

## Got Goat Chapter 4 Displaying Database Data

We have created the model schema in the Driver app, the table in database, and entered records. We also discovered how to perform popular CRUD operations. In this chapter, we will create a template and view for displaying dynamic content on the screen.

### 1. Create HTML file

- a. Navigate to the drivers app/folder and create a new HTML file inside the templates folder similar to below.



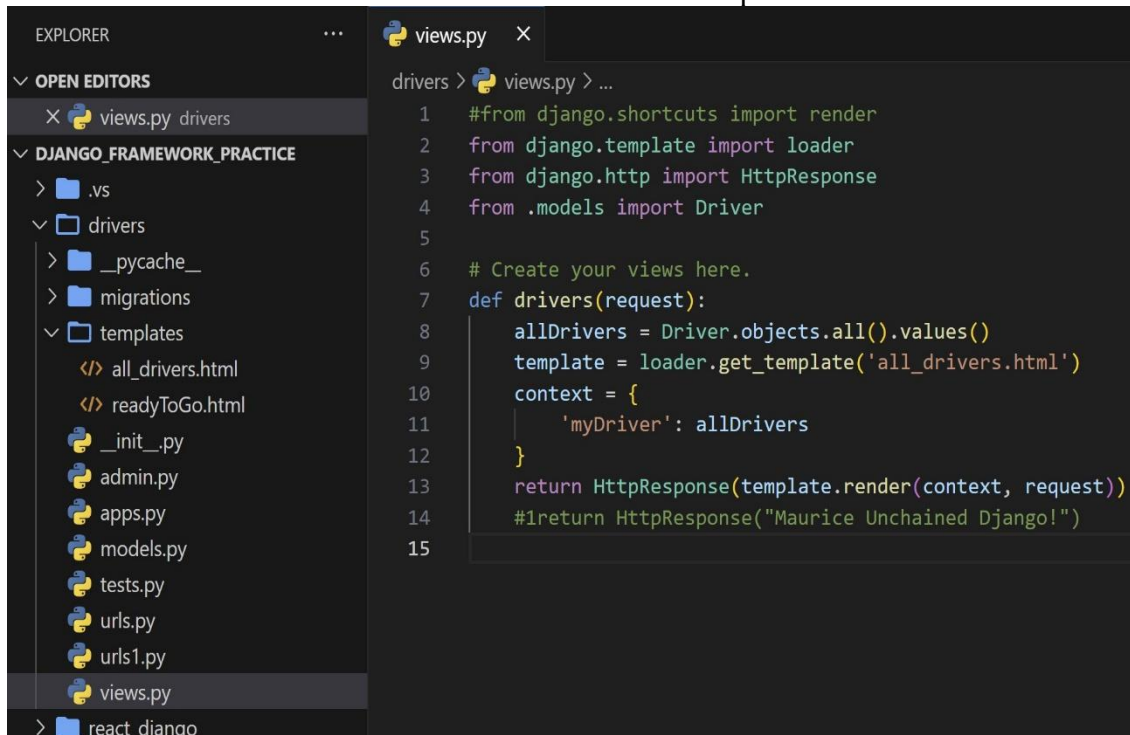
The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays the project structure under 'DJANGO\_FRAMEWORK\_PRACTICE', with the 'drivers' folder expanded to show a 'templates' subfolder. The 'all\_drivers.html' file is selected. The main editor pane shows the content of 'all\_drivers.html' with the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>Drivers</h1>
5 <ul>
6     {%for x in myDriver%}
7     <li>{{x.firstName}} {{x.lastName}} {{x.truckNumber}} {{x.hireDate}}</li>
8     {%endfor%}
9 </ul>
10 </body>
11 </html>
12
```

- b.
  - i. Line 6: The double “%” instructs the view to perform programming logic inside the brackets. More specifically, the iterator will list the specified attributes of each record.

### 2. Update View

- a. We need to make the database model available to the template.

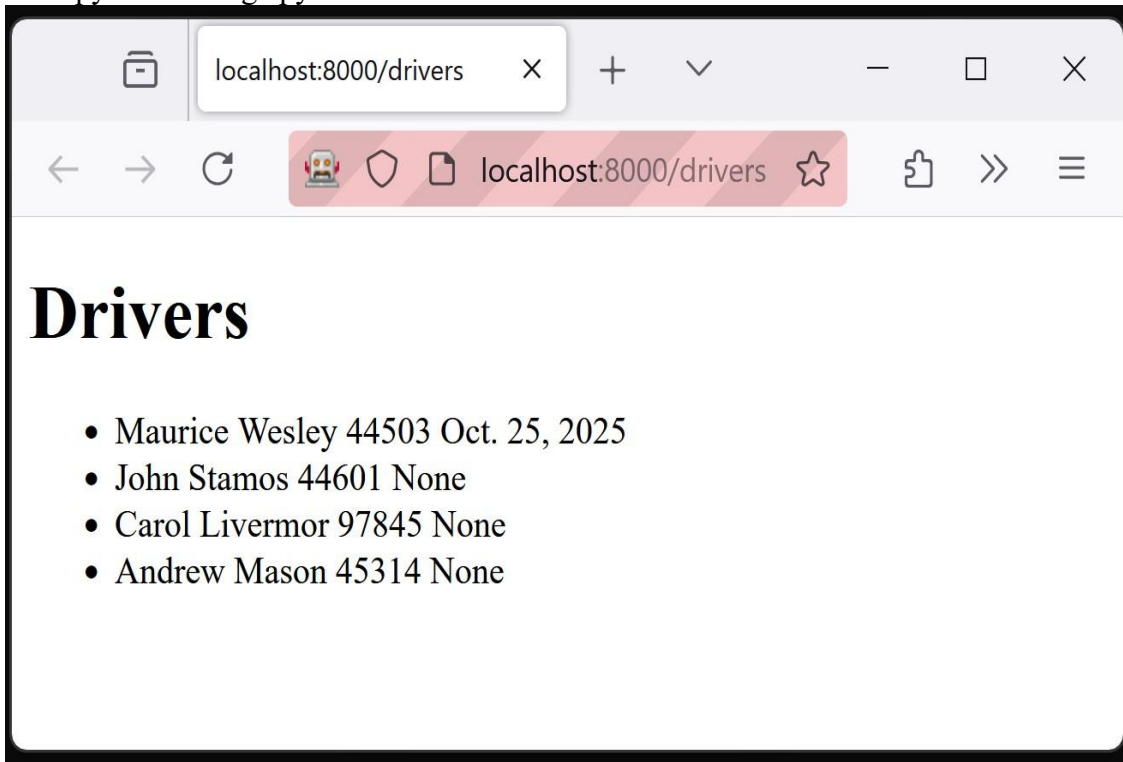


The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays the project structure under 'DJANGO\_FRAMEWORK\_PRACTICE', with the 'drivers' folder expanded to show the 'templates' subfolder. The 'views.py' file is selected. The main editor pane shows the content of 'views.py' with the following code:

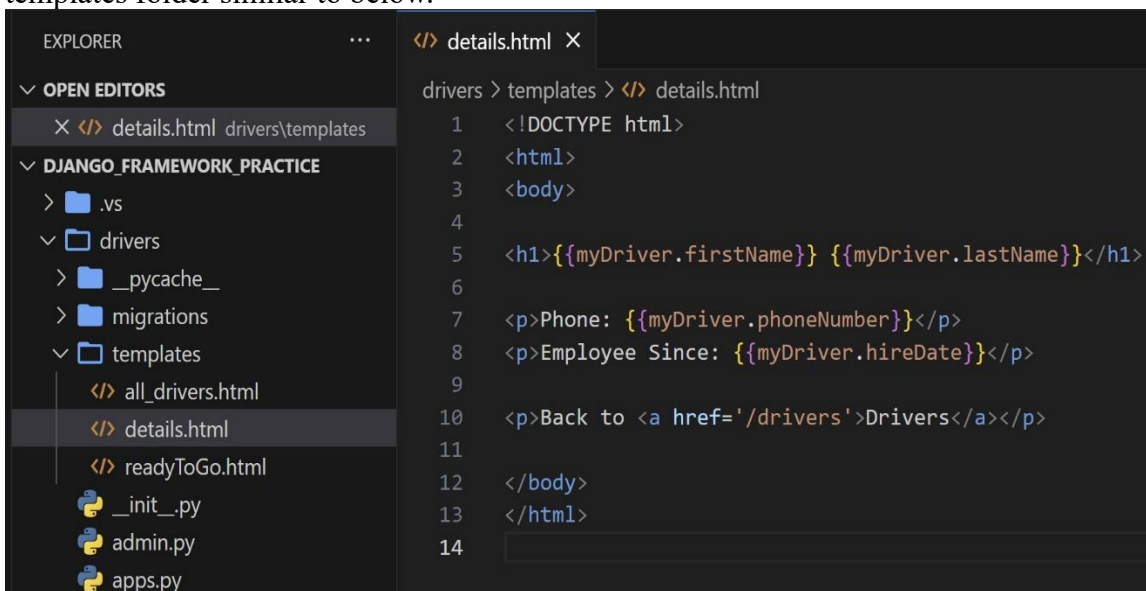
```
1 #from django.shortcuts import render
2 from django.template import loader
