How to Execute HTTP POST Requests in Android

HTTP Post is part of a deprecated HTTP classes like org.apache.http and AndroidHttpClient as of Android 5.1.https://developer.android.com/about/versions/android-5.1.html#http Migrate your code to the HttpURLConnection classes which...

Part 1 of 2:

Creating a Try Block and HttpURLConnection Object

1. **Add internet permissions to the Android Manifest.** The Android Manifest is an XML file that provides important information to the Android system that determines device compatibility and access to features. In the 'AndroidManifest.xml' file, enter the following line to provide internet access.

   ```
   android:name="android.permission.INTERNET" />
   ```

2. **Create a try block.** In Java, a try statement is an exception handler that will prevent the program from crashing if it is not able to perform an action. Because this will require a connection to a network location, a try statement will catch an exception if it is not able to establish a connection. You can add this to a new Java method.[2]

   ```
   try { //enter statements that can cause exceptions }
   ```

3. **Build HttpURLConnection and URL objects.** Java is an Object-Oriented language. An object is comprised of states and behaviors which is an instance of a class. The HttpURLConnection object sends and receives data over the internet. In your code, begin your new method by creating a URL object and assign it a URL for the HttpURLConnection object to connect to. [3]

   ```
   URL url = new URL('http://exampleurl.com/'); HttpURLConnection client =
    (HttpURLConnection) url.openConnection();
   ```

   1. For best practice, establish the URL and HttpURLConnection objects outside of the try block to make it easier to catch exceptions.

   ```
   URL url = new URL('http://exampleurl.com/'); HttpURLConnection client =
    null; try { client = (HttpURLConnection) url.openConnection(); }
   ```

Part 2 of 2:

Posting the Output Request and Handling Exceptions

1. **Set the request method to Post.** In order to send information to the server, you must set the HttpURLConnection object's type to post and set the output to true using setDoOutput(). Use the

setRequestProperty() function to set a general request property which requires two elements, a key with a known request from the server and a value that's contained within the corresponding key.

1. The setRequestProperty() function is used as the Accept-Encoding request header to disable automatic decompression.

```
client.setRequestMethod('POST'); client.setRequestProperty('Key','
Value'); client.setDoOutput(true);
```

2. **Output the stream to the server.** You must request the output stream from the server in order to be able to write to the output stream, or post, then flush and close the stream when finished.

```
OutputStream outputPost = new BufferedOutputStream(client.
getOutputStream()); writeStream(outputPost); outputPost.flush();
outputPost.close();
```

    1. For performance reasons, it's a good idea to let the server know how large in bytes the content will be. The best method is setFixedLengthStreamingMode(int) when the body length is known,[4] whereas setChunkedStreamingMode(int) is used if it's length is not known.[5] Not using either of the previous methods causes the HttpURLConnection object to buffer the complete body in memory before being transmitted.

```
client.setFixedLengthStreamingMode(outputPost.getBytes().length);
client.setChunkedStreamingMode(0);
```

3. **Catch any exceptions.** After the try statement, use the catch block to check for an exception for input and output with IOException, catch a URL error with the MalformedURL exception and check if the URL does not provide a response in time with the SocketTimeoutException.

```
catch(MalformedURLException error) {
//Handles an incorrectly entered URL } catch(SocketTimeoutException
error) { //Handles URL access timeout. } catch (IOException error) {
//Handles input and output errors }
```

4. **Disconnect from the URL.** After you are finished with your URL connection, you will need to disconnect from the URL. Be sure to check that you are connected to a URL before attempting to disconnect.

```
finally { if(client != null) // Make sure the connection is not null.
client.disconnect(); }
```