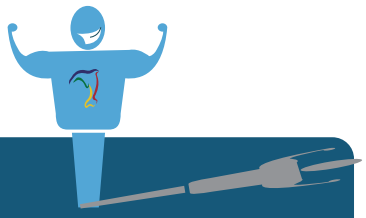
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

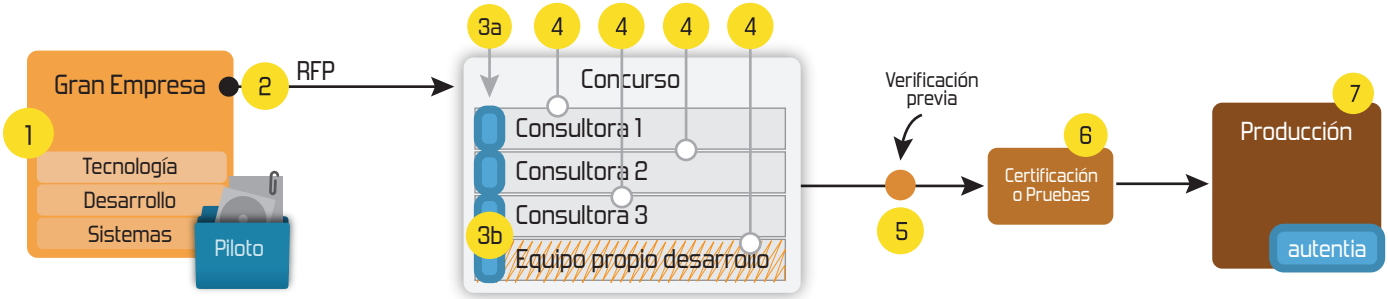
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Alejandro Pérez García

Alejandro es socio fundador de Autentia y nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Ingeniero en Informática y Certified ScrumMaster

Seguir a @alejandropgarci 1,034 seguidores

Si te gusta lo que ves, puedes contratarle para darte ayuda con soporte experto, impartir **cursos presenciales** en tu empresa o para que **realicemos tus proyectos como factoría** (Madrid).
Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación.

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2014-03-04 Tutorial visitado 4 veces Descargar en PDF

Kettle no es una tetera, es la herramienta de ETL de Pentaho!

Creación: 28-02-2014

Índice de contenidos

1. Introducción

Kettle es una herramienta de la suite de Pentaho, de hecho también se la denomina *PDI* o *Pentahoâ€™s Data Integration*.

Kettle es una herramienta de las que se denominan *ETL* (Extract - Transform - Load). Es decir, una herramienta de *Extracción* de datos de una fuente, *Transformación* de esos datos, y *Carga* de esos datos en otro sitio.

Estas tareas son típicas en procesos de migración, integración con terceros, explotación de Big Data, ... y en general se podría decir que son necesarias en casi cualquier proyecto mediano o grande. Por eso Kettle nace con la intención de facilitarnos este trabajo, de forma que no tengamos que entrar en el detalle de la implementación de como se hace cada una de estas tareas, sino que simplemente especificamos qué es lo que queremos hacer. Por eso en muchos sitios se califica a este tipo de herramientas, *herramientas de metadatos*, ya que trabajan a nivel de definición diciendo qué hay que hacer, pero no el detalle del cómo se hace, éste queda oculto a nuestros ojos, lo cual resulta muy interesante en la mayoría de los casos.

2. Entorno

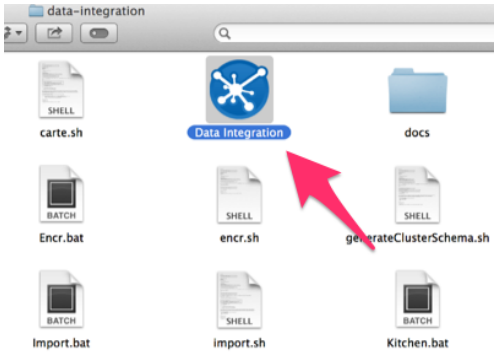
El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.3 GHz Intel i7, 16GB 1600 Mhz DDR3, 500GB Flash Storage).
- NVIDIA GeForce G7 750M
- Sistema Operativo: Mac OS X Lion 10.9.1
- Kettle, PDI Community Edition, 5.0.1.A-stable

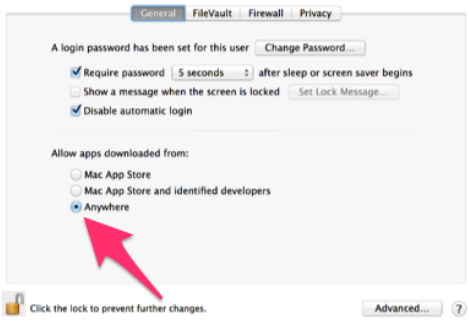
3. Instalación

Lo descargamos de la página Data Integration - Kettle.

Bajamos el zip y lo descomprimimos. En Mac han preparado un launcher con el nombre *Data Integration*. Podemos hacer doble click sobre él y se abrirá **Spoon** que es un entorno gráfico que nos permite trabajar con Kettle.



En mi caso la primera en la frente, porque me salía un mensaje de error que decía algo así como: *"Data Integration" is damaged and can't be opened. You should move it to the Trash.* Esto tiene que ver con el sistema de seguridad de Mavericks, y con la firma de la aplicación, que no es reconocida por el sistema, así que lo que hice fue irme a la configuración de seguridad del sistema y permitir la ejecución de cualquier aplicación.



Ahora volvemos a ejecutar y nos debería dar el típico mensaje de que hemos descargado la aplicación de Internet y nos pregunta si la queremos ejecutar, le decimos que sí.

Ahora ya podemos/debemos dejar las restricciones de seguridad como las tuviéramos antes de cambiarlas en el paso anterior. Y ya no volveremos a tener problemas si ejecutamos de nuevo la aplicación.

4. Nuestro primer trabajo de transformación

Bueno al lío. Si hemos conseguido ejecutar la aplicación veremos que la primera pantalla es:

» Depurar Tomcat en remoto.

Últimos Tutoriales del Autor

» Crea todo un entorno de máquinas virtuales con un solo comando, gracias a Vagrant

» ¿Endemoniado por lo lento que es Gradle en el arranque? Aprende a controlar su Daemon, y vuela!

