Mobile Application Store Collection Service Design Document

Date: 10/21/13

Author: Yoora Yi Tenen

Reviewer(s): Yetish Narayana

Introduction

This document defines the design for the Mobile Application Store Collection Service, and is organized by the following sections:

- Overview (including a Component Diagram)
- Requirements
- Use Cases (including a Use Case Diagram)
- Implementation (including a Class Diagram and Class Dictionary)
- Implementation Details (including a Sequence Diagram)
- Testing
- Risks

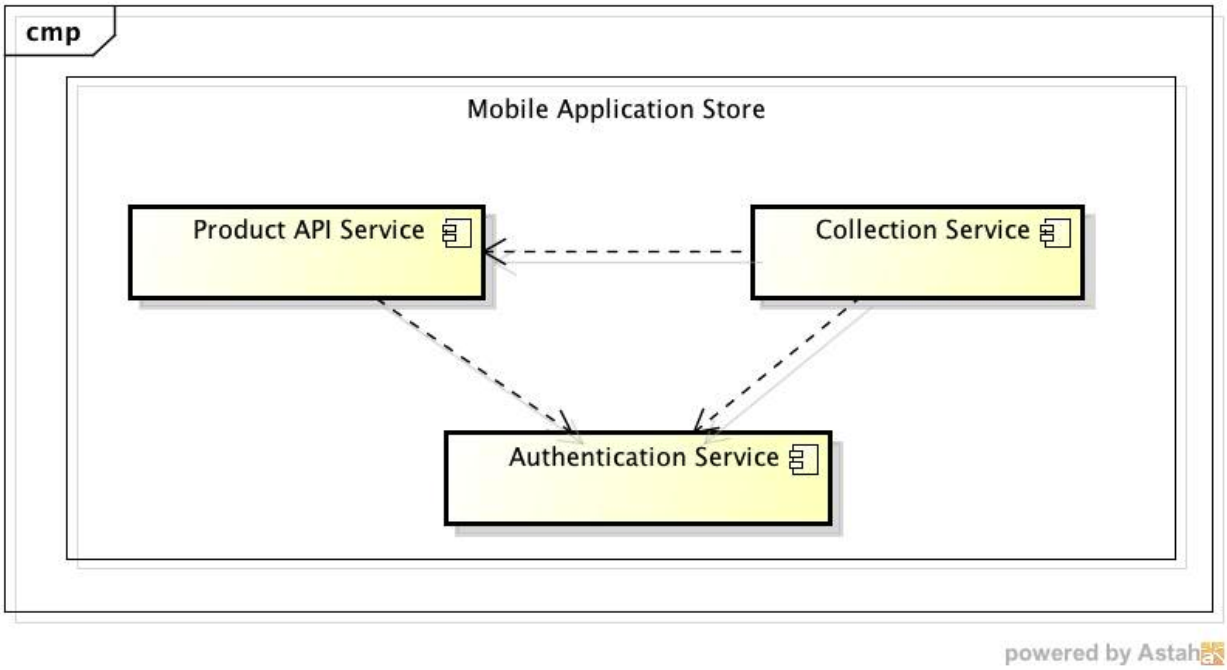
Overview

The Collection Service of the Mobile Application Store oversees the functionality of organizing the store's products into collections. A collection is either static (containing child collections, and a list of contained products) or dynamic (containing child collections, and the search criteria for contained products).

Users interacting with the Collection Service may be categorized as administrators and/or consumers. Administrators can create, add to, search, and iterate collections. Consumers can search and iterate collections. Creating and adding to collections are restricted actions that require the use of a valid access token.

The following component diagram shows the three service modules that comprise the Mobile Application Store. The Collection Service depends on the Authentication Service to provide and validate the access tokens that its restricted interfaces for creating and updating collections require. It also depends on the Product API Service which provides the products that collections

can contain.



Requirements

This section defines the requirements for the Mobile Application Store Collection Service.

