URLs and Views

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#python-web

Table of Contents

Creating a Project

Initial Steps

Creating a Project in PyCharm

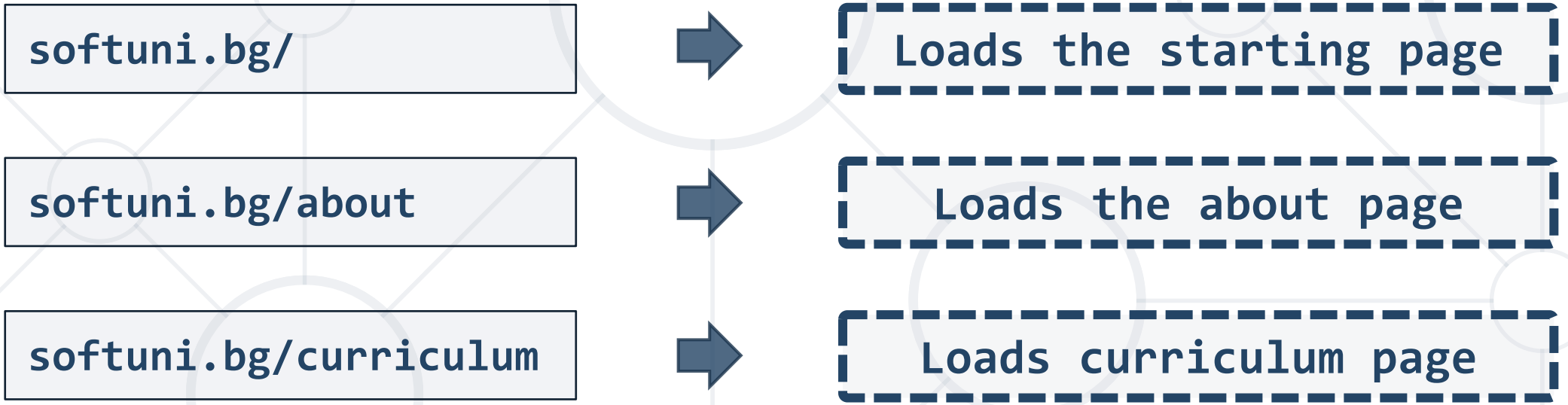
- Start a new project
- Set up a database
- Create a new app called `department`
- Include the app in the project
- Create an `urls.py` file in the app
- Include the `app/urls.py` module in the `project/urls.py`

URLs in Django

Design with No Framework Limitations

URL Configuration

- When a user attempts to access a URL, it **ensures** that a certain **page is reached**



URL Configuration

- How it happens:
  - Django looks for the **`urlpatterns`** variable in the **`urls.py`** file
  - Runs through each URL pattern and stops at the **first matching** one
  - Calls the given **view** and passes an instance of the class **`HttpRequest`**

my-site.com/department

```
urlpatterns = [
    path('department/', views.show_department),
]
```

Dynamic Path Segments (1)

- To **create more pages on a website**, you can add additional paths and views

```
urlpatterns = [
    path('department/1/', views.show_department_with_id_one),
    path('department/2/', views.show_department_with_id_two),
    path('department/3/', views.show_department_with_id_three),
    path('department/4/', views.show_department_with_id_four),
    path('department/5/', views.show_department_with_id_five),
]
```

- In this case, it is better to use **dynamic path segments**

Dynamic Path Segments (2)

- Set one **dynamic URL pattern** for all departments

```
path('department/<department_name>/', views.show_department_by_name)
```

- Optionally, it can include a **converter type** (otherwise, it is converted to a string)

```
path('department/<int:department_id>/', views.show_department_by_id)
```

- The value is passed as an **argument** to the **view**

```
def show_department_by_id(request, department_id):
    ...
```

Default Path Converters

- `str` – matches any **non-empty string**, excluding "/"
- `int` – matches **zero** or any **positive** integer
- `slug` – matches any slug string consisting of ASCII **letters**, **numbers**, **hyphens**, and **underscores**
- `path` - matches any **non-empty string**, **including "/"**
  - Allows you to **match a complete URL path**
- `uuid` – matches a formatted UUID

