Mobile Application Store Product API Design Document

Date: 9/30/13

Author: Yoora Yi Tenen

Reviewer(s): Yetish Narayana

Introduction

This document defines the design for the Mobile Application Store Product API.

Overview

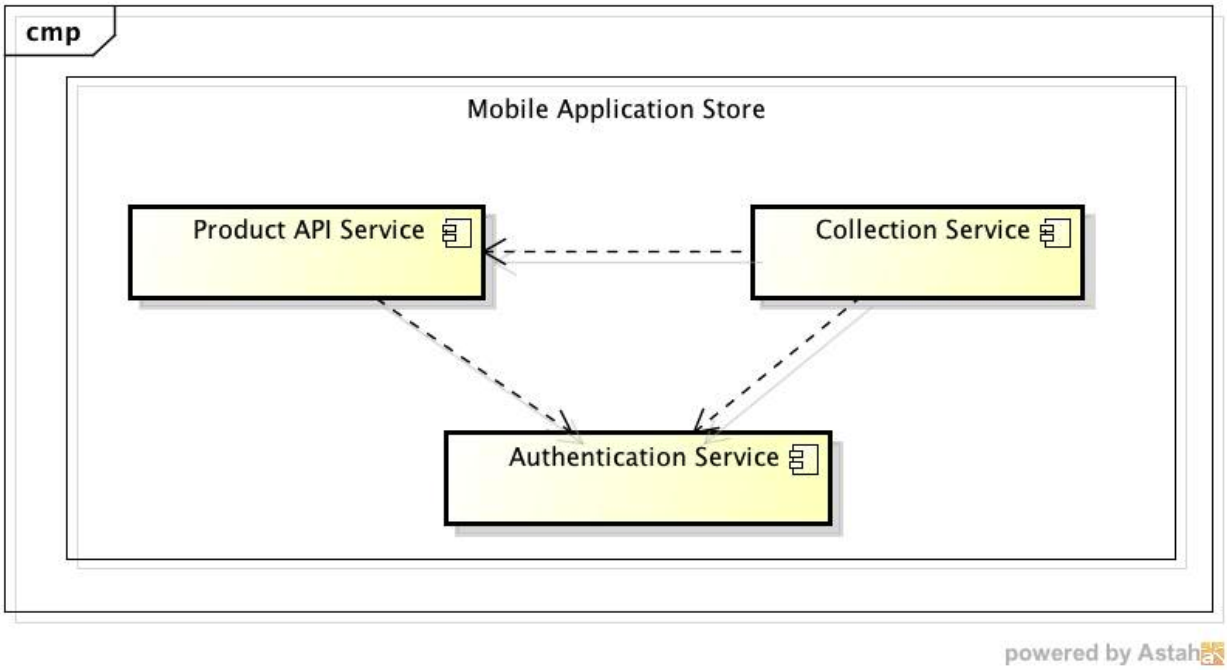
The Product API encompasses the functionality of the product catalog of the mobile application store. The product catalog can contain three different types of products: applications, ringtones, and wallpapers. Users of the product catalog may be categorized as administrators, application developers, and/or consumers.

In order for a product to be properly added to the product catalog, an application developer must provide a valid access token as well as the metadata for the content. This metadata consists of the product's name and description, a thumbnail image, the countries to which the product can be distributed, the supported languages and devices, the price, and the author information.

Administrators of the product catalog maintain the lists of countries and devices that can be associated with products. Adding new countries and devices are also restricted actions and require a valid access token.

All users can search the product catalog's metadata on one or more fields and view a list of the products matching the specified criteria with their details.

The following component diagram shows the three service modules that comprise the Mobile Application Store. The Product API Service depends on the Authentication Service to provide and validate the access tokens that its restricted interfaces for adding countries, devices, and content to the product catalog require. The Collection Service in turn depends on the Product API Service as its functionality is to manage collections of the content that the Product API Service creates and manages.



Requirements

This section defines the requirements for the Mobile Application Store Product API.

