|  |
| --- |
| Macmillan |
| Platform-X Caching Strategy |
| Strategy and implementation of content caching |
|  |
| **David Newman** |
|  |

|  |
| --- |
| This document outlines the strategy for caching content and other artifacts in Platform-X. The implementation of the strategy is also discussed by setting requirements for the cache appliance and how it will be integrated with Platform-X. |

Contents

[Overview 4](#_Toc306100423)

[Cache Strategy 4](#_Toc306100424)

[Data Categories 4](#_Toc306100425)

[Long Term 4](#_Toc306100426)

[Short Term 4](#_Toc306100427)

[Transactional 4](#_Toc306100428)

[Cache Invalidation 5](#_Toc306100429)

[Aging 5](#_Toc306100430)

[Direct 5](#_Toc306100431)

[Caching Mechanism 6](#_Toc306100432)

[Cache Opportunities 6](#_Toc306100433)

[Course 7](#_Toc306100434)

[Invalidation Rules 7](#_Toc306100435)

[Product Course 7](#_Toc306100436)

[Invalidation Rules 7](#_Toc306100437)

[Page Layout 7](#_Toc306100438)

[Invalidation Rules 7](#_Toc306100439)

[Primary Navigation 8](#_Toc306100440)

[Invalidation Rules 8](#_Toc306100441)

[User 8](#_Toc306100442)

[Invalidation Rules 8](#_Toc306100443)

[Enrollment 8](#_Toc306100444)

[Invalidation Rules 9](#_Toc306100445)

[User Course Enrollment List 9](#_Toc306100446)

[Invalidation Rules 9](#_Toc306100447)

[Table of Contents 9](#_Toc306100448)

[Invalidation Rules 10](#_Toc306100449)

[Assignment Center Navigation 10](#_Toc306100450)

[Invalidation Rules 10](#_Toc306100451)

[Content Items 10](#_Toc306100452)

[Invalidation Rules 11](#_Toc306100453)

[Content Templates 11](#_Toc306100454)

[Invalidation Rules 11](#_Toc306100455)

[Instructor E-Portfolio 11](#_Toc306100456)

[Invalidation Rules 11](#_Toc306100457)

[Bookmarks 11](#_Toc306100458)

[Invalidation Rules 12](#_Toc306100459)

[Question List 12](#_Toc306100460)

[Invalidation Rules 12](#_Toc306100461)

[Integrating Caching in Platform-X 12](#_Toc306100462)

[ICacheProvider 12](#_Toc306100463)

[Members 13](#_Toc306100464)

[Proof of Concept 13](#_Toc306100465)

[Invalidating Platform-X Data from External Systems 13](#_Toc306100466)

[PxCacheService 13](#_Toc306100467)

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author | Comments |
| 1.0 | **9/30/2011** | David Newman | Initial rev |
| 1.1 | **8/28/2013** | Cheng-Yi Chu | Updated entitlements |
| 1.2 | **9/17/2013** | Jon Graf | Added distributed caching info. Nginx caching. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

Caching is a technique by which commonly accessed data, some of which may be computationally expensive to recreate, is stored is such a way that it can be retrieved quickly with little or no burden on the system. For the purpose of outlining a general caching strategy to be implemented in Platform-X, all data fall into one of the following categories: Long Term, Short Term, or Transactional. Several sections in the document are devoted to defining these categories and how the data in them will be managed.

In order for data to be cache, there must be some mechanism with the appropriate resources and interfaces necessary to achieve a cache's primary goal, i.e. to store and fetch data quickly. Though this document does not specify a particular caching mechanism it does provide a set of requirements that said mechanism must meet in order to successfully implement the strategy outlined in this document.

# Cache Strategy

## Data Categories

There are three categories of data defined in the caching strategy: Long Term, Short Term, and Transactional. Each category stores data based three criteria. First, how expensive is the data to created when not in the cache? Second, how volatile is the data, i.e. how long can we keep it in the cache? Finally, how important is it for the data to stay in the cache when resources are constrained? The following sections define each category and what data are stored in it.

