

PHP-JRU

PHP JasperReport Utils

Autor: Robert Bruno

Versión 1.0

PHP JapserReport Utils

Robert Bruno

Versión 1.0

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Copy of GNU Lesser General Public License at:
<http://www.gnu.org/copyleft/lesser.txt>

Contact author at: robert.alexander.bruno@gmail.com

Índice de contenido

INTRODUCCION.....	4
Sobre este manual.....	4
¿Que es PHP-JRU?.....	4
CONFIGURANDO PHP-JRU.....	5
Requisitos.....	5
Java/Bridge.....	5
¿Porque usar Java/Bridge?.....	6
Instalar en GNU/Linux o *nix.....	6
Instalar Java Bridge en Windows.....	8
Librerías y archivos jar	15
Jasper Report.....	15
Conectores JDBC.....	16
IREPORT.....	18
Descargar iReport.....	18
USANDO PHP-JRU.....	19
Cargar librerías.....	21
Conexiones jdbc.....	21
ReportManager.....	22
Atributos de ReportManager.....	27
Métodos de ReportManager.....	27

INTRODUCCION

Sobre este manual

La intención de este manual es poder dar una orientación en el uso de PHP-JRU y dar un conocimiento básico de las herramientas que se necesitan para generar reportes en PHP usando iReport.

Es necesario saber que el contenido de este manual puede perder su validez con el tiempo, debido a los cambios en el desarrollo y expectativas a futuro de PHP-JRU.

Aunque se explica ciertos conceptos básicos sobre las herramientas de Jasper Report se da por entendido que el usuario de PHP-JRU ya ha usado con anterioridad y de manera exitosa iReport.

Deseo dar un agradecimiento a Hiber Tadeo Moreno Tovilla, Licenciado en Informática Administrativa en la ciudad de Chiapas, México. Quien realizó los test de instalación y configuración de Java/Bridge en Windows y nos dejó un manual del cual se extrae la información para la realización de este manual.

¿Que es PHP-JRU?

PHP-JRU (PHP Jasper Report Utils) es una librería pensada para generar reportes diseñados bajo la herramienta iReport desde una simple aplicación escrita en PHP.

Para ello esta librería se comunica con la JVM a través de Java/Bridge, y de esta manera poder cargar las librerías provista por JasperReport he indicarle las tareas que deben realizarse para generar los reportes tal cual como si se estuviera haciendo desde una aplicación escrita en java. En esta versión JRU provee de una serie de rutinas para hacer esto una tarea muy fácil.

CONFIGURANDO PHP-JRU

Requisitos

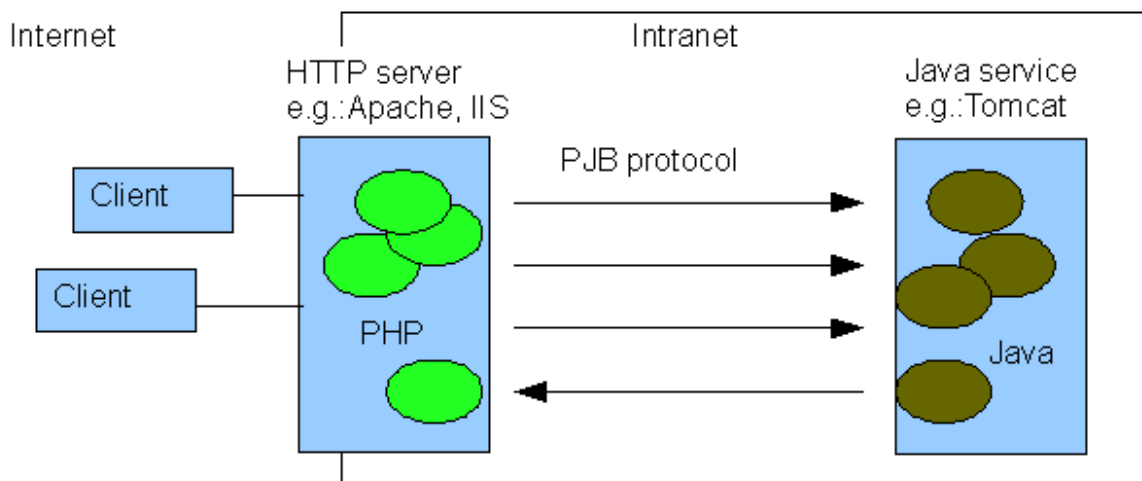
PHP-JRU se comunica con un servidor de aplicaciones Java (mediante Java/Bridge) accede a los recursos y Librerías JasperReport para poder generar un reporte. Por lo que será necesario explicar como enlazar estos elementos.

Java/Bridge

PHP/Java Bridge es una implementación streaming de un protocolo de red basado en XML, que puede ser utilizado para conectar un motor de secuencia de comando nativo, por ejemplo PHP o Python, con una máquina virtual Java. También permite el acceso a los scripts PHP dentro de clases Java.

Es hasta 50 veces más rápido que usar RPC (Remote Procedure Call, requiere menos recursos en el lado servidor web. Es más rápido y más fiable que la comunicación directa a través de la interfaz nativa de Java, y no requiere de componentes adicionales para invocar procedimientos de Java.

La gráfica siguiente describe a groso modo en que consiste este proceso



¿Porque usar Java/Bridge?

Debido a esta interoperabilidad que ofrece Java/Bridge, podemos desarrollar páginas web interactivas utilizando PHP, pero delegar funciones en clases Java, lo que permite usar cada tecnología por sus puntos fuertes y eludir sus puntos débiles

También es cierto, que por los momentos es la tecnología que nos permite que JRU funcione. Dentro de las expectativas futuras se pretende dar soporte nativo para que desde PHP se pueda leer el diseño del reporte y generarlo de forma directa sin la mediación de Java/Bridge.

Instalar en GNU/Linux o *nix

Junto con PHP-JRU viene un archivo llamado PHPJRU.war en la carpeta libs. Este archivo es JAVA/Bridge acoplado con las librerías JasperReport y un conjunto de conectores JDBC, dicho archivo debe instalarse en un servidor de aplicación JEE o un contenedor de servlets, lo mas recomendable es usar Apache Tomcat 6+.

En un sistema operativo basado en GNU/Linux o cualquier *nix, debe instalar el servidor de aplicación Tomcat6 (recomendado) y las dependencia y librerías necesarias (Debe tener instalado el JDK de Java). La primera linea es un ejemplo de como sería en **Fedora** y la segunda en **Debian**. En lo sucesivo a través de este manual así serán las descripción de comando en la consola

```
$ sudo yum install tomcat6 tomcat6-admin-webapps
```

```
# apt-get install tomcat6 tomcat6-admin-webapps
```

Nota: Es recomendable tener instalado php-cgi además de las fuentes que se vayan a usar en el diseño de los reportes, por ejemplo: Arial, Time New Romans.

```
$ sudo service tomcat6 restartomcat6
```

```
# /etc/init.d/tomcat6 start
```

Luego de estar configurado tomcat (compruebe ingresando desde un navegador web a <http://localhost:8080>).

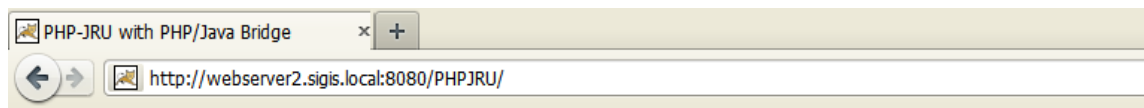
Descargue PHP-JRU de la siguiente URL:

<https://sourceforge.net/projects/php-jru/>

Descomprima el archivo tar.gz y copie el PHPJRU.war que esta en la carpeta libs en el directorio de trabajo de tomcat y reiniciar el servidor.

```
# cp php-jru/libs/PHPJRU.war /var/lib/tomcat6/appweb
```

Compruebe al acceder a la URL <http://localhost:8080/PHPJRU/> debe tener una pagina como la siguiente:



PHP-JRU with PHP/Java Bridge

Bienvenido

Ya configuró y activo PHP/Java Bridge junto con las librerías JasperReport.

Ahora incluya PHP-JRU en su script de php:

```
require_once("../PHP-JRU/PHP-JRU.php");
```

Luego cambie la directiva allow_url_include en archivo de configuración php.ini y active o coloque su valor en On.

Instalar Java Bridge en Windows

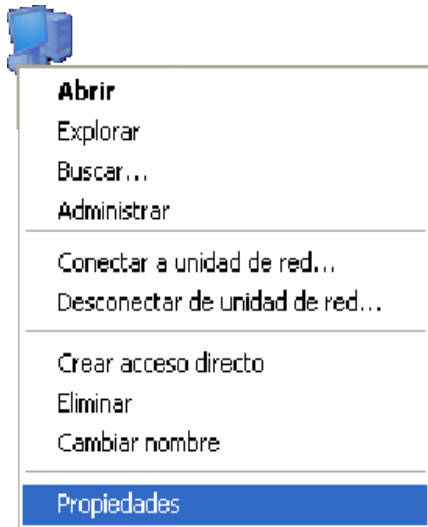
Instalamos Java Jdk SE, para este manual se usa la versión 6. Puede descargarla desde el enlace siguiente:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

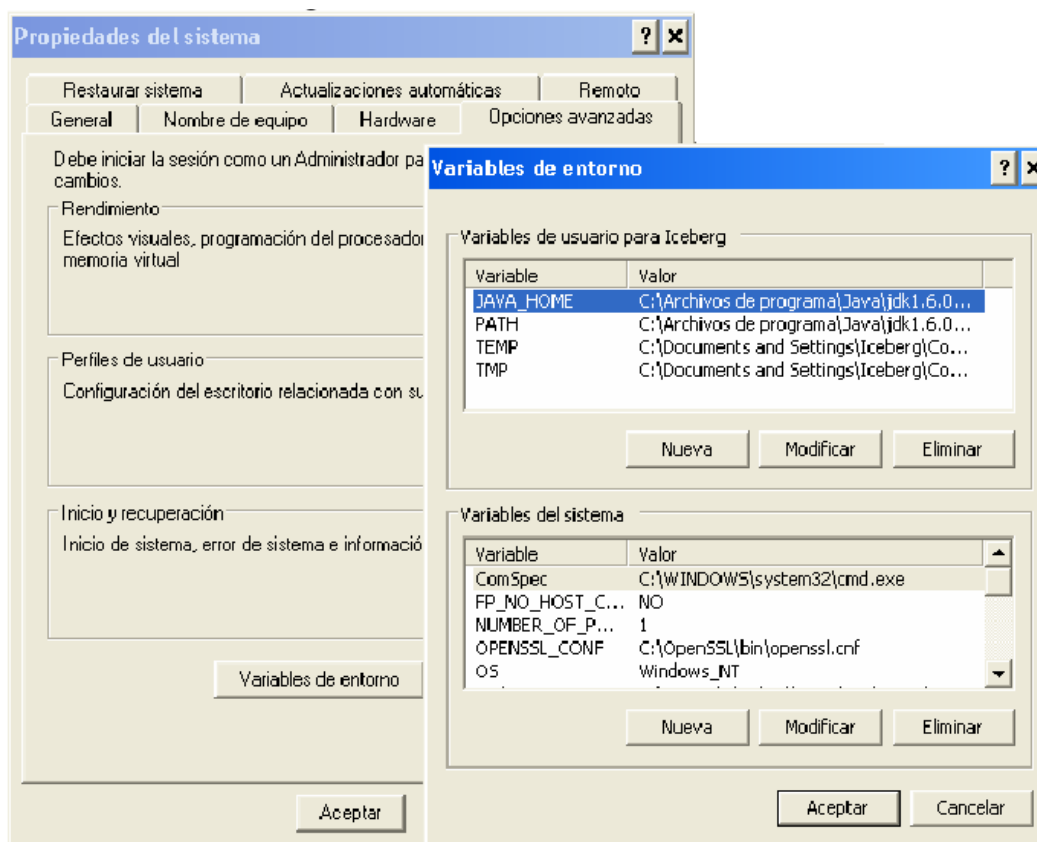
observará una pagina parecida a la siguiente, elija la primera opción:



