

JASPER REPORTS

KURZE EINFÜHRUNG IN EIN OPEN SOURCE
REPORTING TOOL
FÜR DIE OPEN SOURCE VORLESUNG

VON

TOBIAS HOLZMÜLLER

GLIEDERUNG:

<i>Was ist ein Reporting Tool.....</i>	3
Definition eines Reporting Tools.....	3
Allgemeiner Aufbau eines Reports.....	3
Datenbeschaffung für den Report.....	4
Generierung des Reports.....	5
<i>Jasper Reports.....</i>	5
Arbeitsweise von JasperReports.....	6
Arbeiten mit der JasperReports Bibliothek.....	8
<i>Quellen.....</i>	8

WAS IST EIN REPORTING TOOL

Wenn eine Anwendung entwickelt wird, in der größere Datenmengen verarbeitet werden sollten, ist auch die Darstellung der Daten eine größere Aufgabe. Um die Daten nicht nur in der Anwendung gut darstellen zu können, sondern auch in andere Formate konvertieren zu können, oder einen formatierten Ausdruck dynamisch erstellen zu können, bieten die meisten Programmiersprachen wenig Unterstützung. Diese Lücke füllen Reporting Tools. Früher waren diese Tools Bestandteile kommerzieller Datenbank-Management-Systeme. Das wohl bekannteste Reporting Tool ist Crystal Reports von Microsoft. Mittlerweile gibt es mehrere Open Source Implementierungen, die ein hohes Niveau erreicht haben, z.B. JasperReports, JFreeReport und DataVision.

DEFINITION EINES REPORTING TOOLS

“Ein Reporting Tool ermöglicht es, das Layout eines Druckerzeugnisses zu definieren, es mit Daten zu versehen und es in möglichst viele Dateiformate oder Darstellungsmedien exportieren bzw. anzeigen zu können.“

Reporting Tools bestehen aus drei Komponenten:

- 1. Die Definition des Layouts.
- 2. Füllen des Layouts mit Daten aus verschiedenen Quellen.
- 3. Export in das jeweilige Format, d.h., Darstellung auf dem Bildschirm oder senden an den Drucker.

ALLGEMEINER AUFBAU EINES REPORTS

Das Layout des Reports unterteilen die meisten Reporting Tools in 5 Sektionen. Diese werden als Page Header, Report Header, Detail, Report Footer und Page Footer bezeichnet. Sie sind zur Darstellung der jeweiligen Sektion im Dokument zu verwenden. Die verschiedenen Tools haben aber auch noch spezifische Sektionen die nur im jeweiligen Tool definiert sind. Dies ist zum Beispiel der Group Header, der durch den Group Footer ergänzt werden kann. In Abb. 1 ist ein Beispiel einer Rechnung vereinfacht dargestellt. Hieraus sollte ersichtlich sein, wie das Tool Daten mit den entsprechenden Sektionen darstellt. Hier wird in der Detail-Sektion jeweils eine Bezeichnung und ein zugehöriger Preis angegeben. Auf der zweiten Seite wird von den Preisen die Summe gebildet. Wenn der Inhalt auf einer Seite nicht mehr darstellbar ist, wird automatisch mit einer neuen Seite begonnen. Mit den Sektionen Page Header und Page Footer wird eine Kopfzeile und eine Fußzeile definiert, die den Report optisch ansprechender aussehen lässt. Der Report Header wird nur einmal ausgegeben.

