

Hello World: WebSphere Application Server and Application Server Toolkit, V6.1

Publish an enterprise application

Skill Level: Intermediate

[Larina Vera \(larinac@ca.ibm.com\)](mailto:larinac@ca.ibm.com)
Information Developer
IBM Toronto Lab

26 Sep 2006

Updated 29 Nov 2006

Welcome to the fourth tutorial in the ["Hello, World" series](#), which provides high-level overviews of various IBM® software products. This tutorial places you in the role of an IT department administrator who receives Enterprise Java™ Beans (EJBs) and Web modules from the development team. Your responsibility is to assemble, deploy, and manage these modules as a J2EE application on a WebSphere® Application Server using Application Server Toolkit. The tutorial provides practical exercises that teach you how to complete these tasks.

Section 1. Before you start

About this series

This series is for novices who want high-level overviews of IBM software products. The modules are designed to introduce the products and draw your interest for further exploration. The exercises cover only the basic concepts, but are enough to get you started.

About this tutorial

This tutorial provides a high-level overview of WebSphere Application Server Toolkit and WebSphere Application Server. It gives you the opportunity to practice using it with hands-on exercises. Step-by-step instructions show how to package a J2EE application, explore the deployment descriptor files, test and publish the application, and maintain the application on the server.

Objectives

After completing this tutorial, you should understand how to:

- Assemble and publish J2EE applications on a WebSphere Application Server using WebSphere Application Server Toolkit
- Manage the J2EE application on the server using the WebSphere Application Server administrative console
- Read WebSphere Application Server log files to assist in problem determination of the server

Prerequisites

This tutorial is for J2EE application administrators at a beginning level who have a general familiarity with using the Eclipse workbench.

Download and extract the [HelloWorldEJB.jar](#) and [HelloWorldWeb.war](#) files on the computer on which you want to run this exercise.

System requirements

To run the examples in this tutorial, install [WebSphere Application Server V6.1](#) and WebSphere Application Server Toolkit V6.1. WebSphere Application Server Toolkit is provided to customers who purchase WebSphere Application Server. In the CD package, WebSphere Application Server Toolkit V6.1 is available on a separate CD in the WebSphere Application Server V6.1 CD package. Application Server Toolkit is also available on a separate DVD.


The hardware and software requirements for the WebSphere Application Server products are available at the following Web address:

<http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html>

To view the demos included in this tutorial, enable JavaScript in your browser and install Macromedia Flash Player 6 or higher. Download the latest Flash Player at <http://www.macromedia.com/go/getflashplayer/>.

Animated demos

If this is your first encounter with a developerWorks tutorial that includes demos, here are a few helpful hints:

- Demos are an optional way to see the same steps described in the tutorial. To see an animated demo, click the  Show me link. The demo opens in a new browser window.
- Each demo contains a navigation bar at the bottom of the screen. Use the navigation bar to pause, exit, rewind, or fast forward portions of the demo.
- The demos are 800 x 600 pixels. If this is the maximum resolution of your screen, or if your resolution is lower than this, scrolling is necessary to see some areas of the demo.

Section 2. Introduction

About WebSphere Application Server V6.1

WebSphere Application Server is Web application server software that runs with a Web server and is used to deploy, integrate, execute, and manage Java 2 Platform, Enterprise Edition (J2EE) applications.

The *Web server* is IBM HTTP Server, which is based on the Apache HTTP Server (see [Resources](#)) developed by the Apache Software Foundation. It enables a computer that uses the Hypertext Transfer Protocol (HTTP) to serve objects by responding to requests from other programs, such as Web browsers.

Here are some of the common technologies supported by WebSphere Application Server:

- | | |
|--|---|
| <ul style="list-style-type: none">• Java• Web | <ul style="list-style-type: none">• Java Message Service (JMS)• JavaServer Faces (JSF) |
|--|---|

- Servlets
- JavaServer Pages (JSPs)
- Enterprise beans (EJBs)
- Java 2 Platform, Enterprise Edition (J2EE)
- J2EE Connector (J2C)
- Extensible Markup Language (XML)
- Java Database Connectivity (JDBC)
- Java Management Extensions (JMX)
- Web Services

Communicate with a WebSphere Application Server by using an administrative client. Examples of administrative clients are the WebSphere Application Server administrative console and WebSphere Application Server Toolkit, which have graphical user interfaces (GUI). The WebSphere administrative console is a Web-based tool, whereas the WebSphere Application Server Toolkit, discussed in more detail in the next section, is an Eclipse-based tool. The primary use of WebSphere administrative console is for changing and controlling the behavior of the server, whereas WebSphere Application Server Toolkit is for developing, assembling, publishing, and configuring server resource for J2EE application target for the WebSphere Application Server. A non-graphical alternative to control the behavior of the server is the WebSphere administrative tool (wsadmin), which is primarily used for scripting.

