Tops Technology

# Module 16)
# Python DB and Framework

Presented By : Nandni Vala

# CRUD Operations using AJAX

## 1.Using AJAX for making asynchronous requests to the server without reloading the page.

- AJAX (Asynchronous JavaScript and XML) allows you to update parts of a webpage without reloading the entire page.
- **Set Up the Backend View**
- Create a Django view to handle the AJAX request and return a response.
- from django.http import JsonResponse
- def example_ajax_view(request):
-     if request.method == "POST":
-         data = request.POST.get("data_key")  # Retrieve data sent from the client
-         response_data = {"message": f"Received: {data}"}
-         return JsonResponse(response_data)  # Return JSON response

- **Add a URL Route**
- Create a URL pattern for the AJAX view.

```
from django.urls import path
from .views import example_ajax_view
urlpatterns = [
    path('ajax/example/', example_ajax_view, name='ajax_example'),
]
```

- **Create the Frontend Template**
- Add an HTML structure and JavaScript code for making the AJAX request.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AJAX in Django</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h1>AJAX Example</h1>
```

```
<form id="ajaxForm">
    <input type="text" name="data_key" placeholder="Enter data" required>
    <button type="submit">Submit</button>
</form>
<p id="responseMessage"></p>
<script>
    $(document).ready(function () {
        $("#ajaxForm").on("submit", function (e) {
            e.preventDefault();  // Prevent form submission
            const formData = $(this).serialize();  // Serialize form data
            $.ajax({
                url: "{% url 'ajax_example' %}",  // URL of the AJAX view
                method: "POST",
            data: formData,
          headers: { "X-CSRFToken": "{{ csrf_token }}" },  // Include CSRF token
          success: function (response) {
            $("#responseMessage").text(response.message);  // Update the page
          },
          error: function () {
            alert("An error occurred. Please try again.");
          }
        });
      });
    });
  </script>
</body>
</html>
```

- **Enable CSRF Protection**
- AJAX requests in Django require a CSRF token for security. Include the {% csrf_token %} template tag in your HTML or send it manually in headers.
- **Explanation of Code**
- **Backend View**:
  - Handles the AJAX request.
  - Returns a JsonResponse with the data.
- **Frontend**:
- **HTML**: Includes a form for user input.
- **jQuery**: Sends the form data to the server using an AJAX POST request.
- **CSRF Token**: Secures the AJAX request against cross-site request forgery.
- **URL**:
- Maps the AJAX request to the corresponding view.