The Florida State University

College of Arts and Science

Experimenting with Technologies for Development and Integration of Distributed Systems

By

Sergio Souza

November 20th, 2003

Dr. Gregory A. Riccardi
Major Professor

Dr. Lois W. Hawkes
Committee Member

Dr. Daniel G. Schwartz
Committee Member

Acknowledgments

I would like to thanks Dr. Riccardi for introducing me to the technologies of software development and integration, for his writing and editing skills, and for the constant guidance throughout the duration of this project. Also thanks Dr. Hawkes and Dr. Schwartz for being part of the committee.

Abstract

My goal with this project was to learn about cutting edge technologies related to software development and integration. I have covered a wide range of subjects including distributed applications, Java 2 Enterprise Edition, Web services, and Grid services. These technologies have allowed me to construct industry standard software components and understand the creation process of professionally built software that is modular, scalable, secure, and platform independent.

Table of Contents

Experimenting with Technologies for Development and Integration of Distributed Systems

My goal with this project was to learn about software tools related to application development and integration. I intended to work with new technologies that could have a great impact in the computing industry. I looked into Java 2 Enterprise Edition (J2EE), Web services, and Grid services, analyzing their concepts and inspecting some tools used for implementation and integration of software. Experts acknowledge that these technologies will set the standards for software development and integration for distributed environments.

Professionally built software needs to address several issues beyond the application logic. Issues such as security and transaction management comprise the infrastructure code of an application. Adding this code to every application can be a painstaking task. J2EE provides a framework that allows separation of the infrastructure from the business logic of an application. In a distributed environment, applications need to interact with other applications. Web services using XML have emerged as a model that allows applications to communicate with each other over the Internet. Applications using Web services can interact regardless of their platform or implementation. Yet another technology, Grid services, aims to allow widely distributed computers to share resources efficiently and securely.

In order to learn, develop, and integrate applications, I created a small distributed computing environment consisting of four machines with Windows and Linux operating systems. The software tools and applications installed included databases, web servers, application servers, integrated development environments (IDEs), and grid computing software. Using online tutorials, magazine articles, and other publications, I have gained considerable knowledge of J2EE, Web services, and Grid services and have accomplished the following:

- Build a distributed computing environment
- Develop applications using state of the art IDEs
- Create J2EE applications, Web and EJB modules, Servlets, JSPs
- Interact with databases from an IDE
- Build Web services, deploy classes as Web services, create services from WSDL
- Deploy software on application servers
- Build a Grid service computing environment and understand its principles

The remainder of this document explains in more detail the technologies, the environment, and the learning process used to implement and deploy the applications.

1 The Technologies:

This section explains in more detail the distributed service model and the three main technologies discussed in this document. They are Java 2 Enterprise Edition (J2EE), Web services, and Grid services.

1.1 *Distributed Object Model*

A distributed service model requires the processing of data and objects across a network of computers. Manipulating such objects and data across different systems involves many challenges.

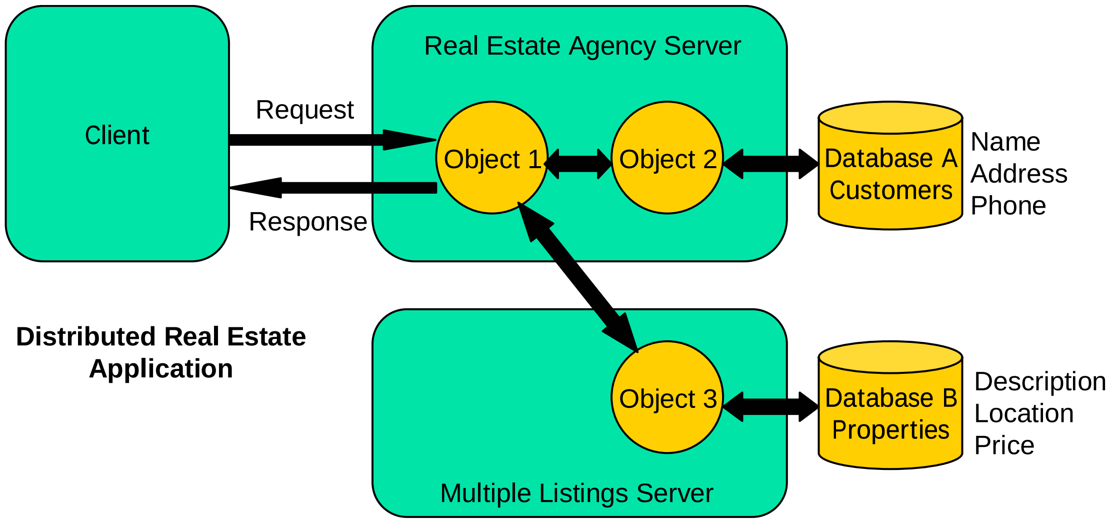


Figure 1. A Distributed Service Model

Figure 1 illustrates a distributed system example in which objects represent services or capabilities and are able to communicate with each other, clients, and databases. Let's assume a real estate company has an application that allows its employees to search for information on customers and properties. Object 1 allows the user to search for customer information. Object 1 also allows the searching of real estate properties using some criteria. If the user submits a real estate search request, Object 1 contacts Object 3, which has access to Database B containing the information on real estate properties. Similarly, if a user submits a search on customers, such request would cause Object 1 to contact Object 2, which has access to Database A containing information about the customers.

In this distributed model scenario, the servers must deploy the objects, make them available and address several issues including:

- Multithreading: the server must be able to handle multiple clients' requests
- Object life-time: the server needs to create and destroy objects as needed
- Communication protocol: the server must allow objects to talk to each other
- Network reliability: the server needs to account for network failures
- Security: the server must address authentication and authorization
- Database connectivity: the model must manage a pool of database connections

Even the simplest distributed system requires an extensive amount of complex code to address the mentioned issues. In the next section I'll describe J2EE as a technology that facilitates the development and management of distributed applications, which are also called enterprise applications.

1.2 J2EE

J2EE is a component-based framework for enterprise applications. It was developed by Sun Microsystems in collaboration with several industry vendors. The framework is designed to simplify the creation and management of enterprise applications by enabling modularity, scalability, and platform independence.

A J2EE application consists of components that are deployed in containers. Components and containers are the two main parts of the J2EE framework. Sun describes components as reusable self-contained functional software units. These components are equivalent to the objects in Figure 1 and comprise the logic of an application. The containers handle the infrastructure code such as security, object life-time, resource pooling, and multithreading.

The separation of the code into containers and components makes development of enterprise applications much easier because vendors write the intricate low-level code of the containers, while the developers are left to focus solely on the logic and user interface of the application.

1.2.1 Components

A J2EE application is made of components. They comprise the application logic and user interfaces and are written in Java. Components are required to be well formed and in accordance to the J2EE specification.

These are the components supported:

- Enterprise JavaBeans (EJBs)
- JavaServer Pages (JSPs)
- Servlets
- Applets
- Application Clients

I will address the EJBs, Servlets, and JSPs components in more detail later in this document.

1.2.2 Containers

The containers provide the infrastructure of an application. This infrastructure handles issues such as multithreading, database connection pooling, security, communication protocols, and object life-time.

The J2EE specification has the following types of containers:

- Application Client container
- Web container
- Enterprise JavaBeans container

The Application container manages Application Clients and Applets components. The Web container is responsible for managing the Servlets and JSPs. The EJB container is left to manage the Enterpriser JavaBeans of an application. Figure 2 illustrates the J2EE components and containers.

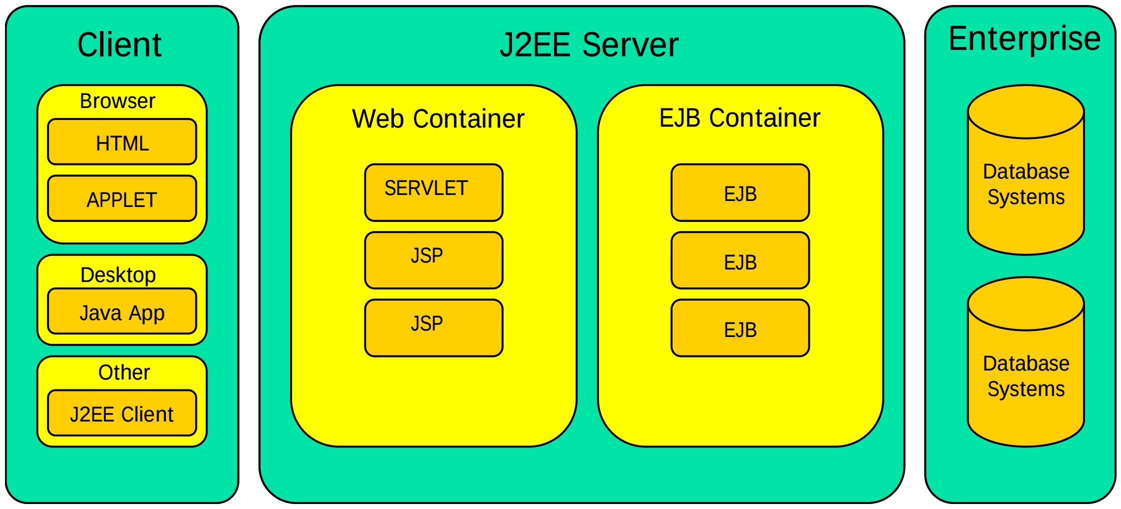


Figure 2. J2EE Tiers, Containers, and Components

J2EE components typically talk to each other via Remote Method Invocation (RMI). Web services and Grid Services allow the use of alternative communication protocols to facilitate integration of systems across the Internet.

1.3 Web Services and Grid Services

Web services and Grid services are technologies that help integration of software applications by allowing services to be used more efficiently over the internet.

The key to Web services is the Web Services Description Language (WSDL). Each WSDL document is an XML document that describes how a service should be invoked and used. The communications protocol is also specified in the WSDL. SOAP over HTTP is the most commonly used. The self-describing WSDL makes Web services a standard that allows services to be accessed remotely in a platform independent manner and without any knowledge of their implementation.

Grid services share the principles of Web services with the addition of some specific services and security to allow communication of widely distributed computers.

Distributed J2EE applications can use Grid services and Web services as an alternative method for deployment of services.

2 The Computing Environment

A four-computer network was put together to simulate a distributed environment. I used one Windows PC and three Linux machines in this configuration. This section lists the details of this computing environment, including all the software tools installed.

2.1 Hardware and Operating Systems

I started this assignment with a three-PC network. The initial settings were not appropriate and I had to make few modifications in order to efficiently develop and run the applications in this project. These modifications included addition of one machine and several modules of memory. A machine running an IDE and a database, for instance, requires at least 750MB of memory to run efficiently. These are some of the specifications of my network:

- Athlon XP (AMD 1800+), Redhat 8, 1.5GB RAM
- Athlon XP (AMD 1200+), Redhat 8, 1.25GB RAM
- Athlon XP (AMD 1600+), Windows XP Pro, 1.25GB RAM
- Intel PIII 450Mhz, Redhat 8, 512MB RAM, 800GB Raid 5

This environment has TCP/IP based networking with file sharing using NFS and Samba. The operating systems listed above reflect the current setup. Prior to this final configuration I have used Window 2000, and Redhat 7.2, 7.3, and 9. The changes of operating systems were made according to the shifting requirements of the software tools in the system. Details on the software tools follow in the next section.

2.2 Software Tools and Applications

The software tools and applications installed consisted of databases, web servers, integrated development environments (IDEs), application servers, and grid computing software. For clarity purposes, I consider two distinct environments in this document. The first is the J2EE / Web services and the other is the Grid.

2.2.1 J2EE / Web Services Environment

I used Websphere as a developing and runtime platform for J2EE and Web services. Websphere is a line of products from IBM that includes the Websphere Application Server, Websphere Studio Application Developer, and Websphere MQ. I also used DB2 as the database, and the IBM version of the Apache Web server. In the beginning of this project I installed the version 4 of the Websphere products. These products required extensive manual configuration along with installation of patches and updates. Once the version 5 of Websphere became available, I made the upgrade in my environment. This last version introduced several new features and made installation much easier. A complete installation manual for DB2 and the Websphere version 5 products used in my environment is included in the last section of this document.

2.2.2 Grid Environment

The Globus Alliance provides Grid computing software tools. Their major product is the Globus Toolkit. Throughout this project I have installed the Globus Toolkit 2.4, Globus Toolkit 3 Alpha, Beta, and Final (GT3). The latter is the current version in my system.

I successfully installed the version 2.4 in the initial phase of this project. This version, however, is not OGSA based and I abandoned it in favor of the GT3 version. I installed the GT3 alpha version and subsequently upgraded to the GT3 beta and GT3 final version as they became available.

GT3 requires some supporting software tools for a successful installation. Some of the tools needed for this version included Apache Ant, IBM JDK, and JAAS library. I also used Postgresql database for use of Reliable File Transfer (RFT).

The installation process of the Globus Toolkit was cumbersome to say the least. The instructions for a complete installation and post-setup of the fully functional Grid software are scattered across several different documents.

3 Building a J2EE Application

In this section I will discuss in more detail some of the J2EE components used in my applications and outline the development process.

3.1 The Components of the Application

As mentioned earlier in this document, J2EE applications are comprised of components. Figure 3 illustrates a J2EE real estate application using a Servlets, EJBs and JSPs. I will discuss these components in this section.

