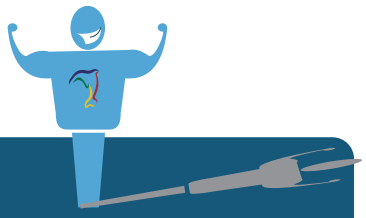
autentia

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...

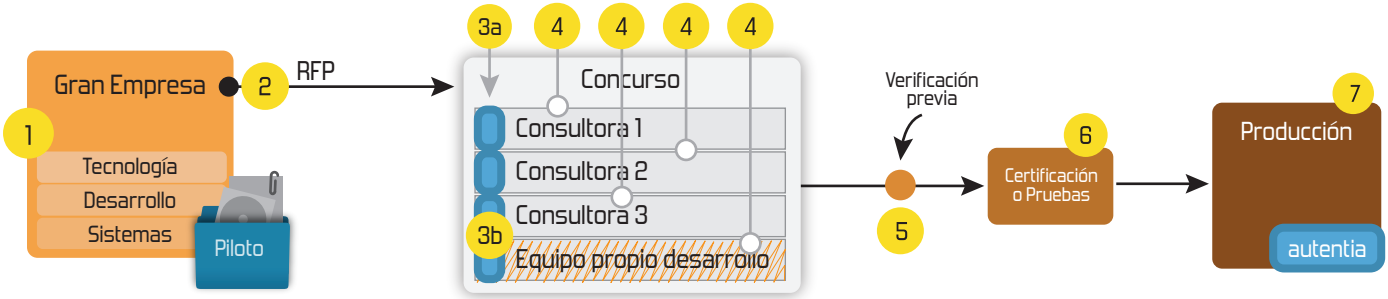
1. Desarrollo de componentes y proyectos a medida



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 conociemiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

DESARROLLADO POR:
Alberto Barranco Ramón

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: 2011-06-28

EpubLib, una librería Java para leer Epub.

0. Índice de contenidos.

1. Introducción

Hoy vamos a ver cómo podemos leer **Epub** (*Electronic Publication*) desde **Java**. Existen algunas librerías para el parseo, la lectura y la generación de Epub, y tras probar algunas, encontramos una que funciona realmente bien y de forma muy sencilla. Se llama **EpubLib**.

EpubLib es un API de Java desarrollado por **Paul Siegmann** y que os podéis descargar de su página principal http://www.siegmann.nl/epublib. Está licenciado bajo **LGPL**. Si queréis saber más sobre la licencia podéis consultar los términos en http://www.gnu.org/licenses/lgpl-java.html

2. Entorno

- Hardware: Portátil MacBook Pro 15' (2.0 GHz Intel i7, 8GB DDR3 SDRAM, 500GB HDD).
- AMD Radeon HD 6490M 256 MB
- Sistema Operativo: Mac OS X Snow Leopard 10.6.7
- Software: Eclipse Helios, EpubLib 3.0-SNAPSHOT

3. Extrayendo la metainformación de un libro

Últimas Noticias

Alfresco Day 2011
XVII Charla Autentia - Grails - Vídeos y Material
¡¡¡ 15 millones de descargas de tutoriales !!!
XVII Charla Autentia - Grails
Charla en WhyFLOSS en el IE: la ppt

Histórico de NOTICIAS

Últimos Tutoriales

Eclipse Indigo, la versión 3.7 de Eclipse
Trabajando con GIT, introducción al uso de los branch y git-completion.bash
Introducción a Spring Security 3.1
Implementando SSO con CAS: ejemplo práctico
Como desarrollar un plugin para Eclipse

Para este tutorial vamos a utilizar el epub "Treasure Island" o en español, "La isla del Tesoro". Los epub no son más que un fichero comprimido con extensión **.epub**, y si los descomprimimos dentro tienen varios ficheros en formato xhtml. De entre ellos, uno de los más importantes es aquel con extensión **.opf**. En este fichero se especifica la metainformación del libro: Autor, editorial, tema del libro, descripción, etc...

