Networking – צריכת תוכן מהרשת

הנושא הבא די מורכב לכן השיעור נחלק לשלבים: בשלב הראשון נבין את העיקרון, בשלב השני נגרום למנגנון לעבוד ברקע ובשלב השלישי נתחבר ל-API.

באופן כללי, אפליקציות המובייל הן יישום קטן עם הון פונקציונליות.
אחת הדרכים שיישומי המוביל מעבירים פונקציונליות כה רבה במכשיר כל כך קטן היא היכולת למשוך מידע ממקורות רבים ומגוונים.
דרך נפוצה למשיכת מידע היא על ידי שימוש ב- HTTP
אנדרואיד מספק דרך פשוטה וקלת משקל על ידי שימוש במחלקה – **HttpURLConnection**

- שימוש במחלקה זו למינימום 2.3 targets וכאשר נצרך מאפיין HTTP בסיסי.
- עפ"י הנחיות "מפתחי האנדרואיד", אפליקציות חדשות אמורות להשתמש במחלקה זו.

<u>קיימת תבנית כללית לשימוש ב- HttpURLConnection: (תרגום מתוך הספר)</u>

1- הוספת HttpURLConnection חדש על ידי קריאה למתודה URL.openConnection() והזנת התוצאות של שיטה זו ל- HttpURLConnection.
2- הכנת הבקשה, המאפיין העיקרי של הבקשה נמצא ב-URI. כותרת הבקשה עשויה לכלול גם מטאדטא (=נתונים לגבי נתונים) כמו הרשאות, מתודות בקשה (get or post), סוגי תוכן מועדפים וקבצי קוקיס.
3- קריאת התגובה. כותרת התגובה בדרך כלל כוללת מטאדאטא כמו תגובה לתוכן הגוף, לסוג ולאורך, תאריכים מותאמים וקבצי קוקיס. גוף התגובה עשוי להיקרא מתוך הזרם החוזר על ידי השיטה: getInputStream()
4- ניתוק. ברגע שגוף התגובה נקרא, ה- HttpURLConnection צריך להיסגר על ידי קריאה ל disconnect(). שיטה זו משחררת את המשאבים שהוחזקו בחיבור כך שהם נסגרים או משמשים לשימוש חוזר.

דוגמא מצוינת אפשר למצוא בקובץ של אייל בנדר- יש גם הסברים בעברית.
אני מוסיפה כאן דברים שצור הרחיב בכיתה לגבי כל מתודה, לפי שלבי העבודה:

1. URL url = new URL ('http://www.....');
url יודע להחזיר אובייקט, הוא יודע ליצור request ולהחזיק response, הוא גם מחזיק בתוכו את המידע לאן להתחבר.

2. con= url.openConection();
בשלב זה מתבצעת בדיקה האם יש קשר לאתר באינטרנט?
חשוב לזכור שעד לנקודה זו בכלל לא יוצאים לאינטרנט וכל מה שאנו מבצעים זו הכנה.
כאן מתבצע הקשר הראשוני – חיוג לרשת.

3. con.getReesponseCode();
- התקנת שרת אינטרנט מגיע עם רשימה מובנית של טעויות אפשריות- לכל טעות יש מספר מזהה. נשתמש בפונקציה זו כדי לקבל דיווח על טעות אפשרית.
- קוד 200 = הכול תקין! נציב פונקציה זו על if כדי לאתר טעויות, כך:

```
If(con.getResponseCode()!=200)
{
Return;
}
```

כלומר, אם הקוד איננו 200 אז קיימת טעות כלשהיא.
קיימת אפשרות נוספת – שימוש בדגל קיים: HTTP_OK וזה זהה ל-200

4. inputStream- זהו אובייקט שיודע להתחבר לקבוץ ו"לשתות" ממנו את המידע. הוא מגיע אלינו כקוד בינארי (0 , 1). המידע שמתקבל חייב המרה.

outputStream- אובייקט שיודע להתחבר לקובץ ולשלוח אליו מידע.
שני האובייקטים הנ"ל מסייעים לקבל את הקובץ במנות קטנות, בניגוד לעבר שאז הקובץ נשלח בבת אחת ויצר בעיות לעיתים קרובות.
כאן אפשר לוודא שהקובץ עובר באופן תקין – כל מנת קובץ נבדקת.
מנה = שורות או בייטים.

.5 Input Stream in = con.getInputStream();
האובייקט con מחזיק בתוכו מידע.
בפועל, עדיין אין יציאה לאינטרנט וזוהי הכנת תשתית ל"תפיסת" המידע שעתיד להתקבל.

