package com.twitter.product\_mixer.component\_library.candidate\_source.audiospace

import com.twitter.periscope.audio\_space.thriftscala.CreatedSpacesView

import com.twitter.periscope.audio\_space.thriftscala.SpaceSlice

import com.twitter.product\_mixer.component\_library.model.cursor.NextCursorFeature

import com.twitter.product\_mixer.component\_library.model.cursor.PreviousCursorFeature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.strato.StratoKeyViewFetcherWithSourceFeaturesSource

import com.twitter.product\_mixer.core.model.common.identifier.CandidateSourceIdentifier

import com.twitter.strato.client.Fetcher

import com.twitter.strato.generated.client.periscope.CreatedSpacesSliceOnUserClientColumn

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class CreatedSpacesCandidateSource @Inject() (

column: CreatedSpacesSliceOnUserClientColumn)

extends StratoKeyViewFetcherWithSourceFeaturesSource[

Long,

CreatedSpacesView,

SpaceSlice,

String

] {

override val identifier: CandidateSourceIdentifier = CandidateSourceIdentifier("CreatedSpaces")

override val fetcher: Fetcher[Long, CreatedSpacesView, SpaceSlice] = column.fetcher

override def stratoResultTransformer(

stratoKey: Long,

stratoResult: SpaceSlice

): Seq[String] =

stratoResult.items

override protected def extractFeaturesFromStratoResult(

stratoKey: Long,

stratoResult: SpaceSlice

): FeatureMap = {

val featureMapBuilder = FeatureMapBuilder()

stratoResult.sliceInfo.previousCursor.foreach { cursor =>

featureMapBuilder.add(PreviousCursorFeature, cursor)

}

stratoResult.sliceInfo.nextCursor.foreach { cursor =>

featureMapBuilder.add(NextCursorFeature, cursor)

}

featureMapBuilder.build()

}

}