

# Snooping on your Webserver with the SuperSnoop Servlet

Techdoc TD101815

Techdocs are available from:  
<http://www.ibm.com/support/techdocs>

Revised: September 9, 2004

Authors:

Louis Wilen      IBM Washington Systems Center  
wilen@us.ibm.com

Lee-Win Tai      IBM Washington Systems Center  
tai@us.ibm.com

## Overview

SuperSnoop is a servlet that displays numerous webserver parameter and configuration values on the invoker's browser. It is useful for performing problem isolation and to help understand how various webserver configuration settings affect operation.

SuperSnoop contains code that requires WebAS 5.0 libraries, but it can be recompiled in a non-WebAS 5.0 for z/OS environment by commenting out indicated sections of code.

The SuperSnoopProj.ear file that is included with this Techdoc will run without changes on most application servers, as the servlet checks the platform on which it is being run and executes WebAS 5.0-specific code only when run under WebAS 5.0.

SuperSnoop is based on the classic SessionSnoop servlet that is described in the O'Reilly book, [Java Servlet Programming](#), by Jason Hunter.

As supplied in the included .jar file, the Context-Root for SuperSnoop is SuperSnoopWeb and the Servlet Mapping is SuperSnoop. Therefore, a typical URL for invoking SuperSnoop is:

**<http://myhost.org/SuperSnoopWeb/SuperSnoop>**

Substitute your actual hostname (and port, if necessary) for myhost.org.

*The SuperSnoop servlet has not been subjected to any formal testing and is supplied without any warranty.*

## Source Code Listing

```
// This is SuperSnoop, a servlet that displays information about a server and
// the environment in which a servlet is running.
// Some of the code in this servlet requires jar files that are supplied in
// the IBM WebSphere Application Server and IBM WebSphere Application Server for
// z/OS libraries. The code that requires WebSphere Application Server
// libraries is identified by comments. However, after being compiled, this
// servlet will run in both a WebAS and non-WebAS server.

import java.io.IOException;
import java.io.PrintWriter;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.Date;
import java.util.Enumeration;
import java.util.Properties;

import javax.naming.Binding;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingEnumeration;
import javax.naming.NamingException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class SuperSnoop extends HttpServlet {

    private static final String ssver = "Version 2004-07-28 02:15:00 PM";
    private static final int K = 1024;

    //Initialize global variables
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    // WebSphere Only
    // This is applicable only to IBM WebSphere Application Server.
    private String getAppServerName() {
        com.ibm.websphere.management.AdminService adminService =
        com.ibm.websphere.management.AdminServiceFactory.getAdminService();
        String appServerName =
            adminService.getCellName()
                + "/"
                + adminService.getNodeName()
                + "/"
                + adminService.getProcessName();
        adminService = null;
        return appServerName;
    }
    //
    // End WebSphere Only

    private String getClassInfo() {
        StringBuffer sb = new StringBuffer();
        sb.append("<h3>Class and Classloader Information</h3>");
        ClassLoader classloader = SuperSnoop.class.getClassLoader();
        sb.append("<h4>Classloader:</h4>" + classloader);

        Method[] m = SuperSnoop.class.getDeclaredMethods();
        if (m.length == 0) {
            sb.append("<br>There are no declared methods in this class.<br>");
        }
    }
}
```

```

    } else {
        sb.append("<h4>Declared Methods:</h4>");
        for (int i = 0; i < m.length; i++) {
            sb.append(m[i] + "<br>");
        }
    }
    Field[] f = SuperSnoop.class.getFields();
    if (f.length == 0) {
        sb.append("<br>There are no fields in this class.<br>");
    } else {
        sb.append("<h4>Fields:</h4>");
        for (int i = 0; i < f.length; i++) {
            sb.append(f[i] + "<br>");
        }
    }
    sb.append(
        "<br>Package to which this class belongs: "
        + SuperSnoop.class.getPackage()
        + "<br>");
    Object[] o = SuperSnoop.class.getSigners();
    if (o == null) {
        sb.append("<br>There are no signers for this class.<br>");
    } else {
        sb.append("<h4>Signers:</h4>");
        String[] s = (String[]) o;
        for (int i = 0; i < s.length; i++) {
            sb.append(s[i] + "<br>");
        }
    }
    return new String(sb);
}

//Process the HTTP Get request
public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    System.out.println("SuperSnoop running");
    PrintWriter out = new PrintWriter(res.getWriter());
    res.setContentType("text/html");

    HttpSession session = req.getSession(true);

    Integer count = (Integer) session.getAttribute("snoop.count");
    if (count == null) {
        session.setMaxInactiveInterval(-1); // Set session timeout
        count = new Integer(1);
    } else {
        count = new Integer(count.intValue() + 1);
    }
    session.setAttribute("snoop.count", count);

    out.println(
        "<HTML><HEAD><TITLE>SuperSnoop " + ssver + "</TITLE></HEAD>");
    out.println("<BODY><H1>SuperSnoop " + ssver + "</H1>");
    out.println(
        "You've visited this page "
        + count
        + ((count.intValue() == 1) ? " time." : " times."));
    out.println("<br><br>");
}

