package com.twitter.search.ingester.pipeline.wire;

import java.util.List;

import java.util.concurrent.ExecutorService;

import javax.annotation.Nullable;

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import org.apache.kafka.clients.consumer.KafkaConsumer;

import org.apache.kafka.clients.producer.Partitioner;

import org.apache.kafka.common.serialization.Deserializer;

import org.apache.kafka.common.serialization.Serializer;

import org.apache.thrift.TBase;

import com.twitter.common.util.Clock;

import com.twitter.common\_internal.text.version.PenguinVersion;

import com.twitter.decider.Decider;

import com.twitter.eventbus.client.EventBusSubscriber;

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

import com.twitter.finatra.kafka.producers.BlockingFinagleKafkaProducer;

import com.twitter.gizmoduck.thriftjava.UserService;

import com.twitter.metastore.client\_v2.MetastoreClient;

import com.twitter.pink\_floyd.thrift.Storer;

import com.twitter.search.common.partitioning.base.PartitionMappingManager;

import com.twitter.search.common.relevance.classifiers.TweetOffensiveEvaluator;

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

import com.twitter.search.ingester.pipeline.strato\_fetchers.AudioSpaceCoreFetcher;

import com.twitter.search.ingester.pipeline.strato\_fetchers.AudioSpaceParticipantsFetcher;

import com.twitter.search.ingester.pipeline.strato\_fetchers.NamedEntityFetcher;

import com.twitter.search.ingester.pipeline.util.PipelineExceptionHandler;

import com.twitter.storage.client.manhattan.kv.JavaManhattanKVEndpoint;

import com.twitter.tweetypie.thriftjava.TweetService;

import com.twitter.util.Duration;

import com.twitter.util.Function;

import com.twitter.util.Future;

/\*\*

\* An "injection module" that provides bindings for all ingester endpoints that we want to mock out

\* in tests.

\*/

public abstract class WireModule {

/\*\* The JNDI property to which this module will be bound. \*/

private static final String WIRE\_MODULE\_NAME = "";

/\*\* The root name of all properties specified in the twitter-naming-production.\*.xml files. \*/

public static final String JNDI\_PIPELINE\_ROOT = "";

/\*\*

\* (Re)binds the given wire module in JNDI.

\*

\* @param wireModule The wire module to bind in JNDI.

\* @throws NamingException If the wire module cannot be bound in JNDI for some reason.

\*/

public static void bindWireModule(WireModule wireModule) throws NamingException {

Context jndiContext = new InitialContext();

jndiContext.rebind(WIRE\_MODULE\_NAME, wireModule);

}

/\*\*

\* Returns the wire module bound in JNDI.

\*

\* @return The wire module bound in JNDI.

\* @throws NamingException If there's no wire module bound in JNDI.

\*/

public static WireModule getWireModule() throws NamingException {

Context jndiContext = new InitialContext();

return (WireModule) jndiContext.lookup(WIRE\_MODULE\_NAME);

}

/\*\*

\* Retrieves the service identifier needed for making mtls requests.

\* @return The service identifier for the current running service.

\*/

public abstract ServiceIdentifier getServiceIdentifier();

/\*\*

\* Creates a new {@code FinagleKafkaConsumer} with a specified consumer group ID.

\*/

public abstract <T> KafkaConsumer<Long, T> newKafkaConsumer(

String kafkaClusterPath, Deserializer<T> deserializer, String clientId, String groupId,

int maxPollRecords);

/\*\*

\* Creates a new {@code FinagleKafkaConsumer} with a specified consumer group ID.

\*/

public abstract <T> BlockingFinagleKafkaProducer<Long, T> newFinagleKafkaProducer(

String kafkaClusterPath, Serializer<T> serializer, String clientId,

@Nullable Class<? extends Partitioner> partitionerClass);

/\*\*

\* Gets a TweetyPie client.

\*

\* @param tweetypieClientId Use this string as the client id.

