package com.twitter.tweetypie

package repository

import com.twitter.servo.util.FutureArrow

import com.twitter.socialgraph.thriftscala.\_

import com.twitter.stitch.SeqGroup

import com.twitter.stitch.Stitch

import com.twitter.stitch.compat.LegacySeqGroup

object RelationshipKey {

def blocks(sourceId: UserId, destinationId: UserId): RelationshipKey =

RelationshipKey(sourceId, destinationId, RelationshipType.Blocking)

def follows(sourceId: UserId, destinationId: UserId): RelationshipKey =

RelationshipKey(sourceId, destinationId, RelationshipType.Following)

def mutes(sourceId: UserId, destinationId: UserId): RelationshipKey =

RelationshipKey(sourceId, destinationId, RelationshipType.Muting)

def reported(sourceId: UserId, destinationId: UserId): RelationshipKey =

RelationshipKey(sourceId, destinationId, RelationshipType.ReportedAsSpam)

}

case class RelationshipKey(

sourceId: UserId,

destinationId: UserId,

relationship: RelationshipType) {

def asExistsRequest: ExistsRequest =

ExistsRequest(

source = sourceId,

target = destinationId,

relationships = Seq(Relationship(relationship))

)

}

object RelationshipRepository {

type Type = RelationshipKey => Stitch[Boolean]

def apply(

exists: FutureArrow[(Seq[ExistsRequest], Option[RequestContext]), Seq[ExistsResult]],

maxRequestSize: Int

): Type = {

val relationshipGroup: SeqGroup[RelationshipKey, Boolean] =

new SeqGroup[RelationshipKey, Boolean] {

override def run(keys: Seq[RelationshipKey]): Future[Seq[Try[Boolean]]] =

LegacySeqGroup.liftToSeqTry(

exists((keys.map(\_.asExistsRequest), None)).map(\_.map(\_.exists)))

override val maxSize: Int = maxRequestSize

}

relationshipKey => Stitch.call(relationshipKey, relationshipGroup)

}

}