package com.twitter.tweetypie

package hydrator

import com.twitter.context.thriftscala.Viewer

import com.twitter.featureswitches.FSRecipient

import com.twitter.featureswitches.UserAgent

import com.twitter.featureswitches.v2.FeatureSwitches

import com.twitter.finagle.mtls.authentication.EmptyServiceIdentifier

import com.twitter.strato.callcontext.CallContext

import com.twitter.tweetypie.client\_id.ClientIdHelper

import com.twitter.tweetypie.core.ValueState

/\*\*

\* Hydrate Feature Switch results in TweetData. We can do this once at the

\* start of the hydration pipeline so that the rest of the hydrators can

\* use the Feature Switch values.

\*/

object FeatureSwitchResultsHydrator {

def apply(

featureSwitchesWithoutExperiments: FeatureSwitches,

clientIdHelper: ClientIdHelper

): TweetDataValueHydrator = ValueHydrator.map { (td, opts) =>

val viewer = TwitterContext().getOrElse(Viewer())

val recipient =

FSRecipient(

userId = viewer.userId,

clientApplicationId = viewer.clientApplicationId,

userAgent = viewer.userAgent.flatMap(UserAgent(\_)),

).withCustomFields(

"thrift\_client\_id" ->

clientIdHelper.effectiveClientIdRoot.getOrElse(ClientIdHelper.UnknownClientId),

"forwarded\_service\_id" ->

CallContext.forwardedServiceIdentifier

.map(\_.toString).getOrElse(EmptyServiceIdentifier),

"safety\_level" -> opts.safetyLevel.toString,

"client\_app\_id\_is\_defined" -> viewer.clientApplicationId.isDefined.toString,

)

val results = featureSwitchesWithoutExperiments.matchRecipient(recipient)

ValueState.unit(td.copy(featureSwitchResults = Some(results)))

}

}