package com.twitter.unified\_user\_actions.adapter

import com.twitter.inject.Test

import com.twitter.timelineservice.fanout.thriftscala.FavoriteArchivalEvent

import com.twitter.unified\_user\_actions.adapter.favorite\_archival\_events.FavoriteArchivalEventsAdapter

import com.twitter.unified\_user\_actions.thriftscala.\_

import com.twitter.util.Time

import org.scalatest.prop.TableDrivenPropertyChecks

class FavoriteArchivalEventsAdapterSpec extends Test with TableDrivenPropertyChecks {

trait Fixture {

val frozenTime = Time.fromMilliseconds(1658949273000L)

val userId = 1L

val authorId = 2L

val tweetId = 101L

val retweetId = 102L

val favArchivalEventNoRetweet = FavoriteArchivalEvent(

favoriterId = userId,

tweetId = tweetId,

timestampMs = 0L,

isArchivingAction = Some(true),

tweetUserId = Some(authorId)

)

val favArchivalEventRetweet = FavoriteArchivalEvent(

favoriterId = userId,

tweetId = retweetId,

timestampMs = 0L,

isArchivingAction = Some(true),

tweetUserId = Some(authorId),

sourceTweetId = Some(tweetId)

)

val favUnarchivalEventNoRetweet = FavoriteArchivalEvent(

favoriterId = userId,

tweetId = tweetId,

timestampMs = 0L,

isArchivingAction = Some(false),

tweetUserId = Some(authorId)

)

val favUnarchivalEventRetweet = FavoriteArchivalEvent(

favoriterId = userId,

tweetId = retweetId,

timestampMs = 0L,

isArchivingAction = Some(false),

tweetUserId = Some(authorId),

sourceTweetId = Some(tweetId)

)

val expectedUua1 = UnifiedUserAction(

userIdentifier = UserIdentifier(userId = Some(userId)),

item = Item.TweetInfo(

TweetInfo(

actionTweetId = tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(authorId))),

)

),

actionType = ActionType.ServerTweetArchiveFavorite,

eventMetadata = EventMetadata(

sourceTimestampMs = 0L,

receivedTimestampMs = frozenTime.inMilliseconds,

sourceLineage = SourceLineage.ServerFavoriteArchivalEvents,

)

)

val expectedUua2 = UnifiedUserAction(

userIdentifier = UserIdentifier(userId = Some(userId)),

item = Item.TweetInfo(

TweetInfo(

actionTweetId = retweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(authorId))),

retweetedTweetId = Some(tweetId)

)

),

actionType = ActionType.ServerTweetArchiveFavorite,

eventMetadata = EventMetadata(

sourceTimestampMs = 0L,

receivedTimestampMs = frozenTime.inMilliseconds,

sourceLineage = SourceLineage.ServerFavoriteArchivalEvents,

)

)

val expectedUua3 = UnifiedUserAction(

userIdentifier = UserIdentifier(userId = Some(userId)),

item = Item.TweetInfo(

TweetInfo(

actionTweetId = tweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(authorId))),

)

),

actionType = ActionType.ServerTweetUnarchiveFavorite,

eventMetadata = EventMetadata(

sourceTimestampMs = 0L,

receivedTimestampMs = frozenTime.inMilliseconds,

sourceLineage = SourceLineage.ServerFavoriteArchivalEvents,

)

)

val expectedUua4 = UnifiedUserAction(

userIdentifier = UserIdentifier(userId = Some(userId)),

item = Item.TweetInfo(

TweetInfo(

actionTweetId = retweetId,

actionTweetAuthorInfo = Some(AuthorInfo(authorId = Some(authorId))),

retweetedTweetId = Some(tweetId)

)

),

actionType = ActionType.ServerTweetUnarchiveFavorite,

eventMetadata = EventMetadata(

sourceTimestampMs = 0L,

receivedTimestampMs = frozenTime.inMilliseconds,

sourceLineage = SourceLineage.ServerFavoriteArchivalEvents,

)

)

}

test("all tests") {

new Fixture {

Time.withTimeAt(frozenTime) { \_ =>

val table = Table(

("event", "expected"),

(favArchivalEventNoRetweet, expectedUua1),

(favArchivalEventRetweet, expectedUua2),

(favUnarchivalEventNoRetweet, expectedUua3),

(favUnarchivalEventRetweet, expectedUua4)

)

forEvery(table) { (event: FavoriteArchivalEvent, expected: UnifiedUserAction) =>

val actual = FavoriteArchivalEventsAdapter.adaptEvent(event)

assert(Seq(expected) === actual)

}

}

}

}

}