» Cómo instalar Gradle, herramienta de automatización de builds

» Ponle color a tu maven

» Lanzando nuestros tests de jasmine-node con IntelliJ IDEA

Últimas ofertas de empleo

2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

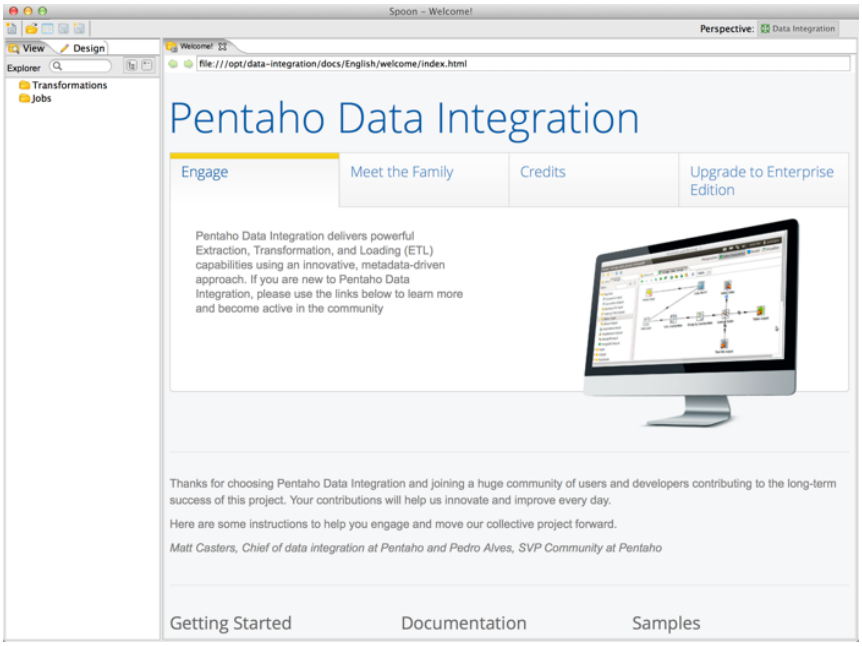
2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

2011-07-06
Otras Sin catalogar - LUGO.

Esta pantalla nos permite definir un repositorio donde guardar todas nuestras *recetas* de transformación, podemos darle tranquilamente al botón de *Cancel* y continuar con la aplicación, guardando en este caso nuestro trabajo en ficheros con la extensión **`ktr`**

La siguiente pantalla que veremos será un consejo. Os recomiendo que los leáis porque os dan muchos trucos de como usar la herramienta. Una vez cerrado el consejo, por fin llegamos a la pantalla principal de la herramienta, donde podemos acceder a gran cantidad de la documentación.



Nuestro primer trabajo va a ser sencillo (podríamos decir que es el *Hola Mundo*, de los ETLs), convertir el CSV:

```
nombre, apellido
Roberto, Canales
Jose María, Toribio
Juan, Alonso
Alfonso, Blanco
Francisco Javier, Martínez
Jose Manuel, Sánchez
```

en el XML:

```
<Rows>
    <Row>
        <msg>Hola, Roberto!</msg>
    </Row>
    <Row>
        <msg>Hola, Jose María!</msg>
    </Row>
    <Row>
        <msg>Hola, Juan!</msg>
    </Row>
    <Row>
        <msg>Hola, Alfonso!</msg>
```

```
        </Row>
        <Row>
            <msg>Hola, Francisco Javier!</msg>
        </Row>
        <Row>
            <msg>Hola, Jose Manuel!</msg>
        </Row>
    </Rows>
```

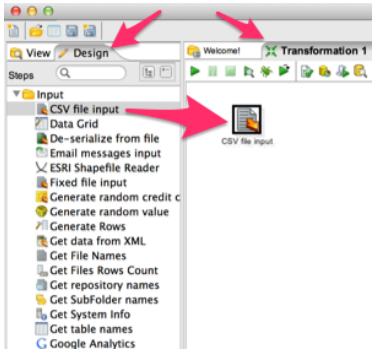
Así que vamos a coger el contenido del CSV y copiarlo en un fichero `names.csv`.

4.1. Leyendo el fichero CSV de entrada

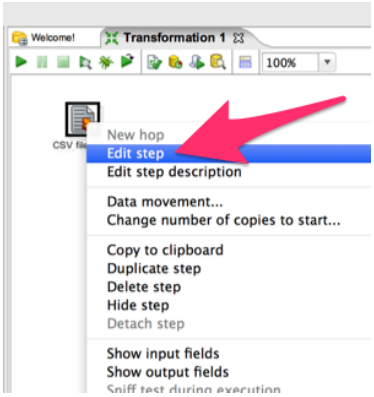
Ahora desde Spoon hacemos *File --> New --> Transformation* (o `Cmd + N`). Con esto creamos un proceso de transformación (*Transformation*) donde iremos creando los pasos (*Steps*) necesarios para convertir la entrada en la salida que esperamos. Un Step es la unidad mínima de trabajo de una Transformación, y se encarga de realizar una tarea específica, por ejemplo leer un fichero, hacer una validación, transformar un dato, escribir en una base de datos, ... En la paleta de la izquierda podemos encontrar multitud de ellos, organizados por categorías, y por cierto, muy útil el buscador que encontraréis justo arriba.

Estos pasos lo iremos uniendo mediante saltos (*Hops*) que nos sirven para ir uniendo los distintos Steps, y definir así el flujo de la información. Un Hop tiene un sólo origen y un sólo destino, pero un Step sí puede tener varios Hops tanto de entrada como de salida.

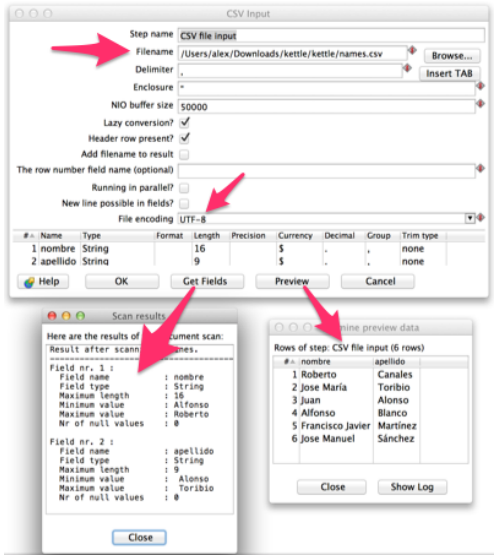
Al crear la transformación el área de la izquierda nos habrá cambiado a la pestaña *Design*, aquí pinchamos y arrastramos el Step *CSV file input*, de forma que debería quedarnos algo similar a la imagen.



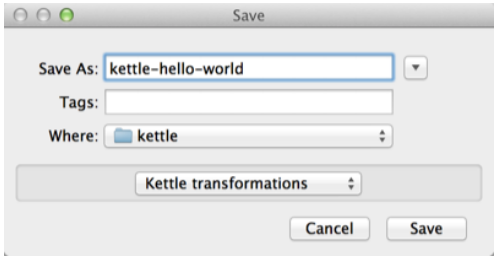
Ahora hacemos botón derecho sobre el icono del CSV y pinchamos sobre *Edit step*, para configurar este paso (también podemos hacer `doble click` sobre el paso para editarlo).



Tenemos que indicar el fichero de entrada `names.csv`, y el encoding en el que está guardado el fichero. Luego si queremos podemos dar a los botones de *Get Fields* o *Preview* para ver si está recuperando correctamente los datos.

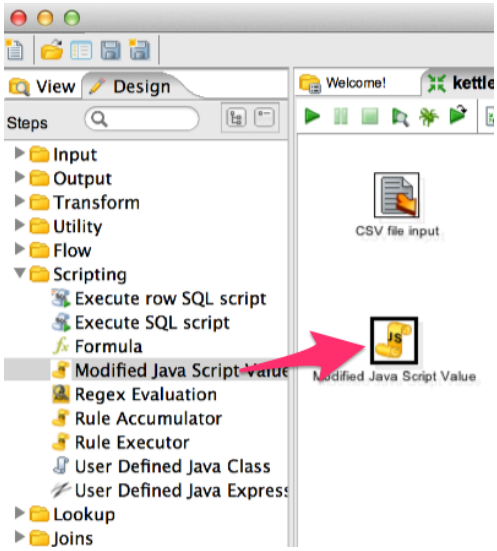


Para no perder lo que tenemos hasta ahora hacemos *File --> Save* (o `Cmd + S`) y guardamos nuestro progreso con el nombre que queramos.



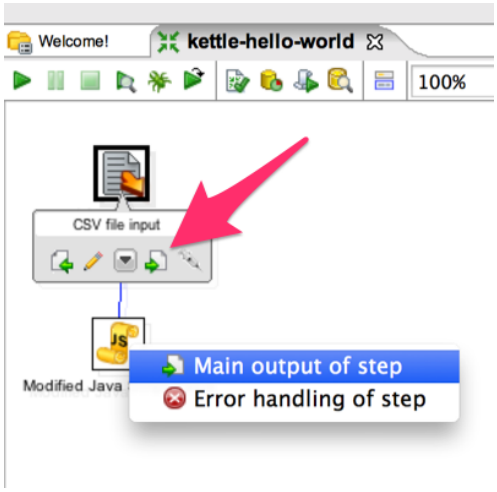
4.2. Trasformando los valores con JavaScript

Ahora vamos a usar el Step *Modified Java Script Value*, para preparar el mensaje que queremos volcar en el XML de salida.

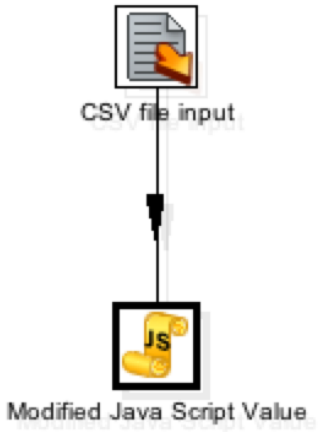


Con este nombre ya os podéis hacer a la idea de para que vale este Step y de la potencia que puede tener. Ya que podemos escribir código para manipular los datos como queramos. Ojo porque mi recomendación sería que busquéis siempre el Step más específico para hacer la tarea que queréis. ¡No os lo hagáis todo a mano!

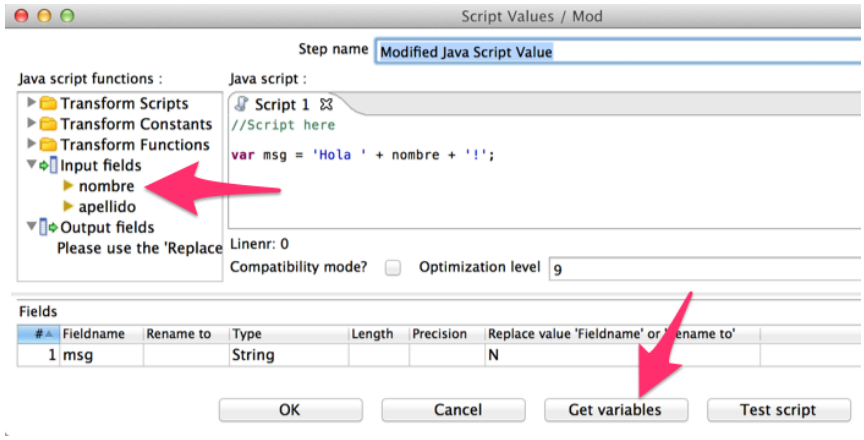
Antes de editar el Step que acabamos de añadir, vamos a unirlo con el que ya teníamos. Para ello hacemos click sobre el Step que lee el CSV y nos aparecerá un pequeño menú abajo. Pinchamos sobre la el icono con la flecha verde saliendo, y sin soltar, arrastramos hasta el Step que acabamos de añadir.



Seleccionamos *Main output step*, y nos debería quedar algo como:



Ahora sí, hacemos doble click sobre el *Step Modified Java Script Value* y editamos sus propiedades.



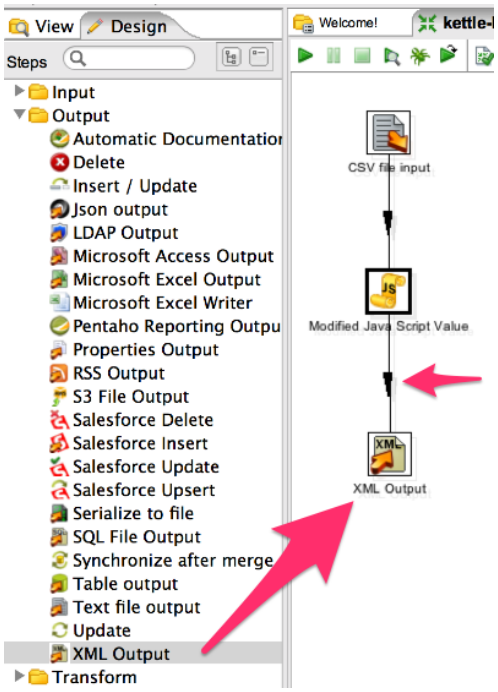
Vemos como hemos puesto un pequeño JavaScript que compone la cadena que queremos como salida, y la guardamos en la variable `msg`. Para componer esta cadena tenemos que usar el campo de entrada `nombre`, este lo podemos escribir o podemos hacer `doble click` sobre el nombre del campo en el desplegable de la izquierda.

Luego es muy importante que definamos cual será la salida de este Step, para ello lo hacemos en el listado de abajo, que podemos rellenar a mano, o simplemente pulsar el botón de *Get Variables*.

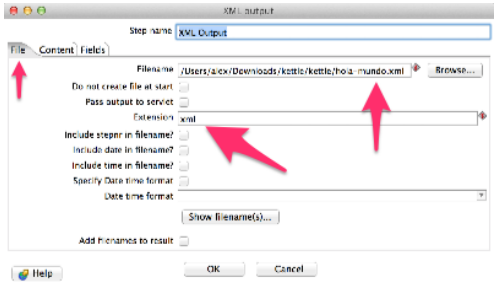
También disponemos de otro botón *Test script* que nos permite probar el script con valores de prueba autogenerados. Lo cual resulta muy interesante.

4.3. Escribiendo el XML

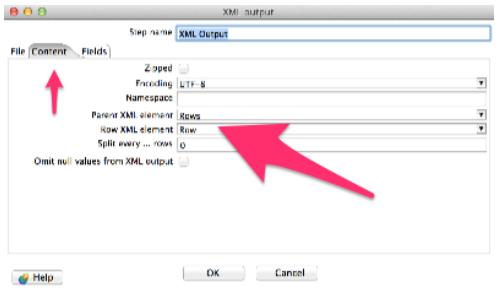
Ya sólo nos queda escribir el XML de salida. Para ello en la paleta de la izquierda, en la categoría de *Output*, encontraremos el Step *XML Output*, lo pinchamos y lo arrastramos a nuestro Transformation, e igual que antes unimos el Step de modificación de datos con el que acabamos de añadir. Debería quedarnos algo como la siguiente imagen.



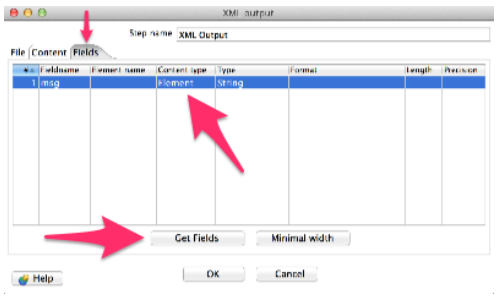
Ahora hacemos `doble click` sobre este último Step para editar sus propiedades. En la primera pestaña de *File* indicamos cual será el fichero de salida.



Ahora nos vamos a la segunda pestaña *Content*. Aquí no vamos a tocar nada, pero es para que veáis que es donde se define el XML: un elemento padre `Rows`, que englobará a todos los registros, y luego cada registro que procesemos irá en su propio elemento `Row`.



Y por último la tercera pestaña *Fields*. Esta sí es importante ya que es donde definimos con qué información queremos trabajar. Damos al botón *Get Fields* y veremos como nos aparecen los tres campos: `nombre` y `apellido` que vienen del primer Step, y `msg` que viene del segundo Step. Borramos `nombre` y `apellido`, ya que no nos interesan y no los queremos en la salida. Y para `msg` definimos el *Content Type*, como *Element* para que en el XML aparezca un elemento con este nombre.



4.4. Ejecutando la Transformation

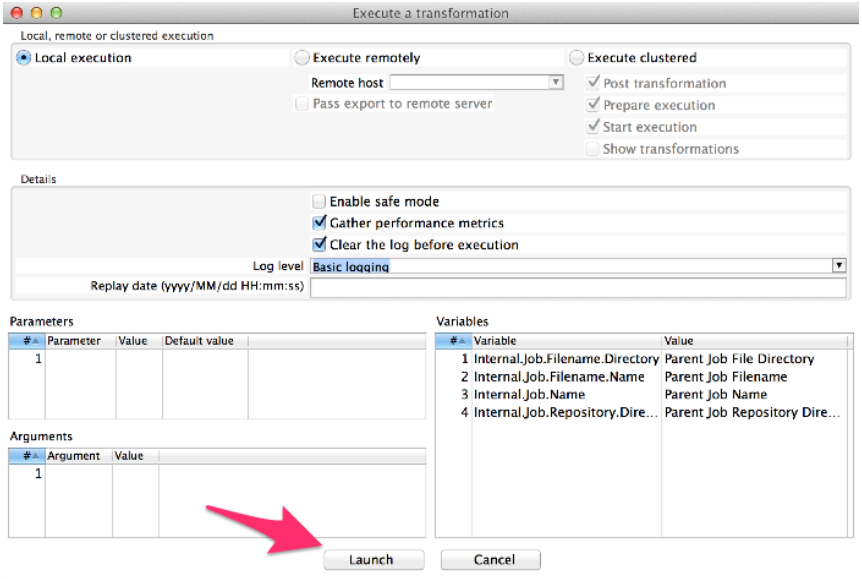
Ya estamos listos para ejecutar nuestro proceso de transformación.

Lo primero que tenemos que tener en cuenta es que **en una Transformation, todos los Steps se ejecutan de forma simultánea**. WTF?!?!?!?! No nos pongamos nerviosos que esto precisamente es lo que le da potencia a Kettle. Imaginaos que queremos procesar grandes volúmenes de datos, no tendría sentido hacer cada paso uno por uno, sería muy lento y necesitaríamos muchos recursos. Kettle tiene la idea de *stream* o flujo, de forma que Kettle no necesita tener cargados todos los registros para procesarlos, sino que los va procesando y pasando por cada Step según los va leyendo de la entrada. Además esto nos permite distribuir los Steps en un cluster de forma que podemos escalar horizontalmente si el proceso de transformación es muy pesado.

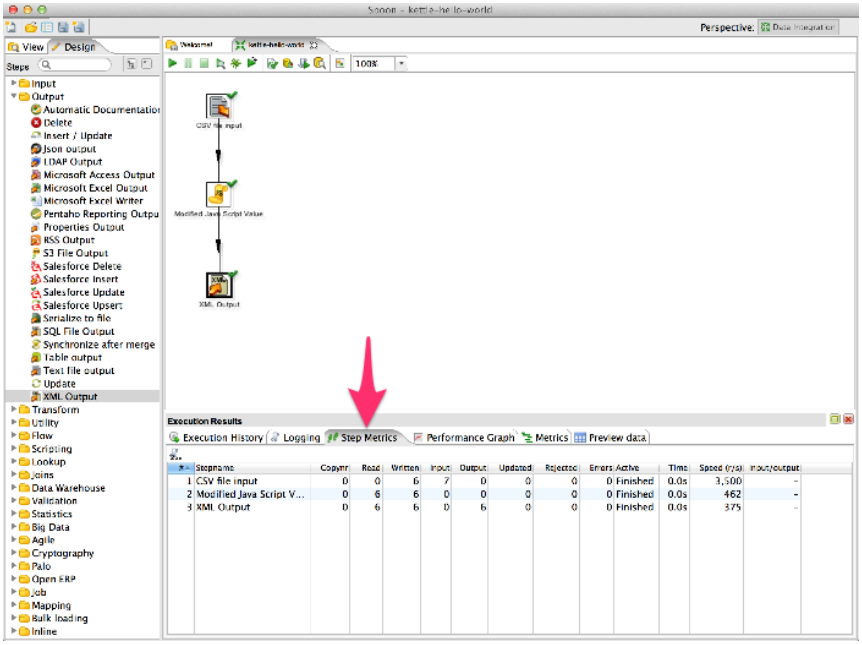
Por todo esto el hecho de que se ejecuten en paralelo es más que conveniente, pero simplemente hay que tenerlo en cuenta mientras diseñamos nuestra transformación para evitarnos sorpresas innecesarias ;)

Antes de ejecutar la transforamción conviene verificar que todo es correcto, para ello hacemos *Action --> Verify* (o `F11`). Spoon se encargará así de comprobar que la transformación es sintácticamente correcta, ver si tenemos Steps inalcanzables, ...

Si hemos verificado que todo es correcto, podemos ejecutar la transformación haciendo *Action --> Run* (0 `F9`). Veremos como nos aparece un panel donde podemos configurar ciertos aspectos de la ejecución, por ejemplo si queremos hacer la ejecución remoto o en cluster.



Damos al botón de Launch y como resultado de la ejecución deberíamos ver algo como:



De forma que podemos ver las estadísticas y los logs de la ejecución.

Y por supuesto podemos/debemos ver que se ha escrito el fichero XML. En mi caso he obtenido el fichero `hola-mundo.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Rows>
    <Row><msg>Hola Roberto&#x21;</msg> </Row>
    <Row><msg>Hola Jose Mar&#xed;a&#x21;</msg> </Row>
    <Row><msg>Hola Juan&#x21;</msg> </Row>
    <Row><msg>Hola Alfonso&#x21;</msg> </Row>
    <Row><msg>Hola Francisco Javier&#x21;</msg> </Row>
    <Row><msg>Hola Jose Manuel&#x21;</msg> </Row>
</Rows>
```

donde se puede ver como Kettle ha tenido la amabilidad de hasta codificarme las tildes :D

4.5. Ejecución de un Transformation por línea de comandos

Todo lo que hemos visto en el apartado anterior tiene muy buena pinta, pero no podemos depender de un entorno gráfico para ejecutar las transformaciones, esto iría totalmente en contra de la idea de automatizar procesos.

Para la ejecución en línea de comandos disponemos de **Pan**. Esta herramienta es un simple script (`.sh` en Unix, Linux, Mac, y `.bat` en Windows) que se encuentra en el mismo directorio que Spoon, y que nos permite lanzar en línea de comandos las Transformations que hemos diseñado gráficamente con Spoon.

Para lanzar la tranformación que hemos preparado basta con ejecutar:

```
$ cd <el directorio donde habíamos descomprimido kettle>
$ ./pan.sh -file <directorio donde hemos guardado la transformación>/kettle-hello-world.k
```

Obtendremos una salida similar a:

```
alex@ciclo:/opt/data-integration$ ./pan.sh -file /Users/alex/Downloads/kettle/kettle/kettle-hello-world.ktr
2014/02/27 14:00:40 - Pan - Start of run.
2014/02/27 14:00:40 - kettle-hello-world - Dispatching started for transformation [kettle-hello-world]
2014/02/27 14:00:40 - XML Output.0 - Opening output stream in encoding: UTF-8
2014/02/27 14:00:41 - CSV file input.0 - Header row skipped in file '/Users/alex/Downloads/kettle/kettle/names.csv'
2014/02/27 14:00:41 - CSV file input.0 - Finished processing (I=7, O=0, R=0, W=6, U=0, E=0)
2014/02/27 14:00:41 - Modified Java Script Value.0 - Optimization level set to 9.
2014/02/27 14:00:41 - Modified Java Script Value.0 - Finished processing (I=0, O=0, R=6, W=6, U=0, E=0)
Attempting to load ESAPI.properties via file I/O.
Attempting to load ESAPI.properties as resource file via file I/O.
Not found in 'org.owasp.esapi.resources' directory or file not readable: /opt/data-integration/ESAPI.properties
Not found in SystemResource Directory/resourceDirectory: .esapi/ESAPI.properties
Not found in 'user.home' (/Users/alex) directory: /Users/alex/esapi/ESAPI.properties
Loading ESAPI.properties via file I/O failed. Exception was: java.io.FileNotFoundException
Attempting to load ESAPI.properties via the classpath.
SUCCESSFULLY LOADED ESAPI.properties via the CLASSPATH from '/ (root)' using current thread context class loader!
SecurityConfiguration for Validator.ConfigurationFile not found in ESAPI.properties. Using default: validation.properties
Attempting to load validation.properties via file I/O.
Attempting to load validation.properties as resource file via file I/O.
Not found in 'org.owasp.esapi.resources' directory or file not readable: /opt/data-integration/validation.properties
Not found in SystemResource Directory/resourceDirectory: .esapi/validation.properties
Not found in 'user.home' (/Users/alex) directory: /Users/alex/esapi/validation.properties
Loading validation.properties via file I/O failed.
Attempting to load validation.properties via the classpath.
validation.properties could not be loaded by any means. fail. Exception was: java.lang.IllegalArgumentException: Failed to load
SecurityConfiguration for Logger.LogServerIP not either "true" or "false" in ESAPI.properties. Using default: true
2014/02/27 14:00:41 - XML Output.0 - Finished processing (I=0, O=6, R=6, W=6, U=0, E=0)
2014/02/27 14:00:41 - Pan - Finished!
2014/02/27 14:00:41 - Pan - Start=2014/02/27 14:00:40.465, Stop=2014/02/27 14:00:41.554
2014/02/27 14:00:41 - Pan - Processing ended after 1 seconds.
2014/02/27 14:00:41 - kettle-hello-world -
2014/02/27 14:00:41 - kettle-hello-world - Step CSV file input.0 ended successfully, processed 6 lines. ( 6 lines/s)
2014/02/27 14:00:41 - kettle-hello-world - Step Modified Java Script Value.0 ended successfully, processed 6 lines. ( 6 lines/s)
2014/02/27 14:00:41 - kettle-hello-world - Step XML Output.0 ended successfully, processed 6 lines. ( 6 lines/s)
```

5. Consiguiendo que nuestra Transformation no sea tan rígida gracias a los Jobs

Justo en el punto anterior estaba hablando de la importancia de poder automatizar los procesos, y sí, eramos capaces de ejecutar la transformación desde la línea de comandos, pero de forma totalmente rígida porque el fichero de entrada y salida son fijos, y no tenemos ningún tipo de control de error, por ejemplo que pasa si el fichero de entrada no existe.

En este punto vamos a ver como podemos hacer la transformación sea un poco más flexible y admita parámetros para configurar su comportamiento o distintos flujos de ejecución.

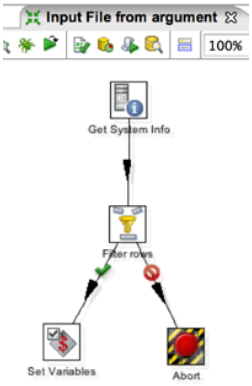
Para ello vamos a introducir un nuevo concepto, el de *Job* (trabajo). Mientras que una Transformation es un conjunto de pasos fijos. Un Job nos permite definir distintos flujos de ejecución, y en función de esos flujos llamar a unas Transformations y otras.

Una *Job Entry* es la unidad de ejecución de un Job (al igual que el Step lo era de la Transformation). Una Entry puede ser desde comprobar la existencia de un fichero, hasta el envío de un email, y por supuesto la ejecución de una Transformation, o incluso de otro Job.

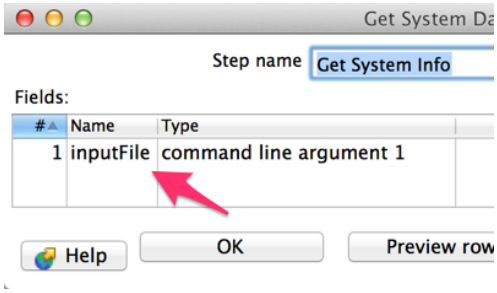
Además hay que destacar que mientras todos los Steps de una Transformation se ejecutan a la vez, las Entry de un Job se ejecutan según el flujo definido, de forma que hasta que no termina una Entry, no empieza la siguiente. **No hay paralelismo entre los Entry de un Job**.

5.1. Transformation para leer el fichero de entrada como parámetro

Vamos preparar una Transformation que se encargue de este trabajo, así que hacemos `Cmd + N` y creamos una nueva con el siguiente aspecto.



Get System Info de la categoría *Input* nos permite leer argumentos de entrada.

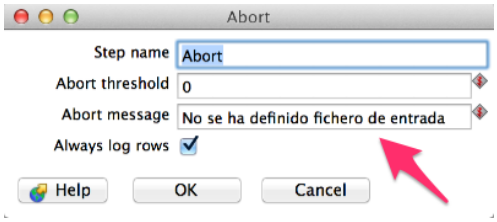


Vemos como hemos configurado el nombre del campo como `inputFile`, donde se guardará lo que venga en el primer argumento de entrada.

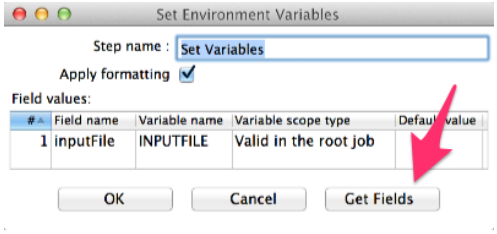
Filter rows, de la categoría *Flow*, nos permite cambiar el flujo de ejecución en función de si una condición es cierta o falsa. Lo vamos a usar para comprobar si el argumento de entrada es nulo o no.

Vemos como si la condición es cierta continuamos el flujo normal, y si es falsa abortamos el trabajo.

El Abort también está en la categoría *Flow*. Aquí sólo destacamos el mensaje de error que hemos añadido.

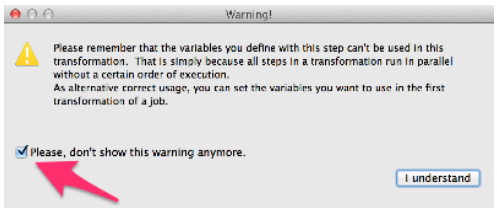


Por último el *Set Variable*, de la categoría *Job*. Este es muy importante ya que lo que hace es guardar el campo, que hemos definido antes, en una variable para que esté disponible para el resto de Steps.



En esta ocasión simplemente hemos dado al botón *Get Fields* para que el se encargue de dar de alta la variable con los valores por defecto.

La primera vez que guardemos este Step, nos saldrá el siguiente mensaje de alerta.

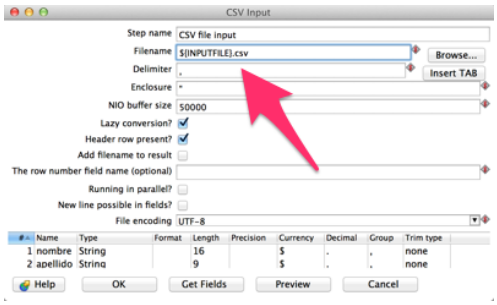


Simplemente nos avisa de que tengamos cuidado cuando cuando usemos este Step ya que, como todos los Steps se ejecutan en paralelo, no tenemos garantía de que la variable esté definida cuando la queremos usar. Para evitar esto lo que hacemos es controlas el flujo de ejecución con el Job y así garantizar que las variables se han definido antes de ser usadas.

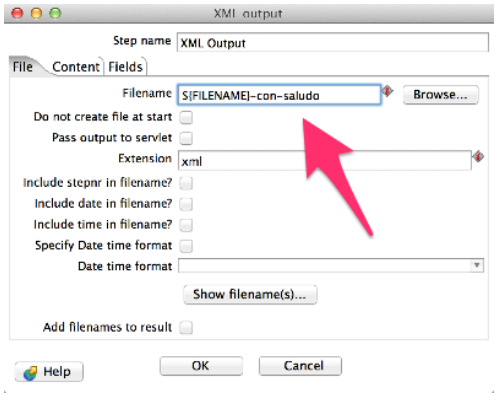
5.2. Modificando la primera Transformation para usar variables

En la primera Transformation que habíamos hecho, editamos el Step *CSV Input*, y donde habíamos puesto la ruta a fuego del

fichero, ponemos el nombre de la variable que hemos definido en el paso anterior. En mi caso `${INPUTFILE}.csv` (nótese que el nombre del fichero vendrá sin extensión y se la añadimos en el Step).



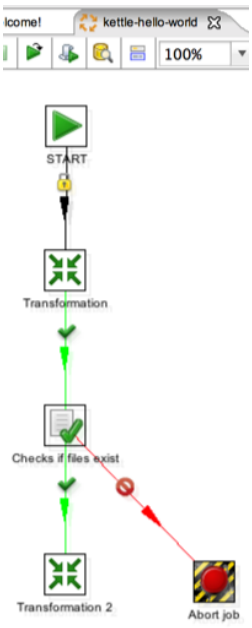
Ahora nos vamos al Step *XML Output* y hacemos lo mismo para cambiar el fichero de salida. Poniéndole el nombre `${INPUTFILE}-con-saludo` (nótese que aquí no añadimos la extensión, ya que la añade el propio Step).



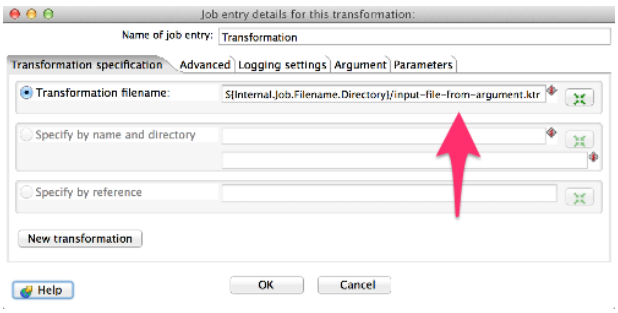
5.6. Creando el Job que lo gestionará todo

Para crear un Job hacemos *File --> New --> Job* (o `Alt + Cmd + N`).

Arrastraremos colocaremos los siguientes elementos:

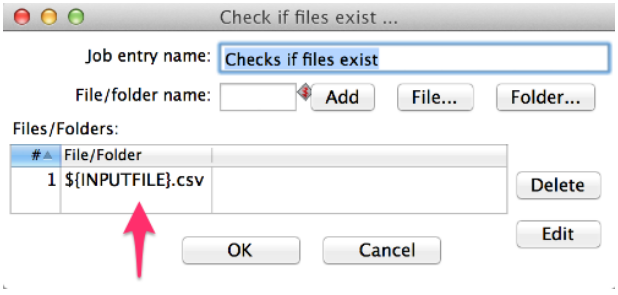


En *Transformation* hacemos referencia a la transformación que lee el argumento de entrada: `input-file-from-argument.ktr`.



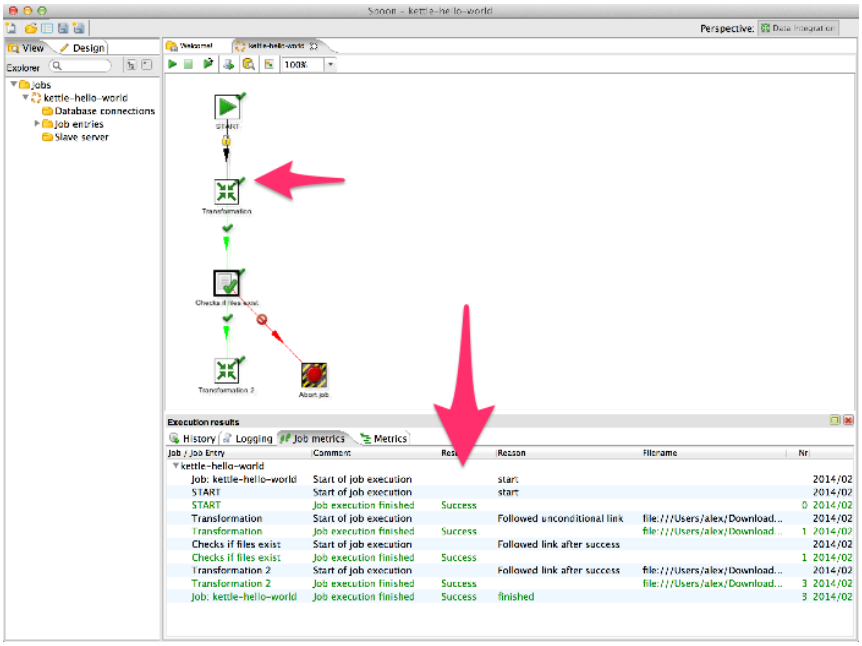
Con Transformation 2, hacemos lo mismo pero haciendo referencia a `kettle-hello-world.ktr`

Con Checks if files exist, hacemos referencia a la variable que hemos definido en la primera transformación.



5.7. Ejectuando el Job

Podemos hacer *Action --> Run* (o `F9`), y veremos algo como:



Podemos observar que en el propio diseño aparecen unos pequeños *ticks* verdes indicando que cada Entry se ha ejecutado correctamente. Además abajo también podemos ver el resultado de la ejecución. Y por supuesto deberíamos comprobar que hemos obtenido el correspondiente fichero de salida.

5.8. Ejecución de un Job por línea de comandos

En esta ocasión la herramienta para ejecutar Jobs en línea de comandos es **Kitchen**, e igual que antes es un script (`.sh` en Unix, Linux, Mac, y `.bat` en Windows) que se encuentra en el mismo directorio que Spoon.

Podemos probarlo con:

```
$ cd <el directorio donde habíamos descomprimido kettle>
$ ./kitchen.sh -file <directorio donde hemos guardado la transformación>/kettle-hello-wor
```

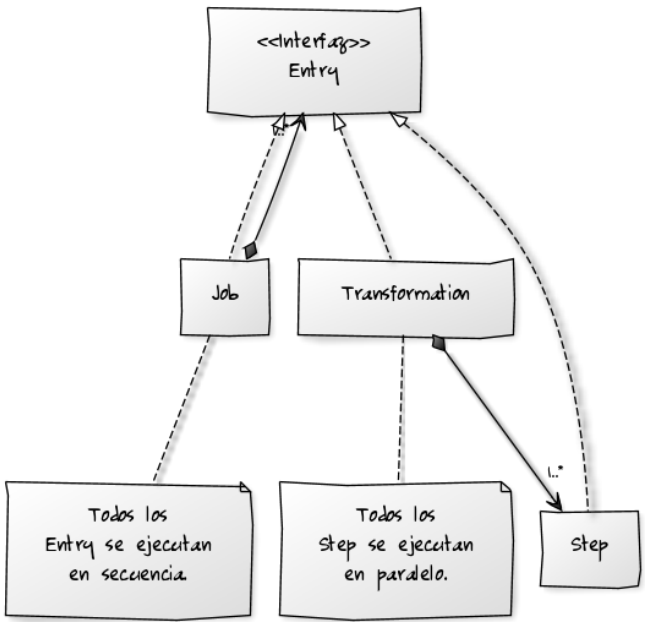
Obtendremos algo similar a:

