**Lab Session :11**

Working in Background

**MainActivity.java**

*/\*  
 \* Copyright (C) 2018 Google Inc.  
 \*  
 \* Licensed under the Apache License, Version 2.0 (the "License");  
 \* you may not use this file except in compliance with the License.  
 \* You may obtain a copy of the License at  
 \*  
 \* http://www.apache.org/licenses/LICENSE-2.0  
 \*  
 \* Unless required by applicable law or agreed to in writing, software  
 \* distributed under the License is distributed on an "AS IS" BASIS,  
 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 \* See the License for the specific language governing permissions and  
 \* limitations under the License.  
 \*/*package com.example.android.whowroteitloader;  
import android.content.Context;  
import android.net.ConnectivityManager;  
import android.net.NetworkInfo;  
import android.support.annotation.NonNull;  
import android.support.annotation.Nullable;  
import android.support.v4.app.LoaderManager;  
import android.support.v4.content.Loader;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.view.inputmethod.InputMethodManager;  
import android.widget.EditText;  
import android.widget.TextView;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
*/\*\*  
 \* The WhoWroteIt app queries the Book Search API for books based  
 \* on a user's search. It uses an AsyncTask to run the search task in  
 \* the background.  
 \*/*public class MainActivity extends AppCompatActivity  
 implements LoaderManager.LoaderCallbacks<String> {  
  
 private EditText mBookInput;  
 private TextView mTitleText;  
 private TextView mAuthorText;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 mBookInput = findViewById(R.id.bookInput);  
 mTitleText = findViewById(R.id.titleText);  
 mAuthorText = findViewById(R.id.authorText);  
  
 if (getSupportLoaderManager().getLoader(0) != null) {  
 getSupportLoaderManager().initLoader(0, null, this);  
 }  
 }

*/\*\*  
 \* onClick handler for the "Search Books" button.  
 \*  
 \* @param view The view (Button) that was clicked.  
 \*/* public void searchBooks(View view) {  
 *// Get the search string from the input field.* String queryString = mBookInput.getText().toString();  
  
 *// Hide the keyboard when the button is pushed.* InputMethodManager inputManager = (InputMethodManager)  
 getSystemService(Context.INPUT\_METHOD\_SERVICE);  
 if (inputManager != null) {  
 inputManager.hideSoftInputFromWindow(view.getWindowToken(),  
 InputMethodManager.HIDE\_NOT\_ALWAYS);  
 }  
  
 *// Check the status of the network connection.* ConnectivityManager connMgr = (ConnectivityManager)  
 getSystemService(Context.CONNECTIVITY\_SERVICE);  
 NetworkInfo networkInfo = null;  
 if (connMgr != null) {  
 networkInfo = connMgr.getActiveNetworkInfo();  
 }  
  
 *// If the network is available, connected, and the search field  
 // is not empty, start a BookLoader AsyncTask.* if (networkInfo != null && networkInfo.isConnected()  
 && queryString.length() != 0) {  
  
 Bundle queryBundle = new Bundle();  
 queryBundle.putString("queryString", queryString);  
 getSupportLoaderManager().restartLoader(0, queryBundle, this);  
  
 mAuthorText.setText("");  
 mTitleText.setText(R.string.loading);  
 }  
 *// Otherwise update the TextView to tell the user there is no  
 // connection, or no search term.* else {  
 if (queryString.length() == 0) {  
 mAuthorText.setText("");  
 mTitleText.setText(R.string.no\_search\_term);  
 } else {  
 mAuthorText.setText("");  
 mTitleText.setText(R.string.no\_network);  
 }  
 }  
 }  
  
  
 @NonNull  
 @Override  
 public Loader<String> onCreateLoader(int id, @Nullable Bundle args) {  
 String queryString = "";  
  
 if (args != null) {  
 queryString = args.getString("queryString");  
 }  
  
 return new BookLoader(this, queryString);  
 }  
  
 @Override  
 public void onLoadFinished(@NonNull Loader<String> loader, String data) {  
 try {  
 *// Convert the response into a JSON object.* JSONObject jsonObject = new JSONObject(data);  
 *// Get the JSONArray of book items.* JSONArray itemsArray = jsonObject.getJSONArray("items");

