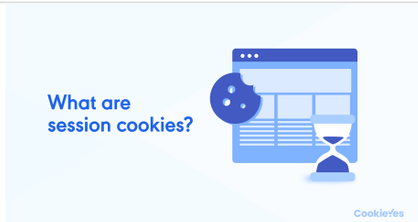
User Management - continue - login

1. cookie & session

What are session cookies? Do they need consent? - CookieYes

Session cookies are cookies that last only for a single user session. Learn more about why they are used, if they need consent and whether they are GDPR compliant.

https://www.cookieyes.com/blog/session-cookies/



npm: cookie-session

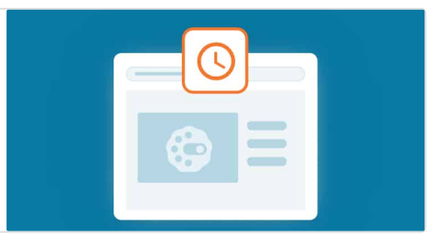
cookie session middleware. Latest version: 2.0.0, last published: a year ago. Start using cookie-session in your project by running `npm i cookie-session`. There are 8262 other projects in the npm registry using

https://www.npmjs.com/package/cookie-session

Everything You Need to Know About Session Cookies - Securiti

A session cookie is a simple snippet of code that a website installs on its visitor's device for temporary use. It helps track real-time changes in a user's activity while on a website, such as adding items to the

https://securiti.ai/blog/session-cookies/



Session Cookie

This definition explains the meaning of Session Cookie and why it matters.

https://www.techopedia.com/definition/4910/session-cookie

2. login page

```
class LoginForm(forms.Form):
    username = forms.CharField(label='Username',
                               widget=forms.TextInput(
                                   attrs={"class": "form-control"}
                               ))
    password = forms.CharField(label='Password',
                               widget=forms.PasswordInput(
                                   attrs={"class": "form-control"}
                               ))
    remember = forms.BooleanField(label='RememberMe',
                                  widget=forms.CheckboxInput(
                                      attrs={"class": "form-check-input"}
                                  ))
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            if name != 'remember':
                if field.widget.attrs:
                    field.widget.attrs['class'] = 'form-control'
                    field.widget.attrs['placeholder'] = field.label
                else:
                    field.widget.attrs = {"class": "form-control",
                                          "placeholder": field.label}
```

```
def login(request):
    if request.method == 'GET':
        form = LoginForm()
        return render(request, 'login.html', {'form': form})

    form = LoginForm(data=request.POST)
    if form.is_valid():
        pass
    return render(request, 'login.html', {'form': form})
```

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="{% static "plugins/bootstrap-3.4.1-dist/css/bootstrap.css" %}">
    <style>
        .w_title {
            color: darkgray;
            font-family: "Eras Light ITC";
            font-weight: bold;
        }
        .container{
            width: 400px;
            padding: 30px;
            margin-top: 10px;
            border: 1px solid gray;
        }
    </style>
</head>
<body>
    <div class="container" style="box-shadow: 5px 5px lightgray; margin-top: 100px">
        <h1 class="w_title" style="text-align: center;">User Login</h1>
        <form method="post" novalidate>
            {% csrf_token %}
            <div class="mb-3">
                <label class="form-label">Email address</label>
                {{ form.username }}
                <div class="form-text" style="color: red">{{ form.username.errors.0 }}</div>
            </div>
            <div class="mb-3">
                <label class="form-label">Password</label>
                {{ form.password }}
                <div class="form-text" style="color: red">{{ form.password.errors.0 }}</div>
            </div>
            <div class="mb-3 form-check">
                {{ form.remember }}
                <label class="form-check-label" for="exampleCheck1">Remember me</label>
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>

