USER SPECIFIC PAGES

Up until this point everyone has been able to access every page of our website and users share a common list of to do lists. This is obviously not ideal and in many instances you want each user to have their own set of information that is specific to them. In this tutorial I will be showing how we can save information specific to each user and how to restrict users to only see certain pages.

django

UPDATING MODELS

What we will start by doing is updating our models so that each To Do List is assigned to a specific user. This way each user can only access their To Do Lists.

We will modify the models.py from within our main app to be the following:

# models.py file

```python
from django.db import models

from django.contrib.auth.models import User

class ToDoList(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="todolist", null=True)

    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name


class Item(models.Model):
    todolist = models.ForeignKey(ToDoList, on_delete=models.CASCADE)
    text = models.CharField(max_length=300)
    complete = models.BooleanField()

    def __str__(self):
        return self.text
```

Add code

Because we've made changes to our Database we need to execute the following commands:

```
python manage.py makemigrations
python manage.py migrate
```

MODIFYING VIEWS.PY

Since we've added a foreign key to ToDoList this means whenever we create one we need to add it to a user. So we will modify the create function inside views.py to do this.

# views.py file

```
def create(response):
    if response.method == "POST":
        form = CreateNewList(response.POST)

        if form.is_valid():
            n = form.cleaned_data["name"]
            t = ToDoList(name=n)
            t.save()

            response.user.todolist.add(t)  # adds the to do list to the current logged in user

        return HttpResponseRedirect("/%i" % t.id)

    else:
        form = CreateNewList()

    return render(response, "main/create.html", {"form": form})
```

Add code

VIEWING ALL OF A USERS LISTS

Now we are going to add a new page that will allow us to view all of the current users To Do Lists. We will create another template called view.html inside of templates > main.

# view.html file

```
{% extends "main/base.html" %}

{% block title %}
View
{% endblock %}

{% block content %}
    {% for td in user.todolist.all %}
    <p><a href="/{{td.id}}">{{td.name}}</a></p>
    {% endfor %}
{% endblock %}
```

Now we will add a url to link to this page:

# urls.html file

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

urlpatterns = [
    path('<int:id>', views.index, name='index'),
    path("", views.home, name="home"),
    path("home/", views.home, name="home"),
    path("create/", views.create, name="index"),
    path("view/", views.view, name="view"),  # <-- added
]
```

VIEWING ALL OF A USERS LISTS

And create a function to render the view:

# views.py file

```
def view(response):
    return render(response, "main/view.html", {})
```

And now when we visit /view we can see a list of all of our To Do Lists. Try logging out (/logout) and logging back in (/login) with a different user and seeing how the view page changes.

RESTRICTING PAGES

Now time to restrict access to users that are not logged in. For this example we will only restrict access to viewing a To Do List that is not yours but you can do this anywhere.

If a user tries to access a To Do List that they did not create we will simply redirect them to the home page. We will need to modify our index function to accomplish this:

# views.py file

```python
def index(response, id):
    ls = ToDoList.objects.get(id=id)

    if ls in response.user.todolist.all():

        if response.method == "POST":
            if response.POST.get("save"):
                for item in ls.item_set.all():
                    if response.POST.get("c" + str(item.id)) == "clicked":
                        item.complete = True
                    else:
                        item.complete = False

                    item.save()

            elif response.POST.get("newItem"):
                txt = response.POST.get("new")

                if len(txt) > 2:
                    ls.item_set.create(text=txt, complete=False)
                else:
                    print("invalid")

        return render(response, "main/list.html", {"ls": ls})

    return render(response, "main/home.html", {})
```