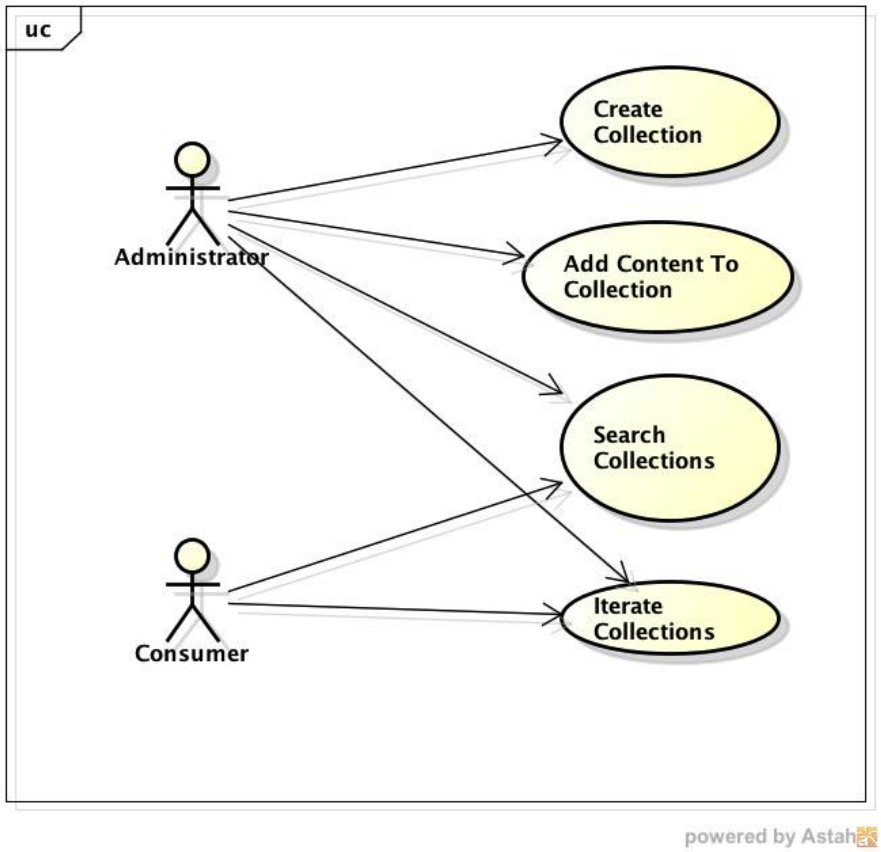
Note: Some of the functionality is considered a restricted interface by means of an access token parameter. The actual validation of the token will be implemented in the Mobile Application Store Authentication Service, but any restricted methods in the Collection Service should support accepting the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

1. Functionality for creating static and dynamic Collections (restricted interface).
2. Functionality for adding content (Collections, Products, product search criteria) to Collections (restricted interface).
3. Functionality for executing text search on Collections (non-restricted interface). Should return the Collections that have the search text in their names and/or descriptions.
4. Functionality for iterating (depth-first) over a specific Collection or all Collections (non-restricted interface).

No memory persistence is required and a single user/thread may be assumed. For the scope of this assignment, Collections will be created, updated, and read, but not deleted.

