**DJANGO FORM**

Collect the data from the end user.

Similar to html forms, but we will write Python code and then html forms will be generated.

Several form field validations are inbuilt in Django. Also create custom validations using Python.

Form data can be easily converted into your objects, model, data structures like list and set.

Forms directly represent the model and that data will be directly saved in the database.

**CSRF : Cross Site Request Forgery.**

Django has inbuilt support to prevent this forgery. When an html form is pushed to the client, Django will automatically include a CSRF underscore token into the form.

This will be a long UUID field, with alpha numeric value. It will be added to the form as hidden field.

Use : {% csrf\_token %}

In the settings.py file, under MIDDLEWARE, the highlighted is responsible for CSRF token generation and validation.

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',  
]

**USING {{form.as\_p}} (Paragraph)**

A screenshot of a computer

Description automatically generated

**USING {{form.as\_table}} (Table)**

A computer screen shot of a registration form

Description automatically generated

**HTTP POST METHOD TO DISPLAY ON THE CONSOLE**

A screenshot of a computer

Description automatically generated

**USING widget attribute**

A screenshot of a computer

Description automatically generated

**USING required in the widget attribute (Default is True)**

A screenshot of a computer

Description automatically generated

**USING clean for a field input, which is greater or less in value.**

A screenshot of a computer

Description automatically generated

**USING Single clean method for multiple inputs.**

def clean(self):  
 user\_cleaned\_data= super().clean()  
 fname = user\_cleaned\_data['firstName']  
 if len(fname)>15:  
 raise forms.ValidationError("Length of First Name should not be greater than 15 !!!")  
  
 eml = user\_cleaned\_data['email']  
 if eml.find('@') == -1:  
 raise forms.ValidationError("Enter Proper valid email id !!!")

"""  
 def clean\_firstName(self):  
 fname = self.cleaned\_data['firstName']  
 if(len(fname)<3):  
 raise forms.ValidationError("Length of First Name should be greater than 3 !!!")  
 return fname  
  
 def clean\_email(self):  
 eml = self.cleaned\_data['email']  
 if eml.find('@')== -1:  
 raise forms.ValidationError("Enter Proper valid email id !!!")  
 return eml  
"""

**Forms.py validator code.**

from django import forms  
from django.core import validators  
import re  
class UserRegistrationForm(forms.Form):  
 GENDER=[('male','Male'),('female','Female')]  
 firstName = forms.CharField(widget=forms.TextInput,  
 validators=[validators.MinLengthValidator(5),  
 validators.MaxLengthValidator(15)])  
 lastName = forms.CharField(widget=forms.TextInput,  
 validators=[validators.MinLengthValidator(5),  
 validators.MaxLengthValidator(15)])  
 age = forms.IntegerField(  
 validators=[validators.MinValueValidator(16),  
 validators.MaxValueValidator(65)])  
 gender = forms.CharField(widget=forms.Select(choices=GENDER))  
 email = forms.CharField(widget=forms.EmailInput)  
 password = forms.CharField(widget=forms.PasswordInput,  
 validators=[validators.MinLengthValidator(8),  
 validators.MaxLengthValidator(12)])  
  
  
def clean\_password(self):  
 pwd = self.cleaned\_data['password']  
 reg = "^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*#?&])[A-Za-z\d@$!#%\*?&]{6,20}$"  
  
 pat = re.compile(reg)  
 mat = re.search(pat,pwd)  
  
 if mat:  
 raise forms.ValidationError("Enter valid Password with Alpha Numeric and Special Characters !!!")  
 return pwd

A screenshot of a computer

Description automatically generated

**Final Output**

Form is Valid

First Name : Kamal Kumar

Last Name : JesuRanjan

Age : 48

Gender : male

Email : kamalrkumar@gmail.com

Password: Sampletest

**A screenshot of a registration form

Description automatically generated**

**Assignment code for password validations.**

from django import forms

from django.core import validators

import re

class UserLoginForm(forms.Form):

userName = forms.CharField(max\_length = 200)

password = forms.CharField(widget = forms.PasswordInput(),validators=[validators.MinLengthValidator(8)])

def clean\_password(self):

inputpassword = self.cleaned\_data.get('password')

if not re.search("[A-Z]+", inputpassword):

raise forms.ValidationError('This password must contain at least 1 uppercase character')

return inputpassword

views.py

from django.shortcuts import render

from . import forms

# Create your views here.

def userLoginView(request):

form = forms.UserLoginForm()

if request.method == 'POST':

form = forms.UserLoginForm(request.POST)

if form.is\_valid():

print("Form is valid")

print("User Name",form.cleaned\_data['userName'])

return render(request,'UserLogin.html',{'form':form})

UserLogin.html

<!DOCTYPE html>

<html lang="en">

<title>User Login Form</title>

<body>

<h1>User Login</h1>

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

{% csrf\_token %}

{{form.as\_table}}

<input type="submit" value="Login" />

</form>

</body>

</html>

QUIZ

A screenshot of a form

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated