Expresiones regulares y creación de filtros en Trados Studio

CURSO EN LÍNEA PARA OKODIA, NOVIEMBRE DE 2022

José Manuel Manteca Merino
TRADUCTOR AUTÓNOMO EN > ES Y FORMADOR EN TRADOS STUDIO
jmmanteca@jmmanteca.com

José Manuel Manteca Merino
jmmanteca@mmanteca.com

Notas previas

El objetivo de este curso es introducir el tema de las expresiones regulares y su uso para la creación de filtros de archivos en Trados Studio. Dado que no se trata de un curso exhaustivo sobre expresiones regulares, solo se verán las expresiones más habituales y que puedan ser más útiles para este propósito.

A la hora de explicar cómo se crean los filtros en Trados Studio, se dan por supuestos conocimientos básicos de cada programa, tales como crear proyectos de traducción e importar archivos de origen a dichos proyectos.

Leyenda de colores

Con el fin de diferenciar mejor los caracteres y las expresiones regulares mencionadas en este documento, así como los resultados coincidentes y descartados, a lo largo de este manual aparecen secciones de texto con diferentes formatos, los cuales se explican en la siguiente tabla:

Texto en rojo y negrita	Representa uno o más caracteres empleados en una expresión regular, pero no toda la expresión regular en sí misma.
Texto en verde oscuro y negrita	Representa una expresión regular concreta.
Texto en azul y en cursiva	Representa un texto de ejemplo usado para probar el correcto funcionamiento de una búsqueda.
Texto azul en negrita resaltado en amarillo	Muestra los resultados que coinciden con la expresión regular concreta empleada en cada caso.
~~**Texto azul en negrita y tachado**~~	Muestra los resultados que NO coinciden con la expresión regular concreta empleada en cada caso.

José Manuel Manteca Merino
jmmanteca@mmanteca.com

Índice de contenidos

José Manuel Manteca Merino
jmmanteca@mmanteca.com

A. Introducción a las expresiones regulares

1. ¿Qué son las expresiones regulares?

Las expresiones regulares (o *regex* en su forma abreviada) son patrones de búsqueda conformados a partir de una secuencia de caracteres, la cual puede ser tan simple como un solo carácter que queramos buscar en un texto. A partir de ahí, podemos complicar la búsqueda con la ayuda de los metacaracteres, cuantificadores, marcadores y otros elementos que veremos con mayor detenimiento en los siguientes apartados.

Como regla general, cuanto más queramos delimitar la búsqueda (es decir, cuantos menos resultados nos interesen), más compleja tendrá que ser. En cambio, cuanto más sencilla sea la regla, más resultados obtendremos (es decir, será menos precisa).

2. Conceptos básicos

Salvo los elementos mencionados antes, el carácter que empleamos en la búsqueda sirve para buscar ese mismo carácter. Lo mismo ocurre con caracteres que no son texto propiamente dichos, como los espacios, tabulaciones o los saltos de línea. Asimismo, en algunas versiones de regex se distinguen las mayúsculas de las minúsculas, por lo que no es lo mismo buscar **Pero** que **pero**. Sin embargo, no es el caso de Notepad++, Xbench, Studio y, en general, de las aplicaciones que incluyen una casilla para diferenciar las mayúsculas de las minúsculas, donde no se distinguen a menos que se marque esta casilla.

Las expresiones regulares no solo pueden usarse en búsquedas, sino que también sirven para reemplazar los resultados por la secuencia que prefiramos. Por ejemplo, podríamos buscar cantidades económicas en inglés (con el símbolo delante) y sustituirlas por otro patrón que coincida con el de la lengua de destino.

Por último, cabe destacar que hay diversas variantes (*flavours* en inglés) con diferencias más o menos sutiles entre una y otra, por lo que algunas expresiones que funcionan en un programa quizás no lo hagan en otro, y viceversa. Por ejemplo, es muy probable que tengamos que modificar una regla creada en Trados si la queremos utilizar en Xbench, pues las variantes de regex de ambos son distintas. En el caso de la herramienta TAO que veremos en estas sesiones, Trados Studio, usa la variante de Microsoft .NET.

3. Notas importantes

Conviene tener en cuenta los siguientes recordatorios cuando trabajamos con las expresiones regulares:

-No todos los problemas pueden solucionarse con *regex,* o el tiempo que hay que invertir para encontrar la regla perfecta no compensa en comparación con hacer la tarea a mano.

-Hay ocasiones en que no existe una única regla, sino que hay varias mediante las cuales se logran los mismos resultados.

-Es mejor aprender primero las reglas más sencillas y después atrevernos a complicarlas poco a poco.

-Es preferible aplicar varias reglas sencillas en vez de obsesionarse buscando una regla única y demasiado compleja que abarque todos los casos.

4. Delimitadores habituales de las expresiones regulares

4.1 Metacaracteres

Cuando hablamos de expresiones regulares, los metacaracteres son aquellos caracteres que no representan el propio carácter, sino que se emplean con una función especial. Para que el programa de turno sepa si estamos buscando el carácter real o el metacarácter, en el primer caso se pone una barra invertida (****) delante. Por ejemplo, **\$** busca el símbolo del dólar.

Cabe destacar que, como se verá en los apartados posteriores, la barra invertida también se usa con la función contraria (es decir, para buscar un carácter con un significado distinto al literal que representa) en algunos elementos.

La siguiente tabla muestra una lista de los metacaracteres más habituales:

Carácter	Significado
^	Inicio de una línea.
$	Final de una línea.
.	Cualquier carácter, salvo saltos de línea, salvo saltos de línea.
()	Grupo de captura.
[]	Un conjunto de caracteres.
{}	Indica la cantidad de veces que debe aparecer el elemento al que sigue.
?	El elemento al que sigue debe aparecer ninguna o una vez.
*****	El elemento al que sigue debe aparecer cero o más veces.
+	El elemento al que sigue debe aparecer una o más veces.
****	Usado para «escapar» un carácter.

Algunos de estos metacaracteres, como los paréntesis, se explican con mayor detenimiento en diferentes secciones.

Como comentábamos antes, si ponemos la barra invertida delante de estos caracteres, los «escapamos» y pasamos a buscar el carácter real. Veamos una tabla con algunos ejemplos:

Carácter	Significado
\^	Acento circunflejo.
\$	Símbolo del dólar.
\.	Punto.
\(	Paréntesis de apertura.
\]	Corchete de cierre.
\{	Llave de apertura.
\?	Interrogación de cierre.
*****	Asterisco.
\+	Símbolo más.
****	Barra invertida.

Sin embargo, cabe aclarar que la barra invertida también puede ser usada con el propósito contrario, es decir, para convertir en caracteres especiales aquellos que no lo son. En la siguiente tabla se muestran algunos ejemplos.

Carácter	Significado
\d	Cualquier número del 0 al 9 (NO la letra de minúscula).
\t	Una tabulación (NO la letra te minúscula).

4.2 Cuantificadores

En las expresiones regulares, no solo podemos crear una regla para lo que queremos buscar, sino también indicar cuántas veces tiene que aparecer dicho elemento para que se cumpla dicha regla. Para tal fin, después de ese elemento usamos los cuantificadores, que se explican en la siguiente tabla:

Carácter	Significado	Ejemplo	Explicación
?	Ninguna o una vez.	**\.?**	Un punto o ningún punto.
*****	Cero o más veces.	**ñ***	Cero o más eñes.
+	Una o más veces.	**¡+**	Una o varias exclamaciones de apertura.
{x}	Exactamente x veces.	**\d{3}**	Exactamente tres números
{x,}	X o más veces.	**\s{2,}**	Dos o más espacios
{x,y}	Entre x e y veces.	**\d{2,4}**	Entre dos y cuatro números

Llegados a este punto, conviene destacar que los cuantificadores mencionados en la primera columna son «avariciosos» (*greedy,* en inglés), es decir, intentan encontrar el resultado más amplio, lo que puede causar que nuestra búsqueda (por ejemplo, **[a-z]+**, una o más apariciones de cualquier letra minúscula) tenga resultados infinitos o poco precisos.

Para evitar que suceda, añadimos el sufijo **?** a nuestra regla, lo que provoca que se obtenga el menor número de coincidencias posibles (la búsqueda se para en el primer resultado). Esta regla ahora es «vaga» (*lazy* en inglés). Pongamos esta frase como ejemplo:

Cuando le pregunté por su nuevo móvil, Pedro me dijo que era: «barato», «sorprendentemente rápido», «bonito» y «ligero».

Si usamos la regla **«.*»** (cero o más caracteres cualquiera salvo saltos de línea entre comillas latinas), esta es la primera coincidencia que se encuentra:

*Cuando le pregunté por su nuevo móvil, Pedro me dijo que era: **«barato», «sorprendentemente rápido», «bonito» y «ligero»**.*

Vemos que los espacios, comas y otras palabras están incluidos en la coincidencia, ya que la búsqueda se para cuando llega a la última coincidencia. En cambio, si añadimos el carácter **?** y usamos la regla **«.*?»** (cero o más caracteres cualquiera entre comillas, con el menor número de concordancias posibles), estos son los resultados que se encuentran:

*Cuando le pregunté por su nuevo móvil, Pedro me dijo que era: **«barato»**, «sorprendentemente rápido», «bonito» y «ligero».*

Aquí la búsqueda se detiene en el primer caso y después se reanuda hasta encontrar todos los resultados. Dicho de otra forma, en el primer caso obtendríamos una sola coincidencia, mientras que en el segundo caso serían cuatro las coincidencias. Así conseguimos mejorar la precisión de las expresiones regulares.

