Reporting For The Web With DataVision

Frank W. Zammetti

During my last web project, a fairly mundane requirement came up: the ability to view management-type reports from within the application. Nothing unusual, however, due to the fact that my employer had not yet decided on an enterprise-wide reporting solution, it was up to me to decide which way to go.

The project was already in motion, budgets set, timelines promised, so there were some quick decisions to be made. First, there was no money to buy a COTS (Commercial Off The Shelf) solution, so open-source was the answer. But which reporting package to choose?

Having extensive experience with Crystal Reports in the past, I wanted something similar. I wanted a GUI report writer. I also needed it to be quick and easy to integrate into my web app. It had to be easy to wrap my brain around in short order. It had to support user-definable parameters since a number of the reports would require that. It had to be capable of generating print-ready output, Adobe Acrobat being the obvious choice.

After a few hours of searching SourceForge and playing with various options, I came upon the solution that met all of my requirements…

Enter DataVision.

DataVision is a Java-based reporting package written by Jim Menard. It has a Swing-based report writing tool, uses iText underneath to generate PDFs most commonly (among other output formats), and is about as simple to use as I could hope for. Plus, being under the Apache Software License, I had the source to address any shortcomings I might find (and in fact I have contributed to the project since).

This article will introduce you to DataVision and show an example of integrating it into a web application.

The DataVision home page can be found at http://datavision.sourceforge.net/. The latest version at the time of this writing is 1.0, and this article is written for that version. Note that due to some public API changes, versions prior to 1.0 WILL NOT work with the sample application

Before we can really begin, we will need a database to play with. It will be left as an exercise to the reader to create a table with the structure shown in figure 1 in whatever RDBMS you would like to use. Please be sure you have a JDBC driver for the system you choose.

Figure 1.
Table Name: DataVision

Field Name	Type	Length
FirstName	VarChar2	20
LastName	VarChar2	30
Age	Number	3

(Note: VarChar2 is whatever simple text type your RDBMS supports, and Number is whatever simple Integer type your RDBMS supports. I created mine in Oracle, hence the Oracle-centric types).

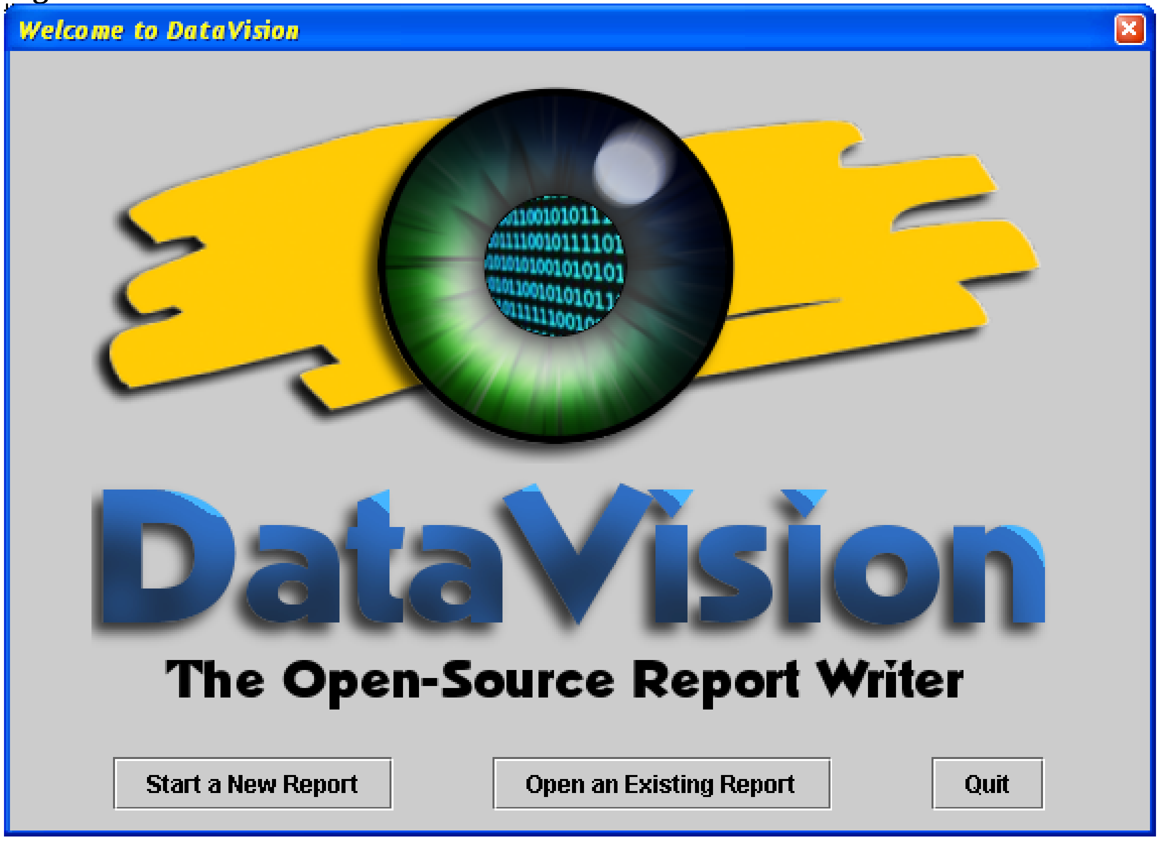
Once the table is completed, please populate it with some data. A simple list of a few of your friends, family and coworkers will be sufficient.

