

# ¿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

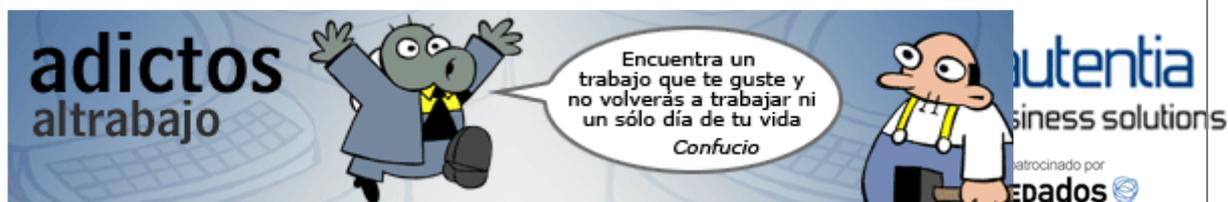
Tareas programadas (Quartz)  
 Gestor documental (Alfresco)  
 Inversión de control (Spring)

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

BPM (jBPM o Bonita)  
 Generación de informes (JasperReport)  
 ESB (Open ESB)

E-mail: Contraseña: 

Deseo registrarme

He olvidado mis datos de acceso

[Inicio](#) [Quiénes somos](#) [Tutoriales](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#) [Charlas](#) [Más](#)

Estás en:

[Inicio](#) [Tutoriales](#) [Java i18n Properties Encoding](#)



**DESARROLLADO POR:**  
Francisco Ferri Pérez

Consultor tecnológico de desarrollo de proyectos informáticos.

Desarrollador de proyectos informáticos, Microsoft Certified IT Professional - Enterprise Administrator

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Catálogo de servicios  
Autentia



[Anuncios Google](#)

[Java](#)

[Web 2.0 Java](#)

[Programming in Java](#)

**Fecha de publicación del tutorial: 2009-02-26**



Share |

[Regístrate para votar](#)

# Java i18n Properties Encoding

## Índice:

- 1. Introducción
- 2. Exposición
- 3. Proposición
- 4. Soluciones
- 5. Consideraciones
- 6. Code Draft
- 7. Proyecto en SourceForge
- 8. Knowledge Request
- 9. Nota del autor

## Introducción

Versión de Java en el momento de escribir este artículo 1.6.0\_22.  
IDE utilizado, Eclipse Helios SR1 (última release hasta el momento).

Voy a exponer el tema de la internacionalización en java mediante ficheros **.properties**, y su problemática, que siempre supone algunos quebraderos de cabeza para aquellos que sigan las instrucciones originales de (lo que era) SUN (ahora Oracle) al respecto.

Una vez quede expuesto el problema a lo largo de este documento, no nos quedaremos ahí, sino que como es habitual en Autentia, voy a proponer una manera de enfrentarse a él y como novedad voy a cargar el código fuente en sourceforge a modo de proyecto, para que todos podamos colaborar en mejorarlo o pedir cambios, que pueden servir para otros en el futuro.

## Últimas Noticias

[Java Code Review](#)

Comentando el libro: Nunca comas solo de Keith Ferrazzi y Tahl Raz.

XII Charla Autentia - LiquiBase - iiiEmisión en Directo!!!

Actualización en los esquemas del tutorial: "Cómo alcanzar el éxito en el sector de la informática"

Comentado: Ingeniería de Software Ágil de E.M. Jimenez

Curso de TDD con Enrique Comba Riepenhausen

[Histórico de NOTICIAS](#)

Últimos Tutoriales



El problema tiene su origen en la política/filosofía de mantener siempre compatibilidad hacia atrás entre versiones anteriores. Si nos remontamos a la versión 1.3.1\_20 (última para Windows, o a la versión 1.3.1\_25 última para Solaris, sendas tuvieron su EOL (End Of Life) el 25 de Octubre de 2004) nos encontramos que podemos bajar una versión "English" y luego un paquete distinto de internacionalización. Esta versión "English" que es la base, venía codificada a ISO-8859-1.

Aunque parezca prehistoria hablar de Java 1.3, aún existen contextos en los que se utiliza. En mi vida profesional podría enumerar varios proyectos en Bancos que siguen en producción basándose en ésta. Por eso cuando creo librerías me gusta que sean compatibles hacia atrás :)

Del siguiente link podéis descargar cualquier versión que haya existido, tanto **JDK**, **JRE** como la **Documentation**: [Oracle Tech Network](#)

## Exposición

Por defecto los ficheros .properties se codifican con el encoding ISO-8859-1, suponiendo un trabajo adicional codificar por ejemplo caracteres en Japonés u otro lenguaje que salga de esa codificación.

Por ejemplo, para escribir en castellano, inglés y chino la palabra "Añadir" tendríamos que escribir en nuestros ficheros .properties:

### Encoding ISO-8859-1

language\_es.properties => Castellano: **A\u00F1adir**

language\_en.properties => Inglés: **Add**

language\_zh.properties => Chino: **\u6DFB\u52A0**

A los sufridos expertos en la materia seguro que les viene a la mente algún plugin de su IDE que les hace el trabajo sucio como Eclipse RBE o la aplicación incluida en la máquina virtual de Java: native2ascii tool.

¿A caso no es esto un problema que nos resta tiempo cada vez que queramos añadir idiomas a cualquier aplicación?, en mi humilde opinión **SI**. Creo que es un trabajo muy costoso, que además termina en algo que si no utilizamos un IDE o ayuda contextual es casi inmantenible.

Imaginemos por ejemplo un fichero lleno de:

**\u6DFB\u52A0, A\u00F1adir, ...**

Y que tengamos que cambiar algunos textos de ese fichero o simplemente crear nuevos idiomas con el mismo modus operandi... seguro que nos viene a la cabeza aquello de: ¡¡NOOOO!!

Segun Sun (ahora Oracle): Los caracteres que se salen del encoding ISO-8859-1 debemos codificarlos así dentro del fichero .properties puesto que java al cargar el fichero como un objeto Properties lo interpreta con el susodicho encoding, y cito textualmente:

API JDK 1.6: [java.util.Properties](#)

**class** java.util.Properties

The load(Reader) / store(Writer, String) methods load and store properties from and to a character based stream in a simple line-oriented format specified below. The load (InputStream) / store(OutputStream, String) methods work the same way as the load (Reader)/store(Writer, String) pair, **except the input/output stream is encoded in ISO 8859-1 character encoding. Characters that cannot be directly represented in this encoding can be written using Unicode escapes ; only a single 'u' character is allowed in an escape sequence.** The native2ascii tool can be used to convert property files to and from other character encodings.

**iBe water my friend!**

## Proposición

¿No sería mejor simplemente tener los ficheros .properties en el encoding que nosotros deseáramos y tuviéramos la opción de indicárselo a Java para que lo tratase correctamente?

Nos evitaríamos así tener que codificar los caracteres del mismo. Es decir para escribir "Añadir" no escribiríamos "A\u00F1adir" sino simplemente "Añadir", o para escribirlo en chino no escribiríamos "\u6DFB\u52A0" sino "新增". Siendo muchísimo más fácil y humano mantener ese fichero de idioma.

Veamos cómo quedaría el fichero anteriormente expuesto por ejemplo codificado en UTF-8:

### Encoding ISO-8859-1

language\_es.properties => Castellano: **A\u00F1adir**

Construcción de componentes en wuija por composición

 Reunión Madrid Ágil 02-11-2010: DDD (Domain Driven Design)

 DataTable con paginación en base de datos con Primefaces

 Resolviendo el cubo de Rubik (II) - ayudas, ejemplo Wink, extensiones y encuesta

 Configuración de jCaptcha en Struts

Últimos Tutoriales del Autor

 Instalación de Ubuntu Desktop 10.04 LTS 32bits en una máquina virtual con VMWare Workstation

 Instalación de Ubuntu Server 8.04 LTS 32bits en una máquina virtual con VMWare Workstation

 Instalación de Ubuntu Server 10.04 LTS 32bits en una máquina virtual con VMWare Workstation

 Instalación de Ubuntu Desktop 8.04 LTS 32bits en una máquina virtual con VMWare Workstation

 Crear una máquina virtual con VMWare Workstation

Síguenos a través de:



Últimas ofertas de empleo

2010-10-11  
 Comercial - Ventas - SEVILLA.

2010-08-30  
 Otras - Electricidad - BARCELONA.

language\_en.properties => Inglés: **Add**

language\_zh.properties => Chino: **\u6DFB\u52A0**

## Encoding UTF-8

language\_es.properties => Castellano: **Añadir**

language\_en.properties => Inglés: **Add**

language\_zh.properties => Chino: **新增**

Visto así casi que podríamos mandar el fichero .properties al traductor, en muchos casos llamado: [Google](#), y de apellido: [Translator](#). Sigue siendo un trabajo tedioso, pero al no tener que convertir: **Añadir** a **A\u00F1adir**, insisto que simplemente es mucho más humano y mantenible por terceros.

Veamos a continuación que para nuestra tranquilidad sí es posible.

## Soluciones

### Solución 1 - "Sólo para valientes"

La primera forma de atacar al problema es pasar en tercera de los .properties y utilizar ficheros XML y un api tipo JDom por ejemplo. Adaptando nuestro código. Esto no está contemplado en este documento, lo expongo simplemente como una idea.

#### Nota para Java 1.7

New Feature - Language-level XML:

Language-level XML support would add support for literal XML construction, data conversion, navigation, and streaming.

Parece ser que variantes a la solución 1 serán más factibles a partir de JDK 1.7, y sobre todo por el incremento de rendimiento que parece tendrá la 1.7 respecto de la 1.6/1.5

### Solución 2 - "Lucha mínima"

La segunda forma de atacar al problema es cargar el fichero .properties indicando el encoding en el momento de su lectura, en el que está formateado y trasformarlo al encoding deseado.

**Nota:** hay ocasiones en las que desconocemos el encoding del fichero .properties, bien porque no lo hemos creado nosotros, o bien porque no tenemos un entorno de desarrollo (IDE tipo Eclipse) o aplicación tipo Notepad++ que nos lo diga...

Si desconocemos el encoding del fichero .properties, para acometer la solución 2 nos encontramos con un problema, puesto que Java no dispone de un mecanismo para averiguar el encoding de un fichero, esto es así por la propia naturaleza del encoding.

Aunque existen algoritmos para intentar averiguarlo, por ejemplo:

**Mozilla** [UniversalCharsetDetection](#) y su implementación [JCharDet](#)

**CodeHaus** [GessEnc](#)

**ICU Project**

Básicamente funcionan leyendo los caracteres del fichero e intentando casar el conjunto en un encoding que los soporte, o también apoyándose en el BOM.

Estos algoritmos, aunque tienden a una precisión altísima, casi del 100%, seguro que en un idioma muy concreto o menos extendido nos encontramos alguna sorpresa. Aún así las tendremos en consideración en nuestra pequeña librería.

## Consideraciones

Hasta aquí parece que la solución es, si tenemos un fichero .properties con encoding ISO-8859-1, cogemos nuestro IDE favorito y le cambiamos el encoding a UTF-8 y utilizamos esta librería que estamos creando. La respuesta es **NO**.

Si cambiamos con en el eclipse o cualquier otro IDE el encoding tendremos que el fichero se convierte en algo repleto de caracteres extraños, de nuevo esto es por la propia naturaleza del encoding.

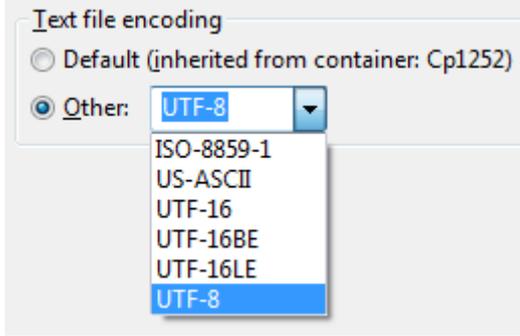
2010-08-24

 [Otras Sin catalogar](#)  
- LUGO.