A continuación se deben configurar las variables de entorno: Hacemos Clic con el botón derecho del mouse sobre MiPc y seleccionamos la opción de Propiedades:

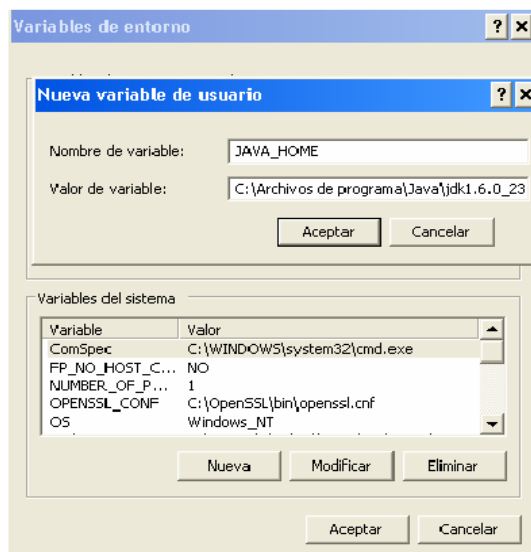


Seleccionamos la pestaña Opciones Avanzadas , y después damos clic en el botón: Variable de entorno



En la imagen están configuradas las variables de entorno para Java, pero esto se hace con los siguientes pasos:

Clic en Nueva y poner la ruta donde esta instalado Java jdk en su equipo, ejemplo: C:\Archivos de programa\Java\jdk1.6.0_23

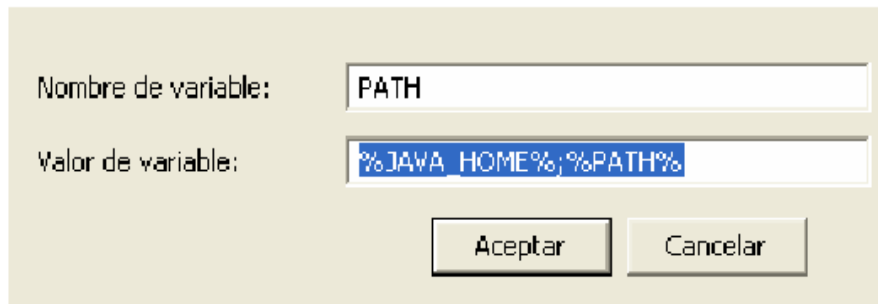


Escribimos en el cuadro de dialogo JAVA_HOME y en la parte de abajo la ruta antes mencionada y damos clic en aceptar:

Agregaremos también el PATH: De la misma manera damos clic nuevamente en Nueva. Y agregamos lo siguiente:

Nombre de Variable: PATH

Valor de Variable: %JAVA_HOME%;%PATH%




Por último damos clic en aceptar y nuevamente aceptar, con esto ya tenemos las variables de entorno Java para Windows.

Configuración de Apache Tomcat

Descargue e instale Xampp con tomcat6 o superior, sin embargo, si lo prefiere puede configurar cada servidor individualmente. En este punto puede ser cualquier otro como WampServer, etc. Se recomienda, por considerarlo mas completo, pero sobre todo porque esta distribución tiene Tomcat 7.0 integrada.

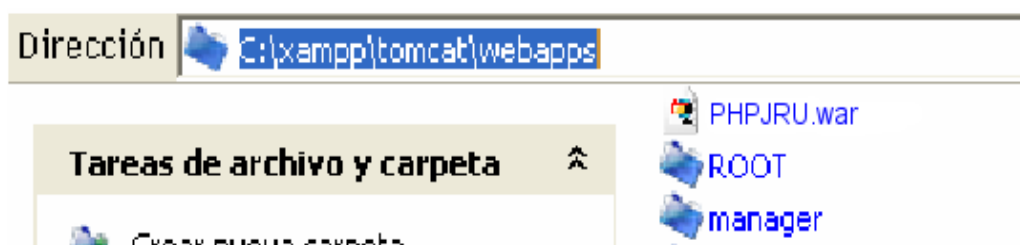
<http://www.apachefriends.org/en/xampp-windows.html#641>

XAMPP for Windows 1.7.4, 26.1.2011		
Version	Size	Content
XAMPP Windows 1.7.4		Apache 2.2.17, MySQL 5.5.8 + PBXT engine (currently disabled), PHP 5.3.5, OpenSSL 0.9.8l, phpMyAdmin 3.3.9, XAMPP Control Panel 2.5.8, Webalizer 2.21-02, Mercury Mail Transport System v4.72, FileZilla FTP Server 0.9.37, SQLite 2.8.17, SQLite 3.6.20, ADOdb 5.11, Xdebug 2.1.0rc1, Tomcat 7.0.3 (with mod_proxy_ajp as connector) For Windows 2000, XP, Vista, 7. See also README
 Installer	66 MB	Installer

Descargue php-jru de la siguiente url:

<https://sourceforge.net/projects/php-jru/>

Dentro de la Carpeta webapps de Tomcat, simplemente colocar el archivo PHPJRU.war que esta en la carpeta libs de JRU, esto es con el fin de que cuando levante el servidor Tomcat, automáticamente reconozca el paquete y descomprima las fuentes.



