Web Application Development on a Linux System With a DB2 Database

By
Alan Andrea

Linux, for a long time now, has been a robust and solid platform for deploying and developing complex applications. Furthermore, it comes at a relatively cheap, or free, cost which encourages even a small business to be able to afford their own Unix environment. It can run on even low-end Intel 386-based computers. Consequentially, many companies have begun to port their popular software applications to the Linux operating system. Such companies include Oracle, Mathematica, Corel, and IBM, just to name a few. Furthermore, they are providing full support and licensing for their products on these Linux environments. Combining this with the power of such technologies as DSL or a Cable modem, very small businesses and consumers can now afford to run their own dedicated application servers or web servers on their very own boxes at home.

With this in mind, my paper will discuss developing an entire web-enabled application utilizing Java servlets and jsp's, on a Redhat 7.1 Linux environment, using the Tomcat Servlet Engine and IBM's DB2 as the database. I will begin first by discussing setting up DB2 on a Linux-based system and some of the essential administrative commands needed to quickly get a DB2 instance up and running and to create a sample schema. I will then discuss developing a sample application against DB2. At the end I will include my entire source code, which will serve as a one stop shopping for many of the essential concepts you need to start building even the most complex of web database applications.

Part I: Setting up Db2 on Linux and Essential Db2 Commands

Installation

I will now discuss the setup and administration of DB2 on Redhat 7.1. For this discussion I am assuming version 7.2 of DB2 running on Redhat. You will first need to acquire the DB2 installation CD-ROM or you may download the 7.2 distribution from the IBM website. www.ibm.com
You can obtain the personal DB2 for free from IBM's website. However, this will not allow connections to the DB2 instance from a remote host OTHER THAN jdbc or odbc connections. However, with the workgroup addition, you can connect from a standard DB2 client software on a remote computer to your DB2 server. Also you have a commercial production license. The workgroup addition costs about $800 dollars for Redhat Linux.

You will need to mount your CDROM device, usually with command like: mount /dev/cdrom /cdrom you then go into the /cdrom/db2 directory and run the setup program called db2setup. Also, if you have not installed it already, you will need to install pdksh-5.2.14-12.i386.rpm, which is available on your Red-hat CDROM. The following are the steps that you should follow, and some considerations, when setting up DB2:

1) Keep in mind the location on the filesystem you want to store the db2 system files to (make sure there is enough space!!!!)
2) What ids and group names you want to use to create the db2 instance user and db2 appserver user with.

3) If you want to have db2setup install, an initial db2 database instance, or just the basic server software (if you decide not to create a database then you will need to create one later on either by issuing the command in the db2 command line utility or in the Db2 control center gui.
4) Do you want to install dataWarehousing support, which will facilitate analytical analysis and multidimensional analysis of your data. Not needed if you are only implementing a transactional environment as opposed to an OLAP environment.

To install DB2 on Red-hat linux you will issue the command db2setup. This is a text based menu-driven utility which will guide you through the installation of DB2. Note, if you are NOT FAMILIAR with db2 and the command line processor, you will want to have the db2setup install the Control Center. This is a Jjava gui application which is more user friendly and builds commands for you. Also, if you are planning on running jdbc connections into db2, you will want to install the Application Development Client The application development client also has the procedure builder which will help with developing db2 stored procedures which are programs that run on the db2 database.
After you are done with the db2setup installer, you will want to set a few environmental variables in your .bash_profile script for your db2 instance user (eg db2inst1):

I recommend the following environmental variables be set in your .profile script for the db2 instance user
---
```
export LD_ASSUME_KERNEL=2.2.5
db2jstrt 6790
db2admin start
db2start
```

The LD_ASSUME_KERNEL is important if you want to run the DB2 Control Center since it requires JDK118 and Redhat 7.1. By default it turns on support for the floating point stack which causes JDK118 programs to fail including the Control Center.

If you have had any issues up to this point with installing DB2, you can find plenty of documentation on the internet at the following urls:
http://www-3.ibm.com/software/data/db2/linux/
http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v7pubs.d2w/en_main

To log into your instance, log in as the db2 instance user: eg db2inst1 and then run db2 from the command line to get into the clp or type db2cc to go into the control center gui.

