package com.twitter.home\_mixer.functional\_component.feature\_hydrator

import com.twitter.finagle.tracing.Annotation.BinaryAnnotation

import com.twitter.finagle.tracing.ForwardAnnotation

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

import com.twitter.home\_mixer.model.request.DeviceContext.RequestContext

import com.twitter.home\_mixer.model.request.HasDeviceContext

import com.twitter.joinkey.context.RequestJoinKeyContext

import com.twitter.product\_mixer.component\_library.model.cursor.UrtOrderedCursor

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

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.QueryFeatureHydrator

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.operation.BottomCursor

import com.twitter.product\_mixer.core.model.marshalling.response.urt.operation.GapCursor

import com.twitter.product\_mixer.core.model.marshalling.response.urt.operation.TopCursor

import com.twitter.product\_mixer.core.pipeline.HasPipelineCursor

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.BadRequest

import com.twitter.product\_mixer.core.pipeline.pipeline\_failure.PipelineFailure

import com.twitter.search.common.util.lang.ThriftLanguageUtil

import com.twitter.snowflake.id.SnowflakeId

import com.twitter.stitch.Stitch

import com.twitter.timelines.prediction.adapters.request\_context.RequestContextAdapter.dowFromTimestamp

import com.twitter.timelines.prediction.adapters.request\_context.RequestContextAdapter.hourFromTimestamp

import java.util.UUID

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class RequestQueryFeatureHydrator[

Query <: PipelineQuery with HasPipelineCursor[UrtOrderedCursor] with HasDeviceContext] @Inject() (

) extends QueryFeatureHydrator[Query] {

override val features: Set[Feature[\_, \_]] = Set(

AccountAgeFeature,

ClientIdFeature,

DeviceLanguageFeature,

GetInitialFeature,

GetMiddleFeature,

GetNewerFeature,

GetOlderFeature,

GuestIdFeature,

HasDarkRequestFeature,

IsForegroundRequestFeature,

IsLaunchRequestFeature,

PollingFeature,

PullToRefreshFeature,

RequestJoinIdFeature,

ServedRequestIdFeature,

TimestampFeature,

TimestampGMTDowFeature,

TimestampGMTHourFeature,

ViewerIdFeature

)

override val identifier: FeatureHydratorIdentifier = FeatureHydratorIdentifier("Request")

private val DarkRequestAnnotation = "clnt/has\_dark\_request"

// Convert Language code to ISO 639-3 format

private def getLanguageISOFormatByCode(languageCode: String): String =

ThriftLanguageUtil.getLanguageCodeOf(ThriftLanguageUtil.getThriftLanguageOf(languageCode))

private def getRequestJoinId(servedRequestId: Long): Option[Long] =

Some(RequestJoinKeyContext.current.flatMap(\_.requestJoinId).getOrElse(servedRequestId))

private def hasDarkRequest: Option[Boolean] = ForwardAnnotation.current

.getOrElse(Seq[BinaryAnnotation]())

.find(\_.key == DarkRequestAnnotation)

.map(\_.value.asInstanceOf[Boolean])

override def hydrate(query: Query): Stitch[FeatureMap] = {

val requestContext = query.deviceContext.flatMap(\_.requestContextValue)

val servedRequestId = UUID.randomUUID.getMostSignificantBits

val timestamp = query.queryTime.inMilliseconds

val featureMap = FeatureMapBuilder()

.add(AccountAgeFeature, query.getOptionalUserId.flatMap(SnowflakeId.timeFromIdOpt))

.add(ClientIdFeature, query.clientContext.appId)

.add(DeviceLanguageFeature, query.getLanguageCode.map(getLanguageISOFormatByCode))

.add(

GetInitialFeature,

query.pipelineCursor.forall(cursor => cursor.id.isEmpty && cursor.gapBoundaryId.isEmpty))

.add(

GetMiddleFeature,

query.pipelineCursor.exists(cursor =>

cursor.id.isDefined && cursor.gapBoundaryId.isDefined &&

cursor.cursorType.contains(GapCursor)))

.add(

GetNewerFeature,

query.pipelineCursor.exists(cursor =>

cursor.id.isDefined && cursor.gapBoundaryId.isEmpty &&

cursor.cursorType.contains(TopCursor)))

.add(

GetOlderFeature,

query.pipelineCursor.exists(cursor =>

cursor.id.isDefined && cursor.gapBoundaryId.isEmpty &&

cursor.cursorType.contains(BottomCursor)))

.add(GuestIdFeature, query.clientContext.guestId)

.add(IsForegroundRequestFeature, requestContext.contains(RequestContext.Foreground))

.add(IsLaunchRequestFeature, requestContext.contains(RequestContext.Launch))

.add(PollingFeature, query.deviceContext.exists(\_.isPolling.contains(true)))

.add(PullToRefreshFeature, requestContext.contains(RequestContext.PullToRefresh))

.add(ServedRequestIdFeature, Some(servedRequestId))

.add(RequestJoinIdFeature, getRequestJoinId(servedRequestId))

.add(TimestampFeature, timestamp)

.add(TimestampGMTDowFeature, dowFromTimestamp(timestamp))

.add(TimestampGMTHourFeature, hourFromTimestamp(timestamp))

.add(HasDarkRequestFeature, hasDarkRequest)

.add(

ViewerIdFeature,

query.getOptionalUserId

.orElse(query.getGuestId).getOrElse(

throw PipelineFailure(BadRequest, "Missing viewer id")))

.build()

Stitch.value(featureMap)

}

}