FULLSTACK DEVELOPMENT (21CS62)

Module 5 - JQuery & AJAX Integration

AJAX (Asynchronous Javascript & XML)

- Django provides features for handling asynchronous requests, facilitated by Ajax.
- Ajax allows webpages to update asynchronously with need to reload the page.
- To make use of Ajax, we should have necessary JS libraries in our project.
- Ajax allows for a smoother & more dynamic experience by fetching & displaying data from server in background without disrupting the current page.

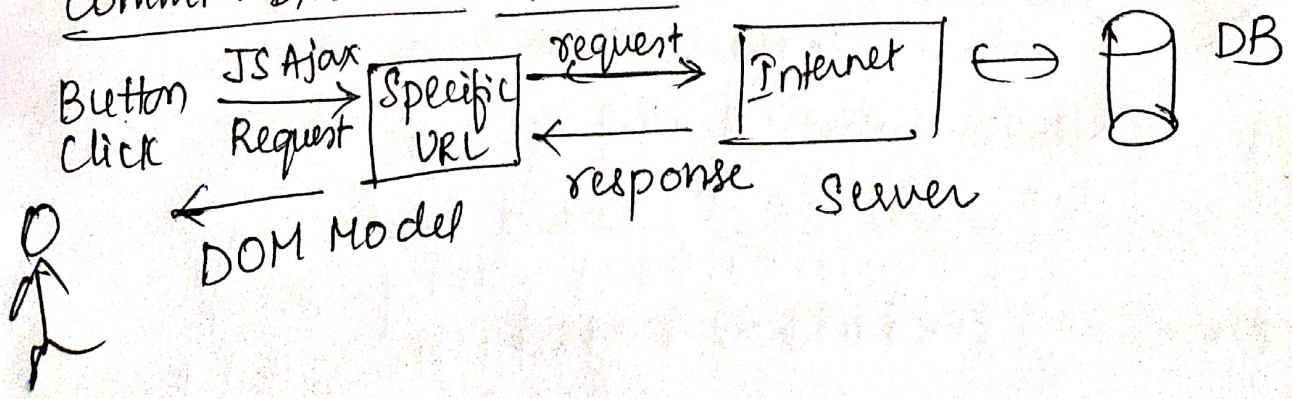
- Client-Side - Javascript -

- On client side JS is used to make async request to server
- Libraries jquery are used to simplify Ajax calls
- JS code is written to handle events - button click & trigger Ajax request to Django server.

- Server side handling - Django views

- Ajax requests handled by views - Views process data → response
- response → JSON, XML, HTML

Commtn b/w Client & Server -



- response is recieved from server – JS code on client process it.
- Depending on content of response – DOM may be updated to reflect changes, display new data, show error msg.

Error Handling:
- client side – error callback — network errors
- server side – Django views — Error responses.

Technologies Ajax is overlaid on –

① Javascript
- Core Role – primary scripting lang
- Interaction with DOM.
- Error Handling

② XML HttpRequest
③ HTML
④ CSS
⑤ XML
⑥ JSON
⑦ Jquery
⑧ Server side Tech – Django
⑨ HTTP/S.

Prgm:- Usage of Ajax in Django

Create appln allow user to submit a form asynchronously using Ajax.

project : ajaxexample
app : ajaxapp

urls.py –
```
from django.urls import path
from ajaxapp import views
urlpatterns = [path('', views.index, name='index'),
path('ajaxsubmit/', views.ajaxsubmit, name=
'ajaxsubmit'),
]
```

index.html page → template for index page

```
<html><head><title> AjaxForm Submission </title>
<script src="https://ajax.googleapis.com/ajax/libs/
        jquery/jquery.min.js">
</script></head>
<body><h1> Ajax Form submission </h1>
<form id = "ajax-form">
<label for="input-data"> Enter Data : </label>
<input type = "text" id = "input-data" name = "input-data">
<button type = "submit"> Submit </button>
</form>
<div id = "result"></div>
<script src = "{% static 'ajaxapp/js/main.js' %}">
</script></body></html>
```

