# Django create view

## Django ModelForm¶

- Django ModelForm is an advanced version of django Form.
- ModelForm is associated with a django model.
- Model form can generate auto form fields using form meta class
- Can also define additional form fields.
- Let's look at the code for model and model form.

my_app/models. py

```
from django.db import models

class Contact(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.EmailField(max_length=255)
    phone = models.CharField(max_length=10, null=True)

    class Meta:
        db_table = "contact"
```

my_app/forms. py

```
from django import forms
from my_app.models import Contact

class ContactModelForm(forms.ModelForm):
    class Meta:
        model = Contact
        fields = ['first_name', 'last_name', 'email']
```

- form `Meta` class attributes
    - *model*: defines model name
    - *fields*: form fields to include

## Form template¶

- the template looks like below.

templates/form_template.html

```
<form action="" method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_table }}
    <input type="submit" value="submit" />
</form>
```

# Django Create View¶

## function based createview¶

- use function based create view for simple functionality.
- we will use contact model form for form validation.
- Let's look at the create view code below.

```
from django.http import HttpResponse
from django.shortcuts import render
from my_app.forms import ContactModelForm

def fbv_create_view(request):
    if request.method.lower() == "post":
        form = ContactModelForm(request.POST)
        if form.is_valid():
            instance = form.save()
            return HttpResponse(f"contact saved: {instance}")
    else:
        form = ContactModelForm()
    context = {'form': form}
    return render(request, 'form_template.html', context)
```

## class based createview¶

- Django comes with generic view `CreateView`
- It handles the requests `GET` and `POST`
- When a `POST` request is recieved then django generic view calls two methods `form_valid` and `form_invalid` after validating the requested data.
- `CreateView` also has other methods like `get_context_data`, `get_initial`, etc.
- For current functionality we just need to override `form_valid` method.
- Let's look at the code below.

```
from django.http import HttpResponse
from django.views.generic.edit import CreateView
from my_app.forms import ContactModelForm

class CBVCreateView(CreateView):
```

```
    form_class = ContactModelForm
    template_name = 'form_template.html'

    def form_valid(self, form):
        instance = form.save()
        return HttpResponse(f"contact saved: {instance}")
```

### configure urls¶

- open `my_app/urls.py` and add below code to it.

```
from django.urls import path
from my_app import views

urlpatterns = [
    # ...
    path('fbv_create_view', views.fbv_create_view, name='fbv_create_view'),
    path('cbv_create_view', views.CBVCreateView.as_view(),
name='cbv_create_view'),
    # ...
]
```

- Access the url http://localhost:8000/my_app/fbv_create_view to see the function based create view
- Access the url http://localhost:8000/my_app/cbv_create_view to see the class based create view
- Open above mentioned urls and fill in the contact info and submit the form to save the contact info to the database.

## References¶


- https://docs.djangoproject.com/en/dev/topics/forms/modelforms/
- https://docs.djangoproject.com/en/dev/ref/class-based-views/generic-editing/#django.views.generic.edit.CreateView
- https://ccbv.co.uk/projects/Django/4.1/django.views.generic.edit/CreateView/