RegEx in URLs

- Using `re_path()` instead of `path()`

```
re_path(r'^archive/(?P<archive_year>202[0-3])/$', views.show_archive)
```

  - Allows you to use **regular expressions** for more **flexible** URL **pattern matching**
  - This can be **useful** when you need **more complex matching** patterns
  - With `re_path()`, each captured **argument** is sent to the view as a **string**

RegEx in URLs

- It's **recommended** to use **named groups** in your regular expressions for **better readability** and **maintainability**
  - Using **unnamed** regex groups can make your code **harder** to understand
- When using a **mix** of **both** styles (`path()` and `re_path()`)
  - any **unnamed groups** from the regular expression pattern are **ignored**
  - only **named groups** are **passed** to the **view** function

Including URLs

- **Include** a **urls.py** module

```
from django.urls import include, path

urlpatterns = [
    ...
    path('department/', include('department.urls')),
]
```

- Django will **remove** the part of the **matched** URL specified in the `include()` function and **pass** the **remaining string** to the **included urls.py** file for further processing

Including a urlpatterns List

- You can **include** a **`urlpatterns` list**

```
urlpatterns = [
    path('profile/', include([
        path('create/', views.create),
        path('edit/', views.edit),
        path('delete/', views.delete),
    ])),
]
```

- It **removes redundancy** from URL configuration modules, especially when a **single pattern prefix** is used **repeatedly**

Function-Based Views

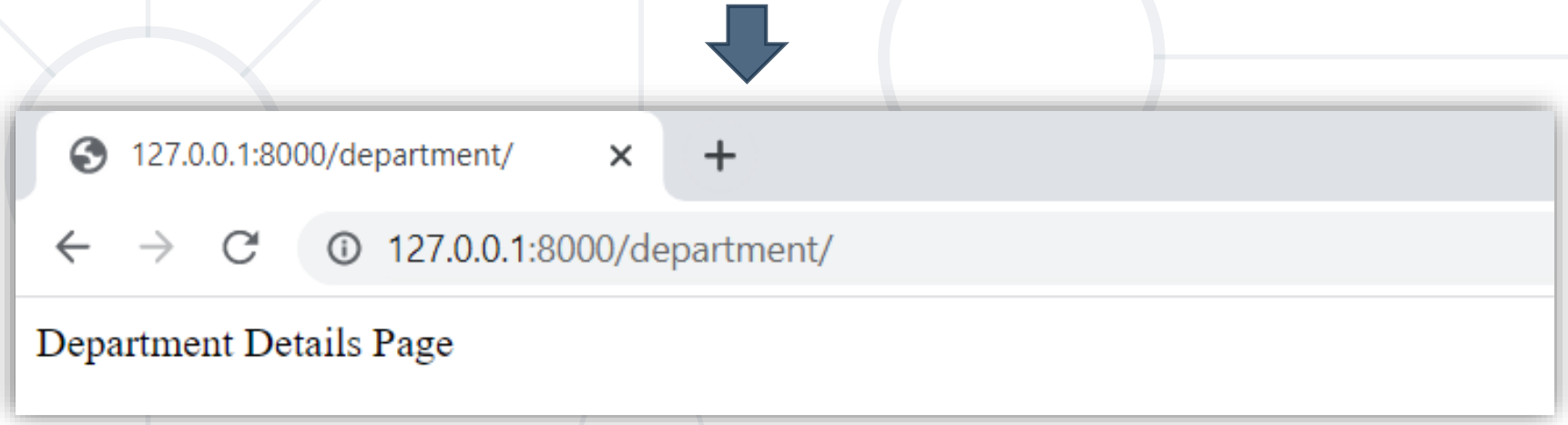
Returning HttpResponse

Views in Django

- **View** in Django contains the **logic** necessary to **produce** the **desired outcome** when a **specific URL** is accessed

```
from django.http import HttpResponse

def show_department(request):
    return HttpResponse("Department Details Page")
```



Views in Django

- Each view receives
  - `HttpRequest` object as its **first** argument (conventionally named `request`)
  - `*args` - matches from **no named groups** in the URL pattern
  - `**kwargs` - matches from **named parts** in the URL pattern
- Each view returns
  - `HttpResponse` object