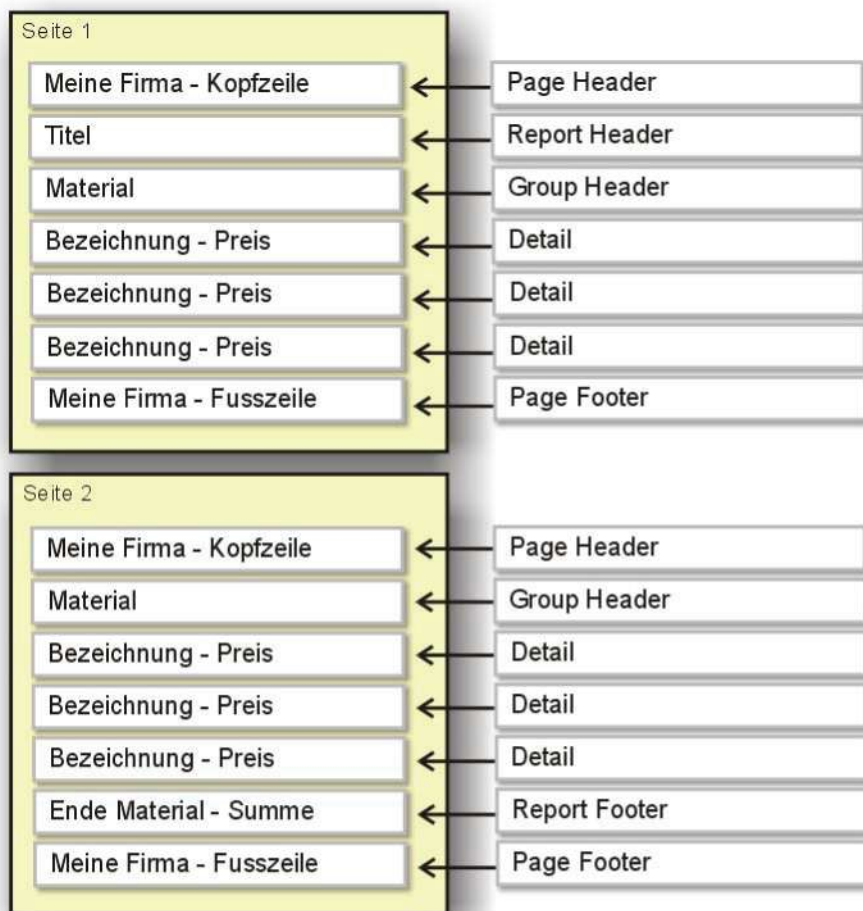


Abb. 1

Die Platzierung grafischer Elemente ist in jeder Sektion möglich, wobei hier nur wenige primitive Objekte zur Verfügung stehen. Diese sind Linien, Polygone und Textfelder. Sie reichen aber normalerweise aus, um komplexe Dokumente zu gestalten. Auch das Importieren von Bildern ist möglich. Ein interessantes Feature sind Subreports, damit ist es möglich, Reports ineinander zu verschachteln. Eine wichtige Eigenschaft von JasperReports ist die rekursive Verwendung von Subreports. Diese Funktion macht JasperReports zu einem der führenden Open Source Reporting Tool.

DATENBESCHAFFUNG FÜR DEN REPORT

In früheren Versionen der Reporting Tools wurden die Daten mittels SQL-Abfragen in den Report eingelesen. Mittlerweile stehen dem Entwickler mehrere verschiedene Datenquellen zur Verfügung, aus denen er mittels Treiber Daten in den Report einlesen kann. Da JasperReports in Java geschrieben ist, hat auch das Objekt, das die Daten aus der Datenquelle holt, Ähnlichkeit mit dem Java-Objekt `java.sql.ResultSet`. Die Abarbeitung des Layouts erfolgt sequentiell. Demnach werden auch die Sektionen sequentiell generiert. Hier tritt eine Schwäche des Konzepts zutage. Daten, die in baumähnlichen Strukturen verfügbar sind lassen sich schlecht tabellarisch abbilden. Als Beispiel können hier XML-Dokumente aufgeführt werden. Dieses Problem wird in JasperReports mittels Subreports und einer Klasse für XML Datenquellen behoben.

GENERIERUNG DES REPORTS

Das Layout wird als XML Variante gespeichert. Nach dem Compilieren wird es in einem reporttool-spezifischen Format abgelegt. In einem weiteren Schritt wird aus dieser Datei das jeweilige Ausgabeformat erzeugt.

