OpenReports

Open Source Web Reporting

OpenReports Installation Guide
Version 3.0

Introduction

OpenReports is a powerful, flexible, and easy to use open source web reporting solution that provides browser based, parameter driven, dynamic report generation and flexible report scheduling capabilities.

OpenReports supports a variety of open source reporting engines, including **JasperReports**, **JFreeReport**, **JXLS**, and **Eclipse BIRT**, to provide support for a wide range of reporting requirements and capabilities. OpenReports also supports **QueryReports** and **ChartReports**, easy to create SQL based reports that do not require a predefined report definition.

OpenReports provides a web based report generation and administration interface with the following features:

- Support for a wide variety of export formats including PDF, HTML, CSV, XLS, RTF, and Image.
- Web based Administration of Users, Groups, Reports, Parameters, and DataSources.
- Flexible Scheduling including Hourly, Daily, Weekly, Monthly and Cron scheduling and multiple recipients.
- Comprehensive Report Parameter support including Date, Text, List, Query, and Boolean parameters.
- Fine-grained security controls access to Reports, Scheduling, and Administration functionality.
- Report Auditing tracks start time, duration, status, and user of every report generated.
- Support for multiple JNDI or Connection Pool DataSources for use in generating reports.
- Support for Drill Down reports and external application integration via secure report generation URL.

<u>Note</u>
This document contains instructions to install and run OpenReports. If you are using OpenReports for the first time, the OpenReports Tomcat bundle is a preconfigured demo installation of OpenReports that contains a sample database and example reports that illustrate many core OpenReports concepts.

Installation

Requirements

Operating System

- Any OS that meets the JDK requirements include Windows XP/2000 and most LINUX/UNIX distributions.

Java Development Kit

- JDK 1.5 or higher.

Server

- Servlet Container that supports the Servlet/Jsp specification 2.4/2.0 such as Tomcat 5.5.X

Database

- Most databases with a valid JDBC driver. See http://hibernate.org/260.html for a list.

Create OpenReports Schema

OpenReports stores information about reports, users, groups, etc. in database tables. These tables must be created before using OpenReports. The SQL scripts for many common databases are located in the /database/schema directory.

Note
The Database Schema Help section of the FAQ includes instructions for creating and updating the OpenReports schema via ANT tasks. In order to use the schema related ANT tasks you must first configure Hibernate as outlined below and use the ANT compile task to compile OpenReports.

Create OpenReports Administration User

Once you have created the OpenReports tables in your database, you must insert records into the REPORT_USER and USER_SECURITY tables in order to create an Admin user that you can use to log into and administer the application.

The following SQL statements create a user with the name 'admin', a password of 'password'.

HSQLDB, MySQL, SQL Server:

```
INSERT INTO REPORT_USER (NAME, PASSWORD,PDF_EXPORT_TYPE) VALUES ('admin','password',0)
```

Oracle

```
INSERT INTO REPORT_USER (REPORTUSER_ID,NAME,PASSWORD,PDF_EXPORT_TYPE) VALUES (hibernate_sequence.nextval, 'admin','password',0)
```

PostgreSQL

```
INSERT INTO REPORT_USER (REPORTUSER_ID,NAME,PASSWORD,PDF_EXPORT_TYPE) VALUES (nextval('hibernate_sequence'),'admin','password',0)
```

This SQL statement gives the user Administrative rights:

```
INSERT INTO USER_SECURITY (USER_ID, ROLE_NAME) VALUES (0, 'ROOT_ADMIN_ROLE')
```

Note
The value of the USER_ID field in the second INSERT statement must match the USER_ID of the record created by the first INSERT statement.

Configure Hibernate

The next step in the configuration process is to configure Hibernate so that OpenReports can connect to your database. In order to configure Hibernate you must set the SQL Dialect, and connection properties in the **openreports.properties** file that is located in the src directory.