Abrir el archivo C:\xampp\tomcat\conf y editamos tomcat-users.xml, entre las llaves <tomcat-users> agregamos las siguientes líneas, que son para el acceso al panel de control de Tomcat

```
<role rolename="admin"/>
<role rolename="manager"/>
<user username="tomcat" password="tomcat" roles="admin,manager"/>
```

```
<tomcat-users>
<!--
NOTE: By default, no user is included in the "manager-gui" role
to operate the "/manager/html" web application. If you wish to use
you must define such a user - the username and password are arbitrary
-->
<!--
NOTE: The sample user and role entries below are wrapped in a comment
and thus are ignored when reading this file. Do not forget to remove
<!-- ... --> that surrounds them.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
-->
<role rolename="admin"/>
<role rolename="manager"/>
<user username="tomcat" password="tomcat" roles="admin,manager"/>
</tomcat-users>
```

```
C:\>cd xampp
C:\xampp>cd tomcat
C:\xampp\tomcat>
```

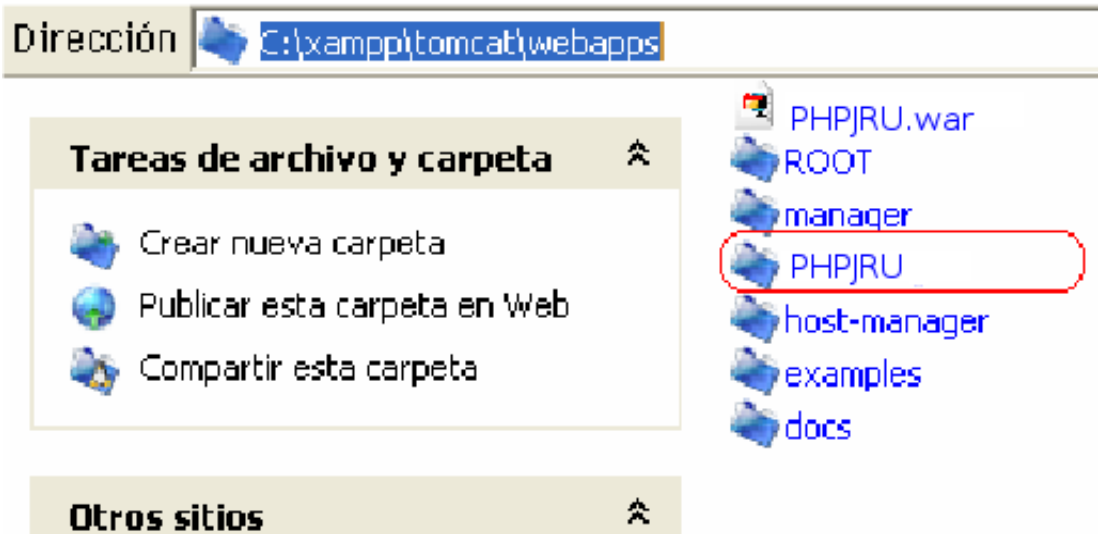
Levantamos el Servicio Tomcat, tecleando: catalina_start.bat

```
C:\xampp\tomcat>catalina_start.bat
```

Con esto se inicia Tomcat y se mostraría algo como lo siguiente:

```
C:\WINDOWS\system32\cmd.exe - catalina_start.bat
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.
java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAcces
sorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at org.apache.catalina.startup.Bootstrap.load(Bootstrap.java:262)
at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:430)
4/02/2011 11:11:56 PM org.apache.coyote.http11.Http11Protocol init
INFO: Inicializando Coyote HTTP/1.1 en puerto http-8080
4/02/2011 11:11:56 PM org.apache.coyote.ajp.AjpProtocol init
INFO: Inicializando Coyote AJP/1.3 en ajp-8009
4/02/2011 11:11:56 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 7053 ms
4/02/2011 11:11:56 PM org.apache.catalina.core.StandardService startInternal
INFO: Arrancando servicio Catalina
4/02/2011 11:11:56 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.5
4/02/2011 11:11:56 PM org.apache.catalina.startup.HostConfig deployWAR
INFO: Despliegue del archivo JavaBridge.war de la aplicaci3n web
4/02/2011 11:12:01 PM org.apache.catalina.loader.WebappClassLoader validateJarFi
le
INFO: validateJarFile(C:\xampp\tomcat\webapps\JavaBridge\WEB-INF\lib\servlet.jar
) - jar not loaded. See Servlet Spec 2.3, section 9.7.2. Offending class: javax/
servlet/Servlet.class
```

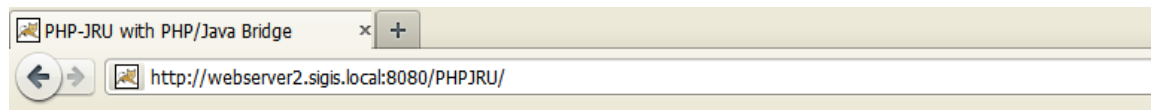
Una vez levantado el servicio Tomcat, verificar en la carpeta webapps, si PHPJRU.war esta descomprimido.



