

¿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)



Estás en: Inicio » Tutoriales » Como intentar averiguar el juego de caracteres de un archivo



DESARROLLADO
POR:
Francisco J. Arroyo

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE



Fecha de publicación del tutorial: 2011-08-10



Share |

Regístrate para votar

Como intentar averiguar el juego de caracteres de un archivo

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Requisitos previos.
- 4. Descripción de los juegos de caracteres.
- 5. Usando librerías para averiguar el encoding.
- 6. Conclusiones.
- 7. Referencias.

1. Introducción

Cuando recibimos un archivo, si queremos leerlo correctamente, debemos saber en que codificación se ha guardado para utilizar la misma codificación a la hora de leerlo. Si nos equivocamos de codificación, seguramente nos aparecerán caracteres extraños o no reconocibles.

El encoding es la correspondencia que damos a un carácter de un determinado juego de caracteres con un número que le identifica. Por ejemplo, si esogemos ASCII como juego de caracteres, vemos que para representar la letra A se utiliza el número 65. Los principales juegos de caracteres, o por lo menos los que con mayor frecuencia me he encontrado son: ISO-8859-1, UTF-8, MacRoman, ISO-8859-15, cp1252.

En este tutorial vamos a comentar muy brevemente los juegos de caracteres y mostrar un par de librerías de las que podemos hacer uso para intentar averiguar la codificación de un stream de bytes.

2. Entorno

- Mackbook Pro
 - Intel Core i7 2Ghz
 - 8GB RAM
 - 500GB HD
 - Sistema Operativo: Mac OS X (10.6.7)

3. Requisitos previos.

En este tutorial no hace falta ningún requisito previo

4. Descripción de los juegos de caracteres.

En este apartado vamos a describir los principales rasgos o peculiaridades de los encodings que hemos comentado anteriormente.

- ASCII
 - Son las siglas de "American Standard Code for Information Interchange" y fue creado en 1963 por el comité estadounidense de estándares (que más tarde sería el instituto nacional estadounidense de estándares, ANSI). El código ASCII utiliza 7 bits para representar los caracteres con lo que es capaz de representar 128 códigos distintos. Los 32 primeros códigos se utilizan para caracteres de control, por ejemplo, el carácter 27 representa la tecla escape (ESC).

TRUCO: Si el stream que estamos leyendo no contiene bytes por encima del 0x7f, entonces está codificado en ASCII.

Catálogo de servicios Autentia

Últimas Noticias

- Pirineos on Tour
- VII Autentia Cycling Day
- Autentia patrocina la charla sobre Java SE 7 en Madrid
- Alfresco Day 2011
- XVII Charla Autentia - Grails - Vídeos y Material

Histórico de NOTICIAS

Últimos Tutoriales

- Uso del componente remoteCommand de primefaces para actualizar el contenido de un componente de lightBox en modo iframe
- Cómo trabajar con JSF2 y el soporte de inyección de dependencias de Spring
- Creando un tema en Liferay 6.0.6
- Usando CallableStatements para ejecutar procedimientos almacenados
- Android: Leer correos de Gmail

Últimos Tutoriales del Autor

- Como desarrollar un plugin para Eclipse
- Google Custom Search Api desde Android
- Instalación de ntfs-3g para Mac OS X
- Consumir un servicio web Axis con Android

Síguenos a través de:

Últimas ofertas de empleo

- ISO-8859-1
 - Es una extensión de ASCII que utiliza 8 bits que da soporte a otros idiomas como el caso del español. También se conoce como ISO Latin 1 . Esta codificación deja los primeros 128 caracteres intactos (los que componen a ASCII) y añaden valores adicionales por encima de los 7 bits. Estos valores son los caracteres diacríticos y letras especiales.
- ISO-8859-15
 - También es conocido como Latin-9. Es una revisión del ISO-8859-1 que incluyo el símbolo del euro. Puede representar inglés, alemán, francés, español, portugués, fines y estonio. Todos los caracteres imprimibles que existen en esta codificación también se encuentran en cp1252.
- MacRoman
 - Era un encoding usado principalmente por Mac OS, actualmente usa UTF-8

Los primeros 128 caracteres son identicos al ASCII por lo que es capaz de representar inglés

TRUCO: Si al leer el stream nos encontramos con los bytes 0x81, 0x8D, 0x8F, 0x90, 0x9D, podemos descartar cp1252 a favor de MacRoman.
- UTF-8
 - Incluye la especificación US-ASCII, por lo que cualquier mensaje ASCII se representa sin cambios. Utiliza simbolos de longitud variable (de 1 a 4 bytes). Si se produce un error al decodificar una cadena, nos mostrará el carácter U+FFFD.

TRUCO: Si el stream que vamos a leer empieza por 0xEF,0xBB,0xBF, podemos decir que está codificado en UTF-8, pero si no empieza por estos valores, no podemos descartar que no sea UTF-8

- 2011-07-06
Otras Sin catalogar - LUGO.
- 2011-06-20
Comercial - Ventas - SEVILLA.
- 2011-05-24
Contabilidad - Especialista Contable - BARCELONA.
- 2011-05-14
Comercial - Ventas - TARRAGONA.
- 2011-04-13
Comercial - Ventas - VALENCIA.

5. Usando librerías para averiguar el encoding.

Ahora vamos a describir par de librerias java que podemos usar para intentar averiguar el encoding.

- **juniversalchardet**
Esta es la librería que mejor resultado me ha dado. Normalmente para averiguar el encoding necesita unas decenas de kilobytes.

```

view plain print ?
01. String detectCharset() {
02.
03.     String encoding;
04.
05.     try {
06.         final FileInputStream fis = new FileInputStream(file.getAbsolutePath());
07.
08.         final UniversalDetector detector = new UniversalDetector(null);
09.         handleData(fis, detector);
10.         encoding = getEncoding(detector);
11.         detector.reset();
12.
13.         fis.close();
14.
15.     } catch (IOException e) {
16.         encoding = "";
17.     }
18.
19.     return encoding;
20. }
21.
22. private String getEncoding(UniversalDetector detector) {
23.     if(detector.isDone()) {
24.         return detector.getDetectedCharset();
25.     }
26.     return "";
27. }
28.
29. private void handleData(FileInputStream fis, UniversalDetector detector) throws IOExcep
30.     int nread;
31.     final byte[] buf = new byte[4096];
32.     while ((nread = fis.read(buf)) > 0 && !detector.isDone()) {
33.         detector.handleData(buf, 0, nread);
34.     }
35.     detector.dataEnd();
36. }

```

- **java.nio.charset**
Esta librería no nos adivina el encoding, pero si podemos especificar un charset e intentar leer un stream de bytes y en caso de que encuentre un carácter que no pueda descodificar nos lanza una excepción. Este método no es efectivo 100%, ya que me he encontrado casos en los que al decodificar un stream no me ha lanzado una excepción y el charset obtenido no era el correcto. Cuanto mayor sea el stream, mayor probabilidad de acierto.

```

view plain print ?
01. String detectCharset() {
02.     for (String charsetName : avaiablesCharsets) {
03.         final Charset charset = detectCharset(file, Charset.forName(charsetName));
04.         if (charset != null) {
05.             return charset.displayName();
06.         }
07.     }
08.     return "";
09. }
10.
11. private Charset detectCharset(File f, Charset charset) {
12.     CharsetDecoder decoder = prepareCharsetDecoder(charset);

