

Mobile Application Store Product API Design Document

Date: 2013-09-28

Author: David Killeffer <rayden7@gmail.com>

Reviewer(s):

Introduction

This document defines the design for the Mobile Application Store Product API, which is a core component of the Mobile Application Store.

Overview

The Product API Service component of the Mobile Application Store is used to allow end-Consumers and Administrators the ability to interact with the products that are stored in the Mobile Application Store. The Product API Service is critical to the overall success of the Mobile Application Store, since the vast majority of revenue generated by the store will be through Consumer interactions with the product catalog, and the Service is the primary broker for this interaction.

Consumers are able to search for content across all categories (application, ringtone, and wallpaper) through a rich search interface. Consumer content searches may query for content across nearly every aspect of a content item, and present the full list of matching content items back to the consumer.

Administrators may likewise search for content similarly to Consumers, but they also have the ability to manage content by adding, editing, or deleting individual content items. Application developers may also manage content similarly to Administrators.

Requirements

This section defines the requirements for the Product API Service system component. Given that future sprints will incorporate authentication and authorization (Authentication Service) and that the Product API Service will need to interact with the Authentication Service, the design accounts for the notion of *protecting restricted interface* use cases, and will use a mock authentication token for now (for example, see requirements #1 and #4).

The Product API should support the following functions:

1. Support a method for adding Countries.
 - a. Countries have the following required properties:
 - i. 2-letter country code

- ii. country name
 - iii. export status, either OPEN or CLOSED
 - b. Adding countries is a restricted interface. ***See protecting restricted interface below (#4).***
- 2. Support a method for adding Devices.
 - a. Devices have the following required properties:
 - i. Device ID (string) (REQUIRED)
 - ii. Device Name (string) (REQUIRED)
 - iii. Manufacturer name (string) (REQUIRED)
 - b. Adding Devices is a restricted interface. ***See protecting restricted interface below (#4).***
- 3. Support a method to add content to the Product Catalog.
 - a. When content is created perform content validation. Validate that required fields are populated, including collections requiring 1 or more elements. ***See protecting restricted interface below (#4).***
 - b. Content has the following features:
 - i. Content name (string) (REQUIRED)
 - ii. Content description (string) (REQUIRED)
 - iii. Author's Name (string) (REQUIRED)
 - iv. Content rating (from 0 to 5 where 5 is best)
 - v. 0 or more Categories of type (string)
 - vi. 1 or more countries of type Country (e.g., where the content can be downloaded to) (REQUIRED)
 - vii. 1 or more devices of type Device that the content is compatible with (REQUIRED)
 - viii. A price for the content (float) (units are BitCoins)
 - ix. 1 or more supported 5-letter language codes (string) (REQUIRED)
 - x. 1 image URL (REQUIRED)
 - xi. Content type value, one of APPLICATION, RINGTONE, or WALLPAPER (ContentType) (REQUIRED)
 - c. Content is an interface concept with the following concrete specializations:
 - i. Application
 - ii. RingTone
 - iii. Wallpaper
 - d. An Application has an additional feature:
 - i. A filesize in bytes that indicates the size of the application (integer) (REQUIRED)
 - e. A Wallpaper has two additional features:
 - i. A pixel width (integer) (REQUIRED)
 - ii. A pixel height (integer) (REQUIRED)
 - f. A RingTone has an additional feature:
 - i. A duration in seconds (float) (REQUIRED)
- 4. Restricted interfaces require an additional access token (GUID) parameter. The validation of the access token and permission check will be deferred to a future sprint

when the Authentication Service is built, but all restricted methods should support accepting the access token as an additional string parameter.