```

```

out.println(
    "Requested sessionID from URL: "
    + req.isRequestedSessionIdFromURL()
    + "<br>");
out.println(
    "Requested sessionID from cookie: "
    + req.isRequestedSessionIdFromCookie()
    + "<br>");
out.println(
    "Requested sessionID valid: "
    + req.isRequestedSessionIdValid()
    + "<br>");
out.println(
    "<a href="
    + res.encodeURL(req.getRequestURI())
    + ">Click to test session tracking via URL rewriting</a>");
out.println("<br><br>");

// WebSphere Only
// Applicable only to IBM WebSphere Application Server.
String st = (System.getProperty("java.vm.vendor"));

if (st.equals("IBM Corporation")) {
    out.println(
        "<p>This appserver name is: " + getAppServerName() + "</p>");
}
//
// End WebSphere Only

// WebSphere for z/OS Only
// Applicable only to IBM WebSphere Application Server for z/OS
if (System.getProperty("os.name").equals("z/OS")) {
    out.println(
        "Server display name: "
        + com
        .ibm
        .websphere
        .runtime
        .ServerName
        .getDisplayName()
        .trim());
    out.println(
        "<br>Server JSAB jobname: "
        +
        com.ibm.websphere.runtime.ServerName.getjsabjbnm().trim());
    out.println(
        "<br>Server JSAB jobid: "
        +
        com.ibm.websphere.runtime.ServerName.getjsabpref().trim());
    out.println("<br><br>");
}

// End WebSphere for z/OS Only

out.println(getClassInfo());

out.println("<h3>Memory Status</h3>");
out.println(logMemStats() + "<br><br>");

out.println("<H3>Saved Session Data:</H3>");
Enumeration anum = session.getAttributeNames();
while (anum.hasMoreElements()) {
    String aname = (String) anum.nextElement();
    if (aname != null) {
        Object avalue = session.getAttribute(aname);
    }
}

```

```

        out.println("Attribute: " + aname + ": " + avalue + "<br>");
    }
}

out.println("<H3>Session Information</H3>");
out.println(
    "Maximum inactive interval: "
    + session.getMaxInactiveInterval()
    + " seconds <BR>");
out.println("Session id: " + session.getId() + "<BR>");
out.println("New session: " + session.isNew() + "<BR>");
out.println("Creation time: " + session.getCreationTime() + "<BR>");
out.println("<I>(" + new Date(session.getCreationTime()) + ")</I><BR>");
out.println(
    "Last Access time: " + session.getLastAccessedTime() + "<BR>");
out.println(
    "<I>(" + new Date(session.getLastAccessedTime()) + ")</I><BR>");

out.println("<H3>Authentication Data</H3>");
out.println("User Name: " + req.getRemoteUser() + "<BR>");
out.println("Authorization type: " + req.getAuthType() + "<BR>");

String principalName =
    (req.getUserPrincipal() == null)
    ? null
    : req.getUserPrincipal().getName();
out.println("Principal Name = " + principalName + "<br>");
out.println(
    "isUserInRole('highrole') = "
    + req.isUserInRole("highrole")
    + "<br>");
out.println(
    "isUserInRole('lowrole') = "
    + req.isUserInRole("lowrole")
    + "<br>");

out.println("<H3>Client Machine Data</H3>");
out.println("Client address: " + req.getRemoteAddr() + "<BR>");
out.println("Client hostname: " + req.getRemoteHost() + "<BR>");

out.println("<H3>HTTP Request Header Data</H3>");
Enumeration enum = req.getHeaderNames();
while (enum.hasMoreElements()) {
    String name = (String) enum.nextElement();
    String value = req.getHeader(name);
    if (value != null) {
        out.println(name + ": " + value + "<br>");
    }
}

String pw = req.getHeader("authorization");
if (pw != null) {
    String pwe = pw.substring(6); // get userid and password
    sun.misc.BASE64Decoder dec = new sun.misc.BASE64Decoder();
    String userpassDecoded = new String(dec.decodeBuffer(pwe));
    out.println(
        "<br>Decoded authorization header = " + userpassDecoded);
    System.out.println("pass = " + userpassDecoded);
}

Properties props = System.getProperties();

out.println("<H3>System Properties</H3>");
for (enum = props.propertyNames(); enum.hasMoreElements();) {
    String key = (String) enum.nextElement();
    out.println(key + " = " + System.getProperty(key) + "<br>");
}

