package com.twitter.product\_mixer.component\_library.selector.sorter.featurestorev1

import com.twitter.ml.featurestore.lib.EntityId

import com.twitter.product\_mixer.component\_library.selector.sorter.Ascending

import com.twitter.product\_mixer.component\_library.selector.sorter.Descending

import com.twitter.product\_mixer.component\_library.selector.sorter.FeatureValueSorter.featureValueSortDefaultValue

import com.twitter.product\_mixer.component\_library.selector.sorter.SorterFromOrdering

import com.twitter.product\_mixer.component\_library.selector.sorter.SorterProvider

import com.twitter.product\_mixer.core.feature.featuremap.featurestorev1.FeatureStoreV1FeatureMap.\_

import com.twitter.product\_mixer.core.feature.featurestorev1.FeatureStoreV1CandidateFeature

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import scala.reflect.runtime.universe.\_

/\*\*

\* Feature Store v1 version of [[com.twitter.product\_mixer.component\_library.selector.sorter.FeatureValueSorter]]

\*/

object FeatureStoreV1FeatureValueSorter {

/\*\*

\* Sort by a Feature Store v1 feature value ascending. If the feature failed or is missing, use an

\* inferred default based on the type of [[FeatureValue]]. For Numeric values this is the MinValue

\* (e.g. Long.MinValue, Double.MinValue).

\*

\* @param feature Feature Store v1 feature with value to sort by

\* @param typeTag allows for inferring default value from the FeatureValue type.

\* See [[com.twitter.product\_mixer.component\_library.selector.sorter.FeatureValueSorter.featureValueSortDefaultValue]]

\* @tparam Candidate candidate for the feature

\* @tparam FeatureValue feature value with an [[Ordering]] context bound

\*/

def ascending[Candidate <: UniversalNoun[Any], FeatureValue: Ordering](

feature: FeatureStoreV1CandidateFeature[PipelineQuery, Candidate, \_ <: EntityId, FeatureValue]

)(

implicit typeTag: TypeTag[FeatureValue]

): SorterProvider = {

val defaultFeatureValue: FeatureValue = featureValueSortDefaultValue(feature, Ascending)

ascending(feature, defaultFeatureValue)

}

/\*\*

\* Sort by a Feature Store v1 feature value ascending. If the feature failed or is missing, use

\* the provided default.

\*

\* @param feature Feature Store v1 feature with value to sort by

\* @tparam Candidate candidate for the feature

\* @tparam FeatureValue feature value with an [[Ordering]] context bound

\*/

def ascending[Candidate <: UniversalNoun[Any], FeatureValue: Ordering](

feature: FeatureStoreV1CandidateFeature[PipelineQuery, Candidate, \_ <: EntityId, FeatureValue],

defaultFeatureValue: FeatureValue

): SorterProvider = {

val ordering = Ordering.by[CandidateWithDetails, FeatureValue](

\_.features.getOrElseFeatureStoreV1CandidateFeature(feature, defaultFeatureValue))

SorterFromOrdering(ordering, Ascending)

}

/\*\*

\* Sort by a Feature Store v1 feature value descending. If the feature failed or is missing, use

\* an inferred default based on the type of [[FeatureValue]]. For Numeric values this is the

\* MaxValue (e.g. Long.MaxValue, Double.MaxValue).

\*

\* @param feature Feature Store v1 feature with value to sort by

\* @param typeTag allows for inferring default value from the FeatureValue type.

\* See [[com.twitter.product\_mixer.component\_library.selector.sorter.FeatureValueSorter.featureValueSortDefaultValue]]

\* @tparam Candidate candidate for the feature

\* @tparam FeatureValue feature value with an [[Ordering]] context bound

\*/

def descending[Candidate <: UniversalNoun[Any], FeatureValue: Ordering](

feature: FeatureStoreV1CandidateFeature[PipelineQuery, Candidate, \_ <: EntityId, FeatureValue]

)(

implicit typeTag: TypeTag[FeatureValue]

): SorterProvider = {

val defaultFeatureValue: FeatureValue = featureValueSortDefaultValue(feature, Descending)

descending(feature, defaultFeatureValue)

}

/\*\*

\* Sort by a Feature Store v1 feature value descending. If the feature failed or is missing, use

\* the provided default.

\*

\* @param feature Feature Store v1 feature with value to sort by

\* @tparam Candidate candidate for the feature

\* @tparam FeatureValue feature value with an [[Ordering]] context bound

\*/

def descending[Candidate <: UniversalNoun[Any], FeatureValue: Ordering](

feature: FeatureStoreV1CandidateFeature[PipelineQuery, Candidate, \_ <: EntityId, FeatureValue],

defaultFeatureValue: FeatureValue

): SorterProvider = {

val ordering = Ordering.by[CandidateWithDetails, FeatureValue](

\_.features.getOrElseFeatureStoreV1CandidateFeature(feature, defaultFeatureValue))

SorterFromOrdering(ordering, Descending)

}

}