package com.twitter.tweetypie

package handler

import com.twitter.servo.exception.thriftscala.ClientError

import com.twitter.servo.exception.thriftscala.ClientErrorCause

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.FilteredState.Unavailable.\_

private[tweetypie] object HandlerError {

def translateNotFoundToClientError[U](tweetId: TweetId): PartialFunction[Throwable, Stitch[U]] = {

case NotFound =>

Stitch.exception(HandlerError.tweetNotFound(tweetId))

case TweetDeleted | BounceDeleted =>

Stitch.exception(HandlerError.tweetNotFound(tweetId, true))

case SourceTweetNotFound(deleted) =>

Stitch.exception(HandlerError.tweetNotFound(tweetId, deleted))

}

def tweetNotFound(tweetId: TweetId, deleted: Boolean = false): ClientError =

ClientError(

ClientErrorCause.BadRequest,

s"tweet ${if (deleted) "deleted" else "not found"}: $tweetId"

)

def userNotFound(userId: UserId): ClientError =

ClientError(ClientErrorCause.BadRequest, s"user not found: $userId")

def tweetNotFoundException(tweetId: TweetId): Future[Nothing] =

Future.exception(tweetNotFound(tweetId))

def userNotFoundException(userId: UserId): Future[Nothing] =

Future.exception(userNotFound(userId))

def getRequired[A, B](

optionFutureArrow: FutureArrow[A, Option[B]],

notFound: A => Future[B]

): FutureArrow[A, B] =

FutureArrow(key =>

optionFutureArrow(key).flatMap {

case Some(x) => Future.value(x)

case None => notFound(key)

})

}