URLs and Views

SoftUni Team

Technical Trainers

Software University

https://softuni.bg

Table of Contents

Have a Question?

sli.do

#python-web

Creating a Project

Review

Creating a Project in PyCharm

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

URLs in Django

Design with No Framework Limitations

URL Configuration (1)

- When an URL is entered by a user, it **ensures** that a certain **result is achieved**

`softuni.bg/`	→	Loads the starting page
`softuni.bg/about`	→	Loads the about page
`softuni.bg/curriculum`	→	Loads curriculum page

URL Configuration (2)

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

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

- The value name 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

- Use **re_path()** instead of **path()**

```
re_path(r'^archive/(?P<archive_year>[2005-2021])/$', views.show_archive)
```

- Makes the matching **limited**
- Each captured argument is sent to the view as a **string**
- Using **unnamed** RegEx groups **isn't** recommended
  - When both styles are **mixed**, any **unnamed groups are ignored** and only named groups are passed to the view function

Including URL modules

- **At any point**, you can **include urls.py modules**

```
from django.urls import include, path

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

- It **chops off** the part of the matched URL ("department/") and **sends the remaining string** to the **included urls.py file** for further processing

Including URLpatterns List

- Or you can **include URLpatterns lists**

```
urlpatterns = [
    path('<page_name>-<page_id>/', include([
        path('add/', views.add),
        path('edit/', views.edit),
        path('delete/', views.delete),
    ])),
]
```

- It **removes redundancy** from URL conf modules where a single pattern prefix is **used repeatedly**

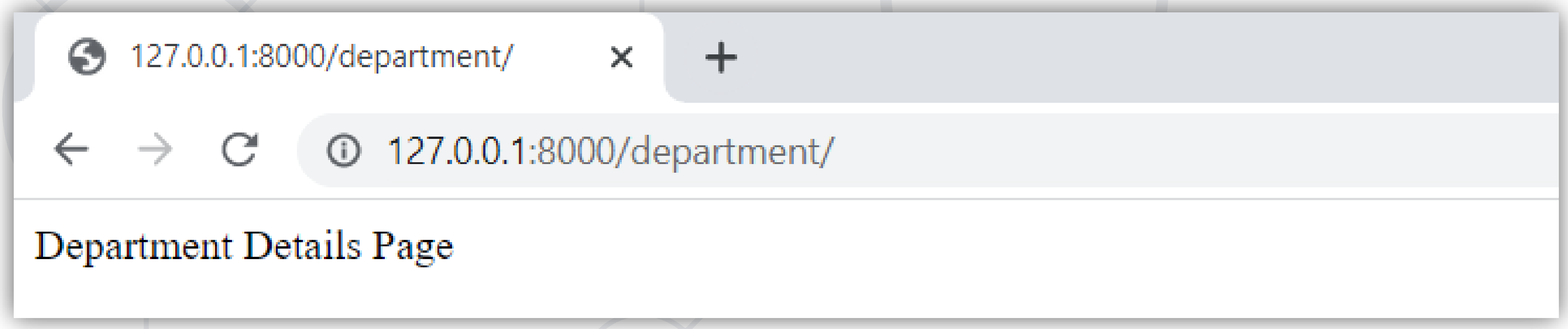
Function-Based Views

Returning HttpResponse

Views in Django (1)

- The view holds the concrete logic to **achieve the expected result** when a certain URL is entered

```
from django.http import HttpResponse

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



127.0.0.1:8000/department/

127.0.0.1:8000/department/

Department Details Page

Views in Django (2)

- Each view receives
  - **HttpRequest** object as its **first** argument (typically 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

Views in Django – Example (1)

