IBM VisualAge® for Java™, Version 3.5

# WebSphere Test Environment

**Note!**
Before using this information and the product it supports, be sure to read the general information under **Notices**.

**Edition notice**

This edition applies to Version 3.5 of IBM VisualAge for Java and to all subsequent releases and modifications until otherwise indicated in new editions.

**© Copyright International Business Machines Corporation 1998, 2000. All rights reserved.**
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents



© Copyright IBM Corp. 1998, 2000

# Chapter 1. WebSphere Test Environment - overview

VisualAge for Java provides the WebSphere Test Environment (WTE), which contains the WebSphere Application Server, Advanced Edition run-time environment. Although the WTE encompasses the JSP file, servlet, and EJB run times and unit testing environment, it does *not* contain the entire WebSphere Advanced run time.

**Attention:** A Java-enabled Web server is packaged with the IBM WebSphere Application Server. The WebSphere Application Server is IBM's Java servlet-based Web application server that helps you deploy and manage Web applications. It is a Web server plug-in based on a server-side Java programming model that uses servlets, JavaServer Pages (JSP) files, and enterprise beans.

The WTE allows you to efficiently develop your servlets and JSP files for WebSphere. Each time you change a method in a servlet, VisualAge for Java incrementally compiles only the modified method (not the entire class), and hotlinks the method into the running program.

A Web application may consist of HTML and JSP files, servlets, and EJB beans. To better organize your project, you may want to create multiple Web applications (for example, you could separate HTML files into different directories).

With the WTE Servlet Engine, you can run multiple Web applications, each having its own document root, and configure various parts of the Web application (for example, the servlet class path and the document root). Please note that you *cannot* run multiple Web applications using the same JSP engine.

When the WebSphere Test Environment is launched, three configuration files are used:

- default.servlet_engine
- <webgroup name>.webapp (default_app.webapp)
- session.xml

You can configure these files to suit your Web application development needs. See "Configuring the default Web application."

We have provided the WebSphere Test Environment Control Center as a central place for you to start, stop, and configure WebSphere Test Environment services. The WebSphere Test Environment Control Center provides a central location for you to start, stop, and configure WebSphere Test Environment services. It allows you to do the following:

- Start, stop, and restart the Servlet Engine, configure JSP settings, and change the Servlet Engine classpath.
- Configure and enable the JSP Execution Monitor.
- Configure, start, and stop the Persistent Name Server.
- Add, remove, and update the DataSource objects.

**RELATED CONCEPTS**

© Copyright IBM Corp. 1998, 2000

Differences between WTE and WAS
JSP and servlets
JavaServer Pages
DataSources

**RELATED TASKS**

Starting, stopping, and configuring WTE services
Configuring the default Web application
Creating and configuring a new Web application
Creating and configuring DataSources using the WTE Control Center
Creating DataSources programmatically

## DataSources

DataSource objects, as defined in the JDBC 2.0 Standard Extension specification, allow you to manage a pool of connections to a database. A DataSource object is a feature provided by the WebSphere Application Server, Advanced Edition, Version 3.0 and higher.

You can create DataSources using the WebSphere Test Environment Control Center, or create them programmatically.

Using connection pools provide you with the following advantages:

- It saves you time. Creating connections is expensive; a DataSource object creates a connection as soon as it is instantiated.
- It simplifies resource allocation. Resources are only allocated from the DataSource objects, and not at arbitrary places in the code.
- It simplifies connection calls. To get a connection in JDBC 1.0, you would need to call Class.forname() on the class name of the database driver, before making DriverManager calls.

DataSource objects work as follows:

1. When a servlet, or other client, wants to use a connection, it looks up a DataSource by name from a JNDI server.
2. The DataSource object then returns a connection to the client.
3. If the DataSource object has no more connections, it may ask the database manager for more connections (as long as it has not exceeded the maximum number of connections).
4. When the client has finished with the connection, it closes the connection.
5. The DataSource object then returns the connection to the available pool.

For details on DataSource objects and their use in the WebSphere Application Server, please refer to the WebSphere Application Server online documentation.

**RELATED CONCEPTS**

WebSphere Test Environment - overview

**RELATED TASKS**

Creating and configuring DataSources using the WTE Control Center
Creating DataSources programmatically

## Differences between WebSphere Test Environment and WebSphere Application Server

The WebSphere Test Environment is a subset of the WebSphere Application Server (WAS), Advanced Edition. This means that although the WTE encompasses the JSP file, servlet, and EJB run times and unit testing environment, it does *not* contain the entire WebSphere Advanced run time. It is important that you note the differences between the WebSphere Test Environment and the WebSphere Application Server.