3 from django.http import HttpResponse
4 from .models import Driver
5
6 # Create your views here.
7 def drivers(request):
8     allDrivers = Driver.objects.all().values()
9     template = loader.get_template('all_drivers.html')
10     context = {
11         'myDriver': allDrivers
12     }
13     return HttpResponse(template.render(context, request))
14 #1return HttpResponse("Maurice Unchained Django!")
15
```

- b.
  - i. Line 4: We are importing the Driver model from driver app models folder
  - ii. Line 8: We are assigning the values of the model to a variable passed in response

- iii. Line 11: The programming logic (myDriver) has to reference an object(allDrivers) for the iterator (for loop in html).
3. Display Records
- a. Run: `python manage.py runserver`



- b.
4. Details HTML
- a. Navigate to the drivers app/folder and create a new HTML file called 'details' inside the templates folder similar to below.

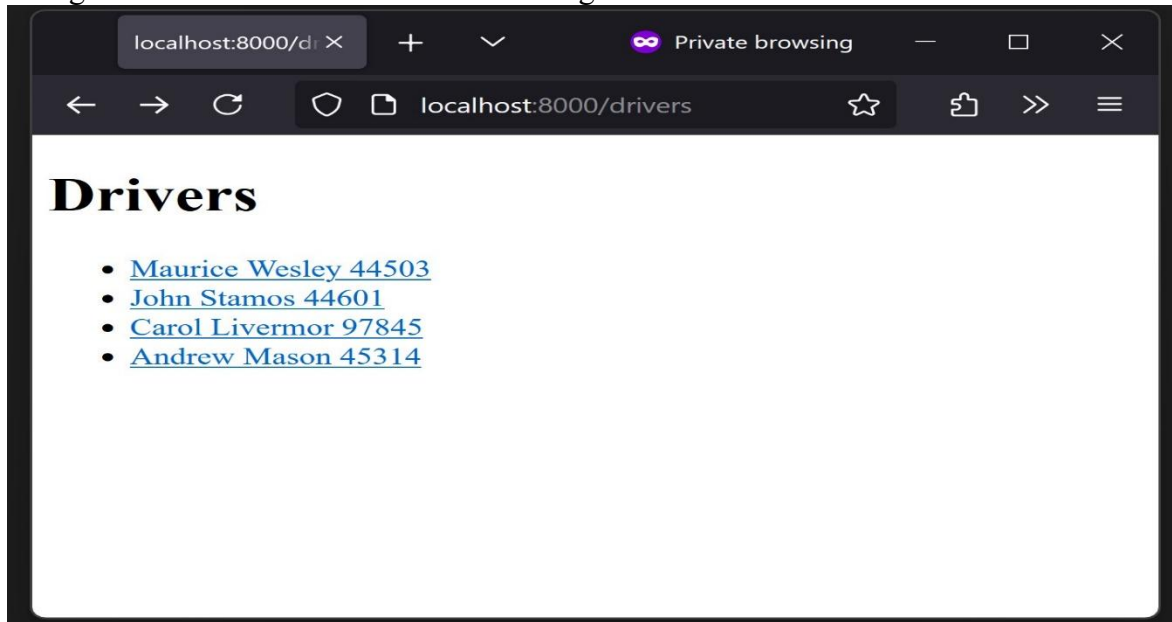


- b.
5. Link To Details
- a. Modify the `all_drivers.html` to link the first and last name to the details page

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'drivers' folder containing 'all\_drivers.html', 'details.html', and 'readyToGo.html'. The code editor shows the content of 'all\_drivers.html' with the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <h1>Drivers</h1>
5 <ul>
6     {% for x in myDrivers %}
7     <li><a href='drivers/details/{{x.id}}'>{{x.firstName}} {{x.lastName}} {{x.truckNumber}}</a></li>
8     {% endfor %}
9 </ul>
10 </body>
11 </html>
12
```

- b.
- c. Navigate over to the terminal to see the changes



- d.
- e. Clicking on the link will generate an error. We have not set up the routing to pass the data from the model to a view.