Use Cases

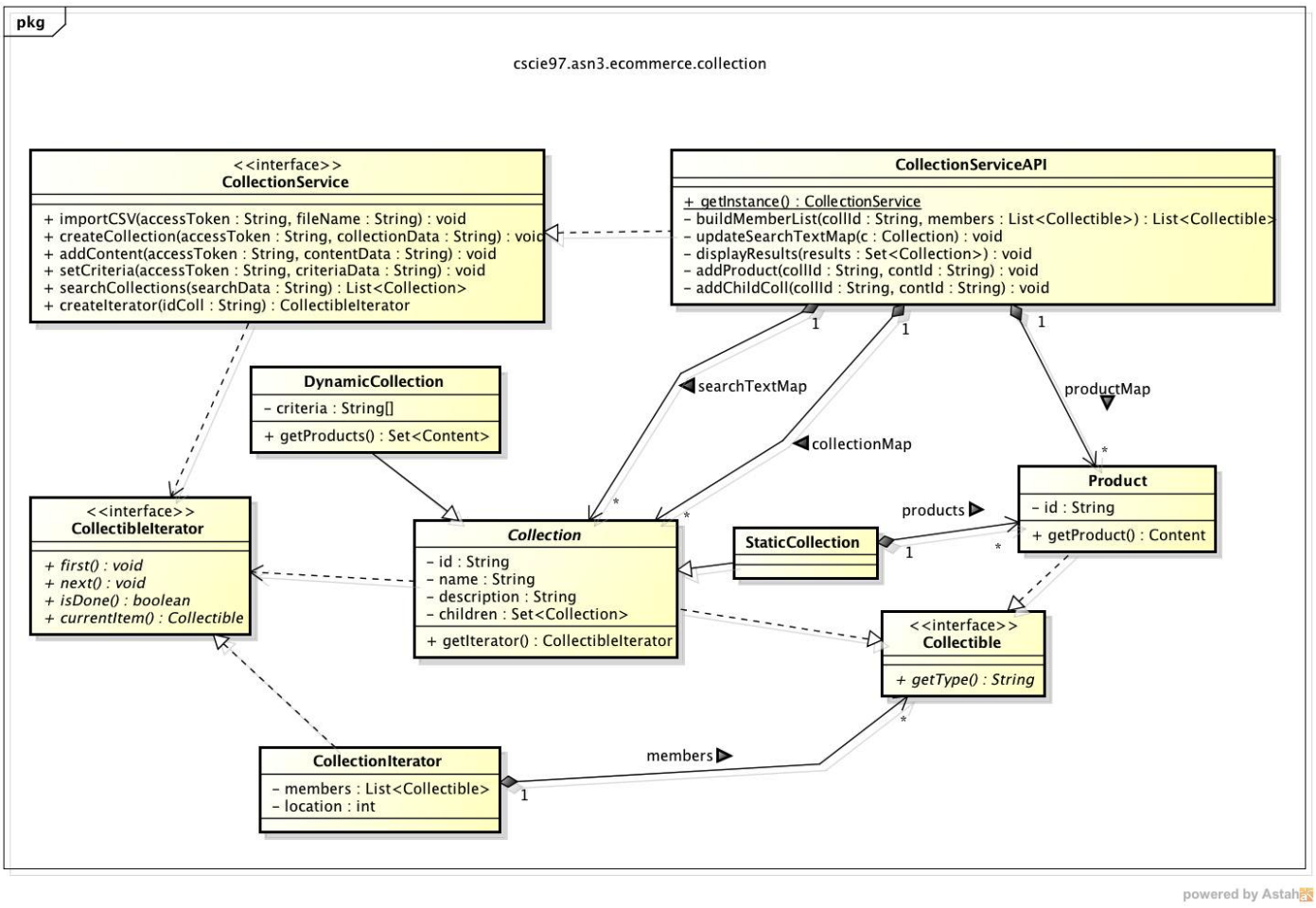
1. Users (consumers or adminstrators) search Collections and view any matches on name or description.
2. Users (consumers or administrators) iterate over Collections to display their details.
2. Administrators create new Collections.
3. Administrators add content (products, product search criteria, child collections) to Collections.



Implementation

Class Diagram

The following class diagram defines the Collection Service implementation classes contained within the package “cscie97.asn3.ecommerce.collection”.



Class Dictionary

This section specifies the class dictionary for the Collection Service. The classes should be defined within the package “cscie97.asn3.ecommerce.collection”.

All classes implement standard public getter methods for their properties and associations.

Collectible

The Collectible interface is used to indicate that an object can be added to a Collection, and is implemented by the Product and Collection classes. Collections are recursive and may contain other Collections. Collections may also contain Products, and so the Collectible interface serves to design the Collection as a composition, facilitating the navigation of its structure.

