Kullanılanlar

Hpptt connection

AsyncTask

RecyclerView

Fragment

Room

SharedPrefrence

Broadcast Reciver

Notification

JobScheular

1.Kısım İnternet üzerinden bağlantıların alınması

NetworkActivity clası:

```
private InputStream dowlandUrl(String urlString) throws IOException{
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setReadTimeout(10000 /* milliseconds */);
    conn.setConnectTimeout(15000 /* milliseconds */);
    conn.setRequestMethod("GET");
    conn.setDoInput(true);

    conn.connect();

    return conn.getInputStream();
}
```

HttpUrlConnection üzerinden bir bağlantı açıp seçilen url'e göre input stream olarak bağlantıyı alıyoruz.

```
public void loadPage(){
    new DownloadXmlTask().execute(mURL);
}


private class DownloadXmlTask extends AsyncTask<String, Void, String>{

    @Override
    protected String doInBackground(String... urls){
        try{
            if(networkActivityChoice.equals("build")){
                return loadXBuildFeaturesFromNetwork(urls[0]).toString();
            }else {
                return loadXmlFromNetwork(urls[0]).toString();
            }
        }catch (IOException e){
            return "";
        }catch (XmlPullParserException e){
            return "";
        }
    }
```

loadPage() metodu ile DowlandXmlTask metodumu dışarı açıyorum.

Build ve News kaynakları(yani haber içerikleri ve lastbuild date bilgilerini almak için) iki farklı parser çağıran asyncTask metodlarım var .

```
private List<NewsParser.News> loadXmlFromNetwork(String urlString) throws XmlPullParserException, IOException{
    InputStream stream = null;
    NewsParser newsParser = new NewsParser();
    List<NewsParser.News> newsS ;
    try{
        stream = dowlandUrl(urlString);
        newsS = newsParser.parse(stream);
        fetchedNews = newsS;
    } finally {
        if(stream != null){
            stream.close();
        }
    }
    return newsS;
}

private List<LastBuildDateParser.BuildFeatures> loadXBuildFeaturesFromNetwork(String urlString) throws XmlPullP
    InputStream stream = null;
    LastBuildDateParser lastBuildDateParser = new LastBuildDateParser();
    List<LastBuildDateParser.BuildFeatures> buildFeatures;
    try{
        stream = dowlandUrl(urlString);
        buildFeatures = lastBuildDateParser.parse(stream);
        fetchedBuilds =buildFeatures;
    } finally {
        if(stream != null){
            stream.close();
        }
    }
    return buildFeatures;
}
```

```
public interface AsyncResponse {
    void processFinish(String output);
}
```

```
public AsyncResponse delegate = null;
```

Kendi yazdığım interface sayesinde mainactivitye implement edip onpostExecute metodu tarafından tektiklenebilir bir metod oluşturuyorum.

```
    @Override
    protected void onPostExecute(String result){
        //
        if(result != null)
        delegate.processFinish(result);
    }
```

2.Kısım inputstream'in parser tarafından işlemesi:

NOT: dökümanda yer almayan 2 ve 3 kısımlar ,Build özelliklerinin parse edilmesi ve parse edilmesi, benzerdir.

NewsParser.class

```
public List parse(InputStream in) throws XmlPullParserException, IOException {
    //initilize parser and features
    try{
        XmlPullParser parser = Xml.newPullParser();
        parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, state: false);
        parser.setInput(in,  inputEncoding: null);
        parser.nextTag();
        return readFeed(parser);
    }
    finally {
        in.close();
    }
}
```

Parser'in dışarı açılan metodu ve özelliklerinin initiliaze edilmesi

```
//model
public static class News{
        public final String title;
        public final String description;
        public final String link;
        public final String pubDate;
        public final String creator;
        public final String category ;
        public final Boolean  isValidCategory ;
```

Okunan imputların saklanacağı veri modelim

isValidCategory kullanıcı müdehalesi olana kadar true olarak tutuluyor.

```
private List readFeed(XmlPullParser parser) throws XmlPullParserException,IOException{
    List news = new ArrayList();

    parser.require(XmlPullParser.START_TAG, ns,  name: "rss");
    parser.next();
    parser.next();
    while(parser.next() != XmlPullParser.END_TAG){
        if(parser.getEventType() != XmlPullParser.START_TAG){
            continue;
        }
        String name = parser.getName();
        if (name.equals("item")) {
            news.add(readNews(parser));
        } else {
            skip(parser);
        }

    }

    return news;
}
```

‘rss’ tagı ile okumaya başlıyorum rss yapısından dolayı benim işime yaramayan iki alt tag kısmına geçiş için .next() metodunu kullanıuyırum.

