## Deserijalizacija JSON-a

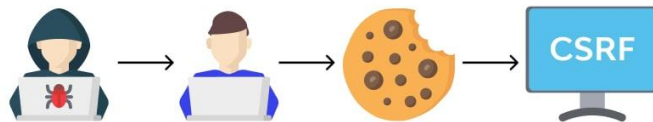
- Suprotna operacija od serijalizacije

- `from django.views.decorators.csrf import csrf_exempt`

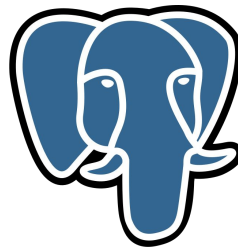
```
@csrf_exempt
def new_location(request):
    data = json.loads(request.body)
    Location.objects.create(name=data['name'], address=data['address'])
    return HttpResponse(status=200)
```

# CSRF

CSRF - Cross site request forgery  
attack



# PostgreSQL



- besplatan i open-source RDBMS
- pgAdmin je najpopularnija open-source administrativna i razvojna Platforma za PostgreSQL
- Povezivanje Django projekta sa PostgreSQL (settings.py):

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql_psycopg2',  
        'NAME': '<database_name>',  
        'USER': '<database_username>',  
        'PASSWORD': '<database_password>',  
        'HOST': '<database_host>',  
        'PORT': '<database_port>',  
    }  
}
```

## Povezivanje projekta sa Postgres bazom

1. Instaliranje postgresql
2. Instaliranje pgAdmin
3. Postavljanje novog password-a sledećim komandama:
  - a. `$sudo -i -u postgres`
  - b. `$psql`
  - c. `$\password`
  - d. Unošenje novog password-a
4. Kreiranje nove baze podataka ili upotreba postojeće postgres baze podataka (pgadmin ili komanda `create database [database_name];`)
5. U okviru pgAdmina registrovanje novog servera
6. Primena kreiranih migracija

## Zadatak za vežbu - mini acs sajt

- Napraviti novi Django projekat sa jednom aplikacijom
- Aplikacija se sastoji iz sledećih modela:
  - Predmet: naziv, smer na kome se predaje, profesor, studenti
  - Smer: naziv
  - Profesor: ime i prezime, email
  - Student: ime i prezime, email
- Jedan predmet se može držati na više smerova
- Na jednom predmetu radi jedan profesor
- U bazu uneti nekoliko testnih podataka
- Napraviti html stranicu koja izlistava sve predmete
- Kada se klikne na jedan od predmeta odlazi se na drugu stranicu koja prikazuje smerove na kojima se ti predmeti predaju, kao i profesora koji predaje
- Na stranici koja prikazuje detalje predmeta postoji forma preko koje se student može prijaviti da mu na email stižu obaveštenja koja se objave na sajtu



## Materijali:

- <https://docs.djangoproject.com/en/4.1/intro/tutorial01/>
- <https://www.igordejanovic.net/courses/tech/django/>
- <https://www.youtube.com/watch?v=t7DrJqcUviA&t=7685s>