UNIVERSITY OF PLYMOUTH

COMP2000: Software engineering 2

Week-7: Notifications, Web services and APIs

Outline

- Notification
- Web services
- Consuming REST APIs
- Async Tasks
- Mid-module review

Notifications

- A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app.

- Users can tap the notification to open your app or take an action directly from the notification.

To get started, you need to set the notification's content and channel using a NotificationCompat.Builder object.

The following example shows how to create a notification with the following:

•A small icon, set by setSmallIcon(). This is the only user-visible content that's required.
•A title, set by setContentTitle().
•The body text, set by setContentText().
•The notification priority, set by setPriority().

•The priority determines how intrusive the notification should be on Android 7.1 and lower. (For Android 8.0 and higher, you must instead set the channel importance.)

Syntax:

```
• NotificationCompat.Builder builder = new
  NotificationCompat.Builder(this, CHANNEL_ID)
          .setSmallIcon(R.drawable.notification_icon)
          .setContentTitle(textTitle)
          .setContentText(textContent)
          .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Create a Notification: Example

```
public void showNotification(){
   NotificationCompat.Builder builder = new NotificationCompat.Builder(this, "0")
         .setSmallIcon(R.drawable.logo)
         .setContentTitle("Notification")
         .setContentText("University of Plymouth")
         .setPriority(NotificationCompat.PRIORITY_DEFAULT);
}
```

Create a notification channel

```
private void createNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is not in the Support Library.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
importance);
        channel.setDescription(description);
        // Register the channel with the system; you can't change the importance
        // or other notification behaviors after this.
        NotificationManager notificationManager =
getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Create a pending Intent

```
Intent notificationIntent = new Intent(this, MenuActivity.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent,
        PendingIntent.FLAG_UPDATE_CURRENT);
```

Show the notification

```
NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);

// notificationId is a unique int for each notification that you must define
    notificationManager.notify(0, builder.build());
```

Explore more details regarding notifications in the following link:
https://developer.android.com/develop/ui/views/notifications/build-notification

Web services and API

Networks

- Remember, a network knows how to transmit and receive bytes. That's all it knows.

