Perseus: The Cache Manager – Specification

Authors:

S Chassande-Barrioz (France Telecom R&D)

P Dechamboux (France Telecom R&D)

L Garcia-Banuelos

Released:	January 27, 2004
Status:	Draft
Version:	1.3

Table of contents

1 INTRODUCTION..4

1.1 Overview..4

1.2 Scope..4

1.3 Rationale..4

1.4 Goals..4

1.5Document Convention..4

2 ARCHITECTURE..5

2.1Overview..5

2.2CacheEntry..6

2.3 CacheEntryFactory 7

2.4 Primitive CacheManager..8

2.5 Cache resizing..10

2.6 Entries management..10

2.7 ReplacementManager..11

Table of figures

FIGURE 1: CACHEMANAGER OVERVIEW..5

FIGURE 2: THE TREE OF CACHEENTRY INTERFACES...6

1 Introduction

1.1 Overview

This document presents the definition (API and concepts) of a cache manager component.

1.2 Scope

1.3 Rationale

1.4 Goals

The primary purpose of a cache manager is to improve the overall system performance. The performance gain is obtained by avoiding costs due to I/O operations related to storage/network devices, recreation-related costs, etc. The cache manager component is meant to be used by several services such as the persistence service and the replication service, among others.

1.5 Document Convention

A Times Roman font is used for the default text.

```
  A courier font is used for code fragments.
```

2 Architecture

2.1 Overview

The role of the cache manager is to keep objects in memory. A cache manager manges entries represented by CacheEntry instances. The figure below represents the overall architecture of the CacheManager component. It shows a composite component exporting two interfaces really implemented by two sub components:

- A primitive cache manager mainting a map entry,
- A replacement manager applying an eviction policy of entries

In addition a listener (CacheEventListener) permits to the user of a cache manager to receive callback on cache entry management. Finally The user of the cache manager must specify a factory of CacheEtnry instance.

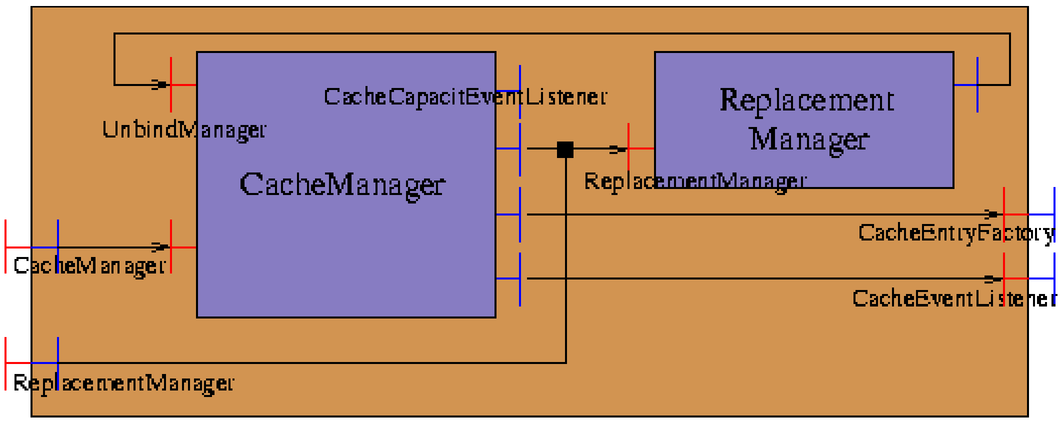

Figure 1: CacheManager overview

2.2 CacheEntry

A CacheEntry is an entry of the CacheManager component. A cache entry references the identifier of the entry (`getCeIdentifier()`) and the real object (`getCeObject()`).

```
package org.objectweb.perseus.cache.api;
public interface CacheEntry {
   Object getCeObject();
   Object getCeIdentifier();
}
```

In order to implement replacement policies and to separate the additional concerns (usage, age), the cache entry concept is extended. The following figure represents the inheritance tree for cache entry interfaces.