About WebSphere Application Server Toolkit V6.1

WebSphere Application Server Toolkit is provided to customers who purchase WebSphere Application Server. Essentially, WebSphere Application Server Toolkit is a GUI tool for developing and assembling J2EE applications into application modules, modifying J2EE deployment descriptors, and installing the application on a WebSphere Application Server. It is built using Eclipse v3.1.2 technology that is based on the Web Tools Platform v1.0.2. The WebSphere Application Server Toolkit also provides a familiar interface for developers experienced with IBM Rational® Software Development Platform products. Those products extend the capabilities of WebSphere Application Server Toolkit by supporting Unified Modeling Language (UML) modeling, Struts development, Web diagrams for Web development, and others.

Section 3. How does WebSphere fit into SOA?

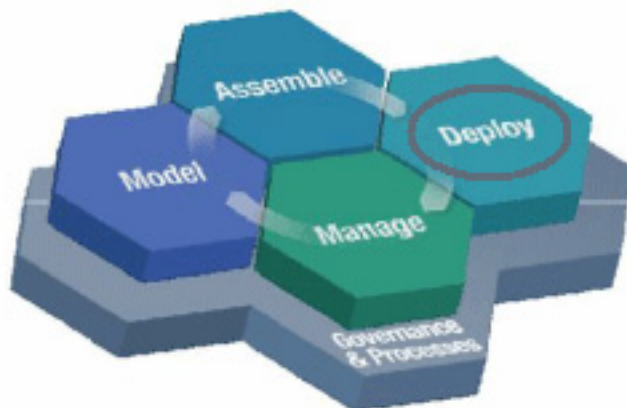
A service-oriented architecture (SOA) is a collection of services that communicate

with each other. A *service* is an individual application that can perform some task, such as updating a personnel record, performing a bank transaction, or retrieving information from a database. The services are self-contained and do not depend on the context or state of the other service. They work within distributed systems architecture.

SOA helps customers increase the flexibility of their business processes, strengthen their underlying IT infrastructure, and retain and reuse their existing assets.

Getting started with SOA is easy with the IBM SOA Foundation. *IBM SOA Foundation* is a carefully selected set of integrated and open-standards-based software from the leading-edge IBM software portfolio. The IBM SOA Foundation supports each stage of the SOA life cycle, which includes four stages: model, assemble, deploy and manage. Underpinning all of these life-cycle stages are governance and processes that provide guidance and oversight for the SOA project.

Figure 1. SOA life cycle



IBM WebSphere Application Server is part of the IBM SOA Foundation and supports the deploy phase of the SOA life cycle. Many software products in the deploy phase of the IBM SOA foundation are built on top of WebSphere Application Server. In addition, WebSphere Application Server can easily be integrated with software from the other three stages of the SOA life cycle. This allows for easy interoperability as the customer's requirements grows.