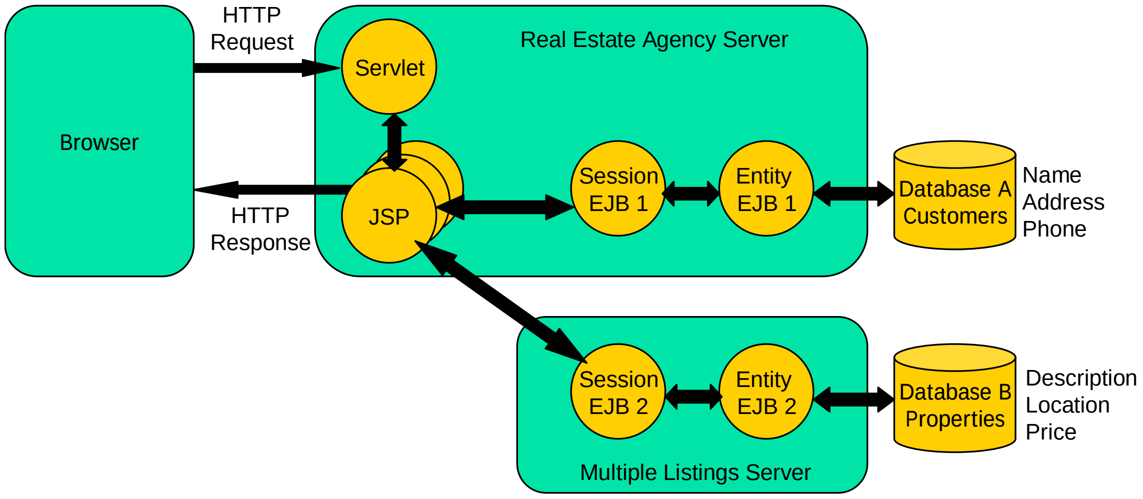


Figure 3. A J2EE real estate application using a Servlet, JSPs, and EJBs

3.1.1 EJBs

EJBs represent components that are responsible for the logic and persistent data of an application. I used two types of EJBs in my applications:

- Entity Beans
- Session Beans

An entity bean manages the persistent data of an application. Figure 3 shows a J2EE real estate application using two entity beans. Entity EJB 1 represents customers name, address, and phone number data of Database A; Entity EJB 2 represents properties description, location, and price data of Database B. These entity beans can use the EJBQL query language to perform database operations such as select, modify, and insert.

Session beans represent the logic of an application and are also used to access the entity beans. J2EE best practices recommend that entity beans are accessed through session beans as shown in Figure 3. Session beans can perform functions such as calculating mortgage payments using EJBs from different servers. Figure 4 illustrates a session EJB accessing data from entity beans to calculate mortgage payments.

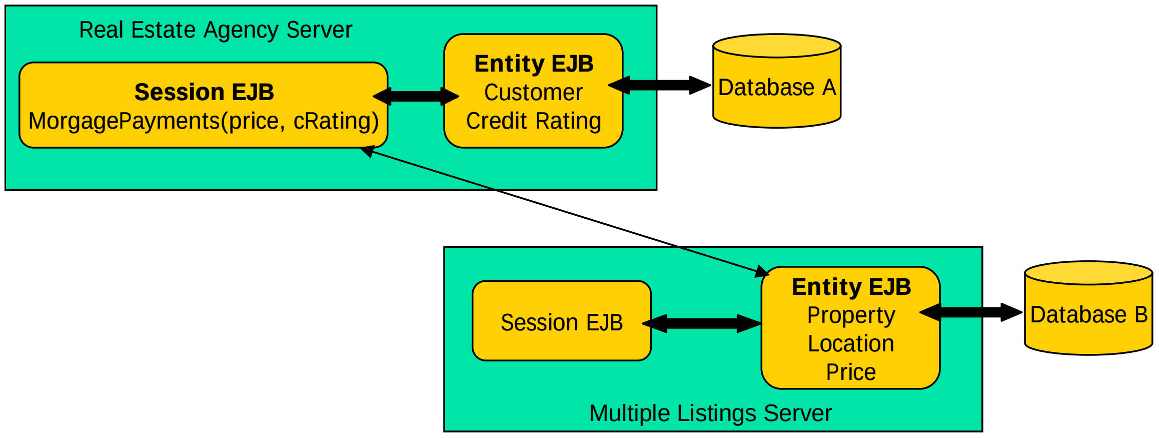


Figure 4. Session EJB accessing data from Entity EJBs

3.1.2 Servlet

Servlets are used to extend the capabilities of Web servers. In a J2EE application, a Servlet would act as a controller, receiving and responding to requests from clients. Figure 3 shows a Servlet that receives clients' requests and dispatches the appropriate JSP. Servlets can also be programmed to contact session beans and perform functions.

3.1.3 JSP

JSPs are pages that can display static and dynamic content. JSPs are commonly used to display content that is a result of a client's request. Typically, a Servlet dispatches a JSP as show in Figure 3. Upon receipt of a client request, the Servlet dispatches the appropriate JSP, which builds a page dynamically using data from session beans.

I developed EJBs, Servlets and JSPs to create J2EE applications. In the following section I will outline the process of creating and assembling a J2EE application.

3.2 Using the Integrated Development Environment (IDE)

I did all the coding and application assembly using IBM's Websphere Studio Application Developer (WSAD). An entire book would be needed to explain all the features and capabilities of this IDE. Developing a J2EE application without its use would be an unbearable job. These are some of the tasks I have been able accomplish using WSAD:

- Create J2EE applications, Web and EJB modules, Servlets, JSPs
- Interact with databases
- Build Web Services, deploy classes as Web Services, create Services from WSDL
- Create standalone Java applications
- Create Web applications using Struts
- Deploy software on application servers

I will now explain some of the development and deployment procedures. This is not intended to be a step-by-step process, but an outline of the process.

3.2.1 Developing a J2EE Application

A J2EE application can contain one or more EJB, Web, and Client modules. In WSAD, a module can also be referred to as a project. Each module has a specific file structure that must be created before any coding is done. The developer specifies the kind of project to be created and WSAD generates the appropriate structure.

I'll start by outlining the creation of the EJB module, followed by the Web module.

3.2.1.1 EJB Module

- Create an EJB project. This project must be associated with an Enterprise project. The user can specify an existing Enterprise project or instruct WSAD to create a new one
- Connect to a database
- Create an Entity bean. WSAD creates this Entity bean based on the database table specified by the developer
- Add methods to the Entity bean. Promote methods to the local interface
- Create a Session bean to serve as a facade for the Entity bean. Add methods and promote them to the remote interface
- Generate RMIC code
- Test the EJBs using WSAD's Universal Test Client

The business methods of the application are now ready. The development of the presentation part of the application is next.

