JasperReports als Nachfolger von Oracle Reports

Wolf G. Beckmann
TEAM GmbH
Paderborn

Schlüsselworte
JasperReports, Oracle Reports, PL/SQL

Einleitung

Oracle Reports ist abgekündigt. Oracle sieht als Nachfolger den BI-Publisher an. Aber passt der BI-Publisher auch zu den Anforderungen meiner Projekte? Wir, die TEAM GmbH, haben schon vor der Abkündigung nach einer Alternative für unsere Logistik-Software ProStore® gesucht und haben JasperReports für uns entdeckt.

Was kann ein Nachfolger von Oracle Reports sein
Oracle Reports ist in der Oracle Welt fest verankert, das hat mehrere Gründe sicherlich ist ein Grund, dass es als Teil der Fusion Middleware gut in Forms und mit der Datenbank zusammenarbeitet beziehungsweise integrier ist. Auf der anderen Seite ist aber auch Oracle Reports sehr funktional. Somit wurde ein Nachfolger für Oracle Reports gesucht, der wie Oracle Reports folgende Attribute besitzt:

- Visueller Editor
- Präzise und absolute Positionierung für Druckmittel
- Einfache Anbindung der Daten
- Effizienz / Geschwindigkeit

TEAM hat daraufhin eine Auswahl an Produkten am Markt bezüglich der Anforderungen getestet. Die Produkte waren im Einzelnen:

- Oracle BI Publisher
- Apache FOP
- Birt
- Crystal Reports

und natürlich

- JasperReports

Und hier hat JasperReports alle Attribute erfüllt.

Komponenten von JasperReports
JasperReports selbst ist eine Java-Library. Aufbauend auf dieser Library werden zwei weitere Produkte angeboten. Der JasperReports Server und das JasperReports Studio.

Dar JasperReports Server ist eine Serverkomponente, die es erlaubt selbstständig Reports auszuführen und entspricht somit weitestgehend dem Oracle Reports Server

Das JasperReports Studio ist eine integrierte Entwicklungsumgebung über die die komplette Entwicklung inklusive Preview und Deployment zum Server durchgeführt wird. Das JasperReports Studio basiert auf Eclipse und stellt das Gegenstück zum Oracle Reports Designer dar.

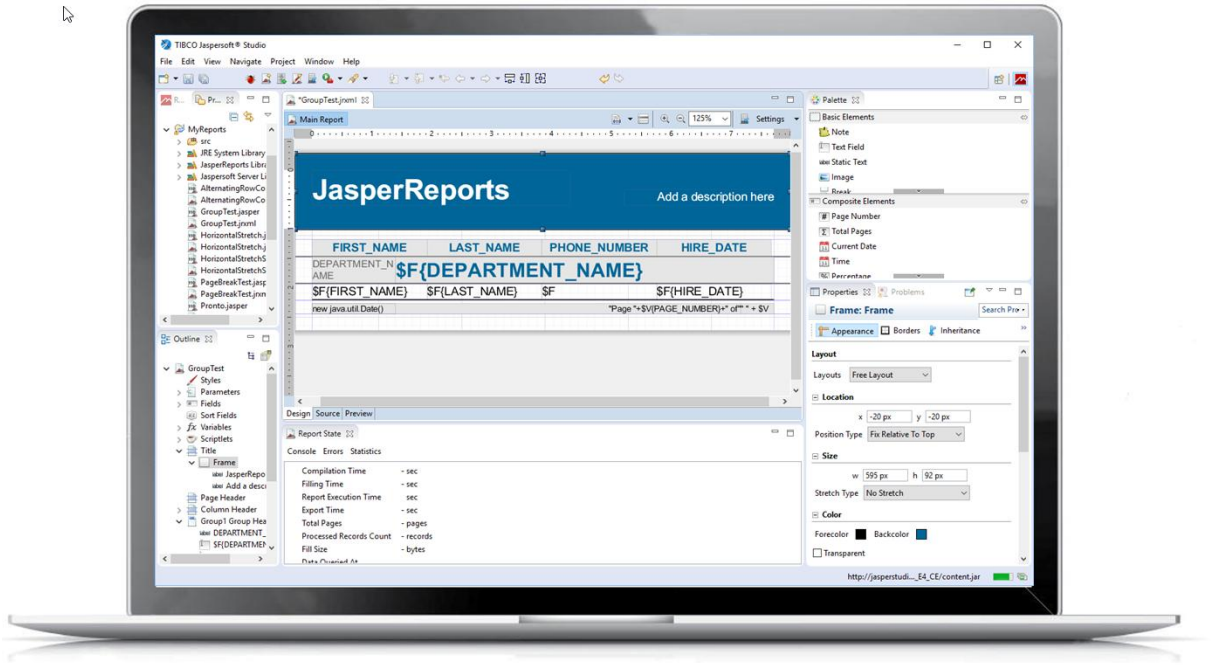
Aufbau eine s JasperReports
Anders als ein Oracle Report werden nicht sogenannte Repeating Frames in einem Report verwendet. Kurz gefasst stellt ein Repeating Frame das mehrzeilige Ergebniss eines Selects dar, ist frei Positionierbar im Report und vergrößert sich mit der Anzahl der Ergebniszeilen.

In JasperReports stellt der ganze Report das Ergebniss eines Select‘s dar. Dazu wird der Report in horizontale Bänder unterteilt, die ihre spezifische Aufgabe bekommen:

- Page-Header
- Page-Footer
- Group-Header
- Group-Footer
- Background
- Detail-Band
- und weitere.

