[Урок 2 LiveData](https://startandroid.ru/ru/blog/27-course/architecture-components/525-urok-2-livedata.html)

В этом уроке рассмотрим возможность совместного использования RxJava и Room. Как получать данные в Flowable, Single и Maybe.

Если вы еще не знакомы с RxJava, то посмотрите соответствующий [курс](https://startandroid.ru/ru/19-course/rxjava/435-urok-1.html).

Подключение к проекту

В build.gradle модуля добавляйте dependencies

|  |  |
| --- | --- |
| 1 | implementation "android.arch.persistence.room:rxjava2:1.0.0" |

Flowable

В Dao указываем для метода выходной тип Flowable

|  |  |
| --- | --- |
| 1  2 | @Query("SELECT \* FROM employee")  Flowable<List<Employee>> getAll(); |

В коде подписываемся и получаем данные

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | db.employeeDao().getAll()         .observeOn(AndroidSchedulers.mainThread())         .subscribe(new Consumer<List<Employee>>() {             @Override             public void accept(List<Employee> employees) throws Exception {                 // ...             }         }); |

subscribeOn в случае с Flowable не нужен. Запрос в базу будет выполнен не в UI потоке. А вот, чтобы результат пришел в UI поток, используем observeOn

Теперь при любом изменении данных в базе, мы будем получать свежие данные в методе accept. И нам не надо будет каждый раз их снова запрашивать самим.

Если при запросе нескольких записей, вместо Flowable<List<Employee>> использовать Flowable<Employee>:

|  |  |
| --- | --- |
| 1  2 | @Query("SELECT \* FROM employee")  Flowable<Employee> getAll(); |

то мы получим только первую запись из всего результата

Если же мы составляем запрос для получения только одной записи, то Flowable<Employee> вполне подойдет. Давайте рассмотрим этот пример подробнее.

Метод в Dao

|  |  |
| --- | --- |
| 1  2 | @Query("SELECT \* FROM employee WHERE id = :id")  Flowable<Employee> getById(long id); |

В коде подписываемся на Flowable

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | db.employeeDao().getById(1)         .observeOn(AndroidSchedulers.mainThread())         .subscribe(new Consumer<Employee>() {             @Override             public void accept(Employee employee) throws Exception {                 // ...             }         }); |

Итак, мы запрашиваем из базы запись по id. И тут возможны варианты.

Если запись есть в базе, то она придет в accept сразу же после подписки. И при каждом последующем обновлении этой записи в базе данных, она также будет приходить в accept.

Если записи нет, то сразу после подписки ничего не придет. А вот если она позже появится, то она придет в accept.

У вышеописанного примера есть минус. Если записи нет в базе, то Flowable вообще ничего нам не пришлет. Т.е. это будет выглядеть так, как будто он все еще выполняет запрос.

Это можно исправить следующим образом:

|  |  |
| --- | --- |
| 1  2 | @Query("SELECT \* FROM employee WHERE id = :id")  Flowable<List<Employee>> getById(long id); |

Хоть мы и ожидаем всего одну запись, но используем не Flowable<Employee>, а Flowable<List<Employee>>. И если записи нет, то мы хотя бы получим пустой лист вместо полной тишины.

Single

Рассмотрим тот же пример с запросом одной записи, но с использованием Single. Напомню, что в Single может прийти только один onNext, либо OnError. После этого Single считается завершенным.

Метод в Dao

|  |  |
| --- | --- |
| 1  2 | @Query("SELECT \* FROM employee WHERE id = :id")  Single<Employee> getById(long id); |

В коде подписываемся

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | db.employeeDao().getById(1)         .subscribeOn(Schedulers.io())         .observeOn(AndroidSchedulers.mainThread())         .subscribe(new DisposableSingleObserver<Employee>() {             @Override             public void onSuccess(Employee employee) {                 // ...             }               @Override             public void onError(Throwable e) {                 // ...             }         }); |

В отличие от Flowable, с Single необходимо использовать onSubscribe, чтобы задать поток для выполнения запроса. Иначе в onError придет ошибка: *java.lang.IllegalStateException: Cannot access database on the main thread since it may potentially lock the UI for a long period of time*.

Снова рассматриваем варианты наличия требуемой записи в базе.

Если такая запись в базе есть, то она придет в onSuccess. После этого Single будет считаться завершенным и при последующих обновлениях этой записи ничего приходить уже не будет.

Если такой записи в базе нет, то мы в onError получим ошибку: android.arch.persistence.room.EmptyResultSetException: Query returned empty result set: SELECT \* FROM employee WHERE id = ?.   
После этого Single будет считаться завершенным, и даже, если такая запись появится в базе, то нам ничего приходить уже не будет.

Maybe

Рассмотрим тот же пример с запросом одной записи, но с использованием Maybe. Напомню, что в Maybe может прийти либо один onNext, либо onComplete, либо OnError. После этого Maybe считается завершенным.

Метод в Dao

|  |  |
| --- | --- |
| 1  2 | @Query("SELECT \* FROM employee WHERE id = :id")  Maybe<Employee> getById(long id); |

В коде подписываемся

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | db.employeeDao().getById(1)         .subscribeOn(Schedulers.io())         .observeOn(AndroidSchedulers.mainThread())         .subscribe(new DisposableMaybeObserver<Employee>() {             @Override             public void onSuccess(Employee employee) {                 // ...             }               @Override             public void onError(Throwable e) {                 // ...             }               @Override             public void onComplete() {                 // ...             }         }); |

С Maybe также необходимо использовать onSubscribe, чтобы задать поток для выполнения запроса.

Рассматриваем варианты наличия требуемой записи в базе.

Если такая запись в базе есть, то она придет в onSuccess. После этого Maybe будет считаться завершенным и при последующих обновлениях этой записи ничего приходить уже не будет.

Если такой записи в базе нет, то мы получим onComplete. После этого Maybe будет считаться завершенным, и даже, если такая запись появится в базе, то нам ничего приходить уже не будет.

В каком случае что лучше использовать?

Flowable подходит, если вы запрашиваете данные и далее планируете автоматически получать их обновления.

Single и Maybe подходят для однократного получения данных. Разница между ними в том, что Single логичнее использовать, если запись должна быть в базе. Если ее нет, вам придет ошибка. А Maybe допускает, что записи может и не быть.