With the database out of the way, let's start with the basics…

Once you have downloaded the DataVision package from the DataVision home page, unzip it and you will find the directory "datavision-1.0.0" has been created. Within this directory are two files, **datavision.bat** and **datavision.sh**. These are the DataVision startup files. If you are using Windows, execute the first, if you are using a *nix system, execute the second. Everything DataVision requires to run will be temporarily added to the classpath for this execution. The only thing that is left up to you is to ensure that any needed JDBC driver is already in your classpath (or you can modify the DataVision startup file if you would prefer the driver only be on the classpath temporarily).

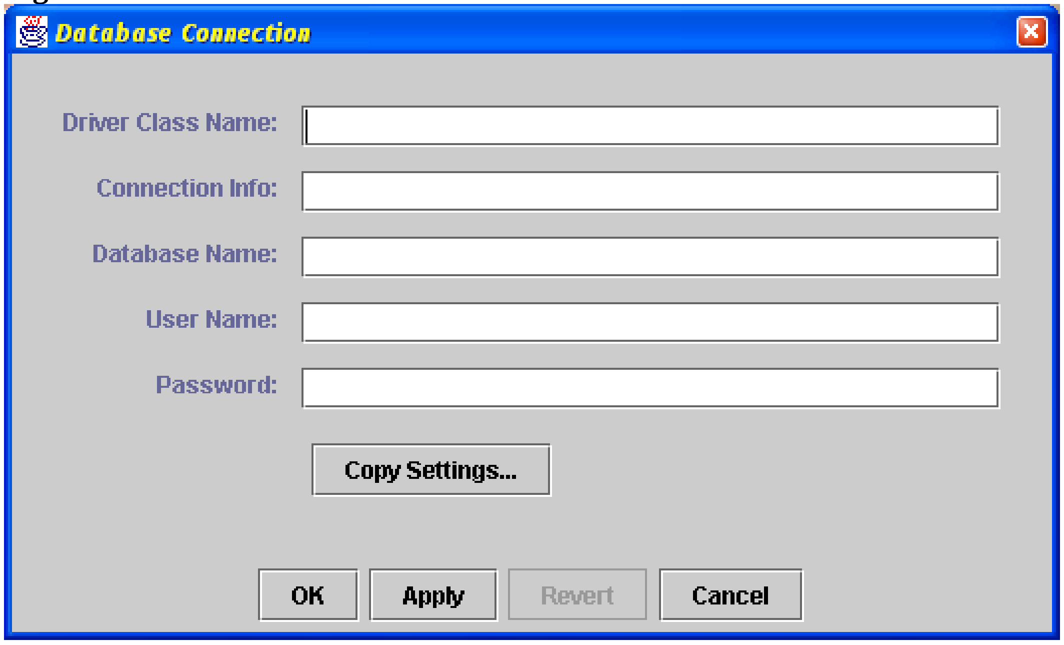
You will be greeted with the very nice startup screen shown in Figure 2 (I did it, so I think it's very nice!).

Figure 2.



Simply click the Start a New Report button to get started. You will be presented with the window in figure 3.

Figure 3.



The fields you need to enter are:

- **JDBC driver class name** You should be able to find the JDBC driver class name from your database driver documentation.

- **Connection Info - Y**ou should be able to find the JDBC connection info string from your database driver documentation. It will probably contain either the name of the database schema or the "sid". Sometimes it is the same as the database name. For example, to connect to the table from figure 1 in my Oracle database on the server “testserver” on port 1521 with the service name DVTEST, the string is **jdbc:oracle:thin:@testserver:1521:DVTEST**. In simplest terms, this is the connection string you would use when programming JDBC.

- **Database Name –** This is sometimes called the schema name. The database name refers to a set of database tables. (This isn't the name "Oracle" or "PostgreSQL".) When DataVision asks the database for the list of table names, it uses the database name to specify which tables to return.