</body>
</html>
```

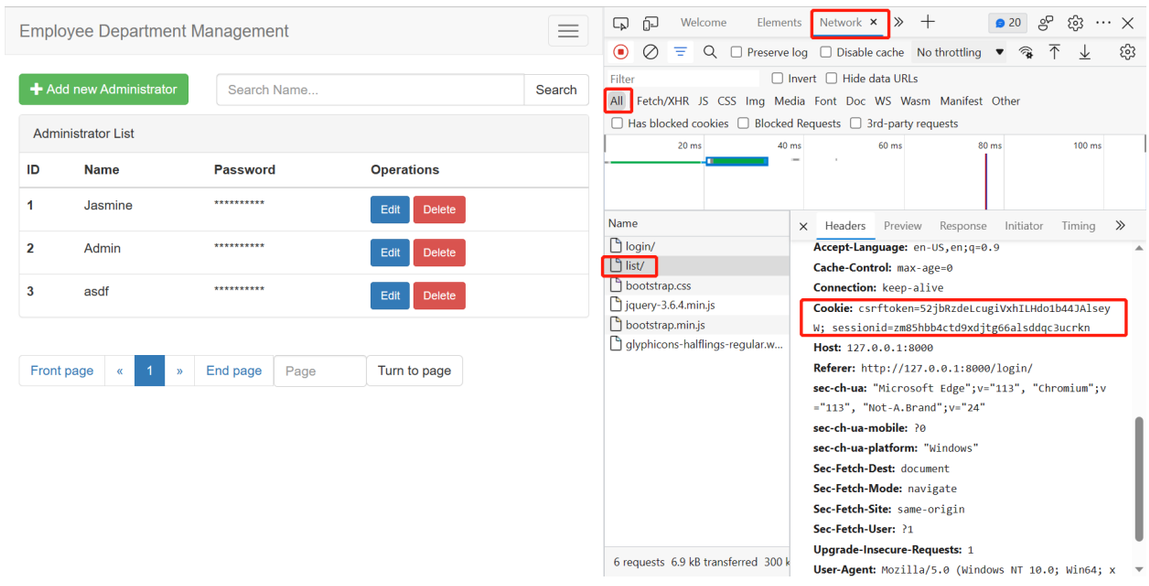
3. connect to database

check username & password

```
class LoginForm(forms.Form):
    username = forms.CharField(label='Username',
                               widget=forms.TextInput,
```

```
                                required=True)
    password = forms.CharField(label='Password',
                                widget=forms.PasswordInput,
                                required=True)

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

        for name, field in self.fields.items():
            if field.widget.attrs:
                field.widget.attrs['class'] = 'form-control'
                field.widget.attrs['placeholder'] = field.label
            else:
                field.widget.attrs = {"class": "form-control",
                                      "placeholder": field.label}

    def clean_password(self):
        pwd = self.cleaned_data.get('password')
        return md5(pwd)


def login(request):
    if request.method == 'GET':
        form = LoginForm()
        return render(request, 'login.html', {'form': form})
    print(2)
    form = LoginForm(data=request.POST)

    if form.is_valid():
        # form.cleaned_data (type:dict)
        # find record in dict where username, password = username, password
        admin_object = models.Admin.objects.filter(**form.cleaned_data).first()

        if not admin_object:
            form.add_error("password", "Username or Password incorrect")
            return render(request, 'login.html', {'form': form})

        # cookie & session
        request.session['info'] = {'id': admin_object.id, 'name': admin_object.username}
        return redirect("/admin/list/")
    return render(request, 'login.html', {'form': form})
```

cookie



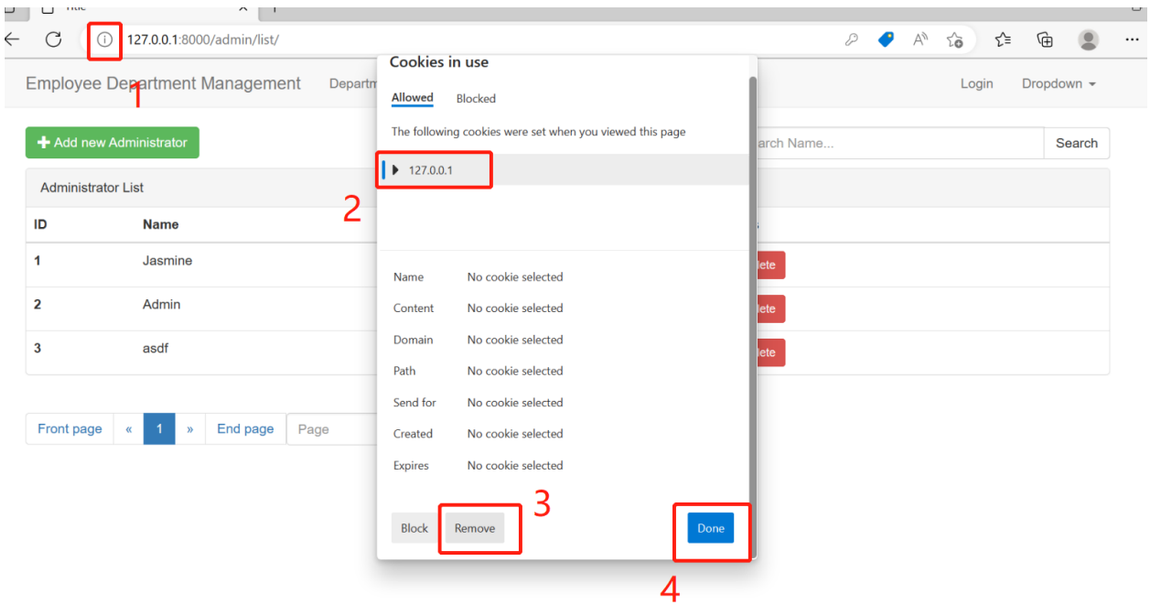
sessionid=zm85hbb4ctd9xdjtg66alsddqc3ucrkn

we will store this id in the database:

