package com.twitter.tweetypie.core

import com.twitter.servo.data.Lens

import com.twitter.tweetypie.Mutation

import com.twitter.tweetypie.thriftscala.Tweet

/\*\*

\* Helper class for building instances of `TweetResult`, which is a type alias

\* for `ValueState[TweetData]`.

\*/

object TweetResult {

object Lenses {

val value: Lens[TweetResult, TweetData] =

Lens[TweetResult, TweetData](\_.value, (r, value) => r.copy(value = value))

val state: Lens[TweetResult, HydrationState] =

Lens[TweetResult, HydrationState](\_.state, (r, state) => r.copy(state = state))

val tweet: Lens[TweetResult, Tweet] = value.andThen(TweetData.Lenses.tweet)

}

def apply(value: TweetData, state: HydrationState = HydrationState.empty): TweetResult =

ValueState(value, state)

def apply(tweet: Tweet): TweetResult =

apply(TweetData(tweet = tweet))

/\*\*

\* Apply this mutation to the tweet contained in the result, updating the modified flag if the mutation modifies the tweet.

\*/

def mutate(mutation: Mutation[Tweet]): TweetResult => TweetResult =

(result: TweetResult) =>

mutation(result.value.tweet) match {

case None => result

case Some(updatedTweet) =>

TweetResult(

result.value.copy(tweet = updatedTweet),

result.state ++ HydrationState.modified

)

}

}