WEB APPLICATION INTEGRATION TECHNIQUES

Dr. Bhasha Anjaria
Assistant Professor
PIT, CSE Department

LESSON PLAN

Subject/Course	Mobile App Development
Lesson Title	Web Application Integration Techniques

LESSON OBJECTIVE

Introduction of AsyncTask	**Implementation of Third-Party Library to Fetch Network Data**
Communication with Web API	**Notifications**
Introduction to JSON data	**Telephony API**
JSON Parsing	**Google API**

Introduction to AsyncTask

✓ Introduction to AsyncTask

- Performs **background tasks** without blocking the UI.
- Allows **UI updates** after background execution.
- **Lightweight alternative** to Threads.
- Common uses: **network calls, database operations, file handling**.

Introduction to AsyncTask

✓ Execution Flow of AsyncTask

- onPreExecute() → Runs **before** background task starts (UI Thread).
- doInBackground() → Runs **in background** (Background Thread).
- onProgressUpdate() → Updates progress on **UI Thread**.
- onPostExecute() → Runs **after task completes** (UI Thread).

✓ Flow of Diagram

- **onPreExecute() → doInBackground() → onProgressUpdate() → onPostExecute()**

Introduction to AsyncTask

✓ Advantages of AsyncTask

- **Simplifies** background processing in Android.
- Avoids **Application Not Responding (ANR)** errors.
- Easy to **update UI** from background tasks.
- Handles small tasks **efficiently** without complex thread management.

Introduction to AsyncTask

✓ AsyncTask Example in Android

- Demonstrates background task execution and UI update.
- Example: Fetching data from a server or simulating a delay.

Introduction to AsyncTask

```
private class MyTask extends
AsyncTask<Void, Integer, String> {
   @Override
   protected void onPreExecute() {
      // Before background task

progressBar.setVisibility(View.VISIBL
E)  }
   @Override
   protected String
doInBackground(Void... params) {
      // Background task
      for (int i = 0; i <= 100; i++) {
         publishProgress(i);
         Thread.sleep(50); // Simulate
work  }
```

```
 @Override
   protected void
onProgressUpdate(Integer... values) {
      // Update UI progress

progressBar.setProgress(values[0]);
   }

   @Override
   protected void
onPostExecute(String result) {
      // After task completion
      textView.setText(result);

progressBar.setVisibility(View.GONE)
;
```

Communication with Web API

✓ What is Web API?

- Allows apps to **communicate over the internet**
- Enables **data exchange** between client and server

✓ How it Works?

- Client sends **request → Server processes → Response returned**
-

Communication with Web API

✓ HTTP Requests and Data Formats

- **Request Types:** GET, POST, PUT, DELETE
- **Data Formats:** JSON (most common), XML, CSV
- Examples:
  - GET → fetch data
  - POST → send new data

✓Endpoints and Authentications

- **API Endpoints:** URL paths to access resources (https://api.example.com/users)
- **Authentication Methods:**
  - API Key
  - OAuth
  - Token-based

✓ Advantages and Tools

- **Advantages:**
  - Real-time data fetching
  - Integration with third-party services
  - Automates backend tasks
- **Tools/Libraries:**
  - Android → Retrofit, Volley, OkHttp
  - Web → Axios, Fetch API

Introduction to JSON data

✓ What is JSON?

- **JSON (JavaScript Object Notation)** is a lightweight data format.
- Used to **store and exchange data** between client and server.
- **Text-based** and **human-readable**.
- **Language-independent** — works with Java, Python, C#, PHP, etc.

✓ Syntax and Structure

- Data in **key–value pairs** → "key": "value"
- **Curly braces {}** → represent **objects**
- **Square brackets []** → represent **arrays**
- Keys in **double quotes**
- Values can be: string, number, boolean, object, array, or null

Introduction to JSON data

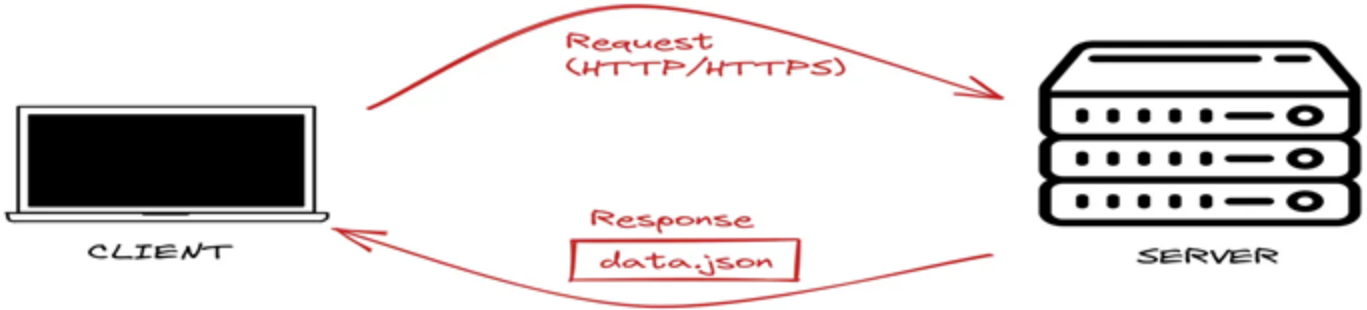
✓ **Example**

```
{
  "name": "Pradeep",
  "age": 24,
  "skills": ["Java", "HTML", "CSS“]
}
```