Luego accedemos desde nuestro explorador web a:

<http://localhost:8080/PHPJRU>

Y debemos ver una pagina como la siguiente:



PHP-JRU with PHP/Java Bridge

Bienvenido

Ya configuró y activo PHP/Java Bridge junto con las librerías JasperReport.

Ahora incluya PHP-JRU en su script de php:

```
require_once("../PHP-JRU/PHP-JRU.php");
```

Para poder acceder a las funciones de Java desde php deberá incluir un archivo .inc que se encuentra en la aplicación que acabamos de montar en el Tomcat, por lo que debemos activar o cambiar la directiva `allow_url_include` en el archivo de configuración de php, el `php.ini`

```
#####  
; Fopen wrappers ;  
#####  
  
; Whether to allow the treatment  
; http://php.net/allow-url-fopen  
allow_url_fopen = On  
  
; Whether to allow include/requi  
; http://php.net/allow-url-inclu  
allow_url_include = On
```

Librerías y archivos jar

Como ya se ha mencionado anteriormente es necesario el uso de JasperReport así como también el conjuntos de jar que se requieran según la estructura y elementos del archivo xml que contiene el diseño, por ejemplo si los datos se obtendrán de MySQL sera necesario tener el driver jdbc para realizar la conexión, o si se requiere colocar una imagen con formato png sera necesario cargar el controlador de imágenes png.

En el archivo PHPJRU.war contiene ya una serie de jars, dentro de los que se encuentras: las librerías jasperReport con todas sus dependencias y algunos controladores Jdbc (Jtds, PostgresSql, Mysql, msqlite) por lo que usted solo cargará librerías adicionales que requiera para alguna funcionalidad.

Para ver los detalles de la librerías que posee PHPJRU.war acceda a:

<http://localhost:8080/PHPJRU>

Jasper Report

JasperReports es una poderosa herramienta open source que tiene la capacidad para generar contenido enriquecido en la pantalla, a la impresora, o en PDF, HTML, XLS, RTF, ODT, CSV, TXT y archivos XML. Está escrito completamente en Java y puede ser utilizado en una gran variedad de aplicaciones para generar contenido dinámico.

El diseño de un reporte representa una plantilla que es utilizada por el motor de JasperReports para ofrecer contenido dinámico que puede ser mostrado en la Web. Los datos almacenados en la base de datos serán organizados de acuerdo con el diseño del reporte que esta definido en archivos JRXML y debe tener una estructura especial. Esta estructura se declara en un archivo DTD suministrado con el motor de JasperReports. Los archivos JRXML son compilados, con el fin de utilizar en el reporte las operaciones de llenado.

Para tener una mejor comprensión de la estructura de un archivo JRXML, es

recomendable utilizar la referencia rápida de JasperReport. Sin embargo no es necesario conocer con mayor detalle la estructura del mismo ya que usaremos iReport para generar los diseños.

Conectores JDBC

Java Database Connectivity, más conocida por sus siglas **JDBC**, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

Cada fabricante de RDBMS es el responsable de ofrecer el conector JDBC, por ejemplo, si se va a utilizar postgres el conector para realizar la conexión con la base de datos debe descargarse de la pagina web de postgres.

Sin embargo para nuestro caso no es necesario conocer a fondo estos conceptos ni descargarse un conector JDBC, al menos que este no este mencionado en el cuadro de abajo. PHP-JRU provee de una serie de adaptadores que existen para cada conector jdbc que viene en PHPJRU.war, la clase PJRUConexion es la encargada de crear las conexiones a la base de datos, los adaptadores existentes son:

Adaptador	JDBC Driver o DataSource
mysql	com.mysql.jdbc.Driver
pgsql	org.postgresql.Driver
xml	net.sf.jasperreports.engine.data.JRXmlDataSource
mssql	net.sourceforge.jtds.jdbc.Driver
excel	net.sf.jasperreports.engine.data.JRxlsDataSource

PJRUConexion. Esta clase es estática y sus atributos y métodos también lo son. Estos atributos estáticos definidos como constantes nos permiten indicar los datos de la conexión en caso de que todos los reportes deban usar la misma fuente de datos. Dicha clase la ubicamos en el archivo PJRUConexion.php y allí podremos indicar los datos de la conexión de la siguiente forma:

```
<?php
    const TYPE      = 'pgsql';
    const HOST      = 'localhost';
    const PORT      = "";
    const USER      = 'usuario';
    const PASSWORD  = 'clave';
    const DATABASE  = 'nombre_base_de_datos';
?>
```

En caso de que desee indicar una conexión solo para un tipo de reporte puede hacerlo de la siguiente forma:

```
<?php
    requiere_one('../php-jru/php-jru.php');

    $conn = PJRUConexion::get('mysql','localhost','3416','basedatos','user','password');
?>
```

A diferencia de las versiones anteriores de PJRU ya no es necesario interactuar directamente con la clase JDBCConexion ni con controladores jdbc, esta tarea se delega directamente a la clase PJRUConexion, con el fin de evitar el uso de código java.

Todos los métodos que se usaban en versiones anteriores para el acceso a librerías y conectores JDBC **quedan en completo desuso**, y se listan a continuación:

setLibrayPath, getLibraryPath, isLibraryLoad, loadLibrary

IREPORT

iReport es un aplicación que ayuda en el diseño visual de reportes a los usuarios y desarrolladores que utilizan las librerías JasperReports.