A continuación vamos a ver el contenido del fichero content.opf de nuestro epub Treasure Island. Empieza de la siguiente forma.

view plain print ?

```
01. <?xml version="1.0" ?>
02. <package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-
    identifier="BookId">
03.     <metadata xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http:/
04.             <dc:title>Treasure Island</dc:title>
05.             <dc:creator>Robert Louis Stevenson</dc:creator>
06.             <dc:date>2009-02-14</dc:date>
07.             <dc:subject>Youth</dc:subject>
08.             <dc:language>en</dc:language>
09.             <dc:publisher>Web Books Publishing</dc:publisher>
10.             <dc:identifier id="BookId">web-books-309</dc:identifier>
11. </metadata>
```

Para extraer esta información con **EpubLib** tan solo tendríamos que hacer lo siguiente:

view plain print ?

```
01. package com.autentia.EpubLibTest;
02.
03. import static org.junit.Assert.*;
04.
05. import java.net.URL;
06.
07. import nl.siegmann.epublib.domain.Book;
08. import nl.siegmann.epublib.epub.EpubReader;
09.
10. import org.junit.BeforeClass;
11. import org.junit.Test;
12. import org.springframework.core.io.ClassPathResource;
13.
14. public class ReaderTest {
15.
16.     private static URL bookURL;
17.
18.     private static EpubReader epubReader;
19.
20.     private static Book book;
21.
22.     @BeforeClass
23.     public static void setBook() throws Exception {
24.
25.         bookURL = new ClassPathResource("Treasure_Island.epub").getURL();
26.         epubReader = new EpubReader();
27.         book = epubReader.readEpub(bookURL.openStream());
28.     }
29.
30.     @Test
31.     public void metadataIsCorrect() {
32.
33.         assertEquals("Robert Louis", book.getMetadata().getAuthors().get(0)
34.         assertEquals("Stevenson", book.getMetadata().getAuthors().get(0).ge
35.
36.         assertEquals("application/epub+zip", book.getMetadata().getFormat()
37.
38.         assertEquals("en", book.getMetadata().getLanguage());
39.
40.         assertEquals("Youth", book.getMetadata().getSubjects().get(0));
41.
42.         assertEquals("Treasure Island", book.getMetadata().getTitles().get
43.
44.         assertEquals("web-books-
    309", book.getMetadata().getIdentifiers().get(0).getValue());
45.
46.         assertEquals("2009-02-
    14", book.getMetadata().getDates().get(0).getValue());
47.     }
48.
49. }
```

Como veis, tan solo me he creado un método setBook (línea 23) en un @BeforeClass para cargar el libro una sola vez. De esta forma estará disponible para todos los métodos de test de esta clase. En

este primer test comprobamos que la metainformación se corresponde con lo que tiene el archivo **content.opf**.

Si os fijáis en la línea 25 estamos usando **ClassPathResource**, que es una clase de Spring. Si queréis cargar el fichero **.epub** de esta forma solo tenéis que añadir a vuestro POM la siguiente dependencia.

view plain print ?

```
01. <dependency>
02.     <groupId>org.springframework</groupId>
03.     <artifactId>spring-context</artifactId>
04.     <version>3.0.5.RELEASE</version>
05. </dependency>
```

4. Leyendo un Epub

Vamos a meternos dentro del archivo **content.opf** de nuevo. Esta vez nos vamos a fijar más abajo. Concretamente en los apartados **<manifest>** y **<spine toc="ncx">**. El contenido es el siguiente:

view plain print ?

```
01. <manifest>
02.     <item id="ncx" href="toc.ncx" media-type="application/x-
    dtbncx+xml" />
03.      <item id="W000Title" href="000Title.html" media-
    type="application/xhtml+xml" />
04.      <item id="W01ZB309" href="01ZB309.html" media-
    type="application/xhtml+xml" />
05.      <item id="W021B309" href="021B309.html" media-
    type="application/xhtml+xml" />
06.      <item id="W03MB309" href="03MB309.html" media-
    type="application/xhtml+xml" />
07.      <item id="W04MB309" href="04MB309.html" media-
    type="application/xhtml+xml" />
08.      <item id="W05MB309" href="05MB309.html" media-
    type="application/xhtml+xml" />
09.      <item id="W06MB309" href="06MB309.html" media-
    type="application/xhtml+xml" />
10.      <item id="W07MB309" href="07MB309.html" media-
    type="application/xhtml+xml" />
11.      <item id="W08ZB309" href="08ZB309.html" media-
    type="application/xhtml+xml" />
12.      <item id="W091B309" href="091B309.html" media-
    type="application/xhtml+xml" />
13.      <item id="W10MB309" href="10MB309.html" media-
    type="application/xhtml+xml" />
14.      <item id="W11MB309" href="11MB309.html" media-
    type="application/xhtml+xml" />
15.      <item id="W12MB309" href="12MB309.html" media-
    type="application/xhtml+xml" />
16.      <item id="W13MB309" href="13MB309.html" media-
    type="application/xhtml+xml" />
17.      <item id="W14MB309" href="14MB309.html" media-
    type="application/xhtml+xml" />
18.      <item id="W15ZB309" href="15ZB309.html" media-
    type="application/xhtml+xml" />
19.      <item id="W161B309" href="161B309.html" media-
    type="application/xhtml+xml" />
20.      <item id="W17MB309" href="17MB309.html" media-
    type="application/xhtml+xml" />
21.      <item id="W18MB309" href="18MB309.html" media-
    type="application/xhtml+xml" />
22.      <item id="W19ZB309" href="19ZB309.html" media-
    type="application/xhtml+xml" />
23.      <item id="W201B309" href="201B309.html" media-
    type="application/xhtml+xml" />
24.      <item id="W21MB309" href="21MB309.html" media-
    type="application/xhtml+xml" />
25.      <item id="W22MB309" href="22MB309.html" media-
    type="application/xhtml+xml" />
26.      <item id="W23MB309" href="23MB309.html" media-
    type="application/xhtml+xml" />
27.      <item id="W24MB309" href="24MB309.html" media-
    type="application/xhtml+xml" />
28.      <item id="W25MB309" href="25MB309.html" media-
    type="application/xhtml+xml" />
29.      <item id="W26ZB309" href="26ZB309.html" media-
    type="application/xhtml+xml" />
30.      <item id="W271B309" href="271B309.html" media-
    type="application/xhtml+xml" />
31.      <item id="W28MB309" href="28MB309.html" media-
    type="application/xhtml+xml" />
```

```
32.      <item id="W29MB309" href="29MB309.html" media-type="application/xhtml+xml" />
33.      <item id="W30MB309" href="30MB309.html" media-type="application/xhtml+xml" />
34.      <item id="W31MB309" href="31MB309.html" media-type="application/xhtml+xml" />
35.      <item id="W32MB309" href="32MB309.html" media-type="application/xhtml+xml" />
36.      <item id="W33ZB309" href="33ZB309.html" media-type="application/xhtml+xml" />
37.      <item id="W341B309" href="341B309.html" media-type="application/xhtml+xml" />
38.      <item id="W35MB309" href="35MB309.html" media-type="application/xhtml+xml" />
39.      <item id="W36MB309" href="36MB309.html" media-type="application/xhtml+xml" />
40.      <item id="W37MB309" href="37MB309.html" media-type="application/xhtml+xml" />
41.      <item id="W38MB309" href="38MB309.html" media-type="application/xhtml+xml" />
42.      <item id="W39MB309" href="39MB309.html" media-type="application/xhtml+xml" />
43.      <item id="W40MB309" href="40MB309.html" media-type="application/xhtml+xml" />
44.      <item id="W41ZB309" href="41ZB309.html" media-type="application/xhtml+xml" />
45.      <item id="WTOC" href="TOC.html" media-type="application/xhtml+xml" />
46.    <item id="style" href="style.css" media-type="text/css" />
47.    <item id="cover" href="cover.jpg" media-type="image/jpeg" />
48.  </manifest>
49.   <spine toc="ncx">
50.    <itemref idref="W000Title" />
51.    <itemref idref="W01ZB309" />
52.    <itemref idref="W021B309" />
53.    <itemref idref="W03MB309" />
54.    <itemref idref="W04MB309" />
55.    <itemref idref="W05MB309" />
56.    <itemref idref="W06MB309" />
57.    <itemref idref="W07MB309" />
58.    <itemref idref="W08ZB309" />
59.    <itemref idref="W091B309" />
60.    <itemref idref="W10MB309" />
61.    <itemref idref="W11MB309" />
62.    <itemref idref="W12MB309" />
63.    <itemref idref="W13MB309" />
64.    <itemref idref="W14MB309" />
65.    <itemref idref="W15ZB309" />
66.    <itemref idref="W161B309" />
67.    <itemref idref="W17MB309" />
68.    <itemref idref="W18MB309" />
69.    <itemref idref="W19ZB309" />
70.    <itemref idref="W201B309" />
71.    <itemref idref="W21MB309" />
72.    <itemref idref="W22MB309" />
73.    <itemref idref="W23MB309" />
74.    <itemref idref="W24MB309" />
75.    <itemref idref="W25MB309" />
76.    <itemref idref="W26ZB309" />
77.    <itemref idref="W271B309" />
78.    <itemref idref="W28MB309" />
79.    <itemref idref="W29MB309" />
80.    <itemref idref="W30MB309" />
81.    <itemref idref="W31MB309" />
82.    <itemref idref="W32MB309" />
83.    <itemref idref="W33ZB309" />
84.    <itemref idref="W341B309" />
85.    <itemref idref="W35MB309" />
86.    <itemref idref="W36MB309" />
87.    <itemref idref="W37MB309" />
88.    <itemref idref="W38MB309" />
89.    <itemref idref="W39MB309" />
90.    <itemref idref="W40MB309" />
91.    <itemref idref="W41ZB309" />
92.  </spine>
```

Si os fijáis en la línea 51, tenemos un item con id = W01ZB309. Y en la línea 4, tenemos el archivo al que se está referenciando con ese id, que es **01ZB309.html**. Este archivo se corresponde con la posición 1 de **<spine toc="ncx">**, que no es ni más ni menos que "toc" table of contents o tabla de contenidos. Este archivo es por lo tanto el contenido nº 1 de la tabla de contenidos. El 0 sería **000Title.html** ya que es lo primero que se mostrará del libro, y los sucesivos se sacan de igual forma. En la línea 45 podemos ver otro archivo que define la tabla de contenidos. Es **TOC.html** y

además de los archivos que representan cada contenido, viene el título que se debería mostrar asociado a dicho archivo.

4.1 Lectura Síncrona

Vamos a ver ahora, como acceder a esta información con **EpubLib**, con un test muy sencillo, que podemos situar debajo del anterior. Lo que vamos a leer es el contenido precisamente de nuestro recurso 1 de la tabla de contenidos, es decir, el archivo **01ZB309.html**, que os pongo a continuación:

01ZB309.html

view plain print ?

```
01. <?xml version="1.0" encoding="utf-8" ?>
02. <!DOCTYPE html PUBLIC "-
    //W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
03. <html xmlns="http://www.w3.org/1999/xhtml">
04. <head>
05. <meta http-equiv="Content-
    Type" content="application/xhtml+xml; charset=utf-8" />
06. <meta name="generator" content="Web Books Publishing" />
07. <link rel="stylesheet" type="text/css" href="style.css" />
08. <title>To S.L.O. and the Hesitating Purchaser</title>
09. </head>
10.
11. <body>
12.     <div class='header'>
13.          <h3>To S.L.O. and the Hesitating Purchaser</h3>
14.     </div>
15.
16.
17. <blockquote><p>
18. To S.L.O., an American gentleman in accordance with whose classic taste
19. the following narrative has been designed, it is now, in return for
20. numerous delightful hours, and with the kindest wishes, dedicated by his
21. affectionate friend, the author.
22. </p></blockquote>
23.
24. <p><br /></p>
25.
26. <pre>
27.                TO THE HESITATING PURCHASER
28.
29.                If sailor tales to sailor tunes,
30.                   Storm and adventure, heat and cold,
31.                If schooners, islands, and maroons,
32.                   And buccaneers, and buried gold,
33.                And all the old romance, retold
34.                   Exactly in the ancient way,
35.                Can please, as me they pleased of old,
36.                   The wiser youngsters of today:
37.
38.                —So be it, and fall on!  If not,
39.                   If studious youth no longer crave,
40.                His ancient appetites forgot,
41.                   Kingston, or Ballantyne the brave,
42.                Or Cooper of the wood and wave:
43.                   So be it, also!  And may I
44.                And all my pirates share the grave
45.                   Where these and their creations lie!
46. </pre>
47.
48.
49.
50. </body>
51. </html>
```

TEST

view plain print ?

```
01. @Test
02. public void weCanReadEpub() throws Exception {
03.
04.     final Reader reader = book.getSpine().getResource(1).getReader();
05.     final char[] text1 = new char[1000];
06.     reader.read(text1);
07.
08.     assertTrue(String.valueOf(text1).startsWith("<?
    xml version=\"1.0\" encoding=\"utf-8\" ?>"));
09.     assertTrue(String.valueOf(text1).contains("the following narrative has
10.     assertTrue(String.valueOf(text1).contains("Storm and advent"));
11.
12.     assertFalse(String.valueOf(text1).contains("ure, heat and cold,"));
13.
14.     final char[] text2 = new char[1000];
15.     reader.read(text2);
16.     assertTrue(String.valueOf(text2).startsWith("ure, heat and cold,"));
17.     assertTrue(String.valueOf(text2).contains("</html>"));
18.
19.     reader.close();
20. }
```

En la línea 4 vemos como estamos cogiendo el recurso 1. Leemos en un buffer de tamaño 1000. Esto sería conveniente que fuera una constante, pero para el ambito de este tutorial es suficiente. Cómo podéis observar estamos leyendo los primeros 1000 bytes o caracteres del archivo **01ZB309.html**.

Para comprobar que esto es cierto consideramos que es suficiente ver que empiece como lo hace dicho archivo, que contiene una frase que debería estar en dichos 1000 primeros bytes, y comprobar que contiene la última parte de esos 1000 bytes como se hace en la línea 10. En la línea 12 estamos justamente probando que no contiene parte de lo que debiese estar en los siguientes 1000 bytes, eso forma parte del segundo trozo que queremos leer. Leemos el siguiente trozo y basta con comprobar que empieza justo donde se quedó el anterior y que termina con el cerrado del archivo **html** como se hace en la línea 17. Por último cerramos el reader.

4.2 Lectura Asíncrona

Si lo que pretendemos es leer partes aisladas de un Epub o posicionarnos dentro de un recurso, **EpubLib** cuenta con una forma sencilla de hacerlo. Lo vamos a ver al intentar obtener la segunda parte del archivo "01ZB309.html". Lo que queremos obtener es justamente el contenido de *text2* del test anterior, pero sin tener que leer *text1*. El posicionamiento se hace de la siguiente forma:

TEST

view plain print ?

```
01. @Test
02. public void weCanPositionInDifferentsBytes() throws Exception {
03.
04.     final Reader reader = book.getSpine().getResource(1).getReader();
05.     final char[] text2 = new char[1000];
06.     reader.skip(1000);
07.     reader.read(text2);
08.
09.     assertTrue(String.valueOf(text2).startsWith("ure, heat and cold,"));
10.     assertTrue(String.valueOf(text2).contains("</html>"));
11.
12.     reader.close();
13. }
```

Et voilà!! Es tan sencillo como saltarnos los bytes que no nos interesan con la función **skip**.

5. Conclusiones

Como habéis podido ver **EpubLib** es una librería muy sencilla, que entre otras muchas cosas nos permite leer de una forma fácil el contenido de los Epub. Ahora os toca a vosotros experimentar con ella un poquito más. Un saludo y espero que os haya sido útil.

6. Información sobre el autor

Alberto Barranco Ramón es Ingeniero Técnico en Informática de Gestión y Graduado en Ingeniería del Software por la Universidad Politécnica de Madrid

Mail: abarranco@autentia.com.

Twitter: @barrancoalberto

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

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)

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

Comentarios