### Long Term

Data in the Long Term category are considered to be either expensive to recreate on-demand or static in terms of volatility. Items in the Long Term cache will stay in the cache until they are either explicitly invalidated or until the caching mechanism removes them while scavenging the cache due to resource limitations. Due to how expensive they are and the fact that they do not change often they are considered very high in terms of cache priority. This means that they will be kept in the cache in place of lower priority items if cache resources run low.

### Short Term

Data in the Short Term category are less expensive to recreate and potentially more numerous than the data in the Long Term category. These data are removed from the cache based on one of several aging mechanism due to their inherit volatility, but may also be removed prematurely if they are explicitly invalidated or the caching mechanism requires their resources to be allocated to higher priority items. Short Term caching will amount to the bulk of data in the cache, and therefore there is a higher probability that they will be removed from cache when resources are constrained.

### Transactional

Data in the Transactional category are not very expensive to recreate and are highly volatile. Ideally, the Transactional cache's lifetime will be bound to a single request and its sole purpose will be to avoid loading duplicate items as there is a low probability that they will change in the context of a single request. Transactional cache expires all of its items as soon as the request is over and therefore contention for resources is not an issue. This means that all transactional data has a high probability of staying in the cache for the entire lifetime of the request.

It should be noted that overuse of the Transactional cache will result in higher memory consumption, and therefore effort should be applied to only cache items that are likely to be requested multiple times per request.

## Cache Invalidation

There are several mechanisms by which an item in the cache can become invalid. The primary and preferred mechanism is through aging. The next most common method is direct invalidation which will only occur if the application sends a signal to the cache, normally only when the application registers a user request that modifies the cached item. The final method is when the caching mechanism itself removes an item due to lack of resources, this is commonly referred to as scavenging.

### Aging

Aging is the preferred method for removing items from the cache. In order to properly manage aging, the application must have an understanding of the cache item's volatility. Highly volatile items cannot be kept in cache for long duration due to the risk of coherency issues between the cached copy of the item and its true state. There are two forms of aging, static and sliding.

#### Static Aging

Static aging is when the application sets a specific duration on the item when it submits it to the cache mechanism. This is typically effective for nonvolatile items, such as those in the Long Term category. It is recommended that statically aged cache items have a high cache priority so that they are not removed prematurely due to scavenging.

#### Sliding Aging

Sliding aging is similar to static aging in that the application must specify a duration when the item is submitted to the cache. The difference is that the duration specified is relative to the item's last cache hit. In other words, every time the item is read from the cache it's lifetime is reset to the initial duration. This is useful for items that are volatile, but frequently accessed. As long as the item is being requested and it hasn't been specifically invalidated by the application, it will continue to be served.

### Direct

Direct invalidation occurs based on a request from the application. This happens in cases where the application gets an update to the cached data from the user. Direct invalidation should occur rarely as it prematurely removes items from the cache, reducing the usefulness of caching in the first place. Direct validation can occur for any item as it happens under specific circumstances.

# Caching Mechanism

This document does not dictate a specific caching technology or vendor, but it does supply enough information to formulate requirements for selection. The following list of requirements was created with only Platform-X in mind, however many of the requirements are general enough in nature that selecting a caching mechanism based on them will allow it to be used in other applications in the enterprise.

* Must be a turn-key, appliance like solution.
* Must be a distributed cache accessible by multiple servers and applications.
* Supports all forms of cache invalidation described in this document including age based (static and sliding), direct, and scavenging.
* Operates over a high-speed protocol such as TCP/IP.
* Allows storage of data in binary form.
* Meets a minimum cache read time of 50ms, including network transfer.
* **Provides an API that allows for direct integration with Microsoft .NET based applications; otherwise provides a fully documented protocol.**
* Cache resources should be expandable in such a way that it would scale with Platform-X as it grows. Scaling should happen is such a way as to not require changes to the application.
* Ability to get analytical information about cache usage. E.g. Cache Hit Ratio, Total Cache size, Average Response time, etc.

