# Lecture 5 - Django Views

**COMP 8347**

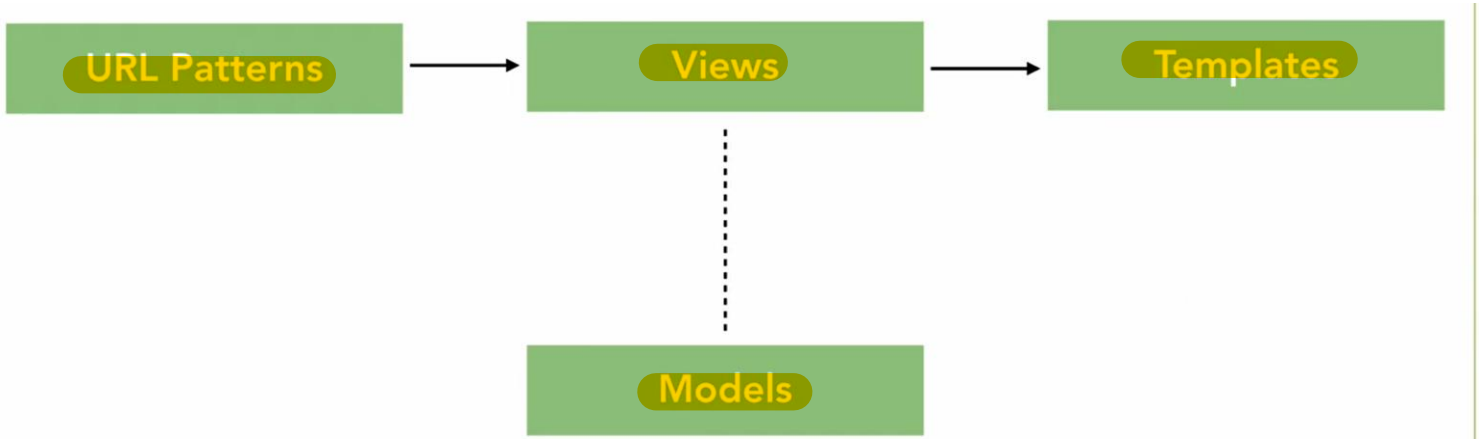
Usama Mir

Usama.Mir@unwindsor.ca

# Django Views

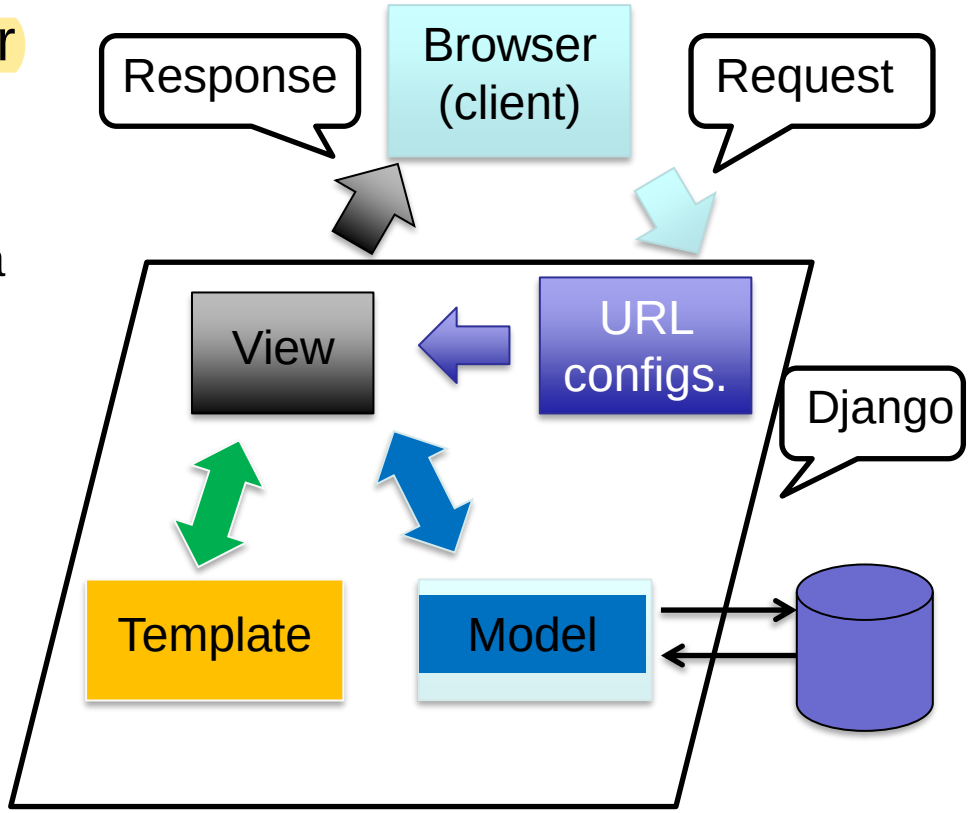
- Topics
  - URLS
  - HTTP Objects
    - Request
    - Response
  - Views

