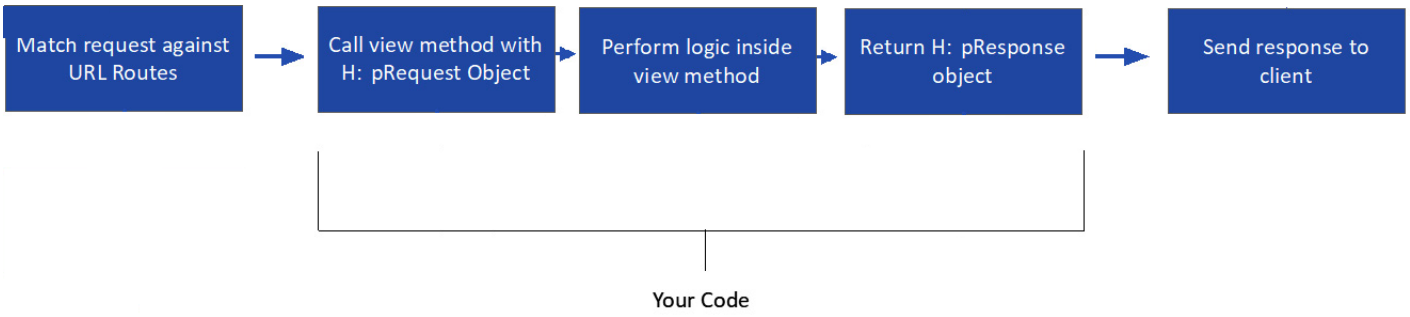
This lab session covers the basics of the Django URL and HTML template. By the end of this session, students should have a very good understanding of how to define URL patterns and create aan HTML template and how both work together to Processing an HTTP request. They should also have already practiced with the Git system and how to use basic commands to push and manage their repository.

The following diagram shows the direction of the transmission of HTTP requests and HTTP responses, between a browser and a web server

Inside the framework, the flow of HTTP request and response is illustrated in the following figure. The sections indicated as **Your Code** are for the code that you write, and the first and last steps are taken care of by Django. Django does the URL matching for you, calls your view code, and then handles passing the response back to the client. Actually, you are going to build and configure a link between a URL and a view where the URL matching is taking place by Django.



Pre-lab Preparation:

1. Complete the previous lab (Lab3): must have the initial Django project with apps (bookmodule & usermodule) and the necessaries configurations.
2. Good understanding of HTTP protocol, mainly the request, response, and HTTP status codes.
3. Basic knowledge about HTTP tags.

Lab Activities: Build and configure a simple link between a URL and a view (simple pattern), along with a simple HTML template.

Task 1: Build your first view function and corresponding URL mapping in the core/urls.py

Step 1: from the root folder, navigate to apps/bookmodule/view.py, then remove `the placeholder text (# Create your views here.)` from views.py and instead insert this content:

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("Hello, world!")
```

Step 2: Open DjangoProjects/urls.py, file (that controls all URL mapping). Import your bookmodule views into the DjangoProjects/urls.py file by adding this line after the other existing imports:

```
...
import apps.bookmodule.views
```

Step 3: Again inside DjangoProjects/urls.py, add a map to the index view to the urlpatterns list by adding a call to the path function[1], with an empty string and a reference to the index function:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', apps.bookmodule.views.index)  #add only this line of code
]
```

Step 4: Run the server to test that everything works as intended. From the ROOT location of your Django project, do the following (don't forget to activate the virtual python environment):

> python manage.py runserver

Important Note: From now on, the development server watches your Django project directory and will restart automatically every time you save a file so that any code changes you make are automatically reloaded into the server. No need to re-run the server again. However, you still have to manually refresh your browser to see changes there (you may press F5 to refresh the browser current webpage).

Step 5: Open a web browser and navigate to http://127.0.0.1:8000. You should see "`Hello, world!`"

[1] If the paths aren't sufficient for defining your URL patterns, you can also use regular expressions. To do so, use re_path() instead of path().

Task 2: Add parameters with HTTP requests[2]

Step 1: From the root folder, navigate to apps/bookmodule/views.py, then, in views.py, modify content as follows:

```
from django.http import HttpResponse
def index(request):
    name = request.GET.get("name") or "world!"  #add this line
    return HttpResponse("Hello, "+name) #replace the word "world!" with the variable name
```

Step 2: Now from the browser navigate to the following urls:

http://127.0.0.1:8000. You should see "Hello, world!"

http://127.0.0.1:8000?name=world!. You should see "Hello, world!"

http://127.0.0.1:8000?name=Mousa. You should see "Hello, Mousa"

Task 3: Build your second view function and corresponding URL mapping with parameters within URL path

Step 1: from the root folder, navigate to apps/bookmodule/view.py, then, in views.py, insert this additional content (index2 view) beside index view:

```
from django.http import HttpResponse
def index(request):
    name = request.GET.get("name") or "world!"
    return HttpResponse("Hello, "+name)

def index2(request, val1 = 0):   #add the view function (index2)
    return HttpResponse("value1 = "+str(val1))
```

