package com.twitter.search.earlybird.util;

import scala.Some;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.base.Preconditions;

import com.twitter.decider.Decider;

import com.twitter.decider.Decider$;

import com.twitter.decider.RandomRecipient$;

import com.twitter.decider.Recipient;

import com.twitter.decider.decisionmaker.MutableDecisionMaker;

import com.twitter.search.common.decider.DeciderUtil;

import com.twitter.search.common.decider.SearchDeciderFactory;

import com.twitter.search.earlybird.common.config.EarlybirdProperty;

/\*\*

\* A Singleton to let any code in Earlybird have the ability to be guarded by a decider key.

\*

\* EarlybirdDecider is a thin wrapper around the Twitter Decider library to provide global access to a single

\* decider configuration. This way any code anywhere can easily be guarded by a Decider key. The initializer requires

\* EarlybirdConfig to be initialized already. Defaults to a NullDecider, which causes all requests for keys to return

\* false.

\*/

public final class EarlybirdDecider {

public static final org.slf4j.Logger LOG =

org.slf4j.LoggerFactory.getLogger(EarlybirdDecider.class);

public static final String DECIDER\_CONFIG = "./config/earlybird-decider.yml";

private static volatile Decider earlybirdDecider = Decider$.MODULE$.NullDecider();

private static volatile MutableDecisionMaker mutableDecisionMaker;

private EarlybirdDecider() { }

/\*\*

\* Initializes the global decider accessor. Requires EarlybirdConfig to be initialized.

\*

\* @return the new decider interface.

\*/

public static Decider initialize() {

return initialize(DECIDER\_CONFIG);

}

/\*\*

\* Initializes the global decider accessor. Requires EarlybirdConfig to be initialized.

\*

\* @param configPath path to the base decider config file.

\* @return the new decider interface.

\*/

@VisibleForTesting public static Decider initialize(String configPath) {

synchronized (EarlybirdDecider.class) {

Preconditions.checkState(earlybirdDecider == Decider$.MODULE$.NullDecider(),

"EarlybirdDecider can be initialized only once.");

mutableDecisionMaker = new MutableDecisionMaker();

if (EarlybirdProperty.USE\_DECIDER\_OVERLAY.get(false)) {

String category = EarlybirdProperty.DECIDER\_OVERLAY\_CONFIG.get();

earlybirdDecider =

SearchDeciderFactory.createDeciderWithoutRefreshBaseWithOverlay(

configPath, category, mutableDecisionMaker);

LOG.info("EarlybirdDecider set to use the decider overlay " + category);

} else {

earlybirdDecider =

SearchDeciderFactory.createDeciderWithRefreshBaseWithoutOverlay(

configPath, mutableDecisionMaker);

LOG.info("EarlybirdDecider set to only use the base config");

}

return earlybirdDecider;

}

}

/\*\*

\* Check if feature is available based on randomness

\*

\* @param feature the feature name to test

\* @return true if the feature is available, false otherwise

\*/

public static boolean isFeatureAvailable(String feature) {

return isFeatureAvailable(feature, RandomRecipient$.MODULE$);

}

/\*\*

\* Check if the feature is available based on the user

\*

\* The recipient'd id is hashed and used as the value to compare with the decider percentage. Therefore, the same user

\* will always get the same result for a given percentage, and higher percentages should always be a superset of the

\* lower percentage users.

\*

\* RandomRecipient can be used to get a random value for every call.

\*

\* @param feature the feature name to test

\* @param recipient the recipient to base a decision on

\* @return true if the feature is available, false otherwise

\*/

public static boolean isFeatureAvailable(String feature, Recipient recipient) {

if (earlybirdDecider == Decider$.MODULE$.NullDecider()) {

LOG.warn("EarlybirdDecider is uninitialized but requested feature " + feature);

}

return earlybirdDecider.isAvailable(feature, Some.apply(recipient));

}

/\*\*

\* Get the raw decider value for a given feature.

\*

\* @param feature the feature name

\* @return the integer value of the decider

\*/

public static int getAvailability(String feature) {

return DeciderUtil.getAvailability(earlybirdDecider, feature);

}

public static Decider getDecider() {

checkInitialized();

return earlybirdDecider;

}

public static MutableDecisionMaker getMutableDecisionMaker() {

checkInitialized();

return mutableDecisionMaker;

}

private static void checkInitialized() {

Preconditions.checkState(earlybirdDecider != Decider$.MODULE$.NullDecider(),

"EarlybirdDecider is not initialized.");

}

}