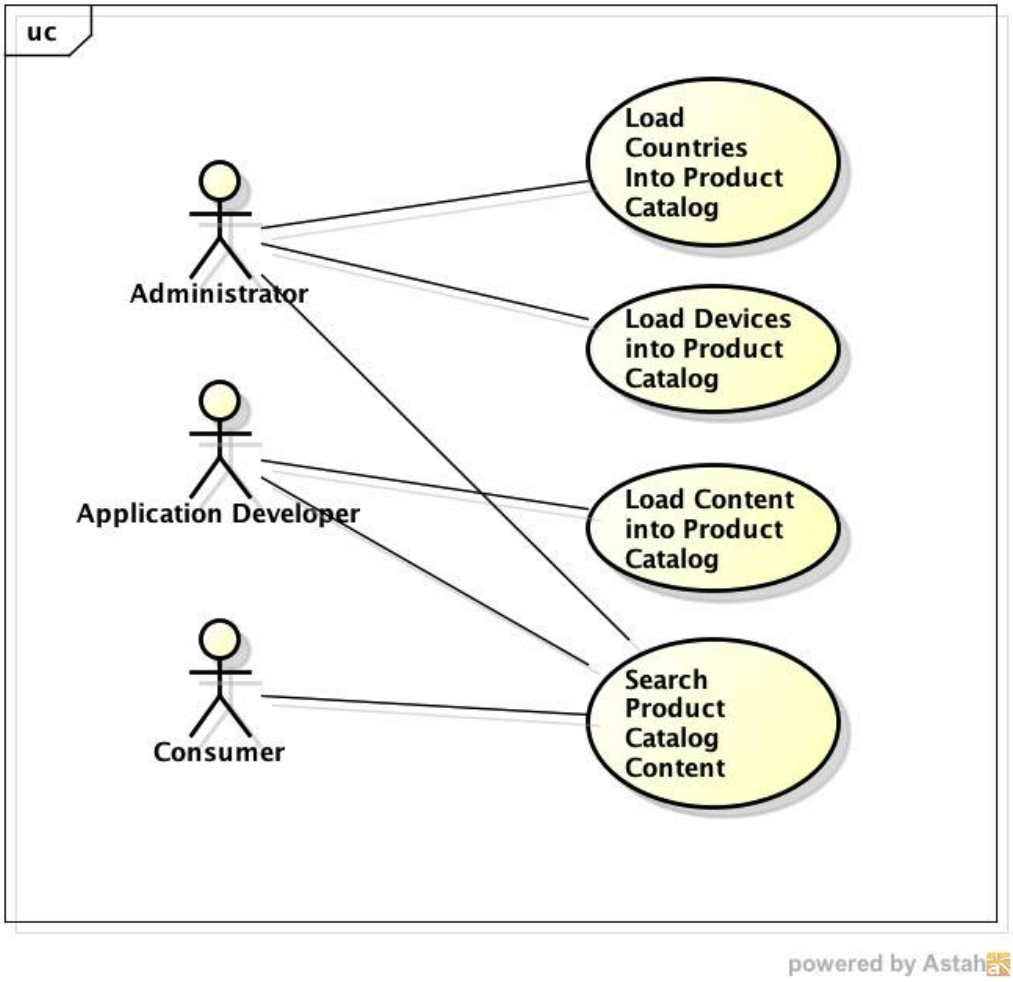
Note: Some of the functionality is considered a restricted interface by means of an access token parameter. The actual validation of the token will be implemented in the Mobile Application Store Authentication Service, but any restricted methods in the Product API should support accepting the access token parameter of type String.

1. Functionality for adding Countries and their required properties (restricted interface).
2. Functionality for adding Devices and their required properties (restricted interface).
3. Functionality for adding Content (products) and their required fields (restricted interface). Content can be concretely defined as Application, RingTone, or Wallpaper, and their fields should be validated before adding.
4. Functionality for searching and displaying Content details (non-restricted interface). Searches should return the Content objects matching the criteria.

No memory persistence is required and a single user may be assumed. For the scope of this assignment, objects will only be created and read, not updated or deleted.