END_TAG olmayana kadar her <item> nesnesini readNews() metodu ile pars ediyorum, START_TAG olduğu zamanlarda bir sonraki itemi okumaya başlayabilirim.

```
    private  News readNews(XmlPullParser parser)throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG, ns,  name: "item");
        String title = null;
        String description = null;
        String link = null;
        String pubDate = null;
        String creator = null;
        ArrayList<String> categoryList  = new ArrayList<>();
        String defaultCategory = "default" ;

        while(parser.next() != XmlPullParser.END_TAG){
            if(parser.getEventType() != XmlPullParser.START_TAG){
                continue;
            }
            String name = parser.getName();
            if(name.equals("title")){
                title = readTitle(parser);
            }else if(name.equals("description")){
                description = readDescription(parser);
            }else if(name.equals("link")){
                link = readLink(parser);
            }else if(name.equals("pubDate")){
                pubDate = readPubDate(parser);
            }else if(name.equals("category")){
                categoryList.add(readcategory(parser));
            }else if(name.equals("dc:creator")){
                creator = readCreator(parser);
            }
            else{
                skip(parser);
            }
        }
```

Her item nesnesinin kendi özelliklerine göre parse edileceği metodlarına yönlendirilmesi.

```
        //if  have any category, update default category
        if(categoryList.size() != 0){
            defaultCategory = categoryList.get(0);
        }
        return  new News(title, description,link,pubDate,creator,defaultCategory);
}
```

Rss den gelen birden fazla kategori olması halinde ilk kategoriyi tutuyorum.

```
    private String readTitle(XmlPullParser parser) throws IOException, XmlPullParserException {
        parser.require(XmlPullParser.START_TAG, ns, name: "title");
        String title = readText(parser);
        parser.require(XmlPullParser.END_TAG, ns, name: "title");
        return title;
    }
```

Örnek title'ın parse edilmesi.

```
private String readText(XmlPullParser parser) throws IOException, XmlPullParserException {
    String result = "";
    if (parser.next() == XmlPullParser.TEXT) {
        result = parser.getText();
        parser.nextTag();
    }
    return result;
}
```

Text içerikli metinlerin okunması.

```
private void skip(XmlPullParser parser)throws XmlPullParserException, IOException{
if(parser.getEventType() != XmlPullParser.START_TAG){
    throw new IllegalStateException();
}
int depth = 1;
while (depth != 0){
    switch (parser.next()){
        case XmlPullParser.END_TAG:
            depth--;
            break;
        case XmlPullParser.START_TAG:
            depth++;
            break;
    }
}
```

İstediğim tagler haricindeki kısımları okumamak için skip() metodu

Depth 1 den başlıyıp her START_TAG gördüğünde bir arttırılıp END_TAG durumda bir azaltılarak istemediğim içerikleri okumadan input streamin içindeki bilgiyi tüketmek(atlamak)