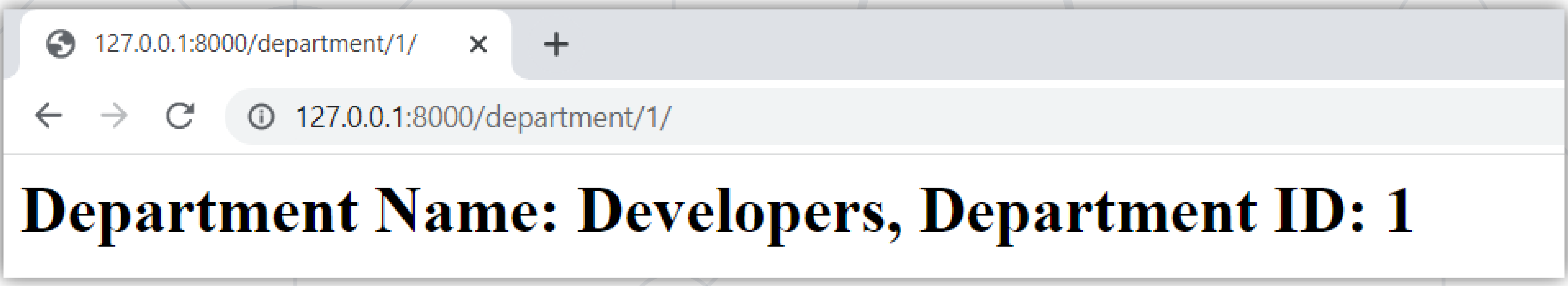
departments/views.py

```
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 = "<html><body><h1>" \
           "Department Name: %s, Department ID: %s" \
           "</h1></body></html>" \
           % (department_name, department_id)
    return HttpResponse(html)
```

Views in Django – Example (2)

departments/urls.py

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

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


127.0.0.1:8000/department/1/

127.0.0.1:8000/department/1/

Department Name: Developers, Department ID: 1

Django Shortcut Functions

- Django shortcut functions are **helper** functions
- They make developing with Django **easier**
- **Connect many different levels** of the Model-View-Template paradigm
  - **`render()`**
  - **`redirect()`**
  - **`get_object_or_404()`**
  - **`get_list_or_404()`**

render() Function

- Combines a **template** with a **context** dictionary
- Returns an HttpResponse object with the **rendered** text
- **Required arguments**
  - **request** - generating this response
  - **template_name** - a full name of a template to use

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

render() Context

- **context - optional argument** (empty dictionary by default)
  - A **dictionary** of values to **add 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 are used by the context **key names**

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() Function

- Use it to **redirect the user** to the **appropriate URL**
  - By passing a **hardcoded URL** to redirect to

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

  - By **passing the name of a view** and optionally some positional or keyword arguments

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

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

Redirecting Example

- Directly inject 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 a department by its name
    return redirect(
        'http://127.0.0.1:8000/department/' + found_department_id
    )

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

- However, it is not dynamic, and it could lead to some issues

Dynamic Redirecting Example (1)

- First, 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 (2)

- 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 a department by its name
    return redirect('department-by-id', found_dep_id)

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

Views Returning Errors

Returning Errors

- Instead of a normal HttpResponse object, a view can return an **HTTP status code**
  - Using **HttpResponse subclasses**
  - Passing an **HTTP status code** to the **HttpResponse** class
  - Raising **Http404** exception

Using HttpResponse Subclasses

- There are list of HttpResponse **subclasses** for several **common HTTP status codes** that can be returned to signify an error

```
from django.http import HttpResponse, HttpResponseNotFound

def employees_by_department_id(request, department_id):
    if ...:
        ...
        return HttpResponse(html)
    else:
        return HttpResponseNotFound('Department was not found')
```

Passing an HTTP Status Code

- If there is no subclass for specific status code, you could **create a return class** for any status code

```
from django.http import HttpResponse

def show_department_by_id(request, department_id):
    if ...:
        ...
        return HttpResponse(html)
    else:
        return HttpResponse(status=501)
```

