package com.twitter.tweetypie

package repository

import com.twitter.stitch.NotFound

import com.twitter.stitch.Stitch

import com.twitter.tweetypie.core.\_

object TweetRepository {

type Type = (TweetId, TweetQuery.Options) => Stitch[Tweet]

type Optional = (TweetId, TweetQuery.Options) => Stitch[Option[Tweet]]

def tweetGetter(repo: Optional, opts: TweetQuery.Options): FutureArrow[TweetId, Option[Tweet]] =

FutureArrow(tweetId => Stitch.run(repo(tweetId, opts)))

def tweetGetter(repo: Optional): FutureArrow[(TweetId, TweetQuery.Options), Option[Tweet]] =

FutureArrow { case (tweetId, opts) => Stitch.run(repo(tweetId, opts)) }

/\*\*

\* Converts a `TweetResultRepository.Type`-typed repo to an `TweetRepository.Type`-typed repo.

\*/

def fromTweetResult(repo: TweetResultRepository.Type): Type =

(tweetId, options) => repo(tweetId, options).map(\_.value.tweet)

/\*\*

\* Converts a `Type`-typed repo to an `Optional`-typed

\* repo, where NotFound or filtered tweets are returned as `None`.

\*/

def optional(repo: Type): Optional =

(tweetId, options) =>

repo(tweetId, options).liftToOption { case NotFound | (\_: FilteredState) => true }

}