Mobile Application Store Product API Results Overview

Date: 2013-10-08
Author: David Killeffer <rayden7@gmail.com>
Reviewer(s): Catherine Bieber <catherine.bieber@gmail.com>

Comments from Peer Design Review

I submitted a request to the TF's to get a design review but unfortunately I missed the original deadline, so the TFs were only able to give my design doc a quick read rather than have a detailed review (I will make sure to give adequate time for my design review in the next assignment).

Here were Cat's comments on my first design document:

> You could simplify the class diagram some by leaving out the standard getters. If you want to convey that there should be a getter but no setter you could try to note "read only" on the properties, or you could just leave that for the class dictionary, which has more granular detail. You indicate two composition relationships pointing at ContentType. This makes me unsure whether it's actually Content or ContentSearch that is responsible for the lifecycle of a ContentType. I notice just "Content" is in italics. I would put a note like <<Abstract>> above the class name just to be clear.
>
> For the paper, I skimmed it and these are a few things I noticed.
>
> The Requirements section seems to be a minimally edited paste of the requirements document. The section is tended to be a shorter summary rather than having full details.
>
> When you discuss use cases, the diagram is mostly taken from the assignment. Consider focusing in on the two highlighted ones and instead breaking them down into more detail, as this design document is meant to focus in on that functionality in particular.
>
> In the class dictionary, all the properties and methods should have descriptions, even if they're just one short sentence. As with the diagram, completely plain-vanilla getters and setters do not need to be included here. They are assumed, and you only need to include them if they are doing anything interesting or out of the ordinary.
>
> This is a purely aesthetic thing but -- it's very helpful to the graders if you structure more detailed diagrams as tall and thin so they consume a page, rather than the sort of "landscape" orientation. That usually leads to very small print and needing to open up the diagram separately. However as long as you provide the separate image so we can view it larger that is fine.
>
> Disclaimer: This was a quick skim and not a full grading, so there may be points I've missed, but on the whole it does otherwise look good so far.
>
> Let us know if you have additional questions.
> -Cat

After receiving Cat's feedback, I tried to revise my class diagram, I also implemented an Importer and SearchEngine classes to handle the import of content and querying for content, and I also refactored a lot of how my code worked and tried to make my implementation more "DRY" and follow the "KISS" principle as much as possible.

I also added required, discrete properties for the Application, Ringtone, and Wallpaper classes, since without having distinct, unique differences between the three types of content items, I felt that we shouldn't have bothered with different classes anyway (so I gave each class what I felt were applicable, distinct required properties).

For me, I felt some confusion over what the TFs and professor were looking for in our design document submissions, but after a first draft, perusing the online discussion forums extensively, and refactoring my code, I feel that the end result of my design document is fairly effective. I don't know that the design document in and of itself made the implementation easier, but I did find that making a class diagram certainly did make the implementation easier (at least at the beginning to create the classes, establish some of the baseline relationships between classes, etc.).

As with all software development though, I found the best thing was to be constantly iterating on the design and the implementation, as well as to be extending the original test cases by adding more devices, content items, and queries to suss out issues and resolve bugs.

Changes from Original Design & Requirements

- Allows partial language code searches to match languages that have the criteria (for example, a content search for languages with code "fr" will match content items that have "fr_ca", French Canadian, as their language code).
- To further emphasize the differences in content item types:
  - Wallpaper has added two properties:
    - pixelWidth
    - pixelHeight
  - Ringtone has added property:
    - durationInSeconds
- To emphasize the separation of concerns, Content items handle the validation of their own types.
- Since Content items are added to collections in the product catalog, to maintain that the ProductAPI only contains a single copy of each item (Device, Country, Application, Ringtone, Wallpaper), each of these types overrides "equals" and "hashCode" to ensure that the Set collections don't contain duplicates
- This implementation makes use of two Apache Commons JARs to facilitate more easily overriding the "equals" and "hashCode" methods in the content item classes, and also to simplify content searches (the ProductAPI's searchContent() method makes use of CollectionUtils to compare the intersection of sets).

- Content items (Application, Wallpaper, and Ringtone) and Device and Country classes override the toString() method to simplify debugging, and to more easily observe all the properties of those objects when querying
- Content item classes (Application, Wallpaper, and Ringtone) each have a static method to validate a given content item of its own type; this keeps the responsibility of determining what constitutes a valid "X" with the class that defines it