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

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

import com.twitter.stitch.Stitch

import com.twitter.strato.client.Fetcher

/\*\*

\* A [[CandidateSource]] for getting Candidates from Strato where the

\* Strato column's View is [[StratoView]] and the Value is a [[StratoValue]]

\*

\* A `stratoResultTransformer` must be defined to convert the [[StratoValue]] into a Seq of [[Candidate]]

\*

\* If you need to extract features from the [[StratoValue]] (like a cursor),

\* use [[StratoKeyViewFetcherWithSourceFeaturesSource]] instead.

\*

\* @tparam StratoKey the column's Key type

\* @tparam StratoView the column's View type

\* @tparam StratoValue the column's Value type

\*/

trait StratoKeyViewFetcherSource[StratoKey, StratoView, StratoValue, Candidate]

extends CandidateSource[StratoKeyView[StratoKey, StratoView], Candidate] {

val fetcher: Fetcher[StratoKey, StratoView, StratoValue]

/\*\*

\* Transforms the value type returned by Strato into a Seq[Candidate].

\*

\* This might be as simple as `Seq(stratoResult)` if you're always returning a single candidate.

\*

\* Often, it just extracts a Seq from within a larger wrapper object.

\*

\* If there is global metadata that you need to include, you can zip it with the candidates,

\* returning something like Seq((candiate, metadata), (candidate, metadata)) etc.

\*/

protected def stratoResultTransformer(

stratoKey: StratoKey,

stratoResult: StratoValue

): Seq[Candidate]

override def apply(

request: StratoKeyView[StratoKey, StratoView]

): Stitch[Seq[Candidate]] = {

fetcher

.fetch(request.key, request.view)

.map { result =>

result.v

.map((stratoResult: StratoValue) => stratoResultTransformer(request.key, stratoResult))

.getOrElse(Seq.empty)

}.rescue(StratoErrCategorizer.CategorizeStratoException)

}

}