JASPER REPORTS

Jasper Reports ist zu einem mächtigen Report Tool gewachsen. Dieses Tool wird seit September 2001 vom Rumänen Teodor Danciu entwickelt. Als Lizenz wird die GNU Lesser Public License in der Version 2.1 angegeben. JasperReports ist streng genommen nur eine Klassenbibliothek, da der Begriff Tool gebräuchlicher ist, wird auch dieser verwendet. Mit JasperReports ist es möglich, aus gängigen Datenquellen mit Hilfe einer Formatvorlage druckfertige Inhalte auf Drucker, Bildschirm oder in Dateien des Formats PDF, HTML, XLS, CSV oder XML zu exportieren. Das Projekt wird ständig erweitert. Daher werden noch weitere Formate hinzukommen. Da es in Java geschrieben ist, kann es in standalone Java-Applikationen oder Webapplikationen eingebaut werden. Die Definition des Layouts erfolgt in einem für JasperReports angepassten XML-ähnlichen Format. Um diese Dateien editieren zu können, gibt es grafische Editoren wie iReport oder ein plugin für Eclipse namens JasperAssistant. iReport ist eine Java-Applikation, die ebenfalls als Open Source Projekt im Sourceforge.net zu finden ist. Das Eclipse plugin ist kostenpflichtig. Diese Editoren sind sehr komfortabel und unerlässlich, da es sehr mühsam wäre, die Layoutdateien mit der Hand zu schreiben. Das nachfolgende Bild zeigt das Standard Layout wie man es vorfindet, wenn man einen neuen Report mit dem Tool iReport generiert.

title
pageHeader
columnHeader
detail
columnFooter
pageFooter
lastPageFooter
summary

Abbildung 2: Standardvorlage von iReport

Als nächstes eines der vielen Samples, die bei JasperReports dabei sind.

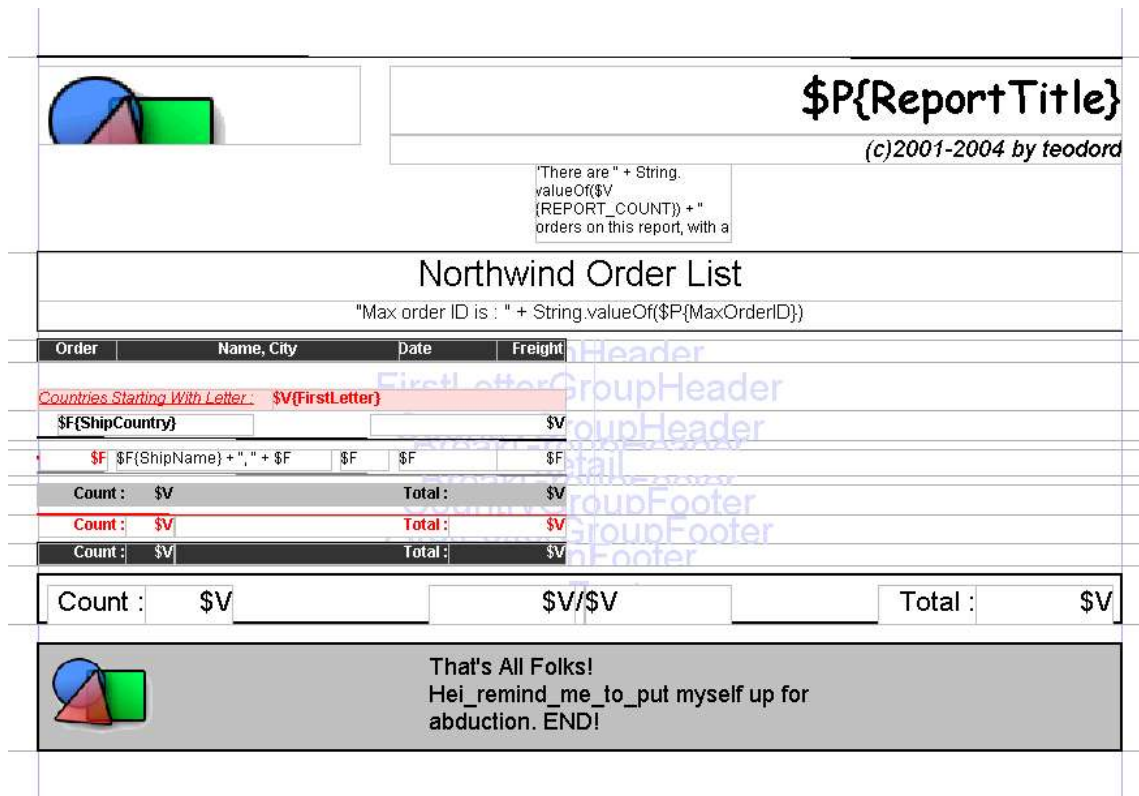


Abb. 3 mit iReport gestaltetes Layout

ARBEITSWEISE VON JASPERREPORTS

Wie weiter oben schon erwähnt, besteht JasperReports aus mehreren Grundkomponenten. Zuerst wird die jrxml Datei (in dieser ist das Layout enthalten) zur jasper Datei compiliert. Hier sind die Sektionen definiert. Die Syntax des Layouterstellungsprogramms, hier iReport, ist intuitiv. Hier werden die statischen und die dynamischen Parameter eingetragen. Es muss auch im Erstellungstool angegeben werden, von welcher und in welcher Form die Daten bereit liegen. Dies kann unter Datenquelle eingestellt werden.