# Review MTV Architecture

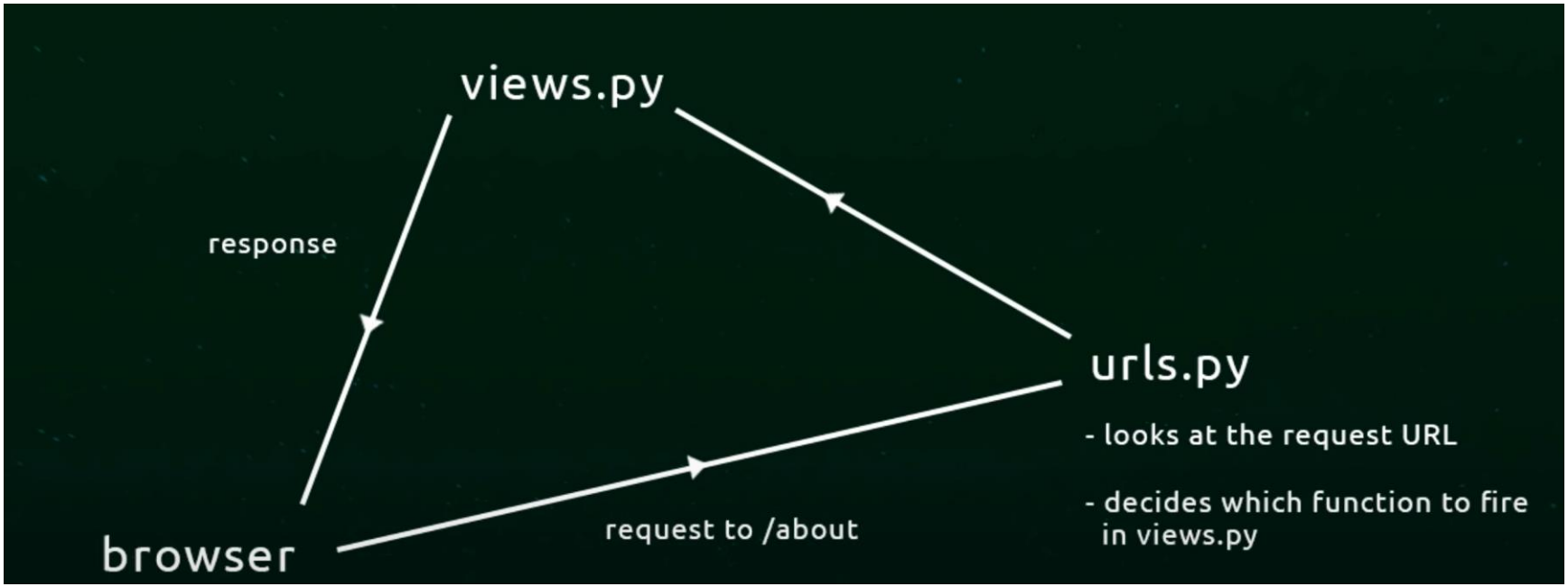


# Choosing a View (Function)

- Django web pages and other content are delivered by views.
  - Each view is represented by a simple Python function (or method)
- Django chooses a view by examining the requested URL
  - Only looks at the part of URL after the domain name.
  - Chooses view that 'matches' associated URL pattern.



# Choosing a View (Function)



# URLconf

- *URLconf (URL configuration): maps between URL path expressions to Python functions (your views).*
- *urlpatterns*: a sequence of Django **paths**

  Example: path('admin/', admin.site.urls),
- URL patterns for your app/project specified in corresponding ***urls.py*** file.