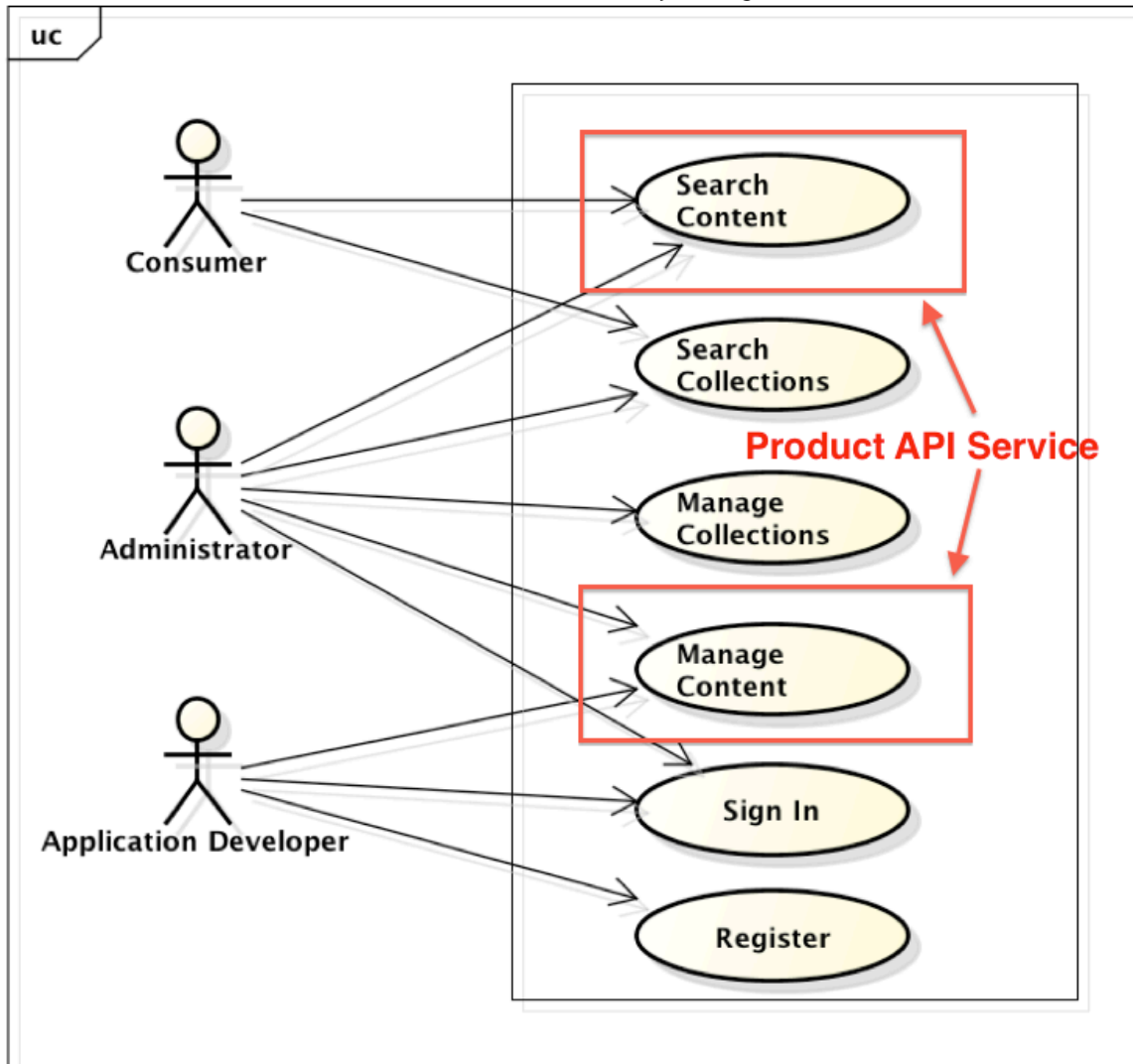
5. Allow consumers to search for and obtain content details.
 - a. This is a non-restricted interface
 - b. Only content compatible with the user's supplied search criteria should be returned
 - c. Search criteria should include:
 - i. Optional list of content categories
 - ii. Optional text search (to match name and description)
 - iii. Optional minimum rating (0-5)
 - iv. Optional maximum price in BitCoins, where 0 is free
 - v. Optional list of language codes
 - vi. Optional country code
 - vii. Optional Device ID
 - viii. Optional list of content types (i.e., Application, RingTone, WallPaper)
 - d. Results should return a list of all matching Content objects

Use Cases

The following use case diagram shows the major functional tasks that each type of user of the Mobile Application Store can perform; the use cases surrounded with red boxes denote those that the Product API Service is responsible for. In future sprint iterations the other use cases will be built out separately from the Product API Service.

This design supports the following use cases:

