package com.twitter.tweetypie

package hydrator

import com.twitter.tweetypie.core.\_

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.QuotedTweet

/\*\*

\* Enforce that users are not shown quoted tweets where the author of the

\* inner quoted tweet blocks the author of the outer quote tweet or the author

\* of the inner quoted tweet is otherwise not visible to the outer author.

\*

\* In the example below, QuoteTweetVisibilityHydrator checks if @jack

\* blocks @trollmaster.

\*

\* {{{

\* @viewer

\* +------------------------------+

\* | @trollmaster | <-- OUTER QUOTE TWEET

\* | lol u can't spell twitter |

\* | +--------------------------+ |

\* | | @jack | <---- INNER QUOTED TWEET

\* | | just setting up my twttr | |

\* | +--------------------------+ |

\* +------------------------------+

\* }}}

\*

\* In the example below, QuoteTweetVisibilityHydrator checks if @h4x0r can view

\* user @protectedUser.

\*

\* {{{

\* @viewer

\* +------------------------------+

\* | @h4x0r | <-- OUTER QUOTE TWEET

\* | lol nice password |

\* | +--------------------------+ |

\* | | @protectedUser | <---- INNER QUOTED TWEET

\* | | my password is 1234 | |

\* | +--------------------------+ |

\* +------------------------------+

\* }}}

\*

\*

\* In the example below, QuoteTweetVisibilityHydrator checks if @viewer blocks @jack:

\*

\* {{{

\* @viewer

\* +------------------------------+

\* | @sometweeter | <-- OUTER QUOTE TWEET

\* | This is a historic tweet |

\* | +--------------------------+ |

\* | | @jack | <---- INNER QUOTED TWEET

\* | | just setting up my twttr | |

\* | +--------------------------+ |

\* +------------------------------+

\* }}}

\*

\*/

object QuoteTweetVisibilityHydrator {

type Type = ValueHydrator[Option[FilteredState.Unavailable], TweetCtx]

def apply(repo: QuotedTweetVisibilityRepository.Type): QuoteTweetVisibilityHydrator.Type =

ValueHydrator[Option[FilteredState.Unavailable], TweetCtx] { (\_, ctx) =>

val innerTweet: QuotedTweet = ctx.quotedTweet.get

val request = QuotedTweetVisibilityRepository.Request(

outerTweetId = ctx.tweetId,

outerAuthorId = ctx.userId,

innerTweetId = innerTweet.tweetId,

innerAuthorId = innerTweet.userId,

viewerId = ctx.opts.forUserId,

safetyLevel = ctx.opts.safetyLevel

)

repo(request).liftToTry.map {

case Return(Some(f: FilteredState.Unavailable)) =>

ValueState.modified(Some(f))

// For tweet::quotedTweet relationships, all other FilteredStates

// allow the quotedTweet to be hydrated and filtered independently

case Return(\_) =>

ValueState.UnmodifiedNone

// On VF failure, gracefully degrade to no filtering

case Throw(\_) =>

ValueState.UnmodifiedNone

}

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

!ctx.isRetweet &&

ctx.tweetFieldRequested(Tweet.QuotedTweetField) &&

ctx.opts.enforceVisibilityFiltering &&

ctx.quotedTweet.isDefined

}

}