```

```

out.println("<H3>java:comp Context</H3>");
try {
    Context initCtx1 = new InitialContext();

    NamingEnumeration nenum1 = initCtx1.listBindings("java:comp");
    out.println(
        "<table border=3 cellpadding=3 bordercolor=blue>"
        + "<tr><td>Name</td> <td>Type</td> <td>Value</td></tr>");

    while (nenum1.hasMoreElements()) {
        Binding binding = (Binding) nenum1.next();
        out.println(
            "<td>"
                + binding.getName()
                + "</td><td>"
                + binding.getClassName()
                + "</td> <td>"
                + binding.getObject()
                + "</td> </tr>");
    }

    out.println("</table>");
} catch (NamingException e) {
    e.printStackTrace(out);
}

out.println("<H3>java:comp/env Context (if any)</H3>");
try {
    Context initCtx2 = new InitialContext();
    NamingEnumeration nenum2 = initCtx2.listBindings("java:comp/env");

    out.println(
        "<table border=3 cellpadding=3 bordercolor=blue>"
        + "<tr><td>Name</td> <td>Type</td> <td>Value</td></tr>");

    while (nenum2.hasMoreElements()) {
        Binding binding = (Binding) nenum2.next();
        out.println(
            "<td>"
                + binding.getName()
                + "</td><td>"
                + binding.getClassName()
                + "</td> <td>"
                + binding.getObject()
                + "</td> </tr>");
    }

    out.println("</table>");
} catch (NamingException e) {
    e.printStackTrace(out);
}
out.println("</BODY></HTML>");
out.close();
}

//Process the HTTP Post request
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    doGet(req, res);
}

private String logMemStats() {
    Runtime r = Runtime.getRuntime();
    StringBuffer s = new StringBuffer("Total memory: ");

```

```
s.append((r.totalMemory() / K));  
s.append("K, free memory: ");  
s.append((r.freeMemory() / K));  
s.append("K");  
  
return s.toString();  
    }  
}
```

## Sample Output

# SuperSnoop Version 2004-07-28 02:15:00 PM

You've visited this page 1 time.

Requested sessionID from URL: false  
Requested sessionID from cookie: false  
Requested sessionID valid: false  
[Click to test session tracking via URL rewriting](#)

This appserver name is: r7cella/r7nodea/r7sr01a

Server display name: r7sr01a  
Server JSAB jobname: R7SR01AS  
Server JSAB jobid: STC00053

## Class and Classloader Information

### Classloader:

com.ibm.ws.classloader.CompoundClassLoader@2e384a89 Local ClassPath:  
/WebSphere/r7cell/AppServer/installedApps/r7cella/SuperSnoop.ear/SuperSnoopWeb.war/WEB-INF/classes:/WebSphere/r7cell/AppServer/installedApps/r7cella/SuperSnoop.ear/SuperSnoopWeb.war:  
Delegation Mode: PARENT\_FIRST

### Declared Methods:

```
private java.lang.String SuperSnoop.logMemStats()
public void
SuperSnoop.doPost(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) throws
javax.servlet.ServletException, java.io.IOException
public void
SuperSnoop.doGet(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) throws
javax.servlet.ServletException, java.io.IOException
private java.lang.String SuperSnoop.getClassInfo()
private java.lang.String SuperSnoop.getAppServerName()
public void SuperSnoop.init(javax.servlet.ServletConfig) throws javax.servlet.ServletException
```

There are no fields in this class.

