**JDBC Connection Pulling**

*Connection pooling* means that connections are reused rather than created each time a connection is requested. To facilitate connection reuse, a memory cache of database connections, called a *connection pool*, is maintained by a connection pooling module as a layer on top of any standard JDBC driver product.

Connection pooling is performed in the background and does not affect how an application is coded; however, the application must use a DataSource object (an object implementing the DataSource interface) to obtain a connection instead of using the DriverManager class.

Once a DataSource object is registered, the application retrieves it from the JNDI naming service in the standard way.

For example:

Context ctx = new InitialContext();

DataSource ds = (DataSource) ctx.lookup("jdbc/SequeLink");

If the DataSource object provides connection pooling, the lookup returns a connection from the pool if one is available. If the DataSource object does not provide connection pooling or if there are no available connections in the pool, the lookup creates a new connection. The application benefits from connection reuse without requiring any code changes. Reused connections from the pool behave the same way as newly created physical connections. The application makes a connection to the database and data access works in the usual way. When the application has finished its work with the connection, the application explicitly closes the connection.

For example:

Connection con = ds.getConnection("scott", "tiger");

**JNDI Naming Services:** The Java Naming and Directory Interface (**JNDI**) is a Java API for a directory service that allows Java software clients to discover and look up data and objects via a name. Like all Java APIs that interface with host systems, **JNDI** is independent of the underlying implementation.

**A partner is sending a file how wm knows what type of document?**

Exchange files securely and reliably webMethods ActiveTransfer is a Managed File Transfer (MFT) software solution that enables you to send and receive files of any size both internally and externally.

Your trading partners need only standard file transfer tools, such as a Web browser, to exchange files with you. That means you can add new suppliers quickly to your extended enterprise or B2B network. Files are encrypted for security. And, because the software provides file acceleration out of the box, it makes it suitable for handling large file transfers over long distances.

Built on the proven webMethods Integration Server, webMethods ActiveTransfer is fully integrated with the webMethods suite, enabling you to replace antiquated and non-secure file transfer systems with a consolidated, easy-to-use platform.

You’ll gain centralized management and control of all file transfer activity. Gain visibility into the state of critical file transfers to discover and resolve file transfer issues as they happen.

Best of all, incoming files can automatically trigger business events, such as ordering and invoicing, to speed up processes. Support for secure file-transfer protocols.

Trading partners need only standard tools such as a Web browser to exchange files with you. That’s because webMethods ActiveTransfer comes ready to support Web browsers and FTP clients. Transfer files using HTTP, HTTPS, FTP, FTPS, SFTP, SCP and WebDAV.

**Creating a Web Service**

You can create a web service starting from Java code or starting from a WSDL file. The following sections describe each approach:

Creating a Web Service From Java

Creating a web Service from WSDL

One way to create a web service application is to start be coding the endpoint in java. If you are developing your Java web service from scratch or have an existing Java class you wish to expose as a web service, this is the most direct approach.

The web service is written as a normal Java class. Then the class and its exposed methods are annotated with the web service annotations @WebService and @WebMethod.

When developing a web service from scratch or based on an existing Java class, WSIT features are enabled using a configuration file. That file, wsit-<*package*>.<*service*>.xml, is written in WSDL format. An example configuration file can be found in the accompanying samples:

wsit-enabled-fromjava/etc/wsit-fromjava.server.AddNumber-

sImpl.xml

**To create a web service from Java, create the following files:**

* These files define the web service and the WSIT configuration for the service:
  + Web Service Implementation Java File
  + wsit-<package>.<service>.xml File
* These files are standard files required for JAX-WS. Examples of these files are provided in the wsit-enabled-fromjava sample directory.
  + AddNumbersException.java
  + custom-schema.xml
  + sun-jaxws.xml
  + web.xml
* These files are standard in any Ant build environment. Examples of these files are provided in the wsit-enabled-fromjava sample directory.
  + build.xml
  + build.properties

**Creating a Web Service From WSDL**

Typically, you start from WSDL to build your web service if you want to implement a web service that is already defined either by a standard or an existing instance of the service. In either case, the WSDL already exists. The JAX-WSwsimport tool processes the existing WSDL document, either from a local copy on disk or by retrieving it from a network address or URL. For an example of using a web browser to access a service's WSDL.

**WSDL File**

You can create a WSDL file by hand or retrieve it from an existing web service by simply pointing a web browser at the web service's URL. The snippet shown below illustrates how to enable the WSIT Reliable Messaging technology in a WSDL file.

<wsp:Policy wsu:Id="AddNumbers\_policy">

  <wsp:ExactlyOne>

    <wsp:All>

      <wsrm:RMAssertion>

        <wsrm:InactivityTimeout Milliseconds="600000"/>

        <wsrm:AcknowledgementInterval Milliseconds="200"/>

      </wsrm:RMAssertion>

    </wsp:All>

  </wsp:ExactlyOne>

</wsp:Policy>

**Broker Concept in webMethods:**

The interactions will happen through Broker. webMethods Integration Server(IS) is mainly used to support the integration of diverse services and communication between systems.

The Components that interact with the Broker.

1. wM Integration Server

2. webMethods JMS with ASF Support

3. webMethods Task Engine

4. webMethods Logging Utility

5. webMethods Process Engine

A webMethods Broker environment mainly consists of following main components.

**1. Broker Server:** This is a runtime components which mainly used for the Publishers and Subscribers to Interact. It also manages memory and disk resources for all the Brokers that reside on it. By default, port 6849 is the Broker Server base port. It uses the default port 6849 for non-SSL requests and ports 6848 and 6847 for SSL requests.

**2. Brokers:** This resides on Broker Server. When a client connects to Broker Server, the client specifies the Broker with which it wants to interact.

**3. Broker User Interface:** This is a administrating component, which we mainly used to monitor Broker Health, log size etc. This is running on My webMethods Server.

**4. Broker Monitor:** This will install automatically when we install Broker Server. As in the name describes that, the Broker Monitor will mainly monitor all the Broker servers status. Broker Monitor continually checks the state of the Broker Server and automatically attempts to re-start it if it stops running. The Broker Monitor default port 6850.

The Broker Monitor configuration file (awbrokermon.cfg) points to the Broker Servers that reside on the host machine. So when BrokerMonitor starts the broker server, it will capture the respective Broker Server PID and using this PID it will monitor the status of the respective Broker servers.

A webMethods Broker environment has only one Broker Monitor even if it hosts multiple Broker Servers. Broker Monitor monitors all of the Broker Servers running in the

webMethods Broker environment. It will automatically attempt to restart any Broker Server that stops running in its environment.

Broker encompasses the following objects

**1. Document types:** which identify the kinds of documents that the Broker’s clients can

exchange.  
**2. Client Groups:** which define specific properties and permissions that Broker applies to

clients.  
**3. Client State Objects:** which maintain information about the individual clients that use

the Broker.

When we install Broker Server, the server will be installed with a "default" Broker, which we will be mainly used to communicate with applications which not mentioned with to which broker it needs to communicate. We can install required number of Brokers apart from this "default" broker.

**Flow Services:**

Flow is a simple, point-and-click programming language, optimized for coding services.

You create and run a **Flow** service by:

* Creating the **Flow** service object
* Adding inputs and processing steps to the service
* Verifying the output from the service

You will create a Flow service by defining the Flow service object in the services folder of the **FLOW\_Tutorial** package.

To create a Flow service:

* locate the **FLOW\_Tutorial** package that you created previously, and expand the folder hierarchy.
* Right-click on the services folder in the **FLOW\_Tutorial** package, then select **New -> Flow Service.**

The New menu displays all the types of objects you can create on the IS.

In the **New Flow Service** dialog, enter customWriteToLog in the Element name field and select Finish.

Designer creates the new flow service and opens it up in the Flow editor. Expand the **services** folder to see your newly created flow service.

By default, the **Tree** tab is selected, showing an empty Flow. The Flow **Palette** may be expanded by clicking on the arrow on the right side of the flow editor.

* Click the **Input/Output** tab on the bottom of the Flow Editor to open.

You may now add inputs to the new service. Open the Variables Palette by clicking the open arrow on the right of the **Input/Output** editor.

**PipeLine:**

The **Pipeline** tab is divided into three sections:

* **Pipeline In**
* <selected service object>
* **Pipeline Out**

The Pipeline toolbar provides various pipeline management functions, including the **Link** tool.

We use the **Pipeline** tools by first selecting one or more variables that you want to act on. When you select a variable, the applicable tools will become available in the toolbar.

For example, to map the variable string1 to the variable**instring1:**

* Select **string1** in the **Pipeline** In section, then select **inString1** in the **Service In** section

The **Link** tool is now active.

* Click on the **Link** tool.

The **Link** tool maps **string1** to **inString1.**

**Creating Certificates in webMethods:**

Each Software AG customer is responsible for creating the required certificates. This describes how to create a self-signed certificate from start to finish, and how to convert and create all the required file formats.

**Generating a Certificate Chain for My webMethods Server:**

For external websites that need to be secured, an external Certificate Authority (CA) such as VeriSign is engaged to sign your certificate requests. Virtually all browsers will have a copy of their public certificates. The browser downloads the certificate from a website, determines the authority that signed it, and if the signing authority’s certificate is in the browser’s list of trusted certificates, then the website is also trusted (provided other aspects of the certificate are also valid). Otherwise, the browser displays a message that that there is a certificate problem.

For internal use, you can dispense with engaging an external CA, as it is possible to set up an internal organizational CA. Provided that the public CA certificate is installed in all connecting browsers, then anything signed by the internal CA will be trusted as well.

**webMethods BPM(Bussiness Process Management):**

Software AG’s webMethods BPM Platform is a unified offering designed to create business applications that are based on orchestrated, managed and monitored end-to-end business processes or workflows. The offering improves process speed, visibility, consistency and agility while reducing costs and driving standardization.