## 6. Create Detail View

- a. Navigate to the views.py file and add the following

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'drivers' folder containing 'all\_drivers.html', 'details.html', and 'readyToGo.html'. The code editor shows the content of 'views.py' with the following code:

```
1 #from django.shortcuts import render
2 from django.template import loader
3 from django.http import HttpResponse
4 from .models import Driver
5
6 # Create your views here.
7 def drivers(request):
8     allDrivers = Driver.objects.all().values()
9     template = loader.get_template('all_drivers.html')
10     context = {
11         'myDrivers': allDrivers
12     }
13     return HttpResponse(template.render(context, request))
14     #!return HttpResponse("Maurice Unchained Django!")
15
16 def details(request):
17     aDriver = Driver.objects.get(id=id)
18     template = loader.get_template('details.html')
19     context = {
20         'myDriver': aDriver
21     }
22     return HttpResponse(template.render(context, request))
23
```

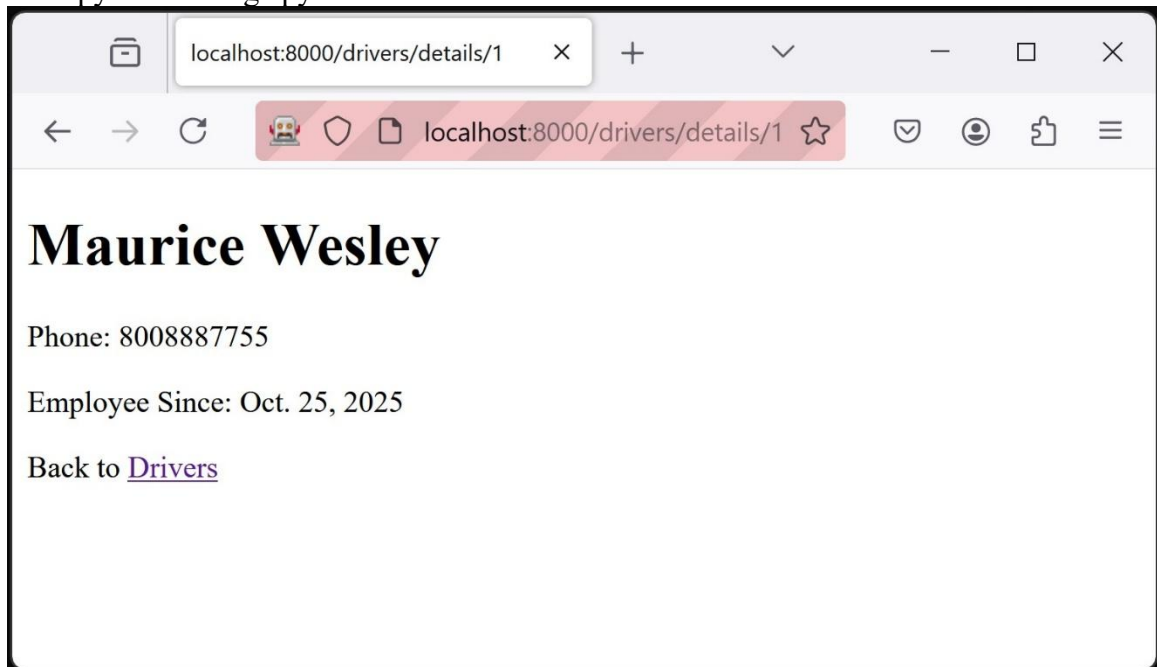
- b.

## 7. Add URL Routing

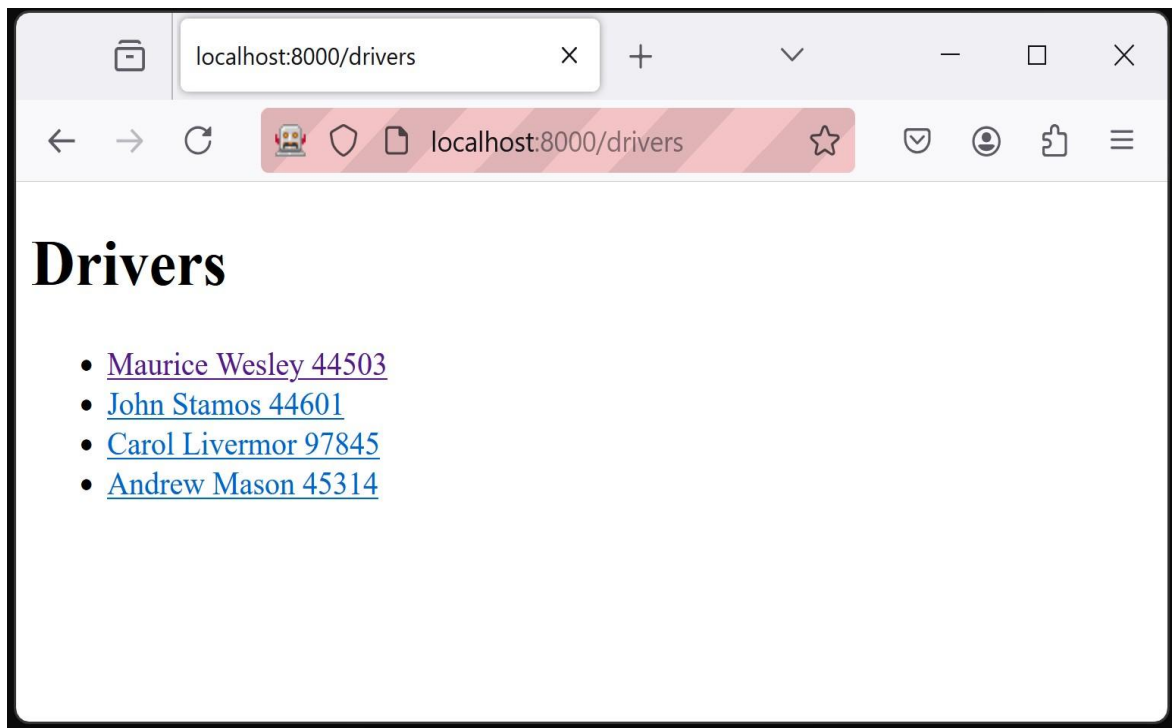
- a. The details URL has to link to the correct view. Open the URLs file and add the following.

```
urls1.py ×
drivers > urls1.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path("drivers",views.drivers, name='drivers'),
6      path("drivers/details/<int:id>", views.details, name='details')
7  ]
8
```