```
mysql> show tables;
+--------------------------------+
| Tables_in_django_ex_user_manage |
+--------------------------------+
| app01_admin                    |
| app01_department               |
| app01_userinfo                 |
| auth_group                     |
| auth_group_permissions         |
| auth_permission                |
| auth_user                      |
| auth_user_groups               |
| auth_user_user_permissions     |
| django_admin_log               |
| django_content_type            |
| django_migrations              |
| django_session                 |
+--------------------------------+
13 rows in set (0.02 sec)

mysql> SELECT * FROM django_session;
+----------------------------------+--------------------------------------------------------------------------------------------------------------+----------------------------+
| session_key                      | session_data                                                                                                 | expire_date                |
+----------------------------------+--------------------------------------------------------------------------------------------------------------+----------------------------+
| zm85hbb4ctd9xdjtg66alsddqc3ucrkn | eyJpbmZvIjp7ImlkIjozLCJuYW1lIjoiYXNkZiJ9fQ:1pyMHA:OUPaIVMl4IHFajiNhJjIFaRPp5PmaPswfMkvjSkBNKs | 2023-05-29 00:41:20.891120 |
+----------------------------------+--------------------------------------------------------------------------------------------------------------+----------------------------+
1 row in set (0.00 sec)
```

4. disable access if haven't login

remove cookies



add cookie check in each list session

```
    # check if login
    # if already login, allowed access, else, reject and redirect to login page
    # get cookie if exist
    info = request.session.get("info")
    if not info:
        return redirect('/login/')
```

this time when click on xxxlist, will redirect to login page.

also, need to add it to each add/edit/delete

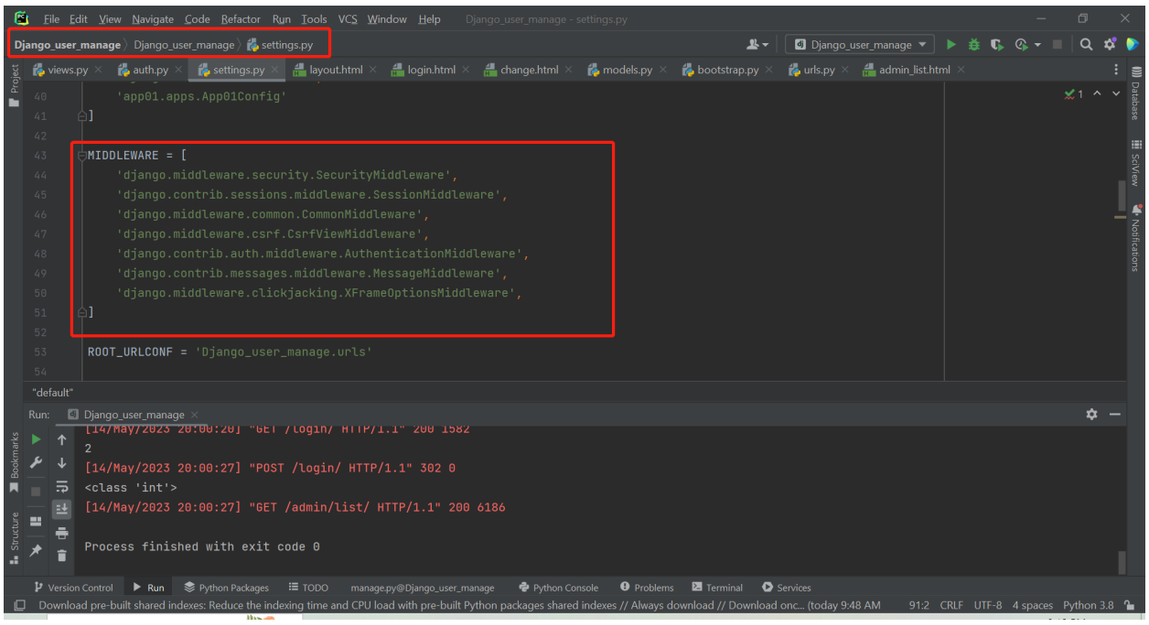
middleware

add middle ware