- We (the programmer) must

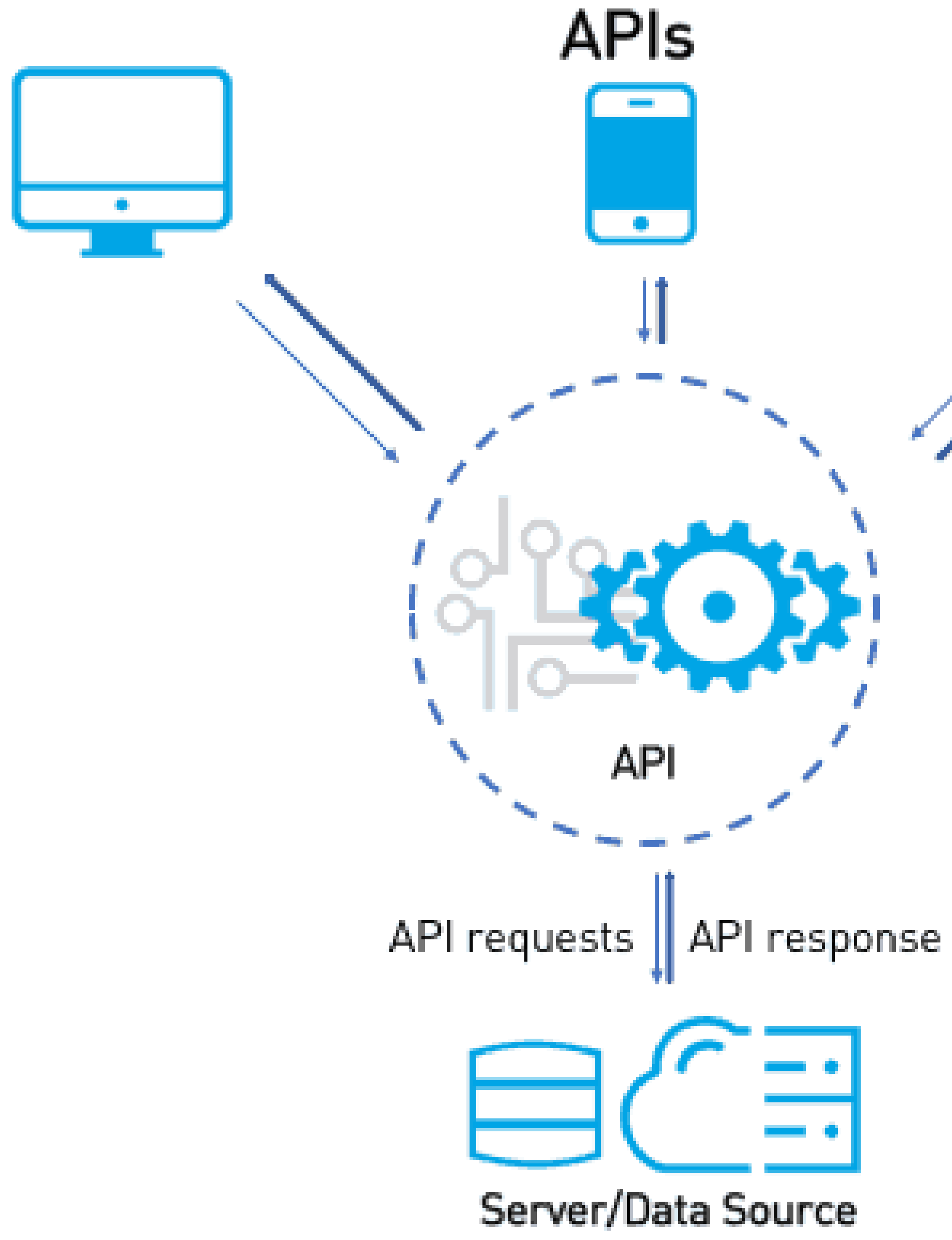
  - Define interface which defines how the applications will communicate.

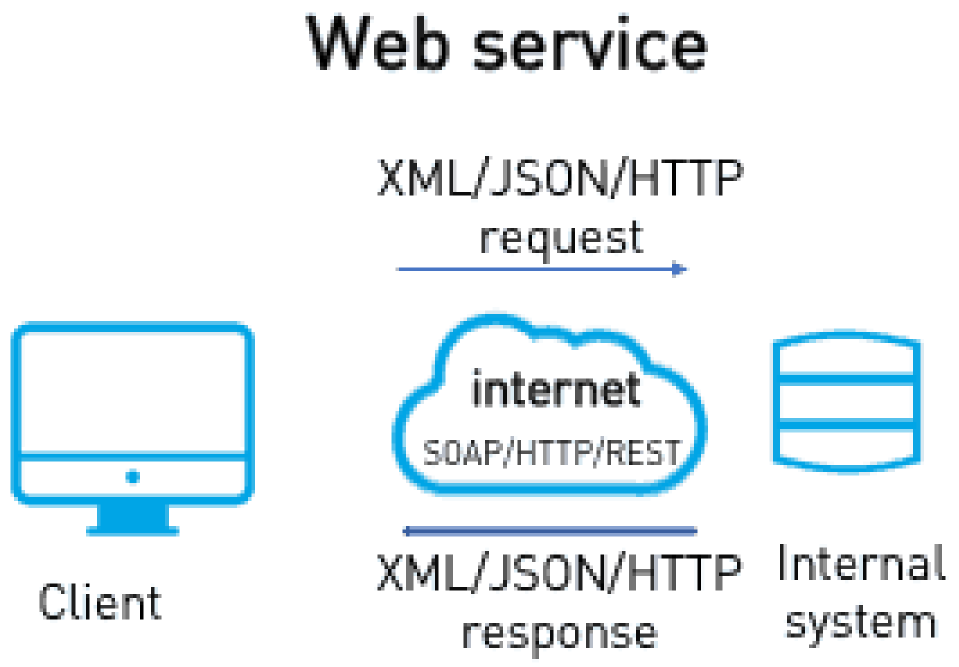
  - Provide mechanism to serialise (Java) or marshal data by flattening a complicated data structure into a stream of bytes. And un-marshal a stream of bytes into a data structure.