Step 2: Again inside DjangoProjects/urls.py, add a map to the index2 view to the urlpatterns list by adding a call to the path function, with an `'index2/<pythontype:parameter1>'`[3] string and a reference to the index2 function. Note that `<pythontype: parameter1>` can be rewritten as `<parameter1>` without specifying the data type of the parameter variable:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', apps.bookmodule.views.index),
    path('index2/<int:val1>/', apps.bookmodule.views.index2)  #add only this line
]
```

---

[2] It is possible to have more than one parameter, like http://127.0.0.1:8000?id=10&name=world!&age=18
Note: this method is GET HTTP request, which is not recommended for security reasons, and use POST HTTP request instead.

[3] For more than one parameter, it will be like
'index2/<pythontype:parameter1>/<pythontype:parameter2>/<pythontype:parameter3>'"

Step 3: Now from the browser navigate to the following urls:

http://127.0.0.1:8000/index2/3/. You should see "value1 = 3"

http://127.0.0.1:8000/index2/10/. You should see "value1 = 10"

http://127.0.0.1:8000/index2/0/. You should see "value1 = 0"

Task 4: Create a simple HTML template

Step 1: Create a templates directory for bookmodule app, by doing the following:

```
> mkdir apps/templates/bookmodule
```

Step 2: Update main/settings.py with template location

```
import os
...
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR = os.path.join(BASE_DIR, "apps" + os.sep + "templates")
TEMPLATES = [{
      ...
      'DIRS': [TEMPLATE_DIR],
      ...
}]
```

Note that many options are available to tell Django how to find templates, which can be set in the TEMPLATES setting. The easiest one is to create a templates directory inside the bookmodule directory. Django will look in this (and in other apps' templates directories) because APP_DIRS is True in the settings.py file. Anyway, this task is intended to utilize the one template inside the apps folder and make it accessible to all apps.

Step 3: Create a file with the name "index.html" within the templates directory '`apps/templates/bookmodule`', and add the following html content:

```
<h1>Hello from a template!</h1>
```

Step 4: Open views.py in the bookmodule directory. We no longer manually create HttpResponse, so you may remove the HttpResponse import line and add this line after the other existing imports:

```
from django.shortcuts import render
```

Step 5: Update the index function, in apps/bookmodule/views.py, so that, instead of returning HttpResponse, it's returning a call to render, passing in request and the template name:

```
def index(request):
    name = request.GET.get("name") or "world!"
    return render(request, "bookmodule/index.html")    #Change HttpResponse  to render function
```

Task 5: Rendering variables in the HTML template that processes a context

Step 1: open the file "index.html", and update the file so that it contains a place to render the name variable:

```
<h1>Hello from {{ name }}!</h1>
```

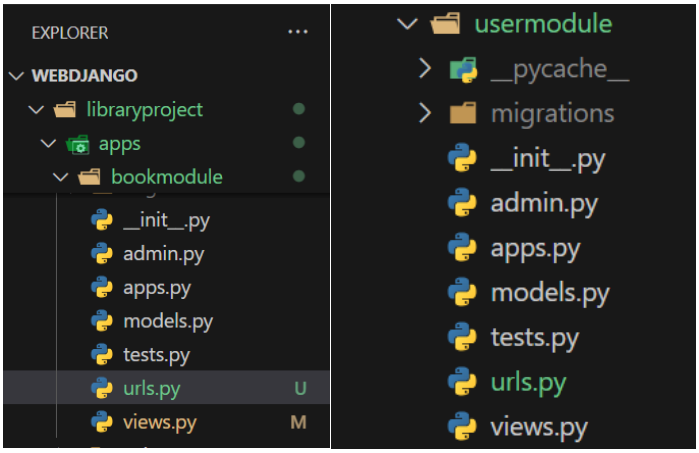
Step 2: Update the index function, in apps/bookmodule/views.py, and add the context dictionary as the third argument to the render function. Change your render line to this:

```
def index(request):
    name = request.GET.get("name") or "world!"
    return render(request, "bookmodule/index.html" , {"name": name})  #your render line
```

Task 6: Define patterns globally (DjangoProjects/urls.py) with specific urls.py file for each app/module

Note that it is common to keep one URL configuration per application in your Django project. Here, we will create a separate URL configuration for each app and add it to our project-level URL configuration.

Step 1: Create a new urls.py file for each app module(bookmodule and usermodule) in the same folder as the views.py file.



Step 2: move urls of bookmodule to apps/bookmodule/urls.py, and include newly created urls.py in core/urls.py using include module.

DjangoProjects/urls.py