```

```

13.     return tryingCharset(f, decoder) ? charset : null;
14. }
15.
16. private CharsetDecoder prepareCharsetDecoder(Charset charset) {
17.     return charset.newDecoder().reset();
18. }
19.
20. private boolean tryingCharset(File f, CharsetDecoder decoder) {
21.     try {
22.         final BufferedInputStream input = new BufferedInputStream(new FileInputStream(f));
23.         final byte[] buffer = new byte[512];
24.         boolean identified = false;
25.
26.         while ((input.read(buffer) != -1) && (!identified)) {
27.             identified = identify(buffer, decoder);
28.         }
29.
30.         input.close();
31.         return identified;
32.     } catch (IOException ioException) {
33.         return false;
34.     }
35. }
36.
37.
38. private boolean identify(byte[] bytes, CharsetDecoder decoder) {
39.     try {
40.         decoder.decode(ByteBuffer.wrap(bytes));
41.     } catch (CharacterCodingException e) {
42.         return false;
43.     }
44.     return true;
45. }

```

- **guessenc**

Esta librería sólo es capaz de reconocer UTF-8, UTF-16LE, UTF-16BE, UTF-32, pero nos puede ser útil si sabemos que lo que vamos a recibir está codificado en uno de esos encodings.

Como experimento o prueba de concepto, escribí un pequeño programa que intenta averiguar el encoding de varios archivos haciendo uso de estas librerías.

El programa es un jar, que se llama aCharsetDetector que podéis bajaros de [aquí](#).

El proyecto lo podéis bajar de [aquí](#).

y aquí tenéis una captura de pantalla mostrando como funciona.

Primero vemos los ficheros con los que vamos a probar haciendo un ls -l en la consola, y a continuación ejecutamos el programa

```

Terminal — bash — 93x13
fjarroyo pruebas_charset]$ls -l files/
total 32
-rw-r--r--@ 1 fjarroyo  staff   39 12 jul 10:28 mac-extended-ascii.txt
-rw-r--r--@ 1 fjarroyo  staff   33 12 jul 10:29 mac-os-roman.txt
-rw-r--r--@ 1 fjarroyo  staff   51 12 jul 10:33 utf8.txt
-rw-r--r--@ 1 fjarroyo  staff   39 12 jul 10:33 windows-latino-1.txt
fjarroyo pruebas_charset]$
fjarroyo pruebas_charset]$java -jar aCharsetDetector.jar files/*
files/mac-extended-ascii.txt: MacRoman(1)
files/mac-os-roman.txt: MacRoman(1)
files/utf8.txt: UTF-8(0)
files/windows-latino-1.txt: ISO-8859-15(1)
fjarroyo pruebas_charset]$

```

En la ejecución, se muestra el nombre del fichero seguido de dos puntos y el encoding que propone la aplicación.

El programa no es efectivo 100%, de hecho falla bastante con muestras pequeñas, pero como prueba de concepto creo que está bien ;).

6. Conclusiones

Si el tamaño de la muestra de la que queremos averiguar el encoding es suficientemente grande, hay bastantes posibilidades de que seamos capaz de adivinar el encoding usando las librerías que se muestran en el punto 5 del tutorial. Si por el contrario, el tamaño es pequeño, se reducen drásticamente las posibilidades de adivinar el encoding porque mucha de las librerías que se pueden utilizar para adivinar el encoding de un archivo lo hacen a través de estadísticas del lenguaje y de los juegos de caracteres.

Y con esto concluye el tutorial, si queréis preguntar cualquier cosa, no dudéis de utilizar el formulario que aparece al final de la página. Un saludo.

7. Referencias.

- <http://www.utf8.com/>
- <http://es.wikipedia.org/wiki/UTF-8>
- <http://docs.codehaus.org/display/GUESSENC/Home>
- <http://download.oracle.com/javase/1.4.2/docs/api/java/nio/charset/package-summary.html>
- <http://code.google.com/p/juniversalchardet/>
- <http://es.wikipedia.org/wiki/ASCII>
- http://en.wikipedia.org/wiki/Mac_OS_Roman

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](#)

Copyright 2003-2011 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

