1. **What settings need to be modified to enable middleware applications?**

Ans To enable a middleware in our app, we will write its name under ‘Middleware’ section in our ‘settings.py’

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

2. **Where in a middleware application would you put code to modify the arguments that will be passed to the view function**

In function based approach:

Before the ‘response = get_response(request)’ we will put our code so that it will be passed to the view function. And we will place our middleware just above the view core.

In class based approach:

In the process_view() function

3. **Where in a middleware application would you put the code to track the size of the responses**

Ans

In function-based approach:

After the ‘response = get_response(request)’ line we can put our code if we want to track the size of our response.

In class based approach:

In pocess_response() function.

4. **Which middleware application enables you to access the user object in the view function**

Ans In Django.contrib.auth.middleware.AuthenticationMiddleware application

5. **Add a new middleware application to the django.middleware file that appends footers with some kind of copyright or information string to every response.**

Ans **middleware.py**

```
from .views import fun




class Footer:

  def __init__(self, get_response):

     self.get_response = get_response

     # One-time configuration and initialization.
  def __call__(self, request):
      response = self.get_response(request)
      return response

  def process_template_response(self, request, response):
      response = fun(request)
      response.context_data['hello'] = "Now in MiddleWare Changed Content"
      return response
```

Urls.py

```
"""middleware URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
  https://docs.djangoproject.com/en/2.1/topics/http/urls/
Examples:
Function views
  1. Add an import:  from my_app import views
  2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
  1. Add an import:  from other_app.views import Home
  2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
```

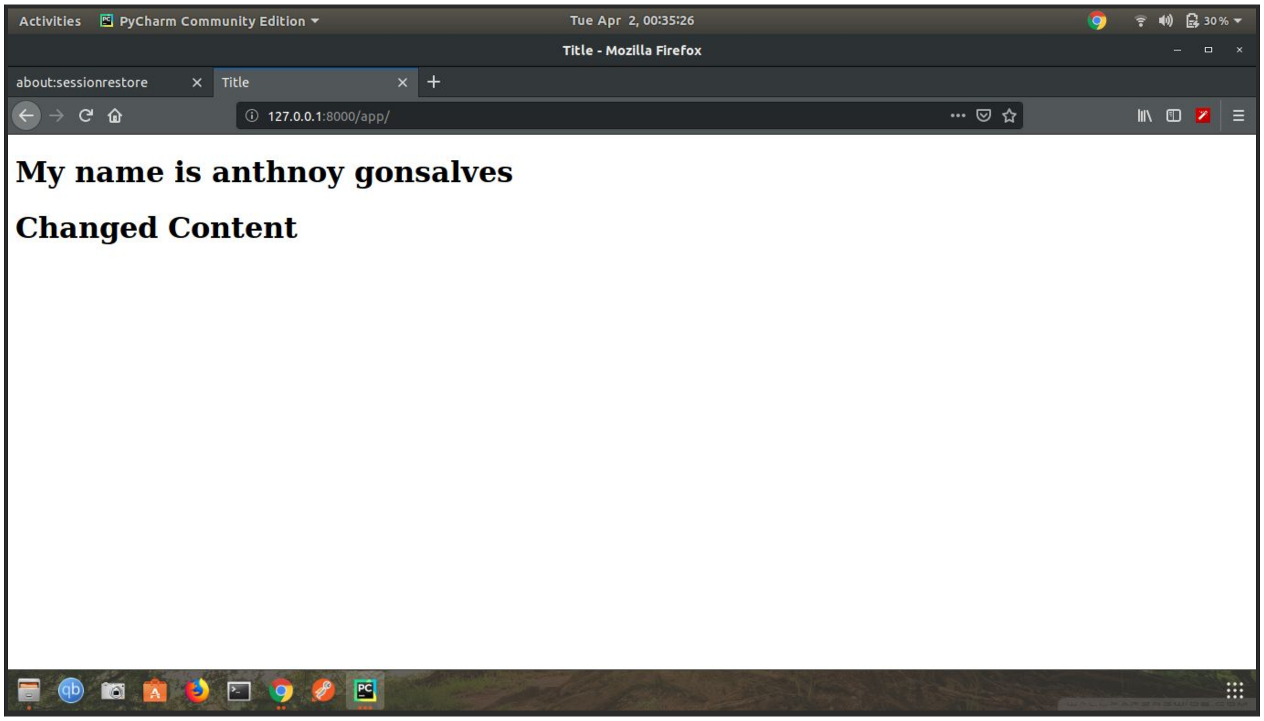
```
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from new_app import views


urlpatterns = [
    path('admin/', admin.site.urls),
    path('app/', views.fun, name='app')
]
```

Views.py

```
from django.template.response import TemplateResponse


# Create your views here.



def fun(request):
    context_data = {'hello': "hello"}
    return TemplateResponse(request, 'file.html', context_data)
```

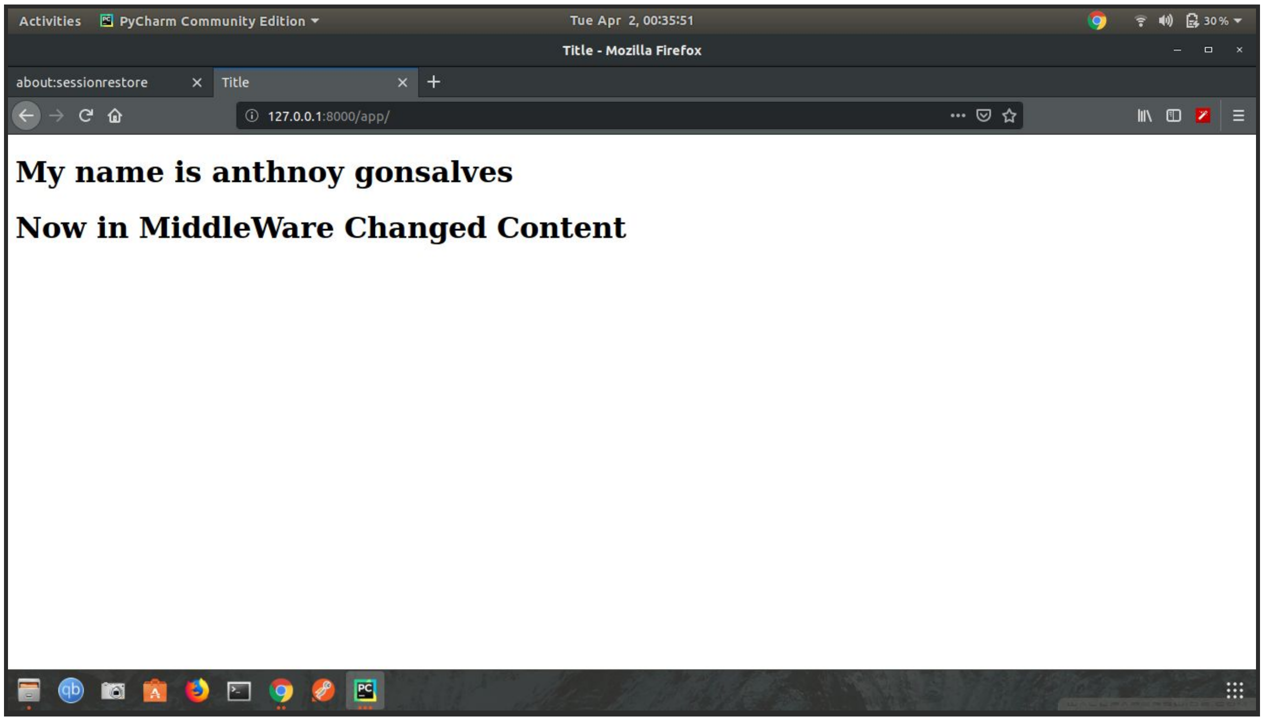
Settings.py

```
MIDDLEWARE = [

    'new_app.middlew.Footer',

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',



]
```



My name is anthnoy gonsalves

Changed Content



```
My name is anthnoy gonsalves
Now in MiddleWare Changed Content
```