Methods

Method Name	Signature	Description
getType	():String	Public method that returns the type (implementing class) of the Collectible.

Product

The Product class implements the Collectible interface and represents the products that Collections can contain. It is a wrapper class that interacts with the Product API to obtain product information for the product having the same product id.

Methods

Method Name	Signature	Description
getProduct	():Content	Public method that interacts with the Product Service API via its getProduct method to return the Content which the Product is wrapping.
getType	():String	Public method that returns the type (implementing class) of the Collectible, in this case “product”.

Properties

Property Name	Type	Description
id	String	Private unique identifier for the Product. Identifier is not case-sensitive. Required.

Collection

The Collection class implements the Collectible interface and is the abstract class for the different types of collections that the Mobile Application Store supports.

Methods

Method Name	Signature	Description
getIterator	():Iterator	Public method that returns an Iterator for the Collection.
getType	():String	Public method that returns the type (implementing class) of the Collectible, in this case “static_collection” or “dynamic_collection”.

Properties

Property Name	Type	Description
id	String	Private unique identifier for the Collection. Identifier is not case-sensitive. Required.
name	String	Private name of Collection. Required.
description	String	Private description of Collection.
children	Set<Collection>	Private set of zero or more child Collections.

StaticCollection

The StaticCollection class extends the Collection class and represents a static collection. Products are added to this type of collection by their identifiers. In addition to the properties defined in the Collection class, the StaticCollection class has a products property.

Properties

Property Name	Type	Description
see **Collection** class		
products	Set<Product>	Private set or zero or more

		Products that the StaticCollection contains.

DynamicCollection

The DynamicCollection class extends the Collection class and represents a dynamic collection. Products are added to this type of collection by search criteria, to be generated once they are requested. In addition to the properties and methods defined in the Collection class, the DynamicCollection class has a criteria property and a getProducts method.

Methods

Method Name	Signature	Description
getProducts	():Set<Content>	Public method that interacts with the Product Service API via its executeSearch method to return the set of Content for the DynamicCollection's search criteria.

Associations

Association Name	Type	Description
see **Collection** class		
criteria	String[]	Private String array containing search criteria for the products that should populate the DynamicCollection.

CollectibleIterator

The CollectibleIterator interface provides the public methods for iteration over Collectible items.

Using the CollectibleIterator pattern provides a standard template for navigating Collections and leaves their structural details open to future modifications.

Methods

Method Name	Signature	Description
first	():void	Public method that sets the CollectibleIterator at the first item. Throws Exception on error.
next	():void	Public method that advances the CollectibleIterator to the next item. Throws Exception on error.
isDone	():boolean	Public method that returns true if the CollectibleIterator is at the end of the items; false if not. Throws Exception on error.
currentItem	():Collectible	Public method that returns the item at which the Iterator is positioned. Throws Exception on error.

CollectionIteratorException

CollectionIteratorException is thrown when an error is encountered iterating over a collection (ex. Collection is empty or the iterator is past the end of the Collection).

CollectionIterator

The CollectionIterator class implements the Iterator interface and provides an Iterator for a Collection. The CollectionIterator conducts a depth-first iteration over contained Collections and Products.

Methods

Method Name	Signature	Description
first	():void	Public method that sets the CollectionIterator at the first Collectible in the Collection. Throws CollectionIteratorException if the Collection is empty.

next	():void	Public method that advances the CollectionIterator to the next Collectible in the Collection. Throws CollectionIteratorException if the CollectionIterator is at or past the end of the Collection.
isDone	():boolean	Public method that returns true if the CollectionIterator has reached the end of the Collection; false if not.
currentItem	():Collectible	Public method that returns the Collectible in the Collection at which the CollectionIterator is positioned. Throws CollectionIteratorException if the Collection is empty.

