package com.twitter.home\_mixer.module

import com.google.inject.Provides

import com.twitter.clientapp.{thriftscala => ca}

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.CandidateFeaturesScribeEventPublisher

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.CommonFeaturesScribeEventPublisher

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.MinimumFeaturesScribeEventPublisher

import com.twitter.inject.TwitterModule

import com.twitter.logpipeline.client.EventPublisherManager

import com.twitter.logpipeline.client.common.EventPublisher

import com.twitter.logpipeline.client.serializers.EventLogMsgTBinarySerializer

import com.twitter.logpipeline.client.serializers.EventLogMsgThriftStructSerializer

import com.twitter.timelines.suggests.common.poly\_data\_record.{thriftjava => pldr}

import com.twitter.timelines.timeline\_logging.{thriftscala => tl}

import javax.inject.Named

import javax.inject.Singleton

object ScribeEventPublisherModule extends TwitterModule {

val ClientEventLogCategory = "client\_event"

val ServedCandidatesLogCategory = "home\_timeline\_served\_candidates\_flattened"

val ScoredCandidatesLogCategory = "home\_timeline\_scored\_candidates"

val ServedCommonFeaturesLogCategory = "tq\_served\_common\_features\_offline"

val ServedCandidateFeaturesLogCategory = "tq\_served\_candidate\_features\_offline"

val ServedMinimumFeaturesLogCategory = "tq\_served\_minimum\_features\_offline"

@Provides

@Singleton

def providesClientEventsScribeEventPublisher: EventPublisher[ca.LogEvent] = {

val serializer = EventLogMsgThriftStructSerializer.getNewSerializer[ca.LogEvent]()

EventPublisherManager.buildScribeLogPipelinePublisher(ClientEventLogCategory, serializer)

}

@Provides

@Singleton

@Named(CommonFeaturesScribeEventPublisher)

def providesCommonFeaturesScribeEventPublisher: EventPublisher[pldr.PolyDataRecord] = {

val serializer = EventLogMsgTBinarySerializer.getNewSerializer

EventPublisherManager.buildScribeLogPipelinePublisher(

ServedCommonFeaturesLogCategory,

serializer)

}

@Provides

@Singleton

@Named(CandidateFeaturesScribeEventPublisher)

def providesCandidateFeaturesScribeEventPublisher: EventPublisher[pldr.PolyDataRecord] = {

val serializer = EventLogMsgTBinarySerializer.getNewSerializer

EventPublisherManager.buildScribeLogPipelinePublisher(

ServedCandidateFeaturesLogCategory,

serializer)

}

@Provides

@Singleton

@Named(MinimumFeaturesScribeEventPublisher)

def providesMinimumFeaturesScribeEventPublisher: EventPublisher[pldr.PolyDataRecord] = {

val serializer = EventLogMsgTBinarySerializer.getNewSerializer

EventPublisherManager.buildScribeLogPipelinePublisher(

ServedMinimumFeaturesLogCategory,

serializer)

}

@Provides

@Singleton

def providesServedCandidatesScribeEventPublisher: EventPublisher[tl.ServedEntry] = {

val serializer = EventLogMsgThriftStructSerializer.getNewSerializer[tl.ServedEntry]()

EventPublisherManager.buildScribeLogPipelinePublisher(ServedCandidatesLogCategory, serializer)

}

@Provides

@Singleton

def provideScoredCandidatesScribeEventPublisher: EventPublisher[tl.ScoredCandidate] = {

val serializer = EventLogMsgThriftStructSerializer.getNewSerializer[tl.ScoredCandidate]()

EventPublisherManager.buildScribeLogPipelinePublisher(ScoredCandidatesLogCategory, serializer)

}

}