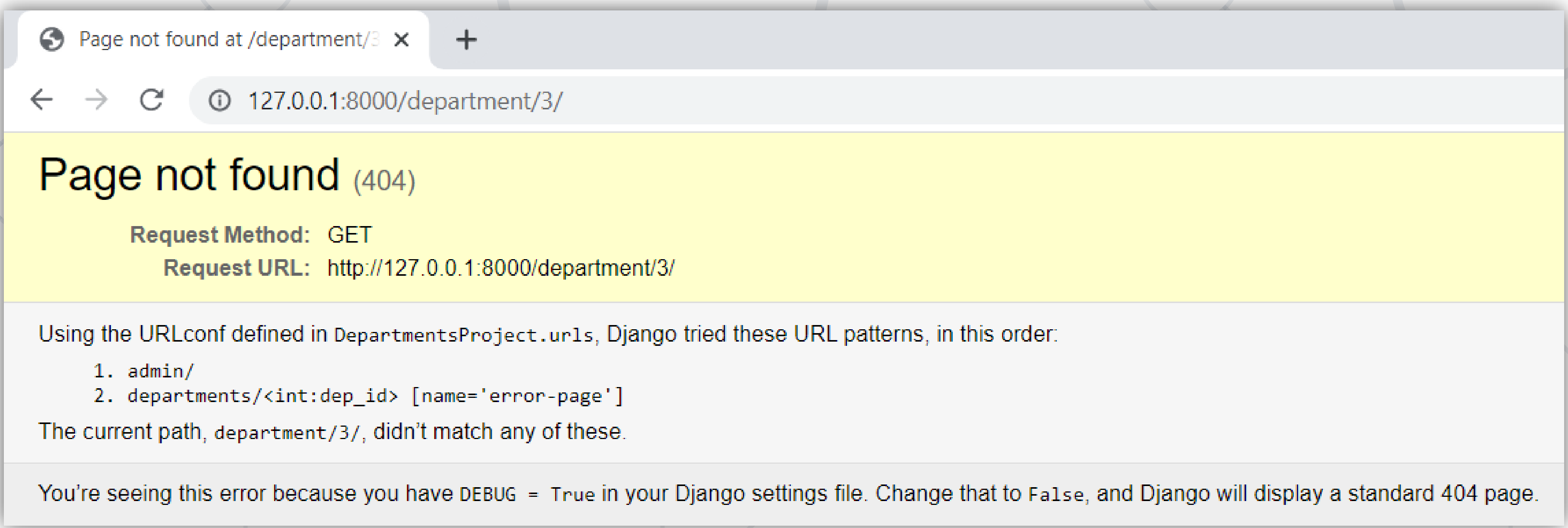
Raising Http404 Exception (1)