- b.
8. Display Drivers and link to details
- a. Run: python manage.py runserver



- b.
- i. We can celebrate the above. The URL displays the page and contents. The driver's table attribute values are displayed with the templated format.
- c. Pressing the link 'Drivers' will navigate the user back to the home page.



- d.
- The first driver has a different color meaning that the link has been visited.

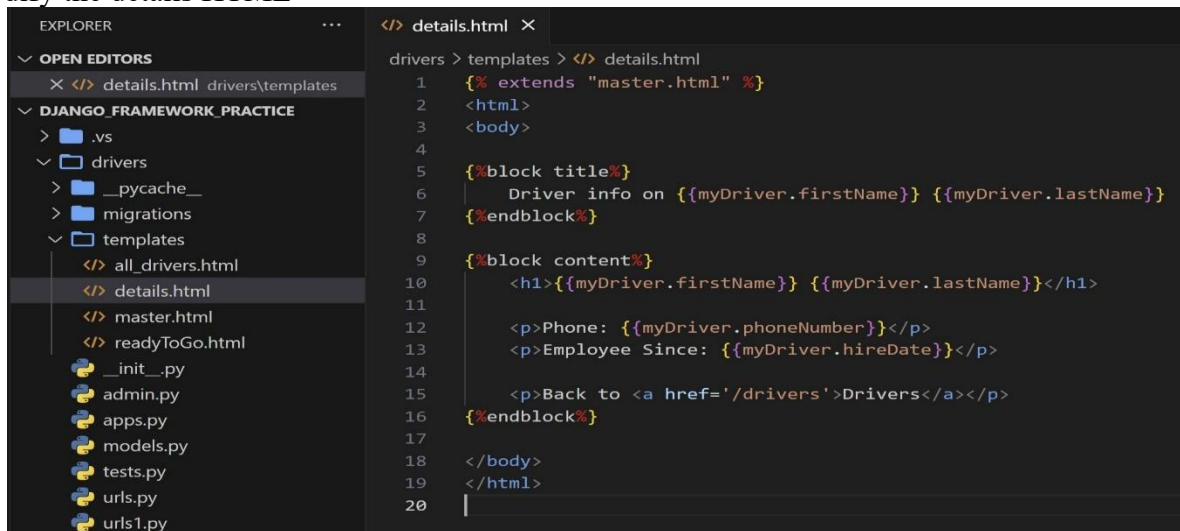
9. Master Template

- Django provides functionality to wrap similar content into blocks. Both the details HTML and all\_drivers HTML have similar structures. We can leverage this similarity into a single template.
- Create a new HTML file in the template folder called 'master.html'. Insert the following.

```
EXPLORER
...
</> master.html X
drivers > templates > </> master.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>
5         {%block title%} {%endblock%}
6     </title>
7 </head>
8
9 <body>
10     {%block content%} {%endblock%}
11 </body>
12
13 </html>
14
```

- c.
- d. Now that we have the master template. We can wrap the detail and all\_drivers HTMLs in programming language blocks. See below.

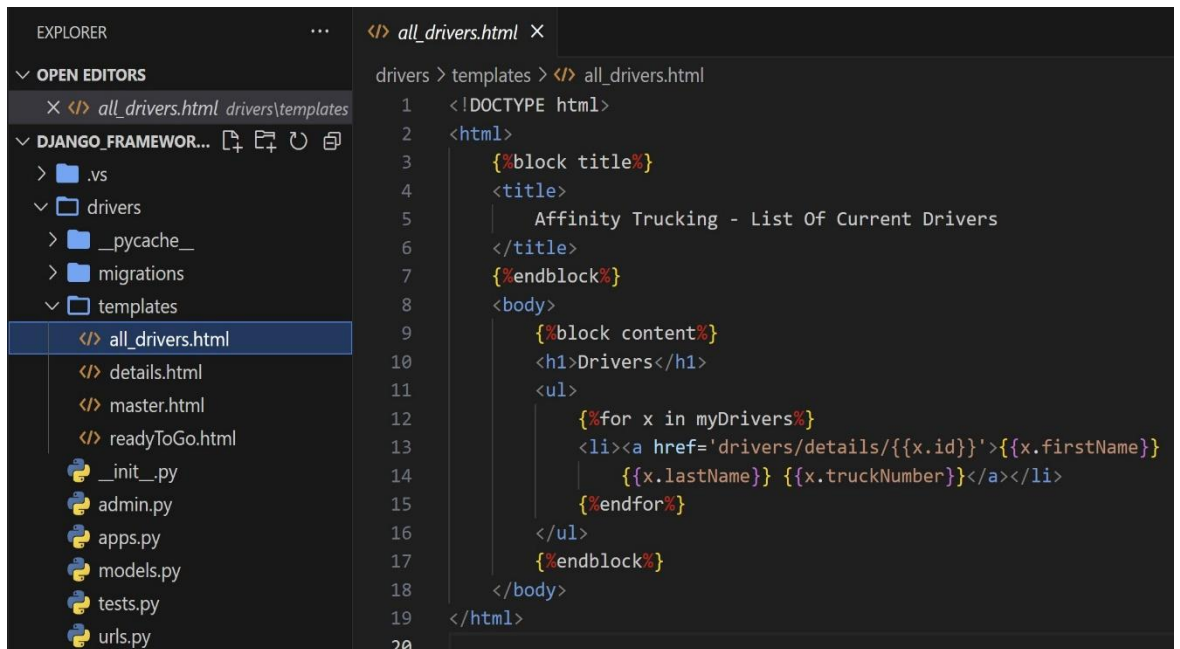
## 10. Modify the details HTML



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left and the Editor window on the right. The Explorer sidebar shows the project structure with the 'templates' folder expanded, and 'details.html' selected. The Editor window displays the content of 'details.html'.