Use Cases

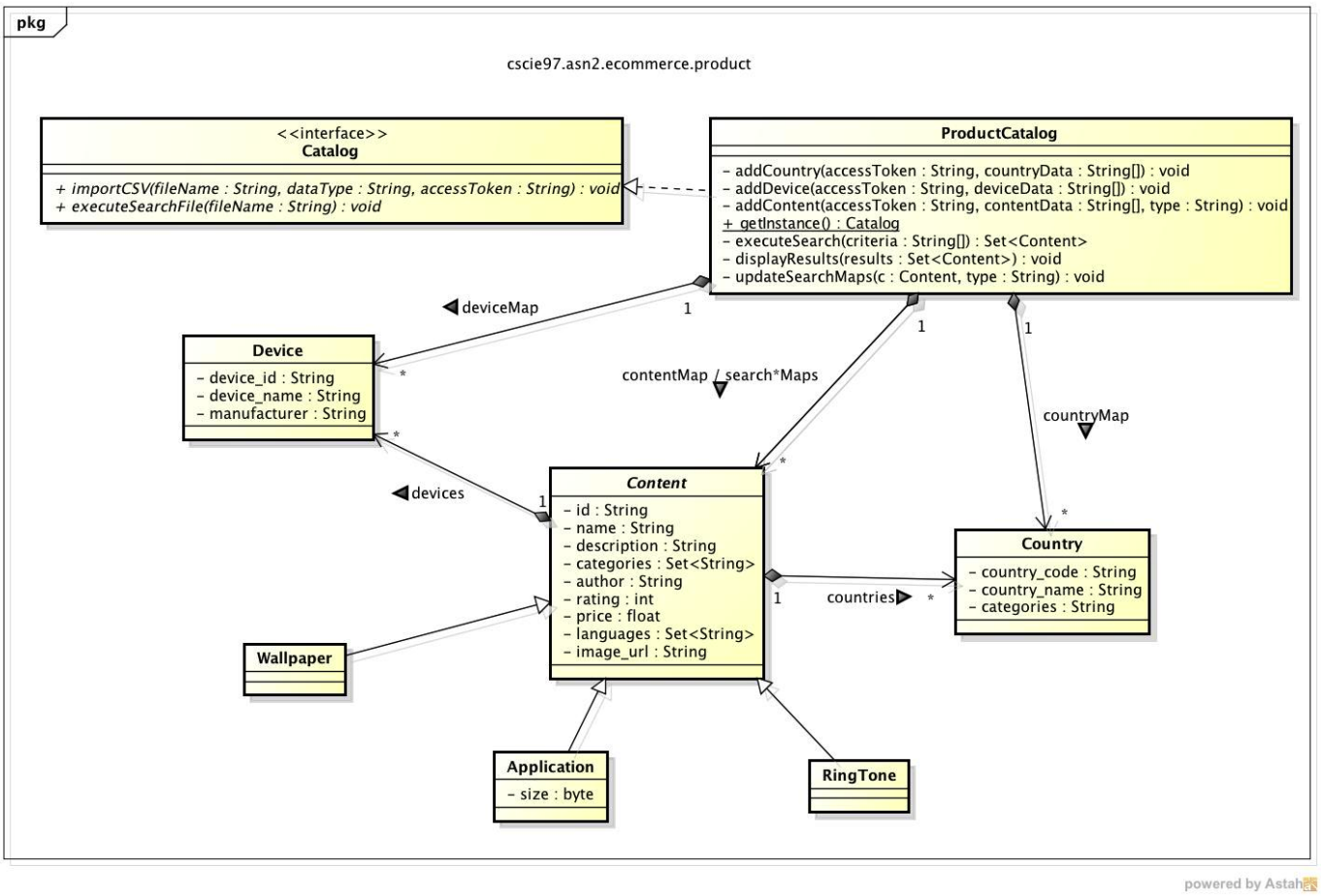
1. Users run searches in the Product Catalog for content falling within certain criteria.
2. Administrators load new Countries into the Product Catalog.
3. Administrators load new Devices into the Product Catalog.
4. Application developers load new Content into the Product Catalog.



Implementation

Class Diagram

The following class diagram defines the Product API implementation classes contained within the package “cscie97.asn2.ecommerce.product”.



Class Dictionary

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

Country

The Country class represents a Country that a product can be distributed to. A Country is uniquely identified by its country_code property. Each Country has an export_status property of “open” or “closed”.

Properties

Property Name	Type	Description
country_code	String	Private unique non-mutable code for the Country. Country code is not case-sensitive and is stored in upper case.

country_name	String	Private unique non-mutable name for the Country. Country name is not case-sensitive.
export_status	String	Private non-mutable export status of the Country. Status can only be “open” or “closed”.

Device

The Device class represents a Device that a product is compatible with. A Device is uniquely identified by its device_id property.

Properties

Property Name	Type	Description
device_id	String	Private unique non-mutable id for the Device. Device id is not case-sensitive.
device_name	String	Private unique non-mutable name for the Device. Device name is not case-sensitive.
manufacturer	String	Private non-mutable name of the device’s manufacturer.

Content

The Content class is the abstract class for the different types of products that can be in the Product Catalog.

Properties

Property Name	Type	Description
id	String	Private unique non-mutable ID for the Content. Id is not case-sensitive.
name	String	Private non-mutable name of

		content. Required.
description	String	Private non-mutable description of content. Required.
categories	Set<String>	Private non-mutable Set of zero or more categories to which the content belongs.
author	String	Private non-mutable name of content's author. Required.
rating	int	Private non-mutable rating of content. Ranging from 0 to 5 where 5 is best.
price	float	Private non-mutable price of the content, in units of BitCoin. Can be 0 for free. Required.
languages	Set<String>	Private non-mutable Set of content's supported language codes.
image_url	String	Private non-mutable URL of the content's thumbnail image.