A través de una interfaz de usuario rica y muy fácil de usar , iReport proporciona todas las funciones más importantes para crear reportes agradables en poco tiempo. iReport puede ayudar a las personas que no conocen las librerías JasperReports a crear reportes complejos sin conocer la sintaxis XML.

iReport está escrito en Java y a partir de la version 0.2.0 fue totalmente reescrito. Por esta razón, hay dos manuales de iReport.

La siguiente lista describe algunas de las características mas importantes de iReport:

- Soporta 98% de las etiquetas de JasperReports.
- Diseñador Visual (wysiwyg) con herramientas para dibujar rectángulos, líneas, elipses, campos de texto, gráficos, subreports ...
- Incorpora editor con sintaxis heighlighting para escribir expresiones.
- Soporte para Unicode y idiomas no latín (ruso, chino, coreano ,...).
- Integra un compilador.
- Soporta todas las bases de datos compatible con JDBC
- Apoyo de todo tipo de JRDataSource
- Asistente para crear reportes automáticamente.
- Soporta subreports.
- Guardar copias de seguridad.
- Soporte plantillas.

Descargar iReport

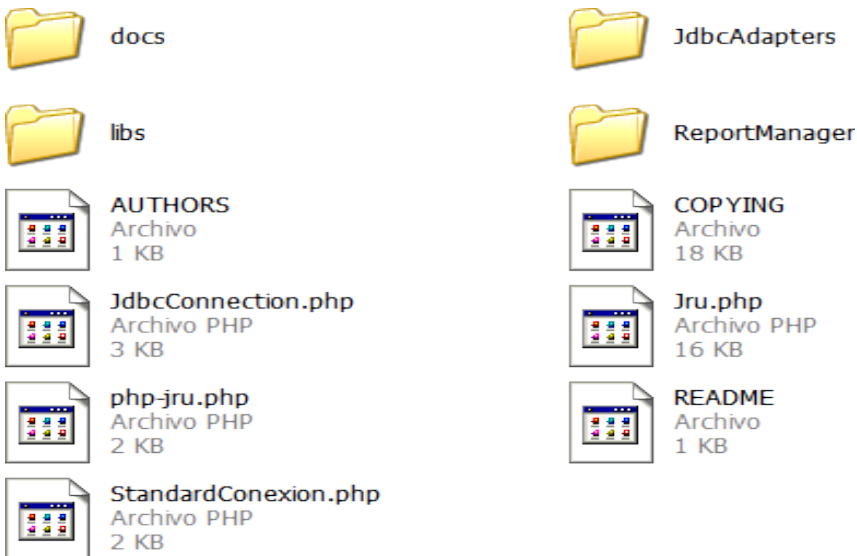
iReport se puede descargar de la pagina de SourceForge en la siguiente URL

http://sourceforge.net/project/showfiles.php?group_id=64348

USANDO PHP-JRU

El concepto y forma de uso de PJRU fue completamente rediseñado a partir de esta versión. El objetivo fue lograr ejecutar un reporte de forma mas dinámica y minimizar el uso de lenguaje java dentro de los script PHP. Sin embargo se pueden seguir generando reportes de la misma forma que se hacían anteriormente, con la excepción de los casos donde se usen la funciones obsoletas antes mencionadas, ya que las cuales no existen en esta versión.

La librería esta organizada de la siguiente forma:



- **docs** Documentación (Manuales, ejemplo, etc ...)
- **JdbcAdapters** Adaptadores para las conexión jdbc.
- **libs** es aquí donde esta las PHPJRU.war. Este archivo es Java/Bridge con las libreas JasperReport y conectores JDBC. Este lo debemos copiar en la carpeta de trabajo de Tomcat, para acceder mediante <http://localhost:8080/PHPJRU>
- **ReportManager** Es aquí donde se encuentran el conjunto de clases que nos facilitan la generación de reportes, enlazando un script PHP con el diseño en .jrxml. Permitiendo que el script PHP sea el que gestione los parámetros y sentencia sql e incluso la conexión jdbc con la que se genera el reporte.

- **AUTHORS, COPYING y README**

contienen autores, licencia y ayuda respectivamente.

- **JdbcConnection.php** contiene la clase JdbcConnection la cual se encarga de crear conexiones jdbc a la base de datos para crear el origen de los datos. Aunque puede seguir usándola de igual forma que en las versiones anteriores, esto no es recomendable, ya que las conexiones JDBC se manejaran siempre mediante la clase PJRUConexion y es sumamente recomendable que se cumpla de esta forma.

- **PJRU.php** Es la clase principal para la generación de reportes, llamada anteriormente JRU. La misma provee de un conjunto de métodos para invocar a las librerías JasperReport y poder interpretar el diseño, compilarlo y luego exportarlo a algún formato. En esta versión se le agrego métodos para la generación de formatos RTF. También se cambio en comportamiento del resto de los métodos para que después de compilar un reporte borre todos aquellos archivos creados en el proceso, como por ejemplo los .jasper, exceptuando que el diseño no sea un jrxml si no un .jasper.

- **PHP-JRU.php** Con este archivo se carga la librería y sus dependencias. Se declaran constantes y se incluye el archivo para el acceso a Java, usando la siguiente instrucción.

```
require_one ('http://localhost:8080/PHPJRU/java/Java.inc');
```

Es aquí donde puede cambiar dicha url para adaptarla a su entorno, o donde se encuentre el Tomcat con las librerías Java/Bridge (PHPJRU.war).

Es además el archivo que debe incluir dentro de su aplicación de la siguiente manera según la ruta en donde este PHP-JRU:

```
requiere_one('../php-jru/php-jru.php');
```

- **PJRConexion.php** En ella se declarada la clase estática que nos permitirá definir las conexión a la base de datos. Cuando se llama al método get de esta clase, el mismo hace un llamado al adaptador indicado en el parámetro type para que este cree una conexión JDBC.