3.2.1.2 Web Module

- Create a Web project. Associate this project with the same enterprise application used for the EJB module
- Create a HTML page. This page calls a Servlet
- Create a Servlet that receive parameters from the previously created HTML and performs the appropriate request, dispatching a JSP page
- Create a helper class. This class calls the Session bean methods and store results in its variables.

- Create a JSP. This file is dispatched by the Servlet and is able to dynamically generate data based on the helper class

3.2.2 Running the J2EE application

Now the application is ready to be fully tested. WSAD includes a built in application server for testing purposes. The following steps are necessary to test the application.

- Configure the XML-based deployment descriptor of the application
- Create a server folder, a server instance and configuration
- Configure the test server by adding the enterprise project
- Configure the data source of the test server
- Start the server and launch the application

3.2.3 Deploying the J2EE Application

The application is now fully tested and can be deployed in a production environment. I used IBM's Websphere Application server, but any J2EE-compliant server such as BEA Weblogic Server can be used. The process I used for deploying the application involved exporting the application from WSAD as an EAR file and following similar steps to those described in section 3.2.2

4 Conclusion:

My objective was to gain knowledge of promising technologies in the field of software development and integration. This project allowed me to have hands-on experience in development, deployment, and integration of applications. I was able to implement full J2EE applications, deploy Web services and build a Grid environment. My exposure to a complex integrated development environment such as Websphere Studio gave me the insight on the process needed to create applications that have desirable features such as modularity and scalability. Furthermore, I became knowledgeable of the benefits of J2EE, Grid and Web services and some of the challenges that they face.

5 References:

- http://java.sun.com/
- http://www.alphaworks.ibm.com/
- http://www-106.ibm.com/developerworks/
- http://www.devx.com/ibm/
- http://www.extremetech.com/
- http://www.jguru.com/
- http://www.hyperdictionary.com/
- http://www.globus.org/
- http://www.webspherecentral.com/
- The Deliberate Revolution: Transforming Integration with XML Web Services - Building Web Services Vol. 1, No. 1 – March 2003, by Mike Burner
- Building J2EE Applications with IBM Websphere – Dale Nilsson, Louis Mauget
- Websphere Studio Application Developer 5.0 Practical J2EE Development – Igor Livshin
- Professional IBM Websphere 5.0 Application Server – Tim Francis, Eric Herness

Appendix

Websphere Installation Guide

This installation guide is a compilation of different IBM documents that aid the setup of DB2 Enterprise Server Edition V8.1, Websphere MQ, IBM HTTP Server V2.0, Websphere Application Server V5.0 (WAS), and Websphere Studio Application Developer (WSAD).

All the server side components were installed on an Intel based PC using Redhat Linux 8.0. The WSAD integrated development environment was installed on a PC running Windows XP professional. The WSAD may also be installed on a Linux platform, however, many users claim the Windows version has a cleaner interface

It is recommended that the installation is done in the order stated in this manual, with the exception of WSAD, which works independently from the other products. Not every step of the installation is stated in this manual. For instance, "Software License Agreement" steps are omitted. When no specific information is given on a certain step, please use the default options and proceed.

The remaining of this installation guide is divided as follows:

- DB2 Installation
- Preparing for Websphere Application Server and Websphere MQ Installation
- Websphere MQ Installation
- IBM HTTP Server Installation
- Websphere Application Server Installation
- Websphere Application Developer Installation

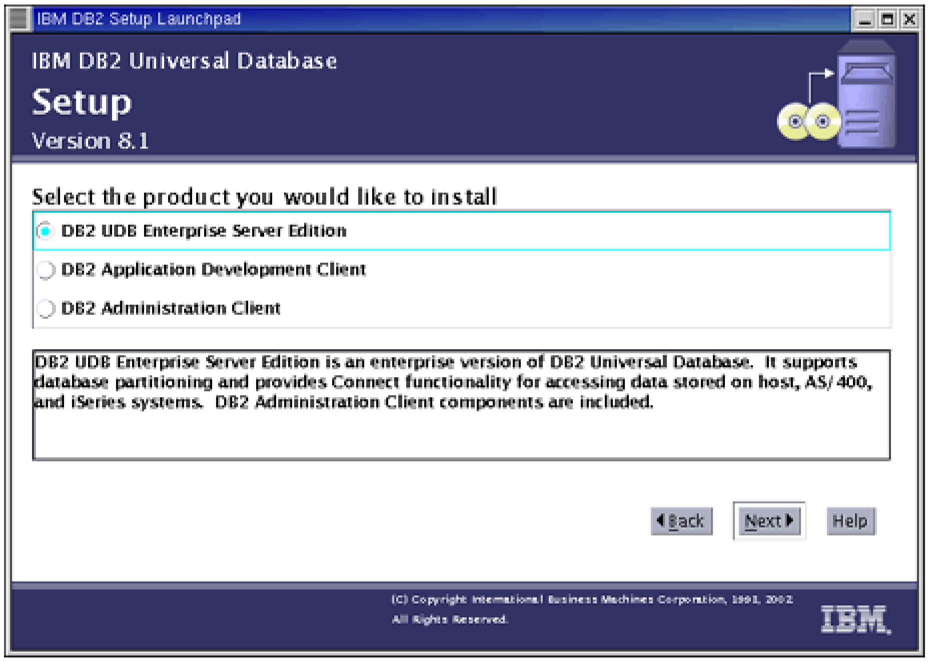
DB2 Installation

- Verify that the pdksh package is installed, otherwise install it from the Redhat 8 CD
[root@box3 root]# rpm -q pdksh

- As root, create a temporary directory and unzip the DB2 zip file
[root@box3 temp]# unzip db81eeux.zip

