package com.twitter.tsp.columns

import com.twitter.stitch

import com.twitter.stitch.Stitch

import com.twitter.strato.catalog.OpMetadata

import com.twitter.strato.config.\_

import com.twitter.strato.config.AllowAll

import com.twitter.strato.config.ContactInfo

import com.twitter.strato.config.Policy

import com.twitter.strato.data.Conv

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

import com.twitter.strato.data.Lifecycle.Production

import com.twitter.strato.fed.StratoFed

import com.twitter.strato.thrift.ScroogeConv

import com.twitter.tsp.thriftscala.TopicSocialProofRequest

import com.twitter.tsp.thriftscala.TopicSocialProofResponse

import com.twitter.tsp.service.TopicSocialProofService

import javax.inject.Inject

class TopicSocialProofColumn @Inject() (

topicSocialProofService: TopicSocialProofService)

extends StratoFed.Column(TopicSocialProofColumn.Path)

with StratoFed.Fetch.Stitch {

override type Key = TopicSocialProofRequest

override type View = Unit

override type Value = TopicSocialProofResponse

override val keyConv: Conv[Key] = ScroogeConv.fromStruct[TopicSocialProofRequest]

override val viewConv: Conv[View] = Conv.ofType

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

override val metadata: OpMetadata =

OpMetadata(lifecycle = Some(Production), Some(PlainText("Topic Social Proof Federated Column")))

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

topicSocialProofService

.topicSocialProofHandlerStoreStitch(key)

.map { result => found(result) }

.handle {

case stitch.NotFound => missing

}

}

}

object TopicSocialProofColumn {

val Path = "topic-signals/tsp/topic-social-proof"

}