# Cache Opportunities

The following table identifies the items that should be cached in Platform-X along with how they fit into the overall strategy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Category | Priority | Invalidation | Scope |
| Course | Long Term | High | Sliding, 60 min | All Users |
| Product Course | Long Term | High | Sliding, 60 min | All Users |
| Page Layout | Short Term | Normal | Sliding, 30 min | All Users |
| Primary Navigation | Short Term | Normal | Sliding, 30 min | All Users |
| User | Short Term | Low | Sliding, 5 min | All Users |
| Enrollments | Short Term | Low | Sliding, 30 min | All Users |
| User Enrollment List | Short Term | Low | Static, 30 min | All Users |
| Table of Contents | Short Term | Normal | Sliding, 5 min | Students |
| ~~Assignment Center~~  ~~Navigation~~ | ~~Short Term~~ | ~~Normal~~ | ~~Sliding, 5 min~~ | ~~Students~~ |
| Content Items | Short Term | Low | Sliding, 30 min | All Users |
| Content Templates | Short Term | High | Sliding, 30 min | Instructors |
| ~~Instructor EPortfolio~~ | ~~Short Term~~ | ~~Low~~ | ~~Sliding, 10 min~~ | ~~Students~~ |
| Bookmarks | Short Term | Low | Sliding, 10 min | All Users |
| Question Lists | Short Term | Low | Sliding, 10 min | All Users |
| Images | Long Term | High | Sliding, 10 min | All Users |
| HTML pages (e-books) | Long Term | High | Sliding, 10 min | All Users |

The following sections describe the data included in the cached copy of each item and the cases that will trigger a direct invalidation.

## Derivative Course

### Purpose:

The course object is loaded multiple times as the user browses each individual page in a specific site. This object contains a variety of settings which are used to control the functionality and the display of different elements on individual pages. These settings rarely change during the average instructor’s review of the site, and never change for a student’s review of the site, and for that reason there is no value in loading this data each time from the backend Agilix system where it is stored.

The loading of the course object does not take a significant amount of time to load from DLAP, but caching this load will increase the load time slightly, and will reduce the overall load on the backend Agilix system. The course object should only need to be loaded by the first user to access the course, and could remain in cache until one of the activation rules is hit, or it expires thanks to lack of use.

### Cache Key:

PX\_COURSE\_{CourseId}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Id | Unique id of the course. |
| Title | Title of the course. |
| GroupSets | Set of user groups in the course. |
| Course Product Name | Name of the product the course was derived from. |
| InstructorName | Name the instructor entered when course was created. |
| InfiniteScroll | Setting that determines if infinite scroll is on or off. |
| MoreResources | Setting that determines if more resources is on or off. |
| ActivatedDate | Date when the course was activated. |
| Categories | Set of ToC categories that is defined for this course. |
| SearchSchema | Searchable categories/tags for this course. |
| LearningObjectives | Set of learning objectives for the course. |
| Properties | Generic properties collection for the course. |

### Invalidation Rules:

* If course is activated.
* If list of Learning Objectives is changed.
* If list of GroupsSets is changed.

## Product Course

### Purpose:

Similar to the derivative course object, when a product course is being viewed, the settings for that course are needed often to help determine what should be displayed or how site functionality should be presented. Unlike the derivative course, for the student AND the instructor these values will never change (as product settings are driven by Sitebuilder). Cache is desired here to reduce the number of times this object is pulled from Agilix using DLAP.

The loading of this course object from DLAP does not take a significant amount of time, but the load from the cache is expected to be faster and will reduce the load on the Agilix backend systems.