With the above steps and notes in hand you should now be well on your way to having a db2 instance up and running. Next, I will talk about commands you will need and the main db2 tool programs included in the db2 distribution which you will need to conduct such tasks as creating users, tablespaces, tables, indexes, stored procedures as well as examining the statistics and performance of your db2 server.

DB2 Administration Utilities and Development tools

DB2 includes numerous utilities for allowing the user to run commands, create commands through wizards, and develop stored procedures or administrating the DB2 Database. The first tool is the CLP (command line processor) which is invoked by typing in db2 at the command prompt in Linux. This will take you into a shell which will allow you to log into your database and type in commands that you wish to run. This is the simplest and most difficult tool to use since it does not provide wizards, and only limited help on entering commands. The following Table lists some useful commands you would commonly use in the clp:

DB2 Command Reference

Command Name	Description	Example
Connect to	Connect to a db2 database	Connect to saleswarehouse;
Force applications all	Terminate all apps connected to this instance	Force applications all
Describe Select * from syscat.tables	Describes the columns in a table, their data types and Lengths. Syscat.tables is A system table that gives You lots of information about All the tables in your database.	
Terminate	Terminates the current session	Terminate
List database directory	Lists all of the databases And locations of files For this database and some Other pertinent info	List database directory
Get instance	Retrieves the current db2 instance that you are using.	Get instance
RESTART DATABASE <DBNAME>	Restarts a database after An ABEND	Restart database salesdb
List indoubt transactions < with prompting>	Lists transactions that Have not fully commited And which the transaction manager Left in an indoubt state. When the <with prompting> Is specified then you can Interactively complete the Commit of these transactions Or roll them back. Or forget it.	List indoubt transactions with prompting
Reorganize table	Reorgs rows in a table which Have become fragmented In order to speed up performance.	Reorg table eastsalestbspc.sales
Runstats	Updates statistics on a table To improve query access Time agaisnt the table. Should be called after many Inserts,updates or after reorg.	Runstats on table sales
List applications	Shows all currently running programs connected to this Db2 instance.	List applications
Catalog < database name> as <alias name> authentication <authentication type> with “comment string”	Allows you to specify An alias name for a Database that you have created in your instance.	Catalog salesdb as corporate_sales Authentication SERVER With “this is the sales database”
Load	Load data into a db2 table From a file. See more Detailed reference At url: http://www.student.math.uwaterloo.ca/~cs448/db2_doc/html/db2n0/frame3.htm#db2n067	

Select tabname from syscat.tables Where tabschema = 'SYSCAT'	Get a listing of system tables In the schema SYSCAT. Useful for showing the important SYSTEM CATALOG TABLES of DB2.	

The next utility I will discuss is the Control Center. This utility requires jre 118 to be installed on your local Linux environment. Furthermore, as I have mentioned above, you will need to set the environmental variable LD_ASSUME_KERNEL to 2.2.5 in order to use the APP!!!! The Control Center is a gui which is used for Creating Instances, tablespaces tables, running dml DDL's and basic select queries. This is the tool that you would want to start out with if you are new to db2 (especially since it generates the commands for you and you can see the DB2 SQL and DDL that is created). With this said, this is a good learning tool. To run this utility, just type in db2cc at the unix prompt. In this utility you can you will see a listing of hosts and below that you can drill into instances and databases and administrate these databases. It is here where you can also set up performance monitors to monitor your db2 instance and get key statistics as to the activity that is occurring on your database. Furthermore, you can see long running queries, or see how many users are connected to your system. To use the performance monitor, right click on your instance name, highlight performance monitor and click start monitor. It will then bring you into the utility for performance monitoring.

On the top menu bar of the Control Center, you can access other useful utilities such as the Command Center Where you can create sql statements and run them (or have the SQL ASSIST wizard create and run the query for you if you are new to sql). When in this utility, you must first login to your database by issuing the command connect to <dbname> using; then <control> <enter> and it will bring up a login screen to allow you to login. Anytime you wish to run your command (sql), just end it with <control><enter> To see the results of your query you can change into the Query Results window. To see an explain plan, use the Access Plan tab at the top. This will show useful information about whether or not your query is using an index to speed up its performance and what index(es) are being used (OR NOT USED)

From the Control Center, you can also access the procedure builder which will allow you to create sql stored procedures to run on your db2 database. This tool also has a debugger built into it to allow you to step through your procedure and add breakpoints in your code for the debugger to stop at. Procedure builder is therefore essentially the tool that allows you to develop procedural sql programs which run on the db2 database and interface with it via embedded sql and cursors. Later I will give a full program written in db2 plsql which will return a reference cursor back to a java Bean to facilitate my sample application.