HttpRequest Object

- Represents an **incoming** **HTTP request** from a **client** (usually a web browser)
- Contains **information** about the **request**
  - including **headers**, **method** (GET, POST, etc.), user **session**, and any **data** submitted in the request
- Typically passed as the **first** **argument** to a **view** function
  - allowing the view to **access** and **process** the **incoming** **requests**

HttpResponse Object

- Represents the **HTTP response** that a view **sends back** to the **client** in **response** to a request
- Contains the **content** that will be **sent** in the **response**
  - along with any relevant **headers** (e.g., **content type**, **status code**)
- Views are **expected** to **return** an `HttpResponse` object
  - which can include **HTML content**, **JSON data**, or any other type of **response** content

Views in Django – Example

Software University

department/views.py

```
from django.http import HttpResponse

def show_department_by_id(request, department_id):
    if department_id == 1:
        department_name = "Developers"
    elif department_id == 2:
        department_name = "Trainers"
        html_output = "<html><body><h1>" \
            "Department Name: %s, Department ID: %s" \
            "</h1></body></html>" \
            % (department_name, department_id)
    return HttpResponse(html_output)
```

Views in Django – Example

Software University

department/urls.py

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

urlpatterns = [
    path('department/<int:department_id>/', views.show_department_by_id),
]
```

Django Shortcut Functions

- Django **shortcut** **functions** are **utility** functions
  - that **simplify** common tasks in Django **development**
- They **streamline** the **process** of working with the Model-View-Template (**MVT**) paradigm by providing **convenient** methods
  - `render()`
  - `redirect()`
  - `get_object_or_404()`
  - `get_list_or_404()`

render()

- Combines a **template** with a **context** dictionary
- Returns an `HttpResponse` object with the **rendered** text
- Simplifies the process of **rendering dynamic content**
  - by **automatically** handling the **template rendering**
  - creating an **HTTP response** with the **rendered content**

render()

- **Required** arguments
  - **request** - The **`HttpRequest`** object used in generating the **response**
  - **template_name** - The full name of the **template** to be used for **rendering**

```
render(
    request=request,
    template_name='department/department_by_id.html',
)
```

render() Context

- **context** – an **optional argument** (empty dictionary by default)
  - A **dictionary** containing **values** that are **added** to the **template context**

```
from django.shortcuts import render

def show_department_by_id(request, department_id):
    ...
    context = {"department_name": "marketing",
               "department_id": department_id}
    return render(
        request=request,
        template_name='department/department-details.html',
        context=context,
    )
```

Context Example In a Template

- The **variable names** in the **context** correspond to the **variables** used in HTML templates

department-details.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Department Info</title>
  </head>
  <body>
    <p>Department Name: {{ department_name }}</p>
    <p>Department ID: {{ department_id }}</p>
  </body>
</html>
```

redirect()

- Is used to **redirect** the **user** to the **appropriate URL**
  - By passing a **hardcoded URL** to redirect to

```
redirect('/some/url/')
```

  - By passing the **name** of a **view** along with **optional** positional or keyword arguments

```
redirect(some_view_name, *args, **kwargs)
```

- It returns an HTTP **status** code of **302**

Redirecting Example

- Directly injecting the URL into the function

department/views.py

```
from django.shortcuts import render, redirect

def show_department_by_name(request, department_name):
    # find the id of the department by its name
    return redirect(
        'http://127.0.0.1:8000/department/' + str(found_department_id)
    )

def show_department_by_id(request, department_id):...
```

- However, this approach is **not dynamic** and could potentially lead to **issues**

Named URLs

- Add a **name** to the **path** in the urls.py module

department/urls.py

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

urlpatterns = [
    path(
        'department/<int:department_id>/',
        views.show_department_by_id,
        name='department-by-id',
    ),
]
```

Dynamic Redirecting Example

Software University

- The **`redirect()`** function **constructs** a **URL** based on the **name** of the **view** and its **parameters**

department/views.py

```
from django.shortcuts import redirect

def show_department_by_name(request, department_name):
    # find the id of the department by its name
    return redirect('department-by-id',
                    department_id=found_department_id)

def show_department_by_id(request, department_id):
    ...
```

reverse()

- URL **reversing** is a process in Django that allows you
  - to **generate** a URL for a **specific view**
  - based on its **name** and any **arguments** it may require