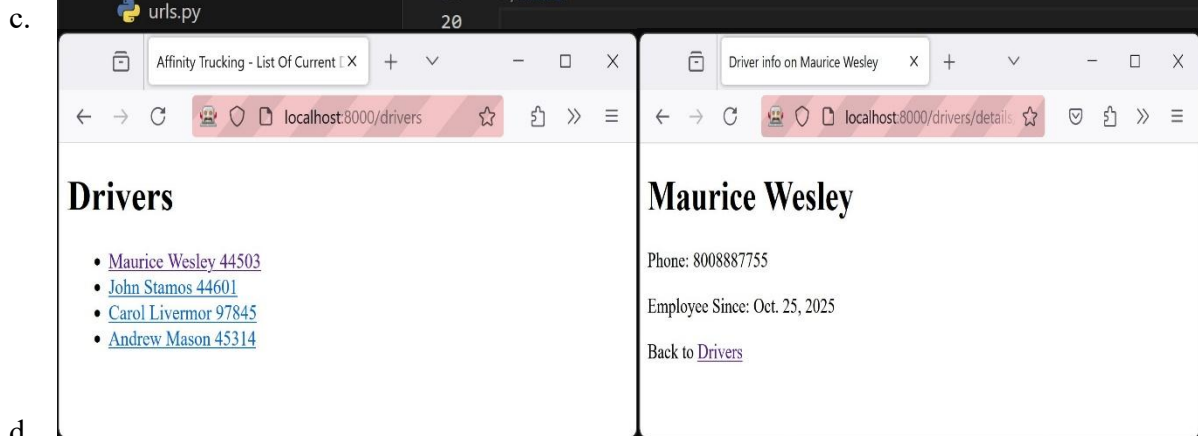
```
drivers > templates > details.html
1  {% extends "master.html" %}
2  <html>
3  <body>
4
5  {%block title%}
6      Driver info on {{myDriver.firstName}} {{myDriver.lastName}}
7  {%endblock%}
8
9  {%block content%}
10     <h1>{{myDriver.firstName}} {{myDriver.lastName}}</h1>
11
12     <p>Phone: {{myDriver.phoneNumber}}</p>
13     <p>Employee Since: {{myDriver.hireDate}}</p>
14
15     <p>Back to <a href="/drivers">Drivers</a></p>
16  {%endblock%}
17
18 </body>
19 </html>
20
```

- a.
- b. The model data output and format are wrapped in blocks. Proceed to edit the all\_drivers HTML.



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left and the Editor window on the right. The Explorer sidebar shows the project structure with the 'templates' folder expanded, and 'all\_drivers.html' selected. The Editor window displays the content of 'all\_drivers.html'.

```
drivers > templates > all_drivers.html
1  <!DOCTYPE html>
2  <html>
3      {%block title%}
4      <title>
5          Affinity Trucking - List Of Current Drivers
6      </title>
7      {%endblock%}
8  <body>
9      {%block content%}
10         <h1>Drivers</h1>
11         <ul>
12             {%for x in myDrivers%}
13                 <li><a href="/drivers/details/{{x.id}}">{{x.firstName}}
14                     {{x.lastName}} {{x.truckNumber}}</a></li>
15             {%endfor%}
16         </ul>
17     {%endblock%}
18 </body>
19 </html>
20
```

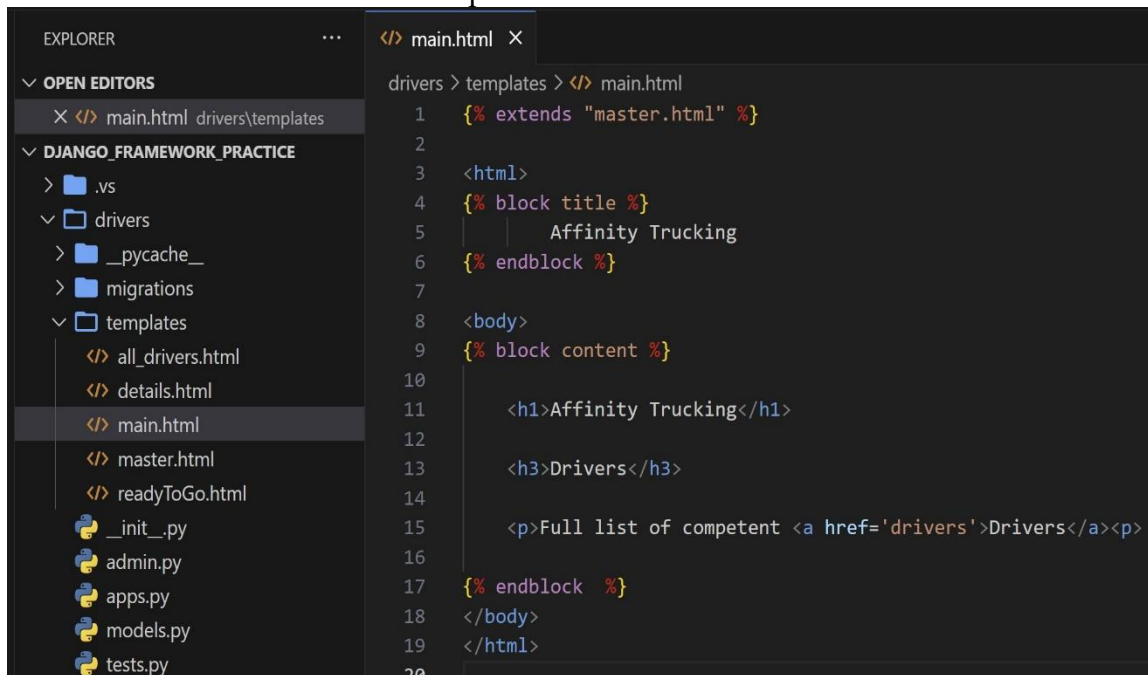


- d.