4.3 Anclas

Las anclas son elementos que marcan el lugar donde deben estar las coincidencias que se quieran encontrar. Además, son caracteres de ancho cero (un carácter invisible que no se puede copiar ni pegar).

Carácter	Significado	Ejemplo	Resultado
^	Inicio de una regla.	**^¿**	Interrogación de apertura situada al inicio de una línea.
$	Final de una regla.	**:$**	Dos puntos situados al final de una línea.
\b	Límite de palabra.	**más\b**	Quiero comprar la ~~más~~cara **más** cara que haya en la tienda de ~~más~~caras.
\B	No es un límite de palabra.	**más\B**	Quiero comprar la **más**cara ~~más~~ cara que haya en la tienda de **más**caras.

4.4 Grupos y rangos

Los grupos son uno o más elementos agrupados entre paréntesis y que funcionan como una única entidad. Dicho de otra forma, el resultado debe coincidir exactamente con lo que se haya expresado entre el paréntesis de apertura y el de cierre y en el mismo orden de izquierda a derecha.

Carácter	Significado	Ejemplo	Resultado
(XY)	X e Y (en su conjunto).	**(Traduc)ción**	Yo estudio **Traducción** e Interpretación.
(X|Y)	X o Y.	**(Traduc|Interpreta)ción**	Yo estudio **Traducción** e **Interpretación**.

En cambio, los rangos son uno o más elementos agrupados entre corchetes, pero se busca solo uno o varios de esos elementos y no todos, y sin ningún orden en concreto.

Carácter	Significado	Ejemplo	Resultado	
[XYZ]	Cualquier carácter que sea X, Y o Z.	**Mé[jx]ico**	**Méjico** **México**	~~Médico~~
[^XYZ]	Cualquier carácter que no sea X, Y ni Z.	**Mé[^j]ico**	**México** **Médico**	~~Méjico~~
[a-z][1]	Cualquier letra minúscula.			
[a-d]	Cualquier letra minúscula entre la a y la de.			
[A-Z]	Cualquier letra mayúscula.			
[A-J]	Cualquier letra mayúscula entre la a y la jota.			
[0-9] | **\d**[2]	Cualquier número del 0 al 9.	**\d\s€**	**Tengo 9 € en el bolsillo.**	
[1-3]	Cualquier número del 1 al 3.			

Aquí podemos observar que el acento circunflejo tiene otro significado del que hemos visto antes. En este caso, sirve para anular o negar lo que viene después.

[1] En muchas variantes, no incluye las letras con virgulilla (letras con acentos gráficos, la eñe, etc.). Entonces se usa la expresión **\p{L}** (o las variantes **\p{Lu}** y **\p{Ll}** para las mayúsculas y minúsculas respectivamente).
[2] En algunas variantes, incluye las cifras de otros alfabetos y otras expresiones numéricas, como las fracciones.

4.5 Espacios y saltos de línea

Aparte de los caracteres «visibles», tales como las letras, los números y los símbolos, en las expresiones regulares también es posible buscar y reemplazar caracteres invisibles, como espacios, tabulaciones, saltos de línea, etc. En la siguiente tabla vemos una relación de los más habituales.

Carácter	Significado	Ejemplo	Explicación
\s	Un espacio.	**\d\s**	Un número seguido de un espacio.
\S	Cualquier carácter que no sea un espacio.	**\d\S**	Un número que no esté seguido de un espacio (por ejemplo, números en códigos alfanuméricos).
\u00A0	Un espacio duro.	**\u00A0\€**	Un espacio duro seguido del símbolo del euro.
\t	Una tabulación.	**\.\t**	Un punto seguido de una tabulación.
\n	Un salto de línea.	**:\n**	Dos puntos seguidos de un salto de línea.
\r	Un retorno de carro.		

Como se puede deducir, estos elementos llevan la barra invertida delante para diferenciarlos del propio carácter literal. Así mismo, se pueden buscar otros espacios o elementos diferentes del espacio duro si sabemos el código Unicode de ese carácter. En ese caso, el patrón es **\u0000**, en el que hay que sustituir los ceros por dicho código. Por ejemplo, el espacio fino duro (o irrompible) se busca con el patrón **\u202F**.

4.6 Sustitución de cadenas

Las expresiones regulares no se limitan a las búsquedas, sino que también se pueden emplear en sustituciones. Así pues, a diferencia de los caracteres comentados en los apartados anteriores, los que figuran en la siguiente lista se utilizan para indicar por qué elemento queremos sustituir la expresión que usemos en la búsqueda

Para separar los elementos en la búsqueda, los agruparemos entre paréntesis y los contaremos empezando por la izquierda. Según el programa, se puede usar el símbolo del dólar o la barra invertida seguidos del número correspondiente para señalizarlos.

Carácter		Significado
$X	**\X**	El grupo situado en la posición X de la cadena.
$1	**\1**	El primer grupo de una cadena.
$0	**\0**	Toda la cadena.

Por ejemplo, supongamos que queremos sustituir la coma de los millares por un espacio. En el caso de la búsqueda, emplearemos una expresión similar a la siguiente:

(\d)(\.)(\d+) Un número seguido de un punto y de uno o más números.

Hemos dividido la expresión en tres grupos, cada uno de los cuales aparece entre paréntesis. Nuestro objetivo es dejar el primer y el tercer grupo sin cambiar y sustituir el punto por un espacio, para lo cual usaremos la siguiente expresión en el cuadro **Reemplazar.**

\1 \3

Como se puede observar, el orden del primer y del tercer elemento no ha cambiado, mientras que el segundo ha desaparecido y ha sido sustituido por un espacio.

En la siguiente tabla tenemos más ejemplos de estas búsquedas y sustituciones:

Frase original	**Búsqueda**	**Sustitución**	**Resultado**
Este año he pagado 480 258,3 € de IVA.	**(\s\d{3}\s\d{3},\d\s€)(\sde\sIVA)**	**\2\1**	Este año he pagado de IVA 480 258,3 €.
Este año he pagado 480 258,3 € de IVA.	**(\s\d{3}\s\d{3},\d\s€\sde\sIVA)**	**un total de $0**	Este año he pagado un total de 480 258,3 € de IVA.

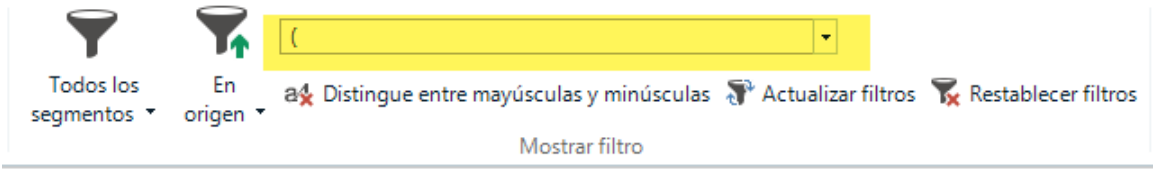
5. Ejemplos prácticos

5.1 Filtros de segmentos

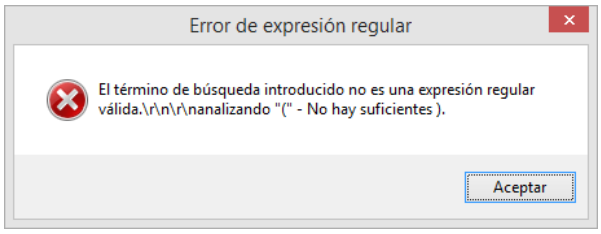
La herramienta TAO que utilizamos para estas sesiones, Trados Studio, permite filtrar los segmentos de un archivo o conjunto de archivos que estemos traduciendo para que se muestren solo los que cumplan unos criterios concretos. Las expresiones regulares son de gran ayuda en esta situación, ya que hacen que los filtros sean todavía más precisos.

Filtros de segmentos en Studio:

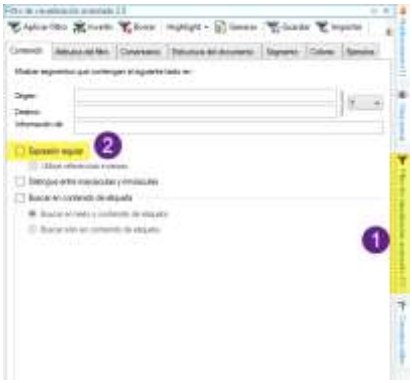
Los filtros de segmentos de Studio se encuentran en la pestaña **Revisión** de la cinta de opciones. En la sección **Mostrar filtro** hay un cuadro de búsqueda que podemos emplear para escribir las expresiones regulares.



Cabe destacar que la búsqueda con expresiones regulares está activada por defecto, lo que explica que aparezcan errores crípticos cuando buscamos ciertos caracteres sin escapar (es decir, sin poner la barra invertida delante), como ocurre con el paréntesis de apertura y el de cierre, la interrogación de cierre, el asterisco y tantos otros.



Studio también cuenta con unos filtros avanzados denominados **filtros de visualización avanzado 2.0,** a los cuales accedemos desde la barra lateral derecha. En este caso, las expresiones regulares están desactivadas, así que tendremos que marcar la casilla correspondiente, tal como se aprecia en la captura.



