package com.twitter.home\_mixer.product.scored\_tweets.side\_effect

import com.twitter.finagle.tracing.Trace

import com.twitter.home\_mixer.model.HomeFeatures.AncestorsFeature

import com.twitter.home\_mixer.model.HomeFeatures.AuthorIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.DirectedAtUserIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.EarlybirdScoreFeature

import com.twitter.home\_mixer.model.HomeFeatures.FavoritedByUserIdsFeature

import com.twitter.home\_mixer.model.HomeFeatures.FollowedByUserIdsFeature

import com.twitter.home\_mixer.model.HomeFeatures.FromInNetworkSourceFeature

import com.twitter.home\_mixer.model.HomeFeatures.InReplyToTweetIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.InReplyToUserIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.QuotedTweetIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.QuotedUserIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.RequestJoinIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.ScoreFeature

import com.twitter.home\_mixer.model.HomeFeatures.SuggestTypeFeature

import com.twitter.home\_mixer.param.HomeMixerFlagName.ScribeScoredCandidatesFlag

import com.twitter.home\_mixer.product.scored\_tweets.model.ScoredTweetsQuery

import com.twitter.home\_mixer.product.scored\_tweets.model.ScoredTweetsResponse

import com.twitter.home\_mixer.product.scored\_tweets.param.ScoredTweetsParam.EnableScribeScoredCandidatesParam

import com.twitter.inject.annotations.Flag

import com.twitter.logpipeline.client.common.EventPublisher

import com.twitter.product\_mixer.component\_library.side\_effect.ScribeLogEventSideEffect

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.functional\_component.side\_effect.PipelineResultSideEffect

import com.twitter.product\_mixer.core.model.common.identifier.SideEffectIdentifier

import com.twitter.product\_mixer.core.model.common.presentation.CandidatePipelines

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.timelines.timeline\_logging.{thriftscala => t}

import javax.inject.Inject

import javax.inject.Singleton

import com.twitter.util.logging.Logging

/\*\*

\* Side effect that logs scored candidates from scoring pipelines

\*/

@Singleton

class ScribeScoredCandidatesSideEffect @Inject() (

@Flag(ScribeScoredCandidatesFlag) enableScribeScoredCandidates: Boolean,

eventBusPublisher: EventPublisher[t.ScoredCandidate])

extends ScribeLogEventSideEffect[

t.ScoredCandidate,

ScoredTweetsQuery,

ScoredTweetsResponse

]

with PipelineResultSideEffect.Conditionally[

ScoredTweetsQuery,

ScoredTweetsResponse

]

with Logging {

override val identifier: SideEffectIdentifier =

SideEffectIdentifier("ScribeScoredCandidates")

override def onlyIf(

query: ScoredTweetsQuery,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: ScoredTweetsResponse

): Boolean = enableScribeScoredCandidates && query.params(EnableScribeScoredCandidatesParam)

/\*\*

\* Build the log events from query, selections and response

\*

\* @param query PipelineQuery

\* @param selectedCandidates Result after Selectors are executed

\* @param remainingCandidates Candidates which were not selected

\* @param droppedCandidates Candidates dropped during selection

\* @param response Result after Unmarshalling

\*

\* @return LogEvent in thrift

\*/

override def buildLogEvents(

query: ScoredTweetsQuery,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: ScoredTweetsResponse

): Seq[t.ScoredCandidate] = {

val returned = (selectedCandidates ++ remainingCandidates).map(toThrift(\_, query, false))

val dropped = droppedCandidates.map(toThrift(\_, query, true))

returned ++ dropped

}

private def toThrift(

candidate: CandidateWithDetails,

query: ScoredTweetsQuery,

isDropped: Boolean

): t.ScoredCandidate = {

t.ScoredCandidate(

tweetId = candidate.candidateIdLong,

viewerId = query.getOptionalUserId,

authorId = candidate.features.getOrElse(AuthorIdFeature, None),

traceId = Some(Trace.id.traceId.toLong),

requestJoinId = query.features.flatMap(\_.getOrElse(RequestJoinIdFeature, None)),

score = candidate.features.getOrElse(ScoreFeature, None),

suggestType = candidate.features.getOrElse(SuggestTypeFeature, None).map(\_.name),

isInNetwork = candidate.features.getTry(FromInNetworkSourceFeature).toOption,

inReplyToTweetId = candidate.features.getOrElse(InReplyToTweetIdFeature, None),

inReplyToUserId = candidate.features.getOrElse(InReplyToUserIdFeature, None),

quotedTweetId = candidate.features.getOrElse(QuotedTweetIdFeature, None),

quotedUserId = candidate.features.getOrElse(QuotedUserIdFeature, None),

directedAtUserId = candidate.features.getOrElse(DirectedAtUserIdFeature, None),

favoritedByUserIds = convertSeqFeature(candidate, FavoritedByUserIdsFeature),

followedByUserIds = convertSeqFeature(candidate, FollowedByUserIdsFeature),

ancestors = convertSeqFeature(candidate, AncestorsFeature),

requestTimeMs = Some(query.queryTime.inMilliseconds),

candidatePipelineIdentifier =

candidate.features.getTry(CandidatePipelines).toOption.map(\_.head.name),

earlybirdScore = candidate.features.getOrElse(EarlybirdScoreFeature, None),

isDropped = Some(isDropped)

)

}

private def convertSeqFeature[T](

candidateWithDetails: CandidateWithDetails,

feature: Feature[\_, Seq[T]]

): Option[Seq[T]] =

Option(

candidateWithDetails.features

.getOrElse(feature, Seq.empty)).filter(\_.nonEmpty)

override val logPipelinePublisher: EventPublisher[t.ScoredCandidate] = eventBusPublisher

}