- Run db2setup
[root@box3 temp]# ./db2setup

- Choose Install Products and select DB2 UDB Enterprise Server Edition



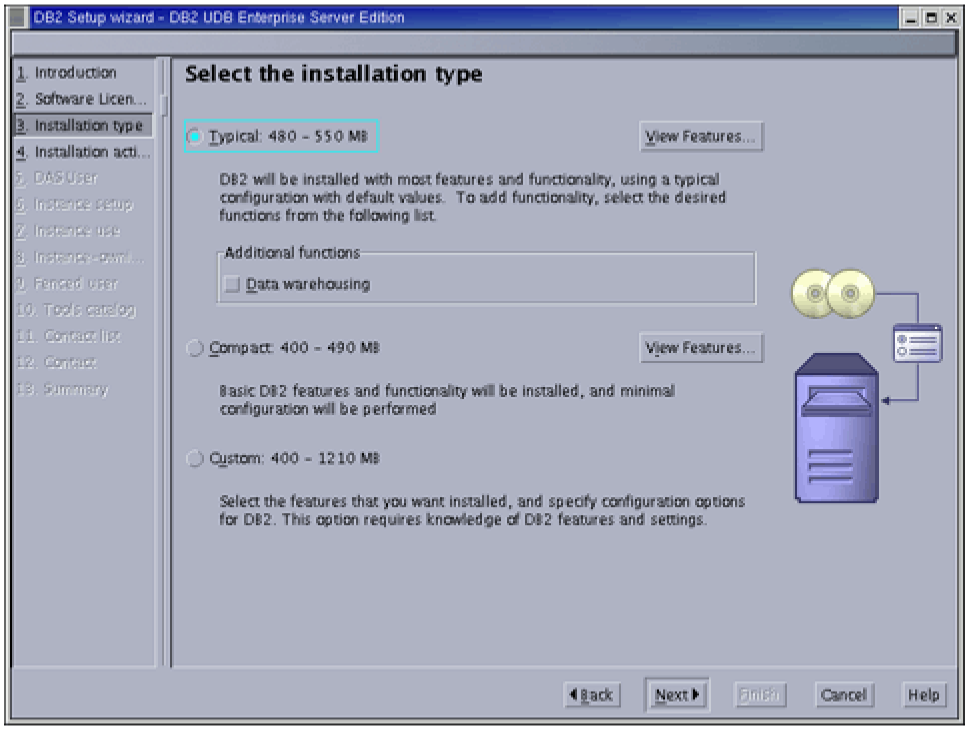
Select the product you would like to install

[✓ DB2 UDB Enterprise Server Edition]
DB2 Application Development Client
DB2 Administration Client

DB2 UDB Enterprise Server Edition is an enterprise version of DB2 Universal Database. It supports database partitioning and provides Connect functionality for accessing data stored on host, AS/400, and iSeries systems. DB2 Administration Client components are included.

(C) Copyright International Business Machines Corporation, 1993, 2002
All Rights Reserved.

- Select Typical installation



Select the installation type

Typical: 480 - 550 MB

DB2 will be installed with most features and functionality, using a typical configuration with default values. To add functionality, select the desired functions from the following list.

Additional functions

Data warehousing

Compact: 400 - 490 MB

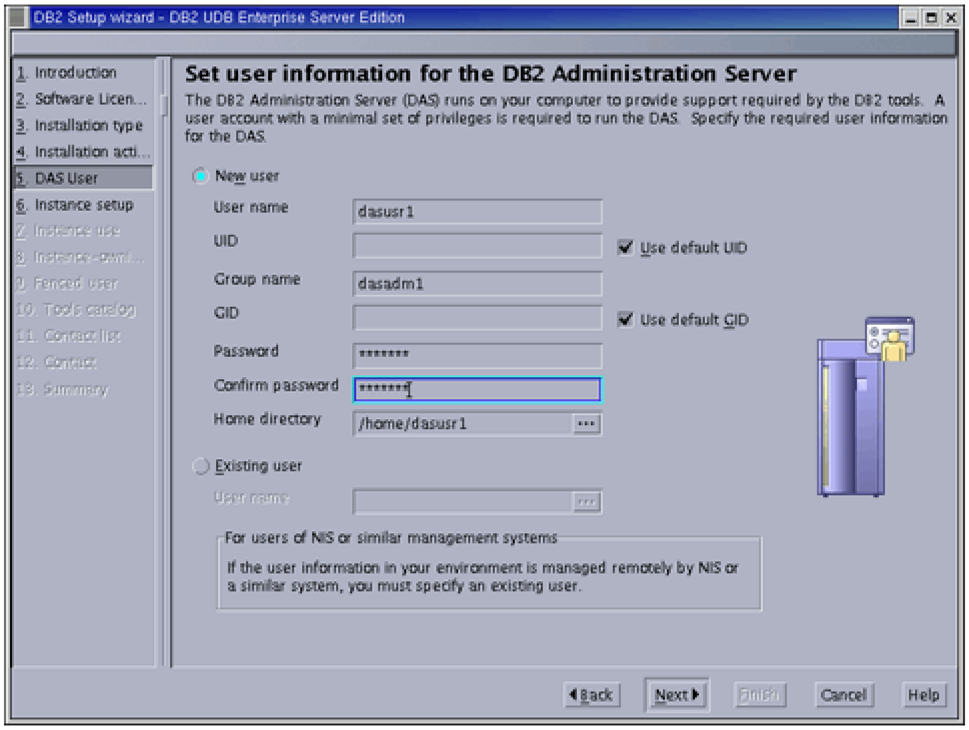
Basic DB2 features and functionality will be installed, and minimal configuration will be performed

Custom: 400 - 1210 MB

Select the features that you want installed, and specify configuration options for DB2. This option requires knowledge of DB2 features and settings.

- Select Single-partition instance

- For the user setup screens, leave all default options, typing only the passwords


DB2 Setup wizard - DB2 UDB Enterprise Server Edition

Set user information for the DB2 Administration Server

The DB2 Administration Server (DAS) runs on your computer to provide support required by the DB2 tools. A user account with a minimal set of privileges is required to run the DAS. Specify the required user information for the DAS.

New user
User name: dasusr1
UID: [✓ Use default UID]
Group name: dasadm1
GID: [✓ Use default GID]
Password: *******
Confirm password: *******
Home directory: /home/dasusr1

Existing user
User name

For users of NIS or similar management systems
If the user information in your environment is managed remotely by NIS or a similar system, you must specify an existing user.

- Select Create a DB2 Instance

- Select Do Not Prepare the DB2 Tools...

- On the Set Up the Administration Contact List screen, select local and deselect Enable notification. Click OK when warning pops-up.

- Select Defer this task until after installation is complete

- Click Finish. Installation should complete without any errors.

DB2 Post Setup

•*Checking and changing Kernel parameters*

The usual default values for message queues (msgmni) and semaphore array parameters are not ideal for DB2 to run properly.
- Checking the parameters

```
[root@box3 root]# ipcs -l
```

- It is recommended that the following 2 lines are added to /etc/sysctl.conf

```
kernel.msgmni = 512
kernel.sem = 250 128000 32 512
```

- Use this command to load the settings
[root@box3 root]# sysctl -p

·*Installing the Sample Database*

The user db2inst1 needs access to the X server.
- You may use the following command if in a secured network

```
[root@box3 root]# xhost +
access control disabled, clients can connect from any host
```

- Starting the DB2 First Steps

```
[root@box3 root]# su - db2inst1
[db2inst1@box3 db2inst1]$ cd sqllib/bin/
[db2inst1@box3 bin]$ ./db2fs
```

- Click on Create Sample Database
- Check DB2 UDB sample, hit OK.