Associations

Association Name	**Type**	**Description**
countries	Set<Country>	Private non-mutable Set of one or more countries that the content can be distributed to.
devices	Set<Device>	Private non-mutable Set of one or more devices that the content is compatible with.

Application

The Application class extends the Content class and represents an Application in the Product Catalog. An Application is uniquely identified by its ID.

In addition to the properties and associations defined in the Content class, the Application class has an additional size property.

Properties

Property Name	**Type**	**Description**
see **Content** class		
size	byte	Private non-mutable size of the application.

RingTone

The RingTone class extends the Content class and represents a RingTone in the Product Catalog. A RingTone is uniquely identified by its ID.

Properties

Property Name	**Type**	**Description**
see **Content** class		

Wallpaper

The Wallpaper class extends the Content class and represents a Wallpaper in the Product Catalog. A Wallpaper is uniquely identified by its ID.

Properties

Property Name	**Type**	**Description**
see **Content** class		

ProductCatalog

Interface:

The **Catalog** interface provides the public methods for the Catalog service: importCSV() and executeSearchFile().

Exception Classes:

ImportException is thrown when an error is encountered accessing or processing a CSV-formatted file for import (ex. file has invalid header fields).

CatalogException is thrown when an error is encountered creating a Country, Device, or Content object (ex. data is incorrectly formatted).

SearchEngineException is thrown when an error is encountered accessing or processing a search file (ex. data is incorrectly formatted).

The ProductCatalog class is responsible for maintaining and providing access to Country, Device, and Content data. It implements the Catalog interface and is instantiated as a Singleton, using the static method getInstance().

The ProductCatalog imports Country, Device, and Content data from CSV-formatted files where collections are pipe-delimited. Import is restricted by an access token parameter. The methods to create new Countries, Devices, and Content are also restricted by the access token parameter. Following the FlyWeight design pattern, only unique instances of these objects are created and added to the associations countryMap, deviceMap, ringtoneMap, wallpaperMap, and appMap.

Sample Country data in CSV format:

```
# country_code, country_name, export_status
AE, United Arab Emirates, open
AF, Afghanistan, open
AG, Antigua and Barbuda, open
AI, Anguilla, open
AL, Albania, open
AM, Armenia, open
US, United States, open CA, Canada, open
MX, Mexico, open IR, Iran, closed
```

Sample Device data in CSV format:

```
# device_id, device_name, manufacturer
iphone_5, IPhone 5, Apple
```

iphone_6, IPhone 6, Apple

lumina, Lumina, Nokia

galaxy_s3, Galaxy S3, Samsung

galaxy_s4, Galaxy S4, Samsung

glass, Google Glass, Google

nexus, Google Nexus, Google

htc_one, HTC One, HTC

Sample Content data in CSV format with pipe-delimited collections:

#content_type, content_id, content_name, content_description, author, rating, categories, export_countries,supported_devices,price, supported_languages, image_url, application_size

application, a1, Angry Birds Seasons, Angry Birds Seasons, Rovio, 5, game|kids,US|CA|MX,iphone_5|iphone_6|lumina,1 .5,en_us|en_ca|en_gb, http://webassets.angrybirds.com/abcom/img/games/223/Icon_download_seasons_223x223.png, 564

All users are permitted to run searches in the ProductCatalog. Searches are provided in CSV-formatted files where lists are pipe-delimited. Search results are printed to stdout and are preceded by the search string. Malformed searches or problems processing the search input file result in a SearchEngineException, which should provide the search causing the exception and some details about what the specific issue was.

Sample search data in CSV format with pipe-delimited collections:

category list, text search, minimum rating, max price, language list, country code, device id, content type list

search for all content with a minimum rating of 4 and a price of 0 or less (i.e. free)

, , 4, 0, , , ,

In order to improve search performance, Maps are set up for each type of criteria. As Content is added to the ProductCatalog, the Maps are updated so that the Content is indexed and can thus be searched faster than by using brute force. For example, searchCountryMap contains keys for each country code that is added to the ProductCatalog, and the values for each key are the Content objects having that country code.