*// Initialize iterator and results fields.* int i = 0;  
 String title = null;  
 String authors = null;  
  
 *// Look for results in the items array, exiting when both the  
 // title and author are found or when all items have been checked.* while (i < itemsArray.length() &&  
 (authors == null && title == null)) {  
 *// Get the current item information.* JSONObject book = itemsArray.getJSONObject(i);  
 JSONObject volumeInfo = book.getJSONObject("volumeInfo");  
  
 *// Try to get the author and title from the current item,  
 // catch if either field is empty and move on.* try {  
 title = volumeInfo.getString("title");  
 authors = volumeInfo.getString("authors");  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
  
 *// Move to the next item.* i++;  
 }  
  
 *// If both are found, display the result.* if (title != null && authors != null) {  
 mTitleText.setText(title);  
 mAuthorText.setText(authors);  
 *//mBookInput.setText("");* } else {  
 *// If none are found, update the UI to show failed results.* mTitleText.setText(R.string.no\_results);  
 mAuthorText.setText("");  
 }  
  
 } catch (Exception e) {  
 *// If onPostExecute does not receive a proper JSON string,  
 // update the UI to show failed results.* mTitleText.setText(R.string.no\_results);  
 mAuthorText.setText("");  
 e.printStackTrace();  
 }  
  
 }  
  
 @Override  
 public void onLoaderReset(@NonNull Loader<String> loader) {  
 *// Do nothing. Required by interface.* }  
}

BookLoader:

*/\*  
 \* Copyright (C) 2018 Google Inc.  
 \*  
 \* Licensed under the Apache License, Version 2.0 (the "License");  
 \* you may not use this file except in compliance with the License.  
 \* You may obtain a copy of the License at  
 \*  
 \* http://www.apache.org/licenses/LICENSE-2.0  
 \*  
 \* Unless required by applicable law or agreed to in writing, software  
 \* distributed under the License is distributed on an "AS IS" BASIS,  
 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 \* See the License for the specific language governing permissions and  
 \* limitations under the License.  
 \*/*

package com.example.android.whowroteitloader;  
  
import android.content.Context;  
import android.support.annotation.Nullable;  
import android.support.v4.content.AsyncTaskLoader;  
  
public class BookLoader extends AsyncTaskLoader<String> {  
  
 private String mQueryString;  
  
 BookLoader(Context context, String queryString) {  
 super(context);  
 mQueryString = queryString;  
 }  
  
 @Override  
 protected void onStartLoading() {  
 super.onStartLoading();  
  
 forceLoad();  
 }  
  
 @Nullable  
 @Override  
 public String loadInBackground() {  
 return NetworkUtils.*getBookInfo*(mQueryString);  
 }  
}

**NetworkUtils:**

*/\*  
 \* Copyright (C) 2018 Google Inc.  
 \*  
 \* Licensed under the Apache License, Version 2.0 (the "License");  
 \* you may not use this file except in compliance with the License.  
 \* You may obtain a copy of the License at  
 \*  
 \* http://www.apache.org/licenses/LICENSE-2.0  
 \*  
 \* Unless required by applicable law or agreed to in writing, software  
 \* distributed under the License is distributed on an "AS IS" BASIS,  
 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 \* See the License for the specific language governing permissions and  
 \* limitations under the License.  
 \*/*package com.example.android.whowroteitloader;  
  
import android.net.Uri;  
import android.util.Log;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.URL;

*/\*\*  
 \* Utility class for using the Google Book Search API to download book  
 \* information.  
 \*/*public class NetworkUtils {  
  
 private static final String *LOG\_TAG* = NetworkUtils.class.getSimpleName();  
  
