package com.twitter.tweetypie.core

import com.twitter.servo.data.Mutation

/\*\*

\* An EditState is a function that changes a value and may generate

\* some state about what was modified. For instance, it may record

\* whether an item was changed, or whether there was an error.

\* EditStates are useful because they are first-class values that can

\* be composed. In particular, it is useful to concurrently access

\* external data to build edits and then apply them.

\*

\* @tparam A The type of the value that is being edited (for instance,

\* having fields hydrated with data from another service)

\*/

final case class EditState[A](run: A => ValueState[A]) {

/\*\*

\* Composes two EditStates in sequence

\*/

def andThen(other: EditState[A]): EditState[A] =

EditState[A] { a0: A =>

val ValueState(a1, s1) = run(a0)

val ValueState(a2, s2) = other.run(a1)

ValueState(a2, s1 ++ s2)

}

}

object EditState {

/\*\*

\* Creates a "passthrough" EditState:

\* Leaves A unchanged and produces empty state S

\*/

def unit[A]: EditState[A] =

EditState[A](ValueState.unit[A])

/\*\*

\* Creates an `EditState[A]` using a `Mutation[A]`.

\*/

def fromMutation[A](mut: Mutation[A]): EditState[A] =

EditState[A] { a =>

mut(a) match {

case None => ValueState.unmodified(a)

case Some(a2) => ValueState.modified(a2)

}

}

}