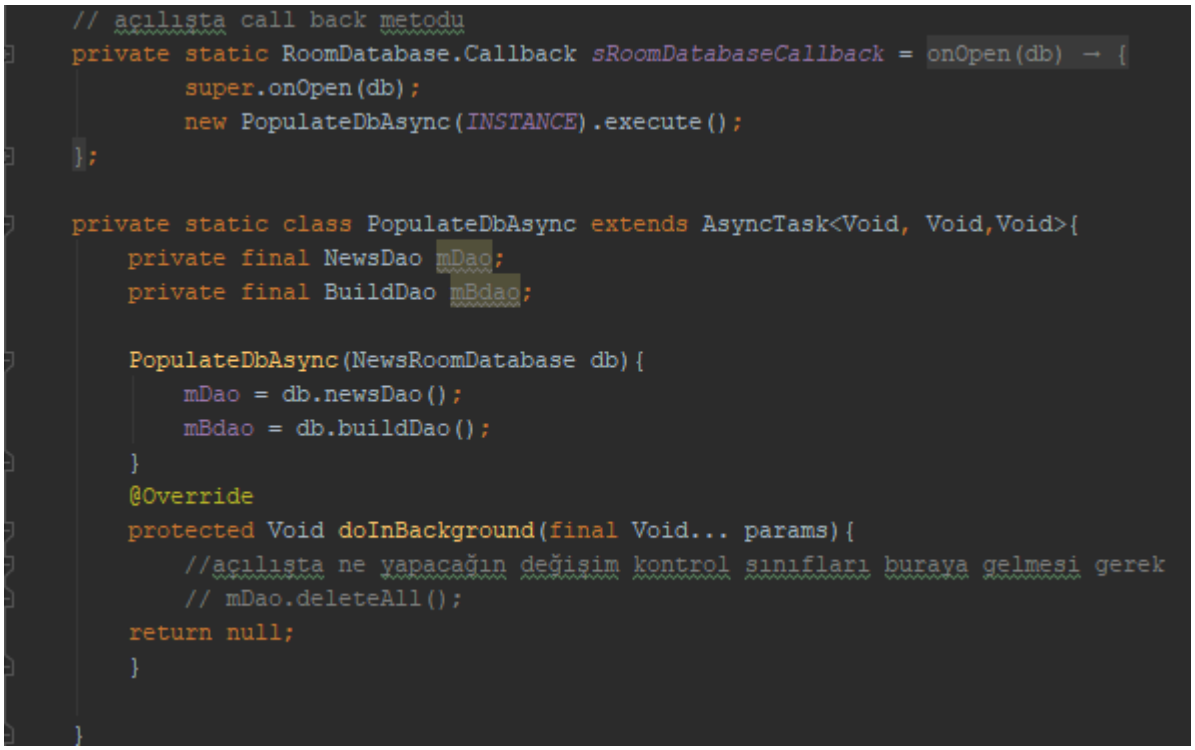
3.Kısım gelen bilgilerin Room ile telefonda saklanması:

3.1 Database oluşturulması:

NewsRoomDatabase clası, singletion bir yapıdadır,

```
@Database(entities = {News.class, Build.class}, version = 4,exportSchema = false)
public abstract class NewsRoomDatabase extends RoomDatabase {

    public abstract NewsDao newsDao();
    private static NewsRoomDatabase INSTANCE;
    public abstract BuildDao buildDao();

    public static NewsRoomDatabase getDatabase(final Context context){
        if(INSTANCE == null){
            synchronized (NewsRoomDatabase.class){
                if(INSTANCE == null){
                    INSTANCE = Room.databaseBuilder(context.getApplicationContext(),NewsRoomDatabase.class,  name: "news_database")
                            .fallbackToDestructiveMigration()
                            .addCallback(sRoomDatabaseCallback) //bu kısmı tekrar editle
                            .build();
                }
            }
        }
        return INSTANCE;
    }
```

Database’in oluşturulması ve sigletion tasarıma uygun birden fazla örnek oluşturulmaması .



```
// açılışta call back metodu
private static RoomDatabase.Callback sRoomDatabaseCallback = onOpen(db) → {
        super.onOpen(db);
        new PopulateDbAsync(INSTANCE).execute();
};

private static class PopulateDbAsync extends AsyncTask<Void, Void,Void>{
    private final NewsDao mDao;
    private final BuildDao mBdao;

    PopulateDbAsync(NewsRoomDatabase db){
        mDao = db.newsDao();
        mBdao = db.buildDao();
    }
    @Override
    protected Void doInBackground(final Void... params){
        //açılışta ne yapacağın değişim kontrol sınıfları buraya gelmesi gerek
        // mDao.deleteAll();
    return null;
    }

}
```

Açılışta çağıralacak metodlar, Data acces object (Dao) yapılarının bağlanması, eğer gerekli ise database populate yapan metod.

3.2 Veri modelinin oluşturulması:

News clası:

```
@Entity(tableName = "news_table")
public class News {


    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "title")
    String title;

    @ColumnInfo(name = "description")
    String description;

    @ColumnInfo(name = "link")
    String link;

    @ColumnInfo(name = "pubDate")
    String pubDate;

    @ColumnInfo(name = "creator")
    String creator;

    @ColumnInfo(name = "category")
    String category ;
```

```
@ColumnInfo(name = "valid_category")
String isValidCategory = "true";

@ColumnInfo(name = "type")
String type;

public News(@NonNull String title, String description, String link, String pubDate, String creator,String category,String type){
    this.title = title;
    this.description = description;
    this.link = link;
    this.pubDate = pubDate;
    this.creator = creator;
    this.category = category;
    this.type = type;
}

@NonNull
public String getTitle() { return title; }

public String getDescription() { return description; }

public String getLink() { return link; }

public String getPubDate() { return pubDate; }

public String getCreator() { return creator; }

public String getCategory() { return category; }

public String isValidCategory() { return isValidCategory; }
```

Tablodaki stunların tanımlanması,modelin constructor ve getter metodları

@CloumInfo ile değişkenin bir stun olduğunu belirliyorum.

