Encoding

Tabla de contenido

[Referencias - 2 -](#_Toc469477265)

[Detección del tipo de error - 2 -](#_Toc469477266)

[CP-1252 => UTF-8 - 2 -](#_Toc469477267)

[UTF-8 => CP-1252 - 2 -](#_Toc469477268)

[Combinación - 2 -](#_Toc469477269)

[Configuración - 2 -](#_Toc469477270)

[JVM - 2 -](#_Toc469477271)

[Tomcat - 3 -](#_Toc469477272)

[Apache - 3 -](#_Toc469477273)

[OpenCms - 3 -](#_Toc469477274)

[Orígen - 3 -](#_Toc469477275)

[Librerías Java - 3 -](#_Toc469477276)

[Solución - 4 -](#_Toc469477277)

[Ficheros de mensajes - 4 -](#_Toc469477278)

[Solución - 4 -](#_Toc469477279)

[Respuesta del servidor e interpretación del navegador - 4 -](#_Toc469477280)

[Solución - 4 -](#_Toc469477281)

[Formularios - 5 -](#_Toc469477282)

[Solución - 5 -](#_Toc469477283)

[Base de datos - 6 -](#_Toc469477284)

[Solución - 6 -](#_Toc469477285)

# Referencias

* Blog: <http://apuntesadeteran.blogspot.com.es/2013/06/abordando-problemas-de-encoding-en-java.html>
* Stackoverflow: <http://stackoverflow.com/questions/138948/how-to-get-utf-8-working-in-java-webapps>

# Detección del tipo de error

## CP-1252 => UTF-8

Si vemos caracteres � en vez de vocales acentuadas o eñes, la aplicación está interpretando como UTF-8 un texto que en realidad está almacenado como CP-1252 o ISO-8859.

Ej. año => a�o

## UTF-8 => CP-1252

Si vemos que las eñes o vocales acentuadas son sustituidas por **dos** caracteres "extraños", la aplicación está interpretando como CP-1252 o ISO-8859 un texto que en realidad está codificado como UTF-8.

Ej. año => aÃ±o

## Combinación

En general, si vemos que las eñes o vocales acentuadas son sustituidas por **2n** caracteres "extraños", se está cometiendo el error anterior **n** veces a lo largo del ciclo de vida del texto.

Ej. año => aÃƒÂ±o

# Configuración

## JVM

La máquina virtual de java debe arrancarse con la opción:

JAVA\_OPTS="-Dfile.encoding=UTF-8"

## Tomcat

1. El conector 8080 de tomcat debe tener la opción URIEncoding a UTF-8. Esto podemos verlo en el fichero **$TOMCAT/conf/context.xml**:

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000" URIEncoding="UTF-8"

redirectPort="8443" />

Opcionalmente, podemos configurar [estos ficheros](#_Solución) para evitar errores de encoding en el paso de parámetros para el caso de los formularios.

## Apache

1. Debemos añadir URIEncoding="UTF-8" en el fichero $TOMCAT/conf/server.xml para el conector 8009 cuando usamos mod\_jk:

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" URIEncoding="UTF-8"/>

1. En el fichero de configuración de Apache **/etc/httpd/conf/httpd.conf** debemos añadir (o descomentar si ya existe):

AddDefaultCharset utf-8

* Para saber la ruta donde está instalado apache podemos hacer:

ps -aux | grep httpd

#apache 28817 0.0 0.1 316864 29404 ? S 03:27 0:00 /usr/sbin/httpd -DFORE

/usr/sbin/httpd -V | grep SERVER\_CONFIG\_FILE

## OpenCms

1. Configurar la indexación para ignorar las tildes editando el fichero $opencms/WEB-INF/solr/conf/schema.xml agregando en el filtro **solr.ASCIIFoldingFilterFactory** la configuración de tipo de campo **text\_general:**

**<filter class="solr.ASCIIFoldingFilterFactory"/>**

# Orígen

## Librerías Java

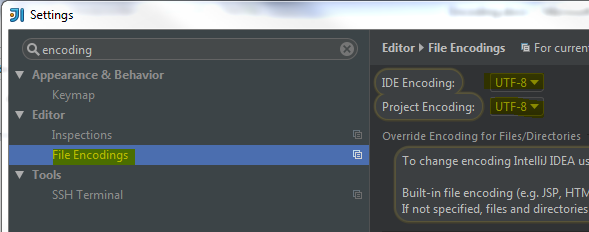
Es el compilador el que lee el fichero y genera el .class correspondiente. ¿Qué codificación usa el compilador? Pues por defecto utiliza la misma que la plataforma en la que se esté ejecutando. Esto es, si compilamos nuestros fuentes en Windows, se usará CP-1252. Si compilamos en Linux, se usará UTF-8.

### Solución

Configurar el IDE para que fuerce el encoding a UTF-8.



En File/Settings



## Ficheros de mensajes

Los ficheros .properties o, en definitiva, los ResourceBundle son interpretados en ISO-8859-1.

### Solución

1. Utilizar el formato \uXXXX, donde XXXX es el código Unicode del carácter.

## Respuesta del servidor e interpretación del navegador

Es posible que el HTML se esté enviando por la red con una codificación, y el navegador la esté interpretando con otra.

Para determinar la codificación que debe emplear un navegador a la hora de mostrar el HTML de una response HTTP, se utiliza la cabecera HTTP **Content-Type**.

### Solución

En JSP:

<%@ page contentType="text/html;charset=UTF-8"%>

Si esta cabecera no aparece, o no indica la codificación, el navegador busca una etiqueta meta presente en el HTML de la página, que a su vez puede ponerse como

<meta http-equiv='Content-Type' content='text/html; charset=utf-8'>

o como

<meta charset='utf-8'>

* Tomcat 6 parece usar ISO-8859-1, independientemente de lo que especifiquemos en la propiedad file.encoding.

## Formularios

Cuando se hace submit de un formulario, el navegador utiliza en el request la misma codificación que haya usado para mostrar la página.

En el lado del servidor, la clase **ServletRequest** tiene un método **getCharacterEncoding**(), que debería devolvernos la codificación de la request. Este método lee el encoding de la cabecera HTTP correspondiente. Sin embargo, los navegadores no suelen enviar esta información por lo que debemos usar una de forma consistente en el response, y asumir que las request nos llegan con esa misma codificación.

* Hay una gran excepción a esta regla (encoding de la request = encoding de la página del formulario), que sucede cuando usamos Ajax. Ajax usa por defecto UTF-8 en el request.

### Solución

1. El encoding utilizado en el envío de parámetros se puede modificar mediante el atributo **accept-charset** de la etiqueta **form**, indicando el **encoding** a usar (sólo el encoding, no todo el Content-Type). Sin embargo, no funciona en todos los navegadores, por lo que no es conveniente utilizarlo.
2. Se puede especificar de forma explícita que el request viene codificada como UTF-8. Esto se puede hacer con el método **setCharacterEncoding(String)** de la clase ServletRequest. Este método simplemente establece con qué codificación se debe leer la request por lo que hay que ejecutarlo antes de leer algún parámetro de la request.
3. En JEE es muy sencillo establecer la codificación de request y response en un filtro y configurarlo en el web.xml. Para esto en el fichero **$TOMCAT/conf/web.xml** disponemos de:

<filter>

<filter-name>setCharacterEncodingFilter</filter-name>

<filter-class>org.apache.catalina.filters.SetCharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<async-supported>true</async-supported>

</filter>

1. También podemos forzar a la aplicación a usar UTF-8 en todas las JSPs. Para esto editamos el fichero **$TOMCAT/webapps/$APP/WEB-INF/web.xml** añadiendo la siguiente directiva:

<jsp-config>

<jsp-property-group>

<url-pattern>\*.jsp</url-pattern>

<page-encoding>UTF-8</page-encoding>

</jsp-property-group>

</jsp-config>

Esto equivale a añadir en todas las JSPs la directiva:

<%@page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>

## Base de datos

Al conectar con una base de datos enviamos y recibimos los datos con un encoding determinado. Debemos asegurarnos que tanto la aplicación como la base de datos utilizan la misma codificación.

### Solución

1. Para ello disponemos de los parámetros de la cadena de conexión. En el caso de MySQL debemos asegurarnos de incluir el parámetro:

"?useEncoding=true&amp;characterEncoding=UTF-8"

1. Cuando creamos una tabla podemos incluir el encoding:

CREATE TABLE `Users` (

`id` int(10) unsigned NOT NULL auto\_increment,

`name` varchar(30) collate utf8\_swedish\_ci default NULL

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT **CHARSET=utf8**;

1. En los ficheros **my.cnf** y **my.ini** de configuración del servidor para MySQL se le puede indicar el encoding por defecto:

[client]

port=3306

default-character-set=utf8

[mysql]

default-character-set=utf8