package com.twitter.product\_mixer.component\_library.premarshaller.urp

import com.twitter.product\_mixer.component\_library.premarshaller.urp.builder.PageBodyBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urp.builder.PageHeaderBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urp.builder.PageNavBarBuilder

import com.twitter.product\_mixer.component\_library.premarshaller.urp.builder.TimelineScribeConfigBuilder

import com.twitter.product\_mixer.core.functional\_component.premarshaller.DomainMarshaller

import com.twitter.product\_mixer.core.model.common.identifier.DomainMarshallerIdentifier

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.model.marshalling.response.urp.\_

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

object UrpDomainMarshaller {

val PageIdSuffix = "-Page"

}

/\*\*

\* Domain marshaller that given the builders for the body, header and navbar will generate a URP Page

\*

\* @param pageBodyBuilder PageBody builder that generates a PageBody with the query and selections

\* @param scribeConfigBuilder Scribe Config builder that generates the configuration for scribing of the page

\* @param pageHeaderBuilder PageHeader builder that generates a PageHeader with the query and selections

\* @param pageNavBarBuilder PageNavBar builder that generates a PageNavBar with the query and selections

\* @tparam Query The type of Query that this Marshaller operates with

\*/

case class UrpDomainMarshaller[-Query <: PipelineQuery](

pageBodyBuilder: PageBodyBuilder[Query],

pageHeaderBuilder: Option[PageHeaderBuilder[Query]] = None,

pageNavBarBuilder: Option[PageNavBarBuilder[Query]] = None,

scribeConfigBuilder: Option[TimelineScribeConfigBuilder[Query]] = None,

override val identifier: DomainMarshallerIdentifier =

DomainMarshallerIdentifier("UnifiedRichPage"))

extends DomainMarshaller[Query, Page] {

override def apply(

query: Query,

selections: Seq[CandidateWithDetails]

): Page = {

val pageBody = pageBodyBuilder.build(query, selections)

val pageHeader = pageHeaderBuilder.flatMap(\_.build(query, selections))

val pageNavBar = pageNavBarBuilder.flatMap(\_.build(query, selections))

val scribeConfig = scribeConfigBuilder.flatMap(\_.build(query, pageBody, pageHeader, pageNavBar))

Page(

id = query.product.identifier.toString + UrpDomainMarshaller.PageIdSuffix,

pageBody = pageBody,

scribeConfig = scribeConfig,

pageHeader = pageHeader,

pageNavBar = pageNavBar

)

}

}