CSRF Explained in Django

What is CSRF?

CSRF stands for **Cross-Site Request Forgery**. It's a type of security attack where a malicious site tricks a user's browser into making an unwanted request to another site where the user is authenticated.

For example, imagine a user is logged into `example.com`. If they visit a malicious site, it could try to make a request like `POST example.com/delete_account` using the user's session. This could be harmful.

How Django Prevents CSRF

Django includes built-in protection against CSRF attacks:

- It requires a **CSRF token** for POST, PUT, PATCH, and DELETE requests.
- The token must be sent as a header or a hidden form input.

If the token is missing or incorrect, Django will block the request with a **403 Forbidden** error.

What is `@csrf_exempt`?

The `@csrf_exempt` decorator tells Django to **skip CSRF validation** for a particular view. This is commonly used for:

- API endpoints (especially when using JSON in POST requests)
- Testing or development purposes

> ❗ Warning: Skipping CSRF protection makes your view vulnerable to CSRF attacks. Use it carefully and only when necessary (e.g., when using token-based authentication instead of cookies).

Example Usage

```
from django.views.decorators.csrf import csrf_exempt

@csrf_exempt
def example_view(request):
    # your logic here
    pass
```

Explanation of the `request` parameter

In Django views, the first parameter is always `request`:

- It represents the **HTTP request** made by the client.
- It is an object of type `HttpRequest`.
- You can use it to access:
- Request method: `request.method`
- Query parameters: `request.GET`
- Form or JSON body: `request.POST` or `request.body`
- Headers: `request.headers`
- Session and user info: `request.session`, `request.user`

The `request` object is the gateway to everything the client sends to your server.

Let me know if you want to see how CSRF tokens are handled in templates, or how to include them manually in headers using Postman or JavaScript.