Indian Journal of Science and Technology, Vol 8(19), DOI: 10.17485/ijst/2015/v8i19/75919, August 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Application Study on Android Application Prototyping Method using App Inventor

Hwansoo Kang[1], Jinhyung Cho[1] and Heechern Kim[2*]

[1]School of Computer Engineering, Dongyang Mirae University, Seoul, 152-714, Korea; hskang,cjh@dongyang.ac.kr
[2]Department of Computer Science, Korea National Open University, Seoul, 110-791, Korea; hckim@knou.ac.kr

Abstract

App Inventor developed by MIT is a visual programming development environment with drag-and-drop interface which aims to help design and implement mobile apps with full functionality on Android operating system. App Inventor's intuitive blocks programming allow mobile app developers to focus on design and programming logic instead of language syntax. Apps development processes are composed of five phrases, which are discovery, wire-framing, prototyping, implementation, and deployment. In software design process, generally software architecture is divided into components called modules. We analyze that a module is composed of related blocks of App Inventor. With applying App Inventor as an Android application prototyping method, it allows android mobile application to be implemented much faster. In this study, we show the development process using App Inventor as a prototyping method is more efficient compared to using Android SDK only.

Keywords: Android SDK, App Inventor, Development Process, Design Method, Mobile Application, Prototyping

1. Introduction

As the wide and fast-growing smart phones require various broadband mobile services, mobile applications should be developed rapidly and enormously to meet rapidly changing market conditions. App Inventor is a visual blocks programming tool and web-based. It has two major elements, the Component Designer and the Blocks Editor. App Inventor allows inexperienced users with little or no programming knowledge to develop mobile apps easily[1,2]. Users can design an app's interface and non-visible components, and integrate them using the Component Designer. With the Blocks Editor, users can specify an apps' behavior and set how the app acts under certain conditions[3]. The Blocks Editor supports graphical guidance for selecting, composing, and understanding program structures, which can help beginners reduce most common programming errors[4]. App Inventor is reinforcing fundamentals of programming, learning new concepts, teamwork and building mobile apps[5]. In mobile apps development, a prototype is a simulating model of a mobile application, usually built for demonstration purposes or as part of the development process. A prototype is early version of software built to test and try some newly proposed design. Prototyping of apps is very important for rapid development of mobile apps. A prototyping tool should be easy for designers to use in the conceptual app design phase. It should be set and available in users' general environment with little difficulties[6]. There are several prototyping methods such as paper based prototyping, blended prototyping and tool based prototyping[7]. In this paper, we are interested in tool based prototyping.

2. Android App Development Process

Android, which is built on top of the Linux kernel, is a mobile operating system. Android app development processes by which new apps are created for the android operating system[8]. The first step in developing an app is to come up with an idea, which is discovery. The second step is wire-framing. This is the stage where you toss

*Author for correspondence

out ideas and see what works. Effective wire-framing allows you to work through your apps' navigation. Wire-framing makes your ideas tangible and easier to evaluate. Wire-framing is a structural representation of screen skeletons for intended layout of an app with the form of simple line-sketches. In Wire-framing stage, developers make a structure design for your app with your preferences. It includes aspects of the feasibility, specifications, platform, and features. This stage helps you verify that the Android app development progresses harmoniously with your requirements. Next step is prototyping, which is gathering detailed requirements and devising screens user interface, and then develop a prototyping app simulating target app. Next step is developing and testing a target app, called implementation. Some of the other features are taken into account including passing information between activities, SQ Lite Database, usage intention, navigation, and list of views in this Android app development process. In order to keep Quality Assurance of App, regular checks are carried out to ensure the application's functionality during the whole period of the Android app development. The app testing process ensures that the app being developed is of high quality with few errors. Final step is deployment, which is to release and publish mobile apps. App deployment is an essential task for preparing for publication and distribution. A release version of the application is created and prepared for deployment to Android-based devices. The release version can be available through a single or multiple distribution channels.