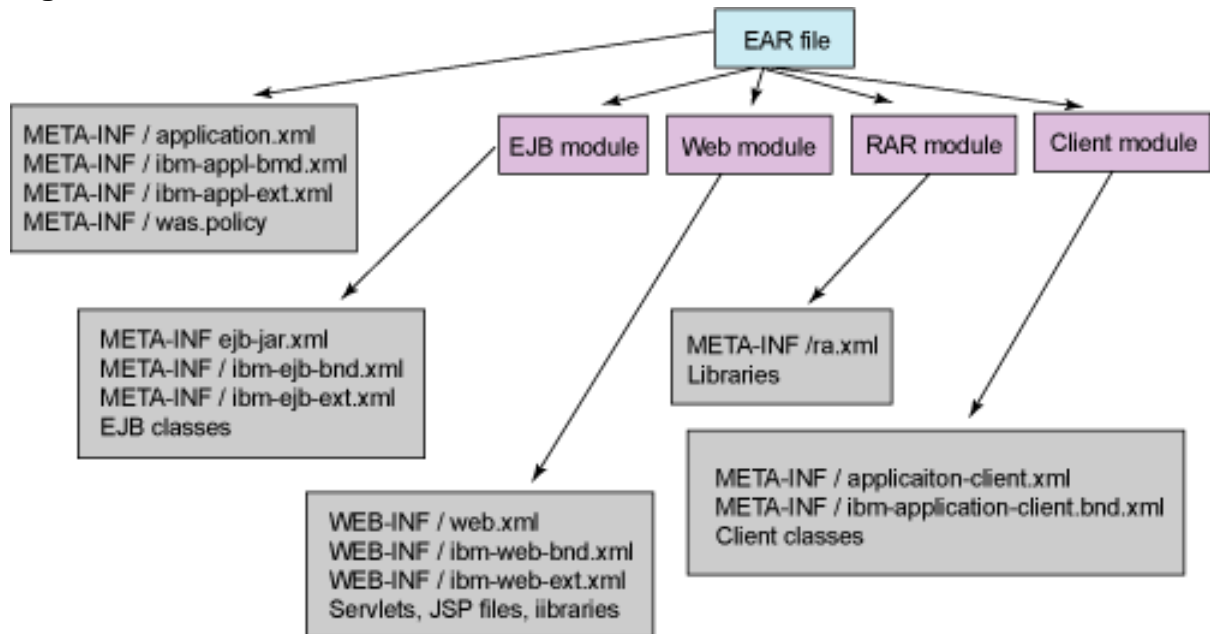
WebSphere Application Server is a key IBM middleware software offering. *Middleware* software connects two separate applications and passes data between them. For example, WebSphere Application Server is a type of middleware that can link a database system to a Web server.

Section 4. Basic concepts

What is Java 2 Platform, Enterprise Edition (J2EE)?

J2EE defines a standard for developing and deploying enterprise applications.

Figure 2. J2EE architecture



An enterprise application called Enterprise archive file (.ear) is composed of one or more of the following J2EE modules:

- *EJB module* is used to assemble enterprise beans, into a single deployable unit called EJB archive file (.jar).
- *Web module* is used to assemble servlets, JSP files, Web pages and other static content into a single deployable unit, called Web archive file (.war).
- *Connector module* is used to assemble resource adaptors that let an application access a resource such as data or an application on a remote server, Enterprise Information System (EIS) using the Java 2 Connector (J2C) architecture into a single deployable unit, called resource adapter archive file (.rar).
- *Application Client module* is used to assemble the files that make up the application into a single Java archive file (.jar).

WebSphere Application Server Toolkit can be used to modify the deployment descriptor information, binding information, and IBM extensions of each module as well as the enterprise application.

Section 5. Beginning your role as an administrator for a WebSphere Application Server

In this Hello, World! Series tutorial, you play the role of an administrator in the IT department of a company. As an administrator your primary tasks are application assembly, deployment, and server resource configuration. The following sections guide you through a series of exercises that show you how to complete those tasks. Each section, or exercise, is a continuation of the previous exercise, so you need to complete the first one before continuing on to the next.

In the scenario designed for these exercises, the development department in your company gives you an EJB module (HelloWorldEJB.jar) and a Web module (HelloWorldWeb.war). Your goal is to publish the application on your company's server. *Publishing* involves copying files (application, resource files, and deployment descriptor files) to the correct location for the server to find and use them.