Cargar librerías

Como ya se ha indicado anteriormente, todas las librerías o archivos .jar necesarios para la generación de reportes deben cargarse en archivo PHPJRU.war. Sin embargo este ya viene con un conjunto de librería y conectores jdbc por lo que usted solo tendrá que hacer esto en casos específicos.

Para saber cuales específicamente son las librerías que tiene PHPJRU acceda desde un explorador web a la dirección:

<http://localhost:8080/PHPJRU>

Conexiones jdbc.

Como ya se ha comentado anteriormente toda conexión se manejará con la clase JRUConexion, el siguiente es un ejemplo:

```
/**
 * incluye PJRU
 * */
require ('../php-jru/php-jru.php');

$conn = JRUConexion::get('pgsql','localhost','basededatos','user','password');
```

Mas adelante se dará cuenta que en la mayor parte de los casos no tendrá que realizar esto, si no que puede delegar esto a JRUConexion si todos los reportes se conectan a la misma fuente de datos.

Al cambiar los datos de la clase PJRUConexion estos se tomaran automáticamente para crear un conexión jdbc al momento de generar los reportes. También puede indicar los datos de la conexión haciendo uso de una URL, en la que la sección de protocolo define el tipo de conexión a realizar:

```
PJRUConexion::get("pgsql://user:pass@host:port/database");
```

ReportManager

Esta es la clase que nos permite crear los reportes y de ahora en adelante **Administrador de Reportes**. Esta diseñada para cargar un archivo jrxml haciendo uso de un script php que usted mismo creará, dicho script funcionará como un plugin o una extensión, que indicará cuales son los parámetros, la sentencia sql y la conexión jdbc para compilar el archivo jrxml. Así como también se manejará métodos callback, para ejecutar acciones antes de que se compile un reporte o después de que este se genere.

Desde ahora en adelante llamaremos a este script **Extensión de Reporte**. La definición del mismo se hace a través de una clase que hereda de ReportExtension. también implementa la interfaz ReportExtensionInterface y su estructura es la siguiente:

```
<?php
class NombreReportExtension extends ReportExtension
{
    public $reportFileName;

    public $alias;

    public $enabled = true;

    public function getParam(){}

    public function getSqlSentence(){}

    public function getHtmlOptions(){}

    public function beforeRun(){}

    public function afterRun($outfilename){}

    public function getConexion(){}
}
?>
```

Esta es la estructura que debe tener cada extensión de reporte. Cada uno de los métodos le dará información a ReportManager para la compilación del archivo jrxml, por lo que estará sumamente ligado al mismo.

Es necesario que la extensión herede de ReportExtension de lo contrario

simplemente no es una extensión de reporte. El nombre de la clase debe cumplir la siguiente norma: **MyextensionReportExtension**

Donde Myextension es el nombre que usted desee darle a la extensión y tiene que empezar con la primera letra en mayúscula, por lo general debería ser el mismo o estar asociado al archivo jrxml. Ejemplo si el jrxml se llama resumen (resumen.jrxml) la extensión de reporte tiene que llamarse ResumenReportExtension y el script php debe cumplir con la misma norma excepto por la primera letra que puede ser en minúscula. De la siguiente forma: resumenReportExtension.php y la clase ResumenExtensionReport.

A continuación se describen los atributos que deben especificar en una extensión de reporte

- **\$reportFileName** este indica el nombre del diseño del reporte, es decir, el nombre del archivo .jrxml. Solo se debe especificar el nombre mas no la extensión, ya que ReportManager buscará automáticamente un .jrxml si se indica la sentencia sql en la extensión de reporte o un .jasper si no se indica. El jrxml se busca en el mismo directorio en el que se encuentre la extensión.
- **\$alias** Es una breve descripción o el nombre que desee mostrar al usuario final. Esto es solo para el uso dentro de la aplicación en la que usted este trabajando.
- **\$enabled** Establece si esta o no habilitada la extensión. Al igual que el parámetro anterior este es solo para el uso de usted en su aplicación. Su valor es de tipo boolean.

De igual forma en cada extensión deben definirse una serie de métodos que indicaran a ReportManager los datos que necesita para compilar y exportar el reporte. Si no se va a retornar nada en alguno de los métodos igual debe definirlos.

Estos métodos serán llamados por ReportManager y desde ellos usted puede

acceder a los valores enviados por el cliente web, a través de post o get. A continuación se describen cada uno de ellos:

- **getParam** En este método debe definir los parámetros del reporte y retornar dichos valores para que reportManager los indique al momento de compilar el reporte. Ejemplo

```
public function getParam()
{
    $parameters = new java ('java.util.HashMap');

    $parameters->put('HEADER_REPORT', 'Inversiones Pedro <br> Área de ventas ');

    $parameters->put('TITULO', "Listado de Productos");

    $parameters->put('REPORT_LOCALE', new Java('java.util.Locale','es', 'VE'));

    return $parameters;
}
```

Este es el único punto en el que usted tendrá que acudir a código Java, ya que por los momentos es la única forma de pasar parámetros al reporte.