### Cache Key:

PX\_COURSE\_{CourseId} (same as derivative course)

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Id | Unique id of the course. |
| Title | Title of the course. |
| GroupSets | Set of user groups in the course. |
| Course Product Name | Name of the product the course was derived from. |
| InstructorName | Name the instructor entered when course was created. |
| InfiniteScroll | Setting that determines if infinite scroll is on or off. |
| MoreResources | Setting that determines if more resources is on or off. |
| ActivatedDate | Date when the course was activated. |
| Categories | Set of ToC categories that is defined for this course. |
| SearchSchema | Searchable categories/tags for this course. |
| LearningObjectives | Set of learning objectives for the course. |
| Properties | Generic properties collection for the course. |

### Invalidation Rules

* External invalidation only, none implemented in PX

## Page Layout

### Purpose:

The page layout of the homepage (and in theory any other page) is currently dynamically generated at the time of display based on folder based “zone” structures that exist in Agilix. These folders contain a list of content items which represent the widgets, and the configuration information for those widgets that are displayed on the page. The zone folders for each page exist under a Agilix node that represents each page as a whole and contains page level properties like title and display status. When the homepage is displayed, DLAP is queried for the core home page object, and the contents of each of the zones that are used to display it. Caching this information will reduce the number of calls made to DLAP on the display of the homepage from 3 per user per course per view, to 3 per course each time the home page was updated.

None of the three calls used to load the home page structure from DLAP take a long time, but the caching of the result will reduce the overall # of calls to DLAP and reduce the numbers of calls against the cache to load this information to 1.

### Cache Key:

PX\_Course\_{CourseId}\_PAGE\_{PageID}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Title | The title of the page to be displayed in the browser bar |
| DisplayType | The display status of the page (just for instructors, for students, for product course only, for derivative course only, etc) |
| Zone Array | Array of zones that make up the page |
| MenuID | The ID of the menu expected to display on the page |

#### Zone Object (for Zone Array):

|  |  |
| --- | --- |
| Property | Description |
| AllowedWidgetMasterList | Set of Widgets allowed in the zone. |
| CurrentWidgetList | Set of Widgets currently in the zone. |
| Widget Array | Configuration information for each Widget in the zone. |

### Invalidation Rules:

* If a new widget is added or removed from the page.
* If a widget's position is changed on the page.
* If a widget is edited, e.g. title changed, any property change, or visibility changed on the page.

## Primary Navigation

### Purpose:

The navigation bars displayed at the top of each page are also dynamically generated off of existing structures in Agilix. Each menu (there can be more than one, but at this time there is only one) is represented by a folder in the Agilix structure, that has child items representing each of the items in the menu. Currently when the menu is displayed (which occurs on each display of a page) a single DLAP call is made to return the children of the page’s menu folder. Caching the menu will reduce the calls to DLAP to 1 per user per page to 1 per page until the menu items are changed.

### Cache Key:

PX\_COURSE\_{CourseId}\_NAV\_{MenuId}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Title | The title of the menu (not used currently by PX) |
| Menu Item Array | Array of menu items that make up the page |

#### Menu Item Object (for Menu Item Array)

|  |  |
| --- | --- |
| Property | Description |
| TocId | Id of the item in the ToC the menu item points to. |
| IsActive | True if the menu item is active or disabled. |
| IsFnE | True if the menu item should open in the FnE window when clicked. |
| DisplayTitle | Title to display for the menu item. |
| TitleUrl | Url the user is taken to when item is clicked. |

### Invalidation Rules

* If a menu item is added or removed.
* If a menu item's title is changed.
* If a menu item's url is changed.
* If a menu item's IsFnE property is changed.
* If a menu item's visibility is changed.
* If a menu item's position is changed.

## User

### Purpose:

The user object is used often by the application to properly display or use information about the current user within the application. Caching the user object will allow for the loading of the users properties from DLAP only once, and the storage of this information in cache when the user is updated (which only can happen externally to PX at this time). Calls to DLAP will be reduced from 1 per user per page to one per user.

The DLAP call to Agilix to load the user data for a single user does not take a long time, but it is expected that the loading of this information from the cache will be somewhat faster, and will reduce the load on the backend Agilix application.

