AYI ACADEMY: TALEND

Material práctico

Módulo 9: REST Web Services

Objetivos

Después de completar este módulo, ud podrá:

- Diseñar un REST web service y usar componentes dedicados para REST.
- Agregar mas de un endpoint a un REST service
- Asignar varios HTTP verbs a un mismo URI endpoint
- Consumir un REST service

Ejercicio

Con esta guía de ejercicios podrás diseñar Talend Jobs que permitan exponer un web services REST. Mientras que el protocolo SOAP está orientado a las operaciones, REST

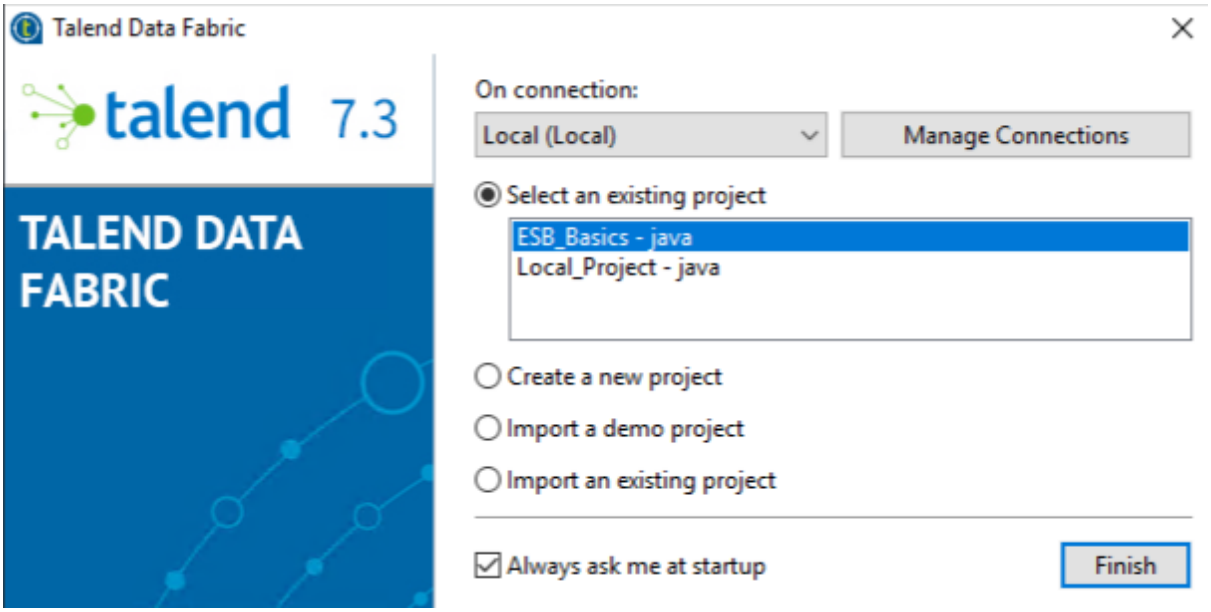
está orientado a los recursos y aprovecha los métodos HTTP (“verbs”) GET, PUT, POST, and DELETE. A diferencia de SOAP, REST no requiere un archivo de contrato.

Este ejercicio muestra cómo crear un web services REST que accede a un recurso usando diferentes URIs y diferentes HTTP verbs.

Opening a project

Inicie su Talend Studio (para esta práctica la version 7.1)

Cree el proyecto ESB_Basics. Si ya existe, abralo.

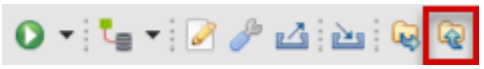


Configurando el proyecto

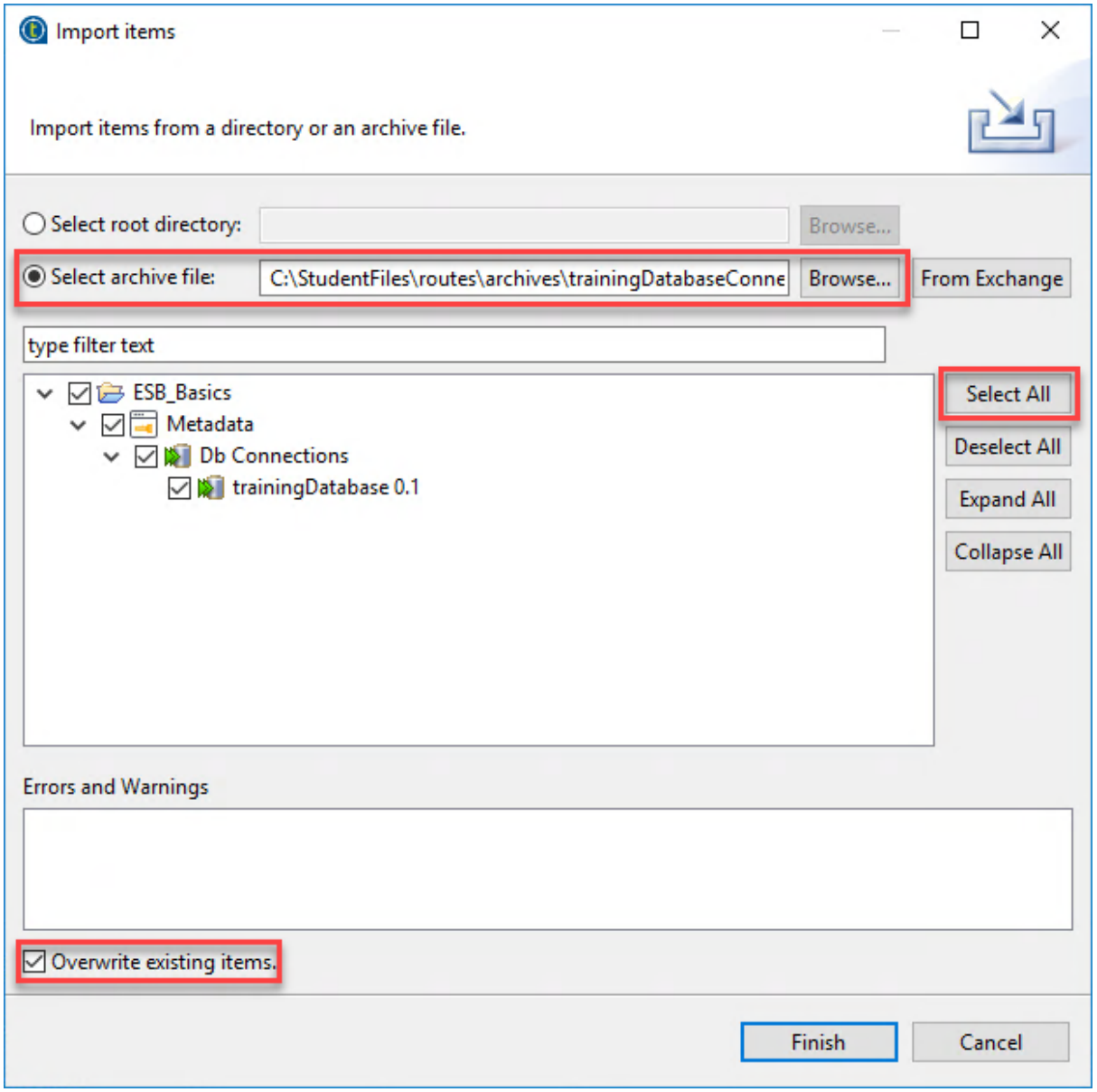
Aesegurese de que la perspectiva **Integration** este seleccionada.

Cargue el archivo trainingDatabaseConnection.zip que se encuentra en la carpeta C:\StudentFiles\routes\archives. Carguelo para importar la metadata de la base de datos al repositorio y así manejar fácilmente el schema de la conexión a MySQL.

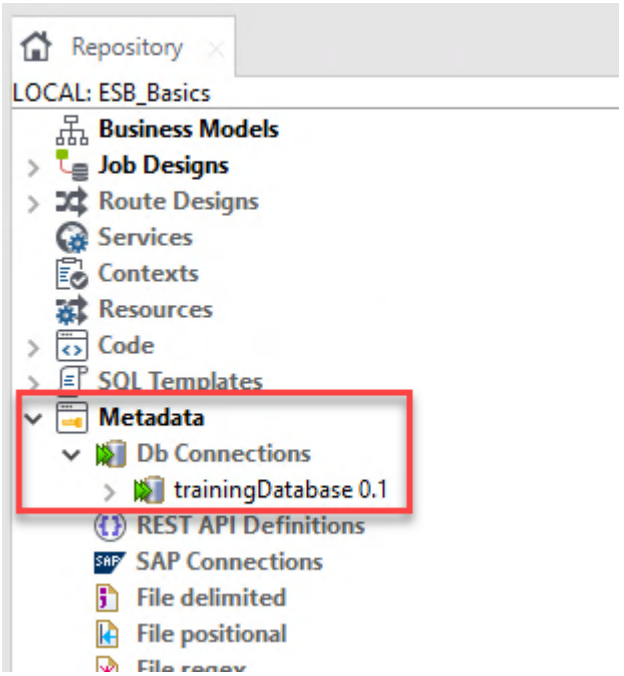
a. En la barra de herramientas **Quick Access** que se encuentra sobre el area de trabajo, haga click en el icono de la carpeta y la flecha hacia arriba.



b. En la ventana de importar items, selecciones el radio button **Select archive file**, luego hacer clicl en el boton **Browse**.
c. Navegue hasta **C:\StudentFiles\routes\archives** y seleccione el archivo **trainingDatabaseConnection.zip**.
d. Click sobre el boton **Select All**.
e. Asegúrese de seleccionar **Overwrite existing items**, luego click en **Finish**.



Con esto has importado la metada de la conexion a BD en el repositorio.