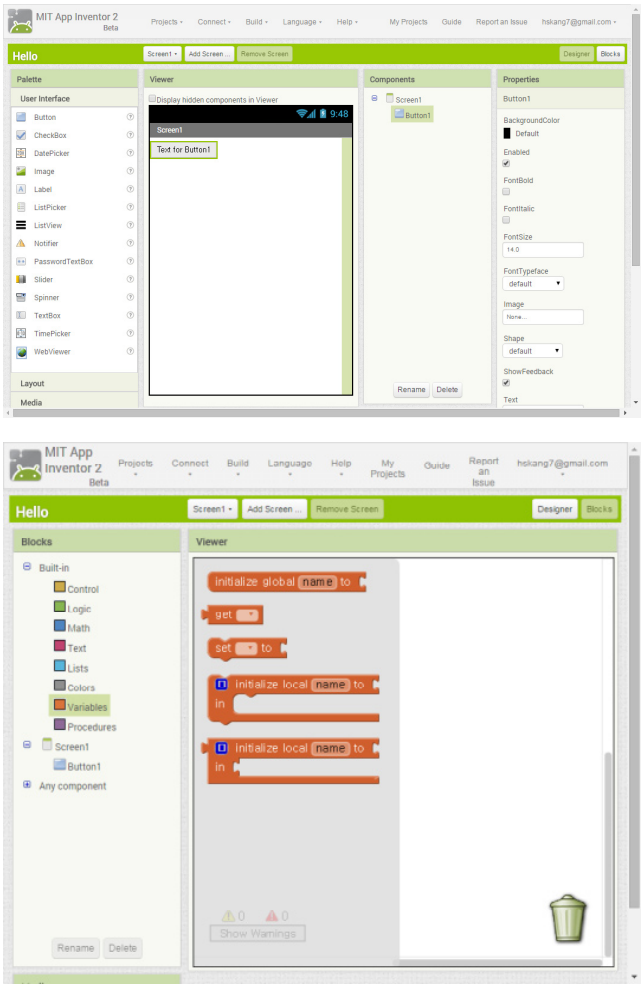

Figure 1. App Inventor elements: Component Designer and Blocks Editor.

3. App Inventor for Prototyping App

We show a sample app for showing that App Inventor is used efficiently as a prototyping tool in the app development processes.

3.1 App Inventor

App Inventor is a web-based developing environment with two main elements, the Component Designer and the Blocks Editor shown in Figure 1. App Inventor makes it easy for users to develop an Android app, although users are lacking in programming experience. With the Component Designer to create an app on Android devices, users can specify the app's visible (e.g., menus or buttons) and non-visible components (e.g., web connections or sounds) and then integrate them. The Component Designer is a user interface design tool with a WYSIWYG interface. Users can drag and drop visual objects into a view, set and change component properties like size and color for each object, and specify the app's general look.

With the Blocks Editor, users can specify an apps' behavior and set how the app acts under certain conditions. The core strength of App Inventor is to provide visual programming interface. It is possible to create full functioning mobile apps by arranging visible and non-visible components and behavior logic blocks with drag-and-drop editing. The blocks language provides easy-to-use programming interface. The decrease of typing rate considerably reduces chances of syntax errors for beginners, the blocks give visual hints to simplify the development task, and only some blocks are held in their positions to minimize the possibility of errors[9–13].

3.2 Target App

Target app project is shown in Figure 2. Target app is showing a message displayed in various text and background colors composed of red, green and blue color value of between 0 and 255.

3.3 Prototyping Target App

We develop a proto type of a target app with App Inventor. App Inventor has two core modules, the Component Designer and the Blocks Editor. The Component Designer designs the app's user interface by arranging visual and non-visual components. Figure 3 shows Designer screen of target app project.

3.4 Blocks Programming

Next, let's program the components using blocks. With the Blocks Editor, you can program the behavior of your app by putting blocks together with intention. Upper blocks of Figure 4 are the blocks that screen will initialize edit text components (tBoxR, tBoxG, and tBoxB) color to red, green, blue and value to 0 when screen1 is opened. Also lower blocks show that when button btnCngLb is clicked, label text of lbDislay is set to tBox Input text.

Figure 5 shows that When btnT Color button is clicked, the color is created as the text color of lbDisplay label and the message in lbDisplay is displayed that color.

The check Color procedure of Figure 6 gets an input value and checks its limits of between 0 and 255, because this value is a red, green or blue color value of 8 bit. If it's less than 0 or not a number at all, the return value will be

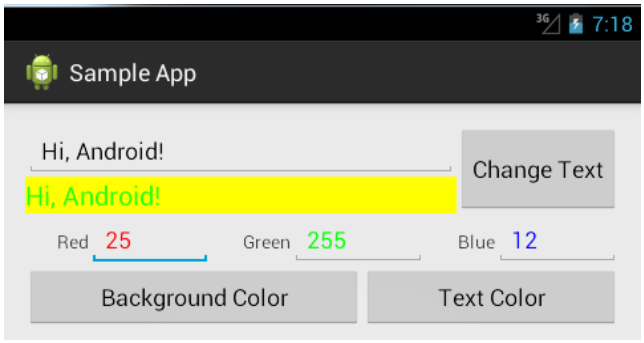

Figure 2. Target app's screen.