@Entity bu clasın tablo olduğunu belirltir.

3.3 Data Access Object oluşturulması:

NewsDao clası, tablo üstünde yapılacak değişikliklerden sorumludur .

```
@Dao
public interface NewsDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insert(News news);

    @Query("DELETE FROM news_table")
    void deleteAll();

    @Query("DELETE FROM news_table WHERE title  = :title")
    void deleteWhereTitle (String title);

    @Query("SELECT * FROM news_table")
    LiveData<List<News>> getAllNews();

    @Query("SELECT * FROM news_table WHERE type = 'Sports' AND valid_category = 'true' ")
    LiveData<List<News>> getAllNewsSport ();

    @Query("SELECT * FROM news_table WHERE type = 'World' AND valid_category= 'true' ")
    LiveData<List<News>> getAllNewsWorld ();

    @Query("SELECT * FROM news_table WHERE type = 'Science' AND valid_category = 'true' ")
    LiveData<List<News>> getAllNewsScience ();

    @Query("SELECT valid_category FROM news_table WHERE title = :title")
    String getCategoryValidnes (String title);

    @Query("UPDATE news_table SET valid_category = 'false' WHERE category =:category")
    void setCategoryFalse ( String category);
```

@Dao bu clasın Dao olduğunu tanımlar.

@Insert ve @Query ile metodlarımı belirlenmesi.

3.4 Repository:

NewsRepository clası:

```
public class NewsRepository {
    private NewsDao mNewsDao;
    private LiveData<List<News>> mAllNews;
    private LiveData<List<String>> allNonValidCategory;
    private LiveData<List<News>> scienceNews;
    private LiveData<List<News>> sportNews;
    private LiveData<List<News>> worldNews;



    public NewsRepository(Application application){
        NewsRoomDatabase db = NewsRoomDatabase.getDatabase(application);
        mNewsDao = db.newsDao();
        mAllNews = mNewsDao.getAllNews();
        scienceNews = mNewsDao.getAllNewsScience();
        sportNews = mNewsDao.getAllNewsSport();
        worldNews = mNewsDao.getAllNewsWorld();
        allNonValidCategory = mNewsDao.getAllNonVaildCategory();
    }
```

Live dataların oluşturulması (data sourcedaki kaynağın izlenmesini sağlayan yapılar)

Class constructor'ı: Dao yapılarının atanması ve database bağlantısının oluşturulması.

```
public LiveData<List<News>> getmAllNews() { return mAllNews; }

public LiveData<List<News>> getScienceNews() { return scienceNews; }
public LiveData<List<News>> getSportNews() { return sportNews; }
public LiveData<List<News>> getWorldNews() { return worldNews; }

public LiveData<List<String>> allNonValidCategory() { return allNonValidCategory;}

public void insert(News...news) { new insertAsyncTask(mNewsDao).execute(news); }

private static class insertAsyncTask extends AsyncTask<News ,Void,Void>{
    private NewsDao mAsyncTaskDao;

    insertAsyncTask(NewsDao dao) { mAsyncTaskDao = dao; }

    @Override
    protected Void doInBackground(final News... params){
        for (News param : params) mAsyncTaskDao.insert(param);
        return null;
    }

}
```

Asenkron olarak Dao üzerinden bilgilerin database'e kaydedilmesi, NewsDao'nun sahip olduğu tüm metodlar için benzer metodlar yazılır.

