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.RecencyCache;

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

public class ProtectedRootAppModule extends TwitterModule {

@Override

public void configure() {

bind(Key.get(EarlybirdCluster.class)).toInstance(EarlybirdCluster.PROTECTED);

bind(EarlybirdServiceScatterGatherSupport.class)

.to(EarlybirdProtectedScatterGatherSupport.class);

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

}

@Provides

@Singleton

LoggingSupport<EarlybirdRequest, EarlybirdResponse> provideLoggingSupport(

SearchDecider decider) {

return new EarlybirdServiceLoggingSupport(decider);

}

@Provides

@Singleton

PartitionLoggingSupport<EarlybirdRequestContext> providePartitionLoggingSupport() {

return new EarlybirdServicePartitionLoggingSupport();

}

@Provides

@Singleton

ValidationBehavior<EarlybirdRequest, EarlybirdResponse> providesValidation() {

return new EarlybirdProtectedValidationBehavior();

}

@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\_protected\_recency\_root", serializedKeyPrefix,

20000L, cacheKeyMaxBytes, cacheValueMaxBytes);

}

@Provides

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

Clock clock,

WarmupConfig config) {

return new EarlybirdProtectedWarmup(clock, config);

}

}