The WebSphere Test Environment offers the following:

- A lightweight run-time environment that loads quickly
- Standalone all-in-one unit testing
- No dependency on WAS installation or availability
- No dependency on an external database unless entity bean support is required
- Ability to debug live server-side code using the IDE debugger
- Support for *configuring* multiple Web applications

As a subset of the WebSphere Advanced Server, the WTE does not offer certain features that are offered by WAS (these are the most important features to note):

- Secure Socket Layer (SSL) and secure HTTP (HTTPS)
- HTTP-style username/password authentication challenge
- WebSphere Systems Administration server and services
- XML Configuration Tool, which produces XML grammar introduced in WAS 3.02 and the export XML configuration. Older XML grammar is used in the WTE configuration.
- WAS personalization APIs
- Security context/API on enterprise beans
- Security APIs on servlet sessions or other security classes typically involved in SignOn, Authentication, or Authorization
- JETACE tool
- Support for *running* multiple Web applications in addition to the default Web application

**RELATED CONCEPTS**

WebSphere Test Environment - overview

**RELATED TASKS**

Starting, stopping, and configuring WTE services

# Chapter 2. Using the WebSphere Test Environment

## Starting, stopping, and configuring WebSphere Test Environment services

Use the WebSphere Test Environment Control Center to start, stop, and configure WebSphere Test Environment services. See "WebSphere Test Environment - overview."

To launch the WebSphere Test Environment Control Center, select **Workspace** > **Tools** > **WebSphere Test Environment**.

**Starting, stopping, and restarting the Servlet Engine**
Use the WebSphere Test Environment Servlet Engine to run multiple Web applications, each having its own document root. The WTE Servlet Engine also allows you to configure various parts of the Web application (for example, the servlet class path and the document root). See "WebSphere Test Environment - overview." Please note that you *cannot* run multiple Web applications using the same JSP engine.

**Important!** When stopping and restarting the Servlet Engine, you must make sure that code running in the Servlet Engine (for example, a servlet) is not also running in the IDE debugger. Otherwise, the Servlet Engine will fail to stop or restart, thereby causing the Servlet Engine to die.

When serving servlets or Java bean classes that reside in the workspace, be sure to edit the classpath in order to add the project that contains the servlets and Java beans. To edit the classpath, click the **Edit classpath** button. After the classpath has been modified, you must restart the Servlet Engine for the changes to take place. The Control Center will automatically add to the classpath the minimium classpath that is required to run the Servlet Engine.

Launch the WebSphere Test Environment Servlet Engine from the WebSphere Test Environment Control Center.

1. Select **Servers** > **Servlet Engine**.
2. In the Servlet Engine window, click **Start the Servlet Engine**. Check the status line messages in the Control Center to ensure that the Servlet Engine is running in VisualAge for Java. Also, check the Console window for the Servlet Engine trace messages; they will be displayed under the following process:
   ```
   com.ibm.ivj.control.WebControlCenter.main() ->
   com.ibm.ivj.control.node.ServletEngineRunner.main()
   ```

Once you have the WebSphere Test Environment running in VisualAge for Java, you can serve JSP and HTML files from the designated document root.

**Attention:** The default document root is set for you during the installation process. We recommend that you store your JSP and HTML files in the default directory:

X:\IBMVJava\IDE\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\

where X:\IBMVJava is the directory in which VisualAge for Java is installed.

© Copyright IBM Corp. 1998, 2000

For details on changing the server root, see “Configuring the default Web application.”

**Configuring, starting, and stopping the Persistent Name Server**
Use the Persistent Name Server to get an EJB bean or DataSource object.

1. Select **Servers** > **Persistent Name Server**.
2. In the Persistent Name Server window, click **Start Name server**. Check the status line messages in the Control Center to ensure that the Persistent Name Server is running in VisualAge for Java. Also, check the Console window for the Persistent Name Server trace messages; they will be displayed under the following process:
   ```
   com.ibm.ivj.control.WebControlCenter.main() ->
   com.ibm.ivj.control.node.NameServerRunner.main()
   ```

To stop the Persistent Name Server, click **Stop Name server**.

**Configuring and enabling the JSP Execution Monitor**
Use the JSP Execution Monitor to monitor the execution of JSP source, the JSP-generated Java source, and the HTML output. See “JSP Execution Monitor.”

1. Select **JSP Execution Monitor Options**.
2. In the JSP Execution Monitor Options page, select how you want the JSP Execution Monitor to be configured. See “Using the JSP Execution Monitor.”

**Adding, removing, and modifying DataSource objects**
Use DataSource objects to manage a collection of connections to a database. See “DataSources.”

