Django ORM ModelForm

Django ModelForm

A **ModelForm** is a Django form automatically generated from a model. It saves time by creating fields, validation, and save logic for you.

---

1. Creating a Model

```
# models.py
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=100)
    published_at = models.DateField()

    def __str__(self):
        return self.title
```

---

2. Creating a ModelForm

```
# forms.py
from django import forms
from .models import Book

class BookForm(forms.ModelForm):
    class Meta:
        model = Book             # the model to build the form
from
        fields = ['title', 'author', 'published_at']  # or "__
all__"
```

Optional: customizing widgets

```
class BookForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = '__all__'
        widgets = {
            'published_at': forms.DateInput(attrs={'type': 'da
te'})
        }
```

---

3. Using the ModelForm in a View

Create View

```
# views.py
from django.shortcuts import render, redirect
from .forms import BookForm

def create_book(request):
    if request.method == 'POST':
        form = BookForm(request.POST)
        if form.is_valid():
            form.save()              # saves the model instanc
e
            return redirect('home')
    else:
        form = BookForm()

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

Edit View

```
def edit_book(request, pk):
    book = Book.objects.get(id=pk)

    if request.method == 'POST':
        form = BookForm(request.POST, instance=book)
        if form.is_valid():
            form.save()
            return redirect('home')

    else:
        form = BookForm(instance=book)

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

---

4. Using ModelForm in Templates

Form Rendering Options

Option A: Simple automatic rendering

```
<form method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
</form>
```

Option B: Custom layout

```
<form method="POST">
    {% csrf_token %}

    <label>Title</label>
    {{ form.title }}
```

```
    <label>Author</label>
    {{ form.author }}

    <label>Published At</label>
    {{ form.published_at }}

    <button type="submit">Save</button>
</form>
```

Displaying field errors

```
{{ form.non_field_errors }}
{{ form.title.errors }}
```

---

5. Adding Validation

ModelForm Clean Methods

Field-level validation

```
def clean_title(self):
    title = self.cleaned_data['title']
    if len(title) < 3:
        raise forms.ValidationError("Title must be at least 3 characters.")
    return title
```

Form-level validation

```
def clean(self):
    cleaned = super().clean()
    # add combined validation here
    return cleaned
```

---

6. Saving Without Committing Immediately

```
form = BookForm(request.POST)
if form.is_valid():
    book = form.save(commit=False)  # create object but don't save to DB
    book.author = book.author.title()
    book.save()
```

✓ Summary

Feature	Behavior
`ModelForm`	Automatically generates a form from a model
`fields = [...]` or `"__all__"`	Controls which fields appear
`form.save()`	Saves the model instance
`instance=`	Used for editing existing objects
`form.as_p`, `as_ul`, `as_table`	Quick rendering options
Validation	Use `clean_<field>()` or `clean()`