### Cache Key:

PX\_USER\_{UserId}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Id | Unique Id of the user in DLAP. |
| FirstName | First name of the user. |
| LastName | Last name of the user. |
| Email | E-mail address of the user. |
| Username | Domain/username format of username. |
| LastLogin (not cached) | Last date and time user logged into the system. |
| ReferenceId | External Id, in our case the user's RA id. |

### Invalidation Rules

* If a user's name is changed (via external system)
* If a user's e-mail is changed (via external system)
* If a user's username is changed (via external system)

## Enrollment

### Purpose:

For each course a user is enrolled in, that user’s status and permissions against that course are stored in an enrollment object. This enrollment object is loaded often to determine what functionality a user should have access to (if the user is an instructor or a student for example). Caching the enrollment will reduce the number of DLAP calls from 1 per user per page per course to 1 per user per course.

The call to DLAP to load the enrollment object is fast, but we expect that moving this into the cache will increase speed slightly, and will reduce the load on the backend Agilix server.

### Cache Key:

PX\_USER\_{UserId}\_Enrollment \_{EnrollmentId}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Id | Unique Id of the enrollment in DLAP. |
| Course | Reference to the course the enrollment is for. |
| Grades | Set of all grades received so far in the course. |
| DomainId | Id of the domain the enrollment is in. |
| Flags | User rights flag. |
| Status | Status of the enrollment, active or otherwise. |
| StartDate | Date the enrollment started. |
| EndDate | Date the enrollment is set to end. |
| Reference | External reference id for the enrollment. |
|  |  |
| Not Cached |  |
| OverallGrade | Current grade for the course to date. |
| PercentGraded | Percent of gradable items that are graded. |
| User | Reference to the user the enrollment is for. |

### Invalidation Rules

* If the status of the enrollment is changed (via external system).

## User Course Enrollment List

### Purpose:

As part of the current site navigation, the user is provided with a list of all the courses that they are enrolled to. This list is loaded using many calls against DLAP. The first to get the list of user enrollment IDs, and the remaining calls are made, one for each ID returned, to convert the enrollment ID to the actual course name that the user is enrolled in. Caching the resulting list of course IDs will significantly reduce the DLAP activity required for this call as it could be made once per user, where as now it is dynamically called every time the user attempts to review the list of courses.

### Cache Key:

PX\_USER\_{UserId}\_{DomainId}\_ENROLLMENTLIST

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Course Array | Array of courses that the user is enrolled in |

#### Course Object (for Course Array):

|  |  |
| --- | --- |
| Property | Description |
| CourseId | Id of the Course. |
| CourseTitle | Title of the Course. |
| DomainId | Id of the Domain the courses and enrollments exist in. |

### 

### Invalidation Rules

* Externally invalidated when the user is enrolled in a new course.

## Item Children

### Purpose:

The hierarchical nature of content in the Agilix solution allows for any given content item to have children. The display of content in the content browser requires calls to DLAP for the loading of content itself, and then an additional call to DLAP for the loading of that content’s children. This call for child items can be filtered by the “category” or the “filter type” of the content (which was applied by SiteBuilder editors). Caching the GetChildItems call with the filter will allow for PX to reduce the number calls to DLAP for these lists of children. DLAP calls will be reduced from 1 per user per content expansion to 1 per content expansion.

The DLAP call to retrieve child content for a specific item for a specific filter does process XPATH to identify the correct items to return. Caching this result will increase the speed of result and will remove the load on Agilix backend server.

### Cache Key:

PX\_COURSE\_{CourseId}\_TOC\_{CategoryId}\_{ParentItemId}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Child Item Array | Array of child items results for the parent ID and category id |

#### Child Item Object (for Child Item Array):

|  |  |
| --- | --- |
| Property | Description |
| Id | Id of the item the TocItem represents. |
| Title | Title of the TocItem. |
| ParentId | Id of the item's parent in the ToC tree. |
| Description | Description of the ToC item. |
| ItemType | Type of content the ToC item represents. |
| Level | Depth of the ToC item in the tree. I.e. how many generations from root. |
| Sequence | Order of the ToC item relative to its siblings. |
| IsHiddenFromStudents | True if the item is hidden from students. |
| IsAssigned | True if the item represents an assignment. |
| DueDate | The date the item is due, if it is assigned. |
| MaxPoints | Number of points the item is worth. |

