JNDIDatasource

(I'd test this solution with Websphere and Jetty)

- Declare the Datasource in your container and provide the jar.
    - Under Websphere (I'm newbee), to define the user and password to use to connect to the Database, add new CustomProperties "user" and "password" (work for MS SQLServer). If you do this after link with a J2C Authorization , it's to late delete and create a new Datasource.
- In the cocoon.xconf file, change your datasource from the <jdbc> style to the <j2ee> style.

```
<!-- use lookup-name instead of dbname tag under j2ee, better control -->
<!-- for Jetty -->
<j2ee name="mydb">
  <lookup-name>java:comp/env/jdbc/mydb</lookup-name>
</j2ee>
<!-- for Websphere -->
<j2ee name="mydb">
  <lookup-name>jdbc/mydb</lookup-name>
  <!-- If you use ParanoidCocoon (I've got some Classloader issue)-->
  <initial-context-factory>com.ibm.websphere.naming.WsnInitialContextFactory</initial-context-factory>
  <provider-url>corbaloc:rir:/NameServiceServerRoot</provider-url>
</j2ee>
```

- at the bottom of the web.xml file for your cocoon instance, add this;

```
 <resource-ref>
     <res-ref-name>jdbc/mydb</res-ref-name>
     <res-type>javax.sql.DataSource</res-type>
     <res-auth>Container</res-auth>
 </resource-ref>
```

It goes just before the </web-app> line at the bottom. You'll need to change the "mydb" reference here also.

You could check the web.xml configuration with this Servlet (call http://host/app/servlet/my.toolkit.dbg.JNDIDataSourceServlet?resource=jdbc/mydb):

```
package my.toolkit.dbg;

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

/**
 * @author D.Bernard (dwayne@java-fan.com)
 */
public class JNDIDataSourceServlet extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    DataSource ds = null;
    Connection con = null;
    PrintWriter out = resp.getWriter();
    resp.setContentType("text/plain");
    try {
      out.println("Looking up DataSource");
      InitialContext ctx = new javax.naming.InitialContext();
      ds = (DataSource) ctx.lookup(req.getParameter("resource"));
      out.println("Getting connection :");
      con = ds.getConnection();
      out.println(con);
      con.close();
    } catch (Exception e) {
      e.printStackTrace(out);
    }
    out.println("Done");
  }
}
```

If you use ParanoidCocoonServlet, you need to define <initial-context-factory> and <provider-url>. You could find the values with the following Servlet. Use the value of "java.naming.factory.initial" to define <initial-context-factory>, and the value of java.naming.provider.url to define <provider-url>.

```
package my.toolkit.dbg;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.naming.InitialContext;

/**
 * @author D.Bernard (dwayne@java-fan.com)
 */
public class RequestInfoSystem extends HttpServlet {

    ////////////////////////////////////////////////////////////////////////
    // CLASS
    ////////////////////////////////////////////////////////////////////////
    //------------------------------------------------------------------------
    //------------------------------------------------------------------------
    private static final void printTitle(PrintWriter out, String title) {
        out.println("----------------------------------------");
        out.println("---  " + title);
        out.println("----------------------------------------");
    }

    //------------------------------------------------------------------------
    //------------------------------------------------------------------------
    private static final void printData(PrintWriter out, String key, String[] values) {
        out.print("  " + key + " : ");
        for(int i = 0; i < values.length; i++) {
            out.print(values[i] + " ");
```

```
        }
        out.println();
    }

    //-------------------------------------------------------------------------
    //-------------------------------------------------------------------------
    private static final void printData(PrintWriter out, String key, Object value) {
        out.println(key + " : " + value);
    }

    //-------------------------------------------------------------------------
    //-------------------------------------------------------------------------
    private static final void printData(PrintWriter out, Map values) {
        if (values == null) {
            out.println("NULL");
        } else {
            for(Iterator it = values.entrySet().iterator(); it.hasNext();) {
                out.println(it.next());
            }
        }
    }

    ////////////////////////////////////////////////////////////////////////////
    // OBJECT
    ////////////////////////////////////////////////////////////////////////////

    //-------------------------------------------------------------------------
    //-------------------------------------------------------------------------
    public void service(HttpServletRequest request, HttpServletResponse response) throws IOException,
ServletException {
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        displaySystemProperties(out);
        displayDefaultInitialContext(out);
    }

    //-------------------------------------------------------------------------
    //-------------------------------------------------------------------------
    private void displaySystemProperties(PrintWriter out) {
        printTitle(out, "SystemProperties");
        printData(out, System.getProperties());
    }

    private void displayDefaultInitialContext(PrintWriter out) {
        printTitle(out, "InitialContext");
        try {
            printData(out, (new InitialContext()).getEnvironment());
        } catch(Exception exc) {
            exc.printStackTrace(out);
        }
    }
}
```