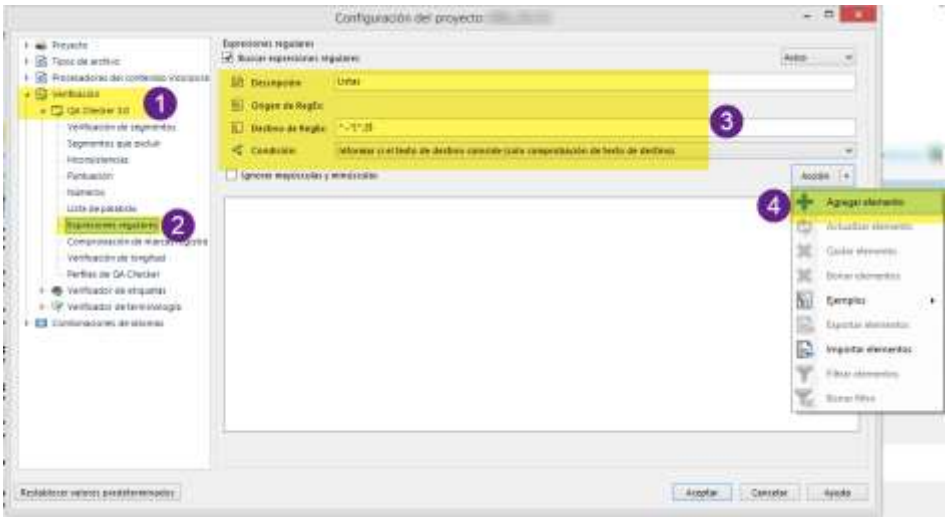
José Manuel Manteca Merino
jmmanteca@mmanteca.com

5.2 Controles de calidad personalizados

Otro uso muy interesante de las expresiones regulares en Trados Studio es la creación de reglas para el control de calidad, pues los ajustes que trae el programa se quedan cortos en muchísimos casos.

En Trados Studio no solo podemos crear nuestras propias reglas, sino que es posible guardarlas para recuperarlas en otros proyectos del mismo cliente, tema, combinación de idiomas, etc. Además, estas reglas permiten llegar adonde el programa no alcanza o se queda corto. También serán reglas más fiables y precisas, por lo que el número de falsos positivos será menor.

Podemos configurar estas reglas en **Configuración del proyecto > QA Checker 3.0 > Expresiones regulares.**



No solo podemos buscar en el segmento de origen o destino o en ambos a la vez, sino que podemos fijar las situaciones que deben darse para que Studio nos avise de un posible error:

Informar si los patrones RegEx de destino y origen coinciden
Informar si el texto de destino coincide pero no el de origen
Informar si el texto de origen coincide pero no el de destino
Informar si el texto de origen coincide (solo comprobación de texto de origen)
Informar si el texto de destino coincide (solo comprobación de texto de destino)
Informar si el segmento de origen y destino coinciden pero con un recuento diferente
Expresión de búsqueda agrupada: informar si el texto de origen coincide pero no el de destino
Expresión de búsqueda agrupada: informar si el texto de origen y de destino coinciden

Si queréis ampliar conocimientos sobre este tema, podéis mirar los vídeos **gratuitos** de las formaciones que he impartido al respecto.

https://melodiadetraduccion.com/webinarios-sobre-regex-durante-la-traduentena/

5.3 Archivo de ejemplo

Si queremos practicar, podemos usar el siguiente texto de ejemplo para llevar a cabo unas comprobaciones lingüísticas y ortográficas con la ayuda de las expresiones regulares. Es el mismo texto que el que aparece en el archivo **ejemplo_regex.docx:**

Este es un archivo de prueba para comprobar el correcto funcionamiento de las expresiones regulares. Sorprendentemente, estas expresiones no sirven solamente para hacer búsquedas, sino también para sustituciones.

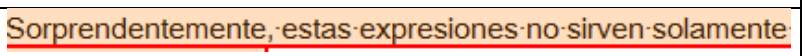
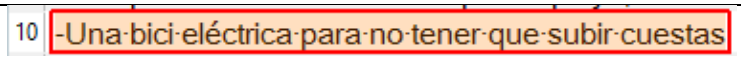
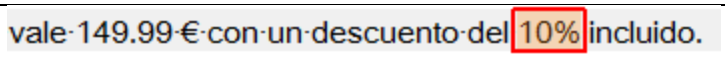
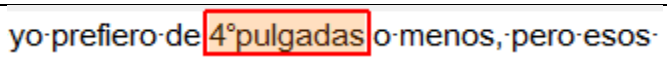
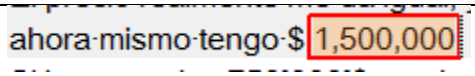
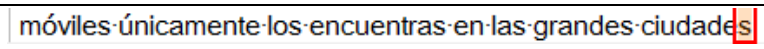
Hoy he estado probando un nuevo teléfono móvil que vale 149.99 € con un descuento del 10% incluido. En otra tienda, valía 169,99 euros con IVA incluido. Tiene una pantalla de 5,5" y 2.048 Mb (2 Gb) de memoria. En mi caso, esa pantalla me resulta tremendamente grande. Yo prefiero de 4 pulgadas o menos, pero esos móviles únicamente los encuentras en las grandes ciudades

El precio realmente me da igual, ya que puedo permitirme comprar cualquiera, pues en la cuenta corriente ahora mismo tengo $ 1,500,000. Si le sumas los 750 000 $ que he ganado en la lotería pero que no me han ingresado, salen 2.250.000 $ que puedo gastar en comprarme casi cualquier cosa, como las siguientes:

-Un apartamento en Laredo a pie de playa;
-Una bici eléctrica para no tener que subir cuestas
-Un ordenador que ríete tú de los de la NASA;
-Una mochila donde quepa todo lo que llevo cuando voy de tradugrino;
-Billetes de avión para cualquier destino
-Un robot aspirador que limpie mis mansiones;
-Y todas las cosas que me dejo en el tintero.

Sinceramente, no creo que sea descabellado pensar constantemente en este tipo de cosas. Ahora voy a dejar de soñar y a apagar el despertador, que ya son las 07:30. Hoy va a ser un día intenso: me toca trabajar hasta las 18.00.

En la siguiente tabla vemos lo que se pretende conseguir con cada uno de los ejemplos, la expresión regular empleada para hacerlo, una explicación al respecto y una captura de pantalla de los resultados que encuentra Studio. Dada la limitación de espacio y el carácter no exhaustivo del manual, en las capturas de pantalla no se muestran todos los resultados.

Objetivo	**Regex**	**Explicación del regex**	**Resultado**
Dos o más adverbios acabados en -mente en el mismo segmento.	**(.+mente){2,}**	Dos o más apariciones de uno o más caracteres seguidos de «mente».	
Elementos de listas que no acaben en punto y coma.	**^-.*[^;]$**	Guion al inicio de línea, varios caracteres cualesquiera, SIN punto y coma y final de línea	
Porcentajes que no estén separados de la cifra.	**\d+%**	Uno o más números seguidos de un símbolo de porcentaje.	
Expresión literal de un símbolo en vez del propio símbolo.	**\d+\spulgadas**	Uno o más números seguidos de *pulgadas*.	
Millares y millones separados por coma o punto.	**\d+([.,]\d{3})+**	Uno o más números seguidos de una o más series de tres números precedidas por punto o coma.	
Final de frase que no acaba en punto y seguido.	**[^\.]$**	Un final de línea NO precedido por un punto.	

Si queréis consultar más sobre este tema, podéis ver las sendas formaciones gratuitas gratuita que di para Trágora sobre el uso de las expresiones regulares en el control de calidad de Trados Studio. Tenéis el enlace en:

https://melodiadetraduccion.com/webinarios-sobre-regex-durante-la-traduentena/

5.4 Expresiones regulares en la preparación de archivos

Antes de crear un filtro para archivos de texto complejos en Trados Studio, a veces es necesario pasar primero por un editor de texto para hacerles unos ajustes con la ayuda de las expresiones regulares. Lo veremos mejor con un ejemplo:

```
<source>Both sides have changed since last synchronization.</source>CRLF
<target></target>CRLF
CRLF
<source>Cannot determine sync-direction:</source>CRLF
<target></target>CRLF
CRLF
<source>No change since last synchronization.</source>CRLF
<target></target>CRLF
CRLF
<source>The database entry is not in sync considering current settings.</source>CRLF
<target></target>CRLF
CRLF
<source>Setting default synchronization directions: Old files will be overwritten with newer
files.</source>CRLF
<target></target>CRLF
```

En este archivo, el texto de origen se encuentra entre una etiqueta *source* de apertura y otra de cierre, mientras que el texto de destino está delimitado por el par de etiquetas *target*. En Trados Studio tendremos que indicar que el primer caso es texto no traducible y que no se debe importar, a la vez que hacemos lo contrario en el segundo caso. Sin embargo, como ahora mismo las etiquetas *target* están vacías, no se mostraría nada en la TAO.

Es entonces cuando necesitamos usar la siguiente expresión regular:

(<source>)(.*?)(<\/source>)(\r\n)[3](<target>)(.*?)(<\/target>)

Ahora desglosaremos la regla:

(<source>)(.*?)(<\/source>)

Cualquier carácter situado entre las etiquetas *source*. Lo separamos en tres grupos porque queremos copiar el segundo grupo y pegarlo entre las etiquetas *target*.

(\r\n)(<target>)(.*?)(<\/target>)

Un retorno de carro y un salto de línea seguidos de cualquier carácter situado entre las etiquetas *target*. Usamos de nuevo los grupos para que sea más fácil copiar y pegar el contenido de una parte a otra.

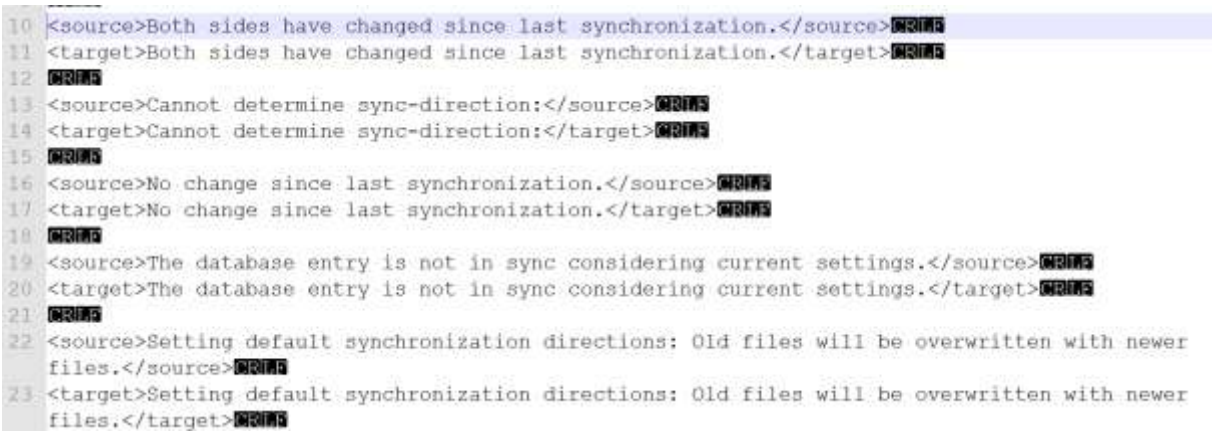
En el cuadro **Reemplazar,** emplearemos la siguiente expresión:

\1\2\3\4\5\2\7

[3] Los saltos de línea y párrafo se indican así en Notepad++, que es el editor de texto usado en este caso, por lo que puede variar si se emplean otros editores de texto.

José Manuel Manteca Merino
jmmanteca@mmanteca.com

Nos hemos limitado a copiar el segundo grupo y pegarlo en el sexto lugar, mientras que los demás elementos permanecen en sus posiciones originales. Vemos el resultado:



Esta es una muestra de la potencia de las expresiones regulares. Con un simple buscar y reemplazar, nos hemos ahorrado un tiempo valioso que habríamos tenido que pasar haciendo lo mismo, pero de forma manual y de uno en uno. Si lo extrapolamos a varios o incluso a centenares de archivos, se hace evidente que saber de expresiones regulares (sin que sea necesario dominar la materia) puede ayudarnos en el trabajo del día a día.

5.5 Tareas de posproducción en archivos ya traducidos

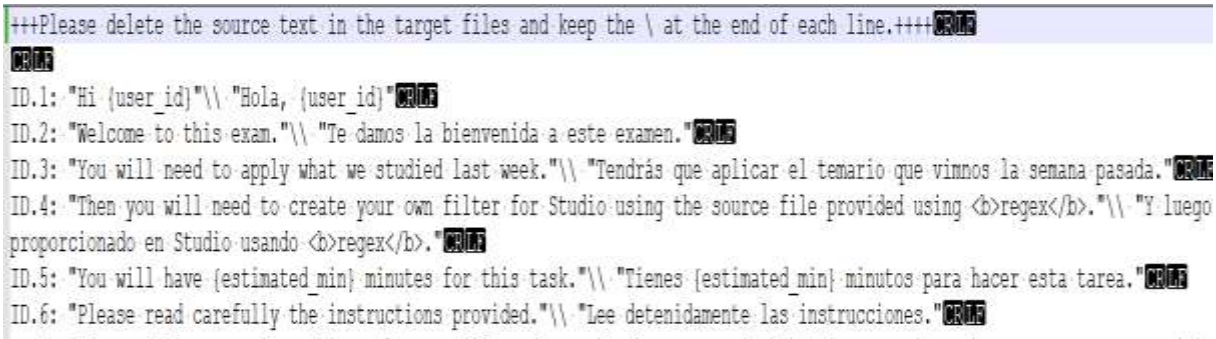
Las tareas de posproducción son procesos que se aplican a los archivos de destino generados por la TAO con el fin de que se ajusten a las exigencias del cliente.

Algunas situaciones en que debemos recurrir a las expresiones regulares en durante estas tareas de posproducción son las siguientes:

-Convertir entidades HTML a su carácter real o viceversa para garantizar que el archivo se importa correctamente de vuelta en el sistema de cliente.

-Anular acciones que hayamos aplicado durante la preparación de archivos para facilitar su importación en la Trados Studio.

Por poner un ejemplo, en el siguiente archivo se nos pide borrar el texto de origen y dejar solo la traducción manteniendo las barras invertidas:



Buscamos: **(^ID\.\d+:.*?)(".*?")(\\+\s)(".*?")$**

Y sustituimos por: **$1$4$3**

José Manuel Manteca Merino
jmmanteca@mmanteca.com

B. Creación de filtros

1. Identificación de los patrones de apertura y cierre

Cuando recibimos unos archivos con una extensión desconocida para nuestra TAO, lo primero que conviene hacer es intentar abrirlo con un editor de textos como Notepad++ para discernir si se trata de un archivo de texto o de otro tipo. Si nos encontramos ante un archivo de texto, podemos crear un filtro para traducirlo en Studio.

Con frecuencia, los archivos de texto pertenecen a uno de estos dos tipos:

a) Familia de los XML

Muchos de estos archivos de texto están basados por el código XML, que usa una estructura jerárquica y una clasificación por etiquetas para delimitar el contenido. Es el caso de archivos que en nuestra profesión vemos con mucha frecuencia, tales como los XLIFF, TMX, TBX y tantos otros.

Ya en la primera fila podemos averiguar si se trata de un archivo de este tipo, pues todos estos archivos tienen una declaración o presentación donde se hace constar que usan un código XML, tal y como se ilustra en la captura de pantalla.

Aunque lo veremos más adelante, cuando creemos un filtro para archivos de esa índole en Studio, lo tendremos algo más fácil que con otros archivos, pues usaremos el filtro de XML por defecto de esta TAO y solo tendremos que decirle al programa cuáles etiquetas se traducen y cuáles no. Por ejemplo, en la siguiente captura se observa que en este archivo las etiquetas *target* son las que más nos importan, pues es donde se encuentra todo el contenido traducible.

b) Texto delimitado por expresiones multilingües

En programación también se suelen ver archivos de texto delimitado por alguna regla, más que por un par de etiquetas, como ocurría en el caso anterior. Al principio del archivo suelen aparecer algunas propiedades que no cambiaremos. Debajo figura un código o un identificador de cadena de texto seguido por el texto que hay que traducir.

En la captura podemos ver que el contenido aparece entre comillas, separado de su correspondiente identificador mediante un código numérico (que comienza con el signo menos en algunos casos) y el signo igual. Este patrón no es siempre el mismo, e incluso puede tener excepciones dentro de un mismo archivo.

Por tanto, la clave a la hora de crear un filtro para este tipo de archivos es examinar bien los archivos para identificar las expresiones regulares que permiten indicarle a la herramienta TAO cuál es el contenido traducible y cuál es que no tiene que ser importado en el programa. Asimismo, como explicamos más adelante, Trados Studio cuenta con herramientas que ayudan a comprobar si hemos creado el filtro de forma correcta.

2. Creación de filtros en Studio

2.1. Archivos de texto delimitado

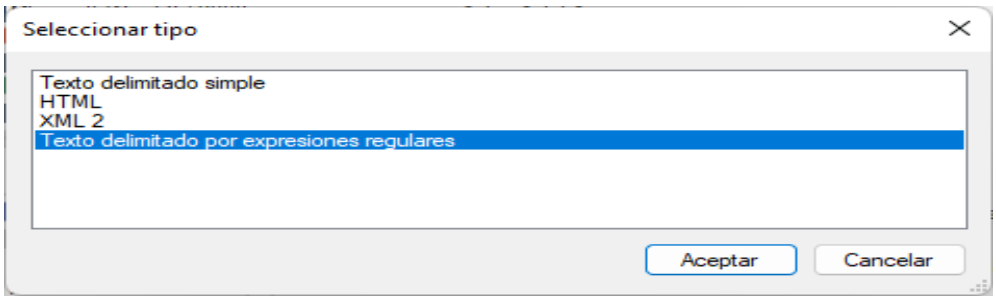
Supongamos que hemos recibido un archivo de ejemplo llamado *randomapp.strings* con este aspecto:

```
"accept_randomapp" = "%@ will attend";

"accept_randomapp_shared" = "%@ answered that %@ will attend";

"accounts_merge_description" = "All future notifications will be sent to the app instead of
to your email client.";

"accounts_merge_description_web" = "You can now merge your accounts. All future notifications
sent to this email address will be found at this website.";
```

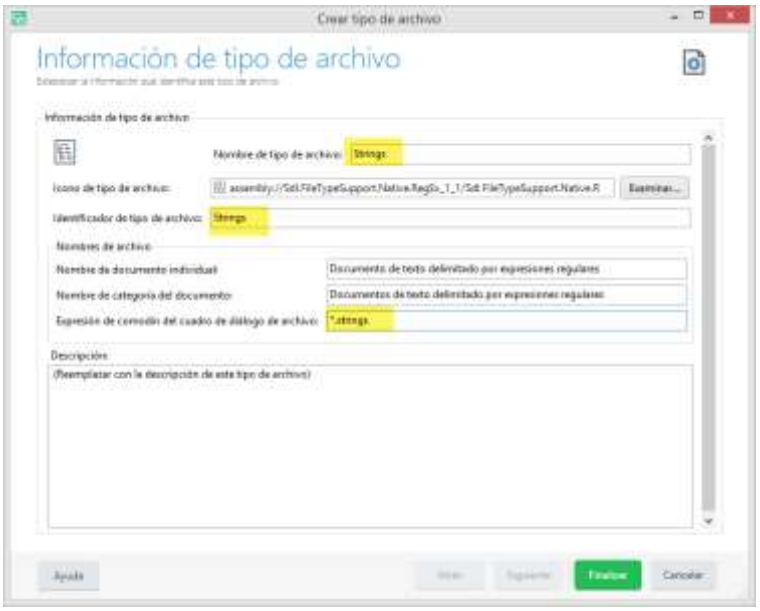
Podemos ver que se trata de un archivo del segundo tipo que hemos visto en el apartado anterior, es decir, un archivo de texto delimitado.

Creación del filtro base

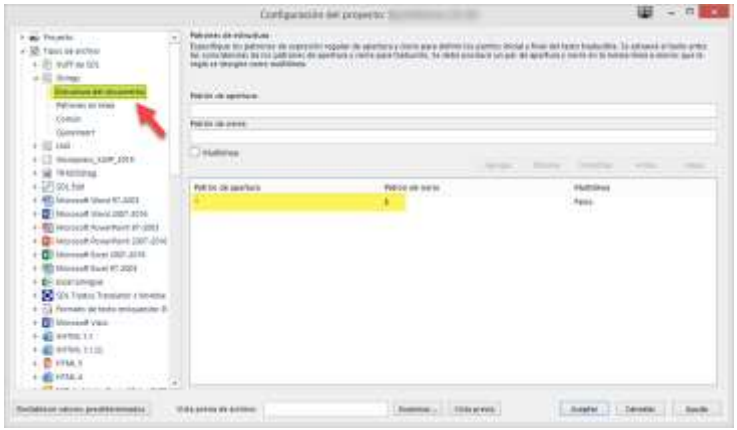
Ahora que lo tenemos claro, vamos a Studio y luego a **Configuración del proyecto > Tipos de archivo > Nuevo,** y ahí elegimos **Texto delimitado por expresiones regulares,** como en esta captura:



En la siguiente ventana cambiaremos la información referente a la extensión del archivo, para que así Studio asigne el filtro correcto de forma automática cuando importemos un archivo de la misma extensión. Después de pulsar en **Finalizar,** Studio habrá creado un filtro con unas reglas por defecto que vamos a modificar en los pasos que siguen.



Ahora iremos a **Estructura del documento** para indicar a Studio los patrones de apertura y cierre, es decir, qué es lo que antecede y sigue al contenido traducible. Studio trae una regla de serie mediante la cual se importa todo el archivo y, dado que no es lo que nos interesa, la seleccionaremos y la modificaremos o borraremos.



Como patrón de apertura, usaremos la siguiente expresión:

^.*?=\s" (inicio de línea seguido de cero o más caracteres, un espacio, el signo igual y comillas).

El patrón de cierre es el siguiente:

";$ (comillas seguidas de punto y coma y del final de línea).

Si pulsamos en **Vista previa,** Studio creará una muestra de cómo quedaría el archivo si lo importásemos en ese momento.

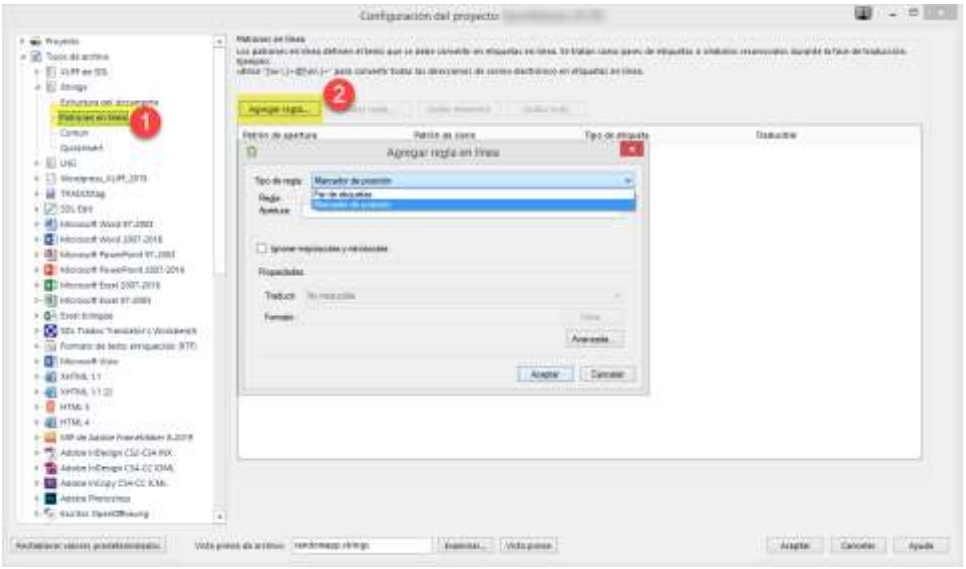
A priori, parece que va todo bien, a la espera de que bloqueemos las variables y el código HTML del archivo (resaltados en la captura).

José Manuel Manteca Merino
jmmanteca@mmanteca.com

Bloqueo de etiquetas HTML y variables

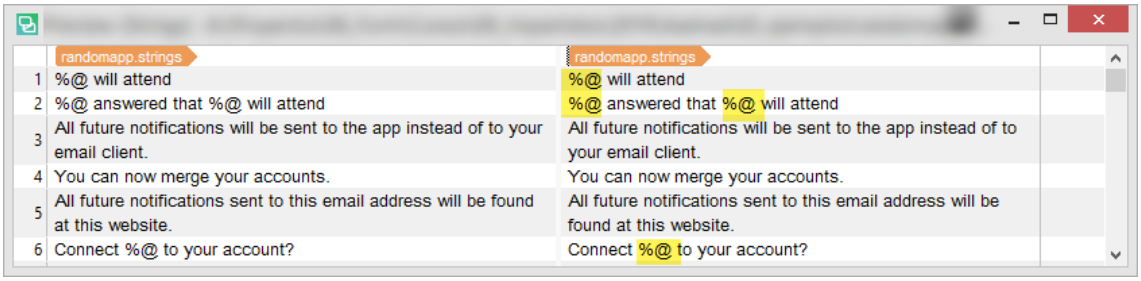
Muchos archivos de texto delimitado contienen código HTML o variables (caracteres comodín cuyo texto cambia según el contexto) que tendremos que convertir en etiquetas por razones de comodidad (porque de lo contrario aparecerán como texto normal) y también para garantizar la integridad del archivo. Si borramos alguna de estas etiquetas, la cambiamos de sitio u añadimos alguna que no esté en el idioma de origen, el control de calidad de Studio nos avisará de esta situación para que podamos enmendar el error.

Estos elementos se añaden usando expresiones regulares desde la sección **Patrones de línea** dentro de la configuración del nuevo filtro. Tendremos pulsar el botón **Agregar regla** y luego elegir entre **Par de etiquetas** (si hay una etiqueta de apertura y otra de cierre, como en muchas etiquetas HTML) o **Marcador de posición** (si se trata de una variable o de un salto de línea, entre otros.).



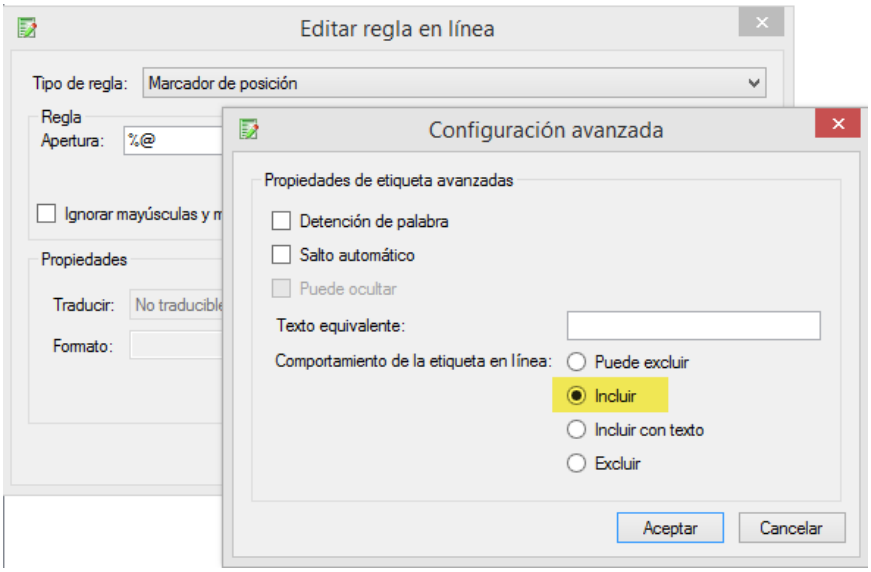
Ahora vamos a ver unos ejemplos de estas reglas:

a) La variable **%@**.

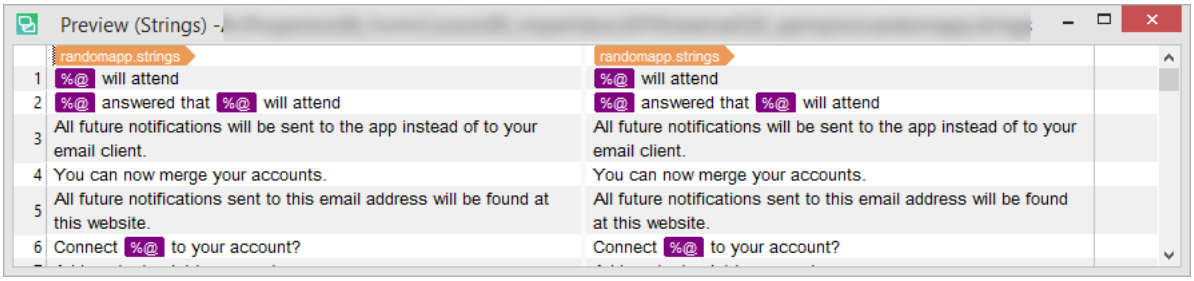


	randomapp.strings	randomapp.strings
1	%@ will attend	%@ will attend
2	%@ answered that %@ will attend	%@ answered that %@ will attend
3	All future notifications will be sent to the app instead of to your email client.	All future notifications will be sent to the app instead of to your email client.
4	You can now merge your accounts.	You can now merge your accounts.
5	All future notifications sent to this email address will be found at this website.	All future notifications sent to this email address will be found at this website.
6	Connect %@ to your account?	Connect %@ to your account?

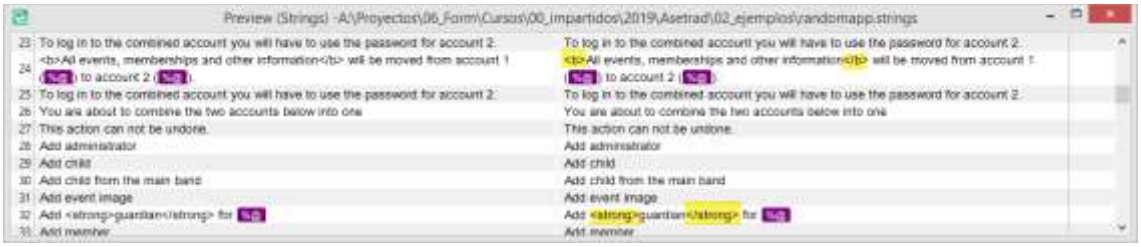
Como ninguno de los dos caracteres es especial en regex, la añadimos tal cual. Sin embargo, como es un elemento imprescindible en el texto por tratarse de una variable de tipo lingüístico, tenemos que indicar a Studio que no la omita si está al principio o final de un segmento. Para ello, pulsamos el botón **Avanzada** y seleccionamos **Incluir** dentro de la sección **Comportamiento de la etiqueta en línea.**



Ahora hacemos clic en **Vista previa** y obtendremos el siguiente resultado:



b) Etiquetas HTML



Como se observa en la captura, hay algunas etiquetas dobles, con su apertura y su correspondiente cierre. Por lo tanto, hay que escribir dos reglas por cada par de etiquetas. Además, por comodidad, añadiremos cada par de etiquetas por separado.

Sin embargo, como muestra de que se pueden agregar usando una sola regla más compleja, vamos a emplear una misma regla para añadir los dos pares señalados en la captura de pantalla anterior.

Para la etiqueta de apertura, la regla es:
<(b|strong)> Usamos la barra vertical para indicar las alternativas, las cuales englobamos mediante los paréntesis.

En el caso de la etiqueta de cierre, la regla es:
<\/(b|strong)> Es casi igual que la regla anterior, salvo que ponemos delante de la barra que indica que es una etiqueta de cierre (/) la barra invertida (\). Esta última sirve para «escapar» la barra (es decir, buscamos el carácter literal)[4].

Y aquí tenemos el resultado:

c) Salto de línea
En algunos segmentos del archivo, observamos que se utiliza **\n** una o dos veces para indicar que hay un salto de línea.

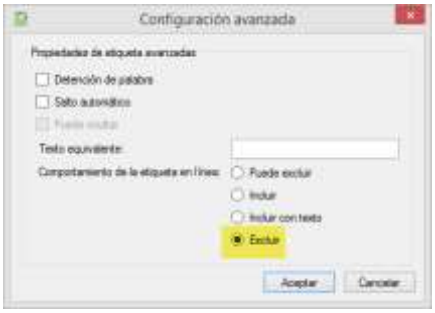
Este es el clásico ejemplo de un marcador de posición, ya que no es un elemento traducible, sino que sirve para separar secciones del texto (es decir, no se trata de una variable). Lo podemos añadir mediante este regex:

\\n+ Como la barra invertida es un carácter especial, le ponemos otra delante para indicar que queremos buscar la propia barra. El signo más sirve para buscar una o más apariciones de este marcador.

Veamos el resultado:

[4] Técnicamente hablando, la barra (/) no es un metacáracter en regex, por lo que no sería necesario «escaparla». Sin embargo, se recomienda hacerlo porque algunos programas podrían confundirla con la barra inversa, lo que invalidaría nuestras reglas.

En este caso, podemos ir más allá para que cada frase aparezca en su propio segmento y no varias en uno solo, lo que resulta farragoso para traducir. Si editamos la regla y pulsamos en **Avanzada** y luego en **Excluir** dentro de la sección **Comportamiento de la etiqueta en línea**, Studio no mostrará estas etiquetas y colocará cada frase en su propio segmento.



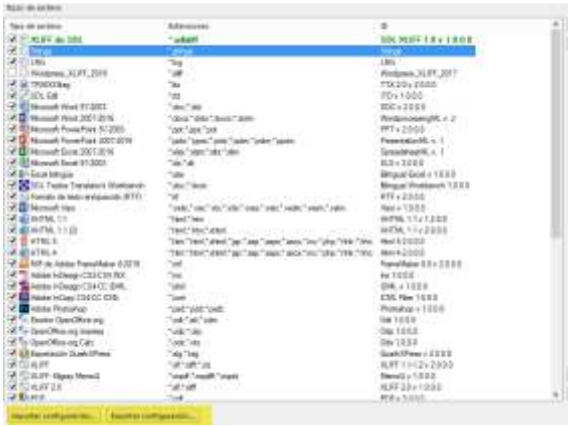
Ahora tenemos un texto mucho más cómodo para traducir:

Así pues, tan solo habría que ir añadiendo todas las reglas que hagan falta y ya tendríamos un filtro que podríamos utilizar con archivos de la misma extensión. En el apartado que sigue vamos a descubrir cómo guardar este filtro para poder reutilizarlo en futuros proyectos.

Cómo guardar el filtro para usarlo en futuros proyectos

Dado que nos llevaría mucho tiempo crear un filtro desde cero siempre que empecemos un proyecto donde traduzcamos archivos de extensiones poco frecuentes pero similares entre ellos, Studio cuenta con la opción de guardar el filtro para que lo podamos usar más adelante, o incluso compartirlo si se trata de un proyecto dividido entre varias personas.

Para tal fin, en la **Configuración del proyecto > Tipos de archivo,** seleccionamos el filtro que queramos guardar y hacemos clic en el botón **Exportar configuración** situado en la esquina inferior izquierda de la ventana.



Entonces solo queda elegir el nombre del archivo y la carpeta donde queremos guardarlo y Studio exportará un archivo de extensión **sdlftsettings.** Si estuviéramos en otro ordenador y quisiéramos usar el filtro, tan solo tendríamos que hacer el paso inverso haciendo clic en el botón **Importar configuración** en esta misma ventana.

La pseudotraducción

Si queremos comprobar que hemos creado el filtro correctamente, podemos usar la función de Studio llamada **pseudotraducción,** mediante mediante la cual se llenan los segmentos de destino de texto falso. Accedemos a dicha función desde **Tareas por lotes > Pseudotraducir.**

Luego se exporta el archivo de destino para comprobar que se crea sin problemas y que no hemos dejado texto traducible sin importar (o viceversa). En la siguiente captura, detectamos que una cadena de texto se había quedado sin importar en Studio:

```
"x_unread_posts.one" = "_%d taxation minus_";

"x_unread_posts.other" = "_%d injector oranges_";

"x_unread_messages.zero" = "%d unread messages"
```

Tan solo tendremos que hacer los cambios en el archivo de origen o en el filtro y repetir el proceso hasta que esté todo en orden y entonces ya podremos traducir este archivo sin quebraderos de cabeza.

2.2. Archivos de la familia de los XML

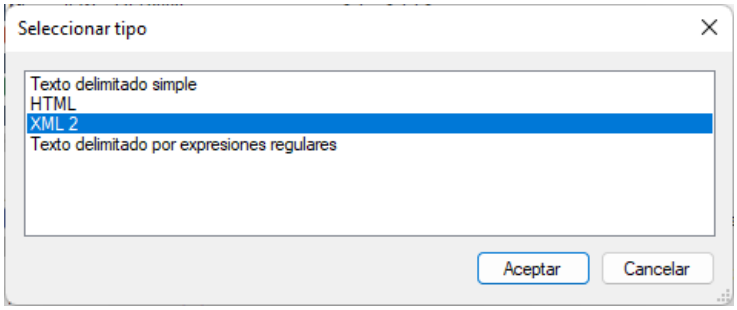
Supongamos que ahora hemos recibido del cliente un archivo de ejemplo llamado *strings.xml* con este aspecto:

```
<?xml version="1.0" encoding="utf-8"?>LF
<resources>LF
    <string name="error_list">Error al cargar datos</string>LF
    <string name="username">Nombre de usuario</string>LF
    <string name="cancel">Cancelar</string>LF
    <string name="accept">Aceptar</string>LF
    <string name="access">Acceder</string>LF
    <string name="loading">Cargando</string>LF
```

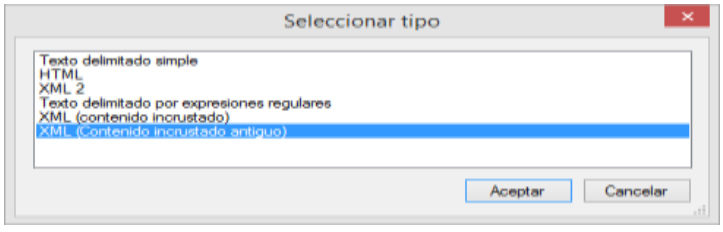
Vemos la declaración XML en la primera línea y que el contenido se encuentra delimitado por una etiqueta de apertura y otra de cierre. Por lo tanto, se trata de un archivo de la familia de los XML.

Creación del filtro base

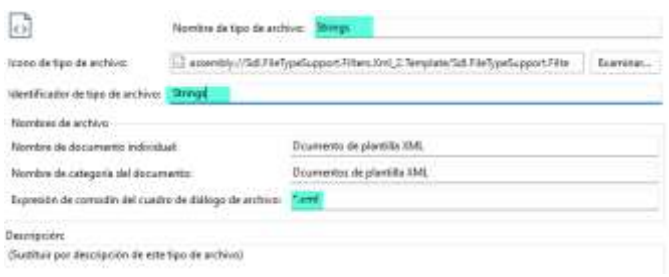
Ahora que lo tenemos claro, vamos a Studio y luego a **Configuración del proyecto > Tipos de archivo > Nuevo,** y ahí elegimos **XML2,** como en esta captura:



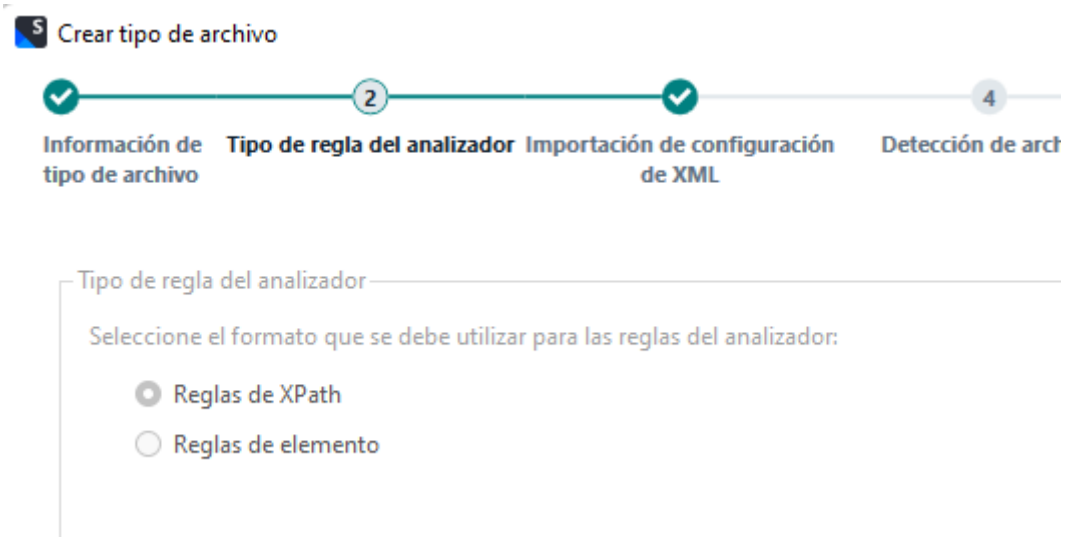
En versiones de Studio anteriores a 2022, también poder elegir entre **XML (contenido incrustado)** o **XML (contenido incrustado antiguo)** si así lo preferimos.



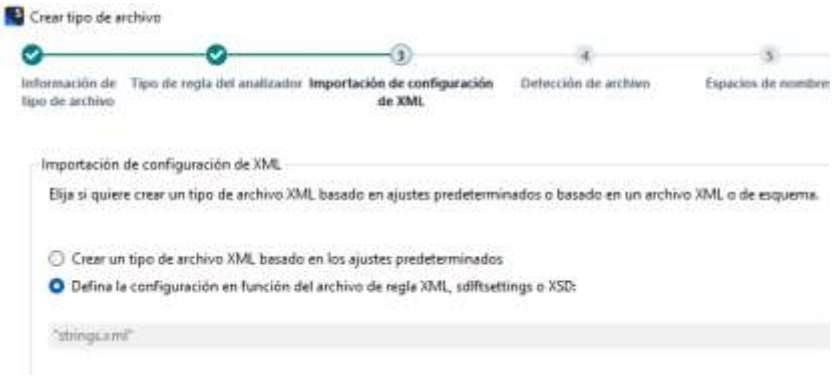
En la siguiente ventana cambiaremos la información referente a la extensión del archivo, para que así Studio asigne el filtro correcto de forma automática cuando importemos un archivo de la misma extensión.



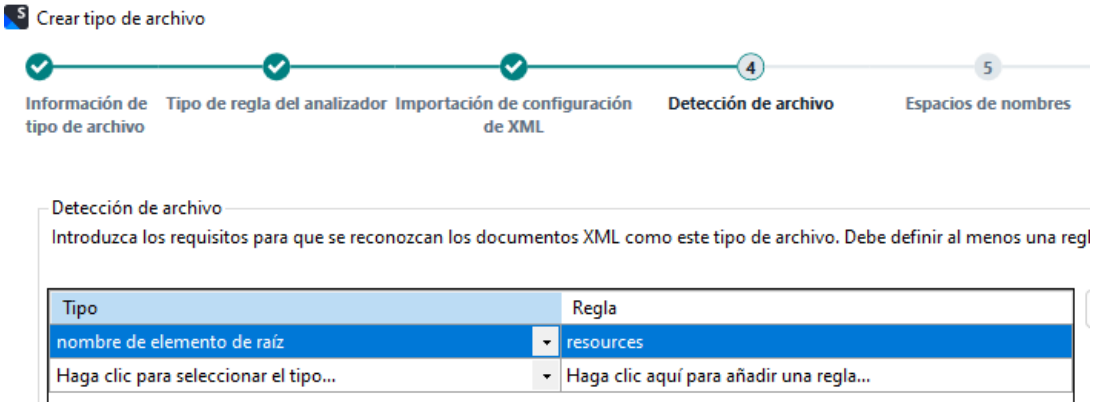
En **Tipo de regla,** podemos utilizar cualquiera de las dos opciones, ya que lograremos resultados parecidos. **XPath** es parecido a las expresiones regulares, pero aplicado a la estructura de los archivos XML.



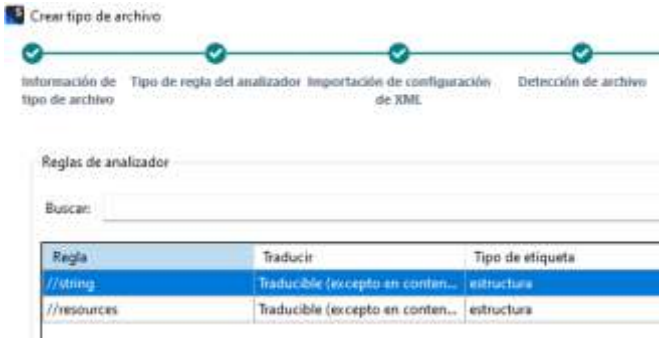
En el paso **Importación de configuración de XML,** buscamos los archivos de origen para que Studio analice su estructura y muestra la lista de etiquetas XML que los conforman. Si no vemos los archivos, tendremos que cambiar su extensión por XML (y volverla a cambiar antes de importarlos).



Después Studio nos pide el elemento raíz (que no hay que cambiarlo en principio). Cada vez que importemos un archivo XML con este elemento raíz, Studio le asignará este filtro de forma automática.

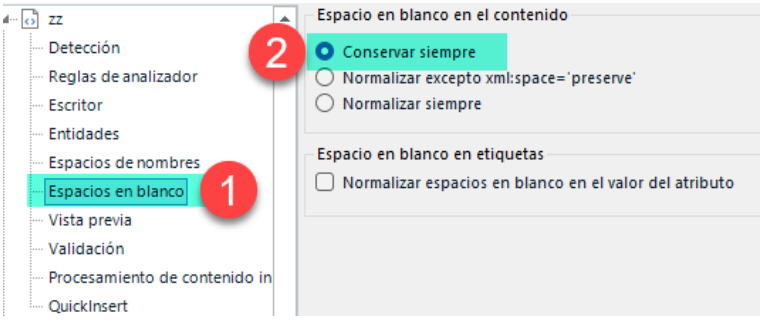


En **Reglas de analizador,** veremos las etiquetas XML que conforman el archivo de muestra en cuestión. En su caso, hay que marcar como **no traducibles** todas aquellas etiquetas que no queramos importar.