## 11. Main Index Page

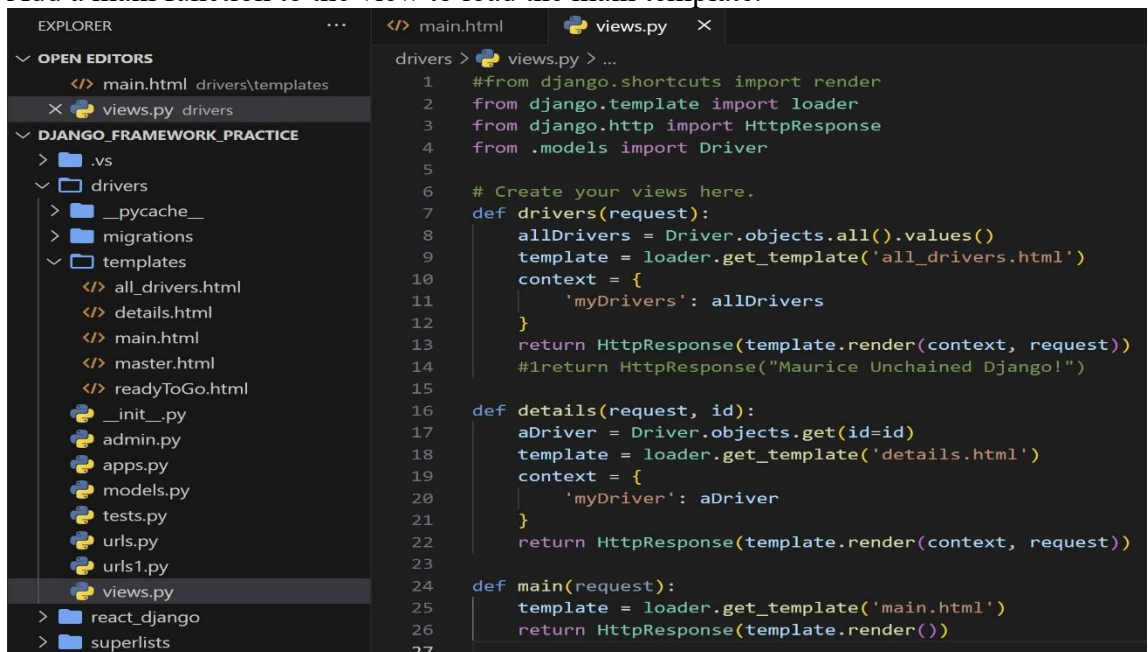
- The main landing page will show an error because our current application does not have a main page.
- Create a main HTML file in the template folder



The screenshot shows the VS Code interface. On the left, the Explorer pane displays the project structure under 'DJANGO\_FRAMEWORK\_PRACTICE'. The 'templates' folder is expanded, showing files: 'all\_drivers.html', 'details.html', 'main.html' (selected), 'master.html', and 'readyToGo.html'. On the right, the editor shows the content of 'main.html'.

```
1 {% extends "master.html" %}
2
3 <html>
4 {% block title %}
5     Affinity Trucking
6 {% endblock %}
7
8 <body>
9 {% block content %}
10
11     <h1>Affinity Trucking</h1>
12
13     <h3>Drivers</h3>
14
15     <p>Full list of competent <a href='drivers'>Drivers</a><p>
16
17 {% endblock %}
18 </body>
19 </html>
```

- 
- 
- 
- Add a main function to the view to load the main template.



The screenshot shows the VS Code interface. On the left, the Explorer pane shows the project structure. The 'views.py' file is selected in the 'drivers' folder. On the right, the editor shows the content of 'views.py'.

```
1 #from django.shortcuts import render
2 from django.template import loader
3 from django.http import HttpResponse
4 from .models import Driver
5
6 # Create your views here.
7 def drivers(request):
8     allDrivers = Driver.objects.all().values()
9     template = loader.get_template('all_drivers.html')
10    context = {
11        'myDrivers': allDrivers
12    }
13    return HttpResponse(template.render(context, request))
14    #return HttpResponse("Maurice Unchained Django!")
15
16 def details(request, id):
17     aDriver = Driver.objects.get(id=id)
18     template = loader.get_template('details.html')
19     context = {
20         'myDriver': aDriver
21     }
22     return HttpResponse(template.render(context, request))
23
24 def main(request):
25     template = loader.get_template('main.html')
26     return HttpResponse(template.render())
27
```

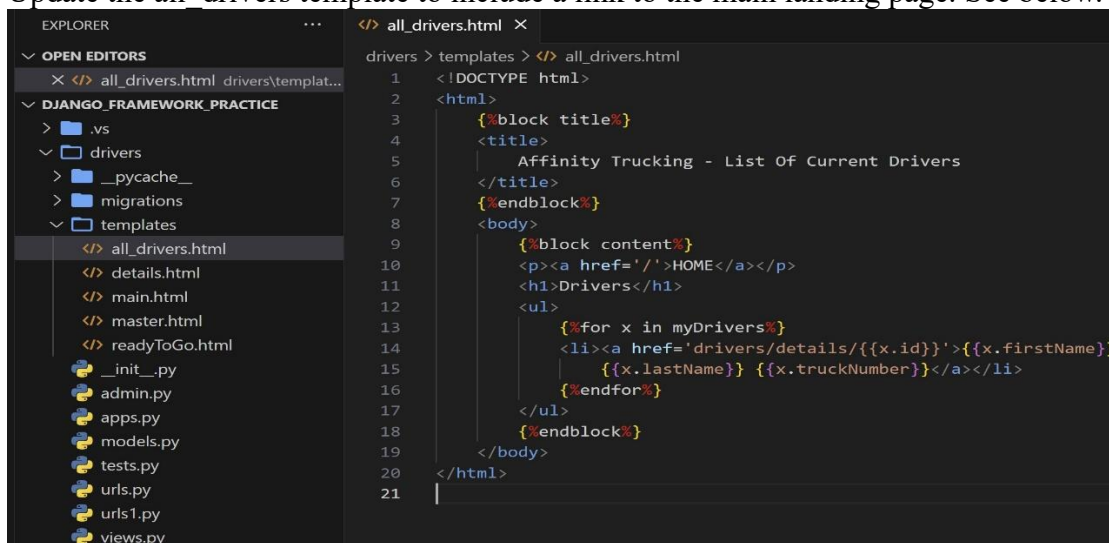
- 
- 
- 
- 
- 
- Add path to the main page

```

urls1.py  X
drivers > urls1.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('', views.main, name='main'),
6      path("drivers", views.drivers, name='drivers'),
7      path("drivers/details/<int:id>", views.details, name='details')
8  ]
9

```

g. h. Update the all\_drivers template to include a link to the main landing page. See below.