1. Select **DataSource Configuration**.
2. In the DataSource window, select **Add** to add a DataSource object from the DataSource list. To update a DataSource, simply add a DataSource with the same name as an existing DataSource; you will then be prompted to replace the DataSource.

Click **Remove** to remove a DataSource object from the DataSource list.

For details, see “Creating and configuring DataSources using the WebSphere Test Environment Control Center.”

**Tip:** When you close the Control Center, both the Servlet Engine and the Persistent Name Server are shut down. (To disable the warning dialog box, select **Don’t show again**. You can re-enable the warning dialog box by deleting the file:

X:\IBMVJava\ide\tools\com-ibm-ivj-ui-webcontrolcenter\com\ibm\ivj
\control\showExitDlg.status

from the file system.) To return to the saved settings, switch to another node. Select **No** when prompted to save the unsaved settings. Then, switch back to the current node.

#### RELATED CONCEPTS

WebSphere Test Environment - overview
Differences between WTE and WAS
JSP Execution Monitor
DataSources

#### RELATED TASKS

# Configuring Web applications

## Configuring the default Web application

When the WebSphere Test Environment is launched, three configuration files are used:

- X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\properties**default.servlet_engine**
- X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\servlets**default_app.webapp**
- X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\properties**session.xml**

**Note:** X:\IBMVJava is the directory in which you installed VisualAge for Java.

Together, these files make up the default Web application. You can configure the default Web application to suit your needs. Note that you must clear the JSP cache files after changing the document root.

**Warning!** Configuring the default Web application is for advanced users only. Faulty configuration may result in the Servlet Engine not launching successfully.

The following section describes the WebSphere XML tags for the three configuration files. Wherever necessary, the tag is specified with a value, or an empty value, and is followed by a description of the tag's meaning and usage.

**default.servlet_engine**
The default.servlet_engine file is the main configuration file for the Servlet Engine.

**<?xml version="1.0"?>**

**<websphere-servlet-engine name="servletEngine">**
Symbolic, user defined name.

**<active-transport>http</active-transport>**
The WTE supports the http transport only; the http transport must be the active transport.

**<transport>**

**<name>http</name>**

**<code>com.ibm.servlet.engine.http_transport.HttpTransport</code>**
Class that implements the transport support. The HttpTransport, which is used in the WTE, provides the embedded transport to support direct HTTP clients to the servlet engine.

**<arg name="port" value="80"/>**
The TCP/IP port used by the http transport.

**<arg name="maxConcurrency" value="50"/>**

The maximum number of request threads in the Servlet Engine. The value controls the maximum number of concurrent requests that can be processed by the Servlet Engine. In the WTE, the value should be small (2 to 5).

**<arg name="server_root" value="$server_root$"/>**

WTE server root.

**</transport>**

**<websphere-servlet-host name="default_host">**

This tag defines a virtual host in the Servlet Engine. The name is symbolic and defined by the user. You can have multiple websphere-servlet-host tags in the document. The default is set to *"Host for VisualAge for Java WebSphere Unit Test Environment"*. To use the Servlet Engine directly, the name must be changed to *"default_host"*.

**<websphere-webgroup name="default_app">**

This tag defines Web application deployment bindings within the Servlet Engine. Each Web group must have a corresponding .webapp file whose name matches the above defined Web group name. You can have multiple Web groups within the document. However, to use more than one of them, you must run ServletEngine directly.

**<description>Default WebGroup</description>**

A description of the group.

**<document-root>C:\mydocroot</document-root>**

The document root of the Web application. This is a single directory path, and is *not* a list of directories. This value controls where the Servlet Engine looks to locate static content(FileServlet) and JSP files (JSPServlet/PageCompileServlet). The default document root for the default_app is set to $approot$/web.
Note that you must clear the JSP cache files after changing the document root.

**<classpath>$approot$/servlets$psep$$server_root$/servlets</classpath>**

This is a list of directories and JAR files that contain the servlet classes and dependent classes for the Web application. The classes loaded by this path are reloadable. Each Web application has it own classpath.

**<root-uri>/</root-uri>**

This is the prefixed path that all resources within the Web application must begin with. It is a single path that cannot contain a protocol, host, or port, and must not end in / or *. See Resource below for information on how the root uri is used in building the URL that the user enters into the browser.

**<auto-reload enabled="true" polling-interval="3000"/>**
Controls whether classes within a Web application are reloaded when a change is detected. The polling-interval, specified in milliseconds, controls how often the Servlet Engine checks for class file changes.

**<shared-context>false</shared-context>**
Controls whether the ServletContext attributes are shared in a distributed cluster. You should only use this in a clustered environment, so set it to false in the WebSphere Test Environment.

**</websphere-webgroup>**

**<mime type="application/postscript">**
Defines a mime type -> extension mapping for this virtual host. The document can contain many mime elements.

**<ext>PS</ext>**
You can provide many extensions for each type, but each type must have at least one extension.

**<ext>ai</ext>**

**<ext>eps</ext>**

**<ext>ps</ext>**

**</mime>**

**</websphere-servlet-host>**

**<hostname-binding hostname="localhost" servlethost="default_host"/>**
This element binds a particular DNS name (what the user enters into the browser) to a particular logical virtual host configuration. The hostname includes both the name and port for the host. If the port is not specified, port 80 is implied when using the Servlet Engine approach (it is 8080 when using SERunner).

The "localhost" hostname adds the default DNS name and IP address of the host machine to the hostname bindings. If the user machine is known on the network by other names, you should add them to the hostname bindings. You can have many host name bindings in the document. By default, the servlethost is set to *"Host for VisualAge for Java WebSphere Unit Test Environment"*. To use the Servlet Engine directly, the servlethost must be changed to *"default_host"*.

</websphere-servlet-engine>

**<webgroup name>.webapp (default_app.webapp)**
Each Web group defined in the default.servlet_engine file has its own .webapp file. The .webapp file loads from the classpath of the Web application (this is defined in the Web group tag). By default, the .webapp file resides in the <install root>/hosts/<virtual host name>/servlets directory.

**<?xml version="1.0"?>**

**<webapp>**
**<name>default</name>**
Symbolic, user-defined name of the Web application.

**<description>default application</description>**
Description of the Web application.

**<error-page>/ErrorReporter</error-page>**
Relative URI of a servlet, JSP, or HTML page that gets called in response to an error during the processing of a servlet. See the ErrorReporter below for an example.

**<servlet>**
Defines a servlet within a Web application. You can have many servlet elements in the document. The following snoop servlet shows an example of a user servlet.

**<name>snoop</name>**
The name by which the servlet instance is known.

**<description>snoop servlet</description>**
A description of the servlet.

**<code>SnoopServlet</code>**
The Java class name or bean name of the servlet.

**<servlet-path>/servlet/snoop</servlet-path>**
A Web path relative to the root uri of the containing web application. This becomes part of the url that the user enters in the browser to execute this servlet.

**<servlet-path>/servlet/snoop.html</servlet-path>**
Another relative web path to the same servlet instance. Any servlet can have 1->N servlet paths. The servlet path is essentially the aliasing function provided in the Version 2.0 product.

**<init-parameter>**
An initialization parameter for the servlet. You can add as many init parameters as you want to each servlet.

**<name>param1</name>**
Name of the parameter.

**<value>test-value1</value>**
Value of the parameter.

**</init-parameter>**

**<autostart>false</autostart>**
If true, the servlet loads and initializes when the servlet engine starts.

**</servlet>**

**<servlet>**

**<name>ErrorReporter</name>**
Special Servlet: Provides a default error handler for the Web application. This is optional and can be replaced with a user-defined error handler.

**<description>Default error reporter servlet</description>**

**<code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code>**

**<servlet-path>/ErrorReporter</servlet-path>**

**<autostart>true</autostart>**

**</servlet>**

**<servlet>**

**<name>invoker</name>**
Special Servlet: Provides the ″load by class name″ function used in the WebSphere Test Environment. This servlet is optional. It allows the user to execute the servlet by entering /servlet/<Java class name>.

Note that /servlet is simply an example of the servlet path to this servlet. You may want to use a different or additional path.

**<description>Auto-registration servlet</description>**

**<code>com.ibm.servlet.engine.webapp.InvokerServlet</code>**

**<servlet-path>/servlet</servlet-path>**

**<autostart>true</autostart>**

**</servlet>**

**<servlet>**

**<name>jsp</name>**
Special Servlet: Provides JSP 0.91 support to the Web application. The user can specify either this servlet, or the JSP 1.0 servlet below, to control JSP function within the Web application.

**<description>JSP 0.91 or JSP 1.0 support servlet</description>**
Servlet description.

**<code>com.ibm.ivj.jsp.debugger.pagecompile.IBMPageCompileServlet</code>**
Use this servlet or JSP 0.91 support in the WTE.

**<code>com.ibm.ivj.jsp.runtime.JspDebugServlet</code>**
Use this servlet for JSP 1.0 support in the WTE.

See the VisualAge for Java JSP/Servlet Development Environment documentation for how to migrate between JSP 0.91 and 1.0 support. You cannot specify both servlets at the same time.

**<init-parameter>**

**<name>workingDir</name>**

**<value>$server_root$/temp/default_app</value>**
The default directory where the JSP 0.91 Page Compiler generates its code.

