TRAINING DAY7 REPORT

01 JULY 2025

CREATING FORMS IN DJANGO

A form is an HTML element used to collect data from users. In Django, forms are used to receive input, validate data, and process it safely and efficiently.

Ways to Create Forms in Django:

→ Using HTML Forms (Manual)

→ Using Django Forms (Form class)

→ Using ModelForm (auto-generated from models)

a) Using HTML Forms

You can manually create an HTML form inside a template:

```
<form method="POST" action="/submit/">
    {% csrf_token %}
    <input type="text" name="name" placeholder="Your Name">
    <input type="email" name="email" placeholder="Your Email">
    <button type="submit">Submit</button>
</form>
```

- Requires handling the data manually in the view.
- Use {% csrf_token %} to prevent CSRF attacks.

b) Django Form Class

Create a form using forms.py:

- Provides built-in validation.
- Automatically renders HTML widgets.

c) Django ModelForm

ModelForm is used when the form is directly linked to a database model.

- Simplifies code.
- Saves data to the database directly using form.save().

HANDLING FORM INPUT

Steps:

1. Check if request is POST.
2. Bind data using form = ContactForm(request.POST).
3. Validate using form.is_valid().
4. Save to DB or process as needed.
5. On GET, show an empty form.

GET AND POST METHODS

What is GET?

→ Sends data via URL query parameters.

→ Used for searching, filtering, or non-sensitive data.

→ Does not modify server data.

→ Example URL: /search/?q=django

What is POST?

→ Sends data inside HTTP request body.

→ Used for form submissions, user login, creating records.

→ Data is not visible in the URL.

→ Safer and supports large data.

Differences Between GET and POST:

Feature	**GET**	**POST**
Visibility	URL (visible)	Request body (invisible)
Use Case	Read/search data	Submit/modify data
Data Limit	Limited (URL length)	No significant limit
Caching	Can be cached	Not cached
Security	Less secure (data in URL)	More secure (CSRF protected)

Get and Post examples -->