package com.twitter.tweetypie

package hydrator

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.FieldByPath

import com.twitter.tweetypie.util.Takedowns

/\*\*

\* Hydrates per-country takedowns which is a union of:

\* 1. per-tweet takedowns, from tweetypieOnlyTakedown{CountryCode|Reasons} fields

\* 2. user takedowns, read from gizmoduck.

\*

\* Note that this hydrator performs backwards compatibility by converting to and from

\* [[com.twitter.tseng.withholding.thriftscala.TakedownReason]]. This is possible because a taken

\* down country code can always be represented as a

\* [[com.twitter.tseng.withholding.thriftscala.UnspecifiedReason]].

\*/

object TakedownHydrator {

type Type = ValueHydrator[Option[Takedowns], Ctx]

case class Ctx(tweetTakedowns: Takedowns, underlyingTweetCtx: TweetCtx) extends TweetCtx.Proxy

val hydratedFields: Set[FieldByPath] =

Set(

fieldByPath(Tweet.TakedownCountryCodesField),

fieldByPath(Tweet.TakedownReasonsField)

)

def apply(repo: UserTakedownRepository.Type): Type =

ValueHydrator[Option[Takedowns], Ctx] { (curr, ctx) =>

repo(ctx.userId).liftToTry.map {

case Return(userReasons) =>

val reasons = Seq.concat(ctx.tweetTakedowns.reasons, userReasons).toSet

ValueState.delta(curr, Some(Takedowns(reasons)))

case Throw(\_) =>

ValueState.partial(curr, hydratedFields)

}

}.onlyIf { (\_, ctx) =>

(

ctx.tweetFieldRequested(Tweet.TakedownCountryCodesField) ||

ctx.tweetFieldRequested(Tweet.TakedownReasonsField)

) && ctx.hasTakedown

}

}