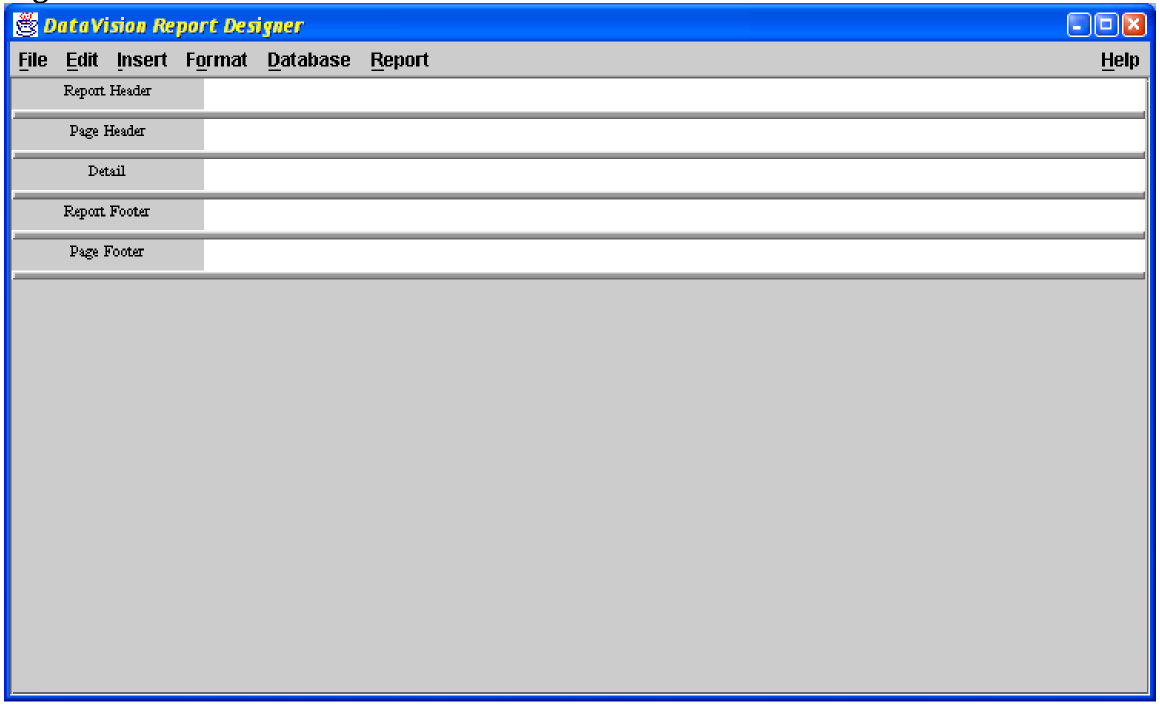
- User **Name -** This is the user name you use to connect to the database.

- **Password - Y**ou must enter the database password each time you open a report.

If you have an existing report, you can use the Copy Settings button to grab all the above setting, except for password, from that report. DataVision stores its reports as XML files, so this button, as well as the Open an Existing Report button on the startup screen, results in a file system browser dialog.

Once you have entered all that information and DataVision has connected to the database, you will see the basic designer window as shown in figure 4.

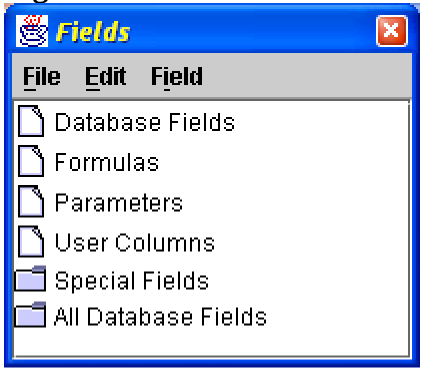
Figure 4.


DataVision Report Designer
File Edit Insert Format Database Report Help
Report Header
Page Header
Detail
Report Footer
Page Footer

If you have ever used a banded report writer before, this will be quite familiar to you. It is not my intention to go into a lot of detail about using the report writer. I will just touch upon enough to get a simple report created, using the table created earlier, that we can integrate into a web application.

To begin, click the Insert menu, then the Database Field option. You will be presented with the Fields dialog shown in figure 5.