.6 Input Stream_reader = new InputSreamReader(input_stream);
הרידר מבצע המרה – מהקוד הבינארי שהתקבל – זה רק שלב ביניים! עדיין אין כאן סטרינג!
השתנה הפורמט כדי שנוכל לעבוד אתו ולבצע את ההמרה בשלב הבא.

.7 input = new BufferedReader(input_stream_reader);
כאן מתבצעת ההמרה הסופית לסטרינג.

.8 while((line = input.readLine())!=null){
response.append(line+"\n");}
בשלב זה נשתמש בלולאה כדי לפרוק את המידע שהתקבל במנות – שורה אחר שורה. בכל פעם שהוא מגיע ל- \n הוא חותך (=סוף שורה) ומכניס לתוך האובייקט. רק בשלב זה מתבצע הקשר הראשוני בפועל עם האינטרנט.
ה- response הוא כל האתר/ המידע ששאבנו.

- שיטת עבודה זו כוללת המון המרות, וזה נראה ארוך ומייגע וגם מיותר... אבל האמת שזה כפתרון לשיטות הישנות (לפני 2005), אז היו מכינים הכול ורק אז מתחילים למשוך וזה יצר מגבלות של נפח נתונים והצריך חיתוכים כדי שאפשר יהיה למשוך.
כאן בכלל לא משנה גודל הקובץ כי יש קלאסים שיודעים למשוך חתיכות ולטפל בטעויות והמשיכות הן מאוד מאוד קטנות.
פעם המכשיר היה מושבת עד שכל הקובץ ירד...

נשתמש באמולטור 2.2 , כי זה לא יעבוד על אמולטורים מתקדמים.

```
TextView tv = (TextView)findViewById(R.id.textView1);
 tv.setMovementMethod(new ScrollingMovementMethod())
```

הוספת המתודה הזאת הופכת את השכבה להיות נגללת – אחרי השלבים הקודמים קיבלנו קוד של HTML והוא מאוד ארוך, ה- textView לא יכול להכיל אותו ונראה רק את החלק העליון של הקוד, לכן, הוספנו את השיטה הנ"ל, כעת ניתן לגלול את המסך ולראות את כל הקובץ שהורדנו.

ה-**URL** הוא קלאס שיורש מ-URI הוא שולט בכל פתיח הקישורים- imap, content, servis וכדומה הוא קלאס גדול יותר ונרחב.

לא לשכוח-
input.close()
והכי חשוב! להוסיף הרשאת internet בקובץ המניפסט!

עבודת כיתה:
לבחור אתר, להוריד את ה-html שלו אליי.
לא לשכוח! חובה להוסיף הרשאה לאינטרנט

ASYNC TASKS

עד כה, הקוד עבד לפי שלבים קבועים ומסודרים, כל פעולה בזמנה.
עכשיו נראה מצבים שבהם מתרחשות מספר פעולות במקביל ומאחורי הקלעים.
אנדרואיד מאלץ אותנו לאמץ את השיטה הזאת בגלל שגוגל מאפשר 5 שניות בלבד לפעולה או שהאפליקציה תיסגר ותצא טעות – ANR.
AsyncTask- מאפשר לבצע פעולות סנכרון בממשק המשתמש, כולל אלו שאורכות יותר מ-5 שניות.
כולל את כל הכלים שאנו צריכים לביצוע הסנכרון.

שלבי עבודה:

ניצור קלאס חדש – לשים לב ל- extends!

```
public class MyTask extend AsyncTask<Params , Progress , Result>
```

Params- סוג הפרמטרים שיישלחו ל- task upon execution
Progress- סוג ההליך שיתבצע מאחורי הקלעים .
Result- קובע את הקידומת של ההליך שמתבצע.

- נשים לב כי שלושת אלו מתחילים באות גדולה – כלומר, הם קלאסים!

האפליקציה רצה באזור שנקרא uitryd, כאן אנו ניגשים לאזור אחר שנקרא MyTask ובו מתבצעת ההורדה או הסנכרון – מחוץ לאפליקציה, בסביבה נפרדת.

```
{
On PreExecute(){

}
@ovrride
String doInBackgruond(string……params){
תוריד סרט
}
onPostExecuted(String resulte){

}
```

מחוץ ליואייטרייד הוא מייצר טרייד חיצוני ושם הוא מוריד את הסרט- מחוץ לאפליקציה שלי הוא רץ בסביבה נפרדת

