package com.twitter.tweetypie.caching

import com.twitter.stitch.Stitch

/\*\*

\* Apply caching to a [[Stitch]] function.

\*

\* @see CacheResult for more information about the semantics

\* implemented here.

\*/

class StitchCaching[K, V](operations: CacheOperations[K, V], repo: K => Stitch[V])

extends (K => Stitch[V]) {

private[this] val stitchOps = new StitchCacheOperations(operations)

override def apply(key: K): Stitch[V] =

stitchOps.get(key).flatMap {

case CacheResult.Fresh(value) =>

Stitch.value(value)

case CacheResult.Stale(staleValue) =>

StitchAsync(repo(key).flatMap(refreshed => stitchOps.set(key, refreshed)))

.map(\_ => staleValue)

case CacheResult.Miss =>

repo(key)

.applyEffect(value => StitchAsync(stitchOps.set(key, value)))

case CacheResult.Failure(\_) =>

// In the case of failure, we don't attempt to write back to

// cache, because cache failure usually means communication

// failure, and sending more requests to the cache that holds

// the value for this key could make the situation worse.

repo(key)

}

}