استخدام Django Forms

مراجعة لبناء Form عن طريق HTML

قبل البدء بعمل Form، قم بإتباع الخطوات التالية:

- إنشاء مشروع منفصل وتطبيق منفصل بإسم form_example.
- إضافة مجلد templates.
- إضافة ملف HTML بداخل مجلد templates و نسميه form-example.html.

```
django-admin startproject form_project
cd form_project
python manage.py startapp form_example
python manage.py runserver

In settings.py >> in INSTALLED_APPS >> Add form_project
In form_example folder >> New -> Directory >> templates
New -> HTML File
```

- بناء Form بنفس الطريقة التي كنا نبني بها Form باستخدام html.

```
<body>
<form method="post">
    <p>
        <label for="id_text_input">Text Input</label><br>
        <input id="id_text_input" type="text" name= "text_inpu
t" value="" placeholder="Enter some text">
    </p>
    <p>
        <label for="id_password_input">Password Input</label><
br>
        <input id="id_password_input" type="password" name="pa
ssword_input" value="" placeholder="Your password">
    </p>
    <p>
```

```
    <input id="id_checkbox_input" type="checkbox" name="checkb
ox_on" value="Checkbox Checked" checked>
    <label for="id_checkbox_input">Checkbox</label>
    </p>
    <p>
    <input id="id_radio_one_input" type="radio" name="radio_in
put" value="Value One">
    <label for="id_radio_one_input">Value One</label>
    <input id="id_radio_two_input" type="radio" name="radio_in
put" value="Value Two" checked>
    <label for="id_radio_two_input">Value Two</label>
    <input id="id_radio_three_input" type="radio" name="radio_
input" value="Value Three">
    <label for="id_radio_three_input">Value Three</label>
    </p>
    <p>
    <label for="id_favorite_movie">Favorite Movie</label><br>
    <select id="id_favorite_movie" name="favorite_movie">
        <optgroup label="Non-Fiction">
            <option value="1">Movie X</option>
            <option value="2">Movie Z</option>
        </optgroup>
        <optgroup label="Fiction">
            <option value="3">Movie A</option>
            <option value="4">Movie B</option>
        </optgroup>
    </select>
    </p>
    <p>
    <label for="id_movies_you_own">Movies You Own</label><br>
```

```
    <select id="id_movies_you_own" name="movies_you_own" multi
ple>
        <optgroup label="Non-Fiction">
            <option value="1">Movie X</option>
            <option value="2">Movie Z</option>
        </optgroup>
        <optgroup label="Fiction">
            <option value="3">Movie A</option>
            <option value="4">Movie B</option>
        </optgroup>
    </select>
    </p>
    <p>
    <label for="id_text_area">Text Area</label><br>
    <textarea name="text_area" id="id_text_area" placeholder
="Enter multiple lines of text"></textarea>
    </p>
    <p>
    <label for="id_number_input">Number Input</label><br>
    <input id="id_number_input" type="number" name="number_inp
ut" value="" step="any" placeholder="A number">
    </p>
    <p>
    <label for="id_email_input">Email Input</label><br>
    <input id="id_email_input" type="email" name="email_input"
value="" placeholder="Your email address">
    </p>
    <p>
    <label for="id_date_input">Date Input</label><br>
    <input id="id_date_input" type="date" name= "date_input" v
alue="2021-05-29">
```

```
    </p>
    <p>
    <input type="submit" name="submit_input" value="Submit Inp
ut">
    </p>
    <p>
    <button type="submit" name="button_element" value="Button
Element">
        Button With <strong>Styled</strong> Text
    </button>
    </p>
    <input type="hidden" name="hidden_input" value="Hidden Val
ue">
</form>
</body>
```

- بداخل ملف views.py نقوم بإضافة التالي.

```
def form_example(request):
    return render(request, "form-example.html")
```

- عمل URL mapping بداخل ملف urls.py.

```
import form_example.views
path('form-example/', form_example.views.form_example)
```

- تشغيل الموقع وتجربة Form
- إدخال بيانات خاطئة في Form و عمل submit نلاحظ ظهور HTML validation
- إدخال المعلومات بشكل صحيح سوف يظهر ERROR كالتالي: CSRF verification failed. Request aborted

التعامل مع POST Data في View

- بداخل ملف views.py نقوم بكتابة:

```
def form_example(request):
    for name in request.POST:
        print("{}: {}".format(name, request.POST.getlist(nam
e)))
    return render(request, "form-example.html", {"method": req
uest.method})
```

- نطبع نوع method بصفحة html لمعرفة نوع method المستخدمة في إنشاء الصفحة (get أو post)
- نضيف CSRF token حتى نحل المشكلة التي ظهرت لنا سابقا بحيث نسمح بعمل submit للform من دون رفض HTTP Permission

```
<h4>Method: {{ method }}</h4>
<form method="post">
     {% csrf_token %}
```

- نقوم بالدخول للموقع، نلاحظ أن method التي تمت طباعتها هي GET.
- ندخل البيانات ونضغط submit ونلاحظ تغير method إلى post.
- عندما نرجع إلى PyCharm terminal سوف نرى values التي أرسلناها إلى server.