- **getSqlSentence** Se indica la sentencia sql que se usará en el reporte. Si no se retorna nada ReportManager buscará un .jasper en vez de un .jrxml, Ejemplo:

```
public function getSqlSentence ()
{
    return 'select * from productos where precio > 1000';
}
```

- **getHtmlOptions** En este caso este es solo un mecanismo que se da a usted para que indique código html asociado al reporte, es solo para su uso dentro de la aplicación en la que este trabajando, ya que en ningún momento ReportManager ejecuta este método. Simplemente retorne un string con código html si desea implementarlo en su aplicación para algún fin.
- **beforeRun** Se ejecuta antes de compilar el reporte y le permite a

usted ejecutar algunas acciones y detener el proceso de compilación al retornar false. Ejemplo:

```
public function beforeRun()
{
    if($_REQUEST['crear_relacion'] == 'on')
    {
        $this->reporte = new Archivos();

        $this->reporte->begin();

        $this->reporte->descripcion = $_REQUEST['titulo'];

        $this->reporte->users_id = Session::get('id');

        $this->reporte->fecha = date('Y-m-d H:i:s');

        $this->reporte->create();
    }else
    {
        return false;
    }
}
```

En este ejemplo se puede apreciar que se evalúa una variable enviada desde el cliente y se realiza una acción en caso de cumplirse de lo contrario se cancela el proceso de generación del reporte al retornar false.

- **afterRun:** Este método es ejecutado por ReportManager luego de que se genera el reporte, y se le envía por parámetro el nombre y ruta completa del archivo generado (pdf, html, odt, Xls o rtf). Lo que este método retorne no le da ningún comportamiento a ReportManager ya que el mismo termina su ejecución luego de llamar a este método. A continuación un ejemplo de la declaración utilidad de este callback

```

public function afterRun($outfilename)
{
    if($outfilename && $this->reporte->id)
    {
        $this->reporte->find_first('id='.$this->reporte->id);

        $this->reporte->mimetype = mime_content_type($outfilename);

        $this->reporte->size = filesize($outfilename);

        $oid = pg_lo_create($this->reporte->db->id_connection);

        $blob = pg_lo_open($this->reporte->db->id_connection,$oid,'w');

        pg_lo_write($blob,file_get_contents($outfilename));

        $this->reporte->data = $oid;

        pg_lo_close($blob);

        $this->reporte->update();

        $this->reporte->commit();

        Flash::notice('Reporte guardado');
    }
}

```

Recuerde que este es solo un ejemplo, dentro del bloque de código de este método usted indicara las acciones que desee realizar con el archivo final: En dado caso que no vaya a realizar ninguna acción, debe de igual forma declarar el método.

Una vez definida la extensión de reporte debe hacer uso de ReportManager para generar el reporte, de la siguiente forma:

```

<?php
include_once('../php-jru/php-jru.php');

$reportManager = new ReportManager();

$reportManager->extensionFolder = 'reportes/extensions/';

$result = $reportManager->RunToBuffer('resumen',PJRU_PDF);

header('Content-type: application/pdf');

print $result;

?>

```

A groso modo podemos deducir lo siguiente: en la primer línea incluye en nuestro código las librerías, a ruta dependerá de donde ubique PHP-JRU. Luego se crea una instancia de ReportManager, se le indica en donde estarán las extensiones de reporte (junto con los diseños jrxml). Ejecutamos el reporte y lo cargamos en la variable \$result que luego imprimimos.

Atributos de ReportManager

- **extensionFolder:** indica la ubicación de las extensión de reporte, donde ademas también deben estar las plantillas jrxml.
- **reportOutDir:** define la ruta donde se va a guardar cada reporte de no indicarse e tomará la ruta de extensionFolder.

Métodos de ReportManager

- **addExtension:** agrega una extensión de reporte a la lista de disponibles.
- **inExtension:** retorna true si la extensión indicada se ha cargado. Recibe como parámetro un string con el nombre de alguna extensión de reporte
- **getAvailableExtension:** retorna un array con la lista de nombres de las extensiones disponibles en extensionFolder.
- **getExtensionInstance:** retorna una instancia de la extensión de reporte indicada por parámetro.

- **delExtension:** elimina de la lista de extensiones disponibles la extensión indicada por parámetro.
- **runToBuffer:** ejecuta la extensión de reporte indicada en un formato específico y retorna el contenido del reporte en un buffer. El primer parámetro es el nombre de la extensión de reporte a ejecutar, el segundo es una constante que indica el formato en el que se va a generar, ejemplo:

```
$result = $reportManager->RunToBuffer('productos',PJRU_PDF);
```

Donde “productos” es el nombre de la extensión de reporte, es decir, \$reportManager buscará un script con el siguiente nombre: productosReportExtension.php. Y PJRU_PDF es la constante que define el formato. \$result es el contenido binario del reporte generado. En el siguiente cuadro se describe las constantes definidas para cada tipo de reporte.

CONSTANTE	VALOR
PJRU_PDF	pdf
PJRU_OPEN_DOCUMENT	odt
PJRU_EXCEL	xls
PJRU_HTML	html
PJRU_RICH_TEXT	rtf

- **runToFile:** Ejecuta un reporte según la extensión indicada y lo guarda en un archivo directamente. El primer parámetro indica el nombre de la extensión de reporte a ejecutar, el segundo indica el formato. Retorna el nombre del archivo generado.

A diferencia de runToBuffer esta función no retorna el contenido del reporte si no el nombre y la ruta, ejemplo:

```
$result = $reportManager->RunToFile('productos',PJRU_PDF);
```

En este caso los parámetros son de la misma forma que en el método antes explicado. El valor de \$result es solo el nombre y ruta completa del reporte.

Es importante resaltar que en el proceso de ejecución de la extensión de reporte, se genera un archivo .jasper que es eliminado inmediatamente que se termina el proceso. Y siempre en ambas funciones runToFile y runToBuffer se genera el reporte físicamente en el disco, la diferencia es que al llamar a runToBuffer el reporte es leído y cargado en memoria y luego eliminado del disco.