V. Draichuk, Master student
A. Panishev, D.E., Prof., research advisor
V. Shadura, senior teacher, language advisor
Zhytomyr State Technological University

THE SIGNIFICANCE OF REACTIVE PROGRAMMING USE FOR THE DEVELOPMENT OF ANDROID OS APPLICATIONS

Android is one of the most relevant mobile operating system. This operating system was developed by Google and is based on Linux. The basic element of the operating system is the Dalvik virtual machine. It is necessary to recognize that the future of PC – is in a portable devices, tablets, e-books, netbooks and smartphones, and all these are Android based.

In May 2010, Google achieved 100 thousand activations per day. In December 2010, there were 300 thousand activations. In May, at the Google I/O conference announced the statistics, according to which about 400 thousand new devices on Android platform were activated every day. In July 2011, Andy Rubin (Andy Rubin), the Vice president of Google, who is responsible for the development of Android platform, announced the overcoming of new limits - 500,000 activations per day with increasing distribution of platform in the 4.4% per week. This statistics includes only information about the first registration of new devices.

In July, 2011, there were already sold more than 100 million Android-devices which had been developed by 36 professionals, their distributed networks reached 215 operators. The productivity of Google Play has overcome a mark of 200 thousand applications. The results of Google Play are the installation of about 4.5 billion of application copies. 84% of smartphones were sold in the third quarter of 2014, running on Android OS. Therefore, the application development for Android today is extremely important.

While working with Android one can often see how all the functional code in the methods is based in lifecycle activity/fragment. In fact, this approach has some reasons to exist - "lifecycle methods" are just the stages of system components and specially designed filling of their code. In addition, UI framework described by xml-files already gives us the basic separation of logics and interface. But it is offen difficult to use such approach effectively and this division is not always possible. It ultimately results in the developing all the codes in onCreate, which adversely affects the transparency of code and makes almost impossible its modification and support.

RxJava is a new technology that is now one of the hottest topics of discussion in the Android-programmers community. The reactive programming is a programming paradigm, based on data flow and distribution changes. This means that programming languages should be able to express static or dynamic data streams easily, and implement the execution model which will automatically send changes via data flow. Resilience, sensitivity, focus on events and scalability are the main principles of reactive programming. If one follows them, it will make the work with code easier. Reactive programming is the development based on asynchronous data streams.It can seem not to be new. The typical mouse actions are asynchronous so it is nothing new about working

with such kinds of actions. One can create data streams from all entities desired; by not just the mouse cursor and movements.The flow can inhance anything. These may be the variables, the information of the user types, the properties, data structures, etc. For example, the Twitter feed will emit a stream of data in the same way as the mouse actions such as movement or click. You can listen to the flow and react to it accordingly.

Study the programing features is important because the application of this technology makes it easier to develop code and handle errors. This technology provides the extended filter possibilities and helps to make processing results of methods calls easier.

Some perspectives of using RxJava in Android apps:

– Usage of RxJava simplifies the multithread RESTful API calls;

– Usage of RxAndroid optimizes the behavior of basic Android UI components;

– Simplified work with threads from the main application.

Using RxJava technology provides many benefits in developing and supporting apps, first of all the results of all transactions are always predictable. We know about all the errors and potential problem areas that may arise in the code develpoment and how to deal them.

The principle of sensitivity in action is as follows. The database connection or server is maintained due to the timeout, the call will attempt to recover an error. A caching executes the parallel processing result. The orientation related the events is based on the process of execution the request, we will always have the responce from events, successful or unsuccessful completion of the request, the event completion, etc. The code becomes easily expandable and requires obtain almost no changes. If we need to make a bug handling or maintaining the stack errors, that can be easily processed by components of RxJava.

The most appropriate cases to use RxJava are:

– UI actions as mouse movements, button clicks;

– The actions related WebSocket API;

– Events such as changing the features, registration procedure, and so on.

It is not recommended to use RxJava only to iterate collections, it is much better to use the regular iterators.

Rx is based on Observer template. The idea of Rx is in the absence of information about the sequence that is valuable or is over. But one is able to control over when starting and stoping taking values. The basic building blocks of code are reactive Observables and Observers. Observable is a source of data and Observer –is a user. Generation data through Observable always occurs at one and the same course of action: Observable gives a certain amount of data and exits - either successfully or with error. Each Observers, signed on Observable, has a method called Subscriber.onNext() for each item of data flow, after which someone can be called Observer.onComplete() or

Observer.onError(). This is very similar to the regular pattern of "observer", but there is one important distinction. Observables often begin to generate data before somebody is subscribed to them.

You can use operators over Observable, such as flatmap, filter, zip, merge, cast, etc. Operators can be used in between Observers and Observables for manipulations with data. In RxJava there are many operators, so it would be better to focus on some of them.

- Create is to create Observable create from nothing by calling observer methods;
- From is to convert any other object or data structure in Observable;
- Map is to transform the elements emitted by Observable by calling the application function to each element;
- Timer is to create Observable, which emits one element after the delay;
- Start is to create Observable, which emits the function;
- Filter is to select only those elements of Observable, which pass the predicate test;

Reactive applications are more resistant to the bugs and unexpected errors, usage of this technology makes the code clearer and more flexible. Many routine work can be translated into a library that gets the work done better than Android-base components. This allows focusing on the implementing the things that really should be developed.

Rx works perfectly with highloaded frontend apps. But the potential of the technology does not relate only the client side, it works great with the server side and databases. In fact, RxJava is a key component in server-side API of Netflix. Rx is not a framework, not limited to one specific type of application or language. This is actually a paradigm that can be used when developinging any event driven software.