**</init-parameter>**

**<init-parameter>**

**<name>jspemEnabled</name>**

**<value>true</value>**
Enables and disables the JSP Execution Monitor support by default.

**</init-parameter>**

**<init-parameter>**

**<name>scratchdir</name>**

**<value>$server_root$/temp/JSP1_0/default_app</value>**
The directory where the JSP 1.0 Page Compiler generates its code by default.

**</init-parameter>**

**<init-parameter>**

**<name>keepgenerated</name>**

**<value>true</value>**

**</init-parameter>**

**<autostart>true</autostart>**

**<servlet-path>*.jsp</servlet-path>**

**</servlet>**

**</webapp>**

**session.xml**
The session.xml file controls the session management functions in the Servlet Engine.

**<?xml version=″1.0″?>**

**<session>**

**<sessions-enabled>true</sessions-enabled>**
Enables and disables session support.

**<session-manager-name>Session Manager</session-manager-name>**
Symbolic name for the session manager.

**<session-data>**

**<url-rewriting-enabled>false</url-rewriting-enabled>**
Enables and disables URL rewriting.

**<protocol-switch-rewrite-enabled>false</protocol-switch-rewrite-enabled>**
Enables and disables protocol switch rewriting.

**<cookie-data>**
Controls the cookie parameters.

**<enabled>true</enabled>**
Enables and disables cookie support.

**<name>sessionid</name>**
Name of the session cookie.

**<comment>WebSphere Session Support</comment>**
Cookie comment.

**<domain></domain>**
Cookie domain.

**<maximum></maximum>**
Cookie maximum.

**<path>/</path>**
Cookie path.

**<secure>false</secure>**
Enables and disables secure cookie support.

**</cookie-data>**

**<timeout>1800</timeout>**
Specifies the session timeout in seconds.

**<size>1000</size>**
Specifies the maximum number of active sessions.

**<enable-overflow>true</enable-overflow>**
Specifies how to handle session management overflows.

**<enable-measurements>true</enable-measurements>**
Enables and disables EPM performance data collection.

**</session-data>**

The following tags in the session.xml file are not configurable by the user in the WebSphere Test Environment. These tags must be left with their default values.

**<session-store>**
Controls how sessions are stored. By default, an in-memory session is used.

Note that a persistent session is used in a cluster and not in the WTE.

**<persistent-store>false</persistent-store>**
Enables and disables a persistent session.

**<persistence-type>directodb</persistence-type>**
Sets the persistence method (directodb, ejb).

**<persistence-database>db2</persistence-database>**
Sets the persistence database.

**<persistence-multirowschema>false</persistence-multirowschema>**
Sets the database schema used for a session.

**<persistence-cache>true</persistence-cache>**
Enables and disables session caching.

**<persistence-asyncupdate>false</persistence-asyncupdate>**
Enables and disables asynchronous session to db updates.

**<persistence-connectionsize>50</persistence-connectionsize>**
Sets the number of connections to the session database.

**<persistence-datasource-name>jdbc/db2/sample</persistence-datasource-name>**
Sets the database name used by session.

**</session-store> <db2-info>**
Database information for DB2.

**<driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>**

**<tablename>sessions</tablename>**

**<url>jdbc:db2:was</url>**

**<owner></owner>**

**<userid></userid>**

**<password></password>**

**<native-access>false</native-access>**

**</db2-info>**

**<oracle-info>**
Database information for Oracle.

**<driver>oracle.jdbc.driver.OracleDriver</driver>**

**<tablename>sessions</tablename>**

**<url>jdbc:oracle:oci8:@</url>**

**<owner>scott</owner>**

**<userid>scott</userid>**

**<password>tiger</password>**

**</oracle-info>**

**</session>**

**RELATED CONCEPTS**

WebSphere Test Environment - overview

**RELATED TASKS**

Starting, stopping, and configuring WTE services
Creating and configuring a new Web application

## Creating and configuring a new Web application

You can create a new Web application in addition to using the default configurations as provided in the default.servlet_engine and *.webapp files (see "Creating the default Web application").

When you have added the IBM WebSphere Test Environment feature to your workspace, complete the following steps to set up and use the Servlet Engine directly. The following scenario describes how to configure the desired behavior. The scenario references a Java source file, ServletEngineConfigDumper.java, which is available with the WebSphere Application Server.