- Exit First Steps

·*Checking the installation*

```
[db2inst1@box3 db2inst1]$ db2start
10-07-2003 21:11:01     0   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

- You may additionally use the db2jstrt command to allow remote connections

```
[db2inst1@box3 db2inst1]$ db2level
DB21085I  Instance "db2inst1" uses "32" bits and DB2 code release "SQL08010"
with level identifier "01010106".
Informational tokens are "DB2 v8.1.0.0", "s021023", "", and FixPak "0".
Product is installed at "/opt/IBM/db2/V8.1".
```

```
[db2inst1@box3 db2inst1]$ db2
          ...
          ...
db2 => list database directory

 System Database Directory

 Number of entries in the directory = 1

Database 1 entry:

 Database alias                       = SAMPLE
 Database name                        = SAMPLE
 Local database directory             = /home/db2inst1
 Database release level               = a.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0

db2 => connect to sample

   Database Connection Information

 Database server        = DB2/LINUX 8.1.0
 SQL authorization ID   = DB2INST1
 Local database alias   = SAMPLE

db2 => select * from employee
db2 => quit
DB20000I  The QUIT command completed successfully.
```

Control Center

The Control Center is a Graphical User Interface for DB2 and allow users to create, access, and manipulate databases. Please remember that the user db2inst1 needs access to X server.
- The following command starts DB2 Control Center

```
[db2inst1@box3 db2inst1]$ db2cc
```

Preparing for Websphere Application Server and Websphere MQ installation

WAS and MQ require some specific user and group setup prior to installation

- Create the following groups and user with these commands:

```
[root@box3 root]# groupadd mqm
[root@box3 root]# groupadd mqbrkrs
[root@box3 root]# useradd -g mqm -G mqbrkrs -m mqm
```

- Add user root to both mqm and mqbrkrs groups. This can be done using Redhat User Manager GUI.

- Create a password for user mqm

```
[root@box3 root]# passwd mqm
```

- Log off and back on

Websphere MQ Installation

- Unzip the supplied MQ zip file

```
[root@box3 temp]# unzip mqs531ux.zip
```

- The licensing program must be run prior to installation

```
[root@box3 temp]# ./mqlicense.sh
```

- Install the following packages in this order:

```
[root@box3 temp]# rpm -Uvh MQSeriesRuntime-5.3.0-1.i386.rpm
[root@box3 temp]# rpm -Uvh MQSeriesSDK-5.3.0-1.i386.rpm
[root@box3 temp]# rpm -Uvh MQSeriesServer-5.3.0-1.i386.rpm
[root@box3 temp]# rpm -Uvh MQSeriesClient-5.3.0-1.i386.rpm
[root@box3 temp]# rpm -Uvh MQSeriesSamples-5.3.0-1.i386.rpm
[root@box3 temp]# rpm -Uvh MQSeriesMan-5.3.0-1.i386.rpm
```

All the necessary MQ packages are now installed.

IBM HTTP Server Installation

The Websphere Application Server comes bundled with the IBM HTTP Server version 1.3.26. However, this included software is not compatible with Redhat 8. IBM recommends that the IBM HTTP Server V2.0 is installed separately.
Make sure DB2 is installed. This ensures that the IBM's Java SDK is installed.

- Update your PATH environment variable.

```
export PATH=/opt/IBMJava2-131/jre/bin:$PATH
```

- Load changes and check Java version

```
[root@box3 root]# java -fullversion
java full version "J2RE 1.3.1 IBM build cxia32131-20020622"
```

- Unzip the supplied MQ zip file and c to the created directory

```
[root@box3 temp]# unzip HTTPServer.linux.2042.tar
[root@box3 temp]# cd IHS-2.0.42/
```

- Start the installation

```
[root@box3 IHS-2.0.42]# java -jar setup.jar
```

Accept defaults and choose a typical install. The installation should complete without any errors.

Due to some incompatibility issues, there might be some broken links that need to be fixed.

- Use the following command to verify links:

```
[root@box3 IHS-2.0.42]#  ls -l /usr/lib/libgsk*
```

If the links are broken, they should be highlighted in red and blinking.

- Use the following command to fix it:

```
[root@box3 IHS-2.0.42]# ln -sf /usr/local/ibm/gsk5/lib/* /usr/lib
```

The HTTP server is now installed and ready for use.

Websphere Application Server Installation

- Unzip the supplied WAS zip file

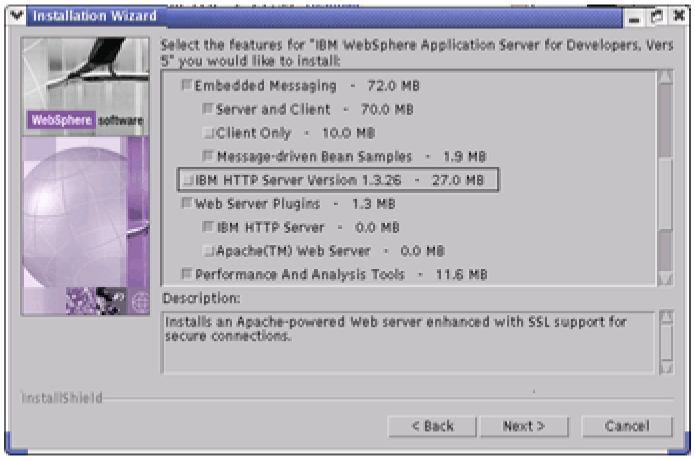
```
[root@box3 temp]# unzip wsnd5ahu.zip