Package to which this class belongs: null

There are no signers for this class.

## Memory Status

Total memory: 262398K, free memory: 97119K

## Saved Session Data:

Attribute: snoop.count: 1

## Session Information

Maximum inactive interval: -1 seconds  
Session id: tefP4y2F8Ht52ioMitQLerd  
New session: true  
Creation time: 1092145763153  
(Tue Aug 10 13:49:23 GMT 2004)  
Last Access time: -1  
(Wed Dec 31 23:59:59 GMT 1969)

## Authentication Data

User Name: null  
Authorization type: null  
Principal Name = null  
isUserInRole('highrole') = false  
isUserInRole('lowrole') = false

## Client Machine Data

Client address: 192.168.137.51  
Client hostname: testhost.wcxe.ibm.com

## HTTP Request Header Data

accept: \*/\*  
accept-language: en-us  
accept-encoding: gzip, deflate  
user-agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)  
host: wschost.wscdom.ibm.com:7548  
connection: Keep-Alive  
cookie: w3saudid=d002000000001525310694411550000426799.0009529BCD; locale=es\_UY;  
PBC\_NLSP=en\_US; PISearchString=NULL; PISDate=NULL; PIEDate=NULL;  
IBMISP=43e3dcb1775d11d8986a9a28d74320f5-43e3dcb1775d11d8986a9a28d74320f5-5193e11214571  
17de2787b7f3246eccc; w3ibmProfile=200011271307240660404651785|gAME|897|FSD|null;  
msp=alreadyOffered; ipclInfo=cc%3Dus%3Blc%3Den%3Bac%3Dall