- Unlike **HttpResponseNotFound**, it is an **exception**
- It returns an application's **standard error page** and an HTTP **404 status code**

```
from django.http import Http404

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

Raising Http404 Exception (2)

- When **DEBUG** is **`True`**, the provided message to **`Http404`** will appear in the **standard 404 debug template**


Page not found at /department/3

127.0.0.1:8000/department/3/

Page not found (404)

Request Method: GET
Request URL: http://127.0.0.1:8000/department/3/

Using the URLconf defined in `DepartmentsProject.urls`, Django tried these URL patterns, in this order:

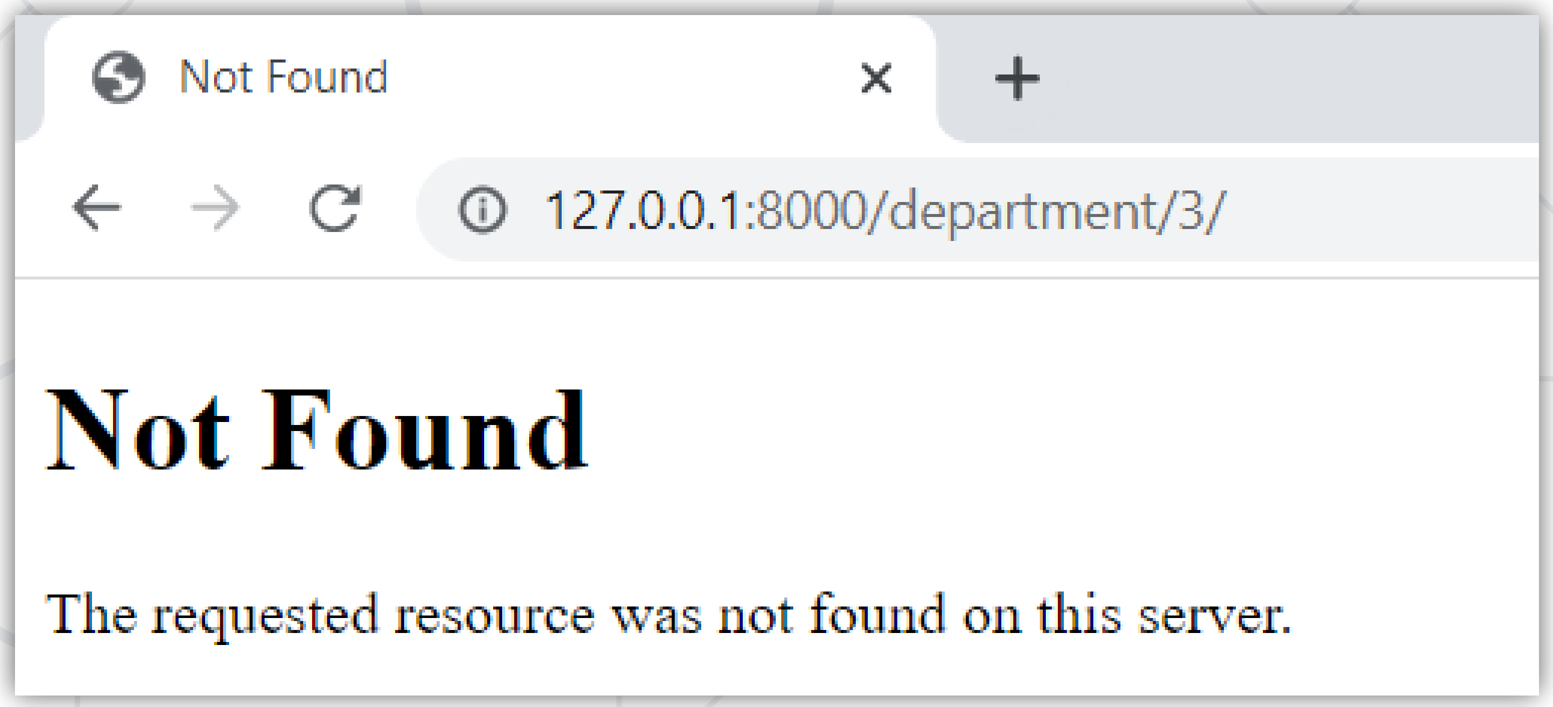
```
1. admin/
2. departments/<int:dep_id> [name='error-page']
```

The current path, `department/3/`, didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

Raising Http404 Exception (3)

- When **DEBUG** is set to **`False`**, Django provides a **default 404 page** for this exception

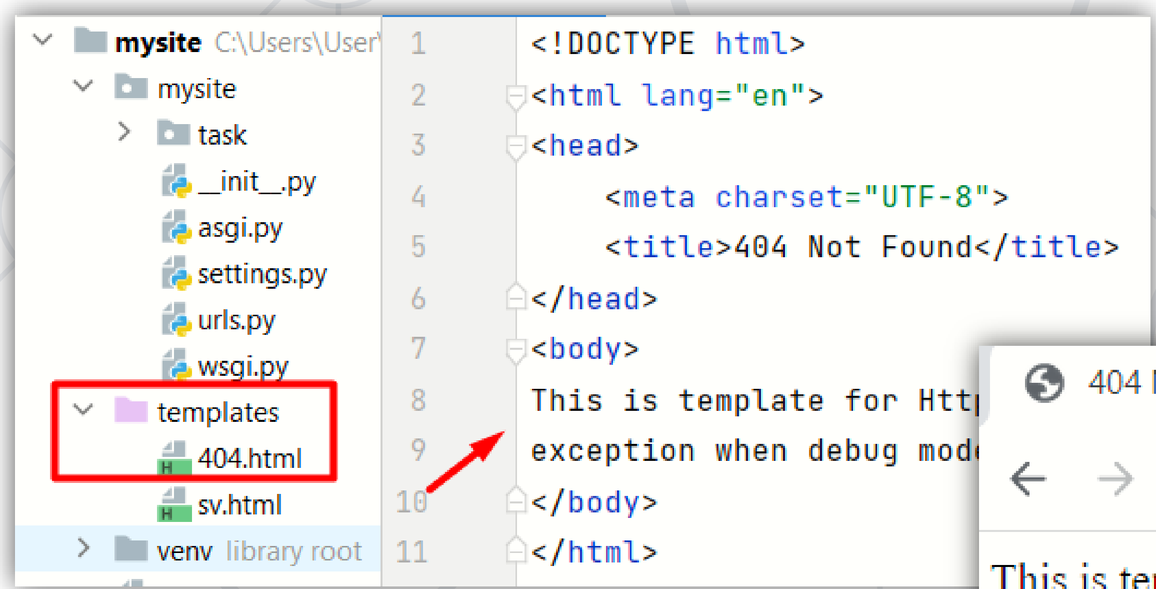


Not Found

The requested resource was not found on this server.

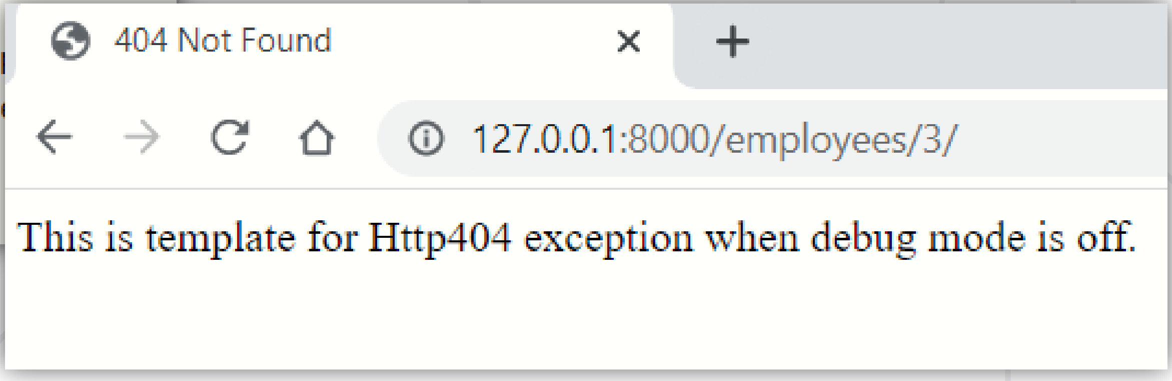
Raising Http404 Exception

- To show a customized page, create a **404.html** template
- This template will be served when **DEBUG** is set to **False**



```
mysite C:\Users\User\
  mysite
    task
    __init__.py
    asgi.py
    settings.py
    urls.py
    wsgi.py
  templates
    404.html
    sv.html
  venv library root
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>404 Not Found</title>
</head>
<body>
This is template for Htt
exception when debug mod
</body>
</html>
```



This is template for Http404 exception when debug mode is off.

Demo

Live Exercise in Class

Summary

- The **views.py** file contains the logic for when a given URL is reached
- The **urls.py** file uses the views.py file to configure the URL's
- It is strongly desirable to **avoid hard-coding** URL patterns

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

Software University

DraftKings

POKERSTARS | CAREERS

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

pre zero

Postbank

Решения за твоето утре

SOFTWARE GROUP

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

SmartIT

INDEAVR

Serving the high achievers

Bosch.IO

PHAR VISION

createX

taulia®

Educational Partners

CODE IT UP

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