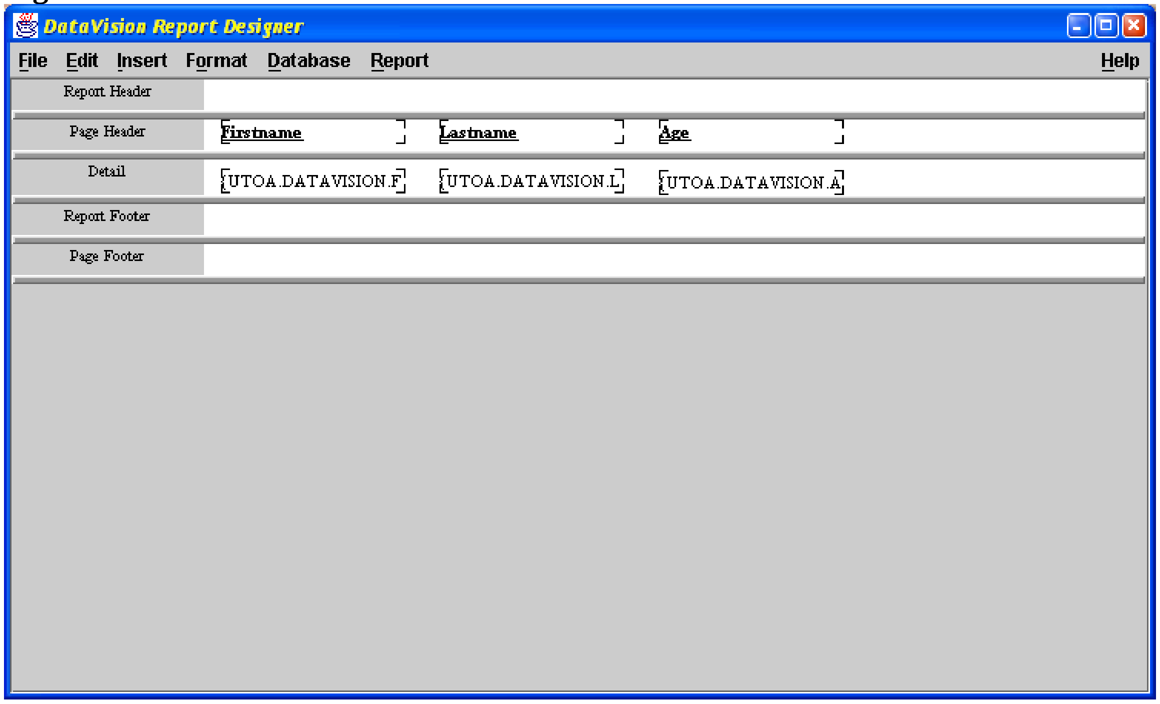
Figure 5.



Double-click the All Database Fields option. This will show you all the tables in the namespace you provided, and you can expand them to see the fields of each table. Through this, find the DataVision table you created.

To add a field to the report, simply drag it from the Fields dialog. We are going to add all three fields, so drag the First Name, Last Name and Age into the Detail section of the report. Your designer should then look something like Figure 6.

Figure 6.



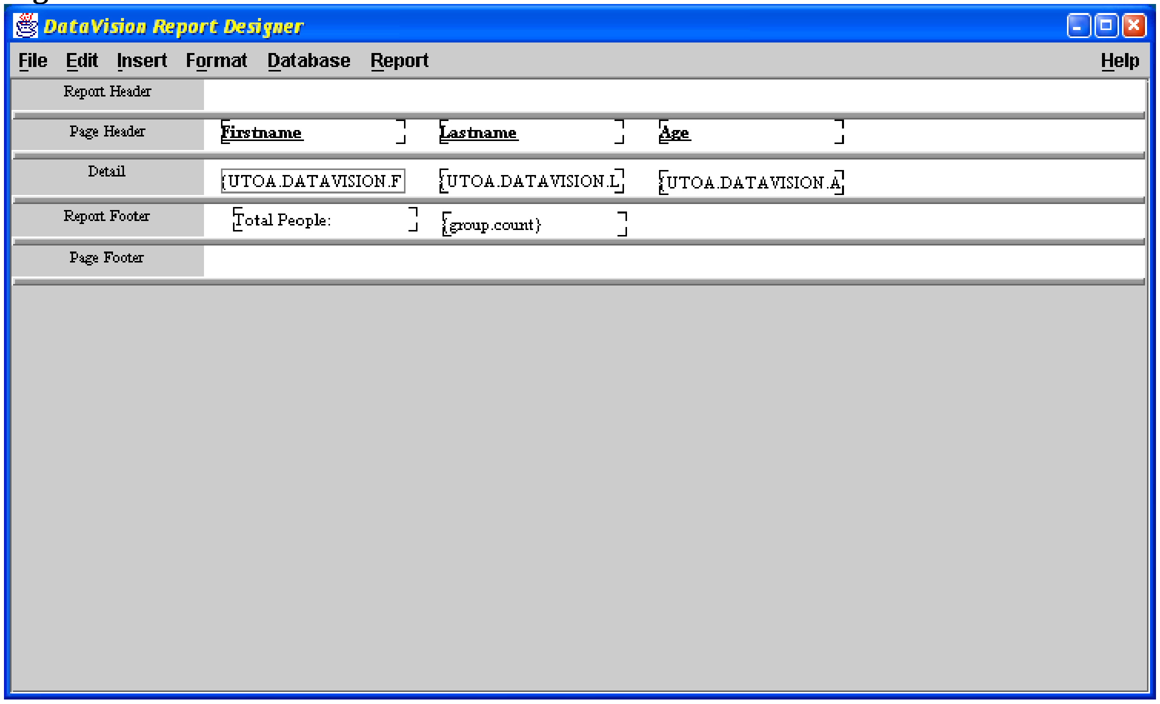
Don’t worry about lining anything up, this can be rough. As you can see, DataVision kindly adds field headings for us in the Page Header section.

Now let’s insert a total count of people. First, insert a simple text field with the content “Total People: “. To do this, select Insert, then Text. You will notice that your cursor turns into an insert I-Bar image. Click somewhere in the Report Footer section and a text field will be inserted. Simply type your text.

