COMBINANDO REPORTES EN JAVA

JasperReports

Analizaremos cómo podemos utilizar una de las librerías más conocidas en el lenguaje Java, para poder trabajar de una manera sencilla con los reportes realizados con otras herramientas.

Un punto muy importante de los sistemas se presenta a la hora de mostrar la información resultante de los procesos de las aplicaciones y/o del día a día, principalmente, cuando esto implica tomar decisiones comerciales o gerenciales. Dicha información, por lo general, está almacenada en bases de datos o, en su defecto, en archivos planos.

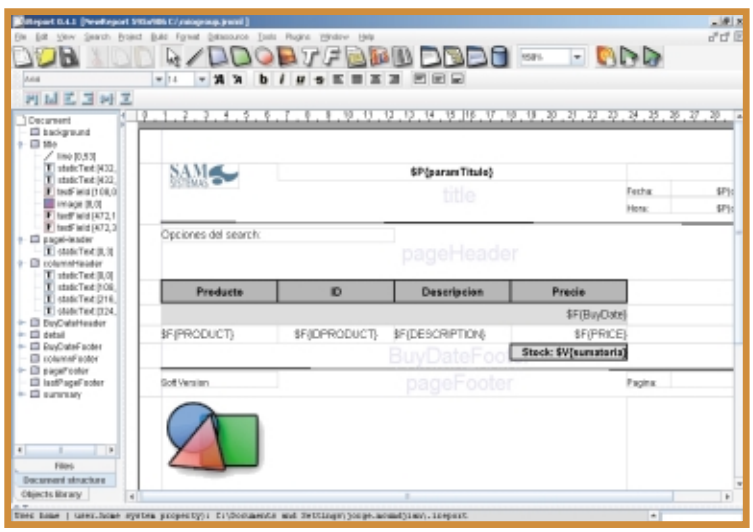
Es posible realizar consultas a las bases de datos, leer estos archivos, diseñar y codificar ventanas o interfaces de usuarios, para interactuar con esa información. Pero este proceso se torna complicado cuando es necesario sumarle la funcionalidad de impresión, la "customización" rápida de la información que estamos mostrando, o bien cuando la aplicación tiene múltiples interfaces, es decir, cuando el usuario puede utilizarla como un cliente Windows o desde un navegador.

A partir de estos puntos, toma importancia la generación de reportes, tanto para obtener dinamismo en las consultas, como para lograr múltiples vistas e, incluso, para facilitar el mantenimiento posterior y su extensibilidad.

Para Java, durante un tiempo, éste fue uno de los puntos más débiles del lenguaje. Hoy en día, existen diversas librerías y herramientas dedicadas (varias de ellas, Open Source) para la rápida generación de reportes. Veremos en esta ocasión la librería **JasperReports**, una de las más conocidas e interesantes, que, combinada con herramientas para el diseño, facilita y agiliza la generación, la previsualización y la impresión de los reportes.

JasperReports es, precisamente, una poderosa herramienta para generar reportes en Java, con la habilidad de producir contenido completo para la pantalla, directo para impresora o en diferentes formatos de archivo (PDF, XLS, CSV y XML, entre otros).

JasperReports es, precisamente, una poderosa herramienta para generar reportes en Java, con la habilidad de producir contenido completo para la pantalla, directo para impresora o en diferentes formatos de archivo.



[Figura 1] **Diseño visual del reporte con IReport.**

La librería es 100% Java, y puede reutilizarse tanto en aplicaciones cliente y cliente/servidor, como en aplicaciones web, J2EE, etc.

JasperReports permite organizar la información obtenida desde una base de datos relacional, a través de conectores JDBC, en diseños de reportes predefinidos en un formato XML.

Son varios los conceptos que deben conocerse, especialmente, su estructura de paquetes. Para mejorar su entendimiento, a lo largo de este artículo presentaremos un ejemplo y describiremos los pasos por seguir.

Conociendo la librería

La estructura de paquetes y las clases más importantes son:

- net.sf.jasperreports.engine.**JasperExportManager**
- net.sf.jasperreports.engine.**JasperPrintManager**
- net.sf.jasperreports.engine.**JasperFillManager**
- net.sf.jasperreports.engine.**JasperCompileManager**
- net.sf.jasperreports.view.**JasperViewer**
- net.sf.jasperreports.view.**JasperDesignViewer**

Estas clases presentan una abstracción para facilitar la vista del reporte, el diseño, la impresión, el llenado, la compilación, etc. Las primeras cuatro permiten manejar el motor de generación del reporte, y las últimas dos (las pertenecientes al paquete view), la visualización de los reportes. A continuación, describiremos su funcionalidad principal.

Armando un reporte

Los reportes se generan basados en un diseño (xml), armado y compilado antes de la puesta en marcha del motor generador del reporte. Una vez lis-

to el diseño, el motor realiza la carga de datos a través de una conexión JDBC a una base de datos relacional, respetando los campos y las consultas predefinidas.

Cumplidos estos pasos, entran en juego las clases que permiten exportar, imprimir o visualizar el reporte, para darle fin a su ciclo de vida.

Vamos por partes

A partir de ahora, nos dedicaremos a ver cuáles son los pasos que se deben seguir para utilizar la librería JasperReports.

Diseñando

El **diseño** representa un '**template**' que va a ser utilizado por el motor para armar el reporte de acuerdo con su estructura, y completado con la información obtenida desde la base de datos.

El documento de diseño está representado por un archivo XML que mantiene la estructura de un archivo DTD definido por el motor de JasperReports, que se obtiene al descargar las librerías.

Para poder darle vida al reporte y completarlo, previamente es necesario compilar el archivo.

La generación de un diseño implica editar un archivo XML siguiendo el formato publicado por:

```
<!DOCTYPE jasperReport PUBLIC
  "-//JasperReports//DTD Report Design//EN"

"http://jasperreports.sourceforge.net/dtds/
  jasperreport.dtd">
```

Habitualmente, estos documentos XML cuentan con una estructura similar a la de cualquier documento de texto, lo cual facilita su entendimiento. Principalmente, se siguen estas secciones:

title	Título del reporte
pageHeader	Encabezado del documento
columnHeader	Encabezado de las columnas
detail	Detalle del documento. Cuerpo
columnFooter	Pie de la columna
pageFooter	Pie del documento
sumary	Cierre del documento

El orden en el que están enunciadas las secciones, comúnmente llamadas '**band**', es el orden en que, por lo general, se declaran dentro de un documento, según su nivel.

Dentro de cada una de las secciones pueden encontrarse, también, '**tags**' (delimitadores), similares a los del lenguaje HTML, para indicar campos de texto (<textfield>), tamaño (<size="">), altura (<height="">), ancho (<width="">), etc.

Incluso, se encuentran las consultas que se realizan a la base de datos para cargar el reporte, indicadas por el delimitador: <query String>, así como también, la referencia a los parámetros, variables y campos del reporte, a través de los delimitadores $P{parámetro}, $V{variable} y $F{campo} respectivamente, etc.

Para entender mejor la estructura de este archivo XML, veamos a continuación un breve ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "-//JasperReports//DTD
    Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/
  jasperreport.dtd">
<jasperReport name="Reporte .CODE" >
  <parameter name="Title" class="java.lang.String"/>
  <queryString><![CDATA[select name, cost from
product]]></queryString>
  <field name="NOMBRE" class="java.lang.String"/>
  <field name="PRODUCTO" class="java.lang.String"/>
  <title>
    <band height="50">
      <textField>
        <reportElement x="0" y="0" width="200"
height="50" />
        <textFieldExpression>$P{Title}
```

Aplicaciones con ANT

http://jasperreports.sourceforge.net
Página oficial de Jasper Reports, con herramientas, documentación, noticias y todo lo referido a esta comunidad.

www.businessobjects.com/products/reporting/crystalreports/java/default.asp
Página de Crystal Reports para Java, con recursos, documentos y hasta una versión trial para bajar y probar este software.

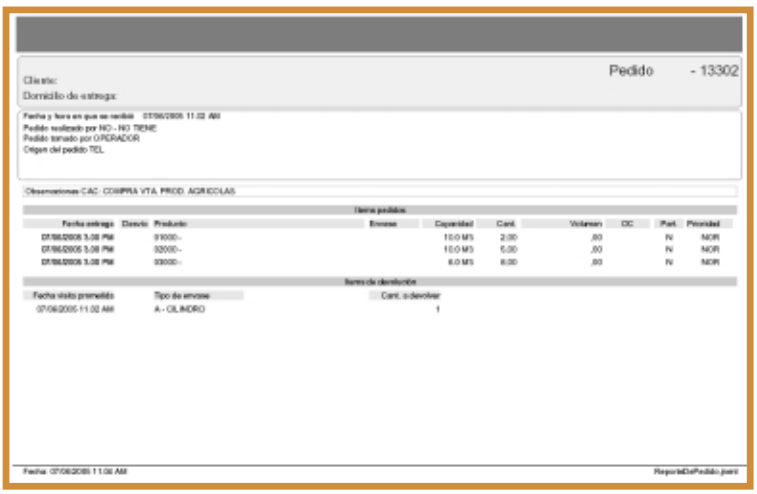
www.jfree.org/jfreereport
Sitio oficial de una alternativa a Jasper Reports, aún en estado beta, pero realmente interesante.

www.ftponline.com/javapro/whitepapers/reporting/crystalreports
White Paper sobre reportes desde Crystal Reports, en Java, de la comunidad JavaPro.

www.reportingengines.com
Sitio con gran cantidad de software para hacer reportes desde Java, con posibilidad de exportar a Excel, etc.

www.java4less.com
Java for Less ofrece una variedad de componentes para Java, entre los cuales se incluye RReport, que cuenta con un diseñador visual.

```
</textFieldExpression>
        </textField>
      </band>
    </title>
    <pageHeader>
          <textField>Encabezado</textField>
    </pageHeader>
    <columnHeader>
      <band height="20">
        <staticText>
          <reportElement x="180" y="0"
             width="180" height="20"/>
          <textElement>
            <font isUnderline="true"/>
          </textElement>
          <text><![CDATA[NAME]]></text>
        </staticText>
...
```



[Figura 2] **Muestra de la publicación de un reporte.**

Una herramienta para tener en cuenta es IReport (**ireport.sourceforge.net**). En la [Figura 1] podremos ver el ambiente de desarrollo.

Compilación

Una vez que el diseño está listo, es necesario compilarlo antes de poder iniciar el proceso de llenado. La compilación se lleva a cabo a través del método **compileReport()** expuesto en la clase **net.sf.jasperreports.engine.JasperManager**.

En este proceso, el diseño es transformado en un objeto serializable del tipo **net.sf.jasperreports.engine.JasperReport**, que luego se guarda en disco. Este archivo (objeto serializado) es utilizado cuando la aplicación completa el reporte con información.

Para previsualizar un diseño, puede utilizarse la clase **net.sf. jasperreports.view.JasperDesignViewer**, a cuyo método **main()** es posible entregarle el archivo xml o el archivo compilado, a fin de visualizarlo.

Generación

Una vez concluida la etapa de diseño del reporte, vamos a darle vida completándolo con el contenido obtenido directamente de la base de datos o de cálculos intermedios.

Así como existen clases que abstraen la complejidad de compilación del reporte, también las hay para la carga de datos del reporte. La clase **net.sf.jasperreports.engine.JasperManager** expone los métodos fillReportXXX(), a través de los cuales nos permitirán realizar la carga de datos del reporte, pasándole como parámetros el **objeto de diseño** (o bien, el archivo que lo representa en formato serializado) y la **conexión JDBC** a la base de datos desde donde se obtendrá la información que necesitemos.

El resultado de este proceso es un objeto que representa un documento listo para ser impreso, un objeto serializable del tipo **net.sf.jasperreports.engine.JasperPrint**. Este objeto puede guardarse en disco para su uso posterior, o bien puede ser impreso, enviado a la pantalla o transformado en PDF, XLS, CSV, etc.

Visualización

Luego de cumplir los pasos anteriores, son varios los caminos que podemos tomar. Por ejemplo, podemos optar por mostrar un reporte por pantalla, imprimirlo o bien transformarlo en algún tipo particular de archivo, como PDF.

- **Mostrar un reporte por pantalla:** utilizaremos la clase **net.sf. jasperreports.view.JasperViewer**, la cual, a través de su método **main()**, recibe el reporte, ya en su etapa final, para mostrar.
- **Imprimir el reporte:** utilizaremos los métodos printReport(), printPage() o printPages(), contenidos en la clase **net.sf. jasperreports.engine.JasperPrintManager**.
- **Exportar los datos a un formato de archivo especial:** podemos utilizar los métodos **exportReportXXX()**, expuestos en la clase **net.sf.jasperreports.engine.JasperExportManager**.

Un ejemplo

Veamos el funcionamiento de este motor realizando un ejemplo de código, el cual nos permitirá entender lo visto.

El siguiente es el código que deberemos utilizar:

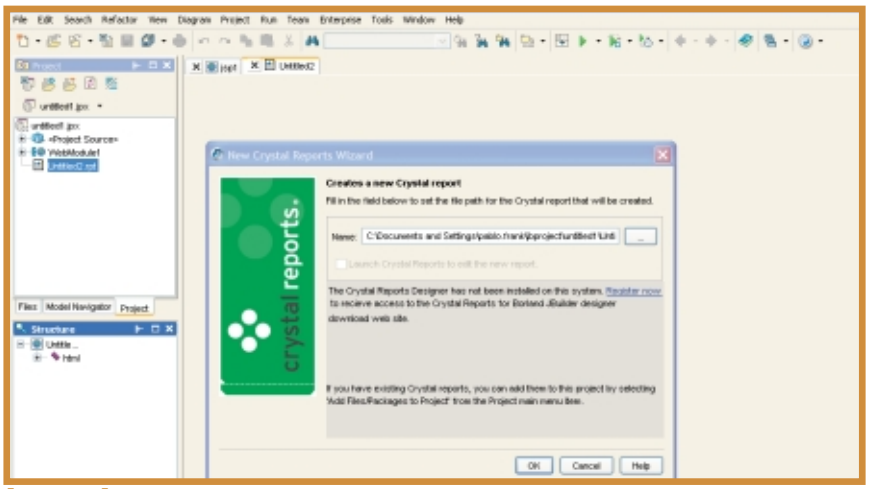
```
Map parametrosReporte;          // Mapa de parámetros del reporte
JasperDesign disenioReporte;    // diseño del reporte
JRDesignQuery queryReporte;     // consulta a la base para
                                   armar el reporte

// Armamos el MAP de parámetros adicionales a pasar al reporte
  (los nombres están determinados por el XML diseñado
parametrosReporte.put("Identificacion", "121");
parametrosReporte.put("NumeroSecuencia", "01");

// Cargamos el diseño desde el archivo xml, la extensión por
  defecto: .jrxml
disenioReporte = JRXmlLoader.load("DisenoXML.jrxml");
```

Algunas herramientas interesantes

Jasper Assistant: www.jasperassistant.com
JasperPal: jasperpal.sourceforge.net
OpenReports Designer: opensourcesoft.net
Sunshine Reports: www.sunshinereports.com



[Figura 3] **Configurando Crystal Reports para Java.**

```
// Armamos y asignamos el query
 queryReporte.setText("Select * from CLIENTES");
 disenioReporte.setQuery(queryReporte);

  // Compilamos el reporte y genera un objeto JasperReport
  JasperReport report = JasperCompileManager.compileReport
     (disenioReporte);

/* Llenamos el reporte, pasando el reporte compilado, los
   parámetros y la conexión a DB y se genera un objeto
   JasperPrint. La información se obtendrá desde la conexión,
   a partir del query indicado */
JasperPrint reporteListo = JasperFillManager.fillReport(report,
 parametrosReporte, DBConnection);

 /* Generamos el objeto de preview del reporte una vez listo*/
 JasperViewer ReporteParaPreview = new JasperViewer(reporteListo,
   false);
 ReporteParaPreview.setTitle("Nuestro Primer Reporte");

 // Mostramos el reporte
 ReporteParaPreview.show();
```

¿Y qué otras opciones existen?

JasperReports no es la única solución para el manejo de reportes en Java; existen otras librerías que posibilitan y facilitan la generación de reportes. Elegimos JasperReports porque es una de las librerías Open Source más completas, pero en la categoría de librerías gratuitas también podemos encontrar otros productos. Algunos de ellos son:

- **Itext:** se utiliza, habitualmente, para generar reportes de formatos planos, en general, sin campos complejos, aunque cuenta con herramientas para mejorarlos. Soporta la posibilidad de ser exportado a PDF. **www.lowagie.com/iText**
- **FOP:** es un reporteador basado en XSL Formatting Objects (XSL – FO). Maneja diversos tipos de formatos, incluso más complejos que Itext. **xml.apache.org/fop**
- **POI:** principalmente utilizado para formatos XLS, tanto para lectura como para escritura. Está basado en Microsoft OLE 2 Compound Document Format. **jakarta.apache.org/poi**

Por el lado de los productos pagos, también existen librerías para el manejo de reportes. Por ejemplo, hay una versión de Crystal Reports for Java, que cumple las funciones de diseñador, más motor generador de reportes.

Incluso, algunas herramientas de desarrollo la traen incorporada en su IDE; una de ellas es, por ejemplo, Borland JBuilder.

Respecto a las librerías Open Source, la mayoría puede agregarse a las herramientas de desarrollo, simplemente, descargando la librería desde el sitio de su fabricante e importando los paquetes .jar, aunque en este caso, seguramente, no se contará con asistentes gráficos.

Para seguir investigando

Para terminar, algunos temas y links importantes para tener en cuenta y utilizar reportes en nuestras aplicaciones:

- El primer paso para comenzar a utilizar las librerías del presente artículo es la descarga del paquete de clases JasperReports completo; no sólo el conjunto de librerías, sino la completa e interesante documentación que incluye y los ejemplos de código.
- Como segundo paso, es útil conocer las características de licenciamiento de JasperReports. Si bien es de código abierto, debemos saber qué podemos hacer y qué no con estas librerías. Todo lo referente a licenciamiento puede consultarse en: **jasperreports.sourceforge.net/license.html**.
- Por último, podemos mencionar que para diseñar reportes y evitar el trabajo manual de generación de archivos XML, es importante conocer diferentes herramientas visuales con el fin de encontrar la que mejor se adapte a cada necesidad o nos permita reducir tiempos en el armado.

Para obtener información adicional, podemos dirigirnos al sitio oficial de JasperReports, **jasperreports.sourceforge.net**.

Sólo queda determinar los reportes que el cliente precisa diseñar, el formato que mejor se adapte a esas necesidades, y reutilizar las librerías para la automatización del proceso de armado. ●

JasperReports no es la única solución para el manejo de reportes en Java; existen otras librerías que posibilitan y facilitan la generación de reportes.