package com.twitter.home\_mixer.model

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

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

import com.twitter.product\_mixer.component\_library.premarshaller.urt.builder.IncludeInstruction

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.item.tweet.TweetItem

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

import com.twitter.timelines.configapi.FSBoundedParam

import com.twitter.timelines.configapi.FSParam

/\*\*

\* Include a clear cache timeline instruction when we satisfy these criteria:

\* - Request Provenance is "pull to refresh"

\* - Atleast N non-ad tweet entries in the response

\*

\* This is to ensure that we have sufficient new content to justify jumping users to the

\* top of the new timelines response and don't add unnecessary load to backend systems

\*/

case class ClearCacheIncludeInstruction(

enableParam: FSParam[Boolean],

minEntriesParam: FSBoundedParam[Int])

extends IncludeInstruction[PipelineQuery with HasDeviceContext] {

override def apply(

query: PipelineQuery with HasDeviceContext,

entries: Seq[TimelineEntry]

): Boolean = {

val enabled = query.params(enableParam)

val ptr =

query.deviceContext.flatMap(\_.requestContextValue).contains(RequestContext.PullToRefresh)

val minTweets = query.params(minEntriesParam) <= entries.collect {

case item: TweetItem if item.promotedMetadata.isEmpty => 1

case module: TimelineModule if module.items.head.item.isInstanceOf[TweetItem] =>

module.items.size

}.sum

enabled && ptr && minTweets

}

}