بناء واستدعاء Django Form

تعرفنا سابقا على بناء HTML Form لكن Django سهلت لنا إنشاء Form عن طريق كتابة Python Class باستخدام مكتبة forms وهذا يساعد بتقليل الأكواد البرمجية التي نقوم بكتابتها عن طريق اتباع الخطوات التالية.

- إنشاء forms.py بداخل مجلد التطبيق form_example
- كتابة الأكواد التالية:

```
from django import forms


RADIO_CHOICES = (
    ("Value One", "Value One Display"),
    ("Value Two", "Text For Value Two"),
    ("Value Three", "Value Three's Display Text")
)


MOVIE_CHOICES = (
    ("Non-Fiction", (("1", "Movie X"), ("2", "Movie Z"))),
```

```
    ("Fiction", (("3", "Movie A"),("4", "Movie B")))
)


class ExampleForm(forms.Form):
    text_input = forms.CharField()
    password_input = forms.CharField(widget=forms.PasswordInput)
    checkbox_on = forms.BooleanField()
    radio_input = forms.ChoiceField(choices=RADIO_CHOICES, widget=forms.RadioSelect)
    favorite_movie = forms.ChoiceField(choices=MOVIE_CHOICES)
    movies_you_own = forms.MultipleChoiceField(choices=MOVIE_CHOICES)
    text_area = forms.CharField(widget=forms.Textarea)
    integer_input = forms.IntegerField()
    float_input = forms.FloatField()
    decimal_input = forms.DecimalField()
    email_input = forms.EmailField()
    date_input = forms.DateField(widget=forms.DateInput(attrs={"type": "date"}))
    hidden_input = forms.CharField(widget=forms.HiddenInput, initial="Hidden Value")
```

في الأكواد السابقة قمنا بالتالي:

- استدعاء forms library الخاصة ب Django.
- إنشاء ExampleForm class يقوم بالوراثة من forms.Form class الخاص ب Django.
- إنشاء fields عن طريق الاستفادة من Django Form.
- نفتح ملف views.py ونقوم بكتابة الأكواد التالية:

```
from django.shortcuts import render
from .forms import ExampleForm



def form_example(request):
```

```
    form = ExampleForm()

    for name in request.POST:
        print("{}: {}".format(name, request.POST.getlist(nam
e)))

    return render(request, "form-example.html", {"method": req
uest.method, "form": form})
```

في الأكواد السابقة قمنا بالتالي:

- استدعاء ExampleForm
- تعريف متغير form ويحتوي نسخة (instance) من ExampleForm()
- إرسال المتغير عن طريق context dictionary
- نفتح ملف html ونقوم بتعديله كالتالي:

```
<body>
<h4> Method: {{ method }} </h4>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <p>
        <input type="submit" name="submit_input" value="Submit
Input">
    </p>
    <p>
        <button type="submit" name="button_element" value="But
ton Element">
            Button With <strong>Styled</strong> Text
        </button>
    </p>
</form>
</body>
```

في الأعلى أضفنا {{ form.as_p }} حيث أن as_p هي Method تساعد بعمل render ل Django Form.

- أخيرا، نقوم بتشغيل الموقع وكتابة بعض البيانات وضغط submit
- لاحظ ظهور القيم التي كتبناها في ExampleForm ثم أرسلناها ل server في PyCharm terminal، تظهر على شكل String

الآن وبسبب إنشائنا form بطريقة مختلفة عن HTML Form سوف نلاحظ أننا قللنا الأكواد البرمجية، بالإضافة إلى أن Django Form يوفر لنا طريقة سهلة لعمل form validation.

التعرف على Form Validation

- نفتح ملف forms.py
- نضيف validation ل DecimalField

```
class ExampleForm(forms.Form):
decimal_input = forms.DecimalField(max_digits=3)
```

- نفتح ملف views.py ونقوم بكتابة التالي:

```
from django.shortcuts import render
from .forms import ExampleForm

def form_example(request):
    if request.method == "POST":
        form = ExampleForm(request.POST)
    else:
        form = ExampleForm()

    if request.method == "POST":
        form = ExampleForm(request.POST)
        if form.is_valid():
            for name, value in form.cleaned_data.items():
                print("{}: ({}) {}".format(name, type(value),
value))
```

```
    return render(request, "form-example.html", {"method": req
uest.method, "form": form})
```