 *// Constants for the various components of the Books API request.  
 //  
 // Base endpoint URL for the Books API.* private static final String *BOOK\_BASE\_URL* =  
 "https://www.googleapis.com/books/v1/volumes?";  
 *// Parameter for the search string.* private static final String *QUERY\_PARAM* = "q";  
 *// Parameter that limits search results.* private static final String *MAX\_RESULTS* = "maxResults";  
 *// Parameter to filter by print type.* private static final String *PRINT\_TYPE* = "printType";  
  
  
 */\*\*  
 \* Static method to make the actual query to the Books API.  
 \*  
 \* @param queryString the query string.  
 \* @return the JSON response string from the query.  
 \*/* static String getBookInfo(String queryString) {

*// Set up variables for the try block that need to be closed in the  
 // finally block.* HttpURLConnection urlConnection = null;  
 BufferedReader reader = null;  
 String bookJSONString = null;  
  
 try {  
 *// Build the full query URI, limiting results to 10 items and  
 // printed books.* Uri builtURI = Uri.*parse*(*BOOK\_BASE\_URL*).buildUpon()  
 .appendQueryParameter(*QUERY\_PARAM*, queryString)  
 .appendQueryParameter(*MAX\_RESULTS*, "10")  
 .appendQueryParameter(*PRINT\_TYPE*, "books")  
 .build();  
  
 *// Convert the URI to a URL.* URL requestURL = new URL(builtURI.toString());  
  
 *// Open the network connection.* urlConnection = (HttpURLConnection) requestURL.openConnection();  
 urlConnection.setRequestMethod("GET");  
 urlConnection.connect();  
  
 *// Get the InputStream.* InputStream inputStream = urlConnection.getInputStream();  
  
 *// Create a buffered reader from that input stream.* reader = new BufferedReader(new InputStreamReader(inputStream));  
  
 *// Use a StringBuilder to hold the incoming response.* StringBuilder builder = new StringBuilder();  
  
 String line;  
 while ((line = reader.readLine()) != null) {  
 *// Add the current line to the string.* builder.append(line);  
  
 *// Since this is JSON, adding a newline isn't necessary (it won't  
 // affect parsing) but it does make debugging a \*lot\* easier  
 // if you print out the completed buffer for debugging.* builder.append("\n");  
 }  
  
 if (builder.length() == 0) {  
 *// Stream was empty. Exit without parsing.* return null;  
 }  
  
 bookJSONString = builder.toString();  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 *// Close the connection and the buffered reader.* if (urlConnection != null) {  
 urlConnection.disconnect();  
 }  
 if (reader != null) {  
 try {  
 reader.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 *// Write the final JSON response to the log* Log.*d*(*LOG\_TAG*, bookJSONString);  
  
 return bookJSONString;  
 }  
}

**Activity\_main.xml**

*<?*xml version="1.0" encoding="utf-8"*?>  
<!--  
 Copyright (C) 2018 Google Inc.  
  
 Licensed under the Apache License, Version 2.0 (the "License");  
 you may not use this file except in compliance with the License.  
 You may obtain a copy of the License at  
  
 http://www.apache.org/licenses/LICENSE-2.0  
  
 Unless required by applicable law or agreed to in writing, software  
 distributed under the License is distributed on an "AS IS" BASIS,  
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 See the License for the specific language governing permissions and  
 limitations under the License.  
 -->*<android.support.constraint.ConstraintLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_margin="16dp"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/instructions"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/instructions"  
 android:textAppearance="@style/TextAppearance.AppCompat.Title"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"/>  
  
 <EditText  
 android:id="@+id/bookInput"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:hint="@string/input\_hint"  
 android:inputType="text"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/instructions"/>  
  
 <Button  
 android:id="@+id/searchButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:onClick="searchBooks"  
 android:text="@string/button\_text"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/bookInput"/>  
  
 <TextView  
 android:id="@+id/titleText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:textAppearance="@style/TextAppearance.AppCompat.Headline"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/searchButton"/>

<TextView  
 android:id="@+id/authorText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:textAppearance="@style/TextAppearance.AppCompat.Headline"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/titleText"/>  
  
</android.support.constraint.ConstraintLayout>

**Screen Shots:**