Complete the following:

1. Create a new Web application, called *config_app*. This new Web application should contain the same servlets as provided with the default_app Web application, along with an additional servlet. The config_app Web application will configure the new servlet.
   a. Create the config_app Web application directory:
      X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\config_app
      where X:\IBMVJava is the directory in which VisualAge for Java is installed.
   b. Create the config_app Web application's *web* and *servlets* directories that map to the servlet files indicated above:
      X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\config_app\web
      X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\config_app\servlets
2. Modify the default.servlet_engine file as follows (the example below demonstrates how to set up a root-uri for the config_app Web application; the root-uri is configured to /configuration/testing):

```
<?xml version="1.0"?>
<websphere-servlet-engine name="servletEngine">
   <active-transport>http</active-transport>
   <transport>
      <name>http</name>
 <code>com.ibm.servlet.engine.http_transport.HttpTransport</code>
         <arg name="port" value="8080"/>
         <arg name="maxConcurrency" value="50"/>
         <arg name="server_root" value="$server_root$"/>
    </transport>
    <websphere-servlet-host name="default_host">
    <websphere-webgroup name="default_app">
         <description>Default WebGroup</description>
         <document-root>$approot$/web</document-root>
<classpath>$approot$/servlets$psep$$server_root$/servlets</classpath>
    <root-uri>/</root-uri>
    <auto-reload enabled="true" polling-interval="3000"/>
    <shared-context>false</shared-context>
</websphere-webgroup>
<!— The following section specifies an additional, distinctly configured Web
application called config_app. For this new Web application, we have
configured a different document root and servlet path from the default_app
Web application —>
<websphere-webgroup name="config_app">
    <description>Testing WebGroup Application</description>
    <document-root>$approot$/web</document-root>
    <classpath>$approot$/servlets</classpath>
<!— The following section demonstrates how to configure a different virtual
path (in this case, /configuration/testing), so that the servlet gets invoked
when the URL http://localhost:8080/configuration/testing is called. —>
    See the config_app.webapp where an additonal servlet alias is specified
which also invokes the "application"
    —>
         <root-uri>/configuration/testing</root-uri>
    <auto-reload enabled="true" polling-interval="3000"/>
    <shared-context>false</shared-context>
</websphere-webgroup>
```

```
<!— Specify any additional, global MIME types here if required —>
...
<mime type="application/octet-stream">
    <ext>bin</ext>
    <ext>class</ext>
</mime>
...
</websphere-servlet-host>
<!—include the hostname and the port —>
<hostname-binding hostname=localhost:8080“ servlethost=”default_host“/>
<hostname-binding hostname=”127.0.0.1:8080“ servlethost=”default_host“/>
</websphere-servlet-engine>
```

3. Create a config_app.webapp file, saving it in the following directory: X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\config_app\servlets
4. Copy and paste the WebSphere ServletEngineConfigDumper servlet into the same directory: X:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\config_app\servlets This servlet is available from the WebSphere examples directory of the WebSphere Application Server.
5. Create an index.html page to be used as your home page. Place this file in your document root. Also in the document root directory, copy and paste the error.jsp file (available from the default_app document root, ..\default_app\web) to handle error messages. The following config_app.webapp example demonstrates how to configure a servlet alias and servlet path (your config_app.webapp file should resemble the following):

```
<?xml version="1.0"?>
<webapp>
    <name>config_app</name>
    <description>Servlet Engine configuration testing
application</description>
    <error-page>/ErrorReporter</error-page>
    <servlet>
        <name>ErrorReporter</name>
        <description>Default error reporter servlet</description>
        <code>com.ibm.servlet.engine.webapp.DefaultErrorReporter</code>
        <servlet-path>/ErrorReporter</servlet-path>
        <autostart>true</autostart>
    </servlet>
    <servlet>
        <name>invoker</name>
        <description>Auto-registration servlet</description>
        <code>com.ibm.servlet.engine.webapp.InvokerServlet</code>
        <servlet-path>/servlet</servlet-path>
        <autostart>true</autostart>
    </servlet>
    <servlet>
        <name>jsp</name>
        <description>JSP support servlet</description>
<!— VisualAge for Java JSP 0.91 support servlet

<code>com.ibm.ivj.jsp.debugger.pagecompile.IBMPageCompileServlet</code>
—>
        <!— VisualAge for Java JSP 1.0 support servlet
```

```
        <code>com.ibm.ivj.jsp.runtime.JspDebugServlet</code>
—>
        <code>com.ibm.ivj.jsp.runtime.JspDebugServlet</code>
        <init-parameter>
            <name>workingDir</name>
            <!— Keep it simple, and use the WTE \temp directory as the
page compile working directory —>
            <value>$server_root$/temp</value>
        </init-parameter>
        <init-parameter>
            <name>jspemEnabled</name>
            <value>true</value>
        </init-parameter>
        <init-parameter>
            <name>scratchdir</name>
            <!— Keep it simple, and use the WTE \temp directory as the
scratch directory for the page compiled code —>
            <value>$server_root$/temp</value>
        </init-parameter>
        <init-parameter>
            <name>keepgenerated</name>
            <value>true</value>
        </init-parameter>
        <autostart>true</autostart>
        <servlet-path>*.jsp</servlet-path>
    </servlet>
    <servlet>
        <name>file</name>
        <description>File serving servlet</description>
        <code>com.ibm.servlet.engine.webapp.SimpleFileServlet</code>
        <servlet-path>/</servlet-path>
        <autostart>true</autostart>
    </servlet>
    <!— Adding this servlet to the config_app Web application for
demonstration purposes —>
    <servlet>
        <name>A Servlet Engine Configuration Tester</name>
        <description>Servlet Engine configuration test servlet</description>
        <code>ServletEngineConfigDumper</code>
<!— Here is where you would apply a servlet alias, if desired. For this
example, the servlet can also be called by "/mydumper“ —>

        <servlet-path>/configDump</servlet-path>
        <autostart>true</autostart>
        <!— This section demonstrates how to provide init parameters to this
servlet or to any other servlet —>
        <init-parameter>
            <name>param1</name>
            <value>some_init_value1</value>
        </init-parameter>
        <init-parameter>
            <name>param2</name>
            <value>some_init_value2</value>
        </init-parameter>
    </servlet>
</webapp>
```

