package com.twitter.tweetypie.caching

import com.twitter.io.Buf

import com.twitter.util.Time

/\*\*

\* How to store values of type V in cache. This includes whether a

\* given value is cacheable, how to serialize it, when it should

\* expire from cache, and how to interpret byte patterns from cache.

\*/

trait ValueSerializer[V] {

/\*\*

\* Prepare the value for storage in cache. When a [[Some]] is

\* returned, the [[Buf]] should be a valid input to [[deserialize]]

\* and the [[Time]] will be used as the expiry in the memcached

\* command. When [[None]] is returned, it indicates that the value

\* cannot or should not be written back to cache.

\*

\* The most common use case for returning None is caching Try

\* values, where certain exceptional values encode a cacheable state

\* of a value. In particular, Throw(NotFound) is commonly used to

\* encode a missing value, and we usually want to cache those

\* negative lookups, but we don't want to cache e.g. a timeout

\* exception.

\*

\* @return a pair of expiry time for this cache entry and the bytes

\* to store in cache. If you do not want this value to explicitly

\* expire, use Time.Top as the expiry.

\*/

def serialize(value: V): Option[(Time, Buf)]

/\*\*

\* Deserialize a value found in cache. This function converts the

\* bytes found in memcache to a [[CacheResult]]. In general, you

\* probably want to return [[CacheResult.Fresh]] or

\* [[CacheResult.Stale]], but you are free to return any of the

\* range of [[CacheResult]]s, depending on the behavior that you

\* want.

\*

\* This is a total function because in the common use case, the

\* bytes stored in cache will be appropriate for the

\* serializer. This method is free to throw any exception if the

\* bytes are not valid.

\*/

def deserialize(serializedValue: Buf): CacheResult[V]

}