Properties

Property Name	Type	Description
location	int	Private placeholder for the location of the CollectionIterator.

Associations

Association Name	Type	Description
members	List<Collectible>	Private list containing the Collections and Products that populate the Collection for which the CollectionIterator is created.

CollectionService

The CollectionService interface provides the public methods for the Collection Service's API.

Methods

Method Name	Signature	Description
importCSV	(accessToken:String, fileName:String,):void	Public, restricted method for importing data from CSV-formatted file to be processed by the CollectionServiceAPI. Delegates to createCollection, addContent, setCriteria, and searchCollections methods for the actions to be performed for the data. Throws Exception on error accessing or processing the file.
createCollection	(accessToken:String, collectionData:String):void	Public, restricted method for creating a new Collection. Throws Exception on error parsing or validating collectionData.
addContent	(accessToken:String, contentData:String):void	Public, restricted method for adding content to an existing Collection. Throws Exception on error parsing or validating contentData.
setCriteria	(accessToken:String, criteriaData:String):void	Public, restricted method for adding product criteria to an existing Collection. Throws Exception on error parsing or validating criteriaData.
searchCollections	(searchData:String):Set<Collection>	Public method for executing text search for names and descriptions of Collections. Returns the set of matching Collections. Throws Exception on error validating the search data.
createIterator	(idColl:String):CollectibleIterator	Public method for creating a new CollectibleIterator for a

		Collection.

CollectionImportException

CollectionImportException is thrown when an error is encountered accessing or processing a CSV-formatted file for import (ex. file has incorrect number of fields or unrecognized action).

CollectibleException

CollectibleException is thrown when an error is encountered creating a Collection or Product (ex. data is incorrectly formatted).

CollectionSearchException

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

CollectionServiceAPI

The CollectionServiceAPI class is responsible for creating and updating the Collections in the Mobile Application Store. It implements the CollectionService interface and is instantiated as a Singleton, using the static method getInstance().

The CollectionServiceAPI imports Collection data and adds content (Products and product criteria) to existing Collections from CSV-formatted files. These actions are restricted by the use of an access token parameter. CollectionServiceAPI makes use of a factory method createCollection to manage the creation and validation of both static and dynamic collections. Following the FlyWeight design pattern, only unique instances of Collections and Products are created and are kept track of in the associations collectionMap and productMap.

Sample Collection data:

```
# define collections
# define_collection, <collection_type>, <collection_id>, <collection_name>,
<collection_description>
define_collection, static, sports_collection, sports, cool sports apps
define_collection, dynamic, cricket_collection, cricket, cricket apps
```

```
# add content to collections
# add_collection_content, <collection_id>, <content_type>, <content_id>
 add_collection_content, sports_collection, product, Yahoo!_Sports
 add_collection_content, sports_collection, product, Score_Center
 add_collection_content, sports_collection, collection, cricket_collection

# set search criteria for dynamic cricket_collection, dynamic collection criteria
are specified as with product api search criteria
# set_dynamic_criteria, <collection_id>, <category list>, <text search>,
<minimum rating>, <max price>, <language list>, <country code>, <device id>,
<content type list>
set_dynamic_criteria, cricket_collection, , Cricket, , , , , ,

# search collection
search_collection, BIRDS
search_collection, free
search_collection, cricket
search_collection, sports
search_collection, neWs
```

All users are permitted to run searches for Collections on name or description. 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 CollectionSearchException, which should provide the search causing the exception and some details about what the specific issue was.

In order to improve search performance, a searchTextMap is updated whenever Collections are added by the service. The searchTextMap stores the words in Collection names and descriptions as keys, and the values for each key are the identifiers for the matching Collections.

