package com.twitter.follow\_recommendations.configapi.candidates

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.models.CandidateUser

import com.twitter.follow\_recommendations.common.models.HasDisplayLocation

import com.twitter.follow\_recommendations.configapi.params.GlobalParams

import com.twitter.servo.util.MemoizingStatsReceiver

import com.twitter.timelines.configapi.Config

import com.twitter.timelines.configapi.HasParams

import com.twitter.timelines.configapi.Params

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* CandidateParamsFactory is primarily used for "producer side" experiments, don't use it on consumer side experiments

\*/

@Singleton

class CandidateUserParamsFactory[T <: HasParams with HasDisplayLocation] @Inject() (

config: Config,

candidateContextFactory: CandidateUserContextFactory,

statsReceiver: StatsReceiver) {

private val stats = new MemoizingStatsReceiver(statsReceiver.scope("configapi\_candidate\_params"))

def apply(candidateContext: CandidateUser, request: T): CandidateUser = {

if (candidateContext.params == Params.Invalid) {

if (request.params(GlobalParams.EnableCandidateParamHydrations)) {

candidateContext.copy(params =

config(candidateContextFactory(candidateContext, request.displayLocation), stats))

} else {

candidateContext.copy(params = Params.Empty)

}

} else {

candidateContext

}

}

}