JS code to handle form submission using Ajax

main.js

```
$(document).ready(function(){
$('#ajax-form').submit(function(event){
event.preventDefault()
var inputData = $('#input-data').val();
$.ajax({ type = 'POST'
        url: '/ajax_submit'},
        data={'inputdata': Inputdata},
        success: function(response){
$('#result').text(response.message)},
    error: function(xhr, errmsg, err){
$('#result').text('Error occured');
});});});
```

views.py

```
from django.shortcuts import render
from django.http import JSONResponse
def index(request):
return render(request,'ajaxapp/index.html')
def ajaxsubmit(request):
if request.method=='POST' & request.isajax():
input_data = request.POST.get('input_data')
response_data = {'message':'Form submitted'}
return JSONResponse(response_data)
else
return JSON response({'error':'Invalid Req'})
```

Settings required for JS use in Django

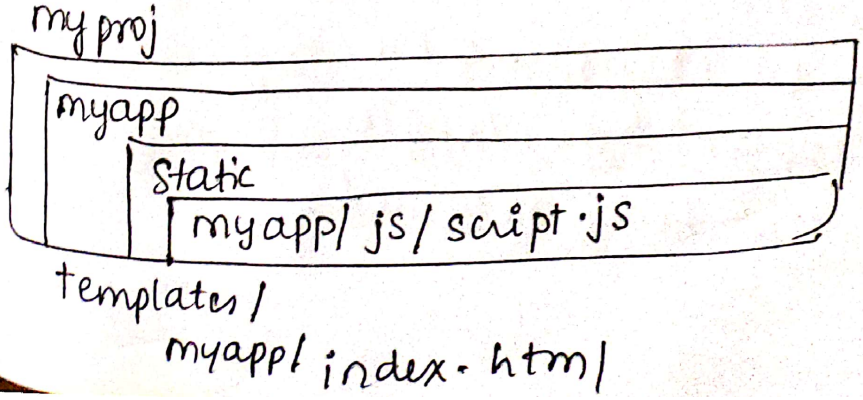
① Django static files setup - In settings.py, we will define the URL to see when reffering to static files

```
STATIC_URL =['/static/'
]
```

Next we will collect all static files into single directory.

```
>python manage.py collectstatic
```

② Organize javascript files



3. Load JS in Templates:

```
<html> <head> <title> Proj </title>
<script src = "{% static 'myapp/js/script.js' %}"></script>
</head> <body> </body> </html>
```

④ Use JS framework or Libraries. — jquery,

```
<script src="https://code.jquery.com/jquery-3.6.0-
min.js"> </script>
```

Download the Library files & place them in static directory.

⑤ Handling Ajax with Django

views.py

```
from django.http import JSONResponse
def m(request):
    data={'message':'Hello'}
    return JSONResponse(data)
```

main.js

```
$.ajax({
    url: '/my-ajax-url/',
    method: 'GET',
    success: function(response){
        alert(response.message);
    }
});
```

{ making Ajax call to Django view

List & explain JQuery Ajax facilities

- jquery simplifies ajax interacts by providing set of methods that allow to easily send HTTP Request, manipulate response & update webpages dynamically.

1) '$.ajax()' - flexible ajax method in jquery
   - Configure various settings & handle complex scenarios
   - url, type, datatype, data, succes, error - key settings.

2) '$.get()' - performing get request
   - url, data, callback, Datatype

3) '$.post()' - post request
   - url, data, callback, datatype

4) '$.getJSON' - GET request that except JSON response
   - url, data, callback

5) '$.getScript()' - load & execute JS file via GET
   - url, callback

6) '$.ajaxsetup()' - setting default headers, global error handling or defining default data type for requests.