  - Maybe we need mechanism to translate data formats on one computer into those on another. (Internet application layer protocols tend to solve this one by encoding everything as text).

Web Services

- Web services are a type of API, which must be accessed through a network connection.
- API: Application Programming Interface
- We will spend most time on Web Services, for a variety of reasons.
- Don't confuse a Web Service with a web site!
- The W3C defines a "web service" as "a software system designed to support interoperable machine-to-machine interaction over a network.
- Web services are completely heterogeneous, multi-language, multi-platform.
- Neither client or server need be written in Java.





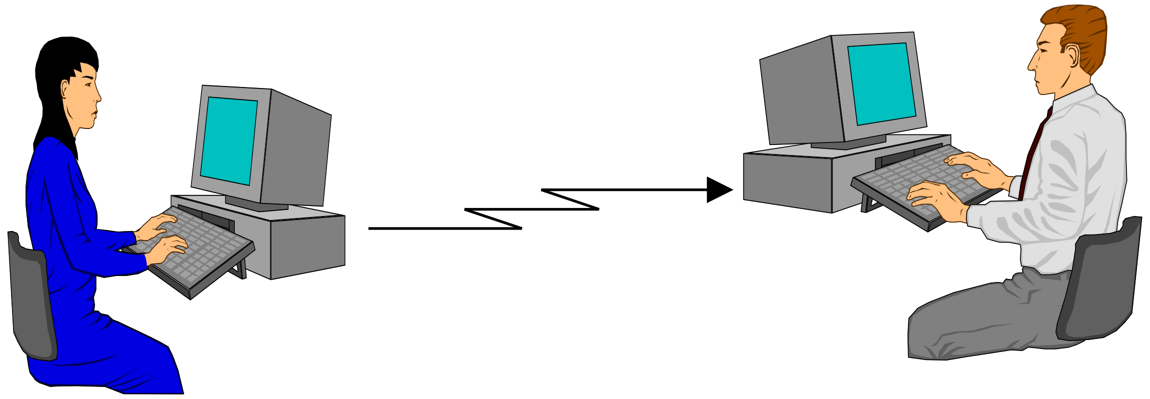
[API vs Web Service: What's the Difference? (with Examples) | RapidAPI]

- Check these websites for more information
- https://www.geeksforgeeks.org/differences-between-web-services-and-web-api/

- https://rapidapi.com/blog/api-vs-web-service/

Web Services

- Web Services are much used for B2B (Business to Business) transactions.



- ...But web services have quite a heavy overhead for networks and processing.
- All web services sit on top of HTTP (GET, PUT etc.), and therefore use port 80.
- And are consequently unlikely to be unaffected by corporate or personal firewalls.

SOAP Web Services

web services are divided into two basic types, SOAP and RESTFful.

- SOAP (Simple Object Assess Protocol)
  - Clear what the rules are (SOAP’s strength)
  - A website exposes an endpoint – an XML schema

SOAP Web Services

- The WSDL contains pretty much everything you need to know about the web service
  - what the methods you can invoke are called
  - what parameters they take
  - What values are returned

- SOAP is complicated, but any IDEs, including NetBeans, can parse the WSDL and automatically import the SOAP web service.

RESTful Web Services

- RESTful web services (Representational State Transfer) seem to be the way the world is going.
- They are considerably leaner and meaner (than SOAP)
- Basically, they are HTTP methods of GET, PUT, DELETE.
- And, there are much fewer rules than in the SOAP encoded ones.
- One of the core concepts of REST seems to be that the URL doesn’t map directly onto a server side script, it maps onto a resource, e.g. there is no server side script or page with the URL
- Data you may send / receive can be encoded in any way you like, but the common ways are
    - JSON (JavaScript Object Notation)
    - XML
    - HTML
    - Plain text

Volley Library

- Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster.
- Volley is available on GitHub.

