package com.twitter.product\_mixer.component\_library.model.candidate.trends\_events

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

import com.twitter.product\_mixer.core.model.common.UniversalNoun

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.event.EventSummaryDisplayType

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.trend.GroupedTrend

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.ImageVariant

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.Url

import com.twitter.product\_mixer.core.model.marshalling.response.urt.promoted.DisclosureType

/\*\*

\* An [[UnifiedTrendEventCandidate]] represents a piece of Event or Trend content.

\* The Event and Trend candidate are represented by different types of keys that Event has a Long

\* eventId while Trend has a String trendName.

\*/

sealed trait UnifiedTrendEventCandidate[+T] extends UniversalNoun[T]

final class UnifiedEventCandidate private (

override val id: Long)

extends UnifiedTrendEventCandidate[Long] {

override def canEqual(that: Any): Boolean = this.isInstanceOf[UnifiedEventCandidate]

override def equals(that: Any): Boolean = {

that match {

case candidate: UnifiedEventCandidate =>

(

(this eq candidate)

|| ((hashCode == candidate.hashCode)

&& (id == candidate.id))

)

case \_ => false

}

}

override val hashCode: Int = id.##

}

object UnifiedEventCandidate {

def apply(id: Long): UnifiedEventCandidate = new UnifiedEventCandidate(id)

}

/\*\*

\* Text description of an Event. Usually this is extracted from curated Event metadata

\*/

object EventTitleFeature extends Feature[UnifiedEventCandidate, String]

/\*\*

\* Display type of an Event. This will be used for client to differentiate if this Event will be

\* displayed as a normal cell, a hero, etc.

\*/

object EventDisplayType extends Feature[UnifiedEventCandidate, EventSummaryDisplayType]

/\*\*

\* URL that servces as the landing page of an Event

\*/

object EventUrl extends Feature[UnifiedEventCandidate, Url]

/\*\*

\* Use to render an Event cell's editorial image

\*/

object EventImage extends Feature[UnifiedEventCandidate, Option[ImageVariant]]

/\*\*

\* Localized time string like "LIVE" or "Last Night" that is used to render the Event cell

\*/

object EventTimeString extends Feature[UnifiedEventCandidate, Option[String]]

final class UnifiedTrendCandidate private (

override val id: String)

extends UnifiedTrendEventCandidate[String] {

override def canEqual(that: Any): Boolean = this.isInstanceOf[UnifiedTrendCandidate]

override def equals(that: Any): Boolean = {

that match {

case candidate: UnifiedTrendCandidate =>

(

(this eq candidate)

|| ((hashCode == candidate.hashCode)

&& (id == candidate.id))

)

case \_ => false

}

}

override val hashCode: Int = id.##

}

object UnifiedTrendCandidate {

def apply(id: String): UnifiedTrendCandidate = new UnifiedTrendCandidate(id)

}

object TrendNormalizedTrendName extends Feature[UnifiedTrendCandidate, String]

object TrendTrendName extends Feature[UnifiedTrendCandidate, String]

object TrendUrl extends Feature[UnifiedTrendCandidate, Url]

object TrendDescription extends Feature[UnifiedTrendCandidate, Option[String]]

object TrendTweetCount extends Feature[UnifiedTrendCandidate, Option[Int]]

object TrendDomainContext extends Feature[UnifiedTrendCandidate, Option[String]]

object TrendGroupedTrends extends Feature[UnifiedTrendCandidate, Option[Seq[GroupedTrend]]]

object PromotedTrendNameFeature extends Feature[UnifiedTrendCandidate, Option[String]]

object PromotedTrendDescriptionFeature extends Feature[UnifiedTrendCandidate, Option[String]]

object PromotedTrendAdvertiserNameFeature extends Feature[UnifiedTrendCandidate, Option[String]]

object PromotedTrendIdFeature extends Feature[UnifiedTrendCandidate, Option[Long]]

object PromotedTrendDisclosureTypeFeature

extends Feature[UnifiedTrendCandidate, Option[DisclosureType]]

object PromotedTrendImpressionIdFeature extends Feature[UnifiedTrendCandidate, Option[String]]