[TOPCODER]
SOFTWARE

Simple Cache 2.0 Requirements Specification

1. Scope

This component will extend the existing simple cache component. The simple cache has several shortcomings that this component will address. Simple Cache 2.0 must support data compression. Also, a robust API should be added to the simple cache to increase the ease of use. For example, the simple cache 2.0 should provide methods to remove subsets and provide an Iterator to manipulate items in the cache.

1.1 Overview

This component is a more sophisticated extension of the simple cache component. It is intended to quickly implement a server side caching strategy that has a robust functionality and is highly configurable.

1.2 Logic Requirements

1.2.1 Speed

The cache must be designed for speed. The additional functionality added to the existing simple cache must not severely hinder the cache's performance.

1.2.2 Data Compression

The cache will support data compression of the entry objects. This compression should occur on the value of key/value pairs inserted into the cache. By default the cache should not compress data. However, the cache should be able to be configured to be able to use compression. The type of data compression should be pluggable.

1.2.3 Robust API

The cache must provide an extensive API including but not limited to

- Add/remove – add or remove multiple items. Based on keys and values, remove entries from the cache.
- Provide value Iterator – expose a value Iterator for the cache
- Remove similar items – remove items from the cache that are of the same class. For example, remove all objects in the cache that are of type Integer.

1.2.4 Eviction Strategies

The cache must be able to support multiple eviction strategies. For example, it could have one eviction strategy based on the size of objects and one based on the amount of memory remaining.

1.3 Required Algorithms

Any new eviction strategy algorithms.

1.4 Example of the Software Usage

The cache would be used on a server to easily cache objects and manage the cached objects. One example would be caching objects sent to the server by the competition arena applet.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

The cache will be used in conjunction with a Cache Client component. The cache client will provide the means of manipulating a cache residing on a server.

The component must implement the following interface:

Set the eviction strategy for the cache:
setEvictionStrategy(strategy:EvictionStrategy):void
Return the size of the cache in bytes:
getSize():long
Return the maximum size of the cache in bytes:
getMaxSize():long
Remove any objects that are in the cache that are larger then the specified value
removeLarger(value:Object):void
Remove any entry in the cache that have been in the cache longer than the object specified by the key object.
removeOlder(key:Object):void
Remove any entry in the cache that is of the same class as the object specified
removeLike(object:Object):void
Remove all entries in the cache that are specified by the keys Set
removeSet(keys:Set):void
Remove any object specified by the regex string. All keys will be represented by strings.
remove(regex:String):void
Return an Iterator for the values in the cache.
iterator():Iterator
Put all of the values and in the mapping into the cache.
put(cacheEntry:Map):void

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.3, Java1.4

2.1.4 Package Structure

com.topcoder.util.simplecache20

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The size of the cache must be configurable

3.2 Technical Constraints

3.2.1 TopCoder Software Component Dependencies:

- Compression Utility
- Simple Cache 1.0 – this cache must extend the simple cache component.

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.2 QA Environment:

- Solaris 7
- RedHat Linux 7.1

- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.