# 📌 GET and POST Requests in Django main topic Django Forms with POST

Code 30

Video 36

## 🔹 What is an HTTP Request?

Whenever a client (browser, mobile app, API consumer) communicates with the server, it uses HTTP requests.  
The two most common request methods are:

* **GET** → Retrieve data (Read).
* **POST** → Send data to the server (Create/Update/Delete).

## 🌍 1. ****GET Request****

* **Purpose**: Fetch data.
* **Where**: Search bars, filters, links.
* **Data visibility**: Query params in the URL.
* **Idempotent** (safe to repeat).
* **Not secure for sensitive data.**

👉 Example:

*# views.py*

**from** django**.**http **import** HttpResponse

**def** search(**request**)**:**

*# request.GET is a dictionary-like object containing query parameters*

    query **=** request**.**GET**.**get("q")   *# fetch ?q=value from URL*

**return** HttpResponse(f"You searched for: {query}")

🔗 URL:

http://127.0.0.1:8000/search/?q=laptop

📌 Output → You searched for: laptop

✅ Use when:

* Searching / Filtering
* Reading data only
* No sensitive data involved

## 📩 2. ****POST Request****

* **Purpose**: Submit or send data to the server.
* **Where**: Login, Signup, Adding students, Creating blog post, etc.
* **Data location**: Inside the request body (not visible in URL).
* **More secure** (works with CSRF token).
* **Not idempotent** (repeating the same POST can insert duplicate data).

👉 Example:

*# views.py*

**from** django**.**shortcuts **import** render

**from** django**.**http **import** HttpResponse

**def** add\_student(**request**)**:**

**if** request**.**method **==** "POST"**:**

        name **=** request**.**POST**.**get("name")  *# fetch from form input*

        roll **=** request**.**POST**.**get("roll")

**return** HttpResponse(f"Student Added: {name}, Roll: {roll}")

**return** render(request**,** "add\_student.html")

**<**!-- add\_student**.**html --**>**

**<**form method**=**"POST"**>**

    {**%** csrf\_token **%**}

**<**input *type***=**"text" name**=**"name" placeholder**=**"Enter name"**>**

**<**input *type***=**"text" name**=**"roll" placeholder**=**"Enter roll"**>**

**<**button *type***=**"submit"**>**Add Student**</**button**>**

**</**form**>**

✅ Use when:

* Form submission
* Passwords or sensitive data
* Creating/Updating/Deleting data

## 🔑 Key Difference Between GET & POST

| **Feature** | **GET** | **POST** |
| --- | --- | --- |
| Data Location | URL query params | Request body |
| Security | Less secure (visible in URL) | More secure (hidden + CSRF token) |
| Use Case | Search, Filter, Display | Form submission, Login, CRUD |
| Idempotent | ✅ Safe to repeat | ❌ Repeating may duplicate actions |

👉 **Rule of thumb:**

* GET → **Read** only.
* POST → **Write** actions (create/update).

# 📌 Django Forms with POST

Forms in Django make it easier to handle input, validation, and security.

### Example Form

# forms.py

**from** django **import** forms

class **Registration**(*forms***.***Form*)**:**

    name **=** forms**.**CharField()

    email **=** forms**.**EmailField()

    city **=** forms**.**CharField()

    password **=** forms**.**CharField(**widget=**forms**.**PasswordInput)

### Views

# views.py

**from** django**.**shortcuts **import** render

**from** student**.**forms **import** Registration

**from** django**.**http **import** HttpResponseRedirect

**def** register(**request**)**:**

**if** request**.**method **==** 'POST'**:**

*# Bind the submitted data into the form*

        form **=** Registration(request**.**POST)

**if** form**.**is\_valid()**:**   *# Django auto validates required fields & data type*

            name **=** form**.**cleaned\_data['name']

            email **=** form**.**cleaned\_data['email']

            city **=** form**.**cleaned\_data['city']

            password **=** form**.**cleaned\_data['password']

*# Print data in console (for debugging)*

            print("Name:"**,** name)

            print("Email:"**,** email)

            print("City:"**,** city)

            print("Password:"**,** password)

*# Redirect after successful form submission*

**return** HttpResponseRedirect('/student/success') # / must write

**else:**

*# If GET request → show empty form*

        form **=** Registration()

**return** render(request**,** 'student/register.html'**,** {'form'**:** form})

**def** success(**request**)**:**

**return** render(request**,** 'student/success.html')

### Template

**<**!-- register**.**html --**>**

**<**!DOCTYPE html**>**

**<**html**>**

**<**head**>**

**<**title**>**Registration Page**</**title**>**

**</**head**>**

**<**body**>**

**<**h1**>**Registration Form**</**h1**>**

**<**form method**=**"POST" novalidate**>**

        {**%** csrf\_token **%**}

        {{ form**.**as\_p }}   **<**!-- Render form fields --**>**

**<**button *type***=**"submit"**>**Submit**</**button**>**

**</**form**>**

**</**body**>**

**</**html**>**

**<**!-- success**.**html --**>**

**<**h2**>**Registration Successful ✅**</**h2**>**

# 📌 Important Concepts You Missed

## 🔒 1. CSRF Token

* Stands for **Cross Site Request Forgery Protection**.
* Django **forces POST forms** to include {% csrf\_token %}.
* Prevents malicious websites from submitting requests on behalf of logged-in users.

## 🧹 2. Form Validation

* form.is\_valid() → checks for required fields, correct data type, etc.
* form.cleaned\_data → dictionary containing **safe, validated data**.

Example:

**if** form**.**is\_valid()**:**

    data **=** form**.**cleaned\_data   *# {'name': 'John', 'email': 'john@example.com', ...}*

## 🔄 3. Post/Redirect/Get (PRG) Pattern

* Problem: If you **refresh the page** after POST → "Confirm Form Resubmission".
* Solution: **Redirect after POST** using HttpResponseRedirect.

👉 Correct flow:

1. User fills form → Submits → POST request
2. Server saves data → Redirects to success page
3. Success page shown with GET request (no resubmission issue)

## 🛠️ 4. Django Form Rendering Options

* {{ form.as\_p }} → Render as <p> tags
* {{ form.as\_table }} → Render as <tr><td> (good for HTML tables)
* {{ form.as\_ul }} → Render as <li>

## ⚡ 5. Initial Values & Error Handling

You can set **default values** in forms or show **error messages**.

class **Registration**(*forms***.***Form*)**:**

    name **=** forms**.**CharField(**initial=**"Enter your name")

    email **=** forms**.**EmailField(**error\_messages=**{'required'**:** 'Email is mandatory!'})

# ✅ Final Takeaways

* **GET** = read-only, visible in URL, safe to repeat.
* **POST** = submit/write, hidden data, requires CSRF token.
* Django **forms** handle validation and security automatically.
* Always use **HttpResponseRedirect after POST** (PRG pattern).
* Use form.cleaned\_data instead of request.POST (safer, validated).