Finally, in addition to the tools I have mentioned above, there are a number of Unix command line Db2 programs which one can call to perform many types important db2 functions. A few of these utilities include The following: db2batch: allows you to read sql statements from a file and run them in batch mode. Db2advis: recommends indexes based upon queries that have been run on the current database. Db2expln shows an explain plan for static sql in stored packages on the database, db2start: starts a db2 instance, db2stop: stops the db2 instance.

Part II: Sample Java Web Application running against DB2

In this section, I will now build a complete working web enabled Java application which runs against a Db2 database. This application is a search engine against a db2 database table of medical sites. The idea is

to enter in a medical topic that you are looking for and retrieve a list of sites that match the words you type in. I will create one table which will hold the title of the site, an abstract of what the site is about and a string of the URL of that site. This application will be accomplished in one screen in which a user will type in what he is looking for in a textfield at the top and I will then load the results of what is found into an iframe at the bottom. This iframe will be built from a jsp, which will display results from a bean which calls a sql stored procedure in db2 to obtain a resultset of the search results.

I will first discuss creating the necessary db2 tablespace and schema to store this medical search database.

A) My tablespace will be called MEDASPACE which I will create this using the following command:

```
CREATE REGULAR TABLESPACE MEDASPACE PAGESIZE 4 K
MANAGED BY SYSTEM
USING ('/data/medaspace/medaspc')
EXTENTSIZE 16
OVERHEAD 11.67
PREFETCHSIZE 16
TRANSFERRATE 0.31
BUFFERPOOL IBMDEFAULTBP
COMMENT ON TABLESPACE MEDASPACE IS 'This is my tablespace for medical searching'
```

B) I will now add a user called MEDAUSER whom will have rights to access this tablespace And create and read from tables in this schema:

```
GRANT
DBADM,CREATETAB,BINDADD,CONNECT,CREATE_NOT_FENCED,IMPLICIT_SCHEMA,LOAD
ON DATABASE TO USER MEDAUSER
GRANT USE OF TABLESPACE MEDASPACE TO USER MEDAUSER WITH GRANT OPTION
```

C) In order to facilitate storing the listing of medical site urls I will create the following table in this Schema:

```
Create table medBASE
(
 medID      integer,
title       varchar(50),
abstract    varchar(400),
url          varchar(75)
);
```

D) Now lets insert some rows into this table so that we have some example data:

```
Insert into medBASE values ( 1001, 'Hernia Operations', 'This is usefull info about hernias',
'http://www.hernia.org');

Insert into medBASE values ( 1002, 'Genetic Healing', 'This is useful info about genetics for healing',
'http://www3.mdanderson.org/depts/genetherapy/');
```

At this point in time, I now have my schema created with the one table that I will require for this application. I am now ready to discuss building the actual web application. To do this I used Borland Jbuilder version 6. I utilized servlets, jsp's and java beans to develop my application. I first call a main servlet which I called, srchScreen. This implements the doPost method and I simply get the servlet context and do a requestDispather.forward to call my

main jsp to display the main search screen. The jsp is called: medaMain.jsp. In this jsp I display a form field at the top and an ifra me below to display the search results. As a user types data into the text field in the form at top, I run javascript to grab the value and then load into the iframe another servlet of which I send the contents of the form field to this servlet as its o nly parameter. This second servlet is called: srchEngine. The servlet srchEngine extends another class that I wrote called baseserv which is a servlet class that implements the do post method and sets a session bean which I called basebean which actually handles connecting to by db2 instance. Secondary to this, I instantiate another bean, called, medaBean, which is where I call a db2 sql stored procedure (with the search string parameter) and obtain a result set from a reference cursor which I will use later on to display the search screen results. Finally, I call my jsp: srchTab in which I use the bean medaBean and the resultset obtained there to traverse the rows of the cursor and display the results in the iframe window. The complete program is listed below.