Introduction to JSON data

✓ JSON in Web Communication

- Client sends a **request** to the server.
- Server responds with **JSON data.**
- JSON helps in **data exchange** between **frontend and backend.**



Introduction to JSON data

✓ Advantages of JSON

- Lightweight & fast
- Easy to read and write
- Supports multiple data types
- Works with almost every programming language
- Used in APIs, configuration files, and databases

✓ JSON Parsing

- JSON parsing is the process of converting JSON (JavaScript Object Notation) data into a format that can be easily used by programming languages. This allows applications to read, manipulate, and utilize data structured in JSON format.

- API data received in JSON format
- Parsing helps:
  - Convert JSON → Java Object
  - Extract required data for app use

✓ JSON Parsing Methods

- Using **org.json** (built-in)
- Using **Gson** (Google library)
- Using **Jackson** (powerful library)

✓ org.json Example

```
JSONObject obj = new JSONObject("{\"name\":\"Rajesh\",\"age\":25}");

String name = obj.getString("name");

int age = obj.getInt("age");
```

- Simple and available in Android SDK

✓ Gson Example

```
Gson gson = new Gson();

Person p = gson.fromJson("{\"name\":\"Rajesh\",\"age\":25}", Person.class);
```

- Converts JSON ↔ Java objects easily

  Requires dependency:

  implementation 'com.google.code.gson:gson:2.10.1'

✓ Jackson Example

```
ObjectMapper mapper = new ObjectMapper();

Person person = mapper.readValue(jsonString, Person.class);
```

- Used in enterprise projects

  Handles complex JSON structures

✓ Parsing JSON from API

```
URL url = new URL("https://api.example.com/data.json");

BufferedReader reader = new BufferedReader(new

InputStreamReader(url.openStream()));

StringBuilder json = new StringBuilder();
```

Implementation of Third-Party Library to Fetch Network Data

✓ Introduction

- Android apps often need to **fetch data from web servers (APIs)**
- Doing this manually with HttpURLConnection is complex
- To simplify, developers use **Third-Party Libraries**

Implementation of Third-Party Library to Fetch Network Data

✓ What is Third-Party Library?

- Ready-made external code developed by others
- Helps perform tasks faster and with fewer errors
- Integrated using **Gradle dependencies**

✓ Popular Networking Libraries

- **Retrofit** – Most popular for REST APIs
- **Volley** – Lightweight and fast
- **OkHttp** – Handles HTTP requests at a low level

✓ Retrofit Libraries

- Developed by **Square**
- Converts JSON response into Java objects automatically
- Works well with **Gson** for JSON parsing
- **Gradle Dependency:**

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

✓ Retrofit Basic example

API Interface	MainActivity

public interface ApiService {
@GET("users")
Call<List<User>>
getUsers();
}
	Retrofit retrofit = new Retrofit.Builder()
	.baseUrl("https://api.example.com/")
	
	.addConverterFactory(GsonConverterFactory.
	create())
	.build();
	
	ApiService service =
	retrofit.create(ApiService.class);

Implementation of Third-Party Library to Fetch Network Data

Volley Library

- Developed by **Google**
- Best for **small and frequent requests**
- Automatically handles caching and threading
- **Dependency:**

```
implementation 'com.android.volley:volley:1.2.1'
```

Volley Example

```
String url = "https://api.example.com/data.json";
JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, null,
    response -> Log.d("Response", response.toString()),
    error -> Log.e("Error", error.toString()));
Volley.newRequestQueue(context).add(request);
```

OkHttp Library

- Also developed by **Square**
- Used internally by Retrofit
- Provides powerful features for custom HTTP requests
- **Dependency:**

```
implementation 'com.squareup.okhttp3:okhttp:4.12.0'
```

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder()