```
from django.urls import reverse

url = reverse('department-by-id', kwargs={'department_id': 1})
```

- Helps to **decouple** **URLs** from **view** functions
- Allows you to **generate URLs** based on the **current state** of your application and the **data** available

URL Reversing Example

- The **`reverse()`** function **generates** a **URL** based on the **name** of the **view** and its **parameters**

department/views.py

```
from django.shortcuts import reverse, redirect

def show_department_by_name(request, department_name):
    # find the id of the department by its name
    url = reverse('department-by-id',
                  kwargs = {'department_id':found_department_id})
    return redirect(url)

def show_department_by_id(request, department_id):
    ...
```

Views Communicating Errors

Communicating Errors

- When a view needs to communicate an **error** status **instead** of **returning** a normal **HttpResponse** object, several options are available
  - Using **HttpResponse Subclasses**
  - Passing an **HTTP Status Code** to the **HttpResponse** Class
  - Raising **Http404** Exception

Using HttpResponse Subclasses

- Django provides **subclasses** of **HttpResponse** like **HttpResponseNotFound**, **HttpResponseServerError**, etc.
  - Can be **returned** directly to **indicate** specific **HTTP error statuses**

```
from django.http import HttpResponse, HttpResponseNotFound

def employees_by_department_id(request, department_id):
    if condition_is_met:
        # Logic for when the condition is met
        ...
        return HttpResponse(html)
    return HttpResponseNotFound('Department was not found')
```

Passing an HTTP Status Code

- A view can **return** an **HttpResponse** object with a **specific status code** using the **status parameter**

```
from django.http import HttpResponse

def show_department_by_id(request, department_id):
    if condition_is_met:
        # Logic for when the condition is met
        ...
        return HttpResponse(html)
    return HttpResponse(status=404)
```

Raising Http404 Exception

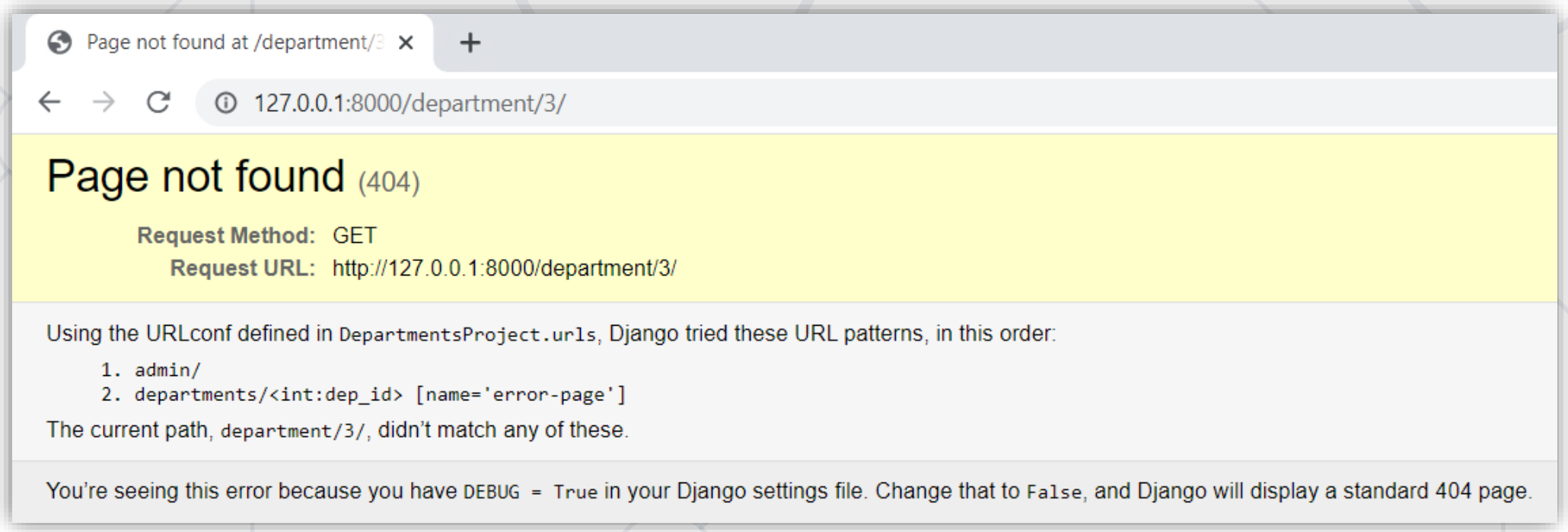
- To indicate a "**Not Found**" **error**, a view can **raise** the **`Http404` exception**
- This will be **caught** by Django's **exception handling** and **result** in a **response** with a **404 status code**