3.5 ViewModel, Verilere UI üzerinden erişilmesi için kullanılır:

NewsViewModel class'ı, repoda olduğu gibi Dao yapsıının tüm metodlarını soyutlar.

```
public class NewsViewModel extends AndroidViewModel {

    private NewsRepository mRepository;
    private LiveData<List<News>> mAllNews;
    private LiveData<List<String>> allNonValidCategory;
    private LiveData<List<News>> scienceNews;
    private LiveData<List<News>> sportNews;
    private LiveData<List<News>> worldNews;


    public NewsViewModel(Application application){
        super(application);
        mRepository = new NewsRepository(application);
        mAllNews = mRepository.getmAllNews();
        scienceNews = mRepository.getScienceNews();
        sportNews = mRepository.getSportNews();
        worldNews = mRepository.getWorldNews();
        allNonValidCategory = mRepository.allNonValidCategory();
    }

    public LiveData<List<News>> getmAllNews() { return mAllNews; }
    public LiveData<List<String>> getAllNonValidCategory() { return allNonValidCategory;

    public LiveData<List<News>> getScienceNews() { return scienceNews; }
    public LiveData<List<News>> getSportNews() { return sportNews; }
    public LiveData<List<News>> getWorldNews() { return worldNews; }

    public void insert(News news) { mRepository.insert(news); }

    public void deleteAll() { mRepository.deleteAll(); }

    public void deleteWhereTitle(String title) { mRepository.deleteWhereTitle(title); }
```

4 MainActivityde verilerin arayüze eklenmesi:

4.1 RecyclerView ve yapılarının oluşturulması:

Activity_main.xml layoutuna view olarak eklenir daha sonra bilgileri tutup viewe bağlaması için custom adapter yazılır, NewsListAdapter classı.

NewsListAdapter:

ViewHolder'ın bindViewHolder yapısı ile RcyclerView view yapısının sahip olduğu viewlere (satırlar) gösterilecek haberleri bağlar, ViewHolder classı bilgilerin tam olarak nasıl bağlanacağına karar verir.

4.2 Hangi haber tipinin yükleneceğinin edileceğinin seçilmesi

```
public static String typeCoice = "world"; // default news source type
Spinner typeSpinner;
```

```
//when get some chance news type spinner chance adapter for that type and notfy
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
    typeCoice = typeSpinner.getSelectedItem().toString();
    switch (typeCoice){
        case "World":
            recyclerView.setAdapter(adapterWorld);
            adapterWorld.notifyDataSetChanged();
            break;
        case "Science":
            recyclerView.setAdapter(adapterScience);
            adapterScience.notifyDataSetChanged();
            break;
        case "Sports":
            recyclerView.setAdapter(adapterSport);
            adapterSport.notifyDataSetChanged();
            break;
    }
}
```

UI da olan spinner yapısının seçimine göre type değiştirmek.

4.3 Network activtiy den gelen bilgileri Room'a geçirmek:

```
public void fetchNewsAndRoom(){
    //chooser set by activity selected news type
    networkActivity.chooseNetworkActivity(typeCoice);
    //start loading  xml
    networkActivity.loadPage();
}
```

Network ile istediğim tipteki bilgileri alıyorum,

```
    //
    networkActivity.delegate = this;
```

PostExeCute metodunu kullanmak için delegate nesnesi üretiyorum.

```
@Override
public void processFinish(String output){
    // onPostExecute(result)
    NewsParser.News fetchedNews;
    News mTempNews;
        //generate entity object for item list size
    if(networkActivity.getFetchedNews() != null)
        for(int i = 0; i< networkActivity.getFetchedNews().size(); i++) {
            //get ith element
            fetchedNews = networkActivity.getFetchedNews().get(i);
            mTempNews  = new News(fetchedNews.title,
                    fetchedNews.description,
                    fetchedNews.link,
                    fetchedNews.pubDate,
                    fetchedNews.creator,
                    fetchedNews.category,
                    typeCoice);
            //insert my entity
            mNewsViewModel.insert(mTempNews);
        }
    LastBuildDateParser.BuildFeatures fetchedBuilds;
    Build mBuildTemp;

    if(networkActivity.getFetchedBuilds() != null){

        for(int i = 0; i< networkActivity.getFetchedBuilds().size(); i++){
            fetchedBuilds = networkActivity.getFetchedBuilds().get(i);
            mBuildTemp = new Build(fetchedBuilds.title,
                    fetchedBuilds.lastBuildDate);
            mBuildViewModel.insert(mBuildTemp);
        }
    }
```

postExecute çalıştığında yani bilgilerim tamamen geldiğinde kontrollerden sonra Rooma kaydediyorum.