Next, click Insert and then Special Field. Drag Group Record Count next to the text field you created.

Now save your report under the name dvtest.xml using the Save As option on the File menu. Just for the sake of this article, save it directly into the datavision-1.0.0 directory. The final report should look something like figure 7.

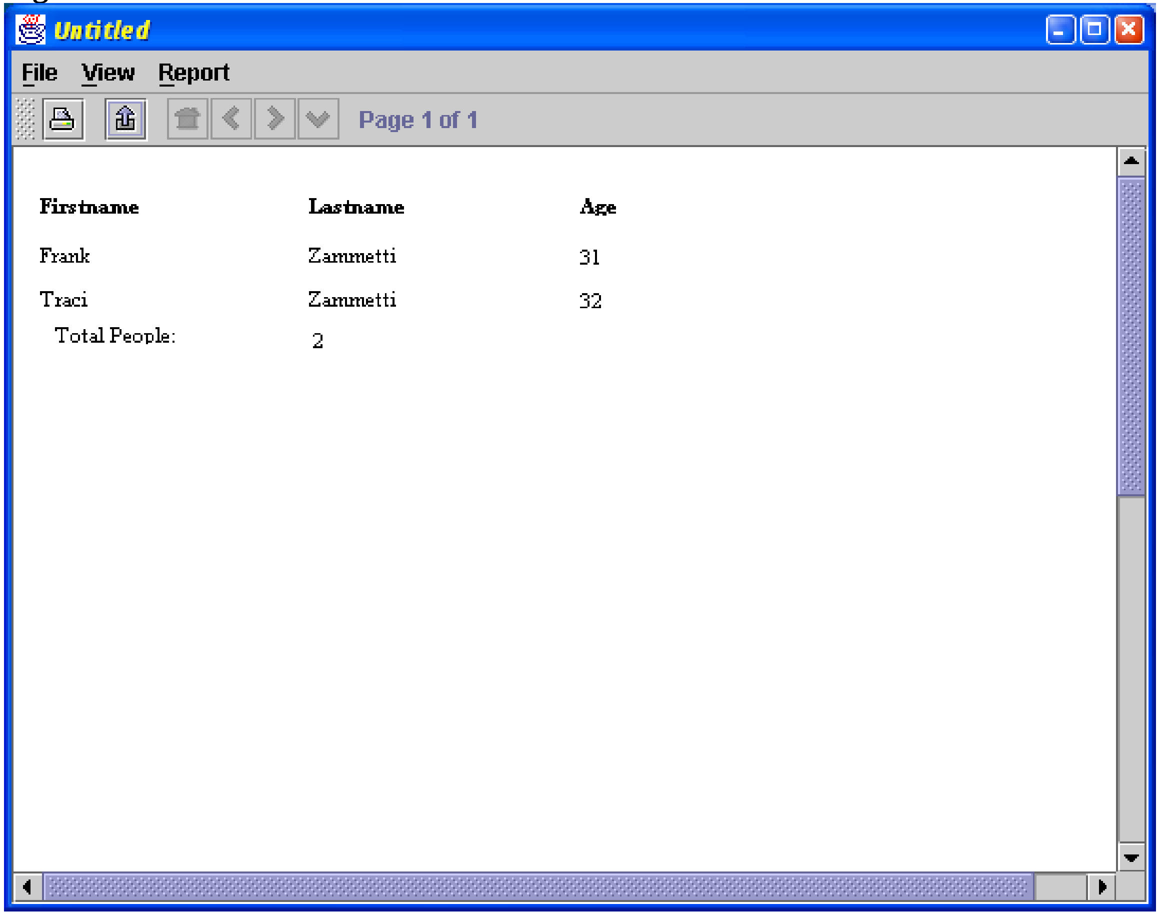
Figure 7.


DataVision Report Designer
File Edit Insert Format Database Report Help
Report Header
Page Header
Detail
Report Footer
Page Footer

Page Header	Firstname	Lastname	Age
Detail	{UTOA.DATAVISION.F	{UTOA.DATAVISION.L}	{UTOA.DATAVISION.A}
Report Footer	Total People:	{group.count}	

To run your report, simply click the Report menu, then the Run option. You should be greeted with something that looks like figure 8.

Figure 8.



Firstname	Lastname	Age
Frank	Zammetti	31
Traci	Zammetti	32
Total People:	2	

Ok, so now we have a report, albeit a simple and ugly one! Let's see what we can do with it. First, close the report designer. One thing we can do is run the report and generate output without using the report designer (or any visual cue actually). This can be helpful if you want to generate the report but not use it immediately. To do this, use a command line like the following:

```
datavision -f out.pdf -p dbpassword dvtest.xml
```

Just replace dbpassword with the actual password for your database and you should wind up with a PDF named out.pdf in the current directory. There is a great deal of flexibility when running a report from the command line like this. I suggest typing datavision with no parameters to see what is possible. The documentation that came with the DataVision download will of course explain it all in detail.