```
from django.http import Http404

def show_department_by_id(request, department_id):
    ...
    else:
        raise Http404
```

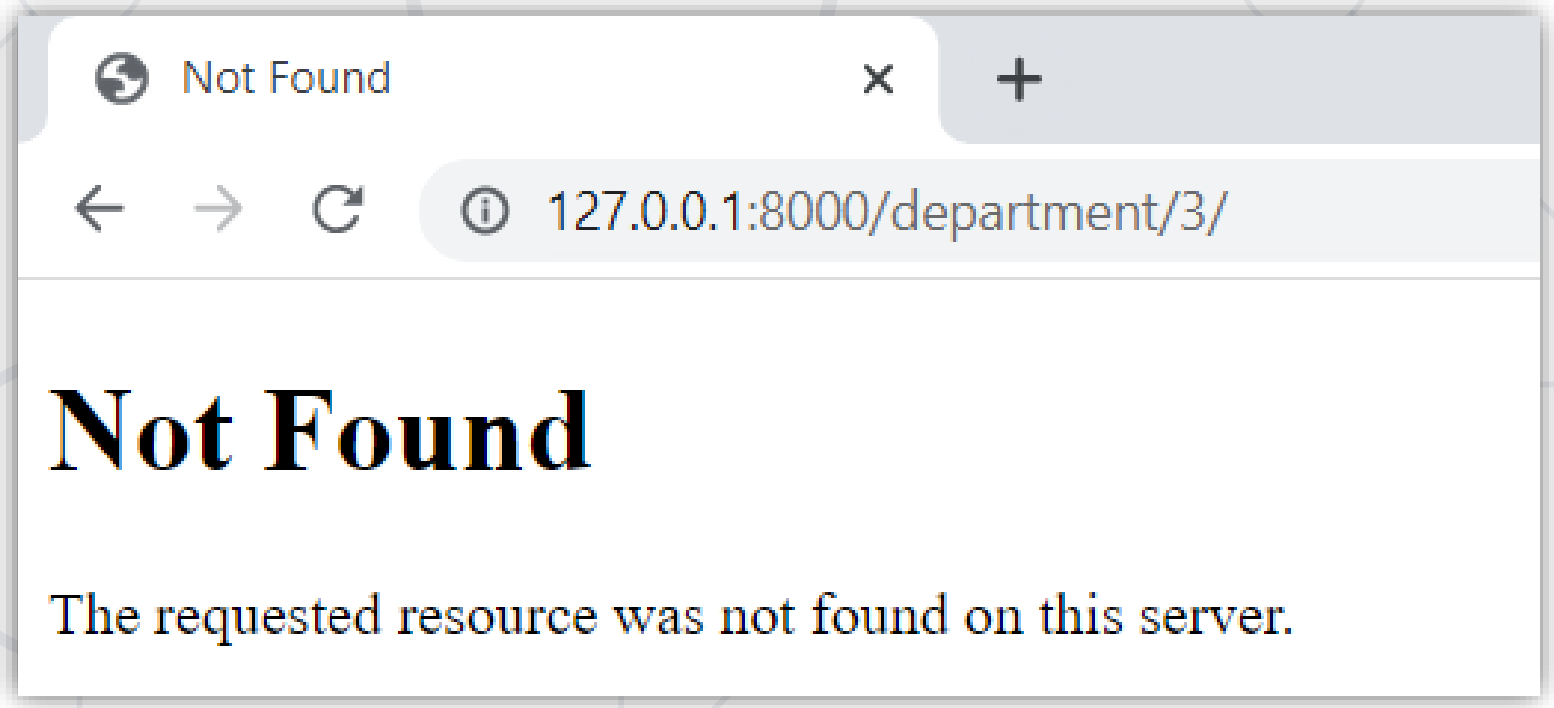
Raising Http404 Exception

- When the **DEBUG** setting is set to **True** in a Django application, any message provided to the **Http404** **exception** will be **displayed** in the **standard 404** **debug template**



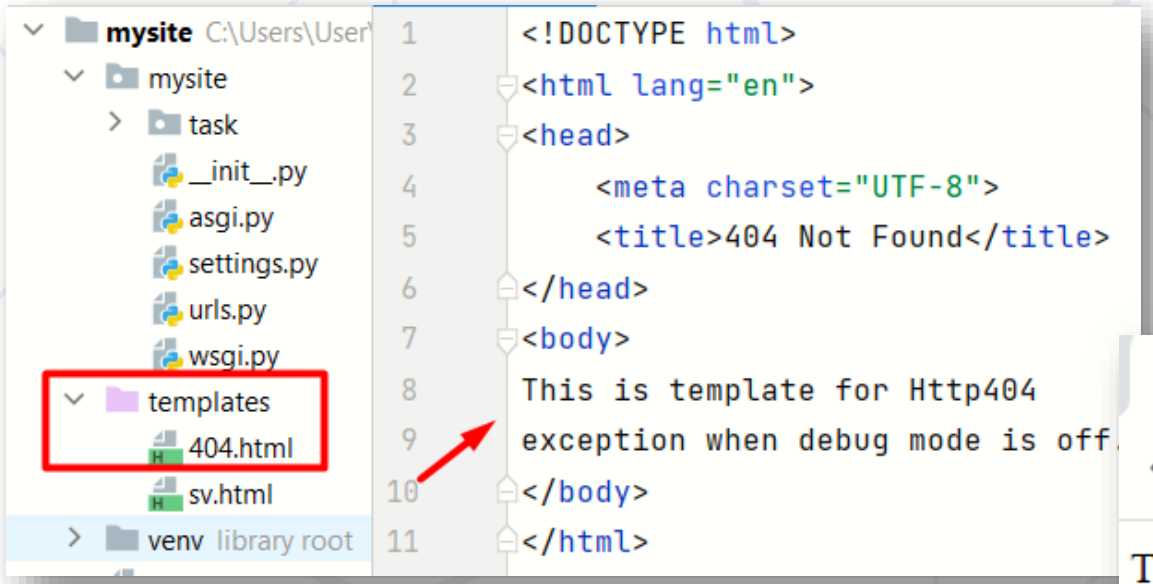