- الآن قم بتشغيل الموقع
- للتأكد من Form Validating التي أضفناها، قم بكتابة جميع البيانات المطلوبة ثم أدخل أكثر من ٣ أرقام في Decimal input field ثم اضغط submit
- سوف تلاحظ ظهور Error Message يطلب إدخال أكثر من ٣ أرقام
- بعد تعديل البيانات قم بالرجوع إلى PyCharm terminal، لاحظ أن البيانات المرسلة تغيرت من String إلى Python Object وذلك لأننا قمنا باستخدام cleaned_data

في الأكواد السابقة:

- قمنا باستخدام is_valid() للتحقق من Form هل هو valid؟
- استخدمنا cleaned_data وهو Dictionary يحتوي بيانات يتم تحويلها إلى Python Object يمكننا رؤيتها عندما نفتح PyCharm terminal

إضافة Field Validation

- نفتح ملف forms.py
- نضيف Validation التالية:

```
text_input = forms.CharField(max_length=3)

password_input = forms.CharField(min_length=8, widget=forms.Pa
sswordInput)

movies_you_own = forms.MultipleChoiceField(required=False, cho
ices=MOVIE_CHOICES)

integer_input = forms.IntegerField(min_value=1, max_value=10)

decimal_input = forms.DecimalField(max_digits=5, decimal_place
s=3)
```

- قم بتشغيل الموقع ثم محاولة إدخال Fields بشكل خاطئ حتى تظهر لنا Validation الخاصة بكل Field.
  - في text_input اكتب أكثر من ثلاثة أحروف
  - في password_input اكتب password أقل من ٨
  - في movies_you_own لاتقم باختيار أي خيار
  - في integer_input اكتب عدد ١١
  - في decimal_input اكتب ١.٥٥٥٥

إضافة Model Form

يقدم Django طريقة لإنشاء form من Model Class بشكل تلقائي عن طريق استخدام ModelForm والمطلوب منا فقط تحديد Model الذي نريد إنشاء form منه وأيضا تحديد fields التي سوف نستخدمها.

لإضافة form يقوم بإضافة وتعديل Publisher قم باتباع الخطوات التالية:

- إنشاء ملف forms.py في تطبيق MoviesListApp
- كتابة الأكواد التالية:

```
from django import forms
from .models import Publisher
class PublisherForm(forms.ModelForm):
    class Meta:
        model = Publisher
        fields = "__all__"
```

في الأعلى قمنا بالتالي:

- إنشاء PublisherForm class يرث خصائص ModelForm class الخاص ب Django مثل: validation rules الخاصة ب Model
- إضافة attribute وهو class Meta لتحديد model و fields التي نريد إنشاء form منها.

- تعديل ملف views.py بالشكل التالي:

```
from django.shortcuts import render, get_object_or_404, redire
ct
from .forms import PublisherForm
from .models import Movies_Info, Publisher
from django.contrib import messages
from .utils import average_rating


def publisher_edit(request, pk=None):
    if pk is not None:
        publisher = get_object_or_404(Publisher, pk=pk)
    else:
        publisher = None
```

```
    if request.method == "POST":
        form = PublisherForm(request.POST, instance=publisher)
        if form.is_valid():
            updated_publisher = form.save()
            if publisher is None:
                messages.success(request, "Publisher \"{}\" was created.".format(updated_publisher))
            else:
                messages.success(request, "Publisher \"{}\" was updated.".format(updated_publisher))

            return redirect("publisher_edit", updated_publisher.pk)
    else:
        form = PublisherForm(instance=publisher)

    return render(request, "forms.html", {"method": request.method, "form": form})
```

في الأكواد السابقة قمنا بالتالي:

- استدعاء messages module حتى نطبع Message تحدد مانفذناه على Publisher object
- إضافة Function وهي get_object_or_404 تساعد باسترجاع أي instance بناء على pk و إذا كان object غير موجود ترجع HTTP 404 Not Found

- إنشاء ملف forms.html ويحتوي الأكواد التالية:

```
{% extends 'base.html' %}
{% block content %}

{% for message in messages %}
<p><em>{{ message.level_tag|title }}:</em> {{ message }}</p>
```

```
{% endfor %}

<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <p>
        <input type="submit" value="Submit">
    </p>
</form>

{% endblock %}
```

- داخل ملف urls.py نقوم بكتابة الأكواد التالية:

```
urlpatterns = [
               path('publishers/<int:pk>/', views.publisher_edit, name='publisher_edit'),
               path('publishers/new/', views.publisher_edit, name='publisher_create')
               ]
```

- قم بتشغيل الموقع
- إضافة publisher جديد ثم لاحظ ظهور message عند الإضافة، أيضا لاحظ أن url تم تحديثه بناء على publisher id
- قم بضغط submit مرة أخرى ثم لاحظ ظهور message عند التعديل على البيانات
- قم بالدخول على صفحة Admin والدخول على بيانات Publisher، لاحظ ظهور البيانات التي أضفنا سابقا.