[root@box3 temp]# cd linuxi386/
```

- Launch the installer

```
[root@box3 linuxi386]# ./LaunchPad.sh
```

- Choose Install the product, select Custom Install.

- Deselect IBM HTTP Server Version 1.3.26 as version 2.0 is already installed. Leave all other default options.



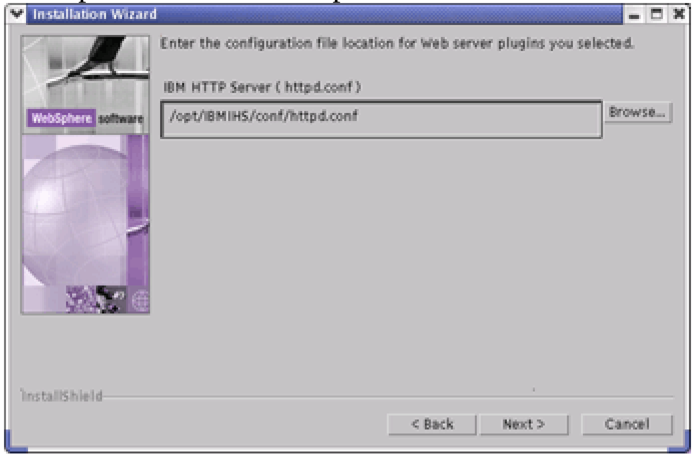
Select the features for "IBM WebSphere Application Server for Developers, Vers 5" you would like to install:

- Embedded Messaging - 72.0 MB
  - Server and Client - 70.0 MB
  - Client Only - 10.0 MB
  - Message-driven Bean Samples - 1.9 MB
- IBM HTTP Server Version 1.3.26 - 27.0 MB
- Web Server Plugins - 1.3 MB
  - IBM HTTP Server - 0.0 MB
  - Apache(TM) Web Server - 0.0 MB
- Performance And Analysis Tools - 11.6 MB

Description:

Installs an Apache-powered Web server enhanced with SSL support for secure connections.

- When asked about the web server httpd.conf file, please provide its path. The default DB2 installation uses /opt/IBMIHS/conf/httpd.conf



Enter the configuration file location for Web server plugins you selected.

IBM HTTP Server (httpd.conf)

/opt/IBMIHS/conf/httpd.conf

The installation takes a while depending on the system. Upon its completion, the First Steps window is launched. Click Start the Server. After a period of time, the First Steps console should display Server Server1 open for e-business. This indicates that the server is started. Additionally, click Verify the Installation to make sure the installation completed properly.
The application server is now fully installed. However, the HTTP server needs further configuration adjustments.

Specifying the correct plugin module in the httpd.conf file

The Websphere Application Server added four lines to the httpd.conf file specified in the installation. The application server, however, is not aware that the HTTP server is V2.0 instead of the supplied V1.3.26. The correct plugin must be manually changed.

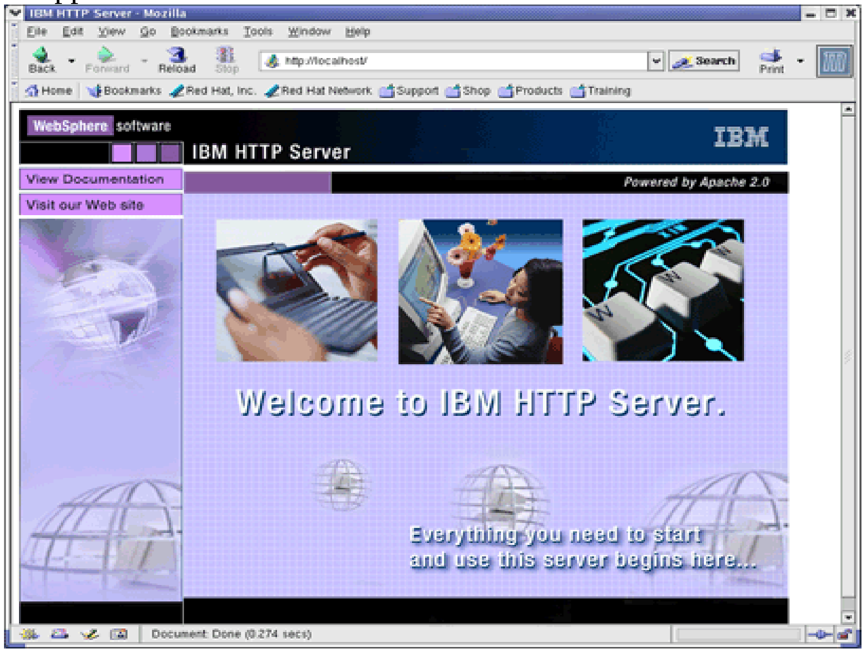
- Open the httpd.conf file (/opt/IBMIHS/conf/httpd.conf)
- At the end of the file, look for the “LoadModule” line, comment it out replacing with:
LoadModule was_ap20_module /opt/Websphere/AppServer/bin/mod_was_ap20_http.so
(note that the above entry was split in two lines in this document. It should be in one line at the httpd.conf file. Follow the format of the “LoadModule” commented-out line)

Starting the Websphere Environment

Starting the HTTP server

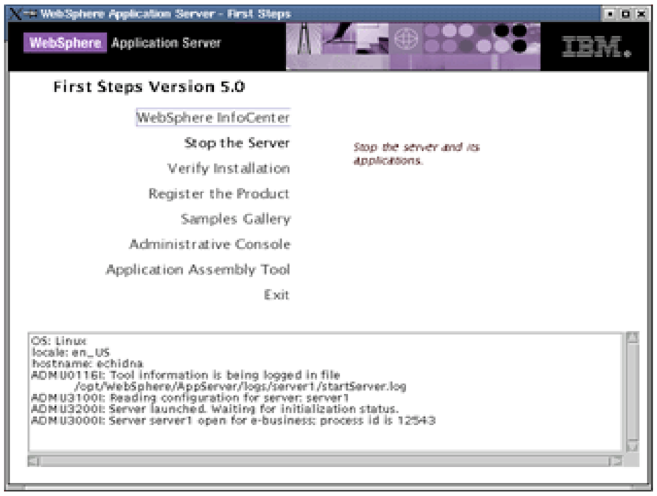
[root@box3 bin]# pwd
/opt/IBMIHS/bin
[root@box3 bin]# ./apachectl start

- Open a browser and enter http://localhost in the URL field. The following welcome screen should appear


IBM HTTP Server - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost/
Search
Print
Home Bookmarks Red Hat, Inc. Red Hat Network Support Shop Products Training
WebSphere software
IBM HTTP Server
IBM
View Documentation
Visit our Web site
Powered by Apache 2.0
Welcome to IBM HTTP Server.
Everything you need to start
and use this server begins here...
Document: Done (0.274 secs)

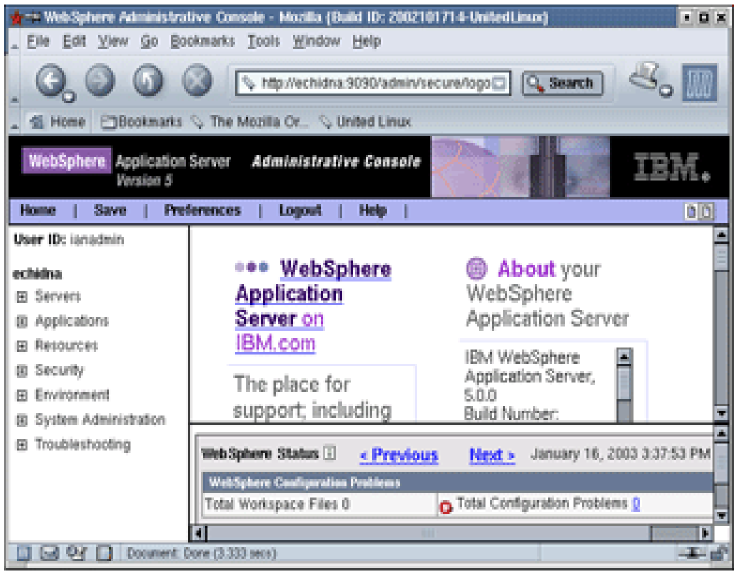
Starting the application server

- If the application server is not started yet, it can be launched manually using
[root@box3 root]# /opt/WebSphere/AppServer/bin/firststeps.sh

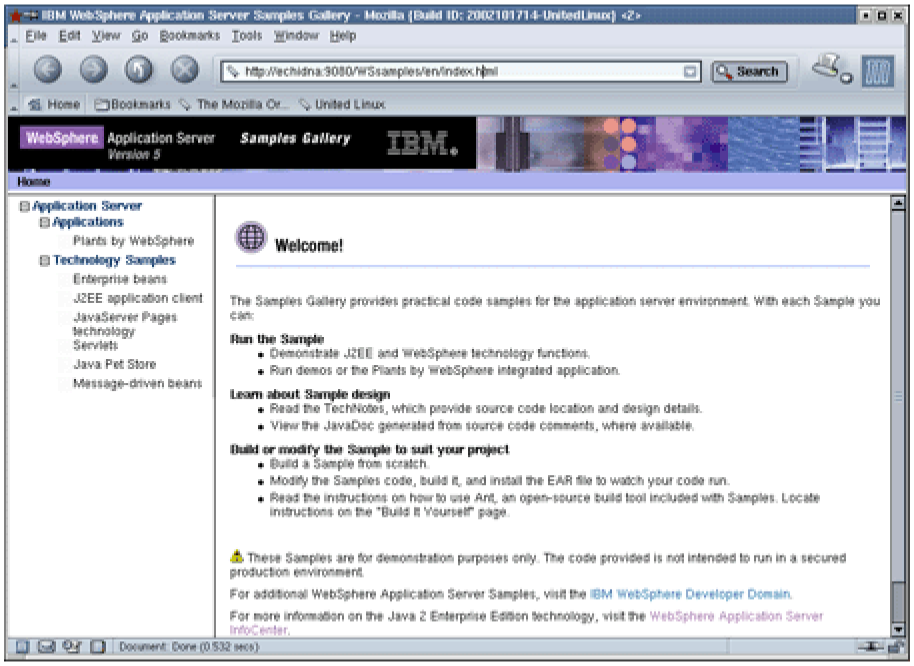


```
OS: Linux
locale: en_US
hostname: echidna
ADMU0116I: Tool information is being logged in file
        /opt/WebSphere/AppServer/logs/server1/startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 12543
```

- Click Start the Server. Once the console displays " open for e-business ..." , you may use your browser to access the Administrative Console
- Type http://localhost:9090/admin


WebSphere Administrative Console - Mozilla {Build ID: 2002101714-UnitedLinux}
File Edit View Go Bookmarks Tools Window Help
http://echidna:9090/admin/secure/logo
Search
Home Bookmarks The Mozilla Or... United Linux
WebSphere Application Server Version 5 Administrative Console
IBM.
Home | Save | Preferences | Logout | Help |
User ID: ianadmin
echidna
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
The place for support, including
About your WebSphere Application Server
IBM WebSphere Application Server, 5.0.0 Build Number:
WebSphere Status
« Previous
Next »
January 16, 2003 3:37:53 PM
WebSphere Configuration Problems
Total Workspace Files 0
Total Configuration Problems 0
Document: Done (3.333 secs)

You may also check the included samples
- Type http://localhost/WSsamples


IBM WebSphere Application Server Samples Gallery - Mozilla {Build ID: 2002101714-UnitedLinux} <2>
File Edit View Go Bookmarks Tools Window Help
http://echidna:9080/WSsamples/en/index.html
Search
Home Bookmarks The Mozilla Or... United Linux
WebSphere Application Server Version 5
Samples Gallery
IBM.
Home
Application Server
Applications
Plants by WebSphere
Technology Samples
Enterprise beans
J2EE application client
JavaServer Pages technology
Servlets
Java Pet Store
Message-driven beans
Document: Done (0.532 secs)

Welcome!

The Samples Gallery provides practical code samples for the application server environment. With each Sample you can:

Run the Sample
- Demonstrate J2EE and WebSphere technology functions.
- Run demos or the Plants by WebSphere integrated application.

Learn about Sample design
- Read the TechNotes, which provide source code location and design details.
- View the JavaDoc generated from source code comments, where available.

Build or modify the Sample to suit your project
- Build a Sample from scratch.
- Modify the Samples code, build it, and install the EAR file to watch your code run.
- Read the instructions on how to use Ant, an open-source build tool included with Samples. Locate instructions on the "Build It Yourself" page.

These Samples are for demonstration purposes only. The code provided is not intended to run in a secured production environment.

For additional WebSphere Application Server Samples, visit the IBM WebSphere Developer Domain.

For more information on the Java 2 Enterprise Edition technology, visit the WebSphere Application Server InfoCenter.

The installation of DB2, Websphere MQ, and Websphere Application Server is now complete. This environment is now ready to handle full J2EE applications.

Websphere Studio Application Developer Installation

WSAD is an integrated development environment that works with several technologies including EJBs, Servlets, JSPs and Struts. This IDE also incorporates a fully functional Websphere Application Server and Cloudscape database. In essence, the developer using WSAD is able to fully develop, test, and deploy J2EE applications in a single environment.

- On a Windows 2000 or XP machine, unzip the provided WSAD zip file using any zip tool such as WinZip
- Make sure no application server or HTTP server is running
- Click the setup.exe icon
- Choose Install IBM Websphere Studio Application Developer
- Select Custom Installation
- Installing only the Websphere Application Server v5.0 as a Run-Time Environment is sufficient
- You may additionally select Examples for Eclipse Plug-in Development and Rational ClearCase instead of the default CVS.

The installation takes a while and initially seems to be halted. Once installation is completed, start WSAD using the Windows Start menu. A window prompts for a workspace directory. This is where the application specific files are stored. You may choose a default workspace location for all your projects, or better yet, determine a different workspace for each project.

One way to manage multiple workspaces is to create a different Windows shortcut for each different project. This can be accomplished by dragging a WSAD (wsappdev.exe) shortcut to the desktop, right-click the shortcut icon and select Properties, add -data c:\directory\of\project -showlocation to the Target textbox. Click OK

Where to go from here?

The Websphere environment is quite complex and can be intimidating initially. Fortunately, there are several online resources that can aid in the understanding and usage of all Websphere tools. These online tutorials show how to write a simple Java class to full J2EE applications. Here are some examples:

http://www7b.software.ibm.com/wsdd/

http://www.devx.com/ibm

http://www.webspherecentral.com/

http://www-106.ibm.com/developerworks/