Raising Http404 Exception

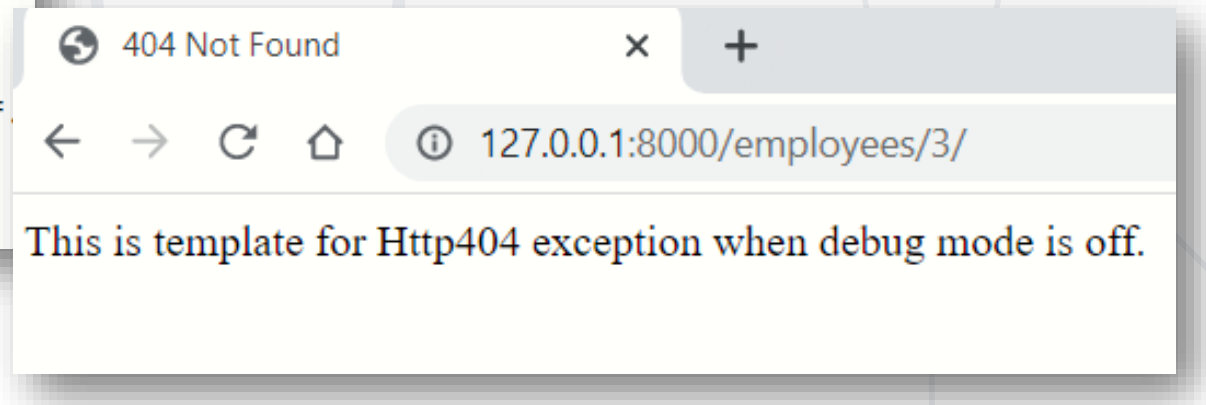
- When the **DEBUG** setting is set to `False` in a Django application, Django will **display** a **default 404 page** for this **exception**



Raising Http404 Exception (4)

- To display a **customized page** for a **404 error**, create a **`404.html` template**
- This **template** will be used when the **`DEBUG`** setting is set to **`False`**