Adding internet access permissions to the Manifest file

```
<uses-permission android:name="android.permission.INTERNET">
</uses-permission>
```

Installing the dependencies

Add the Volley library in the app level build.gradle file.

```
dependencies {

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

Standard request using Volley

- String Request
- JSON Request

- Will use the following API as an example:
- https://jsonplaceholder.typicode.com/todos/

- Create a simple request using StringRequest

```
RequestQueue queue = Volley.newRequestQueue(MainActivity.this);

String url ="https://jsonplaceholder.typicode.com/todos/";
```

```
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
Toast.makeText(MainActivity.this,response,Toast.LENGTH_LONG).show();
}
```

- In case of Errors, we should implement ErrorListener

```
}, new Response.ErrorListener() {
@Override
public void onErrorResponse(VolleyError error) {
    Toast.makeText(MainActivity.this,"Error",Toast.LENGTH_LONG).show();
}
```

- Then we should add the request to the RequestQueue.

```
queue.add(stringRequest);
```

Jason Request

Volley provides the following classes for JSON requests:

- JsonArrayRequest: A request for retrieving a JSONArray response body at a given URL.

- JsonObjectRequest: A request for retrieving a JSONObject response body at a given URL, allowing for an optional JSONObject to be passed in as part of the request body.

- Create a JsonArrayRequest

```
JsonArrayRequest jsonArrayRequest = new JsonArrayRequest(Request.Method.GET,url, null,
new Response.Listener<JSONArray>(){

}
```

- Implement onResponse method

```
@Override
public void onResponse(JSONArray response) {
    textView.setText("Response: " + response.toString());
    Toast.makeText(MainActivity.this,response.toString(),Toast.LENGTH_LONG)
    .show();
}
```

- Implement onErrorResponse method

```
public void onErrorResponse(VolleyError error) {
    Toast.makeText(MainActivity.this,"something went wrong",Toast.LENGTH_LONG)
.show();

}
```

```
// Add request.
 queue.add(jsonArrayRequest);
```

Get JASON object

```
JSONObject object= response.getJSONObject(0);

 title=object.getString("title");
```

- For more details regarding Volley library and setting requests, please visit the following website :
- https://developer.android.com/training/volley

AsyncTask class

Deprecated AsyncTask in Android, alternatively we could use:

1. **A Thread (we covered in previous lecture)**
2. **A combination of an Executor and a Handler (more details here: https://developer.android.com/guide/background/asynchronous/java-threads**

Async Task

Defining an AsyncTask

Create the AsyncTask method for downloading network data from the API, and implement the onPreExecute(), doInBackground(), and onPostExecute() methods.

```
public class MyAsyncTasks extends AsyncTask<String, String, String>
{
....
}
```

```java
@Override
protected void onPreExecute() {
    super.onPreExecute();
    // display a progress dialog for good user experience

}
```

Example on using progress dialog

```java
progressDialog = new ProgressDialog(MainActivity.this);
progressDialog.setMessage("processing results");
progressDialog.setCancelable(false);
progressDialog.show();
```

```
@Override
protected String doInBackground(String... params) {
    // Fetch data from the API in the background.

    return result;
}
```

```java
@Override
protected void onPostExecute(String s) {
    super.onPreExecute();
    // dismiss the progress dialog after receiving data from API
    progressDialog.dismiss();
}
```

To implement AsyncTask, create an object in the MainActivity and then execute it.

```
MyAsyncTasks myAsyncTasks = new MyAsyncTasks();
myAsyncTasks.execute();
```

Important note regarding AsyncTask and thread

- Don’t forget to kill all your threads when your Activity (or Fragment) enters a paused state.
- stop ()
- suspend()
- cancel(true)

- If you leave Threads or AsyncTasks running after your activity has gone, weird things will happen.

- If you want to leave a background task permanently running, use a Service.

```
@Override
protected void onCancelled() {
    super.onCancelled();
    // do something
}
```

- Check-in code:

Any comment/ feedback/ question regarding the today lecture?

- Menti.com
- **1730 0111**

Mid module review:
Could you give feedback regarding COMP2000?

- Menti.com
- **1730 0111**

Thank you