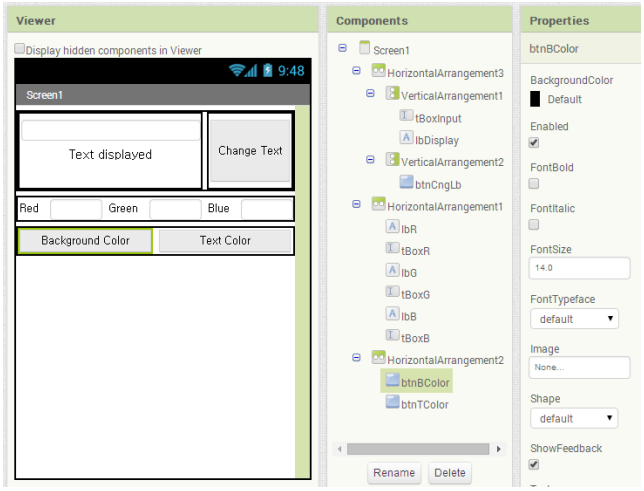

Figure 3. Designer screen of target app.

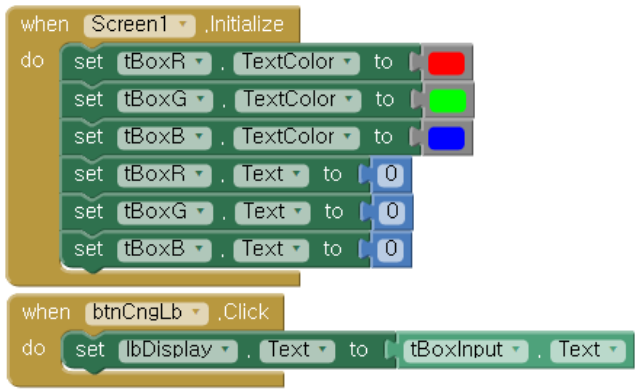

Figure 4. Event handing.

Figure 5. Blocks for setting color.

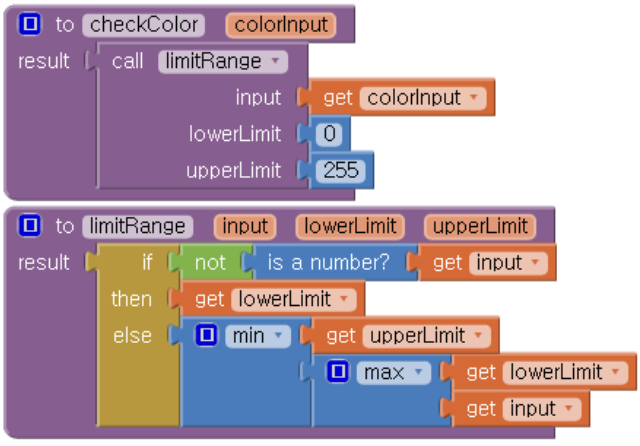

Figure 6. Blocks for procedure.

0. If it's greater than 255, the result will be 255. The check Color procedure calls the limit Range procedure. The limit Range procedure is a procedure having an input, a lower limit and an upper limit of parameters, and checks the limits of input within that range, and then returns and then returns a value of between the lower limit 0 and the upper limit 255.

The limit Range procedure is interested. If the input of the procedure is not a number, the result is the lower limit 0. If the input of the procedure is a number, the procedure returns a final number between 0 and 255. To restrict the range, first of all the procedure take the maximum of the input and the lower limit 0 and then returns the minimum of the result of the maximum and the upper limit 255.

3.5 Easy to Implement Java Source of App

App Inventor's block can be translated easily into app java source. Therefore, with prototyping target app with App Inventor, we can develop a target app rapidly. For example, blocks for when Screen1 is initialized can be translated into app java source of following Figure 7.

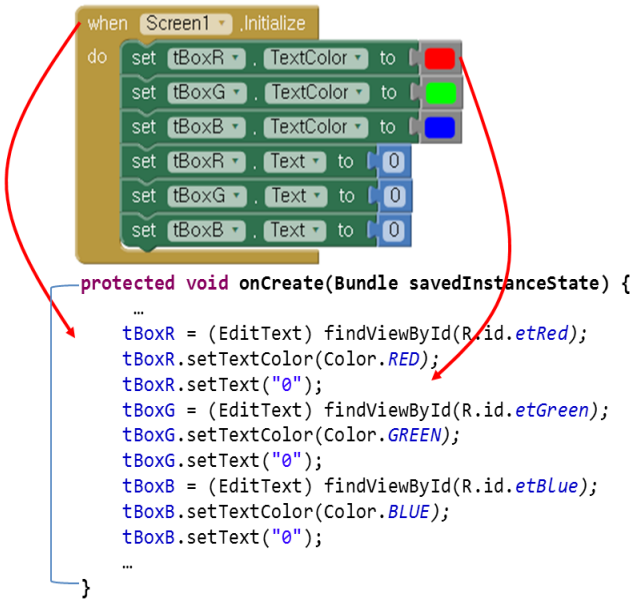

Figure 7. Blocks and java source.