In conclusion, Linux and DB2 combined offer a very cost effective and powerful way to develop complex applications in a low budget environment. It is well suited for powerful and robust applications running on a cost constrained b udget. With the above ideas in hand You should now be well on your way toward understanding how to go about the technical aspects of setting up and developing your own DB2 applications on Linux.

Source Code for my Medical Search Engine Sample Application:

I) Bean Classes: Here are the two beans that I use. baseBean is used for holding the connection to the DB2 Database. The other bean class is medaBean which is used for calling the db2 stored procedure via the callablestatement to obtain a resultset from db2 of the rows which meet the search conditions the user enters.

```
//-----------------------------------------------------------------------------------------------------
//              Java Class    baseBean:  used for holding and initiating connection to DB2
//-----------------------------------------------------------------------------------------------------

package medasearch;
import java.sql.*;

public class baseBean
{
 private Connection conn=null;

public Connection getConn()
{
  if ( conn == null )
    setConn();
 return conn;
}
public void setConn()
{
 try
  {
   Class.forName( "COM.ibm.db2.jdbc.net.DB2Driver");
```

```
 if ( conn == null )
  conn = DriverManager.getConnection( "jdbc:db2://supernova.webintel-systems.com:6790/rentals", "db2inst1", "mentat01");
 }
  catch ( Exception sqle ) { } ;
}  // end setCon

}
//-------------------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------------------------
```

```
//-------------------------------------------------------------------------------------------------------
//                Java Class     medaBean:   used for calling DB2 stored procedure to
//                  obtain a resultset of the   rows which match what you are searching for
//-------------------------------------------------------------------------------------------------------
```

```
public class medaBean
{
 public ResultSet rset;

public void doSearch( Connection conn, String srchCriteria )
{
  // Open up a connection to db2 database and retreive the

  System.out.println(" sc " + srchCriteria );
 if ( conn == null )
  System.out.println(" conn is null ");

try
 {
  CallableStatement cstmt =  conn.prepareCall ( " call   DB2INST1.getSearch( ?)  " );
  cstmt.setString(1, srchCriteria);
  rset =  (ResultSet) cstmt.executeQuery();

 }
  catch ( Exception ex) { System.out.println(" bean error " + ex.getMessage() ); }

}
//-------------------------------------------------------------------------------------------------------
```

II) Servlet Classes: Here I show my three servlets. First, baseserv which is the base servlet class which I use to instantiate the basebean. All servlet classes extend this class so that I can run common routines such as initiating database connections without having to recode the logic for every servlet. This is a great time saving step. Second, I have my servlet, srchScreen.class which is the initial servlet that calls the main display jsp. Finally, I have my servlet called srchEngine which calls my bean medaBean to get The results from db2 and receives as a parameter the text of what is currently being searched for.

```
//-------------------------------------------------------------------------------------------------------
//                Java Class     baseserv:   used for instantiating the basebean to the
//                  current session so that we can hold the base bean and all of its connection
//                 info in the Servlets session context.
//-------------------------------------------------------------------------------------------------------

package medasearch;
```

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class baseserv  extends HttpServlet
{
baseBean  BaseBean = null;
Connection conn = null;
//-------------------------------------------------------------------------------------------------
public void doPost(HttpServletRequest req, HttpServletResponse res)  throws ServletException, IOException
 {
   HttpSession session = req.getSession (true);

//   Allocate a new Bean if Needed to handle the connections to the database:    Reuse already opened
//    Connections and open once per session and not once per transaction

 if ((baseBean)session.getAttribute("BaseBean") == null)
 {
  BaseBean = new baseBean();
  session.setAttribute("BaseBean", BaseBean);

 } else
   BaseBean = (baseBean)session.getAttribute("BaseBean");
   conn = BaseBean.getConn();
 }
}


//-------------------------------------------------------------------------------------------------------
//                Java Class     srchScreen:   which is first servlet called which calls
//                    my jsp to display the search screen
//-------------------------------------------------------------------------------------------------------
package medasearch;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

/**
 * <p>Title: MedaSearch Medical Searching app</p>
 * <p>Description: Demonstration Application against DB2</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Co mpany: </p>
 * @author Alan T. Andrea
 * @version 1.0
 */

public class srchScreen extends HttpServlet {

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

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

 RequestDispatcher rd;
// /wisapps/jsp/medasearch/
 rd =  getServletContext().getRequestDispatcher("/jsp/medasearch/medaMain.jsp");
 rd.forward(request,response);
```