### Invalidation Rules

* If an item's title is changed the keys containing the parent ID are invalidated.
* If any item property is changed, keys containing the parent ID are invalidated.
* If a child is added to a parent, keys containing the parent ID are invalidated
* If an item is moved from one parent to another, the keys containing both the former and the new parent ID are invalidated

## Assignment Center Navigation – REMOVED

*Assignment center list is highly user dependant, and caching a complete list of all assigned items with their states, due dates, etc per user is prohibitively expensive.*

### Purpose:

Assignment Center Navigation is highly dynamic and therefore the list of items in the Assignment Center is cached rather than the final navigation. The final display of the items in Assignment Center is calculated from the cached list of items based on each item's properties and the user viewing the Assignment Center.

### Cache Key:

PX\_COURSE\_{CourseId}\_AC

The data cached for the Assignment Center is simply a list of the item Ids for the content that is currently in the Assignment Center. The actual data for each item is stored in the general content item cache discussed in the next section.

### Invalidation Rules

* If an item is moved to a different section in the Assignment Center the general item cache for that item is invalidated.

## Content Items

### Purpose:

Content items are loaded from Agilix by ID to display titles, descriptions, or any set of fields in many areas across the site. Caching individual content items would reduce the number of calls to DLAP by storing the cacheable (non user based) fields on content after they are first loaded.

Cache Key:

PX\_COURSE\_{CourseId}\_ITEM\_{ItemId}

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Id | Id of the item. |
| Type | Type of the item. |
| Title | Title of the item. |
| Description | Description of the item. |
| ParentId | Id of the item's parent. |
| Sequence | Order of the item relative to its siblings. |
| TemplateParentId | Id of the template the item was created from. |
| Rubric | Information about any rubric that applies to this item |
|  |  |
| Not Cached |  |
| IsBookmarked | True if the item is bookmarked. |
| DueDate | Date item is due if assigned. |
| MaxPoint | Points the item is worth if assigned. |

### Invalidation Rules

* If an item's title is changed.
* If an item is deleted.
* If an item is assigned.
* If an item's sequence is changed.
* If an item's parent is changed.
* If an item has a child added to it, or removed from it.
* If an item's properties are changed.

## Content Templates

### Purpose:

The list of templates that are available on a course are loaded every time the user attempt to create new content in either the content browser or the assignment center, and any time the user attempts to review a content items’ available templates. Caching the DLAP call used to load the templates will allow this call to be preformed once per course instead of every time a user attempts to create new content.

### Cache Key:

PX\_COURSE\_{CourseId}\_TEMPLATES

### Fields:

|  |  |
| --- | --- |
| Property | Description |
| Template Item Array | Array of Template Items |

#### Template Item (for template item array):

|  |  |
| --- | --- |
| Property | Description |
| Id | Id of the item. |
| Title | Title of the template. |
| Type | Template type, which controls if it should be shown in the content browser, the assignment center, or both |
| TemplateParentID | ID of base template from which the current template was created |
| Description | Description of the template. |

### Invalidation Rules

* If a template's title is changed.
* If a template is deleted.
* If a new template is added.

## Instructor E-Portfolio - REMOVED

Instructor EPortfolio structure will be cached with the same get child items cache discussed above

### Cache Key:

PX\_COURSE\_{CourseId}\_EPORTFOLIO

### Fields:

|  |  |
| --- | --- |
| Properties | Description |
| ItemId | Id of the item that is in the e-portfolio. |
| ItemType | Type of the item in the e-portfolio. |
| OwnerId | Id of the user that added the item to the e-portfolio. |
| DateAdded | Date the item was added to the e-portfolio. |
| Children | Set of child e-portfolio items. |
| Title | Title of the e-portfolio item. |

### Invalidation Rules

* When instructor adds a new e-portfolio item.
* When instructor removes an e-portfolio item.
* When instructor changes the title of an e-portfolio item.
* When instructor assigned an e-portfolio item.