Methods

Method Name	**Signature**	**Description**
importCSV	(accessToken:String, fileName:String):void	Public, restricted method for importing data from CSV-formatted file to be processed by the

		CollectionServiceAPI. Each line is evaluated for the required action and passed to the corresponding method. Throws CollectionImportException on error accessing or processing the file.
createCollection	(accessToken:String, collectionData:String):void	Public, restricted method for creating a new Collection and adding to the collectionMap. Is restricted via the accessToken. Throws CollectibleException on error parsing or validating collectionData.
addContent	(accessToken:String, contentData:String):void	Public, restricted method for adding content to an existing Collection. Delegates to private methods addProduct and addChildColl. Throws CollectibleException on error parsing or validating contentData.
setCriteria	(accessToken:String, criteriaData:String):void	Public, restricted method for adding product criteria to an existing Collection. Throws CollectibleException on error parsing or validating criteriaData.
searchCollections	(searchData:String):List<Collection>	Public method for executing text search for names and descriptions of Collections. Returns the list of matching Collections. Throws CollectionSearchException on error validating the search data.
createIterator	(idColl:String):CollectibleIterator	Public method for creating a new CollectibleIterator for a Collection. Throws CollectibleException if

		Collection does not exist.
buildMemberList	(collId:String,members:List<Collectible):List<Collectible>	Private helper method for recursively building a CollectionIterator's member list. Stops building list when a duplicate Collection is encountered as this indicates a cycle in the Collection tree.
updateSearchTextMap	(c:Collection):void	Private helper method for updating the searchTextMap with a newly added Collection.
displayResults	(results:Set<Collection>):void	Private helper function for printing Collection search results.
addProduct	(collId:String,contId:String):void	Private helper function for adding a child Product to a Collection. Makes a Call to the Product API method getProduct to check whether content exists. Throws CollectibleException if collection is not static or product doesn't exist.
addChildColl	(collId:String,contId:String):void	Private helper function for adding a child Collection to a Collection. Throws CollectibleException if collection to be added as a child doesn't exist.
getInstance	():CollectionService	Private method for returning a reference to the single static instance of the CollectionServiceAPI.

Associations

Association Name	**Type**	**Description**

collectionMap	Map<String, Collection>	Private association for maintaining active set of valid Collections. Key is Collection's id and value is the associated Collection.
productMap	Map<String, Product>	Private association for maintaining active set of valid Products. Key is Product's id and value is the associated Product.
searchTextMap	Map<String, Set<Collection>>	Private association for maintaining a lookup map on text search criteria. Key is a word and value is the Set of matching Collections.

