package com.twitter.usersignalservice

package base

import com.twitter.frigate.common.store.strato.StratoFetchableStore

import com.twitter.storehaus.ReadableStore

import com.twitter.strato.client.Client

import com.twitter.strato.data.Conv

import com.twitter.twistly.common.UserId

import com.twitter.util.Future

/\*\*

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

\* Strato Column.

\*

\* Extends this when the underlying store is a single Strato column.

\* @tparam StratoKeyType

\* @tparam StratoViewType

\* @tparam StratoValueType

\*/

trait StratoSignalFetcher[StratoKeyType, StratoViewType, StratoValueType]

extends BaseSignalFetcher {

/\*

Define the meta info of the strato column

\*/

def stratoClient: Client

def stratoColumnPath: String

def stratoView: StratoViewType

/\*\*

\* Override these vals and remove the implicit key words.

\* @return

\*/

protected implicit def keyConv: Conv[StratoKeyType]

protected implicit def viewConv: Conv[StratoViewType]

protected implicit def valueConv: Conv[StratoValueType]

/\*\*

\* Adapter to transform the userId to the StratoKeyType

\* @param userId

\* @return StratoKeyType

\*/

protected def toStratoKey(userId: UserId): StratoKeyType

/\*\*

\* Adapter to transform the StratoValueType to a Seq of RawSignalType

\* @param stratoValue

\* @return Seq[RawSignalType]

\*/

protected def toRawSignals(stratoValue: StratoValueType): Seq[RawSignalType]

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

StratoFetchableStore

.withView[StratoKeyType, StratoViewType, StratoValueType](

stratoClient,

stratoColumnPath,

stratoView)

.composeKeyMapping(toStratoKey)

.mapValues(toRawSignals)

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

underlyingStore.get(userId)

}