Django Model Form

It is a class which is used to create an HTML form by using the Model. It is an efficient way to create a form without writing HTML code.

Django automatically does it for us to reduce the application development time. For example, suppose we have a model containing various fields, we don't need to repeat the fields in the form file.

For this reason, Django provides a helper class which allows us to create a Form class from a Django model.

Let's see an example.

Nested Structure in C

Keep Watching

Django ModelForm Example

First, create a model that contains fields name and other metadata. It can be used to create a table in database and dynamic HTML form.

**// model.py**

1. from \_\_future\_\_ **import** unicode\_literals
2. from django.db **import** models
4. **class** Student(models.Model):
5. first\_name = models.CharField(max\_length=20)
6. last\_name  = models.CharField(max\_length=30)
7. **class** Meta:
8. db\_table = "student"

This file contains a class that inherits ModelForm and mention the model name for which HTML form is created.

**// form.py**

1. from django **import** forms
2. from myapp.models **import** Student
4. **class** EmpForm(forms.ModelForm):
5. **class** Meta:
6. model = Student
7. fields = "\_\_all\_\_"

Write a view function to load the ModelForm from forms.py.

**//views.py**

1. from django.shortcuts **import** render
2. from myapp.form **import** StuForm
4. def index(request):
5. stu = EmpForm()
6. **return** render(request,"index.html",{'form':stu})

**//urls.py**

1. from django.contrib **import** admin
2. from django.urls **import** path
3. from myapp **import** views
4. urlpatterns = [
5. path('admin/', admin.site.urls),
6. path('index/', views.index),
7. ]

And finally, create a **index.html** file that contains the following code.

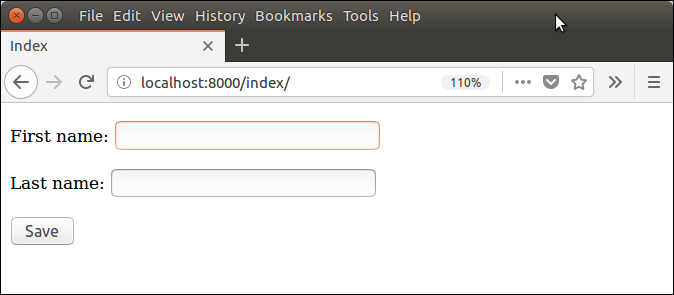
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Index</title>
6. </head>
7. <body>
8. <form method="POST" **class**="post-form">
9. {% csrf\_token %}
10. {{ form.as\_p }}
11. <button type="submit" **class**="save btn btn-default">Save</button>
12. </form>
13. </body>
14. </html>

Run Server

Run the server by using **python manage.py runserver** command.

After that access the template by **localhost:8000/index** URL, and it will produce the following output to the browser.

**Output:**



# Django Forms

Django provides a Form class which is used to create HTML forms. It describes a form and how it works and appears.

It is similar to the **ModelForm** class that creates a form by using the Model, but it does not require the Model.

Each field of the form class map to the HTML form **<input>** element and each one is a class itself, it manages form data and performs validation while submitting the form.

Lets see an example, in which we are creating some fields too.

1. from django **import** forms
2. **class** StudentForm(forms.Form):
3. firstname = forms.CharField(label="Enter first name",max\_length=50)
4. lastname  = forms.CharField(label="Enter last name", max\_length = 100)

A StudentForm is created that contains two fields of CharField type. Charfield is a class and used to create an HTML text input component in the form.

The label is used to set HTML label of the component and max\_length sets length of an input value.

When rendered, it produces the following HTML to the browser.

1. <label **for**="id\_firstname">Enter first name:</label>
2. <input type="text" name="firstname" required maxlength="50" id="id\_firstname" />
3. <label **for**="id\_lastname">Enter last name:</label> <input type="text" name="lastname" required maxlength="100" id="id\_lastname" />

#### **Note: Django Form does not include <form> tags, or a submit button. We'll have to provide those ourselves in the template.**

Commonly used fields and their details are given in the below table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Class** | **HTML Input** | **Empty value** |
| BooleanField | class BooleanField(\*\*kwargs) | CheckboxInput | False |
| CharField | class CharField(\*\*kwargs) | TextInput | Whatever you've given as empty\_value. |
| ChoiceField | class ChoiceField(\*\*kwargs) | Select | '' (an empty string) |
| DateField | class DateField(\*\*kwargs) | DateInput | None |
| DateTimeField | class DateTimeField(\*\*kwargs) | DateTimeInput | None |
| DecimalField | class DecimalField(\*\*kwargs) | NumberInput | None |
| EmailField | class EmailField(\*\*kwargs) | EmailInput | '' (an empty string) |
| FileField | class FileField(\*\*kwargs) | ClearableFileInput | None |
| ImageField | class ImageField(\*\*kwargs) | ClearableFileInput | None |

Let's see a complete example to create an HTML form with the help of Django Form class.

## Building a Form in Django

Suppose we want to create a form to get Student information, use the following code.

1. from django **import** forms
2. **class** StudentForm(forms.Form):
3. firstname = forms.CharField(label="Enter first name",max\_length=50)
4. lastname  = forms.CharField(label="Enter last name", max\_length = 100)

Put this code into the **forms.py** file.

## Instantiating Form in Django

Now, we need to instantiate the form in **views.py** file. See, the below code.

**// views.py**

1. from django.shortcuts **import** render
2. from myapp.form **import** StudentForm
4. def index(request):
5. student = StudentForm()
6. **return** render(request,"index.html",{'form':student})

Passing the context of form into index template that looks like this:

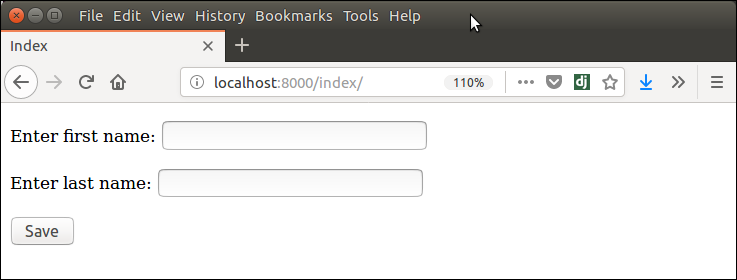
**// index.html**

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Index</title>
6. </head>
7. <body>
8. <form method="POST" **class**="post-form">
9. {% csrf\_token %}
10. {{ form.as\_p }}
11. <button type="submit" **class**="save btn btn-default">Save</button>
12. </form>
13. </body>
14. </html>

Provide the URL in urls.py

1. from django.contrib **import** admin
2. from django.urls **import** path
3. from myapp **import** views
4. urlpatterns = [
5. path('admin/', admin.site.urls),
6. path('index/', views.index),
7. ]

Run Server and access the form at browser by **localhost:8000/index,** and it will produce the following output.



There are other output options though for the <label>/<input> pairs:

* {{ form.as\_table }} will render them as table cells wrapped in <tr> tags
* {{ form.as\_p }} will render them wrapped in <p> tags
* {{ form.as\_ul }} will render them wrapped in <li> tags