Para ver la imporación puedes ir a Repository > Metadata > Db Connections.

RECUERDEN EDITAR LA CONEXIÓN PARA ADAPTARLA A SU BASE.

Tarea

Este ejecicio muestra cómo crear un simple web services REST.

A diferencia del estándar SOAP, un web services REST no necesita contrato, ya que el estándar REST define todas las operaciones que puede invocar en un recurso.

Como hemos aprendido, un recurso es expuesto en una URI. Para instancia a un servicio REST, debes proporcionar esta URI junto con el HTTP verb. Cada HTTP verb representa una posible operación con la que tú puedes ejecutar el recurso. Los cuatro HTTP verbs con:

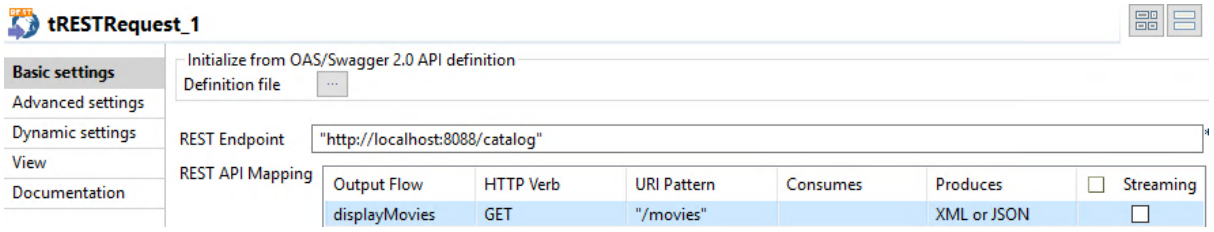
- GET—ejecuta una operación READ en el recurso.
- POST—ejecuta una operación CREATE en el recurso.
- PUT—ejecuta una operación UPDATE en el recurso.
- DELETE—ejecuta una operación DELETE en el recurso.

In estes ejercicio desarrollaremos un servicio REST.

Desarrollando un servicio REST

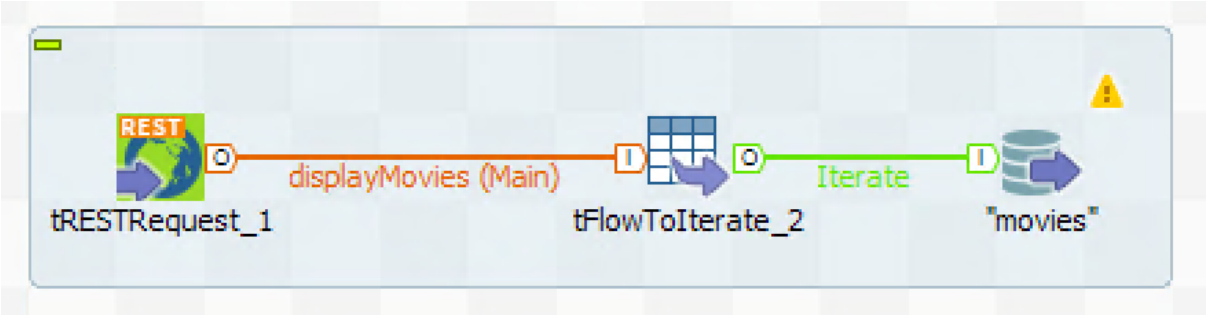
En Talend Studio, en el **Repository**, cree una caperta llamada WebServices, y dentro de ella cree un nuevo **Standard Job** y nombrelo *RESTCatalog_v1*.
Agregue el componente **tRESTRequest**. Del lado del desarrollador, este componente define el endpoint URI de tu REST services, patrones URI, y HTTP verbs. Del lado del despliegue, este componente también es responsable de exponer el servicio en tu ambietne Talend.
En la vista de **Component** del componente **tRESTRequest**, en **REST endpoint** ingrese *"http://localhost:8088/catalog"*. Este parametro setea la URL base donde tu servicio REST sera desplega y accedido.
Para crear la parte de su servicio que le permite a un cliente acceder al catálogo de películas, click en el boton **Plus (+)** y agregue una nueva línea en la seccion **REST API mapping**. Cada nueva linea agrega una operacion al servicio REST.

a. En la nueva lina de la secciónOn the new line of the **REST API Mapping**, en la columna **URI pattern**, ingrese *"/movies"*.
b. En la columna **HTTP Verb**, seleciones **GET**.
c. En la columna **Output Flow**, click en el boton **(...)**, ingrese *displayMovies* y haga click en **OK**. Una nueva ventana se abre, y nos permite definir un schema de salida. Deje vacio y haga click en **OK**.
Esta configuración significa que creó un nuevo servicio REST y se va a exponer en la URL *http://localhost:8088/catalog*.
El servicio de catalogos ofreces un recurso movies con verb HTTP GET. Este recurso puede ser accedido concatenando la URL del servicio y el patron URI. El patron URI es *"/movies"*.
Para leer el catalogo movie, debes invocar la url del servicio *http://localhost:8088/catalog/movies* con verb GET.

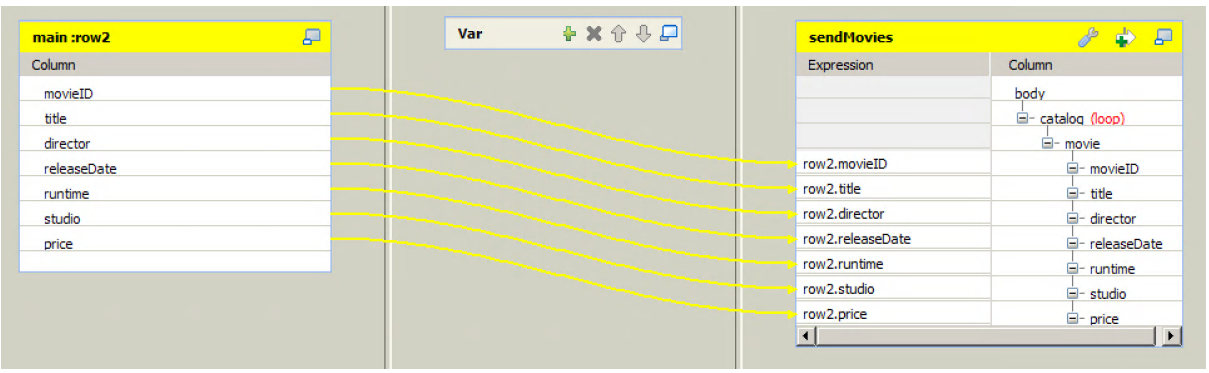


Cuando un cliente consulta el recurso movies en su servicio de catálogo con el verbo GET, se espera que lea las entradas de la tabla de películas en su base de datos MySQL y las devuelva en un mensaje de respuesta XML. El trabajo que necesita crear es simple: lea la tabla de películas en su base de datos, formatee la respuesta como XML y envíela de vuelta al cliente
Necesita leer datos de una base de datos MySQL, lo que significa que necesita usar un componente tDBInput. Este componente no acepta una conexión de fila principal como entrada. Desafortunadamente, el componente tRESTRequest ofrece solo una Main row como conector. Como resultado, utiliza un componente tFlowToIterate, que puede dividir una fila principal y ofrecer un activador de iteración.
Agrega un componente **tFlowToIterate** luego del componente **tRESTRequest**.

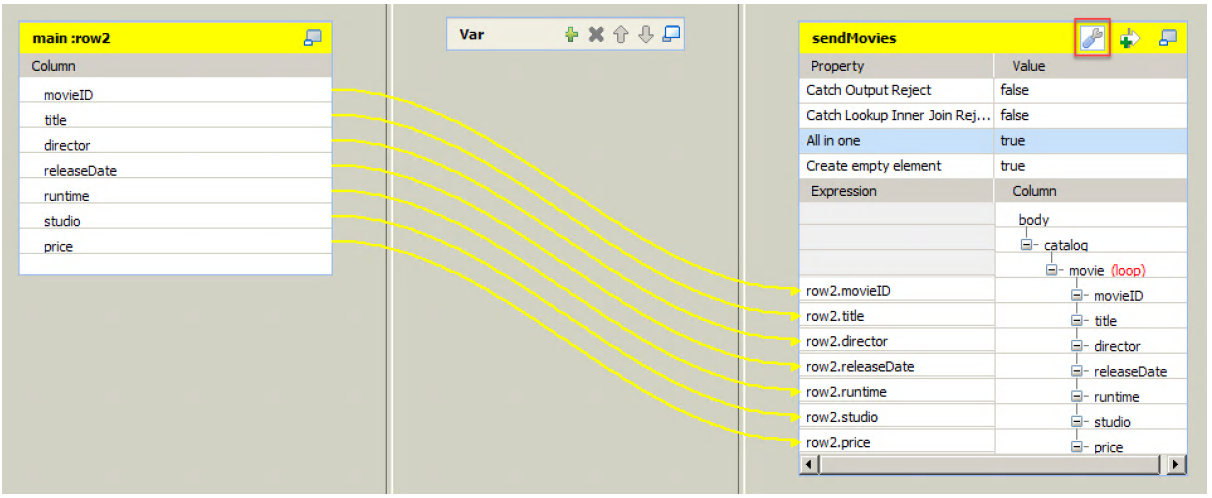
Click derecho en el componente **tRESTRequest** . en el menú contextual, busca **displayMovies** que es el flujo que se definió en los parametros. Seleccione esta salida y una al componente **tFlowToIterate** component.
Desde el **Repository**, navegue por la **Metadata** > **Db Connections** > **trainingDatabase 0.1** > **Table schemas** > **movies**. y arrastrela metadata **movies** dentro del diseño el Job; seleccione usar el componente **tDBInput** .
Click derecho en el componente **tFlowToIterate**, seleccione el flujo **Iterate** desde el menu contextual y una al componente **"movies"**.



Para crear el mensaje de respuesta XML, agregue un componente **tXMLMap** a la derecha del componente **"movies" tDBInput**, y luego una los dos componente con un Main row.
El mensaje de respuesta es enviado al cliente con el componente llamado **tRESTResponse**. coloque un componente **tRESTResponse** despues del componente **tXMLMap** .
Una los componentes **tXMLMap** y **tRESTResponse** y nombre la salida como *sendMovies*. Cuando le pregunte si quiere el schema del componente target, elija **Yes**.
Double-click **tXMLMap**.
En el editor de **tXMLMap** , en el output **sendMovies** , puedes ver un uno elemento de topo Document llamado **body**. Esto es por default, una respuesta vacia para el componente **tRESTResponse**. debajo del **body**, haga click derecho en el elemento **root** y elija **Rename** desde el menu. Renombre el elemento root como *catalog*.
El elemento **catalog** es ahora el tag root de nuestro documeto XML. Un catálogo contiene varios elementos de películas. Haga click derecho en **catalog** y seleccione **Create Sub-Element**. Como nuevo label, ingrese *movie* y haga click en **OK**.
Para crear la estructura del item movie, seleccione todos los elementos del **input (main)**, y muevalos dentro del elemento **movie**. Cuando se le pregunte como crear esos elemento, seleccione **Create as sub-element of target node**. Tu tXMLMap deberia verse como en la siguiente imagen.



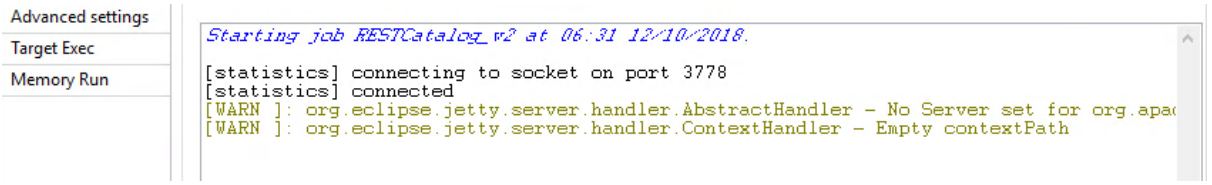
El elemento loop de tu estructura XML todavía está en el tag catalog, deberias moverlo al tag movie. Para cambiar el elemento de loop, click derecho en **movie**, elije **As loop element** desde el menú y click en **OK**.
Abre la configuración de **tXMLMap** (el botón de herramienta en el box de **sendMovies**). Para agregar todas las líneas de datos que vengan desde la base de datos en un único documento XML, setea true en el parámetro **All in one**.



Cierra el editor de tXMLMap, haciendo click en **OK** y luego guarda los cambios de tu job. Tu REST service está listo para ser ejecutado.

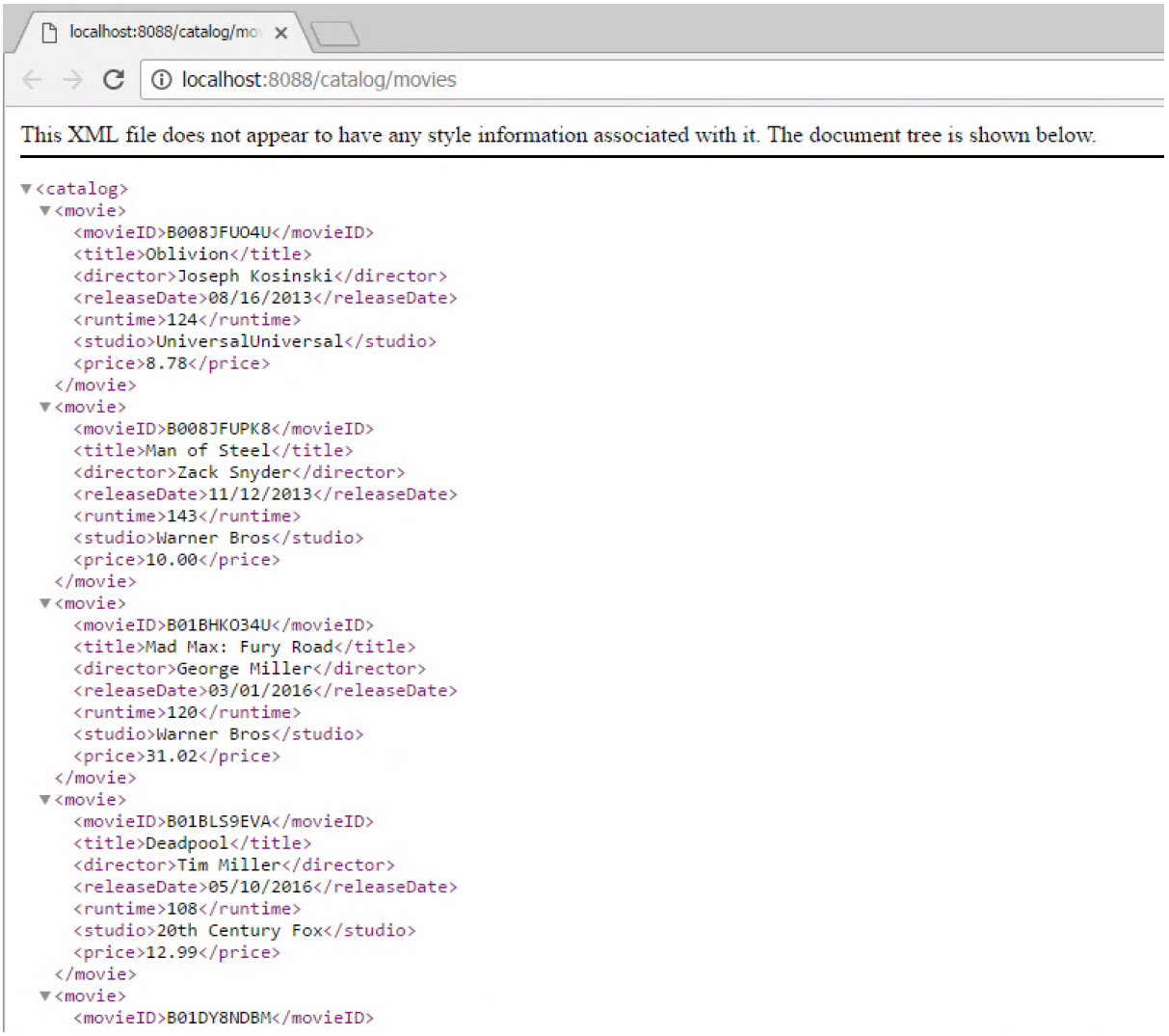
Ejecutando el servicio

Run the service.



como puedes ver en el mensaje INFO log, el servicio está publicado en *http://localhost:8088/catalog*.
Para probar el servicio, abre un web browser.
En la barra de navegación, ingresa *http://localhost:8088/catalog/movies*. No necesitas especificar el verbo GET ya que el web browser lo envía por default.

El servicio REST debería mostrar la respuesta con un mensaje XML similar al que se muestra en la próxima imagen abajo:



No debes olvidar hacer kill en el Job si no necesitas hacer nada mas.

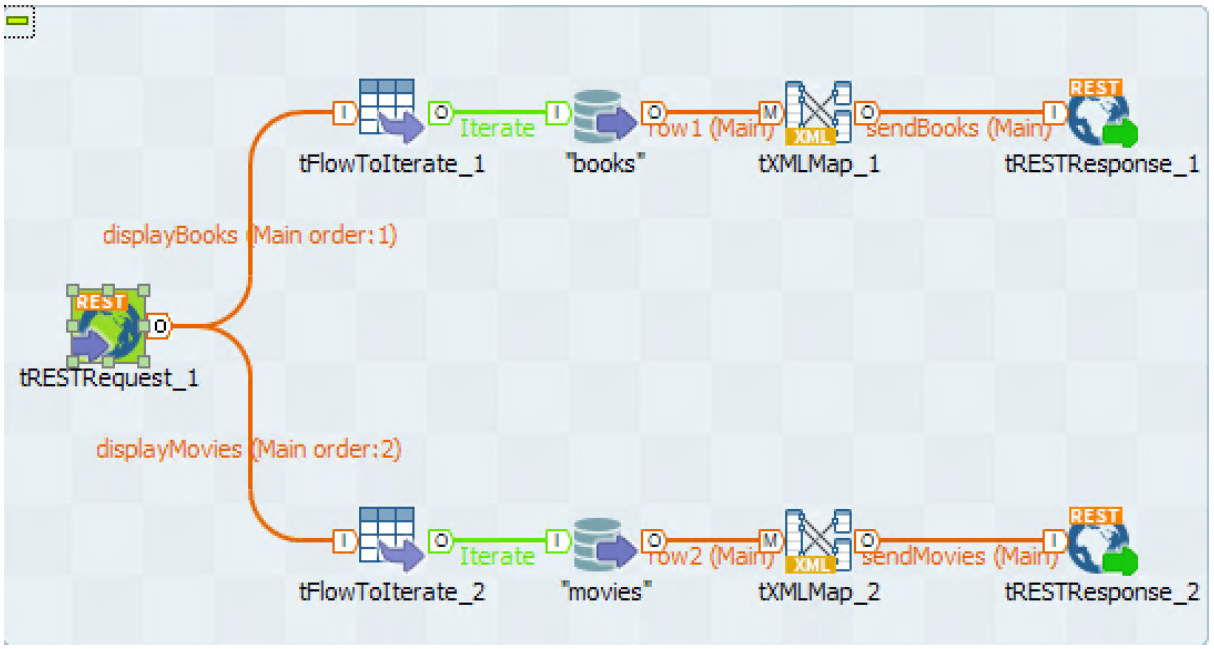
Agregando un segundo patron URI

En la vista **Component** del componente **tRESTRequest**, click en el boton **Plus (+)** y agregue una nueva linea en la seccion **REST API Mapping**, y setee los siguiente parametros:

- **URI pattern**: *"/books"*
- **HTTP Verb**: *GET*
- **Output Flow**: *displayBooks*

Repita los pasos del desarrollo del servicio REST que nos trae informacion de las movies para crear un segundo path en el REST services que nos devuelva la

información sobre libros desde la base de datos.



Cuando su servicio esté funcionando continúe con el siguiente ejercicio.

Tarea

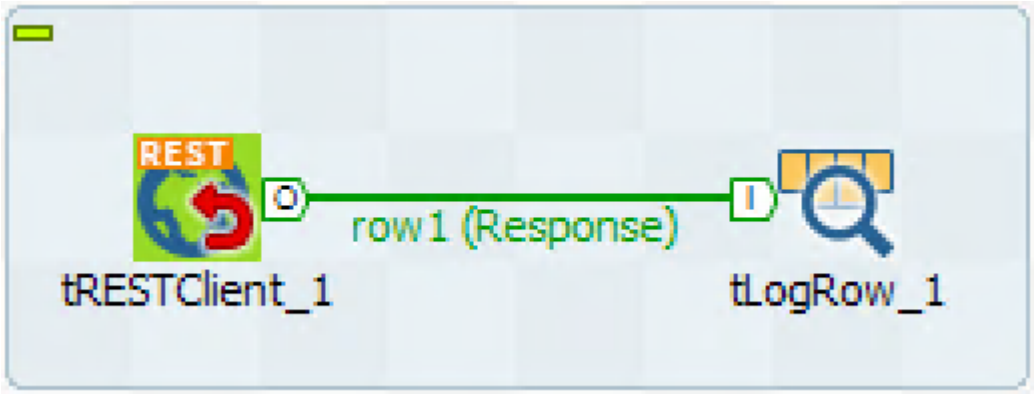
Este ejercicio muestra cómo consumir un RESt webservices desde otro DI Job.

Usando el componente tRESTClient

Cree un nuevo DI Job y nombrelot *RESTConsumer_v1*.
Agregue los siguiente componentes en su diseño:

- **tRESTClient** (consume un servicio REST)
- **tLogRow** (muestra el flujo en la consola)

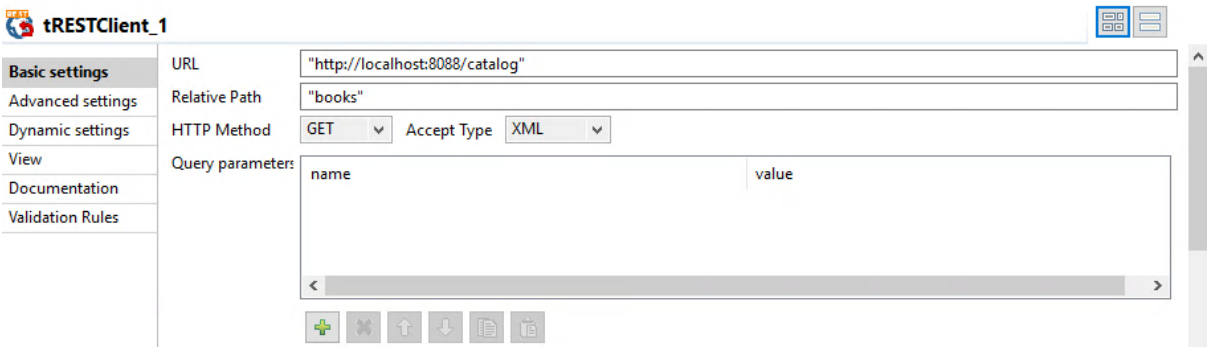
Una los componentes **tRESTClient** y **tLogRow** usando la row **Response**:



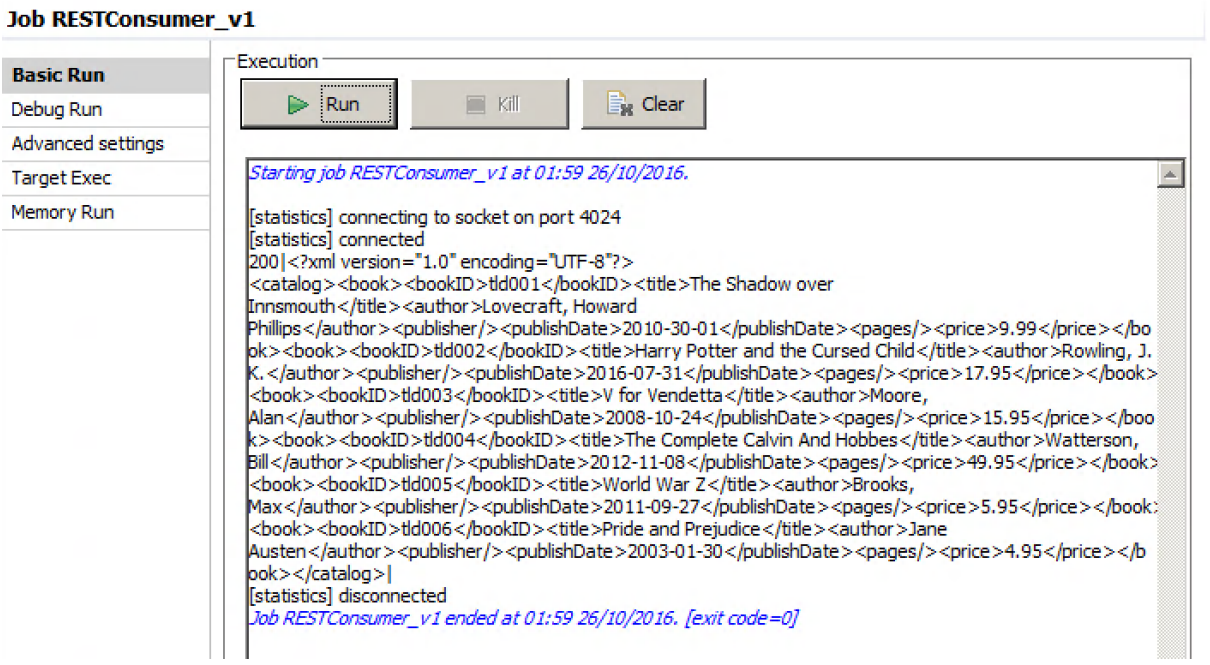
Hay dos diferentes rows posibles y disponibles para los outputs de tRESTClient: **Response**, es usado para la respuesta ok del servicio, y **Error**, en el caso que recibas un mensaje de error.

configura el componente **tRESTClient**:

- En el box de la **URL,** ingresa *"http://localhost:8088/catalog"*.
- En el box de **Relative Path**, ingresa *"books"*.
- En el menu **HTTP Method** , selecciona **GET**.



ejecuta el Job. La respuesta del REST se verá en consola.



No olvides hacer click en kill del job **RESTCatalog_v1** si no lo necesitas para otra prueba.

Ahora sabes cómo crear y consumir servicios REST con Talend Studio. En el siguiente ejercicio, veras más características del componente tRESTRequest.

Tarea

El servicio REST puede recibir y leer parámetros en sus URLs. Este ejercicio muestra cómo configurar y usar esos parámetros.

Agregaremos un nuevo patrón URI al servicio REST desarrollado. El patrón URI contiene movie (o book) ID, y el servicio responde información solo del ítem especificado.

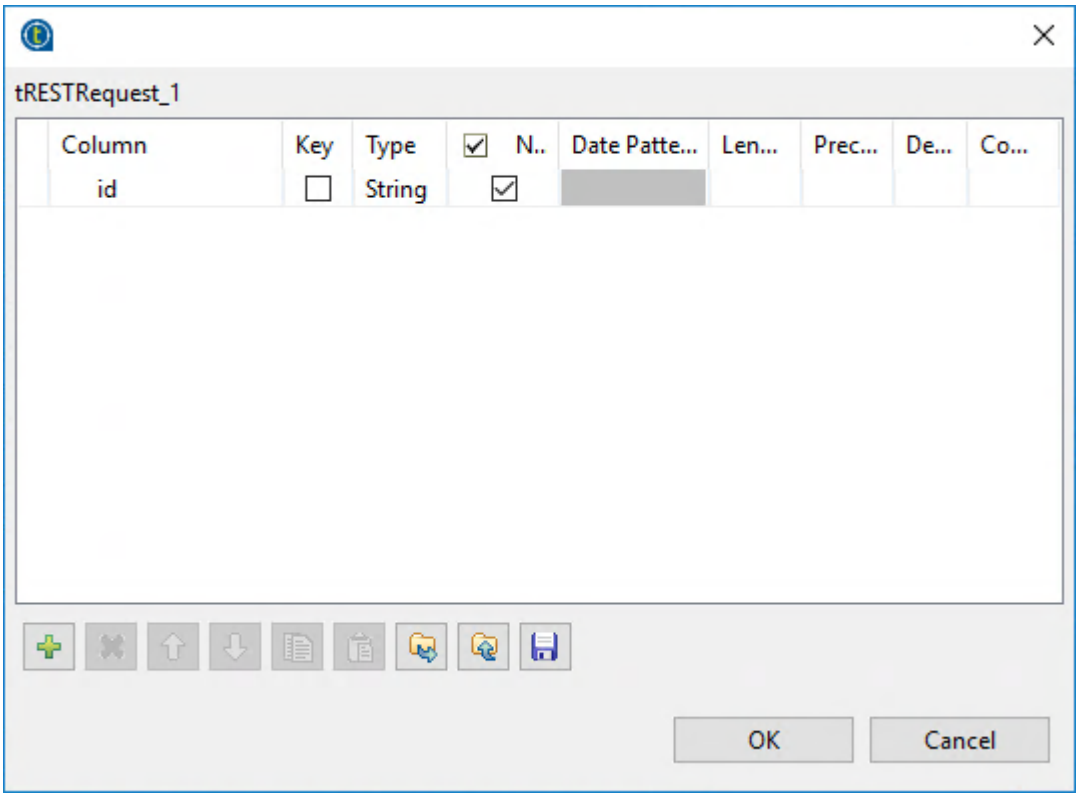
Desarrollando el servicio REST

En el **Repository**, duplica el Job **RESTCatalog_v1** y nombralo *RESTCatalog_v2*.
Abre **RESTCatalog_v2**.
En la vista**Component** del componente **tRESTRequest** , crea una nueva linea de **REST API Mapping** y setea los siguientes parametros:

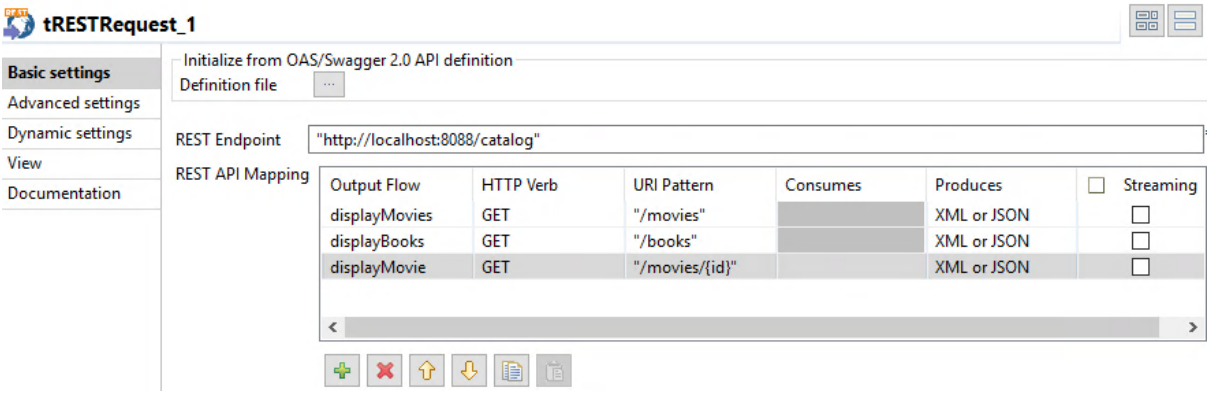
- **URI pattern**: *"/movies/{id}"*
- **HTTP Verb**: *GET*

Observe el elemento **{id}** en el patron URI. Las llaves le dicen al componente tRESTRequest que habrá un elemento variable llamado "id".
En el parámetro **Output Flow** de tu nueva línea **REST API Mapping** , ingrese *displayMovie*. Esta vez, cuando nombre la salida **output** , cuando la ventana **Schema editor** se abra, hacer click en el botón **Plus (+)** y agregar una columna. Este flujo

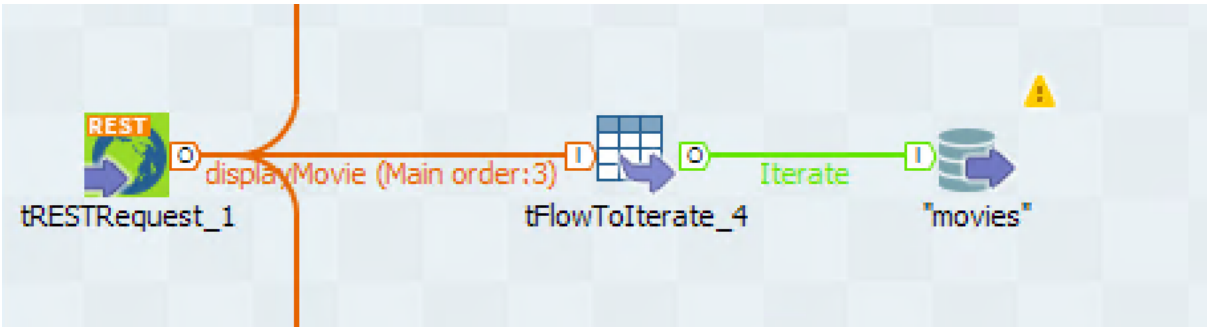
contiene el parámetro **id**.Nombre la columna *id* y setee su tipo como *String*.



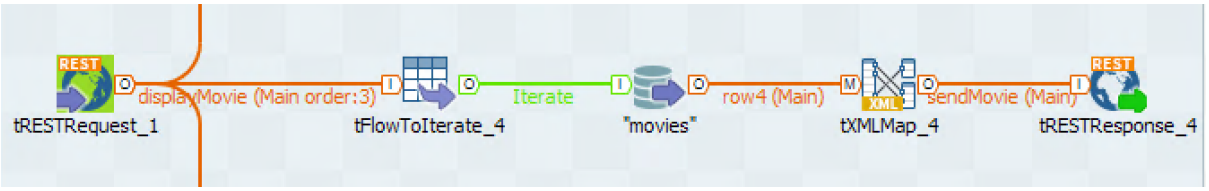
Tu componente tRESTRequest se debería ver como en la siguiente imagen:



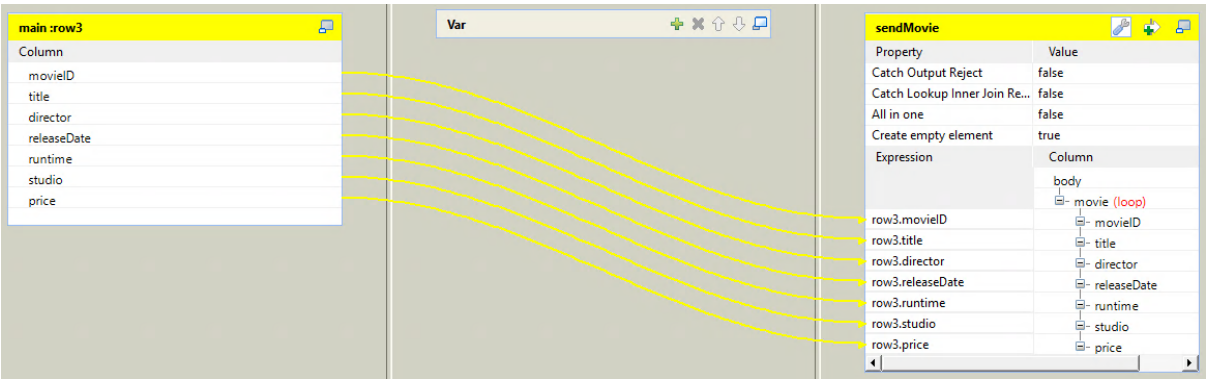
En el diseño, agregar un componente **tFlowToIterate** y unalo con el output **displayMovie** que configuró en el componente **tRESTRequest**.
En el **Repository**, busque en la metadata la tabla **movies** y arrastrela al area de diseño, seleccionando usar el componente **tDBInput**.
Una con un flujo iterate los componentes **tFlowToIterate** y **"movies"**.



Para complentar la opracion de este servicio, agregue un componente **tXMLMap** y un componente **tRESTResponse**. Una el componente **"movies" tDBInput** como el input del componente **tXMLMap**. Cree un nuevo output del componente **tXMLMap** conectandolo con el componente **tRESTResponse** y llame ese flujo *sendMovie*.



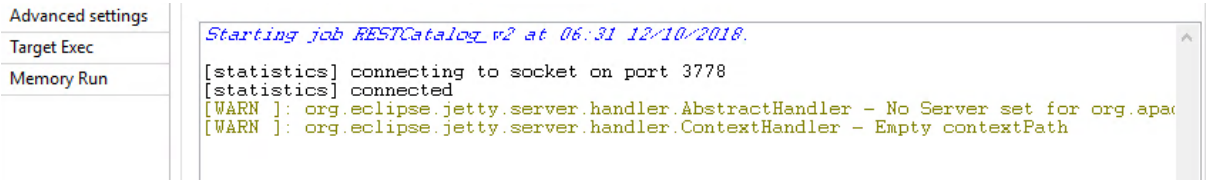
Click en el componente **"movies" tDBInput** y vaya a la vista **Component** de ese componente.
Necesita actualizar la query del sql para que solo traiga los datos del item ID que llego al flujo del serviocio desde el parametro ID de la url. El elemento ID element es propagado desde el componente **tRESTRequest** usando el componente **tFlowToIterate**. Al final de la consulta sql, quite las comillas y agregue:
WHERE `movies`.`movieID` ='"+((String)globalMap.get("displayMovie.id"))+"'"
NRecuerde que puede encontrar facilmente el valor de ID presionando (**Ctrl+spacebar**) en las variables de **tFlowToIterate**.
Configure el componente **tXMLMap** y cree un mensaje de respuesta XML. Deberia ver su configuración como se muestra en la siguiente imagen:



Guarde los cambios del Job.

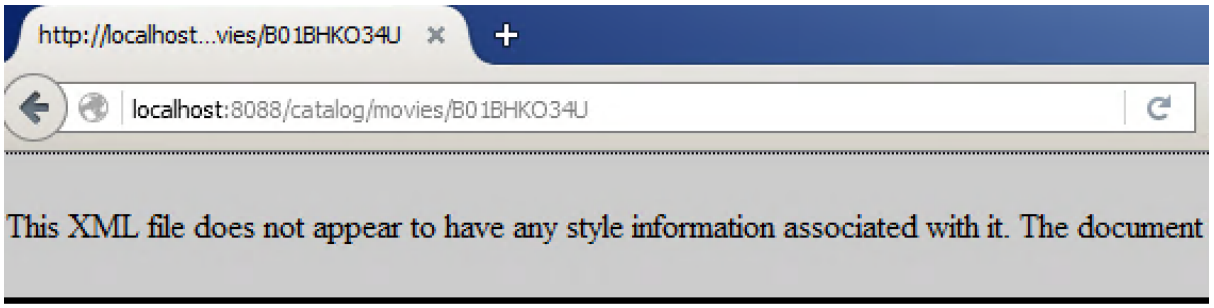
Ejecutando el Servicio

Ejecute el servicio.



En su browser ingrese *http://localhost:8088/catalog/movies/B01BHKO34U* (o use algún movie ID del catalog).

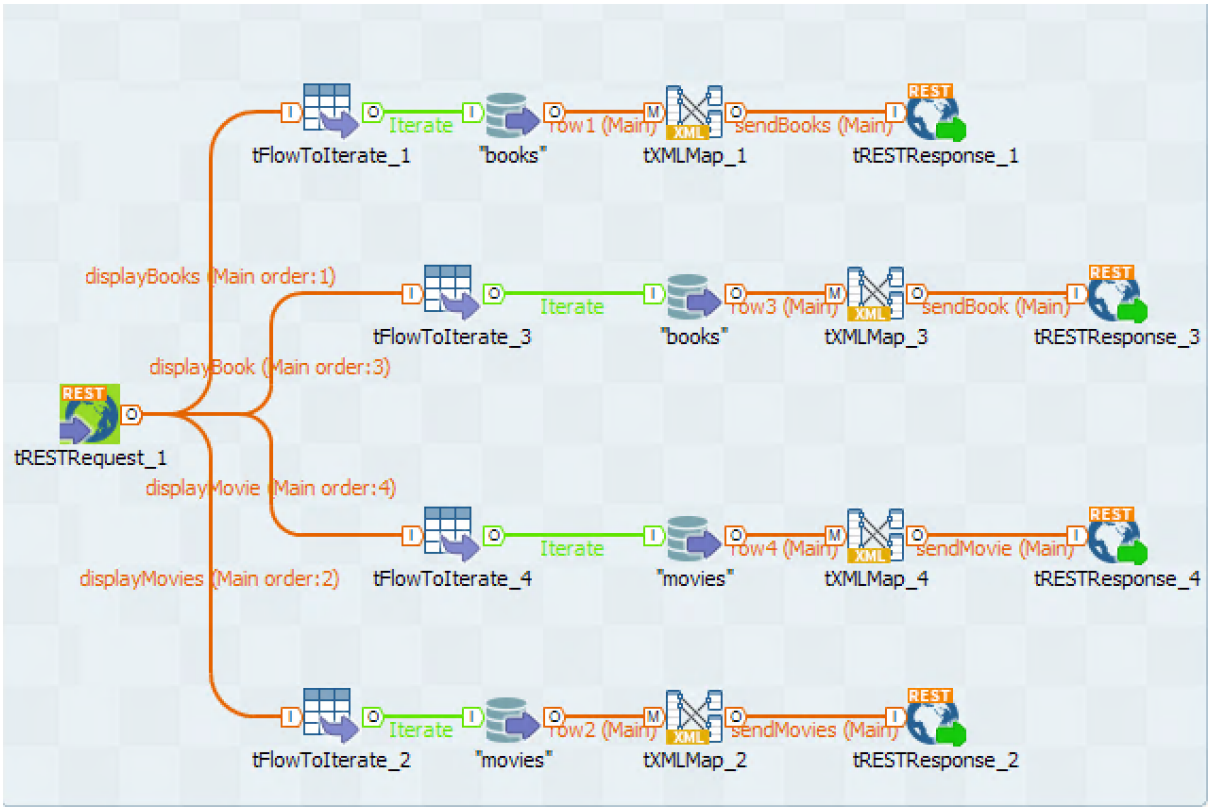
El servicio REST mostrará los datos de una sola movie.




```
- <movie>
    <title>Mad Max: Fury Road</title>
    <director>George Miller</director>
    <releaseDate>03/01/2016</releaseDate>
    <runtime>120</runtime>
    <studio>Warner Bros</studio>
    <price>31.02</price>
  </movie>
```


Desafío

Agregue una cuarta operación al servicio REST conde puedas solicitar la información de un solo libro del catalogo de libros (book), enviando en la url un book ID.



No olvides hacer kill en el job RESTCatalog_v2 Job una vez que ya no lo utilices para pruebas.

Seguidamente vamos a aprender sobre otros HTTP verbs. Ya puedes continuar con el siguiente.

Task outline

Hasta ahora, hemos venido usando el HTTP verb GET, que nos permite leer desde un recurso.

En este ejercicio, vamos a usar el HTTP verb POST. Este verb nos permite escribir en nuestro recurso ya que un POST request siempre viene con datos adjunto: los datos que quieras escribir en el recurso.

Vamos a crear una nueva lineaREST API Mapping en el patron URI /movies con el HTTP verb POST. Esta operación permite que un cliente escriba una nueva entrada de movie en la base de datos.

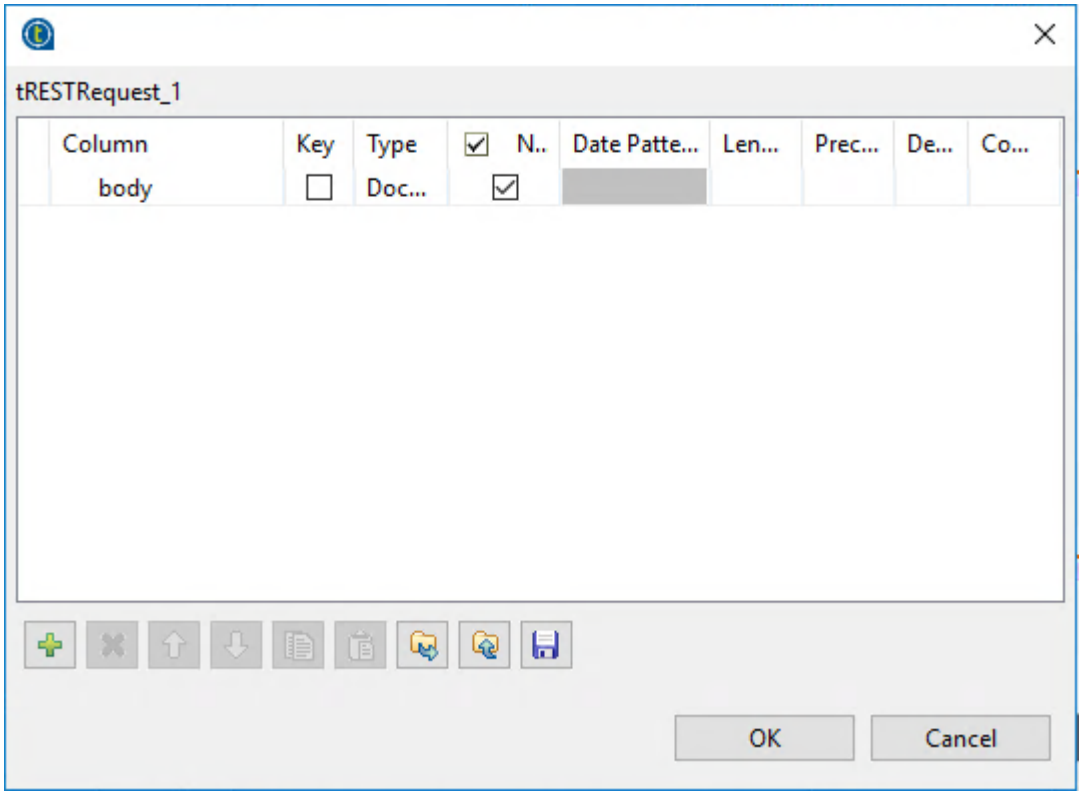
Agregando una nueva operación

En el **Repository**, duplicar el job **RESTCatalog_v2** y nombrarlo *RESTCatalog_v3*. Abra el job **RESTCatalog_v3**.

En la vista **Component** del componente **tRESTRequest**, cree una nueva line **REST API Mapping** y setee los siguientes parámetros:

- **URI pattern**: *"/movies"*
- **HTTP Verb**: *POST*
- **Output Flow**: *addMovie*

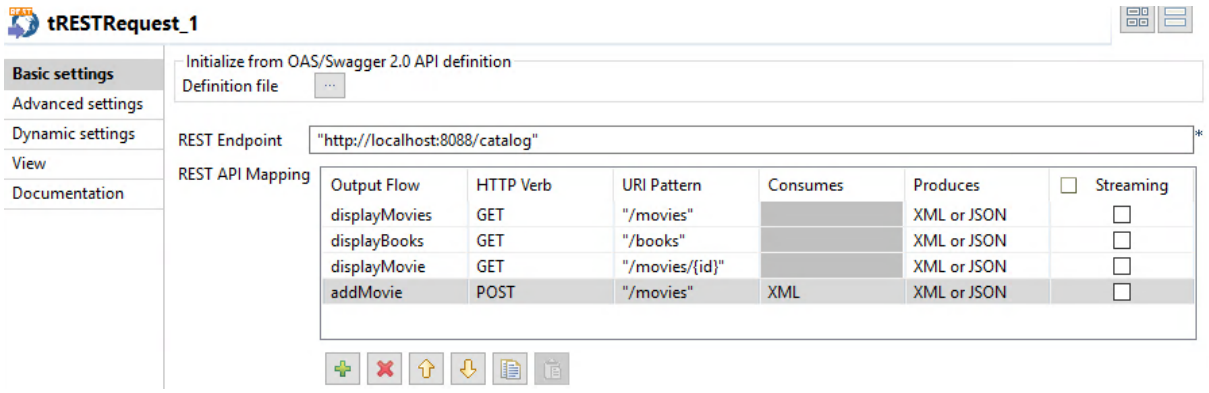
Cuando se abra la ventana de edición de schema, haga click en el botón **Plus (+)** y agregue una nueva columna. Nombre la columna como *body* y setee su tipo como *Document*. Esto indica al componente que la información POST está contenida en el cuerpo de un mensaje.



Cierre el editor de schema, haga click en **OK**.

el tipo document que seteas en el schema puede ser un json o un xml. Además, en la columna **Consumes**, podes elegir un tipo especifico de document. Setea el parametro **Consumes** ara solamente **XML**.

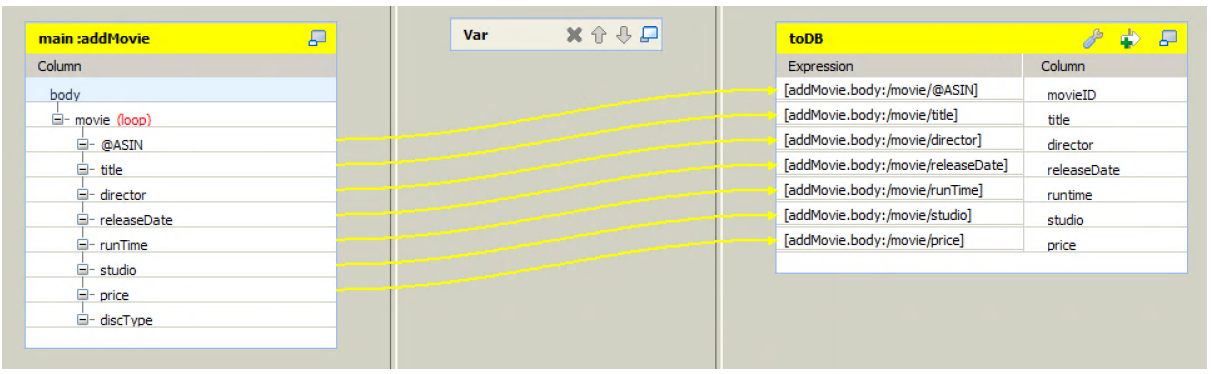
Tu componente **tRESTRequest** debería verse como la siguiente imagen:



El schema del documento xml que será enviado al servcio es un único elemento movie, similar al siguiente:

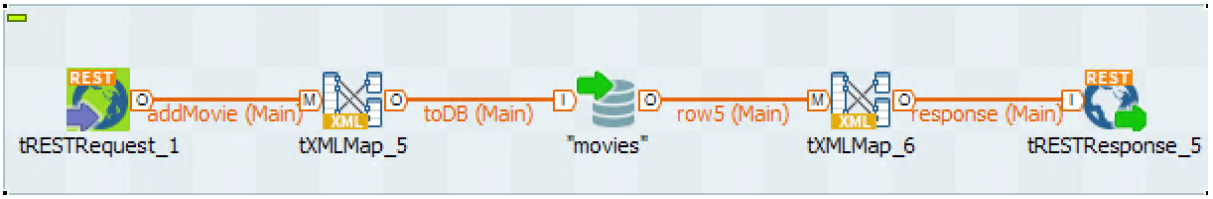
```
<?xml version="1.0" encoding="UTF-8"?>
<movie ASIN="B01DY8NDBM">
    <title>Star Trek</title>
    <director>J.J. Abrams</director>
    <releaseDate>06/14/2016</releaseDate>
    <runTime>126</runTime>
    <studio>Paramount</studio>
    <price>29.96</price>
    <discType>4k</discType>
</movie>
```

Desde el componente **tRESTRequest**, agrega un Main row que contenga el document XML. Necesitas insertar información en la tabla movies de la base de datos y retornar un mensaje al servicio cliente. Agregue un componente **tXMLMap** al diseño y unalo con el output **addMovie** del componente **tRestRequest**.
Desde el **Repository**, navegue por la metadata de la base de datos y busque la tabla **movies** y arrastrela a la derecha del componente **tXMLMap**. Ya que vas a escribir en la base de datos, elije usar el componente **tDBOutput**.
Crea un nuevo output en el componente **tXMLMap**, y nombralo *toDB* y conectalo al componente **"movies" tDBOutput** . Cuando te pregunte si quieres el schema del target, dale click en **Yes**. Esto crea el schema de la tabla movies directamente en el output del componente tXMLMap.
Abre el editor del componente **tXMLMap** haciendo doble click sobre el.
En el input flow **main:addMovie**, click derecho en el elemento **body**, y, en el menu contextual, seleccione **Import From File**. Busque el archivo *C:\StudentFiles\routes\r09_multicast\movie6.xml* y haga click en **Open**.
Mapee los elemntos a las columnas output. Su componente **tXMLMap** se verá como la siguiente imagen:

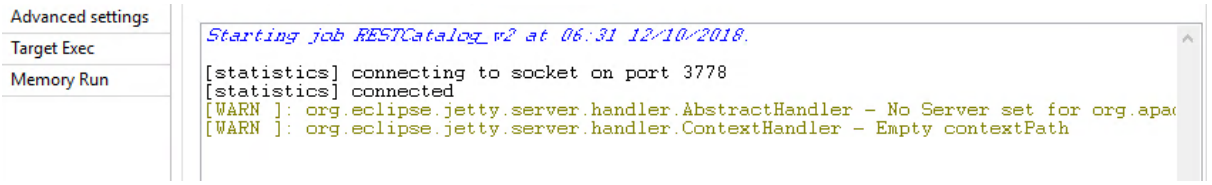


En el editor de **tXMLMap**, haga click para abrir **Tree schema editor** y cambie el tipo de datos a String de los elementos **Runtime** and **Price**.
Cierre el editor de **tXMLMap**, haciendo click en **OK**.
El último ítem que necesitamos desarrollar es el mensaje de respuesta que se retornará al cliente. Recuerda que este servicio rest envía mensajes XML. Para crear el mensaje de respuesta XML, agregue un nuevo componente **tXMLMap** a la derecha del componente **"movies" tDBOutput** y conectelos.
Agregue un componente **tRESTResponse** . Cree un output desde el nuevo componente **tXMLMap**, nombrelo *response* y unalo con el componente

tRESTResponse. El nuevo servicio REST se verá similar a:



Doble-click en el segundo **tXMLMap** y construya y configure el mensaje de respuesta.
En el output flow **response**, haga click derecho en el elemento **root** y renombrelo como *message*.
En la seccion **Expression** de la columna del elemento **message**, ingrese *"Movie "+row5.movieID+" was inserted in the movie catalog"*. asegurese de reemplazar row5 por el name de su imput flow.
Ejecute el servicio.

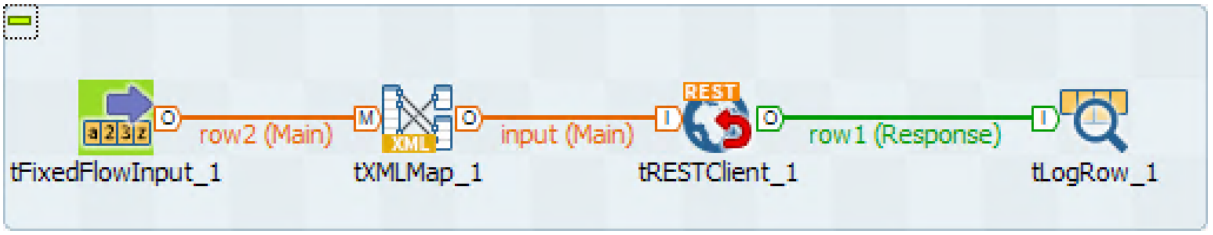


Consumiendo el Servicio

Cree un nuevo job DI Job de nombre *RESTConsumer_v3*.
Agregue los siguientes componente al diseño:

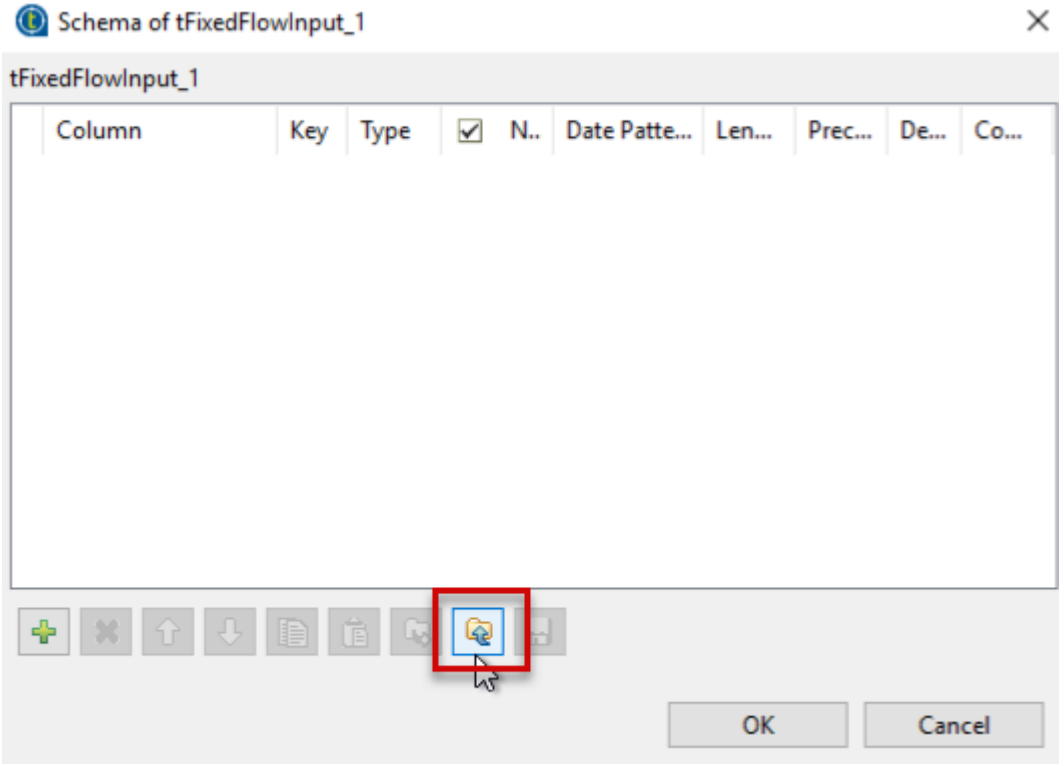
- tFixedFlowInput—usado para generar un row de datos que contienen la información de una movie
- tXMLMap—usado para transformacion la informacion de una movie en un documento XML
- tRESTClient—usado para invocar el webservice REST y obtener una respuesta
- tLogRow—usado para mostrar en consola la respuesta del webservices REST

Una los componentes como se muestra en la siguiente imagen

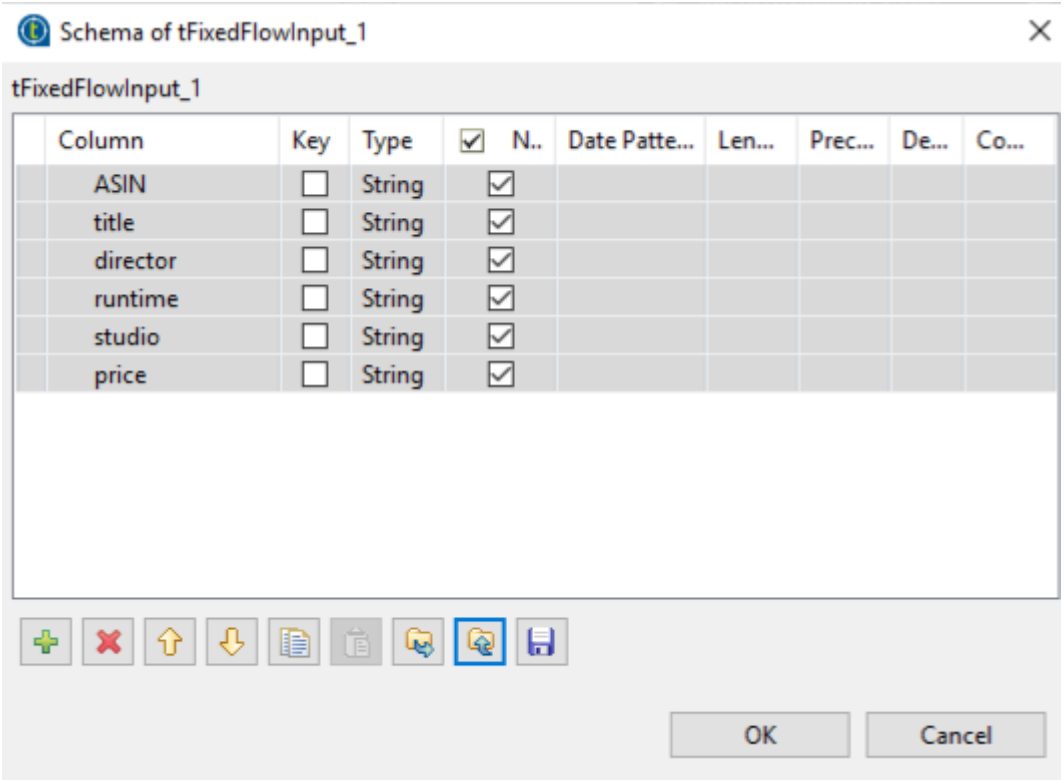


Cuando una el componente **tXMLMap** con el componente **tRestClient**, cuando le pregunte si quiere el schema del componente target, elija **Yes**.
En la vista **Component** del componente **tFixedFlowInput**, haga click en **Edit schema**, click en el boton **(...)**. En la ventana **Schema editor** , cree el schemaa haciendo lo siguiente:

- Click en el boton "Replace all rows by import from xml file"



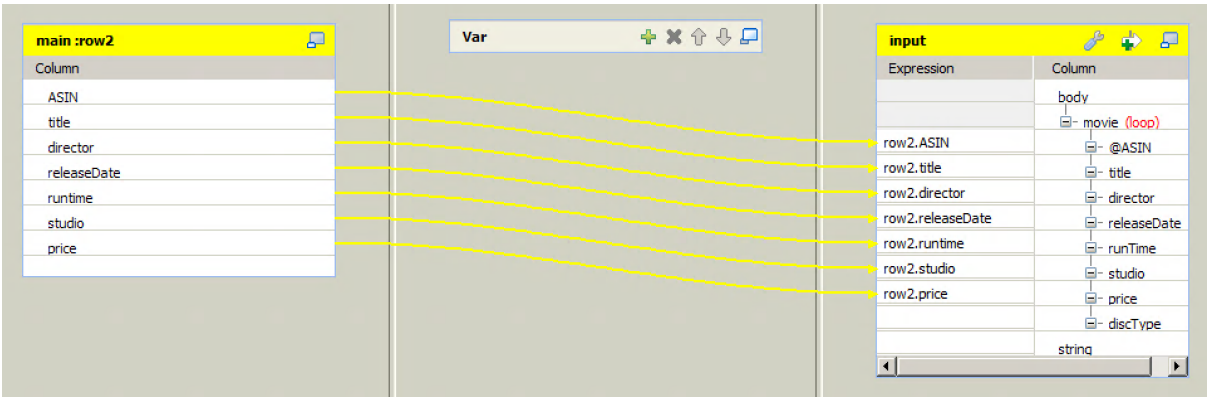
- Seleccione el archivo *C:\StudentFiles\routes\archives\movie_schema.xml*.
- El siguente schema es importado.



- Para guardarlo haga click en **OK**. Click en **Yes** si le preguntan si quiere propagar los cambios.
  En la seccion **Mode**, elija **Use Single Table** e ingrese los siguentes valores:
- **ASIN**: *"B003CPPY88"*

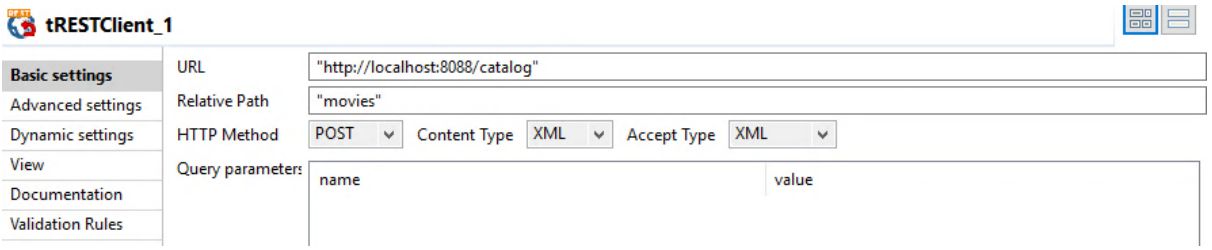
- **title**: *"Stavisky"*
- **director**: *"Alain Resnais"*
- **releaseDate**: *"05/15/1974"*
- **runtime**: *"120"*
- **studio**: *"Cerito Films"*
- **price**: *"12.20"*

Doble-click en el componente **tXMLMap**.En el output flow, click derecho en el elemento **body** , y , en el menu contextual, elija **Import From File**. Busque *C:\StudentFiles\routes\r09_multicast*, seleccione **movie6.xml**, y haga click en **OK**.
el xmlMap de los elementos deberí verse como la siguiente imagen:



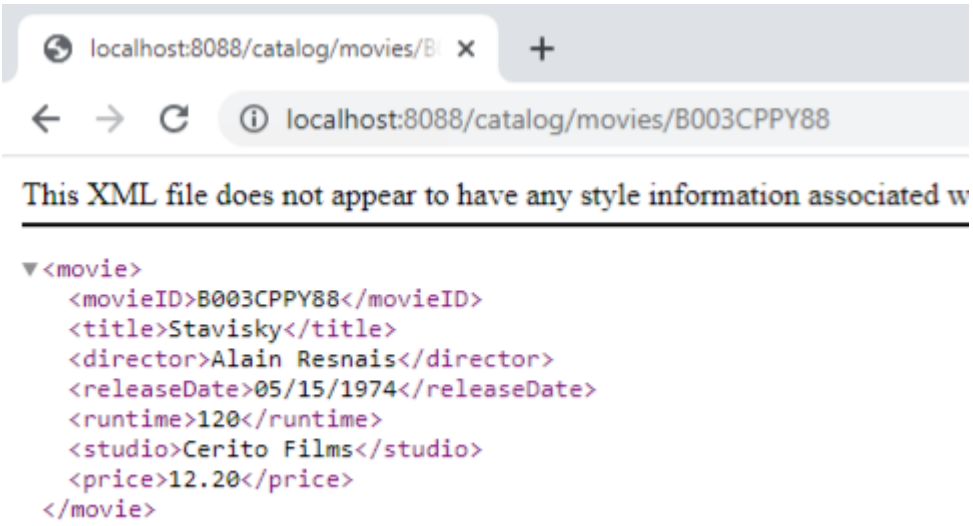
Guardar haciendo click en **OK**.
En la vista **Component** del componente **tRESTClient**, setee los parametros como:

- **URL**: *"http://localhost:8088/catalog"*
- **Relative Path**: *"movies"*
- **HTTP Method**: *POST*
- **Content Type**: *XML*



Guarde y ejecute el Job.

Asegurese que la movie haya sido insertada en la base de datos, invoke su webservices REST:



No olvide de dar click en el botón kill del Job **RESTCatalog_v3** una vez que finalice las pruebas.

Resumen

En este módulo,us aprendió como:

- Diseñar un webservices REST y usar componentes REST dedicados
- Agregar un un endpoint a un servicio REST
- Asignar varios HTTP verbs en el mismo URI endpoint
- Consumir servicios REST