package com.twitter.servo.cache

import com.twitter.util.{Duration, Future}

/\*\*

\* [[Memcache]] is a Cache with types that reflect the memcached protocol. Keys are strings and

\* values are byte arrays.

\*/

trait Memcache extends TtlCache[String, Array[Byte]] {

def incr(key: String, delta: Long = 1): Future[Option[Long]]

def decr(key: String, delta: Long = 1): Future[Option[Long]]

}

/\*\*

\* allows one Memcache to wrap another

\*/

trait MemcacheWrapper extends TtlCacheWrapper[String, Array[Byte]] with Memcache {

override def underlyingCache: Memcache

override def incr(key: String, delta: Long = 1) = underlyingCache.incr(key, delta)

override def decr(key: String, delta: Long = 1) = underlyingCache.decr(key, delta)

}

/\*\*

\* Switch between two caches with a decider value

\*/

class DeciderableMemcache(primary: Memcache, secondary: Memcache, isAvailable: => Boolean)

extends MemcacheWrapper {

override def underlyingCache = if (isAvailable) primary else secondary

}

/\*\*

\* [[MemcacheCache]] converts a [[Memcache]] to a [[Cache[K, V]]] using a [[Serializer]] for values

\* and a [[KeyTransformer]] for keys.

\*

\* The value serializer is bidirectional. Keys are serialized using a one-way transformation

\* method, which defaults to \_.toString.

\*/

class MemcacheCache[K, V](

memcache: Memcache,

ttl: Duration,

serializer: Serializer[V],

transformKey: KeyTransformer[K] = new ToStringKeyTransformer[K]: ToStringKeyTransformer[K])

extends CacheWrapper[K, V] {

override val underlyingCache = new KeyValueTransformingCache(

new SimpleTtlCacheToCache(memcache, ttl),

serializer,

transformKey

)

def incr(key: K, delta: Int = 1): Future[Option[Long]] = {

if (delta >= 0)

memcache.incr(transformKey(key), delta)

else

memcache.decr(transformKey(key), -delta)

}

def decr(key: K, delta: Int = 1): Future[Option[Long]] = incr(key, -delta)

}