package com.twitter.product\_mixer.core.functional\_component.marshaller.response.urt.metadata

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

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

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

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

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

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

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

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

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

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

import com.twitter.product\_mixer.core.model.marshalling.response.urt.metadata.RichFeedbackBehaviorToggleFollowTopicV2

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

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

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

import com.twitter.timelines.render.{thriftscala => urt}

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class RichFeedbackBehaviorMarshaller @Inject() () {

def apply(richFeedbackBehavior: RichFeedbackBehavior): urt.RichFeedbackBehavior =

richFeedbackBehavior match {

case RichFeedbackBehaviorReportList(listId, userId) =>

urt.RichFeedbackBehavior.ReportList(urt.RichFeedbackBehaviorReportList(listId, userId))

case RichFeedbackBehaviorBlockUser(userId) =>

urt.RichFeedbackBehavior.BlockUser(urt.RichFeedbackBehaviorBlockUser(userId))

case RichFeedbackBehaviorToggleFollowTopic(topicId) =>

urt.RichFeedbackBehavior.ToggleFollowTopic(

urt.RichFeedbackBehaviorToggleFollowTopic(topicId))

case RichFeedbackBehaviorToggleFollowTopicV2(topicId) =>

urt.RichFeedbackBehavior.ToggleFollowTopicV2(

urt.RichFeedbackBehaviorToggleFollowTopicV2(topicId))

case RichFeedbackBehaviorToggleMuteList(listId) =>

urt.RichFeedbackBehavior.ToggleMuteList(urt.RichFeedbackBehaviorToggleMuteList(listId))

case RichFeedbackBehaviorMarkNotInterestedTopic(topicId) =>

urt.RichFeedbackBehavior.MarkNotInterestedTopic(

urt.RichFeedbackBehaviorMarkNotInterestedTopic(topicId))

case RichFeedbackBehaviorReplyPinState(replyPinState) =>

val pinState: urt.ReplyPinState = replyPinState match {

case PinnedReplyPinState => urt.ReplyPinState.Pinned

case PinnableReplyPinState => urt.ReplyPinState.Pinnable

case NotPinnableReplyPinState => urt.ReplyPinState.NotPinnable

}

urt.RichFeedbackBehavior.ReplyPinState(urt.RichFeedbackBehaviorReplyPinState(pinState))

case RichFeedbackBehaviorToggleMuteUser(userId) =>

urt.RichFeedbackBehavior.ToggleMuteUser(urt.RichFeedbackBehaviorToggleMuteUser(userId))

case RichFeedbackBehaviorToggleFollowUser(userId) =>

urt.RichFeedbackBehavior.ToggleFollowUser(urt.RichFeedbackBehaviorToggleFollowUser(userId))

case RichFeedbackBehaviorReportTweet(entryId) =>

urt.RichFeedbackBehavior.ReportTweet(urt.RichFeedbackBehaviorReportTweet(entryId))

}

}