```
from django.utils.deprecation import MiddlewareMixin
from django.shortcuts import redirect
```

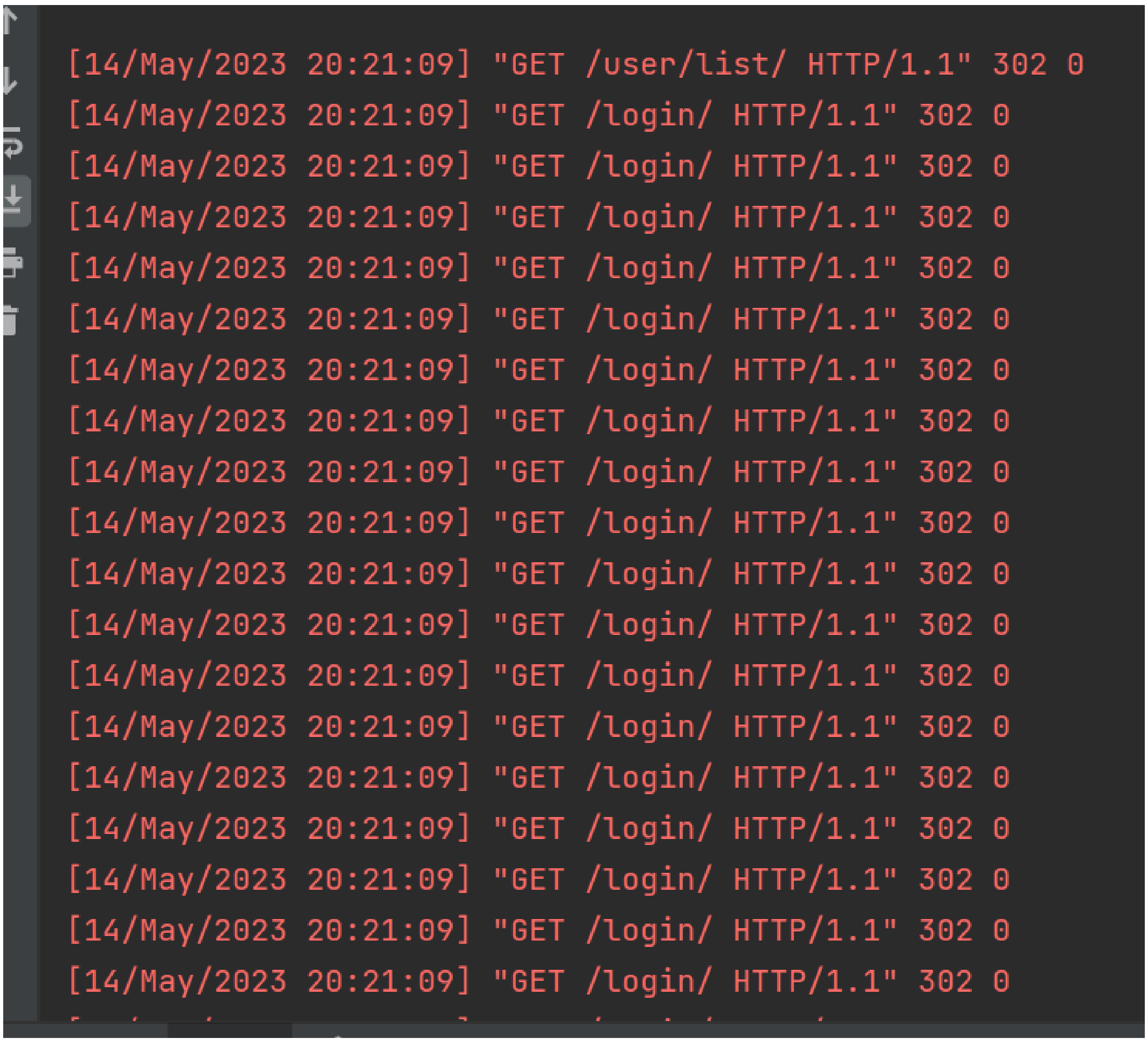
```
class AuthfromMiddleware(MiddlewareMixin):
    def process_request(self, request):
        # if have login
        info_dict = request.session.get("info")
        if info_dict:
            return
        # haven't login
        return redirect('/login/')
```

register middle ware



add here

but when running, we keep getting



```
[14/May/2023 20:21:09] "GET /user/list/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
[14/May/2023 20:21:09] "GET /login/ HTTP/1.1" 302 0
```

because, every time we want to get to login page, we have to login. (also add this middle ware to login page)

Thus we need to exclude login page

```
if request.path_info == '/login/':
            return
```

```
class AuthfromMiddleware(MiddlewareMixin):
    def process_request(self, request):
        if request.path_info == '/login/':
            return
        # if have login
        info_dict = request.session.get("info")
        if info_dict:
            return
        # haven't login
        return redirect('/login/')
```