package com.twitter.unified\_user\_actions.adapter.social\_graph\_event

import com.twitter.socialgraph.thriftscala.Action

import com.twitter.socialgraph.thriftscala.BlockGraphEvent

import com.twitter.socialgraph.thriftscala.FollowGraphEvent

import com.twitter.socialgraph.thriftscala.MuteGraphEvent

import com.twitter.socialgraph.thriftscala.ReportAsAbuseGraphEvent

import com.twitter.socialgraph.thriftscala.ReportAsSpamGraphEvent

import com.twitter.socialgraph.thriftscala.WriteEvent

import com.twitter.socialgraph.thriftscala.WriteRequestResult

import com.twitter.unified\_user\_actions.thriftscala.{ActionType => UuaActionType}

object SocialGraphEngagement {

/\*\*

\* This is "Follow" event to indicate user1 follows user2 captured in ServerProfileFollow

\*/

object ProfileFollow extends BaseSocialGraphWriteEvent[FollowGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileFollow

override def getSubType(

e: WriteEvent

): Option[Seq[FollowGraphEvent]] =

e.follow

override def getWriteRequestResultFromSubType(

e: Seq[FollowGraphEvent]

): Seq[WriteRequestResult] = {

// Remove all redundant operations (FollowGraphEvent.redundantOperation == Some(true))

e.collect {

case fe if !fe.redundantOperation.getOrElse(false) => fe.result

}

}

}

/\*\*

\* This is "Unfollow" event to indicate user1 unfollows user2 captured in ServerProfileUnfollow

\*

\* Both Unfollow and Follow use the struct FollowGraphEvent, but are treated in its individual case

\* class.

\*/

object ProfileUnfollow extends BaseSocialGraphWriteEvent[FollowGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileUnfollow

override def getSubType(

e: WriteEvent

): Option[Seq[FollowGraphEvent]] =

e.follow

override def getWriteRequestResultFromSubType(

e: Seq[FollowGraphEvent]

): Seq[WriteRequestResult] =

e.collect {

case fe if !fe.redundantOperation.getOrElse(false) => fe.result

}

}

/\*\*

\* This is "Block" event to indicate user1 blocks user2 captured in ServerProfileBlock

\*/

object ProfileBlock extends BaseSocialGraphWriteEvent[BlockGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileBlock

override def getSubType(

e: WriteEvent

): Option[Seq[BlockGraphEvent]] =

e.block

override def getWriteRequestResultFromSubType(

e: Seq[BlockGraphEvent]

): Seq[WriteRequestResult] =

e.map(\_.result)

}

/\*\*

\* This is "Unblock" event to indicate user1 unblocks user2 captured in ServerProfileUnblock

\*

\* Both Unblock and Block use struct BlockGraphEvent, but are treated in its individual case

\* class.

\*/

object ProfileUnblock extends BaseSocialGraphWriteEvent[BlockGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileUnblock

override def getSubType(

e: WriteEvent

): Option[Seq[BlockGraphEvent]] =

e.block

override def getWriteRequestResultFromSubType(

e: Seq[BlockGraphEvent]

): Seq[WriteRequestResult] =

e.map(\_.result)

}

/\*\*

\* This is "Mute" event to indicate user1 mutes user2 captured in ServerProfileMute

\*/

object ProfileMute extends BaseSocialGraphWriteEvent[MuteGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileMute

override def getSubType(

e: WriteEvent

): Option[Seq[MuteGraphEvent]] =

e.mute

override def getWriteRequestResultFromSubType(e: Seq[MuteGraphEvent]): Seq[WriteRequestResult] =

e.map(\_.result)

}

/\*\*

\* This is "Unmute" event to indicate user1 unmutes user2 captured in ServerProfileUnmute

\*

\* Both Unmute and Mute use the struct MuteGraphEvent, but are treated in its individual case

\* class.

\*/

object ProfileUnmute extends BaseSocialGraphWriteEvent[MuteGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileUnmute

override def getSubType(

e: WriteEvent

): Option[Seq[MuteGraphEvent]] =

e.mute

override def getWriteRequestResultFromSubType(e: Seq[MuteGraphEvent]): Seq[WriteRequestResult] =

e.map(\_.result)

}

object ProfileReportAsSpam extends BaseReportSocialGraphWriteEvent[ReportAsSpamGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileReport

override def socialGraphAction: Action = Action.ReportAsSpam

override def getSubType(

e: WriteEvent

): Option[Seq[ReportAsSpamGraphEvent]] =

e.reportAsSpam

override def getWriteRequestResultFromSubType(

e: Seq[ReportAsSpamGraphEvent]

): Seq[WriteRequestResult] =

e.map(\_.result)

}

object ProfileReportAsAbuse extends BaseReportSocialGraphWriteEvent[ReportAsAbuseGraphEvent] {

override def uuaActionType: UuaActionType = UuaActionType.ServerProfileReport

override def socialGraphAction: Action = Action.ReportAsAbuse

override def getSubType(

e: WriteEvent

): Option[Seq[ReportAsAbuseGraphEvent]] =

e.reportAsAbuse

override def getWriteRequestResultFromSubType(

e: Seq[ReportAsAbuseGraphEvent]

): Seq[WriteRequestResult] =

e.map(\_.result)

}

}