5/30/2014
GeoPost

# Release Candidate Code Review Write-Up

We did our code review on MainActivity.java. Because this class contains the main logic for the application, we felt that it was very important that the code was cleanly written in order to make it easy to maintain in the future, that the code gracefully handled any error conditions that arise, and that the comments were clear and concise so that future maintainers would have an idea of how the app works. We had a member of the database team, Andrew, do the review of the MainActivity code.

We learned several interesting lessons from our code review. The first is that commenting and documenting code as you go is a difficult task. It takes a lot of dedication and determination to make sure that you keep comments up-to-date when you write and modify code, especially when you have high commenting standards like we do and you have a rapid development cycle. Andrew found many comments in MainActivity that didn’t conform to our Javadoc standard, so we had to spend some significant time cleaning those up. A second thing is that sticking to style conventions is difficult when there are many different developers. Everyone has their own slightly different style, so getting a consistent style throughout the codebase requires a lot of work. However, being consistent with the style makes the code easier to read and maintain overall, which was evident once we made the changes Andrew suggested. Another lesson we learned is that handling various error cases, especially when there are many services being used, takes a lot of thought and consideration. Andrew found several error cases that didn’t have clean handling code because the developers either did not consider the problems or had difficulty determining what the appropriate response would be. These were all lessons that applied to our code base in general, not just MainActivity, so it was great that we learned them, as they helped us clean up code in other classes such as DBQuery and DBStore.

There were also several changes that we made specifically for MainActivity.java. Perhaps the largest modification we made was in how we determine what pins to show when the user selects a different filter from the dropdown menu in the upper right corner of the app. Before, we had four different boolean fields that were all manipulated together when a new option was chosen. This also required us to consult the different fields when choosing which pins to display. The code review spurred us to create a single SortingOption enum field to replace the four boolean fields. This greatly reduced the length and complexity of our code to choose which pins to display, and we think it will provide more flexible extension of filtering options if we decide to add more in the future. A second important modification we made was removing old interfaces and code from the class. There were several methods and chunks of code that we had left from the early stages of development but did not actually use anymore. By noticing that some of the interfaces we had overlapped, we were able to get rid of an interface implementation declaration and discard several methods that were no longer needed. We think this problem is especially obvious for MainActivity because it uses so many different resources. As a very long class that contains most of the main application logic, it was hard for the developers to see the redundancy as they wrote the code. Having a fresh set of eyes take a look allowed us to clean that up.

The code review of MainActivity.java resulted in us making a significant number of changes to the code, both at a high level and a low level. This refactoring has left our code in a more consistent, easier-to-read state, and has left us more confident in the quality of the code.