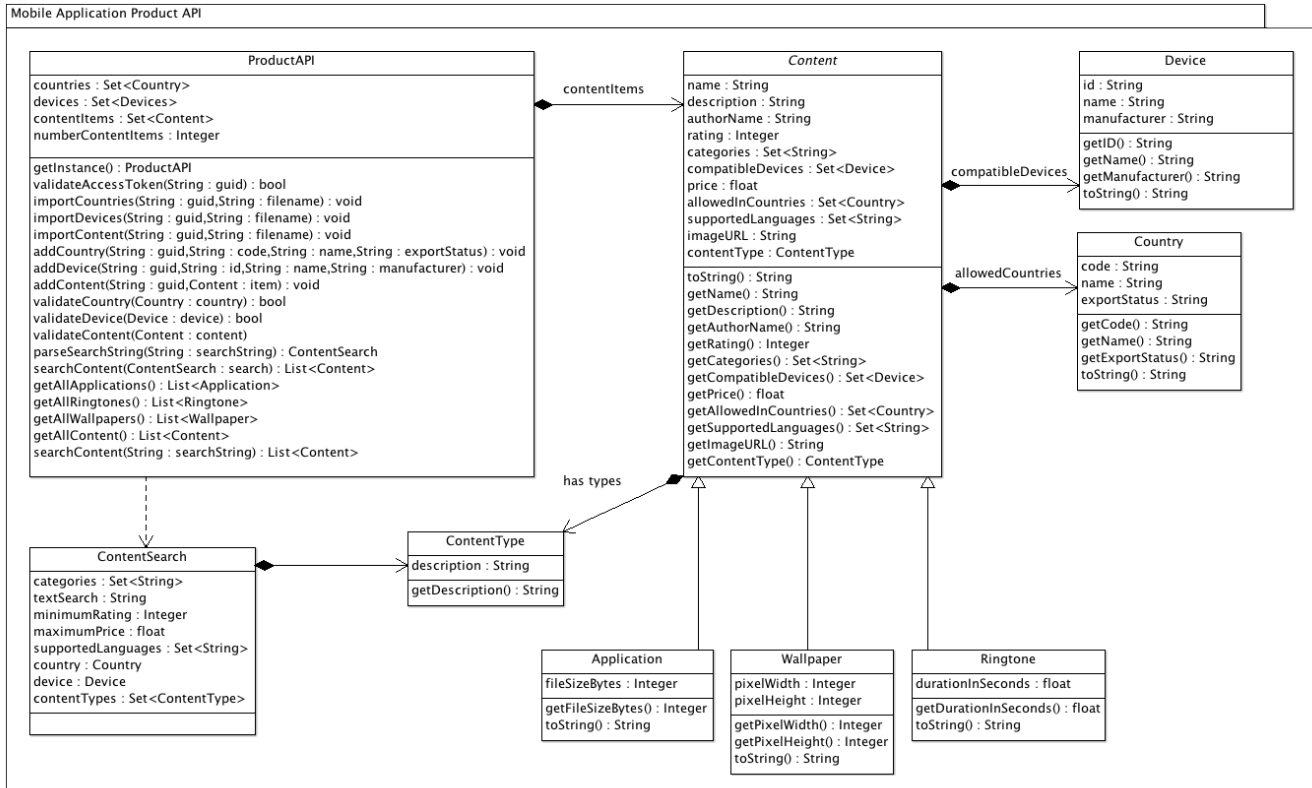
- Search Content (may be performed by Consumers or Administrators)
- Manage Content (may be performed by either Administrators or Application Developers)



Class Diagram

The following class diagram defines the classes defined in this design.

Mobile Application Store Product API Design Document
CSCI E-97 Assignment 2
David Killeffer <rayden7@gmail.com>



Class Dictionary

This section specifies the class dictionary for the Product API. The classes should be defined within the package “cscie97.asn2.ecommerce.product”.

ProductAPI

The ProductAPI is responsible for the public interface for interacting with the product catalog, for end-consumers, application developers, and administrations. The ProductAPI is a singleton and follows the Singleton design pattern. The contentItems property contains references to the entire product catalog of content items.

Methods

Method Name	Signature	Description
getInstance	() : ProductAPI	This method returns a reference to the single static instance of ProductAPI.
validateAccessToken	(String : guid) : bool	If the guid passed is a valid authentication token, returns true; false otherwise. Currently the guids passed to this method will be mocked and may be any value; this method will currently always return true until the Authentication API is built in a future sprint.

importCountries	(String:guid, String:filename) : void	
importDevices	(String:guid, String:filename) : void	
importContent	(String:guid, String:filename) : void	
addCountry	(String:guid, String:code, String:name, String:exportStatus) : void	
addDevice	(String:guid, String:id, String:name, String:manufacturer) : void	
addContent	(String:guid, Content:item) : void	
validateCountry	(Country : country) : bool	
validateDevice	(Device : device) : bool	
validateContent	(Content : content) : bool	
parseSearchString	(String : searchString) : ContentSearch	
searchContent	(String : searchString) : List<Content>	
searchContent	(ContentSearch : search) : List<Content>	
getAllApplications	() : List<Application>	
getAllRingtones	() : List<Ringtone>	
getAllWallpapers	() : List<Wallpaper>	
getAllContent	() : List<Content>	

Properties

Property Name	Type	Description
numberContentItems	Integer	Convenience method that gets the number of content items currently in the contentItems association.

Associations

Association Name	Type	Description
Countries	Set<Country>	Private association for maintaining the current set of Country instances.
Devices	Set<Device>	Private association for maintaining the current set of Devices.
contentItems	Set<Content>	Private association for maintaining the current set of Content items.

ContentSearch

The ContentSearch is used to capture the whole set of potential search parameters (rather than have a search method that takes a very long parameter set. Although not used yet, this object could potentially be used later to allow customers to “save searches” for later use.

