May 04

Middleware:

D:\Django_20MAR_7PM>django-admin startproject middlewareproject1
D:\Django_20MAR_7PM>cd middlewareproject1
D:\Django_20MAR_7PM\middlewareproject1>py manage.py startapp testapp
-->Add app in settings.py

- **views.py**

```
from django.http import HttpResponse
def welcome_view(request):
   return HttpResponse('<h1>Custome Middleware Demo</h1>')
```

- **urls.py**

```
path('hello/', views.welcome_view)
```

Inside testapp folder

- **middleware.py**

```
class ExecutionFlowMiddleware(object):
   def __init__(self,get_response):
      print('init method execution.....')
      self.get_response = get_response
   def __call__(self,request):
      print('Pre processing of request')
      response = self.get_response(request)
      print('Post processing of request')
      return response
```

- **settings.py**

```
MIDDLEWARE = [
          -----
   'testapp.middleware.ExecutionFlowMiddleware'
]
```

Middleware application to show information saying app is under maintenace

-->create project
-->Create testapp
-->Add app in settings.py

- **views.py**

```
from django.http import HttpResponse
def home_page_view(request):
   return HttpResponse('<h1>Hello this response is from view function response</h1>')
```

- **urls.py**

```
path('hello/', views.home_page_view)
```

- **middleware.py**

```
from django.http import HttpResponse
class AppMaintenanceMiddleware(object):
   def __init__(self,get_response):
      self.get_response = get_response
   def __call__(self,request):
      return HttpResponse('<h1>Currently application under maintenance...Please try after 2-days....</h1>')
```

- **settings.py**

```
MIDDLEWARE = [
   -------
   'testapp.middleware.AppMaintenanceMiddleware'
]
```

Middleware application to show meaningful response if view function raises any error.

In the middleware we can define process_exception() method, which will be executed if view function raises any error.

process_exception(self,request,exception)

- **views.py**

```
def home_page_view(request):
   print(10/0)
   return HttpResponse('<h1>This is from view function</h1>')
```

- **urls.py**

```
path('hello/', views.home_page_view)
```

middleware.py

```
from django.http import HttpResponse
class ErrorMessageMiddleware(object):
   def __init__(self,get_response):
      self.get_response = get_response

   def __call__(self,request):
      response = self.get_response(request)
      return response

   def process_exception(self,request,exception):
      # return HttpResponse('<h1>Currently we are facing some technical
problem...pls try after some time....</h1>')
      return HttpResponse(f'<h1>Currenty we are facing some technical
problems<br>The Raised Exception:{exception.__class__.__name__}<br> The
Exception Message:{exception}</h1>')
```

- **settings.py**

```
MIDDLEWARE = [
    'testapp.middleware.ErrorMessageMiddleware'
]
```

<u>Configuration of Multiple middleware classes:</u>

We can configure any number of middlewares and all these middlewares will be executed according to order declared inside settings.py

- **views.py**

```
def home_page_view(request):
   print('This line printed by view function')
   return HttpResponse('<h1>This is from view function</h1>')
```

- **urls.py**

```
path('hello/', views.home_page_view)
```

- **middleware.py**

```
from django.http import HttpResponse
class FirstMiddleware(object):
```

```
    def __init__(self,get_response):
        self.get_response = get_response
    def __call__(self,request):
        print('This line printed by Middleware-1 before processing request')
        response = self.get_response(request)
        print('This line printed by Middleware-1 after processing request')
        return response

class SecondMiddleware(object):
    def __init__(self,get_response):
        self.get_response = get_response
    def __call__(self,request):
        print('This line printed by Middleware-2 before processing request')
        response = self.get_response(request)
        print('This line printed by Middleware-2 after processing request')
        return response
```

- **settings.py**

```
MIDDLEWARE = [
    'testapp.middleware.SecondMiddleware',
    'testapp.middleware.FirstMiddleware',
]
```

- **test.py**

```
import time
class Test:
        def __init__(self):
                print('Constructor Execution.....')
        def __del__(self):
                print('Destructor Execution.....')
l = [Test(),Test(),Test()]
time.sleep(5)
print('End of application')
```