package com.twitter.product\_mixer.core.service.transformer\_executor

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.product\_mixer.core.functional\_component.transformer.Transformer

import com.twitter.product\_mixer.core.service.Executor

import com.twitter.stitch.Arrow

import com.twitter.util.Try

import javax.inject.Inject

import javax.inject.Singleton

/\*\*

\* For wrapping [[Transformer]]s that are applied per-candidate

\*

\* Records a single span for running all the components,

\* but stats per-component.

\*/

@Singleton

class PerCandidateTransformerExecutor @Inject() (override val statsReceiver: StatsReceiver)

extends Executor {

def arrow[In, Out](

transformer: Transformer[In, Out],

context: Executor.Context,

): Arrow[Seq[In], Seq[Try[Out]]] = {

val perCandidateArrow = wrapPerCandidateComponentWithExecutorBookkeepingWithoutTracing(

context,

transformer.identifier

)(Arrow.map(transformer.transform)).liftToTry

wrapComponentsWithTracingOnly(

context,

transformer.identifier

)(Arrow.sequence(perCandidateArrow))

}

}