package com.twitter.product\_mixer.core.functional\_component.marshaller

import com.twitter.product\_mixer.core.model.common.Component

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

import com.twitter.product\_mixer.core.model.marshalling.HasMarshalling

object TransportMarshaller {

/\*\* Avoid `malformed class name` exceptions due to the presence of the `$` character \*/

def getSimpleName[T](c: Class[T]): String = {

c.getName.lastIndexOf("$") match {

case -1 => c.getSimpleName

case index => c.getName.substring(index + 1)

}

}

}

/\*\*

\* Marshals a [[MarshallerInput]] into a type that can be sent over the wire

\*

\* This transformation should be mechanical and not contain business logic

\*

\* @note this is different from `com.twitter.product\_mixer.core.functional\_component.premarshaller`

\* which can contain business logic.

\*/

trait TransportMarshaller[-MarshallerInput <: HasMarshalling, +MarshallerOutput] extends Component {

override val identifier: TransportMarshallerIdentifier

def apply(input: MarshallerInput): MarshallerOutput

}

/\*\*

\* No op marshalling that passes through a [[HasMarshalling]] into any type. This is useful if

\* the response does not need to be sent over the wire, such as with a

\* [[com.twitter.product\_mixer.core.functional\_component.candidate\_source.product\_pipeline.ProductPipelineCandidateSource]]

\*/

object NoOpTransportMarshaller extends TransportMarshaller[HasMarshalling, Any] {

override val identifier: TransportMarshallerIdentifier = TransportMarshallerIdentifier("NoOp")

override def apply(input: HasMarshalling): Any = input

}