7) '$.ajaxStart() & '$.ajaxStop()'

8) '$.ajaxError() & 'ajaxSuccess()'

XHTML HHp Request & Response

- XHTML is variant of HTML that follows rules of XML development.
- HHp Request tells incoming HHp request from a client to the django server. It collects metadata about request like - header, method, URL, query parameter etc. The request has several key components.
  - Request line – HTTP Method (GET, POST), URL of resource, HTTP version
  - Headers – Info about request, Host, UserAgent, Accept (types of content client can handle)
  - Body – Incase of POST request, body may have data sent from client to server

```
GET /example.html HTTP/1.1
Host: www.example.com
Accept - appln /xhtml +xml, text/html, q=0.9.
Accept - Language : en-US, en : q=0.5
Accept÷ Encoding : gzap, deflate
User-Agent : Mozilla/5.0
Connection : Keep alive.
```

- HHp Response - Server respond to client's request with several components.
  - Status Line – OK, Not found etc.
  - Header : - content type, Length
  - Body – content being returned.

HTTP/1.1 200 OK
Content-Type : appln/xhtml+xml, charset=UTF-8
Content-Length : 1024
Content-Language : en-US
Cache-Control : max-age=3600
Last-modified = Mon : 26 August 2023 12:0:00 GMT.
E Tag : "5d8c...."
Server : Apache/2.4.41

Iframe utilization in Django for content loading

① Embedding Django views - Iframe can load content from another view within same project. Useful for displaying diff section of page without refreshing it

```
<iframe src = "{% url 'view name' %}"
  width = '600'  height = '400' > </iframe>
```

② Loading external content - src attribute helps us to load external content - vids, maps, 3rd party services

```
<iframe src = "abc.com" w = --- h = ---> </iframe>
```

③ Dynamic content Loading - change content based on user.

```
<button onclick = "doc. getElementById('myIframe').src =
  {% url 'another-view' %} Load Content </button>
```

④ Security considerations -

```
<iframe src = "abc.com", sandbox = "allow scripts" >
```

⑤ Responsiveness & Design - consistent aspect ratio
↳ good experience

JSON Use with Ajax in Django application

- JSON - lightweight data interchange format that is
  - easy for human to read & write & easy for machine to parse & generate.
  - based on subset of JS programming lang
  - language independent.
  - used for transmitting data b/w server & Webappln.
  - simple, easy, data represented as key value pairs

- Steps to use JSON with AJAX in Django -

i) Create a django view for handling AJAX requests.
ii) return JSON Response from Django view
iii) Make an AJAX call using jQuery.

Eg:- views.py

```
from django.http import JSONResponse
from django.views.decorators.http import require_
    http_methods
def example view (request):
    if request.Method == 'POST' and request.is_ajax():
    data = { 'message': 'Hello',
             'Status': 'success' }
    return JSONResponse(data)
return JSONResponse ( {'error': 'Invalid Request'},
    status = 400)
```

urls.py urlpattern = [path ('ajax/example/', example_view,
name = 'ajax_example'),]

main.js
```
$(document).ready(function(){
$('#myButton').click(function(){
$.ajax({ url : "{%. url 'ajax_example' %.}",
        type : "POST",
        data : {{csrf_token}} },
      datatype : "json",
      success : function(response){
      $('#result').html(response.message);}
    else{
      $('#result').html('An error occured.');
      }
    },
    error: function(xhr, status, error){
    console.log('AJAX request failed :', error);
    }
});  });  });
```

template.html
```
<button id = "myButton"> Send Ajax Request
</button>
<div id = "result"></div>
```

Using Jquery UI Autocomplete in Django -

- Query is a fast, small & feature rich JS library.
- It simplifies various tasks like HTML document traversal - manipulation, event handling, animation & Ajax interaction with web.
- Benifits of JQuery -
  - DOM Manipulation
  - Event Handling
  - AJAX support
  - Cross Browser compatibility
  - Extensibility.
  - Animation

Eg:- myproject/app/views.py

```
from django.shortcuts import render
def index(request):
    return render(request, 'myapp/index.html')
```

index.html

```
<html><head><title><script src=
"https://code.jquery.com/jquery-3.6.0.min.js">
</script>
<script>
    $(document).ready(function(){
    $('#myButton').click(function(){
     alert('Button clicked'); }); });
</script></head>
<body><button id="myButton">Click Me</button>
</body></html>
```

urls.py urlpatterns = [path('', index, name='index'),]

python manage.py runserver

- This setup demonstrates the integration of jquery with Django from scratch.