.url("https://api.example.com/data.json")
    .build();
```

```
client.newCall(request).enqueue(new Callback() {
    public void onResponse(Call call, Response response) {
        Log.d("Data", response.body().string());
    }
    public void onFailure(Call call, IOException e) {
        e.printStackTrace();
    }
});
```

Implementation of Third-Party Library to Fetch Network Data

✓ Advantages of using Libraries

- Less code, faster development
- Handles network errors and caching
- Easy JSON integration
- Secure and efficient communication

Notification in Android

✓ Introduction

- Notifications alert users about **events or updates**
- Can appear in:
  - Status bar
  - Lock screen
  - Notification drawer
- Helps **keep users engaged**

Notification in Android

✓ Types of Notifications

- **Simple Notification –** Shows title and message
- **Big Text / Inbox Style –** Expands to show more info
- **Action Notification –** Includes buttons for user actions
- **Progress Notification –** Shows progress (e.g., file download)

Notification in Android

✓ Basic Notification Example

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "CHANNEL_ID")
    .setSmallIcon(R.drawable.ic_notification)
    .setContentTitle("New Message")
    .setContentText("You have a new notification")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT);
NotificationManagerCompat manager = NotificationManagerCompat.from(this);
manager.notify(1, builder.build());
```

Notification in Android

Steps to show Notification

- **Create Notification Channel** (Android 8.0+)

```
NotificationChannel channel = new NotificationChannel("CHANNEL_ID", "My Channel", NotificationManager.IMPORTANCE_DEFAULT);
NotificationManager manager = getSystemService(NotificationManager.class);
manager.createNotificationChannel(channel);
```

- **Build Notification** using NotificationCompat.Builder
- **Show Notification** using NotificationManagerCompat.notify()

Notification in Android

✓ Key Notes

- **Notification Channel** is mandatory for Android 8.0+
- Always use **small icon** (setSmallIcon())
- Can add:
  - **Sound** → setSound()
  - **Vibration** → setVibrate()
  - **Action buttons** → addAction()

Notifications in Android

✓ Use Cases

- New message alerts
- App updates
- Reminders / Alarms
- Download progress

Telephony API in Android

✓ Introduction

- **Telephony API** allows Android apps to **interact with phone services.**
- Provides information such as:
    - Phone network
    - SIM details
    - Call status
    - Device identifiers

✓ Key classes in Telephony API

- **TelephonyManager –** Main class to access phone info
- **SmsManager –** Send SMS programmatically
- **PhoneStateListener –** Monitor call and network state

✓ TelephonyManager Example

```
TelephonyManager tm = (TelephonyManager)
getSystemService(TELEPHONY_SERVICE);
String networkOperator = tm.getNetworkOperatorName();
String simSerial = tm.getSimSerialNumber();
String deviceId = tm.getDeviceId(); // Deprecated in API 26+
Log.d("TELEPHONY", "Operator: " + networkOperator + ", SIM: " + simSerial);
```

✓ PhoneStateListner Example

```
PhoneStateListener listener = new PhoneStateListener() {
   @Override
   public void onCallStateChanged(int state, String incomingNumber) {
      if(state == TelephonyManager.CALL_STATE_RINGING) {
         Log.d("CALL", "Incoming number: " + incomingNumber);
      }}};
TelephonyManager tm = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
tm.listen(listener, PhoneStateListener.LISTEN_CALL_STATE);
```

✓ Permission Required

- Required in AndroidManifest.xml:

```
<uses-
permissionandroid:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
```

✓ Use cases of Telephony API

- Detect incoming calls
- Send SMS programmatically
- Read SIM / network info
- Implement call logs or caller features

Google API in Android

✓ Introduction

- Google APIs allow apps to **access Google services**
- Commonly used for:
  - Maps & Location
  - Authentication (Google Sign-In)
  - Firebase services (Analytics, Messaging, Storage)
  - Drive & Calendar integration

✓ Key Google APIs for Android

- **Google Maps API –** Display maps, markers, routes
- **Google Sign-In API –** Authenticate users with Google account
- **Google Drive API –** Upload/download files
- **Firebase API –** Push notifications, analytics, storage
- **Places API –** Get information about nearby places

✓ Google Maps Example

```
SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
      .findFragmentById(R.id.map);
mapFragment.getMapAsync(new OnMapReadyCallback() {
   @Override
   public void onMapReady(GoogleMap googleMap) {
      LatLng location = new LatLng(28.6139, 77.2090); // New Delhi
      googleMap.addMarker(new MarkerOptions().position(location).title("Marker
in Delhi"));
      googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(location,
10));} });
```

✓ Google Sign-In Example

```
GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
    .build();
GoogleSignInClient mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
Intent signInIntent = mGoogleSignInClient.getSignInIntent();
startActivityForResult(signInIntent, RC_SIGN_IN);
```

✓ Key Notes

- **Requires API Key** for most Google APIs
- Must enable API in **Google Cloud Console**
- Always check permissions before accessing services

SIX PHRASE
a Veranda Enterprise

THANK YOU!