4.4 Kullanıcın istediği kaynağı görüntülemek için LiveData ve Rcycler view bağlantısı

```
recyclerView = findViewById(R.id.recyclerLayout);
adapterScience = new NewsListAdapter( context: this);
adapterSport = new NewsListAdapter( context: this);
adapterWorld = new NewsListAdapter( context: this);
recyclerView.setAdapter(adapterWorld);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));


//3 difrent adapter for 3 difrent data source
mNewsViewModel = ViewModelProviders.of( activity: this).get(NewsViewModel.class);
mBuildViewModel = ViewModelProviders.of( activity: this).get(BuildViewModel.class);

mNewsViewModel.getScienceNews().observe( owner: this, (Observer) (news) → {
        adapterScience.setNews(news);
});

mNewsViewModel.getSportNews().observe( owner: this, (Observer) (news) → {
        adapterSport.setNews(news);
});

mNewsViewModel.getWorldNews().observe( owner: this, (Observer) (news) → {
        adapterWorld.setNews(news);
});
```

3 farklı adapter oluşturup Live data yapılarına observer bağlarayarak olası bir data değişiminde rcycler view yapsını tekrar güncelliyorum (adapteri tekrar bağlıyorum).

5 Sağa ve Sola kaydırma aksiyonları:

```
ItemTouchHelper helper = new ItemTouchHelper(
        new ItemTouchHelper.SimpleCallback( dragDirs: 0, swipeDirs: ItemTouchHelper.LEFT |ItemTouchHelper.RIGHT) {
            @Override
            public boolean onMove(@NonNull RecyclerView recyclerView, @NonNull RecyclerView.ViewHolder viewHolder, @NonNull R
                return false;
            }
            //when swipe actions happend check whic adapter and do start another actvity for that data
            @Override
            public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
                int position = viewHolder.getAdapterPosition();
                News mNews = null;
                switch (typeCoice){
                    case "World":
                        mNews = adapterWorld.getNewsAtPosition(position);
                        break;
                    case "Science":
                        mNews = adapterScience.getNewsAtPosition(position);
                        break;
                    case "Sports":
                        mNews = adapterSport.getNewsAtPosition(position);
                        break;
                }
                switch (direction){
                    case ItemTouchHelper.LEFT:
                        readNews(mNews);
                        break;
                    case ItemTouchHelper.RIGHT:
                        mNewsViewModel.setCategoryFalse(mNews.getCategory());
                        break;
                }
                recyclerView.getAdapter().notifyItemChanged(position);
            }
```

İtemTouchHelper yardımı ile sağa kaydırıldığında itemi silme (vaildCategorinin false yapılması ve live data tarafından göz ardı edilmesi), sola kaydırma aksiyonunda ise DescriptionActivity ile haberin ayrıntılı okunmasını sağlıyorum.

6 BroadcastReciver ile wifi bağlantısını kontrol etmek

Wifi bağlı olduğu zaman kullanıcıya haberleri güncellemesini hatırlatan bir toast mesajı yayınlar.

CustomReciver classı: app tarafından kayıt edilen (filitreler) broadcast mesajları yakalandığında ne yapılacağını belirler

```
@Override
public void onReceive(Context context, Intent intent) {
    NetworkInfo activeNetwork = null;
    String intentAction = intent.getAction();
    String  networkState = "" ;
    String  networkType = "";
    String toastMessage = "unknown intent action";

    //delay for connecting wifi
    try {
        Thread.sleep( millis: 1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    ConnectivityManager cm = (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);
    if(cm != null) {
         activeNetwork = cm.getActiveNetworkInfo();
    }
    if(activeNetwork != null){
        networkState = activeNetwork.getState().toString();
        networkType = activeNetwork.getTypeName();
    }

    if (intentAction != null) {
        if (intentAction.equals(WifiManager.NETWORK_STATE_CHANGED_ACTION)) {
            if (networkState.equals("CONNECTED") && networkType.equals("WIFI")) {
                toastMessage = "wifi is online, right time to refresh news! ";
                Toast.makeText(context, toastMessage, Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Broadcast mesajı geldiğinde connectivity menager ile aktif bağlantının bilgilerini activeNetwork'e alıyoruz, eğer bağlantı var ise bilgilerini tip ve state olarak alıyoruz, buradaki Thread.sleep kodu wifi bağlantılarının yapısından dolayı bağlantının geç kurulması durumundan kaynaklı bir bekleme süresi ekliyoruz.

Bağlantı ve tip bilgilerimiz uyuşuyor ise toast mesajı yayınlıyoruz.

