package com.twitter.usersignalservice

package base

import com.twitter.bijection.Codec

import com.twitter.storage.client.manhattan.kv.ManhattanKVClientMtlsParams

import com.twitter.storehaus.ReadableStore

import com.twitter.storehaus\_internal.manhattan.ManhattanCluster

import com.twitter.storehaus\_internal.manhattan.ManhattanRO

import com.twitter.storehaus\_internal.manhattan.ManhattanROConfig

import com.twitter.storehaus\_internal.util.HDFSPath

import com.twitter.twistly.common.UserId

import com.twitter.util.Future

import com.twitter.storehaus\_internal.util.ApplicationID

import com.twitter.storehaus\_internal.util.DatasetName

/\*\*

\* A Manhattan signal fetcher extending BaseSignalFetcher to provide an interface to fetch signals

\* from a Manhattan dataset.

\*

\* Extends this when the underlying store is a single Manhattan dataset.

\* @tparam ManhattanKeyType

\* @tparam ManhattanValueType

\*/

trait ManhattanSignalFetcher[ManhattanKeyType, ManhattanValueType] extends BaseSignalFetcher {

/\*

Define the meta info of the Manhattan dataset

\*/

protected def manhattanAppId: String

protected def manhattanDatasetName: String

protected def manhattanClusterId: ManhattanCluster

protected def manhattanKVClientMtlsParams: ManhattanKVClientMtlsParams

protected def manhattanKeyCodec: Codec[ManhattanKeyType]

protected def manhattanRawSignalCodec: Codec[ManhattanValueType]

/\*\*

\* Adaptor to transform the userId to the ManhattanKey

\* @param userId

\* @return ManhattanKeyType

\*/

protected def toManhattanKey(userId: UserId): ManhattanKeyType

/\*\*

\* Adaptor to transform the ManhattanValue to the Seq of RawSignalType

\* @param manhattanValue

\* @return Seq[RawSignalType]

\*/

protected def toRawSignals(manhattanValue: ManhattanValueType): Seq[RawSignalType]

protected final lazy val underlyingStore: ReadableStore[UserId, Seq[RawSignalType]] = {

ManhattanRO

.getReadableStoreWithMtls[ManhattanKeyType, ManhattanValueType](

ManhattanROConfig(

HDFSPath(""),

ApplicationID(manhattanAppId),

DatasetName(manhattanDatasetName),

manhattanClusterId),

manhattanKVClientMtlsParams

)(manhattanKeyCodec, manhattanRawSignalCodec)

.composeKeyMapping(userId => toManhattanKey(userId))

.mapValues(manhattanRawSignal => toRawSignals(manhattanRawSignal))

}

override final def getRawSignals(userId: UserId): Future[Option[Seq[RawSignalType]]] =

underlyingStore.get(userId)

}