2010-06-25

 [T. Información -](#)  
[Analista /](#)  
[Programador -](#)  
[BARCELONA.](#)

Por ejemplo en Eclipse sería, en las opciones del fichero:



### Encoding original ISO-8859-1

```
add      = Añadir
new      = Nueva búsqueda
save     = Guardar búsqueda
send     = Compartir con un amigo
```

### Encoding cambiado manualmente UTF-8

```
add      = Añadir
new      = Nueva bñsqueda
save     = Guardar bñsqueda
send     = Compartir con un amigo
```

Para tener un fichero correcto tenemos que crear un fichero nuevo desde 0, y estando vacío asignarle el encoding y escribir su contenido. O copiar el contenido de otro y arreglar todos los caracteres extraños aparecidos.

Entonces podremos utilizar este API correctamente

## Code Draft

El api que acompaña a este documento dispondrá de varias opciones para hacer la carga de los ficheros .properties, ésta es mi propuesta:

### Sin nuestro api tenemos el método original de Java:

**ResourceBundle.getBundle(..)**

### Con nuestro api:

1. "El menos adecuado" - Cargar el fichero .properties asumiendo que su encoding original es ISO 8859-1 (Java .properties default) y tratándolo como de un UTF-8:

**ResourceBundleEx.getBundle(..)**

Esto nos permite trabajar con los .properties originales, sin cambiarles el encoding, pero nos evita que se muestren caracteres extraños, estando el límite en todo lo que el UTF-8 soporte.

Advertencia: Pueden no mostrarse correctamente algunos caracteres.

2. "El más adecuado" - Cargar el fichero .properties habiendo establecido su encoding a cualquiera que nosotros indiquemos y lo trataremos como el que nos convenga (clase Charset)

**ResourceBundleEx.getBundle(.., "ISO-8859-1", "UTF-8")**

**ResourceBundleEx.getBundle(.., Charset, Charset)**

Esta opción es la mas custom, nos permite trabajar con un .properties al que le hayamos cambiado el encoding como nos haya parecido.

3. "No incluido en el api de este tutorial" - Cargar el fichero .properties detectando su encoding automáticamente mediante algún algoritmo de detección y tratándolo como el encoding que nosotros indiquemos (clase Charset)

**ResourceBundleEx.getBundle(.., "UTF-8")**

**ResourceBundleEx.getBundle(.., Charset)**

Con esta opción nos la jugamos con un algoritmo de detección automática de encoding, que como hemos visto no son 100% fiables por la naturaleza propia del encoding, pero es tan válida como la opción 2.

## Proyecto en SourceForge

Disponible en: <http://sourceforge.net/projects/javai18npropenc/>

Para bajar el código fuente con subversion:  
<https://javai18npropenc.svn.sourceforge.net/svnroot/javai18npropenc>

Dejo para descargar las 2 librerías y sus respectivos proyectos de test donde utilizamos el api a modo de ejemplo sobre ficheros .properties con diferentes encondings para cada caso.

i18npropenc13.jar -> versión para JDK 1.3, 1.4 y 1.5

i18npropenc16.jar -> versión para JDK 1.6

## Knowledge Request

Todo esto ha sido creado con la intención de compartir conocimiento y apoyaros en vuestros proyectos, espero que mediante los comentarios en esta página y los del propio proyecto de sourceforge:

Expreséis todas vuestras dudas.

Hagáis feature-requests.

Notifiquéis bugs :)

**Y sobre todo espero que os sea de utilidad el código.**

### Nota del autor

Este documento está escrito con el único fin de informar de forma abierta y gratuita al lector respecto del tema que trata, no está exento de contener imprecisiones o información incorrecta.

Si usted quiere añadir cualquier información al documento por favor siéntase libre de ponerse en contacto con el autor directamente en su correo electrónico [fferri@autentia.com](mailto:fferri@autentia.com). Así todos conseguimos tener la mejor información posible.

Si desea comentar el documento dispone de los comentarios habilitados a continuación del mismo. Estamos encantados de saber qué opina o si le ha sido de utilidad.

Muchas gracias por su tiempo, espero que esta lectura le haya sido como mínimo de tanto provecho como para mí fue escribir el documento.

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

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

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

## COMENTARIOS



Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

