**Forms**

Django provides a range of tools and libraries to help you build forms to accept input from site visitors, and then process and respond to the input.

**Forms are two types**

1. HTML forms
2. Django forms

**Advantages of Django Forms**

So, building forms with Django's form class definitely has advantages.

One major advantage that Django's form class gives is validation. This is the biggest advantage of Django's form class. It was built with validation being the key.

Django's form class provides validation on many forms. For example, if you create a form and there are many blank fields (by default all fields are required meaning a user must enter in a value), the form will be submitted; the user will be told to fill in the blank fields before the form is submitted.

Django's form class also provides validation with many different types of data entry. For example, it has an EmailField that validates email address. So, if a user types in, 'Peter' as his email, Django's form states that this is not a valid email address.

In many ways, it gives validation on all forms and all fronts.

You can validate email address, integers, decimals, so on and so forth.

So validation is the corner of Django's form class. It is an excellent validating tool to make sure that all fields of the form are filled and appropriate for what is being asked.

Another advantage of Django forms is gives ease of use when putting the data taken from a form to a database table. You can use the same variable that makes the form fields and insert the values from them into a database table. It provides ease of use when taking data from a form and inserting the data into a database. If you are doing this, Django's form are very suitable and makes this process simpler.

Forms are developed using one predefined package **django.forms**

Django forms are two types

1. Generic form
2. Model form

Generic form, is build individual by place input fields/widgets. Generic forms are also custom forms

Model form, it is build using model. This form connected with model. This form is build using fields of model.

**Basic steps for creating django forms**

1. In application folder create forms.py
2. Build form class by inheriting Form class
3. Use this inside view
4. Display this form inside template

**Forms.py**

from django import forms

class NameForm(forms.Form):

    your\_name=forms.CharField(max\_length=100,label="Your Name")

**views.py**

from django.shortcuts import render

from formsapp.forms import NameForm

# Create your views here.

def get\_name(request):

    if request.method=="POST":

        form=NameForm(request.POST)

    else:

        form=NameForm()

    return render(request,"name.html",context={"form":form})

**name.html**

<form action="/home/" method="post">

    {% csrf\_token %}

    {{ form }}

    <input type="submit" value="Submit">

</form>

**Urls.py**

from django.contrib import admin

from django.urls import path

from formsapp import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('home/',views.get\_name),

]

**Settings.py**

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'formsapp',

]

import os.path

TEMPLATE\_PATH=os.path.join(BASE\_DIR,"templates")

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [TEMPLATE\_PATH],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },