D DOCUMENTO

Programación en Cividas

Informes Jasper y utilidades relacionadas

DIRIGIDO A

Desarrolladores

FECHA

10 de Enero de 2012

Informes Jasper

Inscripción Registro Mercantil de Pontevedra, Tomo 3116, Libro 3116, Folio 30, Hoja PO-38276 – C.I.F.: B-36.499.960

imatia
your innovation partner

Índice

Introducción Informes Jasper

¿Qué es Jasper Reports?

Jasper Reports es una librería para la generación de informes. Está escrita en java y es libre. http://jasperforge.org/ .

El funcionamiento consiste en escribir un xml donde se recogen las particularidades del informe. Este xml lo tratan las clases del Jasper para obtener una salida. Esta salida puede ser un PDF, XML, HTML, CSV, XLS, RTF, TXT.

Ejemplo básico

Para generar un informe con jasper report debemos seguir los siguientes pasos:

1. Generar un fichero .jrxml en el que se configura cómo queremos el informe
2. Compilar el fichero .jrxml para obtener un fichero .jasper
3. Rellenar los datos del informe. Esto generará un fichero .jrprint
4. Exportar el fichero .jrprint al formato que deseemos (pdf, etc). Esto generará el fichero en cuestión.

Generación del fichero .jrxml

El fichero .jrxml se puede generar a mano con este DTD y esta referencia de atributos. De todas formas, una opción mejor es usar la herramienta iReport, que permite generar el fichero .jrxml de forma visual y más fácil.

En este fichero también se configura cual va a ser la fuente de datos (si es una base de datos, un fichero, etc). Incluso si es base de datos, en este fichero se pone el SELECT que devuelve los datos que queremos para el informe.

Compilación del fichero .jrxml

Para compilar el fichero .jrxml y generar el fichero .jasper, desde código podemos poner algo como esto

```
JasperReport report = JasperCompileManager.compileReport("C:\\informes
JAsper\\JRXML\\InformeMySql.jrxml");
```

Rellenado del informe con datos

Ahora hay que rellenar el informe con datos. Desde código se hace con algo como esto

```
JasperPrint print = JasperFillManager.fillReport(report, parameters, conn);
```

donde conn es la conexión con la base de datos. No es necesario indicar el select ni nada similar, puesto que esta información puede estar incluida en el .jrxml .

Obtención del PDF (o tipo deseado)

Finalmente, para obtener el fichero .pdf, la línea es algo como esto

```
JasperExportManager.exportReportToPdfFile(print,
                              "C:\\informes Jasper\\PDF's\\InformePaisesMySQL.pdf");
```

donde el parámetro es el fichero de salida que deseamos.

Informes Jasper en Ontimize

Introducción

Ontimize proporciona un módulo de integración con el motor de informes JasperReports. Este componente englobado dentro de la librería de componentes avanzados (Ontimize More) permite configurar un repositorio para almacenamiento de informes en base de datos y se completa con una sencilla interfaz de acceso (visualización, impresión…) a estos.

Para el diseño de las plantillas que servirán como base para los informes se utilizará la herramienta *open-source* externa *iReport*. Sobre la versión 3.0.0 de este programa, Ontimize proporciona un conjunto de plugins diseñados para falicitar la exportación de informes a los formatos aceptados por el repositorio (zip o jar).

Ontimize Reports

La configuración de este componente no debería suponer ningún problema para el usuario, ya que únicamente se requiere que se sigan los siguientes pasos:

- Servidor:
  - Configuración de la referencia remota de acceso al objeto que accede al repositorio de informes.
  - Creación de la tabla de la base de datos que actúa como repositorio
  - Configuración de la entidad asociada al repositorio
- Cliente:
  - Diseño del formulario que permita subir/descargar/visualizar los informes del repositorio.

Ejemplo básico con IReport

Debido a que esta librería es externa pueden encontrarse multitud de tutoriales y documentación al respecto en internet.

En esta parte se explicará la instalación de los *plugins* que proporciona Ontimize y se continuará con un desarrollo de un informe básico con la versión 3.0 de *iRepotr*. Los pasos son los siguientes:

- Descargar iReport 3.0 (no requiere instalación)
- Familiarizarse con el entorno del programa
- Configurar los *plugins* de Ontimize para *iReport*. Descomprimir este fichero y copiar en *libs* y *plugins* de la carpeta de instalación los archivos correspondientes.
- Arrancar *iReport* y ver que los *plugins* se han instalado correctamente desplegando la opción Conectores del menú viendo si aparecen las opciones: *Utils.Import Report* y *Utils.Export Report*

Creación de un informe

El primer paso consistirá en crear un nuevo documento: *File->New Document* que mostrará las distintas regiones/bandas en las que se podrán disponer los elementos sobre el informe. A continuación se detallará un poco cada una de estas partes:

1. **TITLE** Aparece sólo al inicio del reporte. El título se escribe en esta sección. Ejemplo: "Informe de horas de los empleados"
2. **PAGEHEADER** Aparece en la parte superior de cada página. Puede contener información como la fecha y hora, nombre de la organización, etc.
3. **COLUMNHEADER** Sirve para listar los nombres de los campos que se van a presentar. Por ejemplo: "Nombre del Empleado", "Hora de Entrada", "Hora de Salida", "Horas trabajadas", "Fecha", etc.
4. **DETAIL** En esta sección se despliegan los valores correspondientes a las entradas de campos definidas en la sección anterior. Por ejemplo "Juan Perez", "09:00", "18:00", "9","2005-04-27"
5. **COLUMNFOOTER** Puede presentar información sumarizada para cada uno de los campos. Por ejemplo "Total de Horas Trabajadas: 180"
6. **PAGEFOOTER** Aparece en la parte inferior de cada página. Esta parte puede presentar, el contador de páginas como "Página 1/7".
7. **SUMMARY** Esta sección se usa para proporcionar información sumarizada de los campos presentes en la sección "detail" por ejemplo para el caso de las horas trabajadas de cada empleado se puede definir un objeto gráfico tipo "pie" para tener una mejor comparación y comprensión visual de los datos.

El siguiente paso será definir el origen de datos del que se van a extraer los valores para imprimir en el informe. En este ejemplo se utilizará una tabla de la base de datos, aunque el API de Ontimize Reports pemite que el informe se rellene también desde un *EntityResult*. Un nuevo origen de datos se añade a través de la opción:

Data -> Connections/Data sources con la opción *New*, habilitando una nueva conexión *JDBC*.

Una vez que la conexión ha sido definida y se ha probado que funciona ya se puede definir la consulta que determinará que campos van a estar disponibles para realizar el informe. Esa consulta se define en la opción de *iReport*:

Data->Report Query:

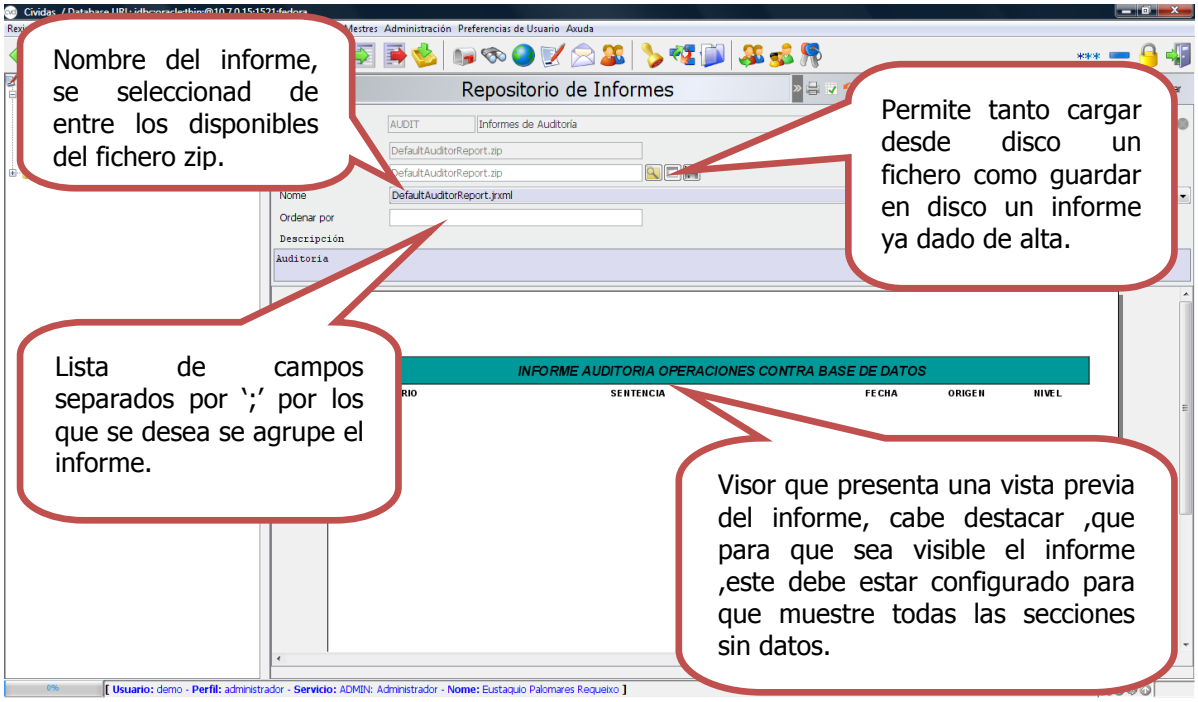
Informes en la plataforma Cividas

La plataforma Cividas utiliza las funcionalidades de Ontimize reports para facilitar la configuración, creación y generación de informes.