Methods

Method Name	Signature	Description
importCSV	(fileName:String, dataType:String, accessToken:String):void	Public, restricted method for importing data from CSV-formatted file into the ProductCatalog. Each line is passed to the corresponding ProductCatalog add method per indicated dataType. Throws ImportException on error accessing or processing the file.
addCountry	(accessToken:String, countryData:String):void	Private method for creating a new Country and adding to the countryMap. Is restricted via the accessToken. Throws CatalogException on error parsing or validating countryData.
addDevice	(accessToken:String, deviceData:String):void	Private method for creating a new Device and adding to the deviceMap. Is restricted via the accessToken. Throws CatalogException on error parsing or validating deviceData.
addContent	(accessToken:String, contentData:String):void	Private method for creating new Content and adding to the associated content Map. Updates the criteria Maps accordingly. Is restricted via the accessToken. Throws CatalogException on error parsing or validating contentData.
updateSearchMaps	(c:Content, type:String):void	Private method for updating the criteria Maps with new Content.
executeSearchFile	(fileName:String):void	Public method for parsing searches read from a CSV-formatted file. Delegates

		to executeSearch for the individual searches. Throws SearchEngineException on error accessing or processing the search file.
executeSearch	(criteria:String[]:Set<Content>	Private method for searching the ProductCatalog. Uses the criteria maps to find matching Content and returns a Set of the objects.
displayResults	(results:Set<Content>):void	Private method for displaying the results of a search against the ProductCatalog. Prints search string first, and then the details of each matching Content instance.
getInstance	():Catalog	Private method for returning a reference to the single static instance of the ProductCatalog.

Associations

Association Name	**Type**	**Description**
countryMap	Map<String, Country>	Private association for maintaining active set of valid Countries. Key is country_code and value is the associated Country.
deviceMap	Map<String, Device>	Private association for maintaining active set of valid Devices. Key is device_id and value is the associated Device.
contentMap	Map<String, Content>	Private association for maintaining active set of valid Content. Key is the Content id and value is the associated Content.

searchCatMap	Map<String, Set<Content>>	Private association for maintaining a lookup map on category search criteria. Key is the category and value is the Set of matching Content instances.
searchTextMap	Map<String, Set<Content>>	Private association for maintaining a lookup map on text search criteria. Key is a word and value is the Set of matching Content instances.
searchRatingMap	Map<Integer, Set<Content>>	Private association for maintaining a lookup map on minimum rating search criteria. Initialized with keys 0, 1, 2, 3, 4, 5. Key is the minimum rating and value is the Set of matching Content instances.
searchPriceMap	Map<Float, Set<Content>>	Private association for maintaining a lookup map on maximum price search criteria. Key is the maximum price and value is the Set of matching Content instances.
searchLangMap	Map<String, Set<Content>>	Private association for maintaining a lookup map on language search criteria. Key is the language and value is the Set of matching Content instances.
searchCountryMap	Map<String, Set<Content>>	Private association for maintaining a lookup map on country search criteria. Key is the country_code and value is the Set of matching Content instances.
searchDeviceMap	Map<String, Set<Content>>	Private association for maintaining a lookup map on device search criteria. Key is the device_id and value is the Set of matching Content

		instances.
searchTypeMap	Map<String, Set<Content>>	Private association for maintaining a lookup map on product type search criteria. Initialized with keys “application”, “ringtone”, “wallpaper”. Key is the type and value is the Set of matching Content instances.

Implementation Details

The Product API service is accessed via the ProductCatalog class. This is a Singleton class whose public methods permit instantiation of the Singleton, the import of CSV-formatted files, and the execution of searches. The import of Country, Device, and Content data is restricted via an access token parameter, as is the creation of new Country, Device, and Content objects and the addition of those objects to association maps as appropriate.

Searches are not restricted and are validated before being executed. The method executeSearchFile() parses the search strings in a search file and passes to the processing method executeSearch() a String array containing the search criteria. The search is conducted by checking the ProductCatalog’s search Maps for each specified criterion and returning the intersection of the Sets of matching Content objects, the details of which can then be printed to stdout.

Testing

A test driver class named TestDriver should be implemented in package cscie97.asn2.test with a static main() method that accepts as parameters the file names for four CSV-formatted files containing Country, Device, Content, and search data. The main() method first calls the ProductCatalog method getInstance() to create the Singleton, and then subsequently the ProductCatalog’s importCSV() method for each of the Country, Device, and Content files. It then calls the ProductCatalog method executeSearchFile() using the search file name as a parameter.

Risks

Not using a system for validating languages and categories may risk duplicate and non-normalized language and category values being associated with Content.

There is also the potential for duplicates to be added to the ProductCatalog if there are misspellings or extra spaces between words in product names. For this version of the Product API, it will be assumed that there are no misspellings in product names and as such the validation minimum will be to strip leading and trailing whitespace and to do case-insensitive comparisons of String data.

Lastly, as the import files are in CSV-format, care should be taken to account for non-delimiter commas.