package com.twitter.tweetypie

package hydrator

import com.twitter.mediaservices.commons.thriftscala.MediaKey

import com.twitter.tweetypie.repository.\_

import com.twitter.tweetypie.thriftscala.\_

object MediaEntitiesHydrator {

object Cacheable {

type Ctx = MediaEntityHydrator.Cacheable.Ctx

type Type = ValueHydrator[Seq[MediaEntity], Ctx]

def once(h: MediaEntityHydrator.Cacheable.Type): Type =

TweetHydration.completeOnlyOnce(

queryFilter = MediaEntityHydrator.queryFilter,

hydrationType = HydrationType.CacheableMedia,

dependsOn = Set(HydrationType.Urls),

hydrator = h.liftSeq

)

}

object Uncacheable {

type Ctx = MediaEntityHydrator.Uncacheable.Ctx

type Type = ValueHydrator[Seq[MediaEntity], Ctx]

}

}

object MediaEntityHydrator {

val hydratedField: FieldByPath = fieldByPath(Tweet.MediaField)

object Cacheable {

type Type = ValueHydrator[MediaEntity, Ctx]

case class Ctx(urlEntities: Seq[UrlEntity], underlyingTweetCtx: TweetCtx) extends TweetCtx.Proxy

/\*\*

\* Builds a single media-hydrator out of finer-grained hydrators

\* only with cacheable information.

\*/

def apply(hydrateMediaUrls: Type, hydrateMediaIsProtected: Type): Type =

hydrateMediaUrls.andThen(hydrateMediaIsProtected)

}

object Uncacheable {

type Type = ValueHydrator[MediaEntity, Ctx]

case class Ctx(mediaKeys: Option[Seq[MediaKey]], underlyingTweetCtx: TweetCtx)

extends TweetCtx.Proxy {

def includeMediaEntities: Boolean = tweetFieldRequested(Tweet.MediaField)

def includeAdditionalMetadata: Boolean =

opts.include.mediaFields.contains(MediaEntity.AdditionalMetadataField.id)

}

/\*\*

\* Builds a single media-hydrator out of finer-grained hydrators

\* only with uncacheable information.

\*/

def apply(hydrateMediaKey: Type, hydrateMediaInfo: Type): Type =

(hydrateMediaKey

.andThen(hydrateMediaInfo))

.onlyIf((\_, ctx) => ctx.includeMediaEntities)

}

def queryFilter(opts: TweetQuery.Options): Boolean =

opts.include.tweetFields.contains(Tweet.MediaField.id)

}