package com.twitter.search.earlybird\_root;

import javax.annotation.Nullable;

import javax.inject.Named;

import javax.inject.Singleton;

import scala.PartialFunction;

import com.google.inject.Provides;

import org.apache.thrift.protocol.TProtocolFactory;

import com.twitter.app.Flag;

import com.twitter.app.Flaggable;

import com.twitter.common.util.Clock;

import com.twitter.finagle.Service;

import com.twitter.finagle.mtls.authorization.server.MtlsServerSessionTrackerFilter;

import com.twitter.finagle.service.ReqRep;

import com.twitter.finagle.service.ResponseClass;

import com.twitter.finagle.stats.StatsReceiver;

import com.twitter.finagle.thrift.RichServerParam;

import com.twitter.finagle.thrift.ThriftClientRequest;

import com.twitter.inject.TwitterModule;

import com.twitter.search.common.dark.DarkProxy;

import com.twitter.search.common.dark.ResolverProxy;

import com.twitter.search.common.partitioning.zookeeper.SearchZkClient;

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

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

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

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

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

import com.twitter.search.common.util.zookeeper.ZooKeeperProxy;

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.common.EarlybirdFeatureSchemaMerger;

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

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

/\*\*

\* Provides common bindings.

\*/

public class EarlybirdCommonModule extends TwitterModule {

static final String NAMED\_ALT\_CLIENT = "alt\_client";

static final String NAMED\_EXP\_CLUSTER\_CLIENT = "exp\_cluster\_client";

private final Flag<String> altZkRoleFlag = createFlag(

"alt\_zk\_role",

"",

"The alternative ZooKeeper role",

Flaggable.ofString());

private final Flag<String> altZkClientEnvFlag = createFlag(

"alt\_zk\_client\_env",

"",

"The alternative zk client environment",

Flaggable.ofString());

private final Flag<String> altPartitionZkPathFlag = createFlag(

"alt\_partition\_zk\_path",

"",

"The alternative client partition zk path",

Flaggable.ofString());

@Override

public void configure() {

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

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

bind(Clock.class).toInstance(Clock.SYSTEM\_CLOCK);

bind(QueryLangStatFilter.Config.class).toInstance(new QueryLangStatFilter.Config(100));

}

// Used in SearchRootModule.

@Provides

@Singleton

PartialFunction<ReqRep, ResponseClass> provideResponseClassifier() {

return new RootResponseClassifier();

}

@Provides

@Singleton

Service<byte[], byte[]> providesByteService(

EarlybirdService.ServiceIface svc,

DarkProxy<ThriftClientRequest, byte[]> darkProxy,

TProtocolFactory protocolFactory) {

return darkProxy.toFilter().andThen(

new EarlybirdService.Service(

svc, new RichServerParam(protocolFactory, SearchRootModule.SCROOGE\_BUFFER\_SIZE)));

}

@Provides

@Singleton

@Named(SearchRootModule.NAMED\_SERVICE\_INTERFACE)

Class providesServiceInterface() {

return EarlybirdService.ServiceIface.class;

}

@Provides

@Singleton

ZooKeeperProxy provideZookeeperClient() {

return SearchZkClient.getSZooKeeperClient();

}

@Provides

@Singleton

EarlybirdFeatureSchemaMerger provideFeatureSchemaMerger() {

return new EarlybirdFeatureSchemaMerger();

}

@Provides

@Singleton

@Nullable

@Named(NAMED\_ALT\_CLIENT)

ServerSetsConfig provideAltServerSetsConfig() {

if (!altZkRoleFlag.isDefined() || !altZkClientEnvFlag.isDefined()) {

return null;

}

return new ServerSetsConfig(altZkRoleFlag.apply(), altZkClientEnvFlag.apply());

}

@Provides

@Singleton

@Nullable

@Named(NAMED\_ALT\_CLIENT)

PartitionConfig provideAltPartitionConfig(PartitionConfig defaultPartitionConfig) {

if (!altPartitionZkPathFlag.isDefined()) {

return null;

}

return new PartitionConfig(

defaultPartitionConfig.getNumPartitions(), altPartitionZkPathFlag.apply());

}

@Provides

@Singleton

@Nullable

@Named(NAMED\_ALT\_CLIENT)

RootClientServiceBuilder<EarlybirdService.ServiceIface> provideAltRootClientServiceBuilder(

@Named(NAMED\_ALT\_CLIENT) @Nullable ServerSetsConfig serverSetsConfig,

@Named(SearchRootModule.NAMED\_SERVICE\_INTERFACE) Class serviceIface,

ResolverProxy resolverProxy,

RemoteClientBuilder<EarlybirdService.ServiceIface> remoteClientBuilder) {

if (serverSetsConfig == null) {

return null;

}

return new RootClientServiceBuilder<>(

serverSetsConfig, serviceIface, resolverProxy, remoteClientBuilder);

}

@Provides

@Singleton

@Named(NAMED\_EXP\_CLUSTER\_CLIENT)

RootClientServiceBuilder<EarlybirdService.ServiceIface> provideExpClusterRootClientServiceBuilder(

@Named(SearchRootModule.NAMED\_EXP\_CLUSTER\_SERVER\_SETS\_CONFIG)

ServerSetsConfig serverSetsConfig,

@Named(SearchRootModule.NAMED\_SERVICE\_INTERFACE) Class serviceIface,

ResolverProxy resolverProxy,

RemoteClientBuilder<EarlybirdService.ServiceIface> remoteClientBuilder) {

return new RootClientServiceBuilder<>(

serverSetsConfig, serviceIface, resolverProxy, remoteClientBuilder);

}

@Provides

@Singleton

MtlsServerSessionTrackerFilter<EarlybirdRequest, EarlybirdResponse>

provideMtlsServerSessionTrackerFilter(StatsReceiver statsReceiver) {

return new MtlsServerSessionTrackerFilter<>(statsReceiver);

}

}