```
}

  //Clean up resources
  public void destroy() {
  }
}

//-----------------------------------------------------------------------------------------------------------
//                Java Class     srchEng:  Servlet which is called by  medaMain.jsp
//            to  obtain the results of what user is searching for
//-----------------------------------------------------------------------------------------------------------
package medasearch;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

public class srchEng extends baseserv
{

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

 {
 doPost( request, response );
 }

//-----------------------------------------------------------------------------------------------------------
public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
{
  String  srchCriteria = new String();
  HttpSession session = request.getSession (true);

  super.doPost(request,response);

 try
  {

    try
     {
       srchCriteria =  request.getParameter("criteria");
     }
      catch ( Exception ex ) { srchCriteria = ""; }
    medaBean mb = (medaBean) session.getAttribute("MedaBean");


 if ( mb  == null)
  {
    mb = new  medaBean();
    mb.doSearch(conn, srchCriteria);
    session.setAttribute("MedaBean",mb);
  }
 else
   mb.doSearch(conn, srchCriteria);


 RequestDispatcher rd;
 rd =  getServletContext().getRequestDispatcher("/jsp/medasearch/srchTab.jsp");


 rd.forward(request,response);


  } catch ( Exception ex ) { System.out.println( " Error " + ex.getMessage()  ); }

}
```

```
//-----------------------------------------------------------------------------------------------------

}
```

III) JSP Java Server Page classes: Here I have two jsps: 1 for displaying the overall search Screen and another for displaying the results of the search.

```
//-----------------------------------------------------------------------------------------------------------------------------------
//          medaMain.jsp                main jsp to display overall screen:
// ----------------------------------------------------------------------------------------------------------------------------------


<%@ page errorPage="medaMainErrorPage.jsp" %>
<html>
<head>
<title>
medaMain
</title>

<SCRIPT LANGUAGE=JAVASCRIPT>

function srchServ()
{
 this.srchFrame.location="http://supernova.webintel-
systems.com:8080/wisapps/servlet/medasearch.srchEng?criteria="+document.srcForm.elements[0].value;

}


function doSearch()
{
  if (  event.keyCode != 17 && event.keyCode != 16 && event.keyCode != 18 )
    timerID  = setTimeout("srchServ()", 1500);
}

</SCRIPT>

</head>


<body bgcolor="#ffffc0">

<center> <font color=blue size=6> MedaSearch : Medical Information Search Engine </font>
<hr>
<form name=srcForm method="post">
<table border>
<tr>
<td> Enter Search Criteria</td>
<td>  <input name="srchVal" onKeyUp="doSearch()" >  </td>
</tr>
</table>
</form>

<hr>
</center>

<iframe name="srchFrame" src="http://" width=1000 height=700 scrolling=yes >" src="http://" width=1000 height=300 >  </iframe>

</body>
</html>
```

```
//-------------------------------------------------------------------------------------------------------------------------
//     srchTab.jsp     jsp to display results of this search
// ------------------------------------------------------------------------------------------------------------------------


<jsp:useBean id="MedaBean" scope="session" class="medasearch.medaBean"   />

<%@ page errorPage="medaMainErrorPage.jsp"
   import="java.sql.*"
%>

<html>
<head>
<title>
srchTab
</title>
</head>
<body bgcolor="#c0c0c0">

<center>
<table border>
<tr>
<th>Title</th>
<th>Abstract</th>
<th>URL</th>


<%

 ResultSet  rset =   MedaBean.rset;

 int rcount = 0;

 while ( rset.next() )
  {
   %>
    <tr>
     <td><%=rset.getString("TITLE")%></td>
     <td><%=rset .getString("ABSTRACT")%></td>
     <td><A Href="<%=rset.getString("URL")%>"> Click to go to Site </A> </td>
   <%
  ++rcount;
 }
 if ( rcount == 0 )
  {
   %>
    <tr>
     <td colspan=3><font size=3 color=red> Nothing Found for this criteria </font> </td>
     <%

    rset.close();
   }

%>


</table>
</center>

</body>
</html>
```