באקטיביטי מין- אונקרייט:

```
MyTask task = new MyTask()
Task.execute();
```

לפני הוצאה לפועל יש דגל שעולה ובאון אקטיביטי רואים את הרגע שהוא מוציא לפועל. וגם אחרי שהוא יצא לפועל.
אין אפשרות להגיע למשתנים של אפליקציות אחרות, המידע נמצא במרחב זיכרון אחר. וכשאנו יוצאים מהיואייטרייד זה מרחב זיכרון אחר וההורדה רצה שם, מחוץ למרחב הזיכרון שלנו- המשמעות היא שאפשר לעבור לאקטיביטי אחר והוא יוסיף לרוץ ולהוריד.
האפליקציה רצה ביואייטרייד, ההורדה רצה ב- myTask

<u>העברות של מידע</u>-
אם אני רוצה למשל פונקציה שתביא לי מידע מאתרי אינטרנט שונים בכל פעם

```
protected String doInBackground(String... params)
```

זה מבנה ייחודי שעוד לא נתקלנו בו והמשמעות שלו שהוא יכול לאכול סטרינג או מערך של סטרינג.

```
Public MyTask(Activity act){
```

```
Activity = act;
}
```

נותן את האובייקט
זיהוי פריט מאקטיביטי אחר- קידומת אקטיביטי
נוכל להשתמש בזה רק בפונקציה אונפרהאקסיוקט- כי כאן הוא נמצא במייטסק
לא נוכל להשתמש בזה בזמן ההוצאה לפועל אלא רק לפני

לפני היציאה למשימה- שים פרוגרס בר שיעבוד ל-5 שניות
אחרי שהמשימה הסתיימה כבה אותו.

מי שמחזיק את האובייקט וגם נמצא במרחב הזיכרון של האפליקציה- יכול בעזרתו לפנות לתגיות ה-XML באמצעות find view by id
אך ברגע שהוא נמצא על ה- backgrund- אי אפשר לעשות את הפעולות הללו.

סיכומון-
פרה- באפליקציה
בזמן- מחוץ לאפליקציה.
אחרי- באפליקציה.

עבודת כיתה – לעשות את הדוגמא שצור עשה בכיתה-
בשלב שני להוסיף את מה שעשינו בתחילת השיעור במקום הסליפ.

התרגיל היה לקחת את הפונקציה כמו שהיא והציב אותו במקום onsleep

ב- ui trade גוגל יצאו מגירסא 4 ב"חוק" חדש שאסור שפעולות יארכו יותר מ-5 שניות.

Json & api

התקשרות ל- API חיצוני, התממשקות אליו ומשיכת מידע משם כאובייקט ג'ייסון ופארסינג למידע.

שיעורי בית-
http://www.rottentomatoes.com/
להירשם לאתר- אח"כ נקבל איזשהוא key
ניכנס לדינאמיק דקומונטיישן- הכי חשוב שקיבלנו key בלעדיו לא נוכל נקבל הרשאות ל-API ואז יש גישה להכול- נלך ל- search הוא מבקש 3 פונקציות
נותן ג'ייסון מטורף מלא במידע- צריך לעשות לו פארסינג
ריקווסט url הAPI הוא חוצה פלטפורמות- לא אכפת לו מאיפה באת- התפקיד שלנו לדעת איך לשלוח ואיך לקבל.
אם נשים את השורה הזאת במקום כתובת האתר שעשינו בשיעור הוא יתן את הג'יסון הזה.
סטרינג יוראל= והכתובת.
עכשיו כשלוחצים על הכפתור עולה הג'ייסון.
הגייסון הגיע בבלאגן ואנחנו רוצים לפתוח אותו כמו שצריך-
לכן נשתמש בכלי שנקרא:
ג'ייסון אדיטור אונליין
נעשה העתק הדבק לעורך ג'ייסון- נלחץ ימינה והוא ימיר לנו!

תזכורת- מה זה ג"יסון

מבנה {"מפתח":"ערך","מפתח":"ערך"}
וכמובן ג'ייסון מורכב יותר.
נוצר מערך שיש בתוכו ג'ייסונים- אחרי תגית יש מערך
עד כה תגית- ערך, תגית-ערך. ואז תגית ובמקום ערך יש מערך
וייתכן שבתוך ג'יסון נמצא עוד ג'ייסון

ג'ייסון מחליף את XML

בג'אוה יש אובייקט שנקרא:

```
JSONObject json = new JSONObject(str);
String abc = json.getString('title');
JSONObject = json.getJSONObject('contect');
String name = json.getString('name');
JSONArray = json.getJSONArray('cars');
```

נתחיל את החילוץ עם לולאה:

```
For(int i=0; i<arr.size();i++)
{
If([i].getString('id').equals('3'))
משהו
}
```

httpURLConnection
ועוד

באנדרואיד יצאו קלאסים חדשים שבד"כ מספיקים לצרכים.
זהו קלאס שבד"כ מוודא שזה פורמט UTF8
בודק את תקינות ה- URL