Now, let's get to the meat of this article: how do we integrate DataVision into a web application?

If you haven't already done so, download the sample webapp referenced at the end of this article. Install it into your servlet container of choice. It is supplied in exploded format, so just copying the directory created when unzipping it should work. For instance, if you are using Tomcat, simply copy the dvwebapp directory into <TOMCAT_ROOT>\webapps. Before starting your app server, check web.xml and modify the four servlet init parameters to mimic the parameters you entered when creating the report. Also, be sure your container can find your jdbc driver. That should be all you need to do. Start up your app server and give it a shot!
Access the application using http://localhost:8080/dvwebapp (replacing 8080 with whatever port your app server is listening on).

This example outputs HTML. I chose to output HTML for two reasons; first, so that you will have by this point seen output in three different formats: Swing, PDF and HTML. Second, because there are known issues with the Adobe Acrobat browser plug-in which I have personal experience dealing with, I didn't want to risk the sample not working. HTML was a nice, safe choice!

Please note that this example shouldn't be taken as anything other than a demonstration of DataVision in a web application. If you are looking for an example of the best way to write a webapp, this probably is not it. That being said, I don't think I've committed any truly egregious errors!

Let's examine the code a bit. Index.htm is nothing but a simple form that submits to our servlet class, DVWebappServlet. It is a trivial piece of code, so we'll ignore it the rest of the way..

DVWebappServlet is where everything happens. It is a completely typical servlet, and for the sake of simplicity I have only implemented doPost. Figure 9 shows the entirety of that code.

Some quick notes about the webapp…

- It is supplied completely compiled; you don't need to build it initially. If you want to play though, an Ant build script is supplied in WEB-INF/src.

- The servlet-api.jar file in WEB-INF/src is only required for compilation. Your servlet container should already have this.

- The JARs in WEB-INF/lib are required for DataVision to run in the most basic way. iText.jar will be required if you wish to output to PDF. You will find some other JARs in the datavision-1.0.0/lib directory, but these should only be required for the report designer, or potentially if integrating DataVision into a Swing client. For a webapp outputting to HTML, the three you see in that directory should be all you need.

- Feel free to replace the dvtest.xml sample report with one of your design, pointing to a more interesting database to get the hang of things. I encourage you to play with the report designer. You should be comfortable with it in no time if this is not your first exposure to a banded report writer. Note that formulas are written in Ruby (at least until the patch I submitted to extend scripting support to other languages is added!). I had never used Ruby before I started using DataVision, and even now I only know enough to get by, but I have never encountered something I couldn’t accomplish in short order.

Figure 9.

```
package dvwebapp;

import java.io.InputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringReader;
import java.sql.Connection;
import java.sql.DriverManager;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import jimm.datavision.layout.HTMLLE;
import jimm.datavision.Parameter;
import jimm.datavision.Report;

public class DVWebappServlet extends HttpServlet {

  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    // Simple exception handling, which is to say virtually none!  We'll just
    // display the exception and be done with it.
    try {

      // First thing we'll do is read in the report XML.  We'll do this by
      // getting a reference to it through the servlet.  Just read it in to a
      // StringBuffer for later.
      ServletContext sc = getServletContext();
      InputStream is = sc.getResourceAsStream("dvtest.xml");

      // Next we'll get a connection to our database.  I made them servlet init
      // parameters so you can play with new reports and database easy.
      // They mimic the parameters you enter in the DataVision report designer.
      String jdbcDriverClassName = getInitParameter("JDBCDriverClassName");
      String connectionInfo = getInitParameter("ConnectionInfo");
      String userName = getInitParameter("UserName");
      String password = getInitParameter("Password");
      Class.forName(jdbcDriverClassName);
      Connection conn =
        DriverManager.getConnection(connectionInfo, userName, password);

      // Now we'll set up DataVision.  We instantiate a report instance and
      // hand it the database connection and the XML, and tell it we want HTML
      // as our output and pipe it to the servlets response output stream.
      Report report = new Report();
      report.setDatabaseConnection(conn);
      report.read(new org.xml.sax.InputSource(is));
      report.setLayoutEngine(new HTMLLE(new PrintWriter(
        response.getOutputStream())));

      // Finally, run the report!
      report.runReport();

      // Make sure we close that database connection!
      conn.close();

    } catch (Exception e) {
      System.out.println("Exception in DVWebappServlet: " + e);
    }

  } // End doPost().

} // End class.
```

I’m going to skip over the first two steps in the code, namely the reading of the report XML file and the getting of the connection to the database. This is fairly pedestrian code that you should be able walk through on your own easily enough (get it? Pedestrian? Walk through?!?). The portion of the code dealing with DataVision in figure 10 is what we’re really interested in anyway:

Figure 10.