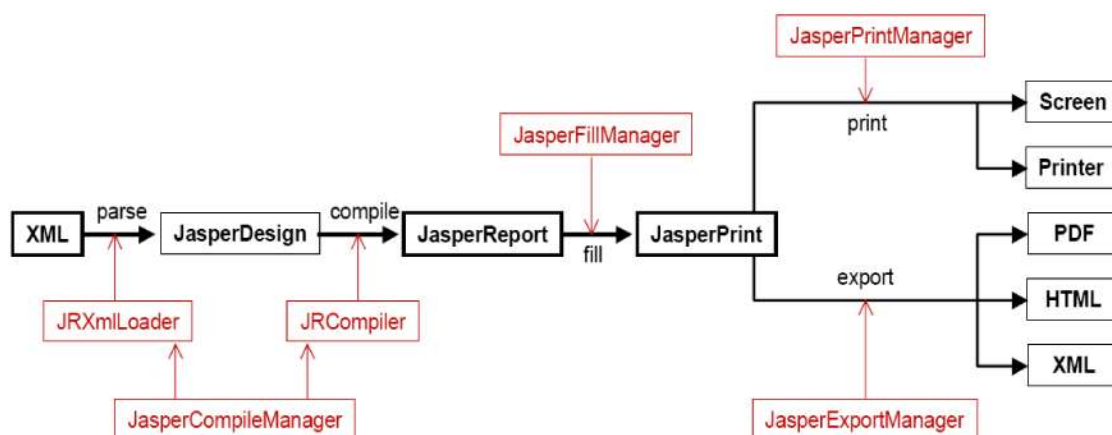


Abb. 4 Klassenübersicht im Detail 1

Das erstellte jrxml File wird vom JasperCompilerManager mit Hilfe einer DTD validiert. Nach der Validierung wird daraus Java Quellcode generiert und dieser wird dann compiliert. Daraus resultiert dann das File mit dem jasper Format. Dies ist mit normalen Editoren nicht lesbar. Es wird im Dateisystem gespeichert und stellt die Grundlage des Reports dar. Dieser Prozess findet nur bei der Erstellung des Reports statt. Das jasper File sollte bereits compiliert in der Anwendung vorliegen, denn der Compilierungsprozess würde sich sehr negativ auf die Performance niederschlagen.

Die jasper Datei wird mittels JasperFillManager mit den Daten aus der angegebenen Quelle ergänzt, wie es in der Abb. 4 und 5 zu sehen ist. Es gibt mehrere verschiedene Möglichkeiten die Daten zu übergeben. Die erste Möglichkeit ist, im Report die dynamischen variablen SQL-Statements anzufügen und das Auslesen der Daten an JasperReports zu übertragen. Dafür wird lediglich eine java.sql.Connection, die die Verbindung zur Datenbank enthält, übergeben. Als zweite Möglichkeit kann eine JRDataSource Implementierung übergeben werden. In der JasperReports Bibliothek ist ein Unterordner Samples. In diesem befinden sich mehrere Implementierungen, die je nach Bedarf verwendet werden können. Auch als java.util.Map lassen sich Parameter übergeben.

