DJANGO FORMS

A form is essentially a way of passing information from your front-end(HTML) to your back-end(in this case python). There are two different methods a form can use, POST and GET. In this tutorial we will discuss how to use django built in forms to add new To Do Lists to our application.

CREATING A NEW PAGE

We are going to need to add a new page to the site that will allow us to create a new ToDoList. We will need to do the following:

- Create a new template
- Create a new view function inside of views.py
- Add a new link/URL

CREATING A NEW PAGE

I am going to call my template create.html

# create.html file

```
{% extends 'main/base.html' %}
{% block title %}
Create New List
{% endblock %}

{% block content %}
    <h3>Create a New To Do List</h3>
    <br>
    <form method="post" action="/create/" class="form-group">
        {% csrf_token %}
        {{form.as_p}}
        <div class="input-group mb-3">
            <div class="input-group-prepend">
            <button name="save" type="submit" class="btn btn-success">Create</button>
            </div>
        </div>
    </form>
{% endblock %}
```

Whenever we create a form from inside an HTML file we need to add the following tag: {% csrf_token %}. This is for security reasons.
The {{form}} tag is what will represent the form that we create from within our python code. We will pass it to the view under the name form and it will generate a form for us.

CREATING A NEW PAGE

we need to create a new python file called forms.py. This file will be in root directory of our app (same place as views.py). It should contain the code shown below

# forms.py file

```
from django import forms


class CreateNewList(forms.Form):
    name = forms.CharField(label="Name", max_length=200)
    check = forms.BooleanField()
```

more information about forms and field data

CREATING A NEW PAGE

The next step is to add a new function inside views.py that will be able to render this html for us.

# views.py file

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

We will add more to this later.

Now we can add a new URL to this page from urls.py.

# urls.py file

```python
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")
]
```

Add code

In this case our form will simply have one field, the name of the form.

RENDERING OUR FORM

Now we can finish the function we created earlier in our views.py file and pass the form we just created to our HTML.
To do this we need to import our form, create an instance of it and pass it in the context of render.

# views.py file

```python
from django.shortcuts import render
from django.http import HttpResponse
from .models import ToDoList, Item

from .forms import CreateNewList


def index(response, id):
    ls = ToDoList.objects.get(id=id)
    return render(response, "main/list.html", {"ls":ls})


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


def create(request):
    form = CreateNewList()
    return render(request, "main/create.html", {"form": form})
```

Add code

Modify code

In this case our form will simply have one field, the name of the form.

GETTING INFORMATION

We have successfully created our form but we have not yet added the functionality to get information from it. To learn how to do this we must first understand the difference between GET and POST.

Whenever you access a web page in django you pass it one of two different requests:

- Post
- Get

Post**:** A post request is used whenever information needs to passed securely from the front end to the back-end. It is also used whenever any changes to system/database are going to be made. The post request will deliver information to your backed in an encrypted and secure way.

Get**:** A get request is used typically when you want to get information from the backed or you don't care about the security of information. Any information passed with a get request will be visible through the URL of the web-browser. An example of a GET request is when a user types in a search phrase and clicks enter, the information is passed through the URL to the backend so it can render the search results. This is fine because we do not care about the security of the search phrase the user types in.

For our form we will use a POST request because we will be modifying the state of the system/database after our form is submitted.

We will now update our newly created function in views.py to handle a POST request:

# views.py file

```python
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()

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

    else:
        form = CreateNewList()

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

We start by checking if we are receiving a POST request which would mean that the form has been submitted. If this is the case we will create a new form and fill it with the data we received from the request. Then we will check if the form is valid, if it is we will create and save a new to do list object. Lastly we will redirect to the to do list we just created.