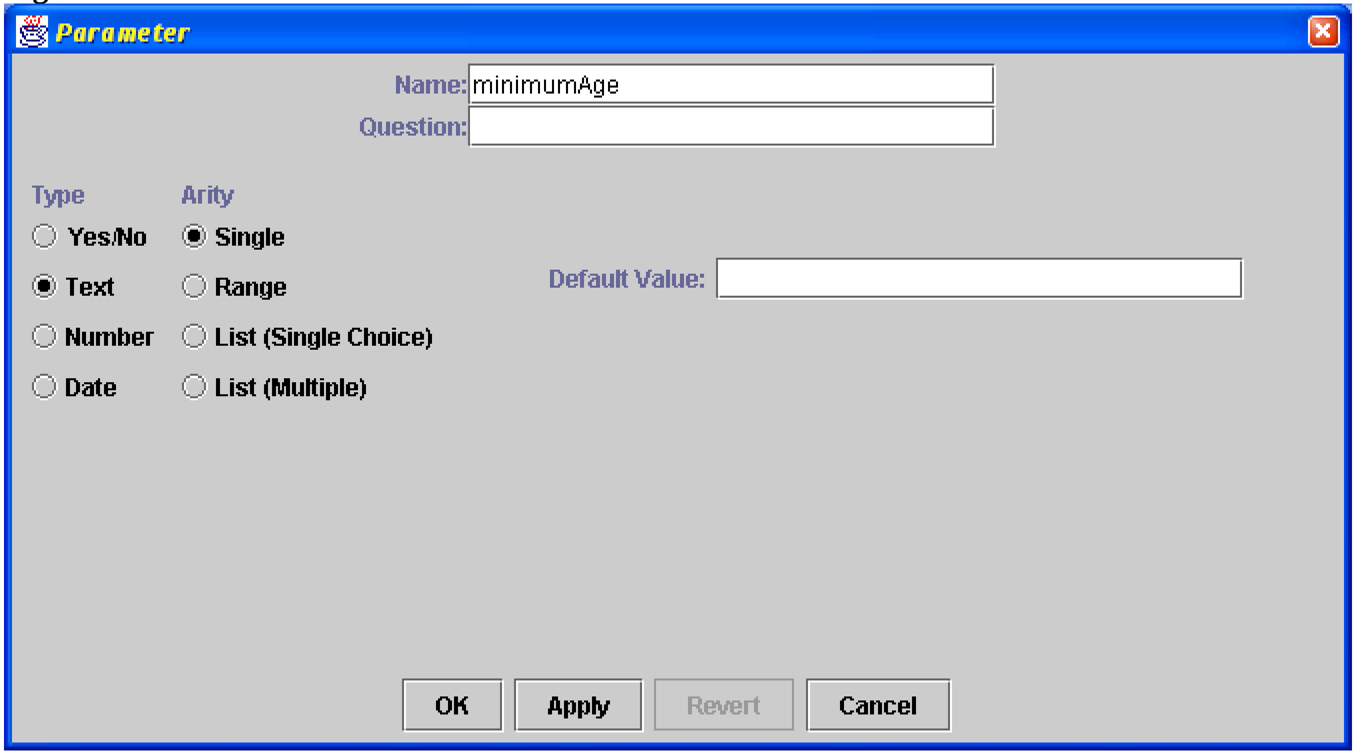
```
      // Now we'll set up DataVision.  We instantiate a report instance and
      // hand it the database connection and the XML, and tell it we want HTML
      // as our output and pipe it to the servlets response output stream.
      Report report = new Report();
      report.setDatabaseConnection(conn);
      report.read(new org.xml.sax.InputSource(is));
      report.setLayoutEngine(new HTMLLE(new PrintWriter(
        response.getOutputStream())));
```

The first thing we do is instantiate a DataVision Report object. The minimum we could do is exactly what this code does: hand it the connection to the database and the report XML. Next we tell it what layout engine we want to use, in this case the HTML layout engine. We have to pass that layout engine a PrintWriter, and in this case it's a PrintWriter wrapped around the servlet responses' output stream. Note that other layout engines require different inputs to their constructors. For example, the PDF layout engine requires an OutputStream object. So don't assume a PrintWriter is what you need! Once that is all done, all that's left to do is tell the report to run itself, and it is returned to the client. Simple!

One other thing I'd like to touch on is report parameters. Many times, a report requires variable pieces of information when they are run. For instance, we might want to show only those people from our table who are more than 20 years old. This is done with a parameter.

You will first need to add the parameter to the report. To do so, open up the report designer and open the report. Click the Insert menu, and then the Parameter Field option. On the Fields dialog, click Field and then New Parameter. Name it **minimumAge** and click Ok. A new dialog will appear:

Figure 11.



