package com.twitter.representation\_manager.columns.user

import com.twitter.representation\_manager.columns.ColumnConfigBase

import com.twitter.representation\_manager.store.UserSimClustersEmbeddingStore

import com.twitter.representation\_manager.thriftscala.SimClustersEmbeddingView

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.simclusters\_v2.thriftscala.SimClustersEmbedding

import com.twitter.simclusters\_v2.thriftscala.SimClustersEmbeddingId

import com.twitter.stitch

import com.twitter.stitch.Stitch

import com.twitter.stitch.storehaus.StitchOfReadableStore

import com.twitter.strato.catalog.OpMetadata

import com.twitter.strato.config.AnyOf

import com.twitter.strato.config.ContactInfo

import com.twitter.strato.config.FromColumns

import com.twitter.strato.config.Policy

import com.twitter.strato.config.Prefix

import com.twitter.strato.data.Conv

import com.twitter.strato.data.Description.PlainText

import com.twitter.strato.data.Lifecycle

import com.twitter.strato.fed.\_

import com.twitter.strato.thrift.ScroogeConv

import javax.inject.Inject

class UserSimClustersEmbeddingCol @Inject() (embeddingStore: UserSimClustersEmbeddingStore)

extends StratoFed.Column("recommendations/representation\_manager/simClustersEmbedding.User")

with StratoFed.Fetch.Stitch {

private val storeStitch: SimClustersEmbeddingId => Stitch[SimClustersEmbedding] =

StitchOfReadableStore(embeddingStore.userSimClustersEmbeddingStore.mapValues(\_.toThrift))

val colPermissions: Seq[com.twitter.strato.config.Policy] =

ColumnConfigBase.recosPermissions ++ ColumnConfigBase.externalPermissions :+ FromColumns(

Set(

Prefix("ml/featureStore/simClusters"),

))

override val policy: Policy = AnyOf({

colPermissions

})

override type Key = Long // UserId

override type View = SimClustersEmbeddingView

override type Value = SimClustersEmbedding

override val keyConv: Conv[Key] = Conv.long

override val viewConv: Conv[View] = ScroogeConv.fromStruct[SimClustersEmbeddingView]

override val valueConv: Conv[Value] = ScroogeConv.fromStruct[SimClustersEmbedding]

override val contactInfo: ContactInfo = ColumnConfigBase.contactInfo

override val metadata: OpMetadata = OpMetadata(

lifecycle = Some(Lifecycle.Production),

description = Some(

PlainText("The User SimClusters Embedding Endpoint in Representation Management Service." +

" TDD: http://go/rms-tdd"))

)

override def fetch(key: Key, view: View): Stitch[Result[Value]] = {

val embeddingId = SimClustersEmbeddingId(

view.embeddingType,

view.modelVersion,

InternalId.UserId(key)

)

storeStitch(embeddingId)

.map(embedding => found(embedding))

.handle {

case stitch.NotFound => missing

}

}

}