```
    //for reciving wifi state broadcast,  if some connectivity actions happend reciver can call customReciver
    IntentFilter filter = new IntentFilter();
    filter.addAction(WifiManager.NETWORK_STATE_CHANGED_ACTION);
    this.registerReceiver(mReceiver,filter);
```

MainActivity OnCreate metodunda reciveri kaydetmek

```
@Override
public void onDestroy(){
    if(mReceiver != null)
        this.unregisterReceiver(mReceiver);
    super.onDestroy();
}
```

Gereksiz kaynak tüketmemek için onDestroy metodunda reciverin kaydını siliyoruz.

7 SettingsActivity, fragment ve shared preferences:

Buradaki amacımız fragment yapısı olarak settings activityi kodlamak, notification alıp almama seçeneklerini sharedpref ile kalıcı olarak saklamak.

7.1 Fragment oluşturmak

Res>xml klasörü altında preferences.xml olarak bir fragment yaratıyoruz

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

    <SwitchPreferenceCompat
        android:defaultValue="false"
        android:key="WiFi_Notification_switch"
        android:summary="Turn of notification"
        android:title="Notification Settings"
        />
</PreferenceScreen>
```

UI elemanı olan bir Switch oluşturuyoruz.

SettingsFragment classı içerisinde onCreatPreferences metodunda kullanılacak xml'i bağlıyoruz

```
@Override
public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
    setPreferencesFromResource(R.xml.preferences, rootKey);
}
```

7.2 SettingsActivity de fragmenti kullanmak

```
public static String KEY_PREF_WİFİ_NOTFY = "WiFi_Notification_switch";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    getSupportFragmentManager().beginTransaction()
            .replace(android.R.id.content, new SettingsFragment())
            .commit();

}
```

```
KEY_PREF_WİFİ_NOTFY
```

İle main activity üzerinde seçimin bundle keyini belirmiş oluyoruz

7.4 Main Activity de seçimin kaydedilmesi:

```
//for didnt change user settings
PreferenceManager.setDefaultValues( context: this, R.xml.preferences,  readAgain: false);

//saving user pref
 sharedPreferences = PreferenceManager.getDefaultSharedPreferences( context: this);
 switchPref = sharedPreferences.getBoolean(SettingsActivity.KEY_PREF_WİFİ_NOTFY,  defValue: false);
```

8 JobScheuler ile notification gönderilmesi

```
private void createNotificationChannel() {
    // Define notification manager object.
    mNotifyManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

    // Notification channels are only available in OREO and higher.
    // So, add a check on SDK version.
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {

        // Create the NotificationChannel with all the parameters.
        NotificationChannel notificationChannel = new NotificationChannel
                (PRIMARY_CHANNEL_ID,
                        name: "Job Service notification",
                        NotificationManager.IMPORTANCE_HIGH);
        notificationChannel.enableLights(true);
        notificationChannel.setLightColor(Color.RED);
        notificationChannel.enableVibration(true);
        notificationChannel.setDescription ("You have new news to read");

        mNotifyManager.createNotificationChannel(notificationChannel);
    }
}
```

Notification channel yaratılması,

SistemService üstünden notyficationManagerin oluşturulması, android sürümü kontrolü sonrasında notification channel oluşturulması.