# Sample urls.py

mysite/urls.py

```
from django.urls import include, path
from django.contrib import admin
urlpatterns = [
   path('admin/', admin.site.urls),
   path('myapp/',
  include('myapp.urls')),
]
```

myapp/urls.py

```
from django.urls import path
from myapp import views

app_name = ‘myapp’
urlpatterns = [
   path(‘ ’, views.index, name='index'),
]
```

- include(*module*, *namespace=None*)
  - urlpatterns can “include” other URLconf modules.
  - This “roots” a set of URLs below other ones
  - When Django encounters include():
    - it chops off part of the URL matched up to that point
    - sends the remaining string to the included URLconf for further processing
  - It is a good practice to use include() when including other URL patterns
    - Only exception in admin.site.urls

# path()

- Syntax:
  - path(route, view, kwargs=None, name=None)
    - Ex. path('about/', views.about, name='about'),
  - ***route***: a string that contains a URL pattern
    - Ex. urlpatterns = [ path('blog/<int:year>/', views.year_archive]
    - angle brackets may include a converter specification (like the int part of <int:section>) which limits the characters matched and may also change the type of the variable passed to the view
    - Django starts at the first **path**, compares requested URL against each route until it finds one that matches.
      - Does not search domain names

# path()

- Syntax:
  - **path(route, view, kwargs=None, name=None)**
  - ***view***: after finding match, Django calls specified view function, with
    - HttpRequest object as the first argument and
    - any "captured" values from the regular expression as other arguments.
  - ***kwargs*** : can pass additional arguments in a dict, to view function.
    - Ex. https://example.com/path/to/page?name=ferret&color=purple
  - ***name***: optional – used to create links

# path()

- Examples:

```
from django.urls import include, path

urlpatterns = [
    path('index/', views.index, name='main-view'),
    path('bio/<username>/', views.bio, name='bio'),
    path('articles/<slug:title>/', views.article,
                name='article-detail'),
    path('articles/<slug:title>/<int:section>/', views.section,
                name='article-section'),
...
]
```

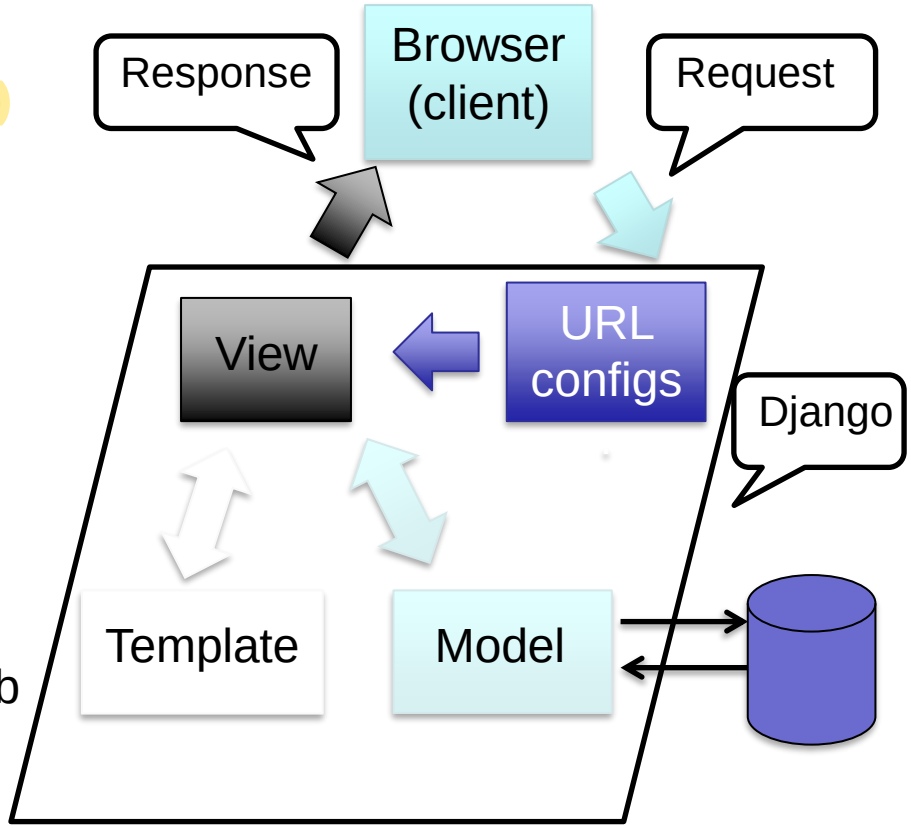
- A **slug** is a short label for something, containing only letters, numbers, underscores or hyphens. They're generally used in URLs.
- Example: https://www.semrush.com/blog/what-is-a-url-slug/

# URL Matching Examples

```
from django.conf.urls import patterns, path
from myapp import views

urlpatterns = [
# ex: /myapp/

# ex: /myapp/5/

# ex: /myapp/5/results/

]
```

# Web Application Flow

- HTTP request arrives at web server
- Web server passes request to Django
- Django creates a request object
- Django consults URLconf to find right view function
  - Checks url against each path
- View function is called with request object and captured URL arguments
- View creates and returns a response object.
- Django returns response object to web server.
- Web server responds to requesting client.