Maestro de informes

Una vez tenemos creado el informe con la herramienta IReport, lo exportamos a un zip (que básicamente consiste en encapsular el jrxml dentro de un zip).

Este zip lo podremos dar de alta en el repositorio de informes de la plataforma, para ello se dispone del siguiente maestro.



Generación de un informe

La plataforma consta de una interfaz (Referencia Remota) para facilitar la generación de informes, tanto desde cliente como desde servidor. La interfaz es:

com.cividas.server.reports.ICreatorReports

y consta entre otros de los siguientes métodos:

```
public FileContainer getFilledReportForCividasParameter(String sCividasParameter, String
reportType, Hashtable hparameters, EntityResult erData, boolean attachFilledReport, String
attachEntity, Hashtable attachKeys, boolean signFilledReport,String certificateParameter, int
sesionId) throws Exception;
```

Este método permite rellenar un informe y si se desea firmarlo y/o adjuntarlo. Sus parámetros de entrada son los siguientes:

- **sCividasParameter**: Cividas parameter que indica el nombre del informe que deseamos rellenar.
- **reportType**: tipo de fichero que queremos que devuelva, posibles valores:
  - ICreatorReports.*PDF_TYPE_RETURN* *Fichero pdf*
  - *ICreatorReports.RTF_TYPE_RETURN* *Fichero rtf*
  - ICreatorReports.*TXT_TYPE_RETURN* *Fichero txt*
- **hparameters**: Parámetros (opcionales) que se usarán para rellenar el informe.
- **erData**: Datos con los que rellenar la banda Detail del informe.
- **attachFilledReport**: Indica si queremos adjuntar el informe.
- **attachEntity**: Entidad de adjuntos contra la que trabajar si se decide adjuntar el informe.
- **attachKeys**: Claves usadas para adjuntar el informe si así se requiere.
- **signFilledReport**: Indica si se quiere firmar el informe (firmar por plataforma).
- **certificateParameter**: Nombre del parámetro que indica que certificado de los dados de alta en la plataforma se usará para firmar.

```
public FileContainer getFilledReportForCividasParameter(String sCividasParameter, String
reportType, Hashtable queryKeys, Vector attributes, String entityProviderName, boolean
attachFilledReport, String attachEntity, Hashtable attachKeys,boolean signFilledReport,
String certificateParameter, int sesionId, Connection con) throws Exception;
```

Este método permite rellenar un informe es muy parecido al anterior sólo que en este se le indica la entidad de la que se quieren obtener los datos para rellenar el informe. Sus parámetros más significativos son:

- **QueryKeys**: Indica las claves de consulta para obtener los datos con los que se rellenará el informe.
- **attributes**:Lista de campos que se necesitan para rellenar el informe.
- **entityProviderName:** Indica la entidad contra la que se realizará la consulta usando los dos parámetros anteriores, el resultado se usará para rellenar el informe.

Ejemplos

A continuación se indican dos ejemplos de uso de la interfaz anteriormente indicada ICreatorReports:

Rellenado de un informe desde cliente:

```
//Obtenemos los datos con los que rellenar el informe
if(gettableOriginData().getSelectedRowsNumber()>0 ) {

        Object oData=gettableOriginData().getSelectedRowData();
        if(oData instanceof Hashtable)
                erReport=new EntityResult((Hashtable)oData);
        else
                erReport=(EntityResult)oData;
}

// conseguimos la referencia remota que se encarga de rellanar el iforme
ICreatorReports creatorReport = (ICreatorReports) getLocator().getRemoteReference(
                                ICreatorReport. CREATOR_REPORT_NAME, getRefLocator().getSessionId());

// rellenamos el informe
FileContainer  fileContainerFilled=creatorReport. getFilledReportForCividasParameter(
                                                        sCividasParameter,
                                                        ICreatorReport. PDF_TYPE_RETURN
                                                        new Hashtable(),
                                                        erReport,
                                                        getRefLocator().getSessionId());
```

Rellenado de un informe desde servidor:

```
// Cogemos  la referencia remota que rellena los informes
ICreatorReports creatorReports=(ICreatorReports)((UtilReferenceLocator) locator).getRemoteReference(
                                ICreatorReports.CREATOR_REPORT_NAME, sesionID);

// rellenamos el informe indicándoloe lq entidad proveedora de datos, las claves de
// consulta y los campos a consultar, en este caso no adjuntamos el informe
//resultante, pero si lo firmamos con el certificado marcado por defecto en la //plataforma.
fileContainer=creatorReports.getFilledReportForCividasParameter(sCividasParameter,
                                                        ICreatorReport. PDF_TYPE_RETURN,
                                                        queryKeys,
                                                        attributes,
                                                        entityProviderName,
                                                        false,
                                                        null,
                                                        null,
                                                        true,
                                                        CividasParameter.DEFAULT_CERTIFICATE_CODE,
                                                        sesionId,
                                                        con);
```