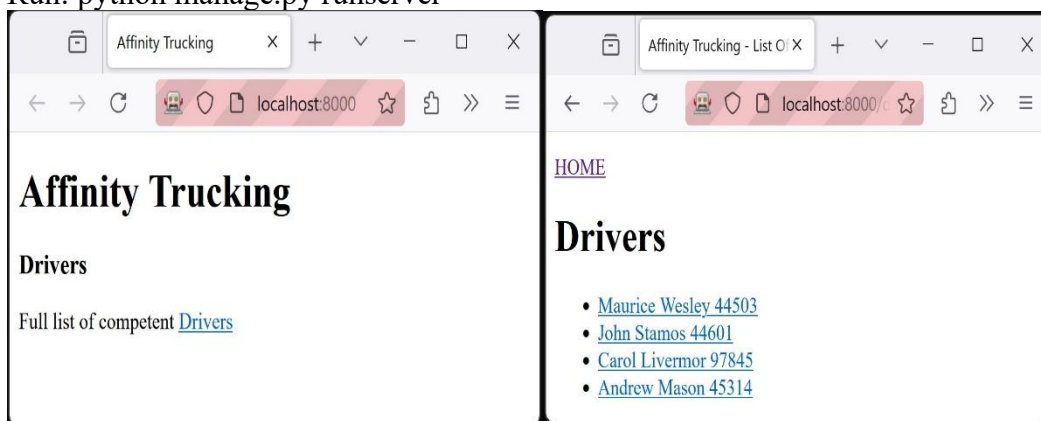
```

EXPLORER
  OPEN EDITORS
    X all_drivers.html drivers\templat...
  DJANGO_FRAMEWORK_PRACTICE
    .vs
    drivers
    _pycache_
    migrations
    templates
      all_drivers.html
      details.html
      main.html
      master.html
      readyToGo.html
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    urls1.py
    views.py

all_drivers.html
1  <!DOCTYPE html>
2  <html>
3      {%block title%}
4      <title>
5          Affinity Trucking - List Of Current Drivers
6      </title>
7      {%endblock%}
8      <body>
9          {%block content%}
10             <p><a href='/'>HOME</a></p>
11             <h1>Drivers</h1>
12             <ul>
13                 {%for x in myDrivers%}
14                 <li><a href='drivers/details/{x.id}'>{{x.firstName}}
15                     {{x.lastName}} {{x.truckNumber}}</a></li>
16                 {%endfor%}
17             </ul>
18             {%endblock%}
19         </body>
20     </html>
21

```

i. j. Run: python manage.py runserver



k. i. Success!!

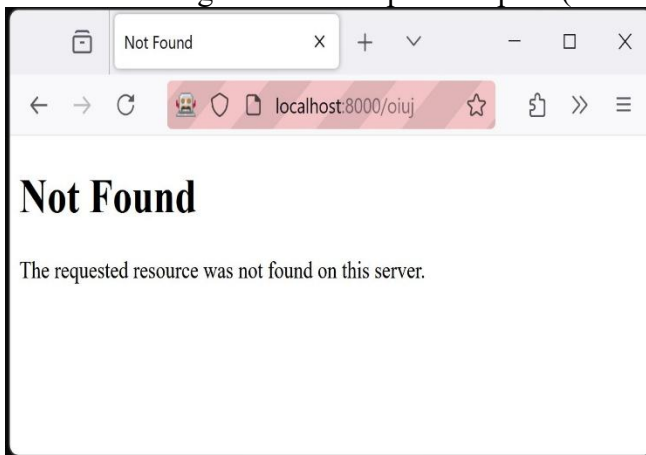
## 12. 404 Error Page

a. Navigating to an unspecified path will yield either a built-in Django error or an error page. To edit the error page, we have to set Debug = False in the settings.py file. See below.



```
settings.py X
superlists > settings.py > ...
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = False
27
28 ALLOWED_HOSTS = ['*']
29
30
```

- b.
- c. If the user navigates to an unspecified path (localhost:8000/oiuj), they will see the below.



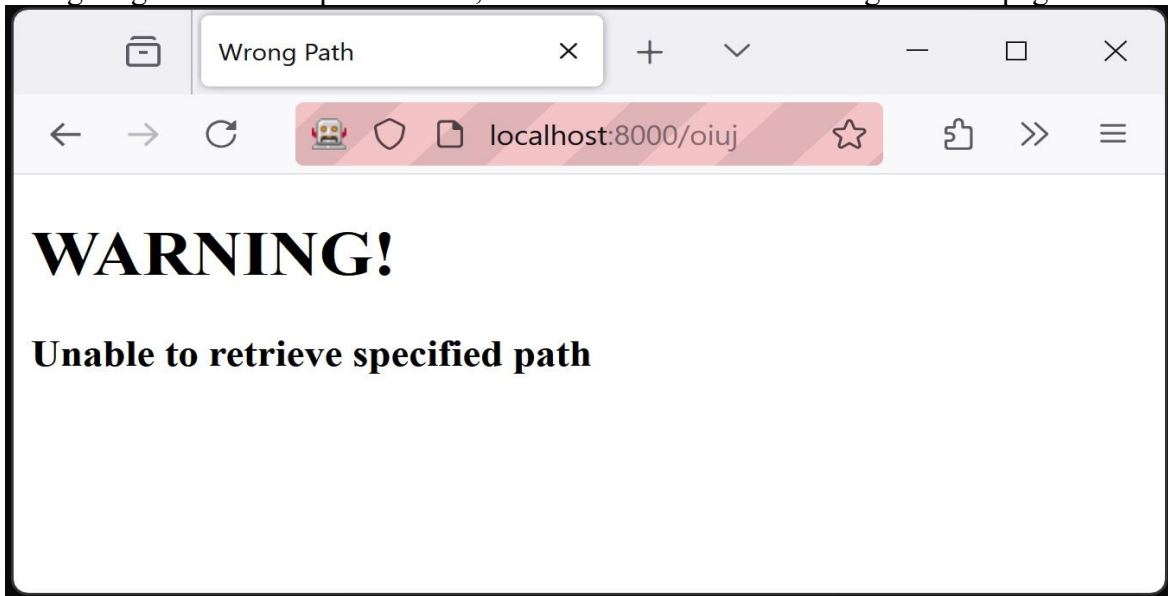
- d.
- e. Django looks for a 404.html template when Django gets a 404 error. The above is displayed when it cannot find a template. Create a 404.html.

```
EXPLORER
OPEN EDITORS
X 404.html drivers\templates
DJANGO_FRAMEWORK_PRACTICE
> .vs
> drivers
> __pycache__
> migrations
> templates
  404.html
  all_drivers.html
  details.html
  main.html
  master.html
  readyToGo.html
  __init__.py
  admin.py
  apps.py

404.html X
drivers > templates > 404.html
1 <!DOCTYPE html>
2
3 <html>
4
5 <title>
6     Wrong Path
7 </title>
8
9 <body>
10
11 <h1>WARNING!</h1>
12
13 <h3>Unable to retrieve spicified path</h3>
14
15 </body>
16
17 </html>
18
```