# Views

- ***View function*: A Python function - takes a Web request, returns a Web response.**
  - called view for short
  - response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image . . .
  - provide nearly all the programming logic
    - perform CRUD operations
  - can reside anywhere in your Python path
    - convention is to put views in a file called **views.py**, placed in your project or application directory

# View: Example 1

```
from django.shortcuts import render
from django.http import HttpResponse

def members(request):
    return HttpResponse("Hello world!")
```

```
urlpatterns = [
    path('', views.members, name='members'),
```

# View: Example 2

models.py | views.py | urls.py

```python
from django.http import HttpResponse
from myapp2.models import Book

# Create your views here.
def index(request):
    return HttpResponse('Index')

def about(request):
    return HttpResponse('This is an about page')
```

models.py | views.py | urls.py

```python
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.urls import path
from myapp2 import views
from django.contrib import admin

app_name = 'myapp2'
urlpatterns = [
    path('', views.index, name='index'),
    path('about/', views.about, name='about'),
    path('admin/', admin.site.urls)
]
```

# Request Objects

- *HttpRequest*: An object with a set of attributes representing raw HTTP request
  - *GET*: An attribute of HttpRequest Object
    - represented as a Python dict subclass QueryDict.
    - GET parameters passed as URL string, but not part of URL itself; do not define a separate resource (view)
    - Example: for the URL /userinfo/ can point to specific user: /userinfo/?name='John Smith'

      **username = request.GET['name']**

# HttpRequest Attributes

- *POST*: An attribute of HttpRequest Object
  - represented as a QueryDict.
  - POST parameters are not part of URL
  - often generate by an HTML form; when user submits form, URL is called with POST dict containing form fields.
  - Example: if there is a form field 'name' and the user enters 'John'

    **request.POST['name']** will return 'John'

- *COOKIES*: Another dict attribute; exposes HTTP cookies stored in request.
  - Ex. On this link: https://www.javatpoint.com/django-cookie

# Other Attributes

- *path*: portion of URL after domain
- *method*: specifies which request method was used – 'GET' or 'POST'
- meta: All available HTTP headers are shown by this attribute.
- *files*: contains information about any files uploaded by a file input form field.

And so on…..

# Response Objects

- View functions return a HttpResponse object. Important attributes:
  - HttpResponse with content : A bytestring representing the content; usually a large HTML string.
  - can be set when creating a response object
    - response = HttpResponse("<html>Hello World</html>")
  - can be set using write method (like a file)
    - response = HttpResponse()
    - response.write("<html>")
    - response.write("Hello World")
    - response.write("</html>")

# Response Objects

- Setting HTTP headers:
  - Treat response object as a dictionary.
  - 'key/value' pairs correspond to different headers and corresponding values.
    - HTTP header fields cannot contain newlines.
  - Example:

```
response = HttpResponse()
response["Content-Type"] = "text/csv"
response["Content-Length"] = 256
```

# HttpResponse Subclasses

- Django provides HttpResponse subclasses for common response types.
  - *HttpResponseForbidden*: uses HTTP 403 status code
  - *HttpResponseServerError*: for HTTP 500 or internal server errors
  - *HttpResponseRedirect*: the path to redirect to
  - *HttpResponseBadRequest*: acts like HttpResponse, but uses a 400 status code
  - *HttpResponseNotFound*: acts like HttpResponse, but uses a 404 status code

# References


- Lectures of Dr. Saja and Dr. Arunita
- https://docs.djangoproject.com/en/4.2/topics/http/views/
- https://docs.djangoproject.com/en/4.2/ref/request-response/
- https://www.youtube.com/watch?v=TblSa29DX6I
- https://www.w3schools.com/django/django_views.php#:~:text=Django%20views%20are%20Python%20functions,located%20on%20your%20app's%20folder
- https://programtalk.com/python-examples/django.http.HttpResponseForbidden/