package com.twitter.search.earlybird\_root;

import javax.inject.Named;

import javax.inject.Singleton;

import com.google.inject.Key;

import com.google.inject.Provides;

import com.twitter.common.util.Clock;

import com.twitter.finagle.memcached.JavaClient;

import com.twitter.inject.TwitterModule;

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

import com.twitter.search.common.decider.SearchDecider;

import com.twitter.search.common.root.LoggingSupport;

import com.twitter.search.common.root.PartitionLoggingSupport;

import com.twitter.search.common.root.SearchRootModule;

import com.twitter.search.common.root.SearchRootWarmup;

import com.twitter.search.common.root.ValidationBehavior;

import com.twitter.search.common.root.WarmupConfig;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

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

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

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

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

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

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

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

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

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

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

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

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

public class RealtimeCgRootAppModule extends TwitterModule {

private static final long RECENCY\_CACHE\_TTL\_MILLIS = 20000L;

private static final long RELEVANCE\_CACHE\_TTL\_MILLIS = 20000L;

private static final long FACETS\_CACHE\_TTL\_MILLIS = 300000L;

private static final long TERMSTATS\_CACHE\_TTL\_MILLIS = 300000L;

@Override

public void configure() {

bind(Key.get(EarlybirdCluster.class)).toInstance(EarlybirdCluster.REALTIME\_CG);

bind(EarlybirdServiceScatterGatherSupport.class)

.to(EarlybirdRealtimeCgScatterGatherSupport.class);

bind(EarlybirdService.ServiceIface.class).to(RealtimeCgRootService.class);

}

@Provides

@Singleton

@RecencyCache

Cache<EarlybirdRequest, EarlybirdResponse> provideRecencyCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, "realtime\_cg\_recency\_root",

serializedKeyPrefix, RECENCY\_CACHE\_TTL\_MILLIS, cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@RelevanceCache

Cache<EarlybirdRequest, EarlybirdResponse> provideRelevanceCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, "realtime\_cg\_relevance\_root",

serializedKeyPrefix, RELEVANCE\_CACHE\_TTL\_MILLIS, cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@StrictRecencyCache

Cache<EarlybirdRequest, EarlybirdResponse> provideStrictRecencyCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(

client, decider, "realtime\_cg\_strict\_recency\_root", serializedKeyPrefix,

RECENCY\_CACHE\_TTL\_MILLIS, cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@FacetsCache

Cache<EarlybirdRequest, EarlybirdResponse> provideFacetsCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, "realtime\_cg\_facets\_root",

serializedKeyPrefix, FACETS\_CACHE\_TTL\_MILLIS, cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@TermStatsCache

Cache<EarlybirdRequest, EarlybirdResponse> provideTermStatsCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, "realtime\_cg\_termstats\_root",

serializedKeyPrefix, TERMSTATS\_CACHE\_TTL\_MILLIS, cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

@Singleton

@TopTweetsCache

Cache<EarlybirdRequest, EarlybirdResponse> provideTopTweetsCache(

JavaClient client,

DefaultForcedCacheMissDecider decider,

@Named(SearchRootModule.NAMED\_SERIALIZED\_KEY\_PREFIX) String serializedKeyPrefix,

@Named(SearchRootModule.NAMED\_CACHE\_KEY\_MAX\_BYTES) int cacheKeyMaxBytes,

@Named(SearchRootModule.NAMED\_CACHE\_VALUE\_MAX\_BYTES) int cacheValueMaxBytes) {

return EarlybirdCacheCommonModule.createCache(client, decider, "realtime\_cg\_toptweets\_root",

serializedKeyPrefix, TopTweetsServicePostProcessor.CACHE\_AGE\_IN\_MS,

cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

SearchRootWarmup<EarlybirdService.ServiceIface, ?, ?> providesSearchRootWarmup(

Clock clock,

WarmupConfig config) {

return new EarlybirdWarmup(clock, config);

}

@Provides

public LoggingSupport<EarlybirdRequest, EarlybirdResponse> provideLoggingSupport(

SearchDecider decider) {

return new EarlybirdServiceLoggingSupport(decider);

}

@Provides

public PartitionLoggingSupport<EarlybirdRequestContext> providePartitionLoggingSupport() {

return new EarlybirdServicePartitionLoggingSupport();

}

@Provides

public ValidationBehavior<EarlybirdRequest, EarlybirdResponse> provideValidationBehavior() {

return new EarlybirdServiceValidationBehavior();

}

}