```
alex@ciclo:/opt/data-integration$ ./kitchen.sh -file ~/Downloads/kettle/kettle/kettle-hello-world.kjb ~/Downloads/kettle/k
2014/02/27 19:25:04 - Kitchen - Start of run.
2014/02/27 19:25:05 - kettle-hello-world - Start of job execution
2014/02/27 19:25:05 - kettle-hello-world - Starting entry [Transformation]
2014/02/27 19:25:05 - Transformation - Loading transformation from XML file [file:///Users/alex/Downloads/kettle/kettle/in
2014/02/27 19:25:05 - input-file-from-argument - Dispatching started for transformation [input-file-from-argument]
2014/02/27 19:25:05 - Get System Info.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2014/02/27 19:25:05 - Set Variables.0 - Setting environment variables...
2014/02/27 19:25:05 - Set Variables.0 - Set variable INPUTFILE to value [/Users/alex/Downloads/kettle/kettle/names]
2014/02/27 19:25:05 - Filter rows.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2014/02/27 19:25:05 - Set Variables.0 - Finished after 1 rows.
2014/02/27 19:25:05 - Set Variables.0 - Finished processing (I=0, O=0, R=1, W=1, U=0, E=0)
2014/02/27 19:25:05 - kettle-hello-world - Starting entry [Checks if files exist]
2014/02/27 19:25:05 - kettle-hello-world - Starting entry [Transformation 2]
2014/02/27 19:25:05 - Transformation 2 - Loading transformation from XML file [file:///Users/alex/Downloads/kettle/kettle/
2014/02/27 19:25:05 - kettle-hello-world - Dispatching started for transformation [kettle-hello-world]
2014/02/27 19:25:05 - XML Output.0 - Opening output stream in encoding: UTF-8
2014/02/27 19:25:05 - CSV file input.0 - Header row skipped in file '/Users/alex/Downloads/kettle/kettle/names.csv'
2014/02/27 19:25:05 - CSV file input.0 - Finished processing (I=7, O=0, R=0, W=6, U=0, E=0)
2014/02/27 19:25:05 - Modified Java Script Value.0 - Optimization level set to 9.
2014/02/27 19:25:05 - Modified Java Script Value.0 - Finished processing (I=0, O=0, R=6, W=6, U=0, E=0)
Attempting to load ESAPI.properties via file I/O.
Attempting to load ESAPI.properties as resource file via file I/O.
Not found in 'org.owasp.esapi.resources' directory or file not readable: /opt/data-integration/ESAPI.properties
Not found in SystemResource Directory/resourceDirectory: .esapi/ESAPI.properties
Not found in 'user.home' (/Users/alex) directory: /Users/alex/esapi/ESAPI.properties
Loading ESAPI.properties via file I/O failed. Exception was: java.io.FileNotFoundException
Attempting to load ESAPI.properties via the classpath.
SUCCESSFULLY LOADED ESAPI.properties via the CLASSPATH from '/ (root)' using current thread context class loader!
SecurityConfiguration for Validator.ConfigurationFile not found in ESAPI.properties. Using default: validation.properties
Attempting to load validation.properties via file I/O.
Attempting to load validation.properties as resource file via file I/O.
Not found in 'org.owasp.esapi.resources' directory or file not readable: /opt/data-integration/validation.properties
Not found in SystemResource Directory/resourceDirectory: .esapi/validation.properties
Not found in 'user.home' (/Users/alex) directory: /Users/alex/esapi/validation.properties
Loading validation.properties via file I/O failed.
Attempting to load validation.properties via the classpath.
validation.properties could not be loaded by any means. fail. Exception was: java.lang.IllegalArgumentException: Failed to
SecurityConfiguration for Logger.LogServerIP not either "true" or "false" in ESAPI.properties. Using default: true
2014/02/27 19:25:05 - XML Output.0 - Finished processing (I=0, O=6, R=6, W=6, U=0, E=0)
2014/02/27 19:25:05 - kettle-hello-world - Finished job entry [Transformation 2] (result=[true])
2014/02/27 19:25:05 - kettle-hello-world - Finished job entry [Checks if files exist] (result=[true])
2014/02/27 19:25:05 - kettle-hello-world - Finished job entry [Transform      n] (result=[true])
2014/02/27 19:25:05 - kettle-hello-world - Job execution finished
2014/02/27 19:25:05 - Kitchen - Finished!
2014/02/27 19:25:05 - Kitchen - Start=2014/02/27 19:25:04.071, Stop=2014/02/27 19:25:05.847
2014/02/27 19:25:05 - Kitchen - Processing ended after 1 seconds.
```

6. Conclusiones

El tutorial ha quedado un poco largo, pero es muy sencillo, casi todo pantallas y configuración por defecto. Con esto nos damos cuenta de lo útil que pueden resultar este tipo de herramientas y la sencillez de su uso. Además si estudiamos un poco su paleta de Steps, podemos percibir la potencia, ya que tenemos opciones para mandar correos, conectarnos a Big Data, conexión por FTP o SSH, ...

A modo de resumen podemos pintar el siguiente UML:



NOTA: Imagen generada con yUML

Y recordamos las utilidades que hemos visto y que son parte de Kettle:

- `spoon` - Aplicación gráfica para diseñar nuestros procesos de extracción, transformación y carga.
- `pan` - Aplicación que nos permite lanzar Transformations en la línea de comandos.
- `kitchen` - Aplicación que nos permite lanzar Jobs en la línea de comandos.

También os dejo los recursos que he utilizado:

- kettle-hello-world.kjb - El Job que gestiona el flujo.

- input-file-from-argument.ktr - La Transformation que lee el fichero como primer argumento de la línea de comandos.
- kettle-hello-world.ktr - La Transformation que convierte el CSV en un XML.
- names.csv - El CSV que se usa como fichero de entrada.
- names-con-saludo.xml - El XML que se obtiene como salida.

7. Sobre el autor

Alejandro Pérez García, Ingeniero en Informática (especialidad de Ingeniería del Software) y Certified ScrumMaster

Socio fundador de Autentia (Desarrollo de software, Consultoría, Formación)

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L. - "Soporte a Desarrollo"

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**