## Bookmarks

### Purpose:

User bookmark lists are displayed on every content page, as it whether the current content item is currently bookmarked. Caching will be added on a user level to store the list of all the content that it is currently bookmarked for that user. Implementing this cache will reduce the number of calls needed against DLAP from 3 (one for loading the current content’s bookmarked status, one for loading the list of bookmark ids for that user, and one for loading the names of the content for a passed list of ids) to 0, as the bookmark list would remain in cache, and that list could be compared to the current item in context to determine if it was bookmarked.

### Cache Key:

PX\_USER\_{UserId}\_BOOKMARKS\_{CourseId}

### Fields:

|  |  |
| --- | --- |
| Properties | Description |
| Bookmark Item Array | Array of bookmarked content items |

#### Bookmark Item (for Bookmark Item Array):

|  |  |
| --- | --- |
| Properties | Description |
| Id | Id of the bookmark. |
| ParentId | Id of the parent bookmark, if bookmark is in a folder. |
| ItemId | Id of the item that the bookmark points to. |
| Title | Title of the item the bookmark points to. |
| Sequence | Order of the bookmark relative to its siblings. |
| Url | Url to where the bookmarked content is stored. |

### Invalidation Rules

* When a new bookmark is added.
* When a bookmark is removed.
* When a bookmark folder title is changed.

## Question List

### Purpose:

Currently to display a list of the details of questions in the quiz, a series of DLAP calls are necessary. The first call returns the list of question IDs, and then successive calls are needed to load batches of questions to get their full results. Caching the final result of question details for an entire quiz will allow for a single call access to this information.

### Cache Key:

PX\_COURSE\_{CourseId}\_QUESTIONS\_{QuizId}

### Fields:

|  |  |
| --- | --- |
| Properties | Description |
| Question Item Array | Array of questions in the quiz |

#### Question Item (for Question Item Array):

|  |  |
| --- | --- |
| Properties | Description |
| Id | Id of the question. |
| ItemId | Id of the quiz the question belongs to. |
| EntityId | Id of the entity the question belongs to. |
| Text | Text of the question. |
| Type | Type of question, e.g. multiple choice, etc. |
| Points (not cached) | Point value of the question. |
| Choices | List of choices for the question. |
| Answer | Correct answer of the question. |
| QuestionXML | XML of the question. |

### Invalidation Rules

* If a new question is added to the item.
* If a question is removed from an item.
* If the order of questions is changed.
* If a question is changed either by changing its text, point value, answer, etc.

## Course Items

### Purpose:

To store/fetch course Items with a specific key. Currently, the only key value used is “GetAllTemplates”.

### CacheKey:

PX\_COURSE\_{courseId}\_{key}

## Question List

### Purpose:

To store/fetch questions by course ID.

### Cache Key:

PX\_COURSE\_{CourseId}\_QUESTIONS

## Item Analysis Report by course

### Purpose:

To store/fetch the item analysis report by course

### Cache Key:

PX\_COURSE\_{courseId}\_ANALYSISREPORTS

## Item Analysis Report by course and content

### Purpose:

To store/fetch the item analysis report by course and content ID

### Cache Key:

PX\_COURSE\_{courseId}\_ITEMANALYSISREPORT\_{contentId}

## Facet Search

### Purpose:

To store/fetch faceted search results. This is currently been used in the SearchController class. The {key} is constructed by combining the fields in facetedQuery.

### Cache Key:

PX\_COURSE\_{EntityId}\_FACETED\_SEARCH\_{keys}

## User by domain and reference ID

### Purpose:

To store/fetch User item by domain and reference ID.

### Cache Key:

PX\_USER\_DOMAIN\_{domainId}\_REF\_{referenceId}

## User by user ID and course ID

### Purpose:

To store/fetch User item by user ID and course ID.

