package com.twitter.follow\_recommendations.common.models

import com.twitter.follow\_recommendations.{thriftscala => t}

import com.twitter.follow\_recommendations.logging.{thriftscala => offline}

case class FollowProof(followedBy: Seq[Long], numIds: Int) {

def toThrift: t.FollowProof = {

t.FollowProof(followedBy, numIds)

}

def toOfflineThrift: offline.FollowProof = offline.FollowProof(followedBy, numIds)

}

object FollowProof {

def fromThrift(proof: t.FollowProof): FollowProof = {

FollowProof(proof.userIds, proof.numIds)

}

}

case class SimilarToProof(similarTo: Seq[Long]) {

def toThrift: t.SimilarToProof = {

t.SimilarToProof(similarTo)

}

def toOfflineThrift: offline.SimilarToProof = offline.SimilarToProof(similarTo)

}

object SimilarToProof {

def fromThrift(proof: t.SimilarToProof): SimilarToProof = {

SimilarToProof(proof.userIds)

}

}

case class PopularInGeoProof(location: String) {

def toThrift: t.PopularInGeoProof = {

t.PopularInGeoProof(location)

}

def toOfflineThrift: offline.PopularInGeoProof = offline.PopularInGeoProof(location)

}

object PopularInGeoProof {

def fromThrift(proof: t.PopularInGeoProof): PopularInGeoProof = {

PopularInGeoProof(proof.location)

}

}

case class TttInterestProof(interestId: Long, interestDisplayName: String) {

def toThrift: t.TttInterestProof = {

t.TttInterestProof(interestId, interestDisplayName)

}

def toOfflineThrift: offline.TttInterestProof =

offline.TttInterestProof(interestId, interestDisplayName)

}

object TttInterestProof {

def fromThrift(proof: t.TttInterestProof): TttInterestProof = {

TttInterestProof(proof.interestId, proof.interestDisplayName)

}

}

case class TopicProof(topicId: Long) {

def toThrift: t.TopicProof = {

t.TopicProof(topicId)

}

def toOfflineThrift: offline.TopicProof =

offline.TopicProof(topicId)

}

object TopicProof {

def fromThrift(proof: t.TopicProof): TopicProof = {

TopicProof(proof.topicId)

}

}

case class CustomInterest(query: String) {

def toThrift: t.CustomInterestProof = {

t.CustomInterestProof(query)

}

def toOfflineThrift: offline.CustomInterestProof =

offline.CustomInterestProof(query)

}

object CustomInterest {

def fromThrift(proof: t.CustomInterestProof): CustomInterest = {

CustomInterest(proof.query)

}

}

case class TweetsAuthorProof(tweetIds: Seq[Long]) {

def toThrift: t.TweetsAuthorProof = {

t.TweetsAuthorProof(tweetIds)

}

def toOfflineThrift: offline.TweetsAuthorProof =

offline.TweetsAuthorProof(tweetIds)

}

object TweetsAuthorProof {

def fromThrift(proof: t.TweetsAuthorProof): TweetsAuthorProof = {

TweetsAuthorProof(proof.tweetIds)

}

}

case class DeviceFollowProof(isDeviceFollow: Boolean) {

def toThrift: t.DeviceFollowProof = {

t.DeviceFollowProof(isDeviceFollow)

}

def toOfflineThrift: offline.DeviceFollowProof =

offline.DeviceFollowProof(isDeviceFollow)

}

object DeviceFollowProof {

def fromThrift(proof: t.DeviceFollowProof): DeviceFollowProof = {

DeviceFollowProof(proof.isDeviceFollow)

}

}

case class AccountProof(

followProof: Option[FollowProof] = None,

similarToProof: Option[SimilarToProof] = None,

popularInGeoProof: Option[PopularInGeoProof] = None,

tttInterestProof: Option[TttInterestProof] = None,

topicProof: Option[TopicProof] = None,

customInterestProof: Option[CustomInterest] = None,

tweetsAuthorProof: Option[TweetsAuthorProof] = None,

deviceFollowProof: Option[DeviceFollowProof] = None) {

def toThrift: t.AccountProof = {

t.AccountProof(

followProof.map(\_.toThrift),

similarToProof.map(\_.toThrift),

popularInGeoProof.map(\_.toThrift),

tttInterestProof.map(\_.toThrift),

topicProof.map(\_.toThrift),

customInterestProof.map(\_.toThrift),

tweetsAuthorProof.map(\_.toThrift),

deviceFollowProof.map(\_.toThrift)

)

}

def toOfflineThrift: offline.AccountProof = {

offline.AccountProof(

followProof.map(\_.toOfflineThrift),

similarToProof.map(\_.toOfflineThrift),

popularInGeoProof.map(\_.toOfflineThrift),

tttInterestProof.map(\_.toOfflineThrift),

topicProof.map(\_.toOfflineThrift),

customInterestProof.map(\_.toOfflineThrift),

tweetsAuthorProof.map(\_.toOfflineThrift),

deviceFollowProof.map(\_.toOfflineThrift)

)

}

}

object AccountProof {

def fromThrift(proof: t.AccountProof): AccountProof = {

AccountProof(

proof.followProof.map(FollowProof.fromThrift),

proof.similarToProof.map(SimilarToProof.fromThrift),

proof.popularInGeoProof.map(PopularInGeoProof.fromThrift),

proof.tttInterestProof.map(TttInterestProof.fromThrift),

proof.topicProof.map(TopicProof.fromThrift),

proof.customInterestProof.map(CustomInterest.fromThrift),

proof.tweetsAuthorProof.map(TweetsAuthorProof.fromThrift),

proof.deviceFollowProof.map(DeviceFollowProof.fromThrift)

)

}

}

case class Reason(accountProof: Option[AccountProof]) {

def toThrift: t.Reason = {

t.Reason(accountProof.map(\_.toThrift))

}

def toOfflineThrift: offline.Reason = {

offline.Reason(accountProof.map(\_.toOfflineThrift))

}

}

object Reason {

def fromThrift(reason: t.Reason): Reason = {

Reason(reason.accountProof.map(AccountProof.fromThrift))

}

}

trait HasReason {

def reason: Option[Reason]

// helper methods below

def followedBy: Option[Seq[Long]] = {

for {

reason <- reason

accountProof <- reason.accountProof

followProof <- accountProof.followProof

} yield { followProof.followedBy }

}

}