Here are some common examples:

HSQL: hibernate.dialect=org.hibernate.dialect.HSQLDialect
MySql: hibernate.dialect=org.hibernate.dialect.MySQLDialect
SQL Server: hibernate.dialect=org.hibernate.dialect.SQLServerDialect
PostgreSQL: hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

The next step is to set the connection properties. If you are going to use a JNDI connection, the only property you need to set is hibernate.connection.datasource. For example:
hibernate.connection.datasource=java:comp/env/jdbc/OrDb

If you are not using JNDI, you must supply all the properties needed to create a connection such as the URL and Driver. Examples of these properties are included in the openreports.properties file.

Configure Scheduler

OpenReports uses **Quartz**, an open source enterprise-class Job Scheduler, to provide the ability to schedule reports to be sent via email.
Quartz is configured, in the **openreports.properties** file, to persist jobs to a database via a JDBCJobStore. In order to use the JDBCJobStore you must:

1. Create the Quartz schema using one of the Quartz database scripts located in the database/schema/quartz directory.
2. Edit the Data Source settings in openreports.properties file in the src directory with the connection information for your database.
3. Set the org.quartz.jobStore.driverDelegateClass property in openreports.properties file to the proper delegate class for your database.

See the **openreports.properties** file for examples of these properties.

Note
For more information on configuring Quartz, visit the Quartz project site at http://www.opensymphony.com/quartz/

Deployment

OpenReports can be deployed as an expanded Web Application or as a WAR file.

In order to deploy OpenReports as an expanded Web Application, first update the properties files in the src directory as described above. Next, use the ANT **compile** task to compile the source code and move the properties files to the **WebRoot/WEB-INF/classes** directory. Finally, copy the entire **WebRoot** directory into your target application server and rename the directory, for example to openpeports.

In order to deploy OpenReports as a WAR file, first update the properties files in the src directory as described above. After updating the properties files, a WAR file can be created by ANT using the **war** task from the **build.xml** file. This task will compile the source code into **WebRoot/WEB-INF/classes**, create an **openreports-x.x.x.jar** file in **WebRoot/WEB-INF/lib** and finally build an **openreports.war** file in the deploy directory.
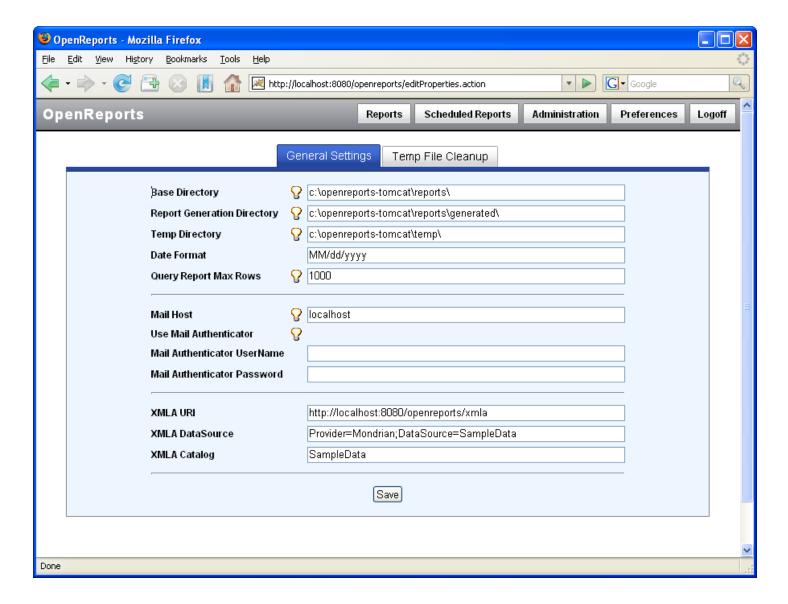
Important
If you are not using JNDI DataSources, the JDBC drivers for all database servers used by OpenReports, including the OpenReports database, the Quartz database, and any reporting databases, must be included in the WebRoot/WEB-INF/lib directory. The FAQ contains links to the JDBC drivers for a number of common databases.