4. Prototyping Significance

A significant function of prototyping is to simulate a target app quickly and easily. The other function is how to help other processes of app development processes in actual development processes. That is, prototyping process is meaningless by itself. It should have connectivity to other processes. As shown in the following Table 1, discovery, wire-framing, implementation and deployment process are related with prototype process closely. As shown in the chapter 2, Android application prototyping method using App Inventor is effective on connectivity with other process of app development processes. Especially Designer of App Inventor helps to wire-framing process. Block coding of App Inventor is very easy, effective in implementation process as well.

Table 1. Prototyping Connectivity with the other app development processes

Step	Process	Connectivity of Prototyping
1	✧ Discovery to come up with an idea	A prototyping tool that simulates a target app easily is effective for good ideas.
2	✧ Wire-framing providing a structural design of screen skeletons at the layout of an app	As wire-framing simulates a target app, a prototype tool having easy screens design is very effective.
3	✧ Prototyping develop a prototyping app simulating target app	A prototype tool should simulate a target app easily and quickly.
4	✧ Implementation developing and testing a target app	Developing a prototyping app should help to develop a target app with original development environment.
5	✧ Deployment essential tasks for preparing for publication and distribution	Prototyping is recommended to help to release and distribute a target app.

5. Conclusion

The process of app development is not so different from the traditional software development process. However, if we develop Android apps more quickly with a project team having developers who deficient in Android SDK and Java programming experience, it will greatly contribute to improving apps development productivity. The App Inventor is used by many departments relating computer science and engineering in college and many students interested in smart apps in the school. Everyone insufficient to programming knowledge can create Android apps by using App Inventor.

You can design Android app on a web page, put a few of logic blocks together on the same page, and test Android app on an emulator or on your phone at the same time.

App Inventor is the advantage of component visual programming, where one can drag and drop visual components, and then give programmable behaviors to logic blocks to develop mobile apps easy. Therefore, App Inventor using for app prototyping is highly efficient because App Inventor is easy to use and intuitive for app programming and App Inventor is effective on connectivity with other process of app development processes.

6. Acknowledgment

This research has been supported by 2014 Academic Research Project funded by Dongyang Mirae University in the Republic of Korea.

7. References

1. MIT App Inventor home page, MIT Center for Mobile Learning. 2014. Available from: http://appinventor.mit.edu
2. MIT App Inventor tutorials page, MIT Center for Mobile Learning. 2014. Available from: http://appinventor.mit.edu/explore/tutorials.html
3. Turbak F, Okerlund J. A preliminary analysis of app inventor blocks programs. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC '13); 2013; San Jose CA.
4. Pokress SC, Veiga JJD. MIT App inventor: Enabling personal mobile computing; PRoMoTo 2013 Proceedings; 2013.
5. Soares A, Martin NL. Teaching non-beginner programmers with app. inventor: survey results and implications. Proceedings of the Information Systems Educators Conference; 2014.
6. Bahr B. Rapid creation of sketch-based,native Android prototypes with Blended Prototyping. PID-MAD 2013 (In Conjunction with Mobile HCI '13); 2013.
7. Jorgensen AP, Collard M. Prototyping iPhone apps: realistic experiences on the device. Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI '10); ACM; 2010. p. 687–90.
8. Eom, Lee. Human-centered software, development methodology in mobile computing environment: agent-supported agile approach. EURASIP Journal on Wireless Communications and Networking. 2013; 2013:111.
9. Hsu Y–C, Rice K, Dawley L. Empowering Educators with Google's Android App Inventor: An Online Workshop in Mobile App Design. British Journal of Educational Technology. 2012; 43(1): E1–5.
10. Wolber D. App Inventor and Real-World Motivation. Proceedings of the 42nd ACM technical symposium on Computer science education (SIGCSE '11); 2011. p. 601–6.
11. Gestwicki P, Ahmad K. App Inventor for Android with Studio-Based Learning. Journal of Computing Sciences in Colleges archive. 2011; 27(1):55–63.
12. Gestwicki P, Ahmad K. Studio-based learning and app inventor for android in an introductory CS course for non-majors. Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13); 2013. p. 287–92.
13. Gestwicki P. App Inventor for Android with Studio-Based Learning. 2014. Available from: https://sites.google.com/site/appinventorsbl/home