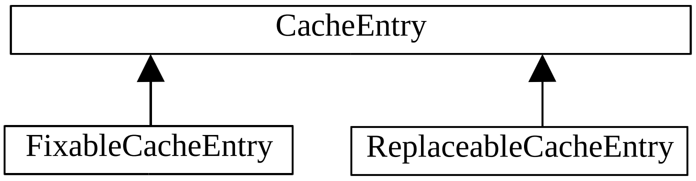

Figure 2: The tree of CacheEntry interfaces

FixableCacheEntry

The first child interface is FixableCacheEntry. It permits to signal the use of an entry of the cache. The `fixCe()` method permits to indicate the start of the real object use, whereas the `unfixCe()` method is the opposite and permits to indicate the end of the real object use. The `getCeFixcount()` retrieves the number of uses of the real object.

```
package org.objectweb.perseus.cache.api;
public interface FixableCacheEntry extends CacheEntry {
   void fixCe();
   void unfixCe() throws UnFixProtocolException;
   int getCeFixCount();
}
```

ReplaceableCacheEntry

The second child interface is ReplaceableCacheEntry. It permits to manage the age of the entry in the cache. The replacement manager can use the age of entries to choose the next entries to remove. The aim of this interface is avoid the use of an internal structure in the replacement manager for managing the age of the cache entries.

```
package org.objectweb.perseus.cache.replacement.api;
public interface ReplaceableCacheEntry
   extends CacheEntry {

   long getCeAge();
   void setCeAge(long age);
}
```

2.3 CacheEntryFactory

A CacheEntryFactory is the component able to create new CacheEntry instances from the identifier and the real object. Here is the definition of the CacheEntryFactory interface:

```
package org.objectweb.perseus.cache.api;
public interface CacheEntryFactory {
   FixableCacheEntry create(Object id, Object obj);
}
```

The use of a separate cache entry factory permits to have generic cache implementation. In addition the user of the cache manager can choose its own composition between the object implementing the CacheEntry interface, the real object and the identifier.

2.4 Primitive CacheManager

The CacheManager is a kind of binder keeping couples (identifier, cache entry). To an identifier corresponds only one cache entry. The primitive cache manager permits to do several actions:

- To add an entry into the cache (`bind`),
- To remove an entry from the cache (`unbind`),
- To search at entry from the cache (`lookup`),
- To signal the use of the entry (`fix`, `unfix`, `touch`).
- To configure the size of cache (`setMaxObjects`, `setMemorySize`).

These differents actions are separated by concern in three interfaces:

- CacheManager
- UnbindManager
- CacheAttributeControleller

The CacheManager interface

The first interface is CacheManager is used by the user of the cache manager. The main methods of the interface CacheManager are:

```
package org.objectweb.perseus.cache.api;
public interface CacheManager {
   CacheEntry bind(Object id, Object object)
        throws CacheException;
```

This method binds a new entry into the cache if there is no existing entry with the same identifier and the maximal cache size is not reached. To build the cache entry instance the primitive cache manager uses the CacheEntryFactory. Secondly he primitive cache manager informes the replacement about the entry adding by calling the `adjustForReplacement` method. Finally the primitive cache manager sends a bound event to the listeners.

```
   CacheEntry lookup(Object id);
```

This method searches an entry in the cache. If the entry does not exist a null value is returned.

```
   void fix(CacheEntry ce) throws CacheException;
```

This method informes the primitive cache manager that the entries is used and must not be evicted from the cacher. If the given CacheEntry instance is not present in the cache, a CacheException is thrown.

```
   void unfix(CacheEntry ce) throws CacheException;
```

This method informes the primitive cache manager that the entries is no more used. If the given CacheEntry instance is not present in the cache, a CacheException is thrown.

```
   void touch(CacheEntry entry) throws CacheException;
```

This method informes the

```
}
```

The UnbindManager interface

The second one is UnbindManager permiting to the replacement manager to remove an entry from the primitive cache manager. In some implementation the instance is not evicted on the call of the unbind method, and will be by the garbage collector later. If the entry is really remove, the primitive cache manager sends an unbound event to listeners.

```
package org.objectweb.perseus.cache.api;
public interface UnbindManager {
   void unbind(CacheEntry, boolean force)
        throws CacheException;
}
```