- create a django view for handling AJAX requests.
- return JSON response from django view.
- write AJAX call using jquery.

AJAX request :-	XML	JSON
Structure	hierarchical with nested elements, supports complex data representation	heirarchical but simple, key-value pair
Metadata	Support through attributes	no native support
Data validation	Robust validation mechanism	No built in validation
Payload	Larger due to verbose syntax	more compact & concise
Performance	Slower to parse & process - verbosity	faster to parse & process due to minimilistic format
readability	less redable	more readable & easy to understand

Practices for optimising performance when using Ajax in Django

1. Minimize data transfer — Send only necessary data
   - Use data compression
2. Optimise backend queries — efficient DB queries
   - indexing
3. Asynchronous Processing — background tasks
   - deffered loading
4. Optimize AJAX requests — Debounce ip
   - Batch requests.
5. Efficient Error Handing — graceful degradation
   - error logging.

Prgm:- Search Appln in Django using AJAX to display courses enrolled by a student -

models.py

```
from django.db import models
class Course(models.Model):
    name = models.CharField(max_length=100)
class Student(models.Model):
    name = models.CharField(max_length=100)
    courses = models.ManyToManyField(Course)
```

forms.py

```
from django import forms
class StudentSearchForm(forms.Form):
    Studentname = forms.CharField(max_length=100)
```

views.py

```
importing render, JSONResponse, Student,
    StudentSearchForm
def search student(request):
    form = StudentSearchForm()
    return render(request, 'searchstudent.html',
        {'form': form})
def search courses(request):
    if request.method == 'GET' & request.is_ajax():
        Studentname = request.GET.get('student_name', '')
        Student = Student.objects.filter(name__icontains=
            student_name).first()
        courses = list(student.courses.values('name'))
        if student else []
        return JSONResponse({'courses': courses})
return JSONResponse({'error': 'Invalid IP',
    status = 400})
```

urls.py

```
[path
urlpattern = [ 'search'/ view.search_student,
    name= 'search_student'),
    path ('search_courses/', view.search_
    courses, name = 'search_courses'),]
```

search_student.html

```
<html> <head> <title> MS </title>
<script src = "https://code.jquery.com/jquery-
    3.6.0.min.js"></script> </head>
<body> <h1> Search for courses </h1>
<form id = "search-form">
  {{ form.as_p }}
<button type = 'submit'> Search </button> </form>
<h2> Courses Enrolled </h2>
<ul id = course-list></ul>
<script>
  $(document).ready(function() {
  $('#search-form').on('submit', function(event) {
  event.preventDefault();
  $.ajax({ url: {%. url "search_courses" %}. },
    type = 'GET', data: $(this).seailaize(),
    success : function(data) {
  $('#course-list').empty();
  if (data.courses.length > 0){
    data.courses.forEach(function(course){
```

```
$('#course-list').append('<li>' + course.name + '</li>');
});
else {
    $("#course.list").append('<li>No courses found</li>'); });
error: function() {
$('#courses-list').empty().append('<li>Error</li>');
} });
});
});
</script></body></html>
```

Methods, Properties of XML, Http, Request, Response Objects -

XML HTTP Request Object

Methods: Open (method, url, asynch)
Send (data)
Set Request Header (head, value)
abort()

Properties- readyState
Status
Response Text
Response XML.

Request Object in Django -

Methods - GET , POST , isajax()

Properties - method - path - User - header

Response Object in Django -

Methods - HttpResponse()
- JsonResponse()

Properties - content_type
- status_code
- charset