Espacios en blanco

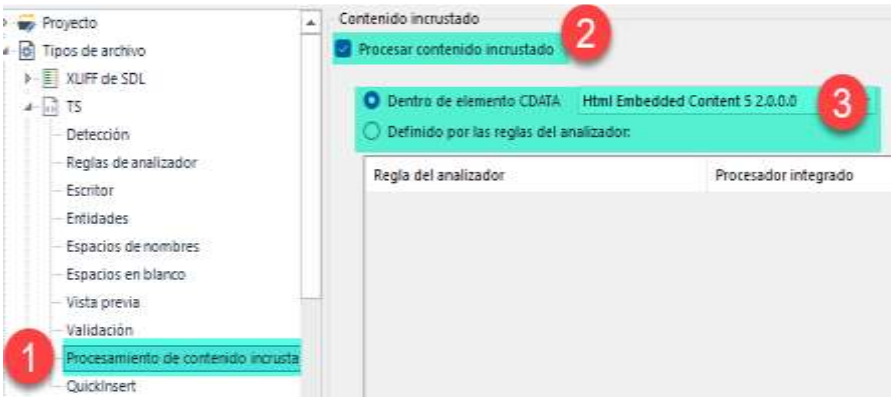
Por defecto, Studio elimina las líneas vacías que sobren en los archivos de destino. En archivos con una estructura compleja, es preferible forzar a Studio a conservarlas activando la opción correspondiente en la sección **Espacios en blanco.** Además, así será más fácil revisarlos con un comparador de archivos.



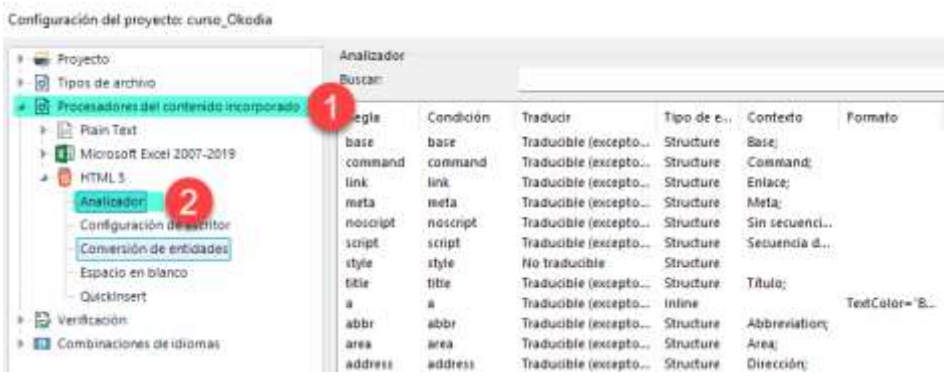
Bloqueo de etiquetas HTML y variables

Al igual que ocurre con los archivos de texto delimitado, es frecuente encontrar en los archivos de tipo XML variables o códigos HTML que tendremos que convertir en etiquetas por los mismos motivos comentados antes.

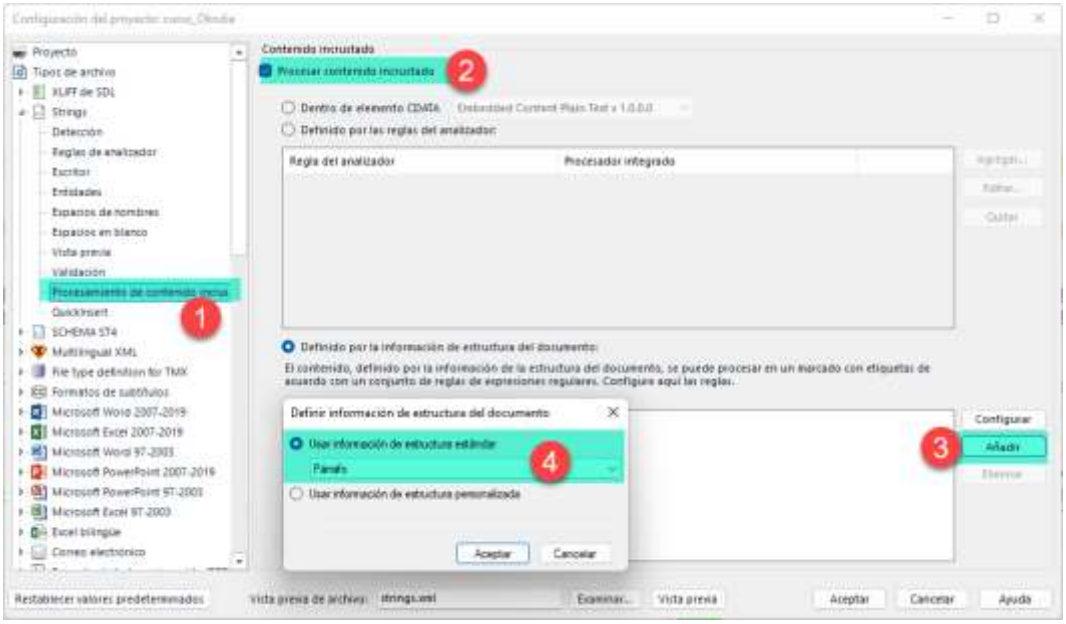
En el caso de las etiquetas HTML, podemos añadirlas una a una o en bloque de forma manual. Sin embargo, es más fácil y rápido usar el procesador para texto HTML propio de Studio si solo hay etiquetas HTML y no variables en el archivo. Elegiremos una de las dos opciones señaladas en la captura en función de si hay secciones CDATA (mediante las cuales se indica al programa que el texto que sigue no es código XML) en el archivo de origen o no.



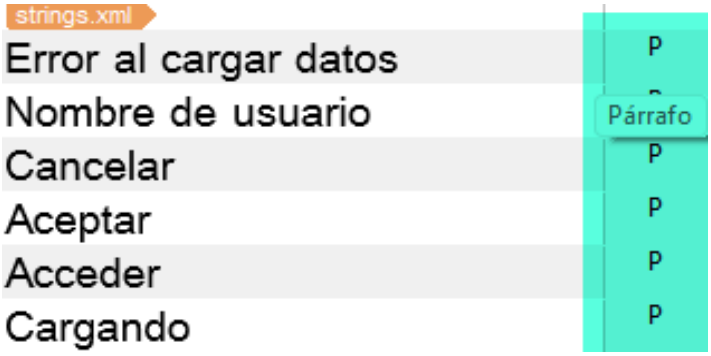
Si usamos el procesador de HTML, podemos modificar como queramos el comportamiento y aspecto de estas etiquetas en **Procesadores del contenido incorporado > HTML 5 > Analizador.**



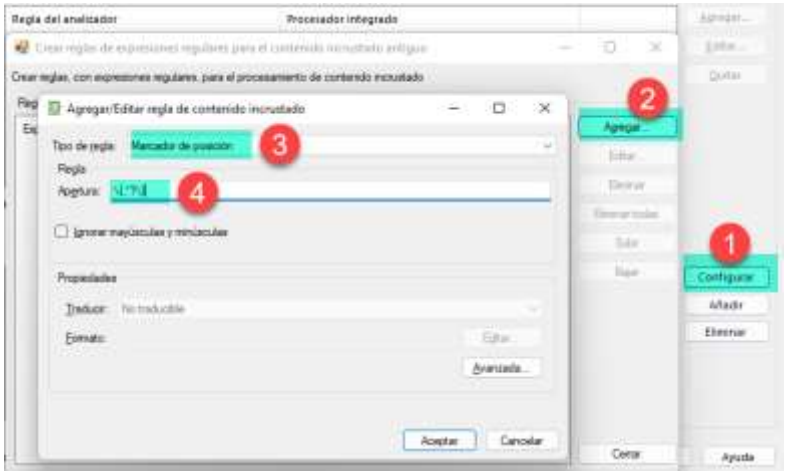
Si añadimos las etiquetas HTML o variables a mano, tenemos que indicar a Studio en qué secciones se deben bloquear las etiquetas (dependerá del XML en concreto). En el archivo **strings.xml,** estas variables aparecen en los párrafos.



Un truco para saber cuáles son estas secciones es usar la vista previa de Studio. Si posamos el ratón en la columna de la derecha, Studio nos indica el tipo de sección en la que aparece ese segmento.

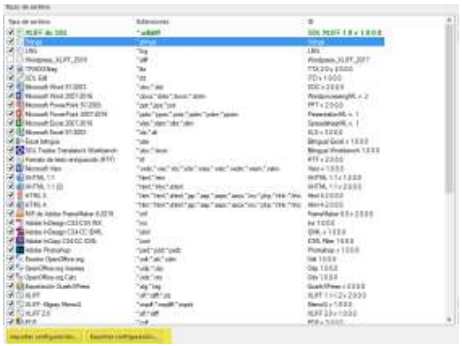


Ahora añadimos la regla **\{.*?\}** para bloquear las variables existentes y después cambiamos el comportamiento a Incluir en las opciones avanzadas para que Studio las muestre en todos los casos.



Cómo guardar el filtro para usarlo en futuros proyectos

Como hemos visto antes, podemos exportar el filtro como un archivo **sdlftsettings** para reutilizarlo en el futuro. Para tal fin, en la **Configuración del proyecto > Tipos de archivo,** seleccionamos el filtro que queramos guardar y hacemos clic en el botón **Exportar configuración** situado en la esquina inferior izquierda de la ventana.



La pseudotraducción

Del mismo modo que describimos con los archivos de texto delimitado, podemos recurrir a la pseudotraducción para comprobar que el filtro funciona correctamente. Accedemos a dicha función desde **Tareas por lotes > Pseudotraducir.**

Luego se exporta el archivo de destino para comprobar que se crea sin problemas y que no hemos dejado texto traducible sin importar (o viceversa). Si detectamos errores, tendremos que repasar el filtro y volver a importar el archivo hasta que todo funcione correctamente.