\* @return A TweetyPie client

\* @throws NamingException

\*/

public abstract TweetService.ServiceToClient getTweetyPieClient(String tweetypieClientId)

throws NamingException;

/\*\*

\* Gets a Gizmoduck client.

\*

\* @param clientId

\* @throws NamingException

\*/

public abstract UserService.ServiceToClient getGizmoduckClient(String clientId)

throws NamingException;

/\*\*

\* Gets the ManhattanKVEndpoint that should be used for the ManhattanCodedLocationProvider

\*

\* @return the JavaManhattanKVEndpoint that we need for the ManhattanCodedLocationProvider

\* @throws NamingException

\*/

public abstract JavaManhattanKVEndpoint getJavaManhattanKVEndpoint()

throws NamingException;

/\*\*

\* Returns the decider to be used by all stages.

\*

\* @return The decider to be used by all stages.

\*/

public abstract Decider getDecider();

/\*\*

\* Returns the partition ID to be used by all stages.

\*

\* @return The partition ID to be used by all stages.

\*/

public abstract int getPartition();

/\*\*

\* Returns the PipelineExceptionHandler instance to be used by all stages.

\*

\* @return The PipelineExceptionHandler instance to be used by all stages.

\* @throws NamingException If building the PipelineExceptionHandler instance requires some

\* parameters, and those parameters were not bound in JNDI.

\*/

public abstract PipelineExceptionHandler getPipelineExceptionHandler();

/\*\*

\* Gets the PartitionMappingManager for the Kafka writer.

\*

\* @return a PartitionMappingManager

\*/

public abstract PartitionMappingManager getPartitionMappingManager();

/\*\*

\* Returns the Metastore client used by the UserPropertiesManager.

\*

\* @return A Metastore client.

\* @throws NamingException

\*/

public abstract MetastoreClient getMetastoreClient() throws NamingException;

/\*\*

\* Returns an ExecutorService potentially backed by the specified number of threads.

\*

\* @param numThreads An advisory value with a suggestion for how large the threadpool should be.

\* @return an ExecutorService that might be backed by some threads.

\* @throws NamingException

\*/

public abstract ExecutorService getThreadPool(int numThreads) throws NamingException;

/\*\*

\* Returns the Storer interface to connect to Pink.

\*

\* @param requestTimeout The request timeout for the Pink client.

\* @param retries The number of Finagle retries.

\* @return a Storer.ServiceIface to connect to pink.

\*

\*/

public abstract Storer.ServiceIface getStorer(Duration requestTimeout, int retries)

throws NamingException;

/\*\*

\* Returns an EventBusSubscriber

\*/

public abstract <T extends TBase<?, ?>> EventBusSubscriber<T> createEventBusSubscriber(

Function<T, Future<?>> process,

Class<T> thriftStructClass,

String eventBusSubscriberId,

int maxConcurrentEvents);

/\*\*

\* Returns a Clock.

\*/

public abstract Clock getClock();

/\*\*

\* Returns a TweetOffensiveEvaluator.

\*/

public abstract TweetOffensiveEvaluator getTweetOffensiveEvaluator();

/\*\*

\* Returns the cluster.

\*/

public abstract EarlybirdCluster getEarlybirdCluster() throws NamingException;

/\*\*

\* Returns the current penguin version(s).

\*/

public abstract List<PenguinVersion> getPenguinVersions() throws NamingException;

/\*\*

\* Returns updated penguin version(s) depending on decider availability.

\*/

public abstract List<PenguinVersion> getCurrentlyEnabledPenguinVersions();

/\*\*

\* Returns a named entities strato column fetcher.

\*/

public abstract NamedEntityFetcher getNamedEntityFetcher();

/\*\*

\* Returns audio space participants strato column fetcher.

\*/

public abstract AudioSpaceParticipantsFetcher getAudioSpaceParticipantsFetcher();

/\*\*

\* Returns audio space core strato column fetcher.

\*/

public abstract AudioSpaceCoreFetcher getAudioSpaceCoreFetcher();

}