## System Properties

java.vendor = IBM Corporation  
com.ibm.ws390.jta.TransactionManager = com.ibm.ws390.tx.rrs.RRSCConnectorTxManager  
com.ibm.ejs.jts.processType = server  
os.name = z/OS  
sun.boot.class.path = /shared/java2/J1.3/lib/rt.jar:/shared/java2/J1.3/lib/i18n.jar:/shared/java2/J1.3/classes  
com.ibm.websphere.topology.cellname = r7cella  
java.vm.specification.vendor = Sun Microsystems Inc.  
java.runtime.version = 1.3.1  
security.use.localos.authorization = true  
user.name = RSASRU  
java.compiler = jitc  
protocol\_https\_port = 7549

```
server_specific_name = r7sr01a
was.install.root = /WebSphere/r7cell/AppServer
com.ibm.websphere.SpecLevel = 1.3
WAS_NODE = r7nodea
user.language = en
was.repository.temp = /WebSphere/r7cell/AppServer/config/temp/
sun.boot.library.path = /shared/java2/J1.3/bin
com.ibm.ws390.INTERNAL.plexname = WSLPLEX
ws390.servename = r7sr01a
java.version = 1.3.1
version = 2.3.10
user.timezone = GMT
server.root = /WebSphere/r7cell/AppServer
javax.rmi.CORBA.UtilClass = com.ibm.ws390.orb.Util
file.encoding.pkg = sun.io
com.ibm.db2.java.TMClassName = com.ibm.ws390.tx.rrs.RRSCConnectorTxManager
file.separator = /
java.specification.name = Java Platform API Specification
traceSettingsFile = config/cells/r7cella/nodes/r7nodea/servers/r7sr01a/trace.dat
java.class.version = 46.0
java.home = /shared/java2/J1.3
com.ibm.websphere.topology.servename = r7sr01a
java.vm.info = J2RE 1.3.1 IBM OS/390 Persistent Reusable VM build cm131s-20040117 (JIT enabled:
jitc)
com.ibm.CORBA.iiop.noLocalCopies = false
os.version = 01.04.00
java.awt.fonts =
path.separator = :
java.vm.version = 1.3.1
vendor-url = http://w3.xml.ibm.com/xsl/java
java.protocol.handler.pkgs = com.ibm.crypto.provider
com.ibm.websphere.topology.nodename = r7nodea
java.awt.printerjob = sun.awt.motif.PSPrinterJob
vendor = IBM Corporation
java.security.policy = /WebSphere/r7cell/AppServer/properties/server.policy
sun.io.unicode.encoding = UnicodeBig
was.repository.root = /WebSphere/r7cell/AppServer/config
java.assistive = ON
java.naming.factory.url.pkgs = com.ibm.ws.runtime
com.ibm.ws390.INTERNAL.DriverPath = null
user.home = /tmp
java.specification.vendor = Sun Microsystems Inc.
org.xml.sax.driver =
com.ibm.ws390.INTERNAL.CBConfig =
java.library.path =
/shared/java2/J1.3/bin:/shared/java2/J1.3/bin/classic:/WebSphere/r7cell/AppServer/lib:/usr/lpp/IBMRAC/lib
java.vendor.url = http://www.ibm.com/
SERVERNAME = R7SR01
java.vm.vendor = IBM Corporation
platform.notASCII = true
java.fullversion = J2RE 1.3.1 IBM OS/390 Persistent Reusable VM build cm131s-20040117 (JIT enabled:
jitc)
java.runtime.name = Java(TM) 2 Runtime Environment, Standard Edition
java.class.path =
/WebSphere/r7cell/AppServer/properties:/WebSphere/r7cell/AppServer/lib/bootstrap.jar:/WebSphere/r7cel
l/AppServer/lib/urlprotocols.jar:/WebSphere/r7cell/AppServer/lib/j2ee.jar:/WebSphere/r7cell/AppServer/lib/l
mproxy.jar
com.ibm.ws390.INTERNAL.ServerName = R7SR01
```

```

ws.ext.dirs =
/shared/java2/J1.3/lib:/WebSphere/r7cell/AppServer/classes:/WebSphere/r7cell/AppServer/lib:/WebSphere/r7cell/AppServer/lib/ext:/WebSphere/r7cell/AppServer:/WebSphere/r7cell/AppServer/web/help:/WebSphere/r7cell/AppServer/deploytool/itp/plugins/com.ibm.etools.ejbdeploy/runtime
java.vm.specification.name = Java Virtual Machine Specification
CBCONFIG =
java.vm.specification.version = 1.0
javax.rmi.CORBA.PortableRemoteObjectClass = com.ibm.ws390.orb.PortableRemoteObjectDelegate
com.ibm.ws390.INTERNAL.ServerInstanceConfiguredSystem =
user.install.root = /WebSphere/r7cell/AppServer
java.io.tmpdir = /tmp
java.vendor.url.bug =
java.awt.graphicsenv = sun.awt.X11GraphicsEnvironment
os.arch = 390
ws390.nodename = r7nodea
java.ext.dirs = /shared/java2/J1.3/lib/ext:/WebSphere/r7cell/AppServer/java/jre/lib/ext
user.dir = /WebSphere/r7cell/AppServer
com.ibm.ws390.INTERNAL.ServerInstanceName = R7SR01A
com.ibm.itp.location = /WebSphere/r7cell/AppServer/bin
line.separator =
cell_name = r7cella
java.vm.name = Classic VM
java.security.auth.login.config = /WebSphere/r7cell/AppServer/properties/wsjaas.conf
com.ibm.ws390.messaging.ma88.enabled = true
java.naming.provider.url = corbaloc:rir:/NameServiceServerRoot
com.ibm.websphere.ServerType = J2EE
file.encoding = ISO8859-1
org.omg.CORBA.ORBClass = com.ibm.ws390.orb.ORB
WAS_CELL = r7cella
protocol_http_port = 7548
node_name = r7nodea
smpe.install.root = /usr/lpp/zWebSphere/V5R0M2
java.specification.version = 1.3
CONFIG_ROOT = /WebSphere/r7cell/AppServer/config

```

## java:comp Context

Name	Type	Value
env	com.ibm.ws.naming.java.javaURLContext	com.ibm.ws.naming.java.javaURLContext(java:comp/env)
websphere	javax.naming.Context	com.ibm.ws.naming.java.javaURLContext(java:comp/websphere)
UserTransaction	java.lang.Object	com.ibm.ws390.tx.UserTransactionImpl@31674a9e
HandleDelegate	java.lang.Object	com.ibm.ejs.csi.HandleDelegateImpl@35bc0a8d
ORB	java.lang.Object	com.ibm.ws390.orb.ORB@3de30a8a

**java:comp/env Context (if any)**

Name Type Value

--- End of Document ---