### Cache Key:

PX\_USER\_{userId}\_ENROLLMENT\_ENTITY\_{courseId}

## Question lists by Chapter or Quiz

### Purpose:

To store/fetch question lists by ListType. There are currently 2 types : “chapterlist” and “quizlist”.

### Cache Key:

PX\_USER\_QBA\_Course\_{COURSE\_ID}\_{ListType}

## User IDs list by reference ID

### Purpose:

To store/fetch a list of IDs for all domains this RA user belongs to.

### Cache Key:

PX\_USERS\_REF\_{referenceId}

## Course list by reference ID

### Purpose:

To store/fetch a list of courses by reference ID.

### Cache Key:

COURSE\_LIST\_REF\_{ref\_id}

## EPortfolio Dashboard Data

### Purpose:

To store/fetch EPortfolio Dashboard data by course ID.

### Cache Key:

PX\_EPORTFOLIODASHBOARDATA\_{courseId}

## Dashboard Data

### Purpose:

To store/fetch Dashboard data by course ID.

### Cache Key:

PX\_DASHBOARDATA\_{courseId}

## Launchpad Content Item Data

### Purpose:

To store/fetch list of Launchpad content item data by course ID.

### Cache Key:

## PX\_LAUNCHPAD\_DATA\_{courseId}

## Minified stylesheet file

### Purpose:

To store/fetch the minified version of CSS file by url path and query.

### Cache Key:

PX\_CSS\_{urlPathAndQuery}

## Widget stylesheet files

### Purpose:

To store/fetch the CSS file needed for the widget type. The value of the {type} parameter currently used are : “platform”, “widget1”, “widget2”, “course”.

### Cache Key:

PX\_WIDGET\_CSS\_{type}

## RAWS Site UserData

### Purpose:

To store/fetch the Site UserData from RAWS. It should only be cached if the access level is greater than 20.

### Cache Key:

{UserId}:{CourseId}

# Integrating Caching in Platform-X

From the perspective of Platform-X, the caching mechanism is just another component. Like all other components in Platform-X there is no assumption made about how it is actually implemented and it is always used through an strictly defined interface. The following listing describes the Platform-X caching interface and supporting types.

## ICacheProvider

The ICacheProvider interface acts as an abstraction around the actual caching mechanism. This prevents Platform-X from being dependent on a particular caching solution.

### Members

The following table lists all of the members of the ICacheProvider interface and describes how they are used.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Parameters | | Description |
| Store | key: string | Use to retrieve item | Stores and item in the cache using the specified key and settings. |
| item: object | Item to store |
| settings: CacheSettings | Settings for item |
| Remove | key: string | Key of item to remove | Removes the item stored under the given key. |
| RemoveByPattern | regex: string | Pattern of item keys to remove | Removes any items stored under a matching key. |
| Fetch | key: string | Key of item to get | Gets the item stored under the given key. |
| FetchByPattern | regex: string | Pattern of item keys to get | Gets any items stored under a matching key. |

## Proof of Concept

In order to test the validity of the interface, a simple implementation was created using System.Web.Caching.Cache as the caching mechanism. Although the mechanism itself does not meet the requirements outlined previously, it does allow development to move forward until a more suitable caching mechanism is selected.

# Invalidating Platform-X Data from External Systems

Platform-X is an application surrounded by an ecosystem of related systems, some of which are responsible for providing supporting information. For instance, there exists and E-Commerce system that is responsible for creating users and determining what courses they have access to. It is therefore necessary that these external systems have the ability to clear specific data cached by Platform-X.

In order to support access to the cache by external systems a Web Service will be created and deployed to a central location. The following listing describes the service and what operations it supports.

## PxCacheService

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Parameters | | Description |
| InvalidateCourse | courseId:string | Reference Id of the course to clear. | Invalidates all information for a course. This includes items, users, enrollments, etc. |
| InvalidateItem | itemId:string | DLAP Id of the item to clear. | Invalidates the item and all cached data dependent on it. |
| InvalidateUser | userId:string | Reference Id of the user to clear. | Clears the user from the cache along with all user related data such as enrollments, bookmarks, etc. |