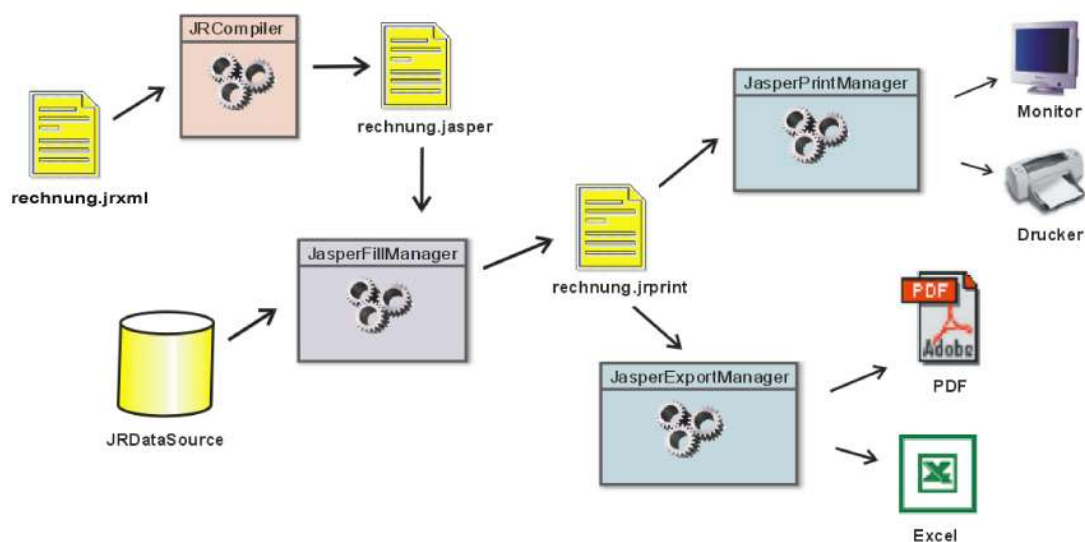


Abb. 5 Darstellung des Compilierungsprozesses

Nach dem Füllprozess entsteht ein JasperPrint-Objekt mit der Dateieindung jrprint. Diese kann als Datei

- auf die Festplatte geschrieben werden,
- im Hauptspeicher belassen werden,
- dem JasperPrintManager übergeben und dort für ein Ausgabemedium aufbereitet werden,
- über den JasperExportManager in das gewünschte Dateiformat konvertiert werden.

Die Kapselung des Codes ist soweit fortgeschritten, dass das Ausführen der Prozesse sehr einfach implementiert werden kann. Der Ausführungsprozess kann über das ANT-Bildtool angestoßen werden oder über Aufrufparameter mit der compile Anweisung. Der größere Aufwand steckt jedenfalls nicht in der Implementierung der Prozesse sondern in der Gestaltung des Layouts.

ARBEITEN MIT DER JASPERREPORTS BIBLIOTHEK

Wenn man beginnt mit dieser Bibliothek zu arbeiten, liest man erst die Seiten im Internet, lädt sich die Bibliothek herunter und legt diese als Projekt in einer Entwicklungsumgebung an. Beim Betrachten der Verzeichnisse kommt man schnell zum Verzeichnis Samples. Dieses ist speziell für den Einsteiger sehr wichtig, da es die Beispiele beinhaltet. Insgesamt stehen 26 verschiedene Beispielreports zur Verfügung. Anhand dieser Beispiele kann man sehr schnell erkennen ob der gewünschte Report damit zu erzeugen ist. Um gleich loslegen zu können, kann man sich noch ein Designtool herunterladen. Die Designtools sind alle auf der Seite jasperreports.sf.net unter GUI Tools zu finden. Wenn man die jrxml Datei vom jeweiligen Beispiel in das Designtool lädt, kann man das Samplesdesign bearbeiten, oder ein Neues erstellen.

Um den Compilierungsprozess zu starten, muss man sich die Sourcen genauer anschauen. Es gibt die Möglichkeit über das ANT-Bildtool die Beispiele auszuführen, die jeweilige Klasse über die Konsole aufzurufen, oder mit einem Wizard in der Entwicklungsumgebung diesen Prozess anzustoßen. Wichtig sind hierbei die Aufrufparameter, die angegeben werden müssen.

Die Parameter für das Bildtool sind im jeweiligen Sample in der Datei build.xml zu finden. Die Übergabeparameter für die Konsole oder den Wizard sind in der Javaklasse in der die verschiedenen Prozesse implementiert sind zu finden.

QUELLEN

1. www.jasperreports.sf.net
2. www.cs.fh-aargau.ch/~gruntz/courses/sem/ss04/jasperreports.pdf
3. JasperReports Ultimate Guide 1.0
4. jasperreports.0.6.8.jar