- f.

- g. Navigating back to the open window, we can see the reflected changes on the page.



- h.
13. Testing Page
- a. Following the same process described above, create a new function in the view called testing that will load a testing template. See below

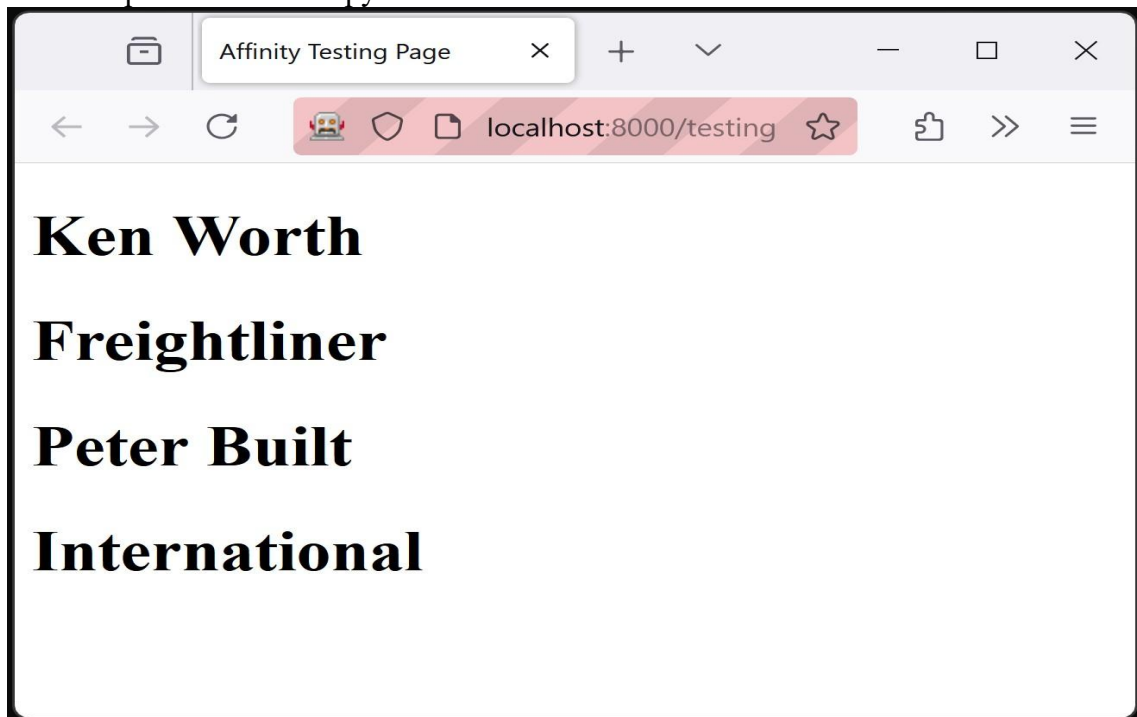
```
views.py x
drivers > views.py > ...
1  #from django.shortcuts import render
2  from django.template import loader
3  from django.http import HttpResponse
4  from .models import Driver
5
6  # Create your views here.
7  def drivers(request):
8      allDrivers = Driver.objects.all().values()
9      template = loader.get_template('all_drivers.html')
10     context = {
11         'myDrivers': allDrivers
12     }
13     return HttpResponse(template.render(context, request))
14     #1return HttpResponse("Maurice Unchained Django!")
15
16     def details(request, id):
17         aDriver = Driver.objects.get(id=id)
18         template = loader.get_template('details.html')
19         context = {
20             'myDriver': aDriver
21         }
22         return HttpResponse(template.render(context, request))
23
24     def main(request):
25         template = loader.get_template('main.html')
26         return HttpResponse(template.render())
27
28     def testing(request):
29         template = loader.get_template('testing.html')
30         context = {
31             'truck': ['Ken Worth', 'Freightliner', 'Peter Built', 'International']
32         }
33         return HttpResponse(template.render(context, request))
34
```

- b.
- c. Now create the testing template in templates folder under the driver app.

The screenshot shows a code editor with two panels. The left panel, titled 'EXPLORER', displays the project structure for 'DJANGO\_FRAMEWORK\_PRACTICE'. It includes folders like '.vs', 'drivers', '\_\_pycache\_\_', 'migrations', and 'templates'. The 'templates' folder is expanded, showing files: '404.html', 'all\_drivers.html', 'details.html', 'main.html', 'master.html', 'readyToGo.html', and 'testing.html'. The right panel shows the content of 'testing.html', which is an HTML template with a title 'Affinity Testing Page' and a body containing a loop for 'truck' items, each displaying an 'x' value.

```
</> testing.html ×
drivers > templates > </> testing.html
1  <!DOCTYPE html>
2
3  <html>
4
5  <title>
6    Affinity Testing Page
7  </title>
8
9  <body>
10    {% for x in truck %}
11      <h1>{{ x }}</h1>
12    {% endfor %}
13  </body>
14
15  </html>
16
```

- d.
- e. We added a view, created the template that the view calls, and now we add the routing to the URL pattern in the url.py file.



- f.
- i. Success!!