package com.twitter.frigate.pushservice.model.ibis

import com.twitter.frigate.common.base.MagicFanoutProductLaunchCandidate

import com.twitter.frigate.magic\_events.thriftscala.ProductInfo

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.params.PushFeatureSwitchParams

import com.twitter.frigate.pushservice.util.PushIbisUtil.mergeModelValues

import com.twitter.util.Future

trait MagicFanoutProductLaunchIbis2Hydrator

extends CustomConfigurationMapForIbis

with Ibis2HydratorForCandidate {

self: PushCandidate with MagicFanoutProductLaunchCandidate =>

private def getProductInfoMap(productInfo: ProductInfo): Map[String, String] = {

val titleMap = productInfo.title

.map { title =>

Map("title" -> title)

}.getOrElse(Map.empty)

val bodyMap = productInfo.body

.map { body =>

Map("body" -> body)

}.getOrElse(Map.empty)

val deeplinkMap = productInfo.deeplink

.map { deeplink =>

Map("deeplink" -> deeplink)

}.getOrElse(Map.empty)

titleMap ++ bodyMap ++ deeplinkMap

}

private lazy val landingPage: Map[String, String] = {

val urlFromFS = target.params(PushFeatureSwitchParams.ProductLaunchLandingPageDeepLink)

Map("push\_land\_url" -> urlFromFS)

}

private lazy val customProductLaunchPushDetails: Map[String, String] = {

frigateNotification.magicFanoutProductLaunchNotification match {

case Some(productLaunchNotif) =>

productLaunchNotif.productInfo match {

case Some(productInfo) =>

getProductInfoMap(productInfo)

case \_ => Map.empty

}

case \_ => Map.empty

}

}

override lazy val customFieldsMapFut: Future[Map[String, String]] =

mergeModelValues(super.customFieldsMapFut, customProductLaunchPushDetails)

override lazy val modelValues: Future[Map[String, String]] =

mergeModelValues(super.modelValues, landingPage)

}