6. Inside the IDE, create a project to contain the ServletEngineConfigDumper servlet source. Add this project class path to the workspace class path for com.ibm.servlet.engine.ServletEngine by using the Properties dialog for this class (right-click the class, and select **Properties**). Import the ServletEngineConfigDumper .java source to this project so that you can later debug it.
7. Locate the com.ibm.servlet.engine.ServletEngine class in your VisualAge for Java directory, and set the following single command line parameter, making sure to include the quotation marks (use your installation path):
   -serverRoot ″c:\IBMVJava\ide\project_resources\IBM WebSphere Test Environment″
8. Instead of applying the workspace projects class path for ServletEngine, add the following class path entries (you may use either forward slashes (Unix style) or backward slashes (NT style)):
   ../IBM WebSphere Test Environment/lib/ns.jar;
   ../IBM WebSphere Test Environment/lib/ibmwebas.jar;
   ../IBM WebSphere Test Environment/lib/servlet.jar;
   ../JFC class libraries/;
   ../IBM Persistence EJB Library/;
   ../IBM JSP Examples/;
   ../Servlet API Classes/;
   ../IBM Data Access Beans/;
   ../IBM XML Parser for Java/;
   ../JSP Page Compile Generated Code/;
   ../IBM WebSphere Test Environment/;
   ../IBM WebSphere Test Environment/properties/;
   ../IBM IDE Utility local implementation/;
   ../IBM IDE Utility class libraries/;
9. Ensure that the active transport specified in default.servlet_engine is HTTP (HTTP is the only supported transport protocol):
   <active-transport>http</active-transport>
   ```
   <transport>
       <name>http</name>
       <code>com.ibm.servlet.engine.http_transport.HttpTransport</code>
       <arg name=″port″ value=″8080″/>
       <arg name=″maxConcurrency″ value=″50″/>
       <arg name=“server_root” value=“$server_root$”/>
   </transport>
   ```
10. Run the `main` method of the com.ibm.servlet.debug.ServletEngine class. Check the Console window to ensure that the Servlet Engine process is running.

**RELATED CONCEPTS**

WebSphere Test Environment - overview
DataSources

**RELATED TASKS**

Starting, stopping, and configuring WTE services
Configuring the default Web application

# Creating and configuring DataSources

## Creating and configuring DataSources using the WTE Control Center

Use DataSource objects for your connection pooling. See “DataSources.”

Through the WebSphere Test Environment Control Center, you can create and configure DataSources efficiently:

1. Launch the WebSphere Test Environment Control Center.
2. Select **Persistent Name Server**. In the Persistent Name Server window, click **Start Persistent Name Server**.
3. Select **DataSource**.
4. In the DataSource window, click **Add**. The Add DataSource dialog opens.
5. In the DataSource name field, enter a name for the DataSource object that you want to create.
6. In the Database driver field, select the driver class name of the database that you want to connect to.
7. In the Database URL field, specify the URL of the database that you want to connnect to.
8. In the Database type field, select the field of the database that you want to connect to.
9. In the Description field, enter a description of the DataSource object. Note that this is optional.
10. It is recommended that you accept the defaults for the remaining fields.
11. Click **Add** to create the DataSource object, and to bind it into the Persistent Name Server context. To update a DataSource, simply add a DataSource with the same name as an existing DataSource; you will then be prompted to replace the DataSource.

For details on creating DataSources manually using the DataSource factory API, see “Creating DataSources programmatically.”