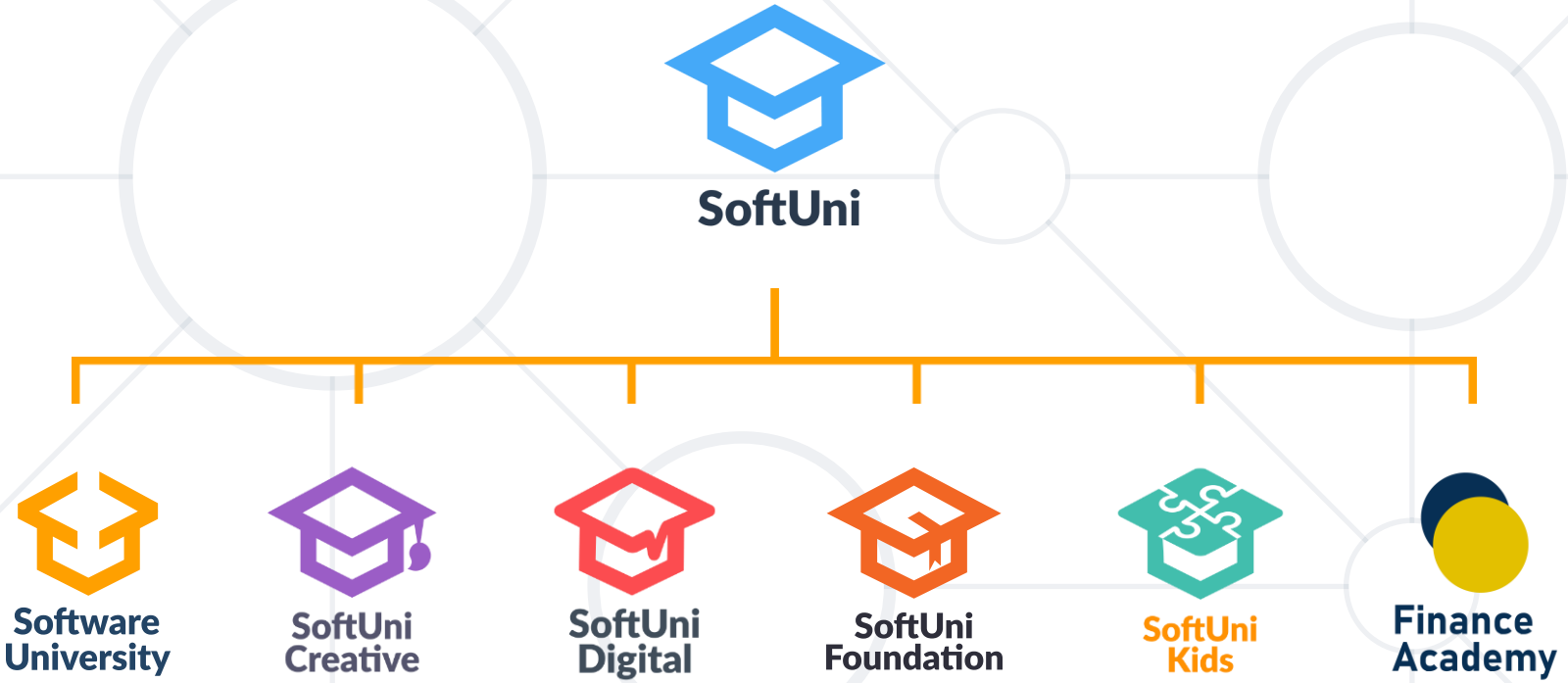




Summary

- The **views.py** file contains the logic for when a specific URL is reached
- The **urls.py** file uses the views.py file to map the URLs patterns
- It is strongly recommended to **avoid hard-coding** URL patterns

Questions?



SoftUni Diamond Partners

Software University

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction, or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg