

Primeros pasos con Pentaho Data Integration

Licencia

Esta obra está bajo una licencia Reconocimiento-No comercial-Compartir bajo la misma licencia 3.0 Unported de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-sa/3.0/deed.es> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

Todas las marcas y logos incluidos o mencionados en este documento pertenecen a sus legítimos propietarios.

Autor

María Carina Roldán
Miembro de la Comunidad Pentaho
email: maria.carina.roldan@gmail.com

Prefacio

Este documento es una introducción a PDI, la herramienta de integración de la suite Pentaho.

El texto se desarrolla en torno al clásico ejemplo "Hola, Mundo", explicando detalladamente los conceptos fundamentales de la herramienta.

Si bien el tutorial está destinado principalmente a aquellos que nunca utilizaron PDI, también puede resultar útil para quienes ya comenzaron a explorarla, puesto que aclara algunos conceptos muy presentes en el foro de PDI, como por ejemplo el uso de variables.

El texto está organizado de la siguiente manera:

- ✓ **Introducción** - En primer lugar se introduce la herramienta, y se dan indicaciones para su instalación.
- ✓ **Hola, mundo** - Este es el primer ejemplo del tutorial, con indicaciones paso a paso.
- ✓ **Hola mundo nuevamente** - Consiste en una versión completa y mejorada del ejemplo, donde se amplian los conceptos vistos en dicho ejemplo.

Al final del documento se dan algunas pautas para seguir trabajando, y se mencionan enlaces a la web para ser tenidos en cuenta al trabajar con PDI.

Contenidos

<u>Licencia.....</u>	<u>ii</u>
<u>Autor.....</u>	<u>ii</u>
<u>Prefacio.....</u>	<u>iii</u>
<u>Contenidos.....</u>	<u>1</u>
<u>Introducción.....</u>	<u>2</u>
<u>Instalación.....</u>	<u>2</u>
<u>Conociendo Spoon.....</u>	<u>3</u>
<u>Catálogo y Archivos.....</u>	<u>3</u>
<u>Iniciando la Interfaz Gráfica.....</u>	<u>3</u>
<u>Hola, Mundo!.....</u>	<u>5</u>
<u>Consigna.....</u>	<u>5</u>
<u>Preparación del Ambiente.....</u>	<u>6</u>
<u>Paso a Paso.....</u>	<u>6</u>
<u>Cómo funciona?.....</u>	<u>14</u>
<u>Verificar, Previsualizar y Ejecutar!.....</u>	<u>14</u>
<u>Pan.....</u>	<u>16</u>
<u>Hola mundo, nuevamente!.....</u>	<u>19</u>
<u>Consigna.....</u>	<u>19</u>
<u>Preparación del Ambiente.....</u>	<u>21</u>
<u>A. Creación de la transformación que pide el parámetro.....</u>	<u>22</u>
<u>B. Parametrización de la transformación hola.ktr.....</u>	<u>25</u>
<u>C. Construcción del trabajo principal.....</u>	<u>27</u>
<u>Cómo funciona?.....</u>	<u>30</u>
<u>Kitchen.....</u>	<u>31</u>
<u>Conclusión.....</u>	<u>33</u>

Introducción

PDI (antes Kettle) es el componente de Pentaho responsable de los procesos de ETL. Si bien el uso más frecuente de toda herramienta ETL, incluyendo a PDI, es la población de almacenes de datos (datawarehouses), PDI se puede utilizar también para otros propósitos:

- Migración de datos entre bases de datos o aplicaciones
- exportación de datos de bases de datos a archivos planos
- carga masiva de información en bases de datos
- limpieza de datos
- integración de aplicaciones
- etc.

PDI es fácil de usar. Todos los procesos se crean en un entorno gráfico donde uno especifica qué hacer sin necesidad de escribir código para indicar cómo hacerlo. Es por esto que se dice que la solución está orientada a meta-data.

PDI se puede utilizar en forma independiente, o integrado con el resto de los componentes de la suite de Pentaho. Como herramienta ETL, es la más popular entre las de código abierto. PDI soporta amplia variedad de formatos de entradas y salidas, incluyendo archivos planos, planillas de cálculo, y conexión con motores de bases de datos tanto comerciales como abiertas. Además de las entradas y salidas posee una amplia gama de transformaciones que permiten manipular datos sin limitaciones.

En este tutorial te vamos a mostrar cuán fácil es trabajar con PDI. A través de la construcción de un simple "Hola, Mundo" vas a conocer las características más utilizadas de la herramienta, y sin dudas te va a dar el puntapié inicial para animarte a armar tu propio proyecto.

Instalación

El primer paso para trabajar con PDI es instalar la herramienta.

PDI se puede descargar desde <http://community.pentaho.com/sourceforge/>

Al momento de escribir este tutorial, la versión liberada más nueva de PDI es la 3.0.3. El archivo de descarga correspondiente es `Kettle-3.0.3.GA-nnnn.zip`

PDI no requiere instalación (salvo que descargues la versión .exe).

El único prerequisite para poder trabajar con PDI es tener instalada la JRE 5.0 o superior. La misma se puede descargar de <http://www.javasoft.com/>

Una vez verificado este prerequisite, simplemente tenés que descomprimir el archivo `zip` en una carpeta a elección. En el caso de estar trabajando con ambientes como Unix, Linux, Solaris, MacOS, es necesario dar permisos de ejecución a los scripts. Si este es tu caso, ejecutá los siguientes comandos: (suponiendo que descomprimiste en la carpeta `Kettle`)

```
cd Kettle
chmod +x *.sh
```

Ya tenés todo lo necesario para comenzar a trabajar.

Conociendo Spoon

Cuando ves capturas de pantalla de **PDI**, lo que estás viendo en realidad son capturas de pantalla de **Spoon**, la interfaz gráfica que permite diseñar y testear todos los procesos de PDI. Los otros componentes de PDI sirven para ejecutar los procesos diseñados en Spoon, y se ejecutan desde terminal como veremos más adelante.

Catálogo y Archivos

En Spoon creamos **Trabajos** y **Transformaciones**. Para guardar los trabajos y las transformaciones, PDI permite elegir entre dos formas de almacenamiento:

- Catálogo en Base de Datos
- Archivos

Si se opta por el catálogo, el mismo debe ser creado la primera vez que se ejecuta el Spoon.

Si se elige el método de almacenamiento en archivo, los trabajos se guardan en disco como archivos con extensión `kjb`, y las transformaciones en archivos con extensión `ktr`.

En este tutorial trabajaremos con el segundo método.

Iniciando la Interfaz Gráfica

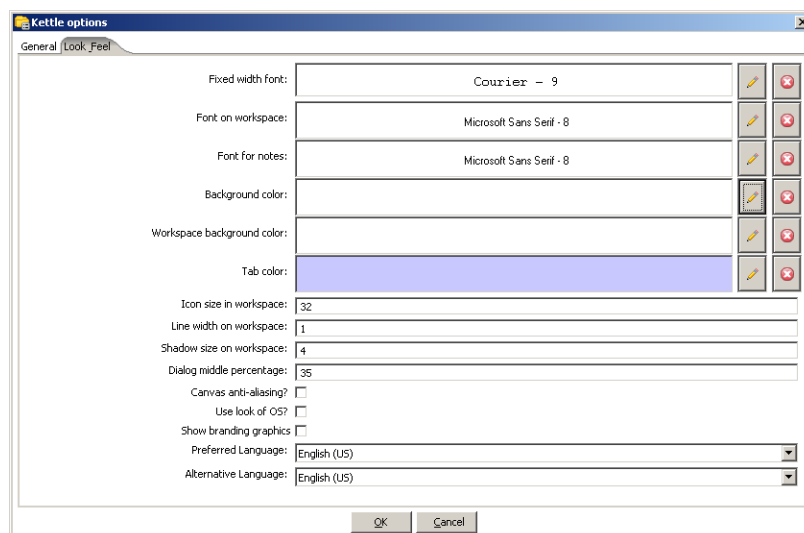
Comencemos a trabajar. Iniciá el Spoon ejecutando:

- o `Spoon.bat` en Windows
- o `Spoon.sh` en otras plataformas como Unix, Linux, ...

Apenas inicia Spoon, se presenta una ventana solicitando los datos de conexión del catálogo. Como dijimos, en este tutorial trabajaremos con archivos. Por lo tanto, seleccioná el botón **No Repository**.



Lo primero que verás será una pantalla de bienvenida. Al ser la primera vez que entrás, probablemente quieras cambiar algún aspecto de la interfaz, en particular el idioma. Para hacerlo, seleccioná **Options...** dentro del menú **Edit**. Se presenta una ventana donde se puede cambiar varios aspectos generales y de visualización. En la solapa **Look Feel**, cambiá el idioma seleccionando español dentro de **Preferred language**. Para que el Spoon reconozca los cambios, será necesario que lo reinicies.



Vas a notar que ahora que los botones están en español: **No Repository** fue reemplazado por **Sin Catálogo**.

Ahora sí estamos preparados para dar la bienvenida al "Hola, Mundo".

Hola, Mundo!

“Hola, Mundo” será nuestra primera experiencia con PDI. A pesar de ser un ejemplo sencillo veremos varias de las características fundamentales de PDI:

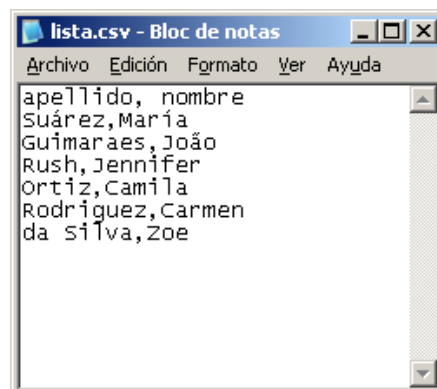
- trabajo con la herramienta Spoon
- Transformaciones
- Pasos y Saltos
- Variables predefinidas
- Previsualización y Ejecución desde la interfaz gráfica
- Ejecución desde línea de comandos con la herramienta Pan.

Observá! Cada vez que veas un cuadro como éste, tené en cuenta que contiene definiciones y consejos importantes que te servirán más allá de la aplicación que le demos dentro del tutorial.

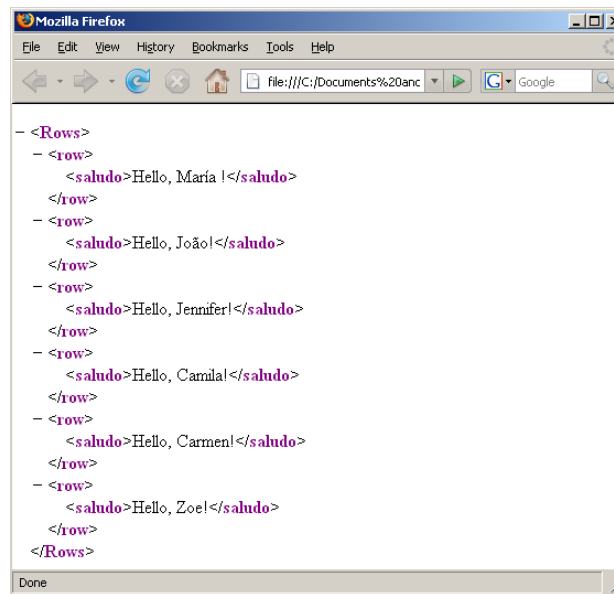
Consigna

Supongamos que tenemos un archivo csv con una lista de personas. Queremos generar un archivo xml que contenga saludos para todas ellas.

Si éste fuera el contenido del archivo de nombres:



Esta sería la salida:



```
- <Rows>
- <row>
  <saludo>Hello, María!</saludo>
</row>
- <row>
  <saludo>Hello, João!</saludo>
</row>
- <row>
  <saludo>Hello, Jennifer!</saludo>
</row>
- <row>
  <saludo>Hello, Camila!</saludo>
</row>
- <row>
  <saludo>Hello, Carmen!</saludo>
</row>
- <row>
  <saludo>Hello, Zoel!</saludo>
</row>
</Rows>
```

La generación del archivo de saludos a partir del archivo plano será la tarea de nuestra primera transformación.

Una **Transformación** es una unidad de ejecución compuesta por **Pasos** enlazados por medio de **Saltos**. Estos Pasos y Saltos conforman caminos por donde los datos fluyen: entran, se transforman y salen. Por eso se dice que una transformación está orientada al **flujo de datos**.

Preparación del Ambiente

Antes de comenzar, creá la carpeta `Tutorial` en un lugar a elección. Ahí guardaremos todos los archivos del tutorial.
Luego creá un archivo como el que vimos en la introducción de este ejemplo, y guardalo con el nombre `lista.csv` en la carpeta creada.
No es obligatorio que este archivo exista, pero su existencia va a facilitar la configuración del primer paso de la transformación.

Paso a Paso

Resolveremos la consigna en tres partes:

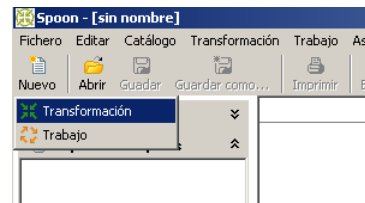
- 1) Crearemos la transformación
- 2) Construiremos el esqueleto de la transformación utilizando Pasos y Saltos
- 3) Configuraremos los Pasos para especificar su comportamiento

Paso 1. Creación de la transformación

- Seleccioná **Nuevo** → **Transformación**. Alternativamente podés crear una transformación desde el menú **Fichero** → **Nuevo** → **Transformación** o presionar **<Ctrl-N>**. Gran parte de la pantalla es ahora ocupada por esta

transformación nueva. Este es el espacio de trabajo donde colocaremos los Pasos y los Saltos.

- Seleccioná la opción **Transformación** → **Configuración**. Se presenta una ventana donde podés especificar propiedades generales de la nueva transformación. En este caso solamente escribí para la transformación un nombre y una descripción, como se ve en esta pantalla:
- Presioná el botón **Guardar**. Guardá la transformación en la carpeta Tutorial con el nombre hola. Se habrá creado el archivo hola.ktr.



Paso 2. Construcción del Esqueleto de la Transformación utilizando Pasos y Saltos

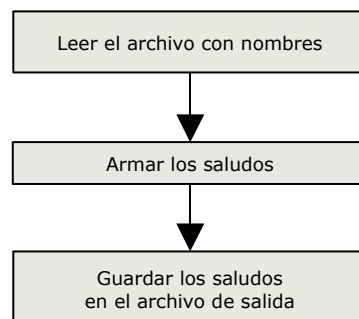
Un **Paso** es la unidad más atómica dentro de una transformación. Existe una gran variedad de Pasos disponibles, agrupados en categorías como Entrada o Salida, entre otras. Cada Paso está concebido para cumplir una funcionalidad específica, que va desde leer un parámetro hasta normalizar un conjunto de datos.

Un **Salto** es la representación gráfica del flujo de datos entre dos Pasos: un origen y un destino. La pila de datos que pasa por ese Salto constituye la **Cadena de Salida** del Paso origen, y la **Cadena de Entrada** del Paso destino.

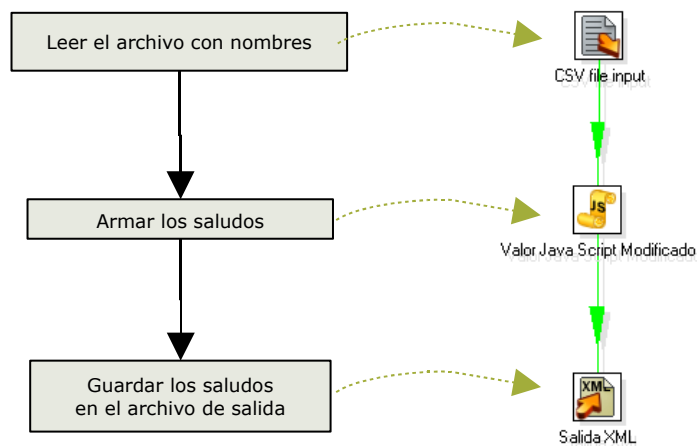
Si bien un salto tiene sólo un origen y un destino, de un Paso puede salir más de un Salto. Si ese es el caso, la Cadena de Salida puede copiarse a todos los Pasos destino del Salto, o puede distribuirse entre esos Pasos.

Asimismo, a un Paso pueden llegar más de un Salto. En este caso, el paso debe tener la capacidad de combinar las Cadenas de Entrada para luego generar la Cadena de Salida.

Veamos lo que la transformación tiene que hacer:



Para cada uno de estos items usaremos un Paso diferente, según el siguiente diagrama:

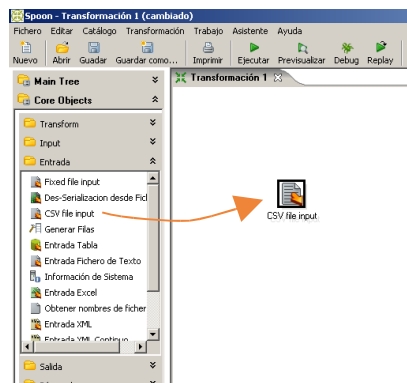


Tené en cuenta que esta correspondencia uno a uno se da porque nuestra transformación es muy sencilla. No siempre es así!

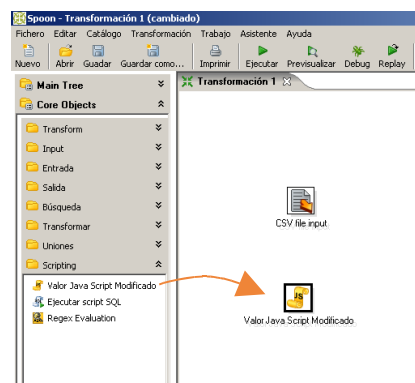
Como verás en los nombres de los Pasos que elegimos, que no toda la herramienta está traducida al español.

Estos son los pasos a seguir:

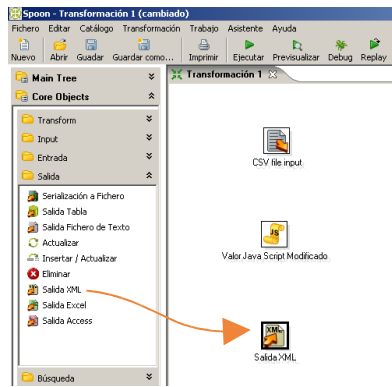
- Identificá dentro de la paleta de Pasos la categoría **Entrada**. Hacé click en el paso **CSV file input** y arrastralo a la pantalla de trabajo.



- Identificá dentro de la paleta la categoría **Scripting**. Hacé click en el paso **Valor Java Script Modificado** y arrastralo de la misma manera.

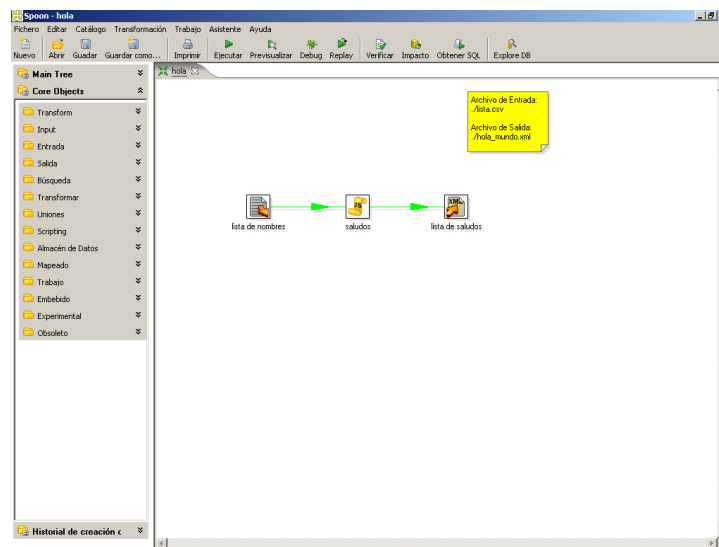


- Identificá dentro de la paleta la categoría **Salida**. Hacé click en el paso **Salida XML** y arrastralo también al espacio de trabajo.



- Enlazá **CSV file input** con **Valor Java Script Modificado** por medio de un salto: Hacé click en el primer Paso y con la tecla <Shift> presionada, arrastrá el cursor hacia el segundo Paso. (en el manual de Spoon se explican otras maneras para crear los Saltos)
- Enlazá **Valor Java Script Modificado** con **Salida XML** de la misma manera.

Luego de seguir estas indicaciones, tendrás la siguiente pantalla:



Paso 3. Configuración de los Pasos para especificar su comportamiento

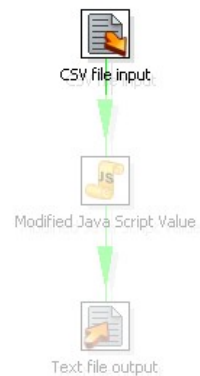
Todos los Pasos tienen asociada una **ventana de configuración**. Estas ventanas varían según la funcionalidad del paso y la categoría dentro de la cual se encuentra, pero todas tienen en común los siguientes datos:

Nombre del Paso: es un nombre representativo dentro de la transformación.

Descripción: Permite ampliar información sobre el propósito del paso.

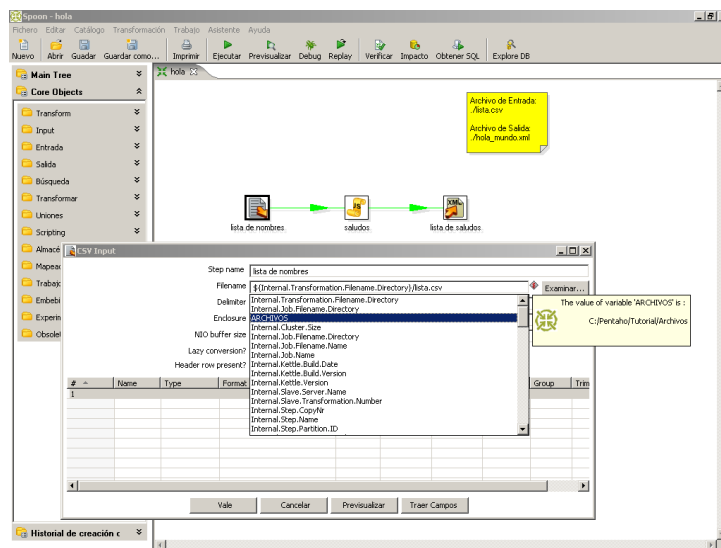
a) Configuración del Paso **CSV file input**

- Hacé doble click sobre el Paso **CSV file input**. Se presentará la pantalla de configuración de este tipo de Paso. Aquí vamos a indicar la ubicación, el formato y el contenido del archivo de entrada.
- **Step name**: Primero que nada *en cualquier paso que configures* no te olvides borrar el nombre que se asigna por defecto y pónelo al Paso un nombre más representativo para la función que va cumplir dentro de la transformación. En este caso le vamos a poner "lista de nombres".
- **Filename**: En este lugar se debe indicar el nombre y ubicación del archivo de entrada. A la derecha del espacio destinado al texto, vas a ver un símbolo con el signo \$.



Este símbolo a la derecha de una caja de texto implica que en esa caja además de escribir texto podés usar **variables**. La variable se puede escribir manualmente como **`\${nombre_de_la_variable}`** o seleccionarla de la ventana de variables disponibles, la cual se accede presionando **<Ctrl-Espacio>**. La ventana muestra tanto las variables predefinidas por PDI como aquellas definidas por el usuario.

Como no hemos creado ninguna variable, sólo vas a ver las variables predefinidas por PDI como se ve en esta pantalla:



Entre estas, seleccioná

`${Internal.Transformation.Filename.Directory}`

Al lado del nombre de la variable escribí el nombre del archivo que creaste. El texto quedará así:

`${Internal.Transformation.Filename.Directory}/lista.csv`

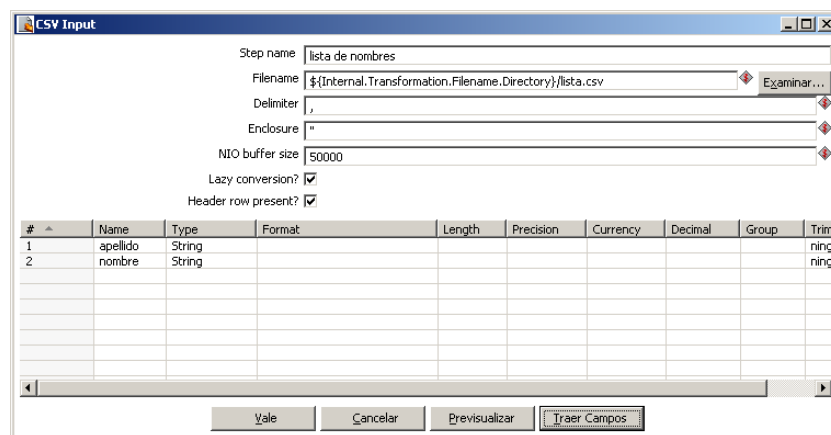
En tiempo de ejecución, la variable será reemplazada por su valor, que será la ruta completa de almacenamiento de la transformación en el disco. La transformación buscará el archivo `lista.csv` en esa ruta.

- **Traer Campos:** Presioná este botón para que la aplicación te traiga a la grilla la lista de nombres de columnas del archivo indicado. Fijate que por defecto este paso asume que el archivo tiene encabezados (**Header row present** está tildado).

El botón **Traer Campos** se encuentra en la ventana de configuración de muchos tipos de Pasos. Su función es cargar una grilla con datos provenientes de fuentes externas (por ejemplo de un archivo plano), o provenientes de pasos anteriores. Si bien los campos pueden escribirse manualmente, este botón sirve de atajo cuando son muchos los campos que tenemos disponibles y queremos utilizar.

La pantalla tiene ahora los nombres de las columnas del archivo de nombres: apellido y nombre.

Ya casi terminamos con este paso. La pantalla ahora debería verse así:



- **Previsualizar:** Simplemente para asegurarte que el archivo se lea correctamente, presioná el botón de previsualización. Se presentará una ventana mostrando datos traídos del archivo: tantos como le solicites (excepto que la cantidad de datos del archivo sea menor).
- **Vale:** Presioná este botón, y ya está configurado el Paso **CSV file input**.

b) Configuración del Paso Valor Java Script Modificado

- Hacé doble click sobre el Paso **Valor Java Script Modificado**. Se presentará la pantalla de configuración de este tipo de Paso, por cierto, muy diferente a la pantalla que vimos en el paso anterior. En este caso, el Paso permite crear código javascript, y lo vamos a usar para construir el mensaje "Hola, " concatenado con cada uno de los nombres del archivo.
- **Step name:** Dale a este Paso el nombre "Construcción del mensaje".



- **Código javascript:** Gran parte de la pantalla de configuración está destinada al código javascript. A la izquierda hay un árbol con funciones disponibles que ayudan a la codificación. En particular, las últimas ramas del árbol contienen todos los campos de entrada (Input Fields) y de salida (Output Fields) disponibles para usar dentro del código. En nuestro ejemplo hay dos campos: `apellido` y `nombre`.

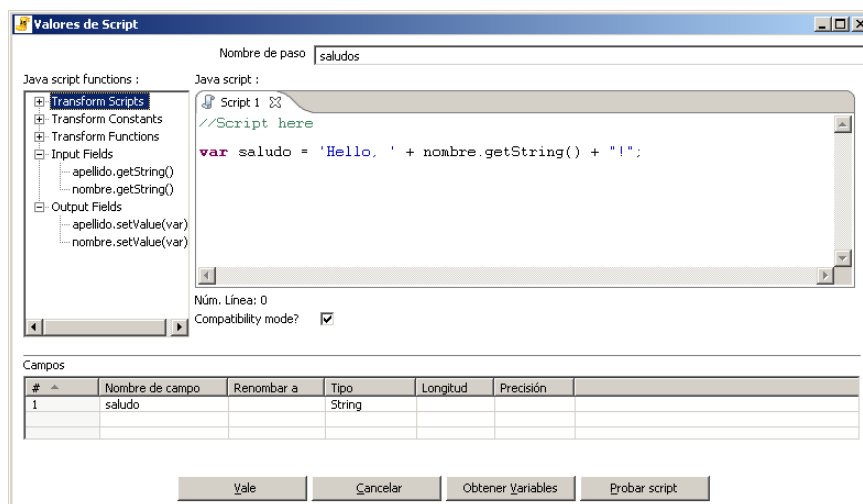
Escribí en esta parte de la pantalla el siguiente código:

```
var saludo = 'Hola, ' + nombre.getString() + "!";
```

El texto `nombre.getString()` se puede escribir manualmente, o haciendo doble click en el mismo texto dentro de "Input Fields" en el árbol de funciones.

- **Campos:** En la parte inferior se puede agregar datos creados en el código y que querramos incorporar al flujo de datos. En este caso, hemos creado la variable de javascript `saludo`. (**No confundir con variables de PDI! No son lo mismo.**) Como la necesitamos para enviarla al archivo de salida, la agregamos en esta grilla.

Ya hemos configurado el Paso. Tenemos este resultado:



A propósito, "modificado" no es un adjetivo para "javascript", sino para el Paso propiamente dicho. No temos aquí un javascript modificado, sino que el paso es una versión mejorada del paso "Valor Javascript" que existía en versiones anteriores de PDI.

- **Vale:** Presioná este botón, y ya está configurado el Paso **Valor Java Script Modificado**.
- Seleccioná el Paso que acabamos de configurar. Para comprobar que realmente el nuevo dato esté saliendo de este paso, vamos a ver los campos de entrada y de salida de este Paso.

Campos de Entrada: son las columnas de los datos que llegan a un Paso.
Campos de Salida: son las columnas de datos que salen de un Paso.

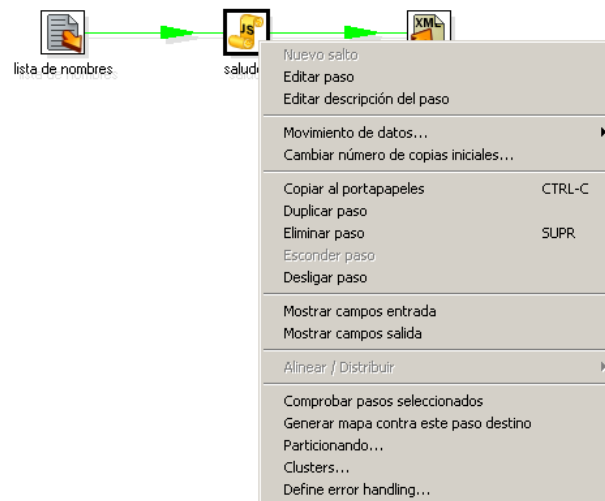
Hay Pasos que simplemente transforman los datos de entrada. En estos casos los campos de entrada y de salida coinciden.

Pero no siempre es así:

Hay pasos que agregan datos a los Campos de Salida. Ejemplo: **Calculadora**

Hay otros pasos que filtran o combinan datos, haciendo que la salida tenga menos datos que la entrada: Ejemplo: **Agrupar por**

Presionando el botón derecho sobre el Paso, aparecerá el siguiente menú contextual:



- Seleccioná **Campos de Entrada**. Vas a ver que los campos de entrada son los que vienen del Paso **CSV file input**: nombre y apellido.
- Seleccioná **Campos de Salida**. Vas a ver que a esos datos, se agregó el dato `saludo`.

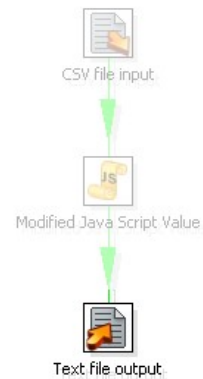
c) Configuración del Paso **Salida XML**

- Hacé doble click sobre el Paso **Salida XML**. Se presentará la pantalla de configuración de este tipo de Paso. Aquí vamos a indicar el nombre y ubicación del archivo de salida, y qué datos queremos incluir en el mismo: podrían ser todos o sólo una parte de los datos que llegan a este Paso.
- **Step name**: Dale a este Paso el nombre "Archivo con saludos".
- **Fichero**: En el lugar destinado al nombre del archivo, escribí:

```
${Internal.Transformation.Filename.Directory}/hola.xml
```

Recordá que el nombre de la variable lo podés seleccionar de la ventana de variables presionando **<Ctrl-Espacio>**.

- **Contenido**: Esta solapa permite configurar el formato de la salida. Dejá los datos propuestos por defecto.
- **Campos**: Presioná el botón **Obtener Campos**. La grilla se completa con los tres datos de entrada. En el archivo solamente queremos incluir el saludo, por lo tanto eliminá el nombre y el apellido.



Por último no te olvides de grabar nuevamente la transformación con todos los cambios realizados.

Cómo funciona?

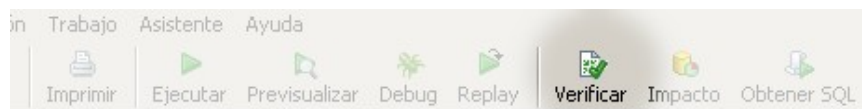
Cuando ejecutamos una transformación, casi todos los pasos se ejecutan en forma simultánea. La ejecución es **asincrónica**. Las filas de datos fluyen a través de los Pasos a su propio ritmo. Cada fila de datos procesada fluye hacia el siguiente Paso sin esperar a las demás.

En la práctica, olvidarse de esta propiedad de las transformaciones, puede ocasionar resultados que no coinciden con los esperados.

Ya está todo configurado. La transformación lee el archivo de entrada, *para cada una de las filas de datos* (en este caso para cada nombre) se ejecuta el código javascript que genera un mensaje y finalmente ese mensaje es enviado al archivo de salida. Siendo este ejemplo tan pequeño y teniendo tan pocos datos, es muy difícil percibir que esta transformación ejecuta sus pasos de forma asincrónica, pero funciona siempre así, por ejemplo: puede ser que en un momento se esté enviando algún mensaje de saludo hacia el archivo de salida, mientras algunas filas apenas están saliendo del Paso que lee el archivo.

Verificar, Previsualizar y Ejecutar!

Una vez que terminamos de construir nuestra transformación, y antes de ejecutarla, vamos a hacer una verificación de la misma. Presioná el botón **Verificar**.



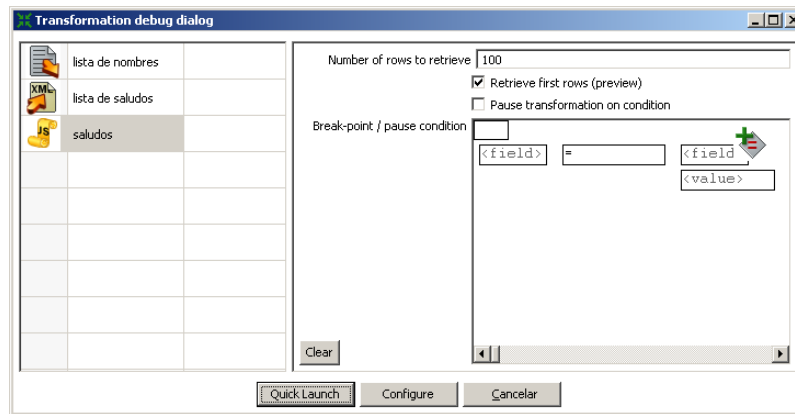
El Spoon realizará un chequeo sintáctico de la transformación, detectando de esta manera errores de lógica como: pasos no alcanzables, falta de datos de conexión en pasos relacionados con bases de datos, entre otros.

Si todo está en orden (debería estar si seguiste los pasos correctamente), procederemos a previsualizar la salida.

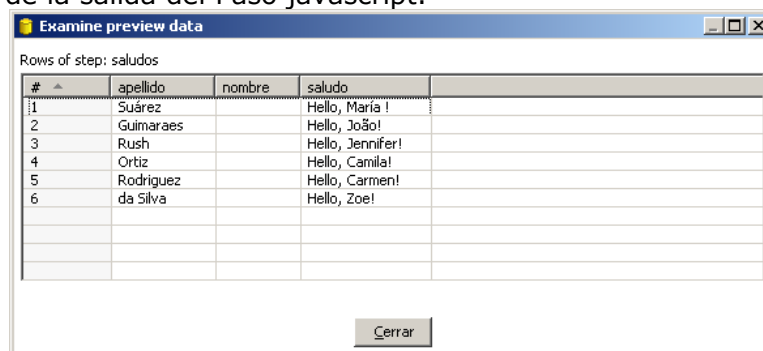
Con el cursores posicionado en el paso javascript, seleccioná el botón **Previsualizar**.



Se presentará la siguiente ventana:



Como podrás ver, por defecto se realizará una previsualización para el paso que tenemos seleccionado. Presioná **Quick Launch**. Aparece entonces una ventana con una muestra de la salida del Paso javascript.



Se muestran aquí los mensajes creados para cada una de las personas del archivo de entrada.

Si la salida es la esperada (nuevamente, debería serlo si seguiste el paso a paso), estamos listos para ejecutar la transformación. Presioná el botón **Ejecutar**.



En primer lugar aparecerá una ventana donde se pueden indicar, entre otros datos, los parámetros de ejecución para la transformación, y el nivel de log deseado. En este caso no vamos a modificar nada y directamente vamos a ejecutar presionando el botón **Ejecutar** al pie de la ventana.

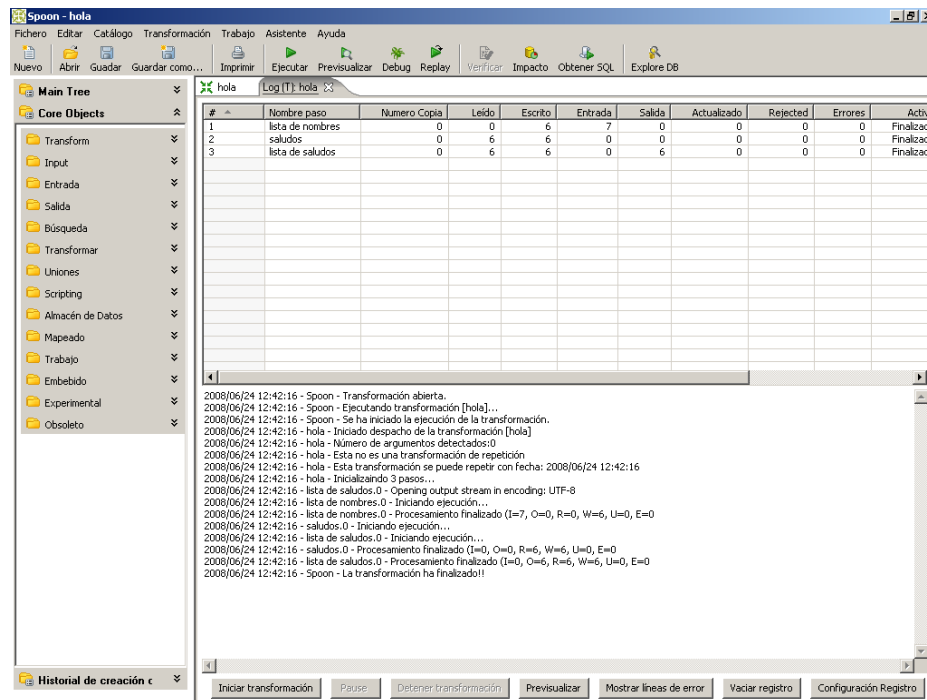
Inmediatamente vas a ver una nueva solapa en el espacio de trabajo: Es la **solapa de log** correspondiente al log de esta ejecución. (Habrás notado que la misma solapa apareció al hacer la previsualización).

La solapa de log tiene dos secciones:

En la parte superior, podemos ver, para cada paso de la transformación, un detalle de las operaciones realizadas. En particular observá estos datos:

- **Leído:** corresponde al número de filas que llegaron de pasos anteriores.
- **Escrito:** corresponde al número de filas de salida hacia los pasos siguientes.
- **Entrada:** número de líneas leídas de archivo o base de datos.

- Salida: número de líneas grabadas en archivo o base de datos.
- Errores: indica si hubo error en este paso. En caso que hubiese error, la fila del log se verá en rojo.



En la parte inferior podrás ver el paso a paso de la ejecución. El detalle mostrado dependerá del nivel de log establecido. Si prestás atención a este detalle, vas a ver reflejado lo que mencionamos más arriba: la ejecución asincrónica de los pasos.

Al final del texto de log verás la línea:

```
Spoon - La transformación ha finalizado!!
```

Si no hubo ningún error, ya podés buscar en el disco el archivo generado `hola.xml` y corroborar su contenido.

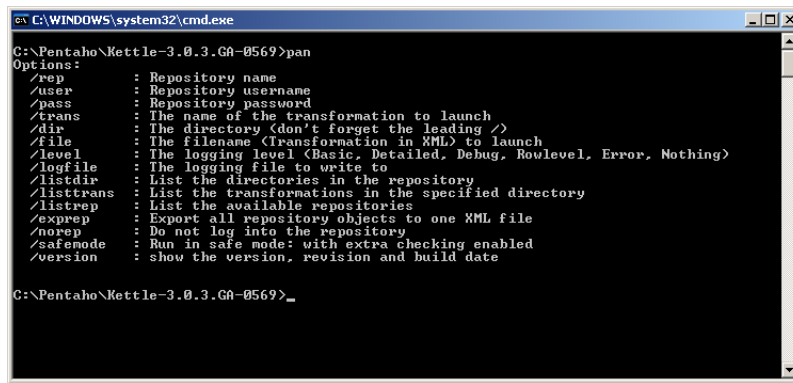
Pan

Pan es la herramienta que permite ejecutar Transformaciones desde consola. El script de ejecución es `pan.bat` (Windows) o `pan.sh` (otras plataformas), y se encuentra en el directorio de instalación de PDI.

Si ejecutás:

```
Pan
```

se presenta una descripción del comando y las opciones disponibles.



```
C:\WINDOWS\system32\cmd.exe
C:\Pentaho\Kettle-3.0.3.GA-0569>pan
Options:
/rep      : Repository name
/user     : Repository username
/pass    : Repository password
/trans   : The name of the transformation to launch
/dir      : The directory (don't forget the leading /)
/file     : The filename (transformation in XML) to launch
/level   : The logging level (Basic, Detailed, Debug, Rowlevel, Error, Nothing)
/logfile  : The logging file to write to
/listdir  : List the directories in the repository
/listtrans : List the transformations in the specified directory
/listrep  : List the available repositories
/exprep   : Export all repository objects to one XML file
/norep    : Do not log into the repository
/safemode : Run in safe mode: with extra checking enabled
/version  : show the version, revision and build date

C:\Pentaho\Kettle-3.0.3.GA-0569>
```

Para ejecutar la transformación que acabamos de crear, ejecutaremos el comando en su forma más simple:

```
Pan /file <ruta_de_la_carpeta_de_trabajos>/hola.ktr /norep
```

`/norep` es un comando para indicarle a la herramienta que no debe conectarse al catálogo.

`/file` precede al nombre de archivo correspondiente a la transformación a ejecutar.

`<ruta_de_la_carpeta_de_trabajos>` es la ruta de la carpeta Tutorial, por ejemplo:

- o `c:/Pentaho/Tutorial (Windows)`
- o `/home/PentahoUser/Tutorial`

El resto de las opciones, como por ejemplo el nivel de log, asume valores por defecto.

Luego de ingresar este comando, la transformación se ejecutará de la misma manera que desde el Spoon, pero escribirá el log en la misma ventana de ejecución, salvo que se direcciona el log a un archivo. El formato del log varía un poco pero básicamente la información es la misma que veíamos en la interfaz gráfica.

```
c:\WINDOWS\system32\cmd.exe

C:\Pentaho\Kettle-3.0.3.GA-0569>pan /file c:\Pentaho\Tutorial\hola.ktr /norep
INFO 24-06 12:44:01.741 <LogWriter.java:println:406> -Pan - Start of run.
2008/06/24 12:44:04:241 GMT-03:00 [INFO] DefaultFileReplicator - Using "C:\DOCUME~1\AR2135~1\CONFIG~
1\Temp\ufs_cache" as temporary files store.
INFO 24-06 12:44:04.787 <LogWriter.java:println:406> -hola - Iniciado despacho de la transformaci
n [hola]
INFO 24-06 12:44:04.787 <LogWriter.java:println:406> -hola - Nmero de argumentos detectados:0
INFO 24-06 12:44:04.787 <LogWriter.java:println:406> -hola - Esta no es una transformaci3n de repe
tici3n
INFO 24-06 12:44:04.819 <LogWriter.java:println:406> -hola - Esta transformaci3n se puede repetir
con fecha: 2008/06/24 12:44:04
INFO 24-06 12:44:04.819 <LogWriter.java:println:406> -hola - Inicializaindo 3 pasos...
INFO 24-06 12:44:04.819 <LogWriter.java:println:406> -lista de saludos.0 - Opening output stream i
n encoding: UTF-8
INFO 24-06 12:44:04.834 <LogWriter.java:println:406> -saludos.0 - Iniciando ejecuci3n...
INFO 24-06 12:44:04.834 <LogWriter.java:println:406> -lista de saludos.0 - Iniciando ejecuci3n...
INFO 24-06 12:44:04.959 <LogWriter.java:println:406> -lista de nombres.0 - Iniciando ejecuci3n...
INFO 24-06 12:44:04.959 <LogWriter.java:println:406> -lista de nombres.0 - Procesamiento finalizad
o <I=7, O=0, R=0, W=6, U=0, E=0>
INFO 24-06 12:44:05.444 <LogWriter.java:println:406> -saludos.0 - Procesamiento finalizado <I=0, O
=0, R=6, W=6, U=0, E=0>
INFO 24-06 12:44:05.444 <LogWriter.java:println:406> -lista de saludos.0 - Procesamiento finalizad
o <I=0, O=6, R=6, W=6, U=0, E=0>
INFO 24-06 12:44:05.522 <LogWriter.java:println:406> -hola - Transformaci3n finalizada.
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -Pan - Finished!
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -Pan - Start=2008/06/24 12:44:04.491, Stop=20
08/06/24 12:44:05.538
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -Pan - Processing ended after 1 seconds.
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -hola -
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -hola - Los procesos saludos'.0 han finalizad
o correctamente, se han procesado 6 lneas. < 6 lneas>
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -hola - Los procesos lista de saludos'.0 han
finalizado correctamente, se han procesado 6 lneas. < 6 lneas>
INFO 24-06 12:44:05.538 <LogWriter.java:println:406> -hola - Los procesos lista de nombres'.0 han
finalizado correctamente, se han procesado 0 lneas. < 0 lneas>
C:\Pentaho\Kettle-3.0.3.GA-0569>
```

Ya cumplimos lo prometido. Una transformaci3n simple que te permiti3 conocer los elementos de trabajo b3sicos de PDI.

Hola mundo, nuevamente!

Ahora que ya creamos y ejecutamos nuestra primera transformación, vamos a incorporarle algunas mejoras. Además de plantearnos como objetivo tener un trabajo de mejor calidad (por más chiquito que sea), esta segunda parte del tutorial nos permitirá introducirnos en algunos conceptos nuevos tan importantes como los vistos en la primera parte:

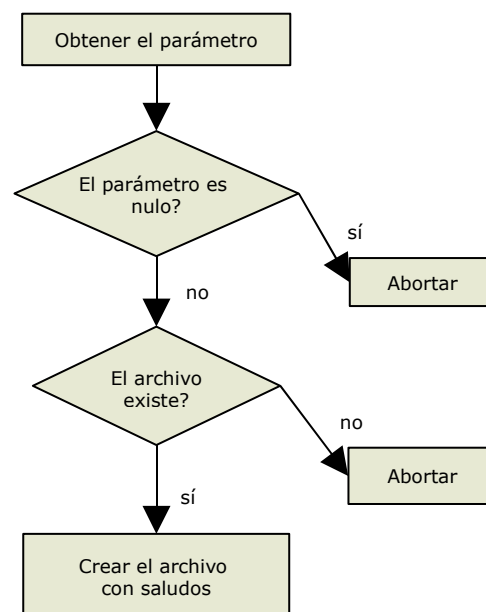
- Trabajos
- Entradas de Trabajo
- Saltos de Trabajos
- Uso de Parámetros e Información del Sistema
- Variables de Usuario
- Archivo de configuración kettle.properties
- Ejecución de Trabajos desde consola con la herramienta Kitchen.

Consigna

Estas son las mejoras que incorporaremos a nuestro ejemplo:

- El archivo de entrada lo buscaremos en una carpeta predefinida diferente a la carpeta donde están las transformaciones. El nombre del archivo en lugar de ser fijo, lo recibiremos como parámetro.
- Validaremos que el archivo exista (ejercicio: ejecutá la transformación anterior con un nombre de archivo inexistente y fijate cómo se comporta!)
- El nombre del archivo de salida lo haremos dependiente del nombre del archivo de entrada.

Dibujemos un pequeño diagrama con las tareas a realizar:



Este diagrama lo vamos a implementar con un Trabajo.

Antes de comenzar, vemos algunas definiciones nuevas:

Un **Trabajo** es un componente compuesto por **Entradas de Trabajo** enlazadas por medio de **Saltos**. Estas Entradas y Saltos están dispuestos de acuerdo al orden de ejecución esperado. Por eso se dice que un job está orientado al **control de flujo**.

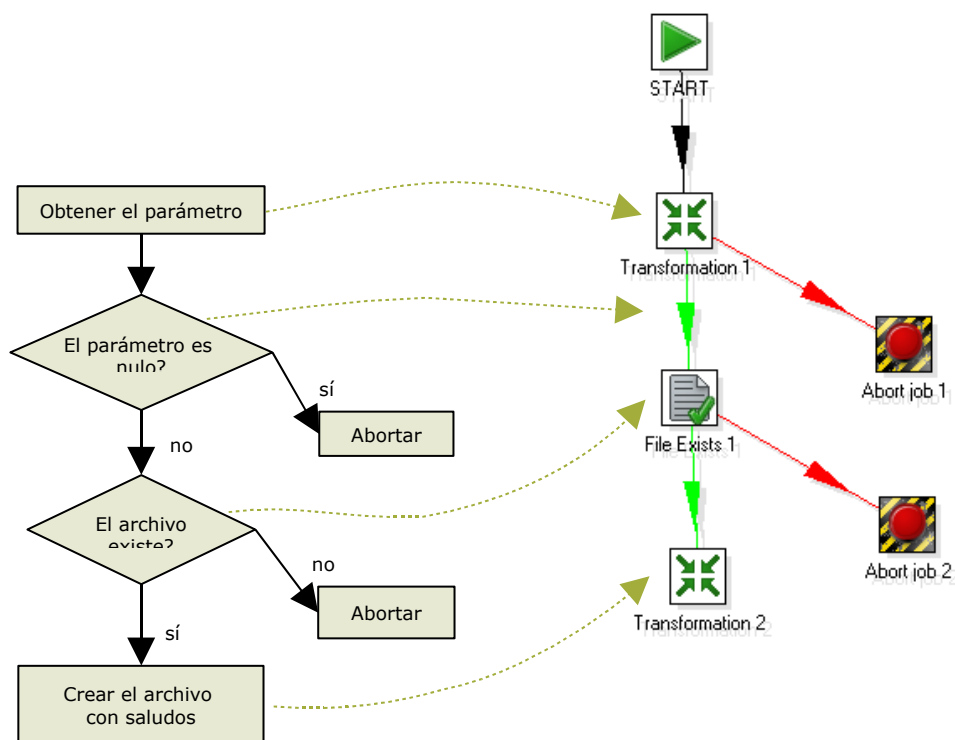
Una **Entrada de Trabajo** es la unidad de ejecución dentro de un Trabajo. Cada Entrada de Trabajo está pensada para cumplir una funcionalidad específica que va desde chequear la existencia de una tabla hasta hacer un envío de email.

Desde un Trabajo se puede ejecutar una Transformación o invocar otro Trabajo, esto es, "**Trabajo**" y "**Transformación**" son dos casos particulares de Entradas de Trabajo.

Un **Salto** es la representación gráfica que identifica la secuencia de ejecución entre dos Entradas de Trabajo.

Si bien un salto tiene sólo un origen y un destino, una Entrada de Trabajo puede ser alcanzada por más de un Salto y de ella puede salir también más de un Salto.

De la misma manera que hicimos con la transformación de la primera parte del tutorial, vamos a intentar establecer una correspondencia entre los elementos del diagrama, y las Entradas de Trabajo y los Saltos que vamos a crear.



De esta manera:

- La solicitud del parámetro será resuelta con una nueva transformación
- La verificación de la existencia del parámetro la haremos utilizando el resultado de la transformación anterior. La decisión de cuál será el paso a ejecutar, la haremos utilizando ejecución condicional.
- La verificación de la existencia del archivo la hará una Entrada de Trabajo específica para dicha funcionalidad.
- La ejecución de la tarea central del Trabajo la hará una variante de la transformación de la primera parte del tutorial.

Preparación del Ambiente

Para esta parte del tutorial, vamos a destinar para los archivos de entrada y salida una carpeta diferente. Creá entonces la carpeta Archivos en la ruta de tu preferencia.

Dentro de esta carpeta, copiá el archivo `lista.csv` o crea uno distinto, respetando el formato. Opcionalmente, cambiale el nombre.

Para no tener que escribir toda la ruta del archivo cada vez que necesitemos hacer referencia a ella, vamos a crear una variable que contenga esta información, y que esté siempre disponible. Para ello vamos a editar el archivo de configuración **kettle.properties**. Este archivo se encuentra en la carpeta

C:\Documents and Settings\\.kettle\ (Windows)

```
$HOME/.kettle (otras plataformas)
```

Y es creado junto con la carpeta `.kettle` la primera vez que se ingresa al Spoon.

Al final del archivo `kettle.properties`, agregá esta línea:

```
ARCHIVOS=<ruta_de_la_carpeta_creada>
```

Por ejemplo:

```
ARCHIVOS=c:/Pentaho/Archivos
0
ARCHIVOS=/home/PentahoUser/Archivos
```

Guardá el archivo.

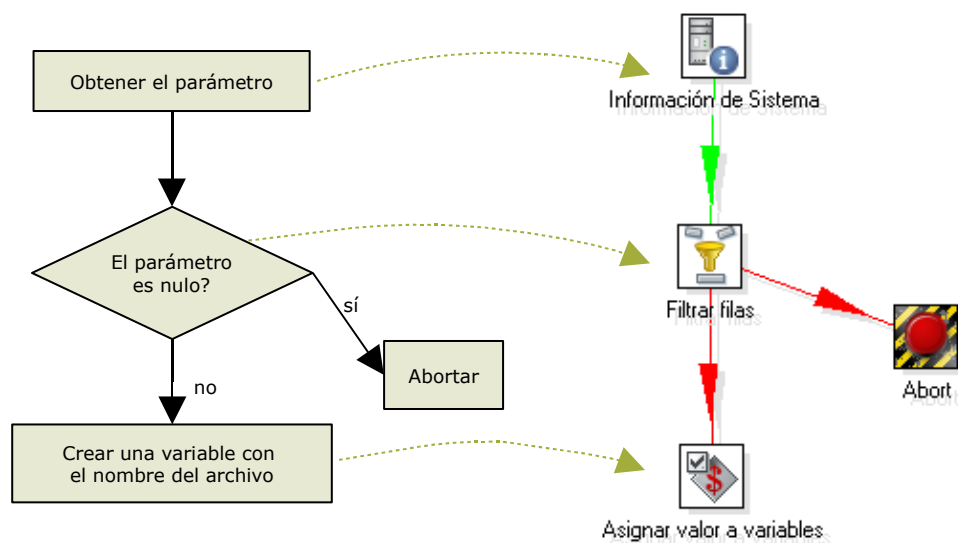
Spoon lee el archivo de configuración en el arranque, por lo tanto para que tu variable `ARCHIVOS` sea reconocida, reiniciá el Spoon.

Ahora estamos listos para comenzar. Resolveremos la consigna en tres partes:

- Crearemos la Transformación que pide el parámetro
- Modificaremos la Transformación `hola.ktr` para parametrizar los nombres de archivo de entrada y de salida.
- Construiremos el Trabajo según el diagrama anterior.

A. Creación de la transformación que pide el parámetro

Esta nueva transformación es muy simple. Este diagrama muestra las tareas involucradas y su correspondencia con Pasos de PDI:



Paso a Paso

1. Creación de la Transformación: Creá una nueva transformación, de la misma manera que hiciste en la primera parte del tutorial. Llamá a esta transformación "obtener_nombre_de_archivo".
2. Llevá a la pantalla de trabajo estos pasos, dales el nombre y unilos con Saltos acorde al diagrama.



Información de Sistema (categoría Entrada)



Filtrar Filas (categoría Transformar)



Abort (categoría Transformar)



Asignar valor a variables (categoría Trabajo)

3. Configuraré los pasos como se explica abajo.

a) Configuración del Paso Información de Sistema (categoría Entrada)

Este paso permite capturar información externa a la transformación, ya sea información del sistema como por ejemplo fecha actual, o parámetros ingresados desde la línea de comando. En este caso vamos a usar el Paso para tomar el primer y único parámetro y guardarlo en el campo "archivo". La ventana de configuración del paso tiene una grilla. En esta grilla, cada fila que completemos generará una nueva columna de datos y su contenido corresponderá a un dato proveniente del sistema.

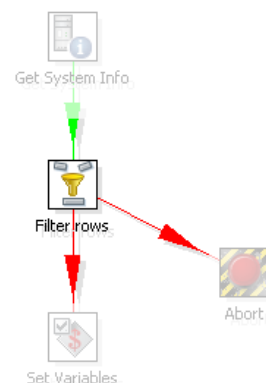
- Seleccioná el Paso con doble-click.
- En el primer casillero escribí bajo la columna **Nombre**, la palabra "mi_archivo".
- Al hacer click en el casillero de la misma fila, bajo la columna **Tipo**, se presentará una ventana con las opciones disponibles. Seleccioná "argumento de línea de comando 1".
- Hacé click en **Vale**.



b) Configuración del Paso Filtrar Filas (categoría Transformar)

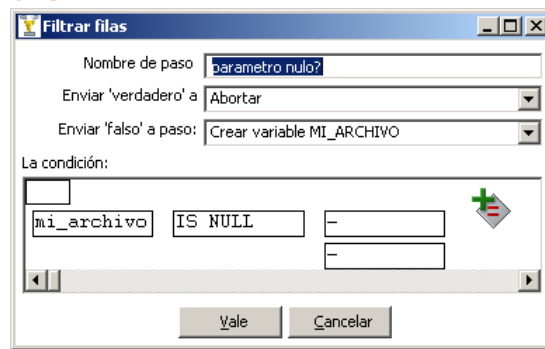
Este paso divide la salida en dos de acuerdo a una condición. Aquellos datos que cumplen la condición siguen un camino, y los que no lo cumplen, siguen otro.

- Seleccioná el Paso con doble-click
- Vamos a escribir la condición: En el cuadro **Field** seleccioná "mi_archivo" y el signo "=" reemplazalo por "IS NULL".
- En el cuadro "**Enviar Verdadero a**" elegí de la lista el Paso **Abortar**.
- En el cuadro "**Enviar Falso a Paso**" elegí de la lista el Paso **Establecer variable**



<screenshot>

- Hací click en **Vale**.



De esta manera, un parámetro en blanco va a llegar al paso **Abortar**. Un parámetro distinto de blanco va a llegar al paso **Establecer variable**.

c) Configuración del Paso **Abort** (categoría **Transformar**)

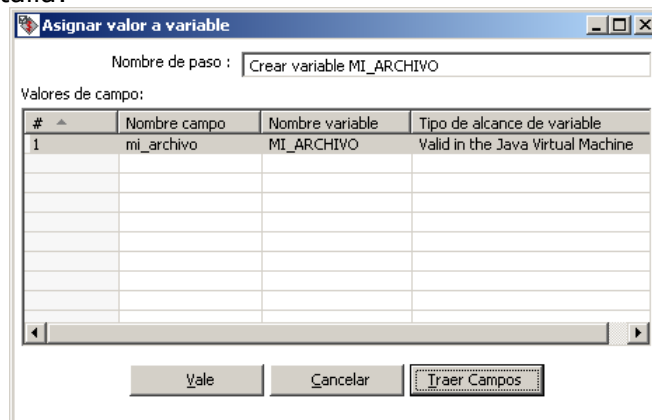
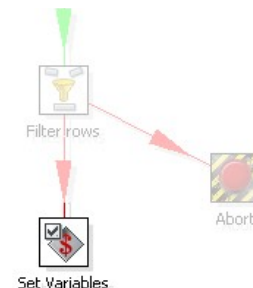
En realidad no hay nada para configurar en este paso. Si un dato llega acá, la transformación aborta. Esto ocasiona que la misma termine como Fallida, resultado que puede usarse (y de hecho lo usaremos) en el control de secuencia del Trabajo.

d) Configuración del Paso **Asignar valor a variables** (categoría **Trabajo**)

Este paso permite crear una variables y asignarles el valor de algunos de los datos de entrada del Paso.

La ventana de configuración del paso tiene una grilla. Cada fila de esta grilla se corresponde con una nueva variable.

- Seleccioná el Paso con doble-click.
- Presioná el botón **Traer Campos**. Aparecerá el único campo que tenemos: `mi_archivo`. El mismo nombre de campo, pero en mayúscula, se propone por defecto como nombre para la variable que estamos creando: `MI_ARCHIVO`. Dejá los datos que se proponen por defecto, como se muestran en esta pantalla:



- Hací click en **Vale**.

De esta manera tenemos creada la variable `MI_ARCHIVO`, que utilizaremos más adelante.

Ejecución

Para chequear el funcionamiento de esta transformación, presioná **Ejecutar**. En la primer ventana observá la grilla `Parámetros`. Esta grilla sirve para escribir los mismos datos que pasaríamos como parámetro al ejecutar desde consola. En la primer fila, bajo la columna `Valor`, escribí "lista". Luego presioná el botón **Ejecutar**.

Si observás el log, vas a encontrar una línea como esta:

```
Asignar valor a variables.0 - Set variable MI_ARCHIVO to value [lista]
```

Hagamos otra prueba. Nuevamente presioná **Ejecutar**, pero esta vez no ingreses parámetros. En el log vamos a encontrar esta salida:

```
Abort.0 - Row nr 1 causing abort : []  
Abort.0 - Aborting after having seen 1 rows.
```

Además el Paso **Abort** en la parte superior de la solapa de log, aparece marcando un error. Esto nos indica que la transformación finaliza con error, como lo previsto.

B. Parametrización de la transformación hola.ktr

Vamos a modificar la transformación `hola` para parametrizar los nombres de los archivos de entrada y salida. Si el parámetro es `xxx`, la transformación leerá el archivo `xxx.csv` y generará el archivo `xxx_con_saludos.xml`. Y como último cambio, vamos a agregar un filtro para descartar los nombres que pudieran venir en blanco.

Paso a Paso

- Abrí la pantalla de configuración del paso **CSV File Input**
- **Filename**: Borrá el contenido de esta caja de texto, y presioná Ctrl-Espacio para ver la lista de variables. Vas a ver en la lista la variable que creamos en el archivo de configuración: `ARCHIVOS`. Seleccionala, y al lado escribí el nombre de la variable que creamos en la transformación anterior, de modo que el texto completo quede así:

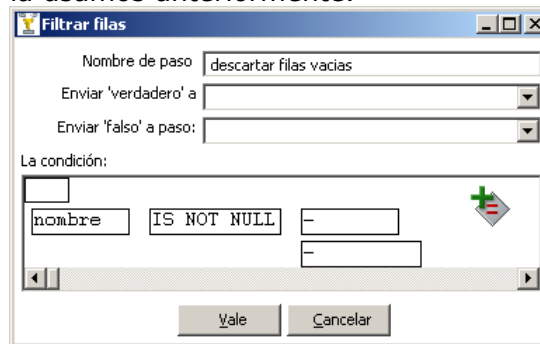
```
${ARCHIVOS}/${MI_ARCHIVO}.csv
```

- Presioná **Vale**.
- Abrí la pantalla de configuración del paso **Salida XML**
- **Filename**: Borrá el contenido de esta caja de texto, y reemplazalo por este:

```
${ARCHIVOS}/${MI_ARCHIVO}_con_saludos.xml
```

- Presioná **Vale**.
- Arrastrá a la pantalla de trabajo un Paso **Filtrar filas**, hasta que el Paso esté sobre el Salto que sale de **CSV Input**. Cuando veas que la línea que el Salto se vuelve más ancha, soltá el botón del mouse. Habrás enganchado el nuevo Paso a la secuencia de la Transformación. Escribí la siguiente condición: En el cuadro **Field** seleccioná "nombre" y el signo "=" reemplazalo por "IS NOT NULL". Los cuadros **Enviar Verdadero a** y **Enviar Falso a Paso** dejalos en blanco. De esta manera, van a seguir al siguiente Paso únicamente los

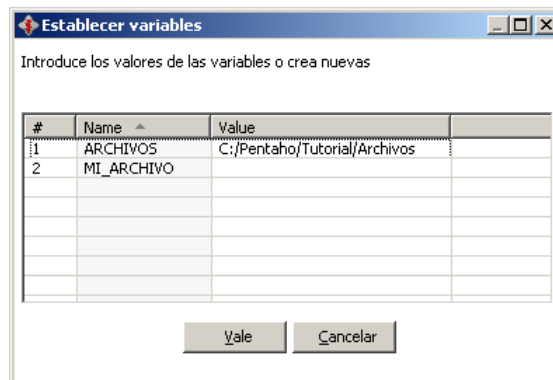
nombres que cumplan la condición, es decir, aquellos que no sean nulos. Esto es una variante en la forma de utilizar el tipo de Paso **Filtrar Filas**, respecto de como la usamos anteriormente.



- Presioná **Vale**.
- Para no perder la transformación original, presioná "Guardar Como" y ponedle el nombre "hola_con_parametros". Se creará el archivo hola_con_parametros.ktr

Ejecución

Para chequear los cambios realizados a la transformación, necesitamos que la variable `MI_ARCHIVO` exista y tenga un valor asignado. Como esta transformación es independiente de la Transformación que crea la variable, para poder ejecutarla tendremos que crear la variable manualmente. Para eso, seleccioná "Establecer variables de entorno" en el menú "Editar". Aparecerán las variables existentes. Al final de la lista, agregá la variable `MI_ARCHIVO`, y ponedle como contenido el nombre del archivo que creaste, o lista si dejaste el existente. No pongas la extensión `.csv`.



Ahora presioná **Ejecutar**.

Vas a ver que en la pantalla de ejecución la variable aparece en la lista de variables existentes.

Presioná el botón **Ejecutar**, y dejá que la transformación haga su tarea.

Luego verificá que en la carpeta de archivos se haya creado el archivo

```
<mi_archivo>_con_nombres.xml
```

donde `<mi_archivo>` corresponde al nombre del archivo que utilizamos como entrada. También asegurate que el contenido sea el esperado.

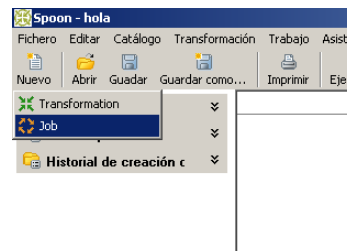
C. Construcción del trabajo principal

El último paso de esta parte del tutorial es la construcción del trabajo principal.

Paso a Paso

1) Creación del Trabajo:

- Seleccioná **Nuevo** → **Trabajo**. Alternativamente podés crear un trabajo desde el menú **Fichero** → **Nuevo** → **Trabajo** o presionar **<Ctrl-Alt-N>**. Gran parte de la pantalla es ahora ocupada por el espacio de trabajo donde colocaremos las Entradas de Trabajo y los Saltos.
- Seleccioná la opción **Trabajo** → **Configuración**. Se presenta una ventana donde podés especificar propiedades generales del trabajo. En este caso solamente escribí para el trabajo un nombre y una descripción.
- Presioná el botón **Guardar**. Guardá el trabajo en la carpeta **Tutorial** con el nombre **hola**. Se habrá creado el archivo **hola.jkb**.



- #### 2) Armado del esqueleto del Trabajo con Entradas de Trabajo y Saltos:
- Así como para las transformaciones teníamos una paleta de Pasos, aquí tenemos una paleta de Entradas de Trabajo. Una diferencia que notarás al principio es que estas Entradas no están agrupadas en Categorías¹. Vamos a armar el Trabajo:
- Llevá a la pantalla estas Entradas de Trabajo, dales el nombre y unilas con Saltos acorde al diagrama.



Start



Transformation x 2



File Exists

- Llevá a la pantalla las siguientes Entradas de Trabajo, dales nombre y unilas al diagrama.



Abort Job x 2

Vas a ver que los Saltos se pintaron de rojo. Luego vamos a explicar por qué.

- Configurá los pasos como se explica abajo.

a) Configuración de la Primera Entrada de Trabajo Transformation

La entrada "Transformation" tiene el objetivo de ejecutar una transformación. Vamos a configurar esta entrada para que ejecute la transformación que obtiene el parámetro.

- Hacé doble click sobre la Entrada de Trabajo para seleccionarla. Aparece la pantalla de configuración para este tipo de Entradas de Trabajo.

¹ En la versión 3.1.0 – aún no liberada – las Entradas de Trabajo aparecen agrupadas en Categorías como General, Mail, File management, entre otras.

- **Transformation filename:** Acá debemos indicar el nombre del archivo de la transformación "obtener_nombre_de_archivo.ktr". Como las Transformaciones y los Trabajos se encuentran en la misma carpeta, vamos a usar como referencia la ruta donde está guardado el Trabajo.

Ingresa este texto:

```
${Internal.Job.Filename.Directory}/obtener_nombre_de_archivo.ktr
```

La variable pueden ser escrita manualmente, o elegida de la lista de variables. De la misma manera, el nombre del archivo se puede escribir, o buscar con el botón **Browse**.

- Presioná **Vale**.

b) Configuración de la Segunda Entrada de Trabajo Transformation

Vamos a configurar esta entrada para que ejecute la transformación que realiza la principal función de este Trabajo.

- Hacé doble click sobre la Entrada de Trabajo para seleccionarla. Aparece la pantalla de configuración para este tipo de Entradas de Trabajo.
- **Transformation filename:** En este caso, ingresá el nombre de la otra transformación:

```
${Internal.Job.Filename.Directory}/hola_con_parametros.ktr
```

- Presioná **Vale**

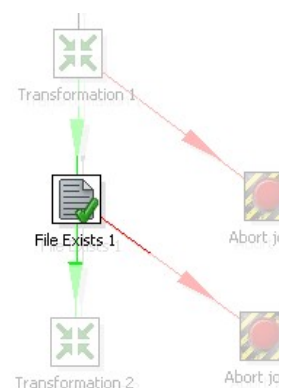
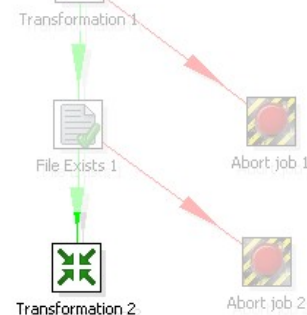
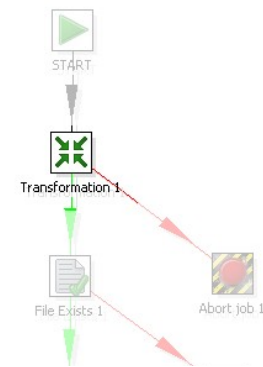
c) Configuración de File Exists

La Entrada **File Exists** verifica la existencia de un archivo. Si el archivo existe, la Entrada tiene un resultado exitoso, y si no, falla.

- Hacé doble click sobre la Entrada de Trabajo para seleccionarla. Aparece la pantalla de configuración para este tipo de Entradas de Trabajo.
- **Filename:** Aquí debemos indicar el nombre completo del archivo cuya existencia queremos verificar. Es el mismo que escribimos en la transformación hola modificada:

```
${ARCHIVOS}/${MI_ARCHIVO}.csv
```

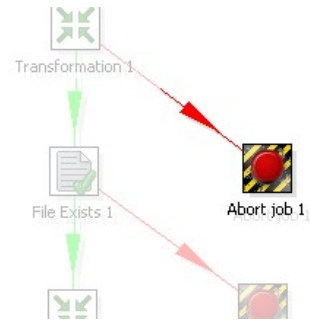
Recordá que la variable `${ARCHIVOS}` está definida en el archivo `kettle.properties`, y la variable `${MI_ARCHIVO}` se crea en la Entrada de Trabajo anterior a esta.



d) Configuración de la Primera Entrada de Trabajo **Abort Job**

La primera Entrada **Abort Job** se ejecutará cuando no se haya indicado un parámetro. Lo que vamos a configurar acá es el mensaje con el motivo por el cual se aborta el Trabajo.

- En **Message** escribí este texto:
No se indicó el parámetro con el nombre del archivo



e) Configuración de la Segunda Entrada de Trabajo **Abort Job**

Esta Entrada **Abort Job** se ejecutará cuando el archivo no exista.

- En **Message** escribí este texto:
El archivo indicado no existe (\$
{ARCHIVOS}/{MI_ARCHIVO}.csv)

En tiempo de ejecución, la herramienta reemplazará las variables por su valor, mostrando por ejemplo, este mensaje:

El archivo indicado no existe
(c:/Pentaho/Archivos/lista.csv)



f) Configuración de los Saltos

La ejecución de una Entrada de Trabajo puede realizarse:

- incondicionalmente: se ejecuta siempre
- sólo cuando la ejecución de la Entrada de Trabajo anterior fue exitosa
- sólo cuando la ejecución de la Entrada de Trabajo anterior fue fallida

Esta ejecución se indica gráficamente por medio de colores en los Saltos:

- Un salto en **negro** indica que la siguiente Entrada de Trabajo se ejecuta incondicionalmente
- Un salto en **verde** indica que la siguiente Entrada de Trabajo se ejecuta si la anterior fue exitosa
- Un salto en **rojo** indica que la siguiente Entrada de Trabajo se ejecuta si la anterior falló

Por el orden en que creamos y enlazamos las Entradas de Trabajo, todos los saltos quedaron configurados tal como lo necesitamos, esto es:

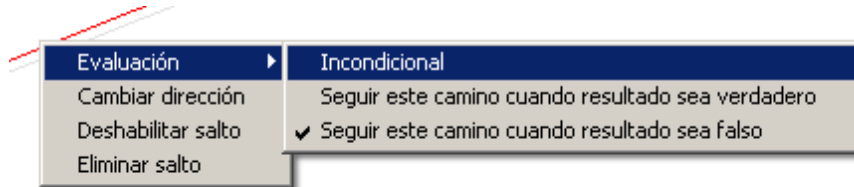
- La primera transformación se ejecuta siempre (El salto que sale de **Start** hacia ella está en **negro**)
- Si la transformación que toma el parámetro no encuentra un parámetro, es decir, falla, el control pasa por el salto **rojo** hacia la Entrada de Trabajo **Abort**.

Si la transformación es exitosa, el control pasa por el salto **verde** que conduce a la Entrada de Trabajo **File Exists**.

- Si el archivo no existe, es decir, la verificación de la existencia del archivo falla, el control pasa por el salto **rojo** hacia la segunda Entrada de Trabajo **Abort**.
- Si la verificación es exitosa, el control pasa por el salto **verde** que conduce a la segunda Entrada de Trabajo **Transformation**.

Si quisiéramos cambiar la condición para la ejecución de una Entrada de Trabajo, estos son los pasos:

- Seleccioná el Salto que llega a esta Entrada de Trabajo
- Presioná botón derecho, con lo cual aparecerá un menú contextual.
- En el menú seleccioná **Evaluación** seguido de una de las tres condiciones que se muestran.



Cómo funciona?

Cuando ejecutamos un job, la ejecución está dada por el orden de las Entradas de Trabajo, el sentido de los Saltos, y la condición bajo la cual se ejecuta o no una Entrada. La ejecución es **secuencial**. Una Entrada de Trabajo debe terminar su ejecución para que pueda iniciarse la ejecución de la o las Entradas de Trabajo que le siguen en el diagrama.

En la práctica, un Trabajo puede ser un salvavidas para subsanar problemas de secuencialidad en las Transformaciones. Si se requiere que una parte de una Transformación termine completamente antes de que comience el resto, una forma de resolverlo puede ser desglosando la Transformación en dos Transformaciones independientes, e invocarlas desde un Trabajo en forma secuencial.

Ejecución

Para ejecutar el Trabajo, en primer lugar necesitamos indicar el parámetro. Como el único lugar donde lo utilizamos es en la transformación

Obtener_nombre_de_archivo, ya que después solamente usamos la variable donde guardamos ese dato, vamos a indicar el parámetro de la siguiente forma:

- Seleccioná con doble click la transformación "obtener_nombre_de_archivo"
- En la ventana que se presenta, hay una grilla titulada **Fields**. En la primer fila de la grilla escribí el nombre del archivo creado en la carpeta Archivos (sin extensión).
- Presioná **Vale**.

Ya quedó establecido el parámetro.

Ahora presioná el botón **Ejecutar**.

Se presenta una ventana con datos generales relativos a la ejecución del Trabajo. Presioná el botón "Ejecutar".

De inmediato se abre la **solapa de log de Trabajos**.

La solapa de log de los Trabajos, al igual que la de las Transformaciones, se encuentra dividida en dos partes:

La parte superior muestra las Entradas de Trabajo. Para cada Entrada de Trabajo que se ejecutó, se muestra entre otros datos, el resultado de la ejecución. Como ya mencionamos, la ejecución de las Entradas es Secuencial, por lo tanto, si una Entrada termina fallida, no veremos en el log las siguientes Entradas, ya que las mismas ni siquiera comienzan a ejecutarse.

En la parte inferior se muestra el detalle del log indicando el comienzo y fin de las entradas del Trabajo. En el caso de las Entradas correspondientes a Transformaciones (de las cuales tenemos dos) se incluye también el detalle de log de las Transformaciones propiamente dichas.

Cuando veas al final del log el texto:

```
Spoon - Trabajo ha terminado.
```

Significa que ya se encuentra disponible el archivo generado. Si el archivo de entrada era: `lista.csv`

el archivo de salida debería llamarse: `lista_con_saludos.xml` y estar situado en la misma carpeta.

Buscalo en la carpeta Archivos y corrobora su contenido.

Cambia el nombre del parámetro, reemplazándolo por un nombre que no se corresponda con un archivo de la carpeta Archivos. Ejecutá el Trabajo. Vas a ver cómo el Trabajo aborta, y aparece en el log el mensaje

```
Abort - El archivo indicado no existe  
seguido del nombre del archivo que pusiste como parámetro.
```

Por último borrá el contenido del parámetro, y ejecutá por tercera vez. En este caso el Trabajo también aborta, y en el log tenemos el siguiente mensaje:

```
Abort - No se indicó el parámetro con el nombre del archivo
```

que es el comportamiento que esperábamos en este caso.

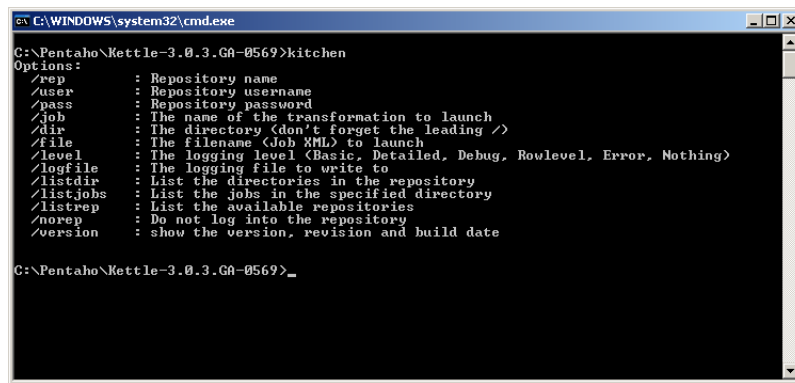
Kitchen

Kitchen es la herramienta que permite ejecutar Trabajos desde consola. El script de ejecución es `Kitchen.bat` (Windows) o `Kitchen.sh` (otras plataformas), y se encuentra en el directorio de instalación de PDI.

Si ejecutás:

```
Kitchen
```

se presenta una descripción del comando y las opciones disponibles.



```
C:\WINDOWS\system32\cmd.exe
C:\Pentaho\Kettle-3.0.3.GA-0569>kitchen
Options:
/rep      : Repository name
/user     : Repository username
/pass    : Repository password
/job      : The name of the transformation to launch
/dir      : The directory (don't forget the leading /)
/file     : The filename (Job XML) to launch
/level    : The logging level (Basic, Detailed, Debug, Rowlevel, Error, Nothing)
/logfile  : The logging file to write to
/listdir  : List the directories in the repository
/listjobs : List the jobs in the specified directory
/listrep  : List the available repositories
/norep    : Do not log into the repository
/version  : show the version, revision and build date

C:\Pentaho\Kettle-3.0.3.GA-0569>_
```

Para ejecutar el Trabajo que acabamos de crear, ejecutaremos el comando en su forma más simple:

```
<Kitchen /file <ruta_de_la_carpeta_de_trabajos>/hola.kjb <par> /norep
```

/norep es un comando para indicarle a la herramienta que no debe conectarse al catálogo.

/file precede al nombre de archivo correspondiente al Trabajo a ejecutar.

<ruta_de_la_carpeta_de_trabajos> es la ruta de la carpeta Tutorial, por ejemplo:

- o c:/Pentaho/Tutorial (Windows)
- o /home/PentahoUser/Tutorial

<par> es el parámetro que el Trabajo espera. Recordá que corresponde al nombre del archivo sin la extensión csv.

El resto de las opciones, como por ejemplo el nivel de log, asume valores por defecto.

Luego de ingresar este comando, el trabajo se ejecutará de la misma manera que desde el Spoon, pero escribirá el log en la misma ventana de ejecución, salvo que se direcciona el log a un archivo. El formato del log varía un poco pero básicamente la información es la misma que veíamos en la interfaz gráfica.

Intentá ejecutar el Trabajo:

- sin parámetros
- con un parámetro inválido (no correspondiente con un archivo existente)
- con un parámetro válido

y verificá que todo funcione de la forma esperada.

Y llegamos al final de esta segunda parte. Construimos un Trabajo que le aplicó una serie de mejoras a la transformación de la primera parte, y nos permitió conocer más elementos de trabajo de PDI, imprescindibles para el día a día de los que trabajamos con esta herramienta.

Conclusión

Ahora que ya viste un panorama de la aplicación, estás listo para explotar toda su capacidad en tus propios proyectos.

Sin duda a partir de ahora te será de utilidad recurrir a las siguientes fuentes:

- **Manuales de Usuario**

No hay (aún) manuales de usuario en español. Los manuales en inglés se encuentran en la carpeta `\docs\English` dentro de la carpeta de instalación de PDI.

- **Ejemplos**

En la carpeta `Samples` dentro de la carpeta de instalación de PDI hay bastantes ejemplos. Los mismos pueden servir para entender cómo funcionan algunos pasos, o para tomar como base para crear tus propios Trabajos y Transformaciones.

- **Foro**

El foro de Pentaho es una fuente interesante de conocimiento. Ahí podés ver los problemas con los que se han enfrentado otros usuarios de PDI y la forma en que se pueden resolver.

Asimismo podés ser vos mismo quien que plantee dudas que se te presentan, o ayudes a otros a resolver sus dificultades una vez que has adquirido experiencia con la herramienta.

El foro de PDI se puede acceder directamente desde aquí:

<http://forums.pentaho.org/forumdisplay.php?f=69>

Además de los mencionados aquí, en la página de bienvenida del Spoon vas a encontrar algunos enlaces adicionales.

Espero que este tutorial haya sido útil para que puedas comenzar ya a trabajar con PDI.

Comentarios, críticas y sugerencias son bienvenidos,
Suerte con PDI!