```
from django.contrib import admin
from django.urls import include, path
urlpatterns = [
    path('admin/', admin.site.urls),
    path('books/', include("apps.bookmodule.urls")), #include urls.py of bookmodule app
    path('users/', include("apps.usermodule.urls"))  #include urls.py of usermodule app
]
```

apps/bookmodule/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index),
    path('index2/<int:val1>/', views.index2)
]
```

apps/usermodule/urls.py

```
from django.urls import path
urlpatterns = [
]
```

Libraryproject/urls.py

```
from django.contrib import admin
from django.urls import path, include
import apps.bookmodule.views


urlpatterns = [
    path('admin/', admin.site.urls),
    path('books/', include("apps.bookmodule.urls")),
    path('users/', include("apps.usermodule.urls"))

]
```

libraryproject > apps > bookmodule > urls.py

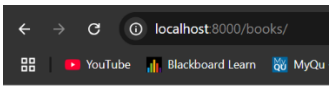
```
1   from django.urls import path
2   from . import views
3
4   urlpatterns = [
5       path('', views.index),
6       path('index2/<int:val1>/', views.index2),
7       path('index2/<str:val1>/', views.index2),
8       path('<int:bookId>', views.viewbook),
9   ]
10
```

libraryproject > apps > usermodule > urls.py

```
1   from django.urls import path
2   urlpatterns = [
3   ]
4
```

Step 3: Now from the browser navigate to the following urls:

http://127.0.0.1:8000/books/. You should see "Hello, world!"



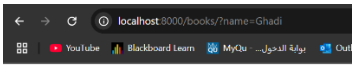
Hello from a template!

Hello, world!

http://127.0.0.1:8000/books/?name=world!. You should see "Hello, world!"

http://127.0.0.1:8000/books/?name=Dr. Faisal. You should see "Hello, Dr. Faisal"

http://127.0.0.1:8000/books/?name=Mousa. You should see "Hello, Mousa"



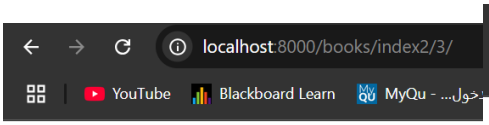
Hello from a template!

Hello, Ghadi

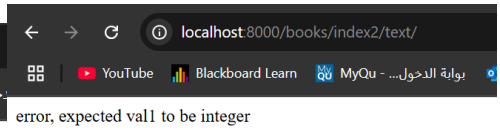
http://127.0.0.1:8000/books/index2/3/.. You should see "value1 = 3"

http://127.0.0.1:8000/books/index2/10/. You should see "value1 = 10"

http://127.0.0.1:8000/books/index2/text/. You should see "error, expected val1 to be integer"



value1 = 3



error, expected val1 to be integer

Task7: Create a URL, view, and HTML to display one book details

Step 1: Create a new function at the location: 'root/apps/bookmodule/views.py' that accept an ID argument:

```
def viewbook(request, bookId):
    # assume that we have the following books somewhere (e.g. database)
    book1 = {'id':123, 'title':'Continuous Delivery', 'author':'J. Humble and D. Farley'}
    book2 = {'id':456, 'title':'Secrets of Reverse Engineering', 'author':'E. Eilam'}
    targetBook = None
    if book1['id'] == bookId: targetBook = book1
    if book2['id'] == bookId: targetBook = book2
    context = {'book':targetBook} # book is the variable name accessible by the template
    return render(request, 'bookmodule/show.html', context)
```

Step 2: In 'root/apps/bookmodule/urls.py', add the following path function to urlpatterns:

```
path('<int:bookId>', views.viewbook)
```

```
libraryproject > apps > bookmodule > urls.py
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('', views.index),
6      path('index2/<int:val1>/', views.index2),
7      path('index2/<str:val1>/', views.index2),
8      path('<int:bookId>', views.viewbook),
9  ]
10
```

Step 3: Create the template at the location: 'root/apps/templates/bookmodule/show.html' and put some code in it.

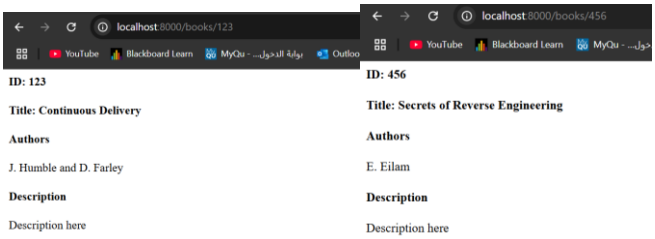
```
<h4>ID: {{ book.id }}</h4>
<h4>Title: {{ book.title }}</h4>
<h4>Authors</h4>
<p>{{ book.author }}</p>
<h4>Description</h4>
<p>Description here</p>
```

```
libraryproject > apps > templates > bookmodule > dj show.html
1  <h4>ID: {{ book.id }}</h4>
2  <h4>Title: {{ book.title }}</h4>
3  <h4>Authors</h4>
4  <p>{{ book.author }}</p>
5  <h4>Description</h4>
6  <p>Description here</p>
7
```

Step 4: Now from the browser navigate to the following urls:

http://127.0.0.1:8000/books/123

http://127.0.0.1:8000/books/456



Additional content: To redirect to another view or URL, use the function redirect, and for details and examples visit the following link:
https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/#redirect