The CacheAttributeController interface

Finally the third interface named CacheAttributeController contains configuration methods. This interface permits to set the maximal cache size in terms of the number of object (`setMaxObjects`) or in terms of memory size (`setMemorySize`). In addition a simple mechanism permits to specify the number of entries to try to evict (`autoCleanSize`).

```
package org.objectweb.perseus.cache.api;
public interface CacheAttributeController
   extends AttributeController {

   final static int NO_LIMIT = -1;
   void setMaxObjects(int size)
        throws IllegalArgumentException, CacheException;
   int getMaxObjects();

   void setMemorySize(int size)
        throws IllegalArgumentException;
   int getMemorySize();

   void setAutoCleanSize(String size);
   String getAutoCleanSize();
```

The auto clean size can can be a fixed value (ex: 124) or a percent value of the maximal cache size (ex: “8%”).

```
}
```

2.5 Cache resizing

When the maximal cache size is changed through the configuration interfec, the CacheManager must send a CacheCapacityEvent to all registered CacheCapacityEventListener, together with the old and the new size.

```
package org.objectweb.perseus.cache.api;
public interface CacheCapacityEventListener {
   void cacheResized(CacheCapacityEvent event);
}

public interface CacheCapacityEvent {
   static final int NOTIFY_CACHE_RESIZE = 1;
   int getEventId();
     int getOldSize();
     int getSize();
}
```

On a cache reduction size, if the current size is greater than the new size, the cache manager asks to the replacement to try to unbind instance. If it is not possible to evict enough cache entries no more entry can be bound and the size will be reduce later when it will be possible. This configuration is in “best effort” mode.

2.6 Entries management

When a cache entry is added (`entryBound`) or removed (`entryUnbound`), the cache manager must send a CacheEvent to all registered CacheEventListener. The CacheEventListener interface can be implemented at the application level, to ensure an appropriate management of objects held in the cache. For instances, the event listener can implement a callback to deal with housekeeping of complex objects (e.g. closing OS resources allocated to the corresponding object), after it is notified of their corresponding eviction.

```
package org.objectweb.perseus.cache.api;
public interface CacheEventListener {
   void entryBound(CacheEvent event);
   void entryUnbound(CacheEvent event);
}

public interface CacheEvent {
   static final int NOTIFY_BIND = 1;
   static final int NOTIFY_UNBIND = 2;
   int getEventId();
   CacheEntry getEntry();
}
```

2.7 ReplacementManager

The role of a replacement manager is to choose a set of objects to be evicted from the cache, whenever cache space is needed (e.g. to store a recently arrived object). Typical implementations for replacement managers include LRU, MRU, and FIFO based.

The main methods of the interface ReplacementManager are:

```
package org.objectweb.perseus.cache.replacement.api;
public interface ReplacementManager {
     void addForReplacement(FixableCacheEntry entry)
          throws CacheException;
```

The `addForReplacement` method must be used to indicate that the ReplacementManager must manage a new entry. The primitive cache manager calls this method when a new entry is bound into the cache.

```
     void adjustForReplacement(FixableCacheEntry entry)
          throws CacheException;
```

The `adjustForReplacement` method must be called whenever an object has been accessed. Thus, calling this method, gives hints about object usage (frequency, recency of access). This hint is used within the replacement algorithm. The primitive cache manager calls this method when an entry is touched

```
     int forceFree(int capacity) throws CacheException;
```

The call to the `forceFree` method forces the replacement manager to free CacheEntry instances from the cache. The primitive cache manager calls this method when no space is availlable or when the auto clean limit is reached.

```
     void unbind(FixableCacheEntry entry, boolean force)
          throws CacheException;
```

The call to the `unbind` method tries to remove an CacheEntry from the cache and the replacement manager. The boolean parameter indicates if the entry must be removed or try to be removed from the cache. In all case if the entry is fix, it will not be evicted. A cache manager user calls this method for managing manually the cache eviction of an entry.

```
     Set unfixedEntries();
}
```