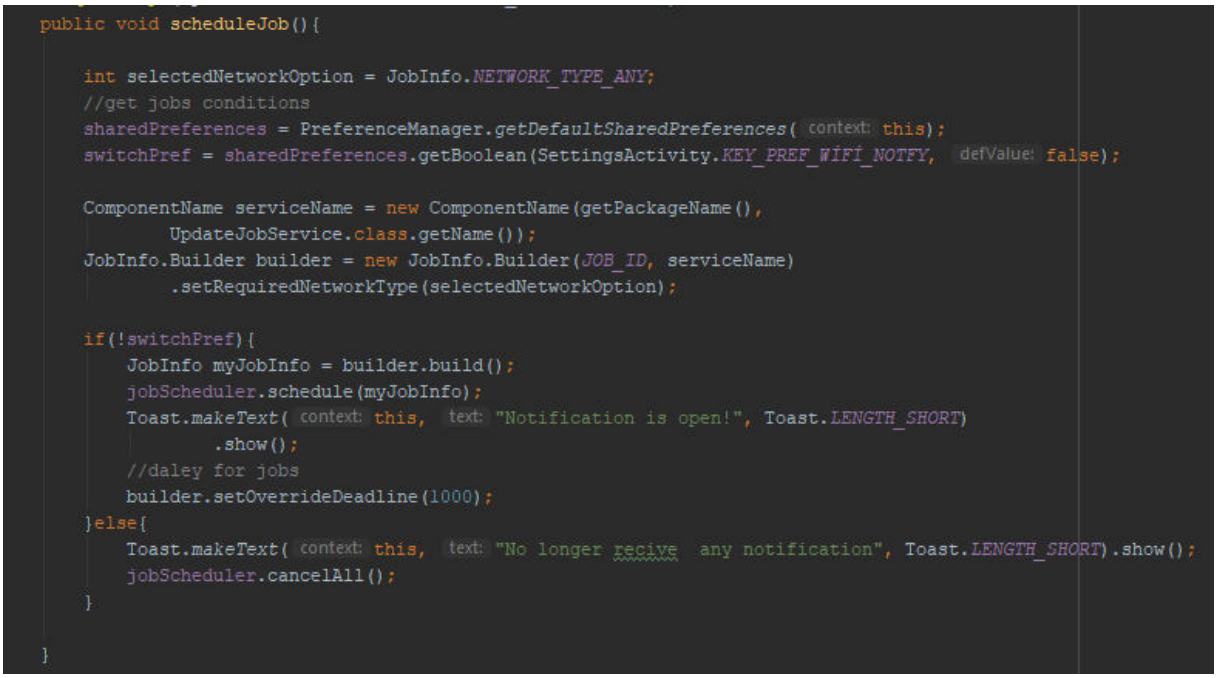
```java
NotificationManager mNotifyManager;
// Notification channel ID.
private static final String PRIMARY_CHANNEL_ID =
        "primary_notification_channel";

@Override
public boolean onStartJob(JobParameters jobParameters) {

    //Create the notification channel
    createNotificationChannel();

    //Set up the notification content intent to launch the app when clicked
    PendingIntent contentPendingIntent = PendingIntent.getActivity
            ( context: this,  requestCode: 0, new Intent( packageContext: this, MainActivity.class),
                    PendingIntent.FLAG_UPDATE_CURRENT);

    NotificationCompat.Builder builder = new NotificationCompat.Builder
            ( context: this, PRIMARY_CHANNEL_ID)
            .setContentTitle("Hey!  ")
            .setContentText("who knows what happened in the world !")
            .setContentIntent(contentPendingIntent)
            .setSmallIcon(R.drawable.ic_job_running)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setDefaults(NotificationCompat.DEFAULT_ALL)
            .setAutoCancel(true);

    mNotifyManager.notify( id: 0, builder.build());
    return false;
}
```

OnStartJob

Notificationchannel oluşturduktan sonra, pending intent oluşturuyoruz, bu sistemin bizi ayağa kaldırmasını sağlıyor.

```java
@Override
public boolean onStopJob(JobParameters params) { return true; }
```

onStopJob, eğer parametre olarak false gönderir ise jobscheulerin görevi tamamlanmış olur ve sistem bir daha jobu tekrar kurmaz.



```
public void scheduleJob(){

    int selectedNetworkOption = JobInfo.NETWORK_TYPE_ANY;
    //get jobs conditions
    sharedPreferences = PreferenceManager.getDefaultSharedPreferences( context: this);
    switchPref = sharedPreferences.getBoolean(SettingsActivity.KEY_PREF_WİFİ_NOTFY,  defValue: false);

    ComponentName serviceName = new ComponentName(getPackageName(),
            UpdateJobService.class.getName());
    JobInfo.Builder builder = new JobInfo.Builder(JOB_ID, serviceName)
            .setRequiredNetworkType(selectedNetworkOption);

    if(!switchPref){
        JobInfo myJobInfo = builder.build();
        jobScheduler.schedule(myJobInfo);
        Toast.makeText( context: this,  text: "Notification is open!", Toast.LENGTH_SHORT)
                .show();
        //daley for jobs
        builder.setOverrideDeadline(1000);
    }else{
        Toast.makeText( context: this,  text: "No longer recive  any notification", Toast.LENGTH_SHORT).show();
        jobScheduler.cancelAll();
    }

}
```

MainACtivity içerisinde jobu kurmak,

sharedPref üstünden switchPref alıyoruz, kullanıcı tarafından seçilmiş notificationların atılıp atılmadığını kotrol eden settings bilgisi,

Builder ile jobu kuruyoruz burada en az bir condition olması OS tarafında bize zorunlu tutuluyor, internete bağlı olduğumuzu kontrol ediyoruz.