Dabei wird Beispielsweise für jede Ergebniszeile das Detail-Band wiederholt angezeigt.
Um mehrere parallele oder geschachtelte Abfragen in einem Report unter zu bringen können Unterabfragen in Listen oder Sub-Tabellen oder in ganzen Sub-Reports untergebracht werden, die wieder im Report an der gewünschten Stelle positioniert werden können.

Demonstration des JasperReport Studios
Um darzustellen wie JasperReports arbeitet erfolgt eine Live-Demonstration:



In dieser Demo wird folgendes demonstriert:

- Konfiguration der Verbindung zu einer Oracle Datenbank
- Erstellen eines Reports der nach Departments gruppiert die Employees mit Name, Gehalt und Einstellungs-Datum darstellt.
- Formatieren der Feldes Gehalt zu einem Währungsfeld

- Formatieren des Feldes Einstellungsdatum zu einem Datumsfeld der Form DD.MM.YYYY
- Hinzufügen eines alternierenden Hintergrund für die Ausgabetabelle zur besseren Darstellung (Zebra-Tabelle)

Der JasperReport Servers
Der JasperReports Server bietet die Möglichkeit Reports in einer Laufzeitumgebung auszuführen. Typischerweise läuft er auf einem Tomcat Servletcontainer oder Glassfisch Application-Server. Als Repository-Datenbank wird eine MySQL oder PostGreSQL Datenbank verwendet.

Der JasperReports Server bietet ein Benutzerberechtigungs-System über das festgelegt werden kann, welcher Benutzer welche Reports sehen und aufrufen kann.

Desweiteren hat der JasperServer eine sehr ausgeprägte REST-Schnittstelle über die viele Funktionaltäten (und natürlich auch das Ausführen von Reports) möglich ist.

In einer kurzen Live-Demo wird gezeigt:
- Deploy des erstellten Reports auf den JasperReports Server über das JasperReports Studio
- Ausführen des Reports auf dem Server

Aufrufen von PL/SQL-Funktionalitäten
Für Oracle Reports Entwickler ist wohl die wichtigste Frage wie PL/SQL –Funktionalitäten aus dem Report heraus aufgerufen werden.

JasperReports bietet die Möglichkeit Java-Funktionen und Ausdrücke ausführen zu lassen.

Zum einen wird für die Anzeige eines Report-Elementes die Daten über eine (Java-Expression) ermittelt, die im einfachsten Fall den Wert der aktuellen Ergebnisspalte des Report-Select's darstellt. Stattdessen kann aber auch Java-Code in sogenannten Scriptlets ausgeführt werden.

Scriptlets sind Java-Classen die um Funktionalitäten zum Zugriff auf Werte des Reports angereichert sind.

Desweiteren werden in einem Scriptlet automatisch zu definierten Zeitpunkten Funktionen aufgerufen, beispielsweise wenn eine neue Seite beginnt, um Berechnungen durchführen zu können.

Der Aufruf von PL/SQL-Funktionalitäten erfolgt in den Scriptlets über JDBC.

Konkret wird bei der Ausführung des Haupt-Select's des JasperReport eine Datenbank-Verbindung aufgebaut. Diese Datenbank-Verbindung wird im Java-Code verwendet um zusätzliche Anfragen an die Datenbank abzusetzen.

Um in einem Scriptlet die aktuelle Datenbankverbindung zu ermitteln kann folgender Code verwendet werden:

```
private Connection getConnection() throws JRScriptletException {
   return (Connection)
(getParameterValue(JRFillParameter.REPORT_CONNECTION));
}
```

JDBC funktioniert ähnlich wie die Datenbank-Package DBMS_SQL. Der Aufruf wird als Zeichenkette übergeben in dem Platzhalter (Bind-Variablen) vorhanden sind, die beim Aufruf mit Werten gefüllt werden und die auch nach dem Aufruf ausgelesen werden können.

Da ein PL/SQL-Block in mit `begin` und `end` eingerahmt werden kann die Arbeit über folgende Hilfsfuntion erleichtert werden:

```
protected CallableStatement preparePlSqlCall(String sql) throws
JRScriptletException, SQLException {
  return getConnection().prepareCall("begin " + sql + "; end;");
}
```

Dann kann der eigentliche Aufruf wie folgt aussehen:

```
public String getDeptName(Integer deptId) throws JRScriptletException {

    try (CallableStatement stmt = preparePlSqlCall(
            ":result := pk_hr_info.get_dept_name(p_department_id => :p_department_id)"))
    {
        stmt.registerOutParameter("result", Types.VARCHAR);
        stmt.setInt("p_department_id", deptId);
        stmt.execute();
        return stmt.getString("result");
    } catch (SQLException e) {
        throw new RuntimeException("Fehler beim Aufruf von get_dept_name", e);
    }
}
```

Diese Beispielfunktion kapselt den PL/SQL-Aufruf de Package-Funktion `pk_hr_info.get_dept_name()` in der Java-Methode `getDeptName()`.

Diese Java-Methode kann dann in einer Expression, die einen Wert im Report-Anzeigt direkt aufgerufen werden.

Das wird in einer kleinen Demo dargestellt in der in der Demo erstellte Report dahingehend erweitert wird, dass der Department-Name nicht mehr aus dem Tabellen-Feld sondern über die oben gezeigte Funktion ermittelt wird.
Anmerkung: Die PL/SQL-Funktion erweitert den Department-Namen um den Standort.

Kontaktadresse:
Wolf G Beckmann
TEAM GmbH
Hermann-Löns-Str. 88
D-33104 Paderborn

Telefon: +49 (0)5254-8008 39
Fax: +49 (0)5254-8008 19
E-Mail wb@team-pb.de
Internet: www.team-pb.de