**RELATED CONCEPTS**

WebSphere Test Environment - overview
DataSources

**RELATED TASKS**

Starting, stopping, and configuring WTE services
Creating DataSources programmatically

## Creating DataSources programmatically

Use DataSource objects for your connection pooling. See “DataSources.”

You can create DataSource objects programmatically using the new DataSource factory API. Once the DataSource object is defined and bound, you obtain the context from the Persistent Name Server in VisualAge for Java, and then perform the lookup.

Creating DataSources manually involves the following:

1. Get an initial context.

2. Create and object.
3. Bind the DataSource to the context.

**Getting an initial context**
When you create a DataSource object, you give it a set of parameters (name, database URL, and database driver class) that uniquely identify it. To make the DataSource persistent, you must bind it to a context, which will store the description of the DataSource in the administration repository.

**Note:** In VisualAge for Java, the administration repository is the database identified in the properties of the Persistent Name Server.

**Creating an object and binding the DataSources to the context**
In VisualAge for Java, you can create DataSources using the DataSourceFactory class.

**Warning!** To create and bind a DataSource in the WebSphere Application Server, use the WebSphere Administration Console. Do not use the DataSourceFactory class to create a DataSource in the WebSphere Administration repository.

When you have created and bound a DataSource to the context, you can refer to it until it is unbound.

The following code snippet was adapted from CreateDataSourceServlet.java. This servlet accepts parameters, using them to create a DataSource.

First, you need to get the initial context:

```
//Get the initial context
Hashtable parms = new Hashtable();
parms.put(Context.INITIAL_CONTEXT_FACTORY ,
     "com.ibm.ejs.ns.jndi.CNInitialContextFactory ");
parms.put(Context.PROVIDER_URL , "iiop:///");
try {
     ctx = new InitialContext(parms );
} catch (Exception e) {
}
```

Second, create the DataSource object. The parameters for creating a DataSource object are as follows:

- **Name** An arbitrary name identifying the DataSource.
- **Database URL** The real database URL used to uniquely identify the database, for example, jdbc:db2:sample, or jdbc:db2://myserver:sample.
- **Database driver classname**
- **Database driver type: JDBC or JTA** Using JTA (Java Transaction API) indicates that the driver can handle two-phase commits. For DB2, version 6.1, specify JTA. For DB2, version 5.2, specify JDBC.
- **Description** An arbitrary description of the DataSource object.

These parameters are set in an Attributes object, which is passed as an argument to one of the DataSourceFactory create methods. The following code snippet assumes that the Attributes oejcts, attr, has been initialized:

```
DataSourceFactory factory = new DataSourceFactory();
DataSource dataSource = null;
```

```
if (dsType.equals("JTA "))
     dataSource = factory.createJTADataSource(attr );
else
     dataSource = factory.createJDBCDataSource(attr );
```

Third, bind the DataSource object to the context (this causes the JNDI bind() method to be called):

```
try {
     factory.bindDataSource(dataSource , ctx);
} catch (javax.naming.NamingException ne) {
}
```

For details on creating DataSource objects using the WebSphere Test Environment Control Center, see "Creating and configuring DataSources using the WebSphere Test Environment Control Center."

**RELATED CONCEPTS**

WebSphere Test Environment - overview
DataSources

**RELATED TASKS**

Starting, stopping, and configuring WTE services
Creating and configuring DataSources using the WTE Control Center

# Notices

Note to U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive*
*Armonk, NY 10504-1785*
*U.S.A.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
*IBM World Trade Asia Corporation*
*Licensing*
*2-31 Roppongi 3-chome, Minato-ku*
*Tokyo 106, Japan*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 1998, 2000

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*Lab Director*
*IBM Canada Ltd.*
*1150 Eglinton Avenue East*
*Toronto, Ontario M3C 1H7*
*Canada*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 1997, 2000. All rights reserved.

# Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow the customer to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

**Warning:** Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

© Copyright IBM Corp. 1998, 2000

# Trademarks and service marks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

- AIX
- AS/400
- DB2
- CICS
- CICS/ESA
- IBM
- IMS
- Language Environment
- MQSeries
- Network Station
- OS/2
- OS/390
- OS/400
- RS/6000
- S/390
- VisualAge
- VTAM
- WebSphere

Lotus, Lotus Notes and Domino are trademarks or registered trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli Enterprise Console and Tivoli Module Designer are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

Encina and DCE Encina Lightweight Client are trademarks of Transarc Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

ActiveX, Microsoft, SourceSafe, Visual C++, Visual SourceSafe, Windows, Windows NT, Win32, Win32s and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Intel and Pentium are trademarks of Intel Corporation in the United States, or other countries, or both.

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.

© Copyright IBM Corp. 1998, 2000