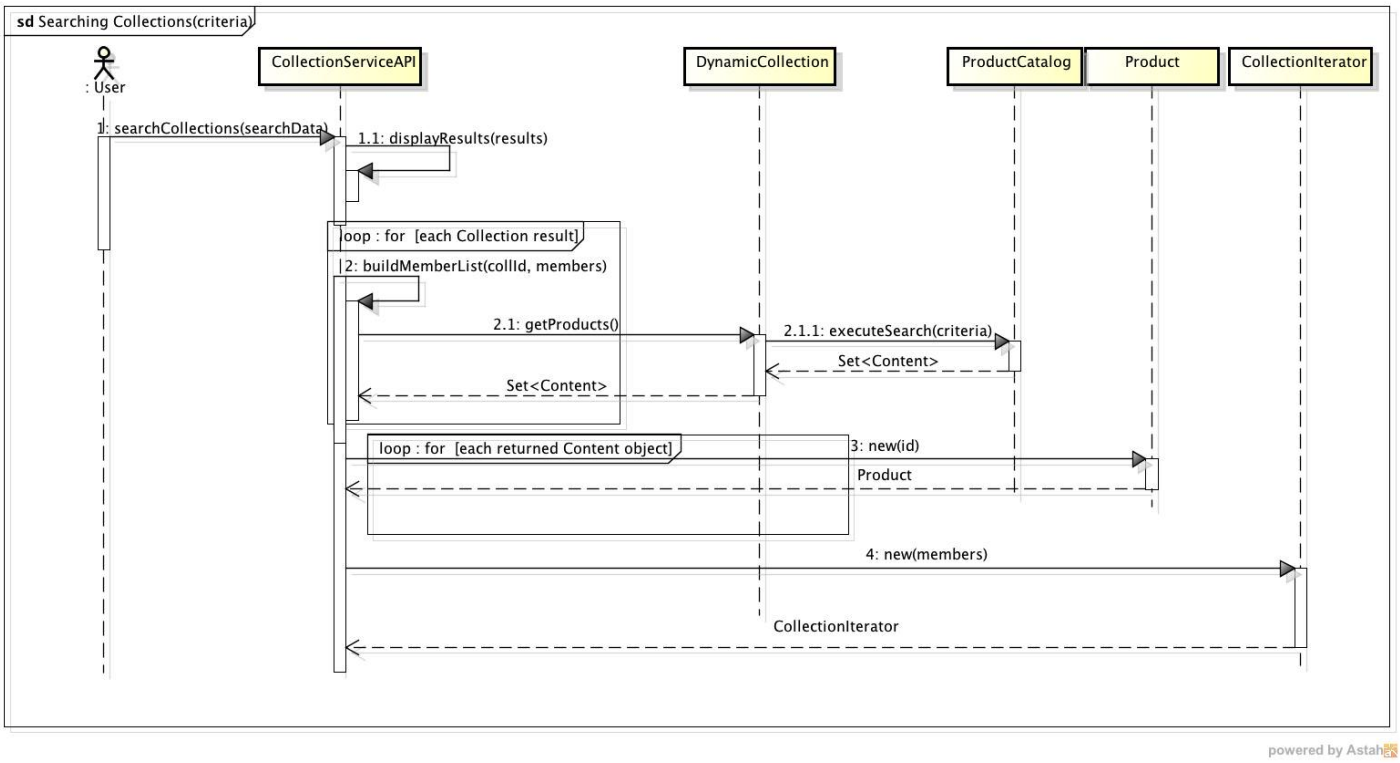
Implementation Details

The Collection Service API is accessed via the CollectionServiceAPI Singleton class. CollectionServiceAPI has public methods to import Collection data, create and update Collections, and search and iterate Collections. The import, creation, and updating of Collections is restricted via an access token parameter.

Collections are composite objects which can contain both child Collections and Products. Products are represented by a wrapper class Product when referring specific products, or by search criteria. Both representations depend on the Product API Service for obtaining the actual objects being represented and their properties. The Product method getProduct() calls the Product API method getProduct(), passing in the Product ID. Product search criteria is sent to the Product API method executeSearch().

Collection searches and iteration are not restricted in their execution. Collection searches are fully managed by the CollectionServiceAPI and processed with the help of private map associations. The iteration of Collections necessitates interaction with the Product API Service, as described in the previous paragraph.

The following sequence diagram depicts the search of Collections and the creation of a CollectionIterator (and the interaction with the Product API Service) in the case that the first Collection result is a StaticCollection and its first child collection is a DynamicCollection.



Testing

A test driver class named TestDriver should be implemented in package cscie97.asn3.test with a static main() method that accepts as parameters the file names for five CSV-formatted files containing Country, Device, Content, Collection, and search data. The main() method first calls the ProductCatalog method getInstance() to create the ProductCatalog Singleton, and then subsequently the ProductCatalog's importCSV() method for each of the Country, Device, and Content files, and the ProductCatalog method executeSearchFile() using the search file name as a parameter.

The main() method then calls the CollectionServiceAPI method getInstance() to create the CollectionServiceAPI Singleton, and subsequently the CollectionServiceAPI's importCSV() method for the Collection data file, which invokes as needed the methods to process and search the data.

Risks

Since collections can contain other collections, cycles may be created in the collection tree. To avoid an infinite iterative loop, the Iterator code should implement some system of keeping track of visited Collectibles.

Products may be added to the ProductCatalog after the time a CollectionIterator is created for a DynamicCollection. An assumption of this implementation is that the CollectionIterator is only

responsible for the existing products at the time of its creation.