Parameter

Name: minimumAge

Question:

Type: Yes/No [✓ Text] Number Date

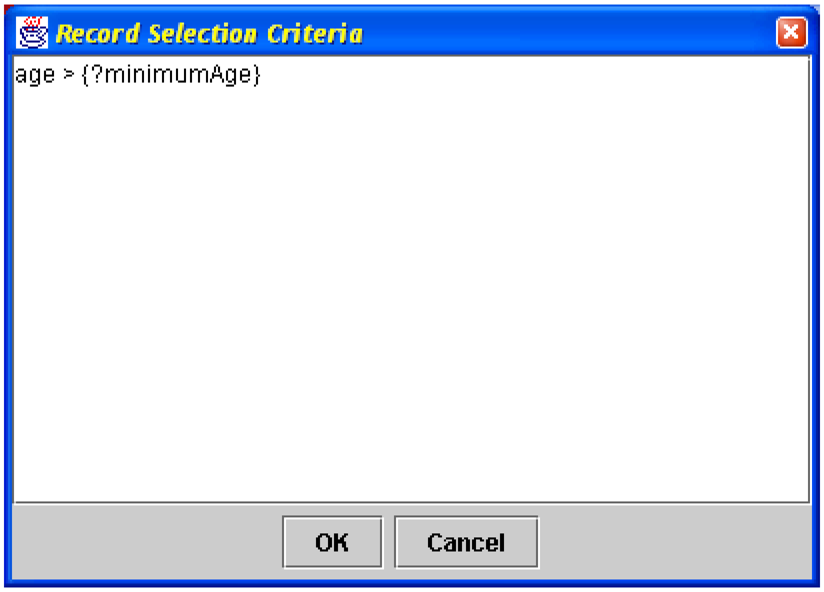
Arity: [✓ Single] Range List (Single Choice) List (Multiple)

Default Value:

Set the Type to Number and click Apply, then Ok. The report now has the parameter in it.

The last step is to make the report use this parameter. To do so, click the Select Records from the Report menu. Be sure you leave the Fields dialog up. You will now see a simple dialog:

Figure 12.



```
age > {?minimumAge}
```

What you are essentially looking at is the WHERE clause of a SQL query. So, if you imagine what the query looks like at this point, it is something like this, where SCHEMA is whatever schema you created the table in:

select SCHEMA.DATAVISION.FIRSTNAME, SCHEMA.DATAVISION.LASTNAME, SCHEMA.DATAVISION.AGE from SCHEMA.DATAVISION

(You can in fact see the query DataVision will run at any time by selecting SQL Query Test from the Database menu).

Now, type in **age >** in the Record Selection Criteria dialog, and drag the minimumAge formula from the Fields dialog after it. Your window's contents should now look like what you see in Figure 12. Click Ok and run the report. You will be prompted for an age and the report should run and only return records with an age greater than your entry. Be sure to save your report at this point!

That was easy, so how do we now use that in our webapp? Simple! Add the code in figure 13 to the DVWebAppServlet right before the line report.runReport();

Figure 13.

```
        Parameter p = report.findParameterByName("minimumAge");
        p.setValue(0, “20”);
```

Replace 20 with whatever age you'd like. Recompile the application and start up your server and access the test page. You should get the same output as you do in the report designer for whatever age you enter for the parameter value. That's all there is to it! Note that the second parameter of the setValue() method accepts an Object. It will be converted to the appropriate type for the parameter you are setting automatically.

I hope that this article, and the accompanying simple webapp example, has shown you the power and ease of DataVision, especially when it comes to web applications. I know of people who have integrated DataVision in Swing applications as well, so the flexibility you get with this package is clearly excellent. I have had virtually no problems using DataVision, and the few I have had have been quickly and easily solved by Mr. Menard and other users on the DataVision mailing list (sign up from the DataVision home page!).

DataVision is one of those fantastic open-source projects which is a real joy to use. It is simple, powerful and does exactly what it is advertised to do. I highly suggest checking it out if you have reporting requirements in your web applications, or any other type of application for that matter.

See you on the mailing list!

About the author
Frank W. Zammetti is a Web Architect Specialist for a leading worldwide financial institution during the day and a PocketPC developer at night. He is the Founder and Chief Software Architect of Etherient. He has over 10 years of professional development experience, and nearly 15 more of "amateur" experience, and he has been developing web-based applications (Intranet applications mostly) almost exclusively for nearly 7 years. Frank holds numerous certifications including SCJP, MCSD, CNA, i-Net+, A+, CIW, MCP and numerous BrainBench certifications. He is a contributor to a number of open-source projects, including DataVision and PocketFrog, as well as having started two, Java Web Parts (http://javawebparts.sourceforge.net) and the Struts Web Services Enablement Project (http://sourceforge.net/projects/strutsws). Frank's resume is available online at http://www.zammetti.com/resume.

Sample webapp: http://www.zammetti.com/articles/rftwwdv /dvwebapp.zip

PDF copy of this article: http://www.zammetti.com/articles/rftwwdv /rftwwdv.pdf

Microsoft Word copy of this article: http://www.zammetti.com/articles/rftwwdv/rftwwdv.doc