Properties

Property Name	Type	Description
categories	Set<String>	
textSearch	String	
minimumRating	Integer	
maximumPrice	float	
supportedLanguages	Set<String>	
country	Country	
device	Device	
contentTypes	Set<ContentType>	

ContentType

This is a container enum type that simply holds the references to the various types of content that is supported; currently this holds APPLICATION, RINGTONE, and WALLPAPER (although in future sprints ebook and other types might be added).

Properties

Property Name	Type	Description
description	String	Contains the content type name; can be one of APPLICATION, RINGTON, WALLPAPER

Content

This abstract class is the primary superclass for all main content type classes (such as Application, Wallpaper, Rington).

Methods

Method Name	Signature	Description
toString	() : String	Returns a string representation of the content item.
getName	() : String	
getDescription	() : String	

getAuthorName	() : String	
getRating	() : Integer	
getCategories	() : Set<String>	
getCompatibleDevices	() : Set<Device>	
getPrice	() : float	
getAllowedInCountries	() : Set<Country>	
getSupportedLanguages	() : Set<String>	
getImageURL	() : String	
getContentType	() : ContentType	

Properties

Property Name	Type	Description
name	String	
description	String	
authorName	String	
rating	Integer	
categories	Set<String>	
price	float	
supportedLanguages	Set<String>	
imageUrl	String	

Associations

Association Name	Type	Description
compatibleDevices	Set<Device>	
allowedInCountries	Set<Country>	
contentType	ContentType	

Device

Represents a device.

Methods

Method Name	Signature	Description
getID	() : String	
getName	() : String	

getManufacturer	() : String	
toString	() : String	

Properties

Property Name	Type	Description
id	String	
name	String	
manufacturer	String	

Country

Represents a country.

Methods

Method Name	Signature	Description
getCode	() : String	
getName	() : String	
getExportStatus	() : String	
toString	() : String	

Properties

Property Name	Type	Description
code	String	
name	String	
exportStatus	String	

Application

Inherits from Content.

Methods

Method Name	Signature	Description
getFileSizeBytes	() : Integer	
toString	() : String	

Properties

Property Name	Type	Description
filesizeBytes	Integer	

Wallpaper

Inherits from Content.

Methods

Method Name	Signature	Description
getPixelWidth	() : Integer	
getPixelHeight	() : Integer	
toString	() : String	

Properties

Property Name	Type	Description
pixelWidth	Integer	
pixelHeight	Integer	

Ringtone

Inherits from Content.

Methods

Method Name	Signature	Description
getDurationInSeconds	() : float	
toString	() : String	

Properties

Property Name	Type	Description
durationInSeconds	float	

Content, Country, Manufacturer, and Device Instance Management

Due to the in-memory nature of this implementation, to optimize memory usage there should only be one instance for each unique Country, Manufacturer, Device, and Content item (where “Content” item is any concrete instance of an Application, Ringtone, or Wallpaper). This follows the Flyweight design pattern (see http://en.wikipedia.org/wiki/Flyweight_pattern).

Implementation Details

The Product API is implemented as a Singleton, as returned by the static `getInstance()` method. When loading Content, Devices, and Manufacturers into the Product Catalog, it is essential to load the data in the following order:

1. Manufacturers
2. Devices
3. Content

Since Devices contain information on Manufacturers, the Manufacturer data must be loaded first. Likewise, since Content contains information on the Devices that the Content supports, the associated Devices must be loaded into the Product Catalog prior to the Content. Administrators may create Manufacturers, Devices, and then Content through the Product API's public "import*" methods, each of which take a CSV file containing the appropriate data.

When Manufacturers, Devices, and Content are loaded into the Product Catalog from the "import*" methods, new objects of those types are created and maintained in-memory in the Product Catalog (in the "contentItems" property).

To simplify searches of content, the design calls for the creation of `ContentSearch` objects. When a client of the Product API initiates a new content search via the `searchContent()` method, the client passes a single string as the search criteria. The `searchCriteria()` method passes control to `parseSearchString()`, which returns a `ContentSearch` object back to `searchContent()`. Then `searchContent()` performs the content search over all the `contentItems` in the Product Catalog based on the criteria in the `ContentSearch` object, and finally returns a list of all matching Content items.

Testing

The `TestDriver` class will exercise both importing content into the product catalog via public methods on the `ProductAPI` as well as searching for specific content items. A static `main()` method will interactively prompt the user for Countries, Devices, and Content CSV files and call the public import methods on the `ProductAPI` to load them into the product catalog. Once all required content is loaded, the user will be prompted to supply a CSV query file that will query the API for matching content and output the results to standard out.

Risks

As risks are uncovered in the development process they will be documented here.