Configuration

After deploying the OpenReports Web Application, you must login to OpenReports as an Administrator and set the **baseDirectory** and **mail.smtp.host** via the General Settings Administration page.

- The baseDirectory property must be set to the full path of the directory containing your JasperReports files.
- The mail.smtp.host property must be set to a valid mail host in order to generate and email scheduled reports.

FAQ

Common Problems

1) **Error loading object from file (or reports not being generated):**

JasperReports requires that you compile your reports using the same version of the JasperReports jar file as the application you are using to run the reports (in this case OpenReports) is using.

To fix this problem, recompile your JasperReports files using the same version of the jar file that is in the openreports/WEB-INF/lib directory.

Another possible solution is to replace the JasperReports jar file in openreports/WEB-INF/lib with the jar file you are using to compile your reports. In most cases they should be compatible, but backwards compatiblity is not guaranteed.

2) **java.lang.NoClassDefFoundError** on Linux:

The JasperReports library will generate java.lang.NoClassDefFoundError exceptions when running on a Linux machine without an X-Server running. The easiest solution is to run in headless mode, which is available in JDK 1.4+. For example, if you are using Tomcat, add the following to your catalina.sh file:

CATALINA_OPTS="-Djava.awt.headless=true"

For more information, search the JasperReports forums on SourceForge for 'headless' or 'NoClassDefFoundError'.

3) Miscellaneous problems inserting records into OpenReports database or Scheduling Reports

First make sure that the dialect in the hibernate.properties file and the driverDelegateClass in the quartz.properties file are set to the correct values for your database.

Next, make sure you are using the latest versions of the appropriate JDBC drivers for your database. Here are links to the JDBC drivers for some common databases:

HSQL - http://sourceforge.net/projects/hsqldb
MySql - http://www.mysql.com/products/connector/j
Oracle - http://www.oracle.com/technology/software/tech/java/sqlj_jdbc
PostgreSQL- http://jdbc.postgresql.org/
SQL Server - http://sourceforge.net/projects/jtds

FAQ - Continued

Encrypted Passwords

OpenReports provides the ability to store OpenReports user passwords as Base64 encoded Strings.

To enable this feature you must edit the ReportUser.hbml.xml file in the src/org/efs/openreports/objects directory. Comment out the default password property entry and uncomment the password property entry that has the type set to org.efs.openreports.util.EncryptedStringUserType

EncryptedStringUserType can be extended if stronger encryption is required.

Database Schema

OpenReports uses Hibernate to provide cross platform database support and the OpenReports distribution includes the DDL for a number of databases. OpenReports also includes two utility classes that can be used to create or update the OpenReports schema. These two classes can be run by ANT using build.xml file included with OpenReports.

1. **ant schemaExporter** - this target will create the entire OpenReports schema in the database indicated in the openreports.properties file.
2. **ant schemaUpdater** - this target can be run to update the OpenReports schema in the database indicated in the openreports.properties file to the latest version.

If you are having problems with the DDL included in the OpenReports distribution, it may be easier to export or update the schema directly using the ANT targets.

Also, make sure that the hibernate.dialect in the hibernate.properties file is set to the proper dialect for your database.

More information on Hibernate can be found at:

http://www.hibernate.org

Resources

For additional information, visit: http://www.oreports.com

License

OpenReports is distributed under the **GPL** License

For alternative licensing, email info@oreports.com

Credits

OpenReports uses the following open-source projects:

JasperReports - Java report-generating library - http://jasperreports.sourceforge.net
WebWork - MVC web application framework - http://opensymphony.org
Hibernate - an object/relational persistence and query service for Java - http://hibernate.org
Quartz - a Java open source enterprise-class Job Scheduler, http://sourceforge.net/projects/quartz
JFreeCharts - a Java charting library - http://www.jfree.org/jfreechart/index.html

OpenReports also uses a variety of other open-source software including software developed by the Apache Software Foundation (http://www.apache.org)