package com.twitter.search.earlybird\_root;

import javax.inject.Named;

import javax.inject.Singleton;

import com.google.inject.Provides;

import com.twitter.finagle.memcached.JavaClient;

import com.twitter.finagle.mtls.authentication.ServiceIdentifier;

import com.twitter.finagle.stats.StatsReceiver;

import com.twitter.inject.TwitterModule;

import com.twitter.search.common.caching.Cache;

import com.twitter.search.common.caching.EarlybirdCacheSerializer;

import com.twitter.search.common.caching.SearchCacheBuilder;

import com.twitter.search.common.caching.SearchMemcacheClientConfig;

import com.twitter.search.common.caching.SearchMemcacheClientFactory;

import com.twitter.search.earlybird.thrift.EarlybirdRequest;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

import com.twitter.search.earlybird\_root.caching.CacheCommonUtil;

import com.twitter.search.earlybird\_root.caching.CacheStats;

import com.twitter.search.earlybird\_root.caching.DefaultForcedCacheMissDecider;

import com.twitter.search.earlybird\_root.filters.PostCacheRequestTypeCountFilter;

import com.twitter.util.Duration;

/\*\*

\* Provides common bindings for cache related modules.

\*/

public class EarlybirdCacheCommonModule extends TwitterModule {

private static final String CACHE\_VERSION = "1";

@Override

public void configure() {

bind(PostCacheRequestTypeCountFilter.class).in(Singleton.class);

bind(DefaultForcedCacheMissDecider.class).in(Singleton.class);

}

@Provides

@Singleton

@Named(CacheCommonUtil.NAMED\_MAX\_CACHE\_RESULTS)

Integer provideMaxCacheResults() {

return 100;

}

@Provides

@Singleton

JavaClient provideMemCacheClient(

StatsReceiver statsReceiver, ServiceIdentifier serviceIdentifier) {

SearchMemcacheClientConfig config = new SearchMemcacheClientConfig();

config.connectTimeoutMs = Duration.fromMilliseconds(100);

config.requestTimeoutMs = Duration.fromMilliseconds(100);

config.failureAccrualFailuresNumber = 150;

config.failureAccrualFailuresDurationMillis = 30000;

config.failureAccrualDuration = Duration.fromMilliseconds(60000);

return SearchMemcacheClientFactory.createMtlsClient(

"",

"earlybird\_root",

statsReceiver,

config,

serviceIdentifier

);

}

/\*\*

\* Create a new Earlybird cache.

\*

\* @param client the memcache client to use.

\* @param decider the decider to use for the cache.

\* @param cachePrefix the common cache prefix for the cache type.

\* @param serializedKeyPrefix the common cache prefix for the cluster.

\* @param cacheExpiryMillis cache entry ttl in milliseconds.

\*/

static Cache<EarlybirdRequest, EarlybirdResponse> createCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

String cachePrefix,

String serializedKeyPrefix,

long cacheExpiryMillis,

int cacheKeyMaxBytes,

int cacheValueMaxBytes) {

return new SearchCacheBuilder<EarlybirdRequest, EarlybirdResponse>(

CACHE\_VERSION,

client,

cachePrefix,

serializedKeyPrefix,

cacheExpiryMillis)

.withMaxKeyBytes(cacheKeyMaxBytes)

.withMaxValueBytes(cacheValueMaxBytes)

.withRequestTimeoutCounter(CacheStats.REQUEST\_TIMEOUT\_COUNTER)

.withRequestFailedCounter(CacheStats.REQUEST\_FAILED\_COUNTER)

.withCacheSerializer(new EarlybirdCacheSerializer())

.withForceCacheMissDecider(decider)

.withInProcessCache()

.build();

}

}