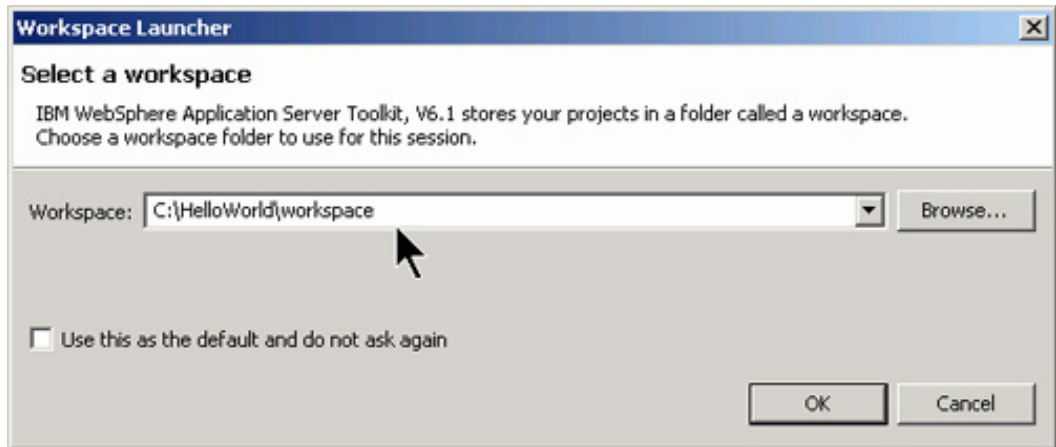
The HelloWorld application is a simple J2EE application that can add a user's name and phone number. In addition, the application lists all the user names and phone numbers that have been successfully added using the HelloWorld application.

Download and extract the [HelloWorldEJB.jar](#) and [HelloWorldWeb.war](#) files on the computer where you will perform the exercises.

Although this Hello World tutorial focuses on the use of Application Server Toolkit for performing the role of an administrator, the tools in Application Server Toolkit are not restricted to administrative tasks; you can also use it to develop applications for WebSphere Application Server.

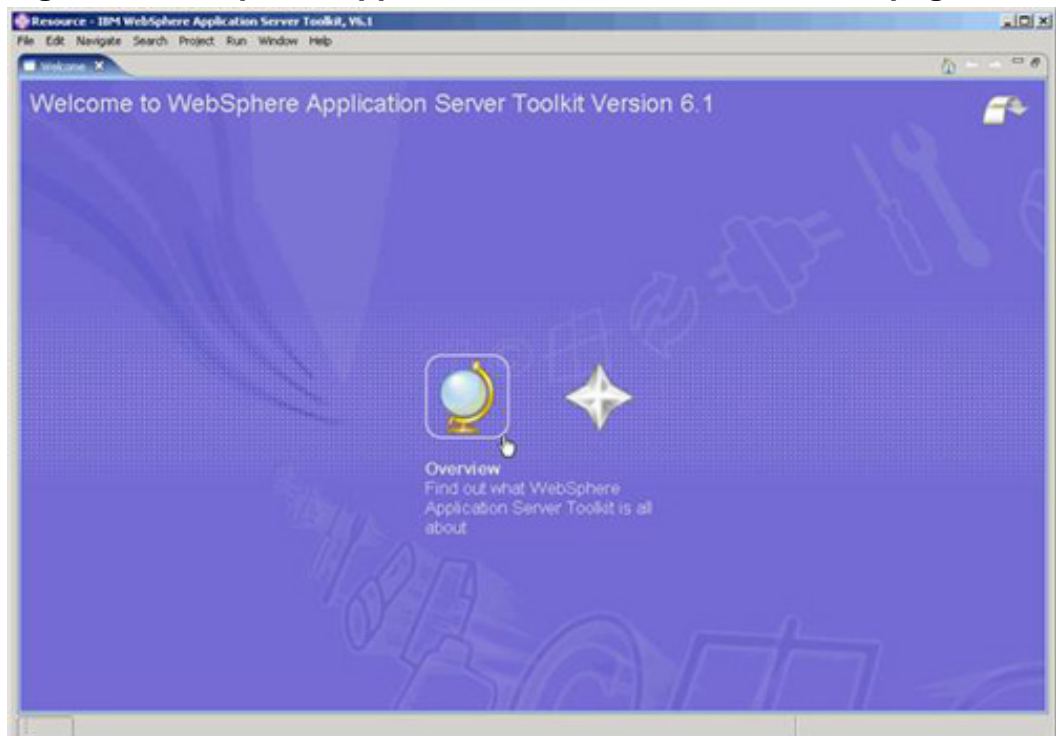
Begin the tasks of an administrator by working with the Application Server Toolkit:

1. Open WebSphere Application Server Toolkit by selecting **Start > Programs > IBM WebSphere > Application Server Toolkit V6.1 > Application Server Toolkit**.
 2. A window displays asking for the workspace directory. A *workspace* specifies the location on your file system to host your assembly activities. In the workspace field, type `C:\HelloWorld\workspace` and click **OK**.
- Figure 3. Select a workspace window**



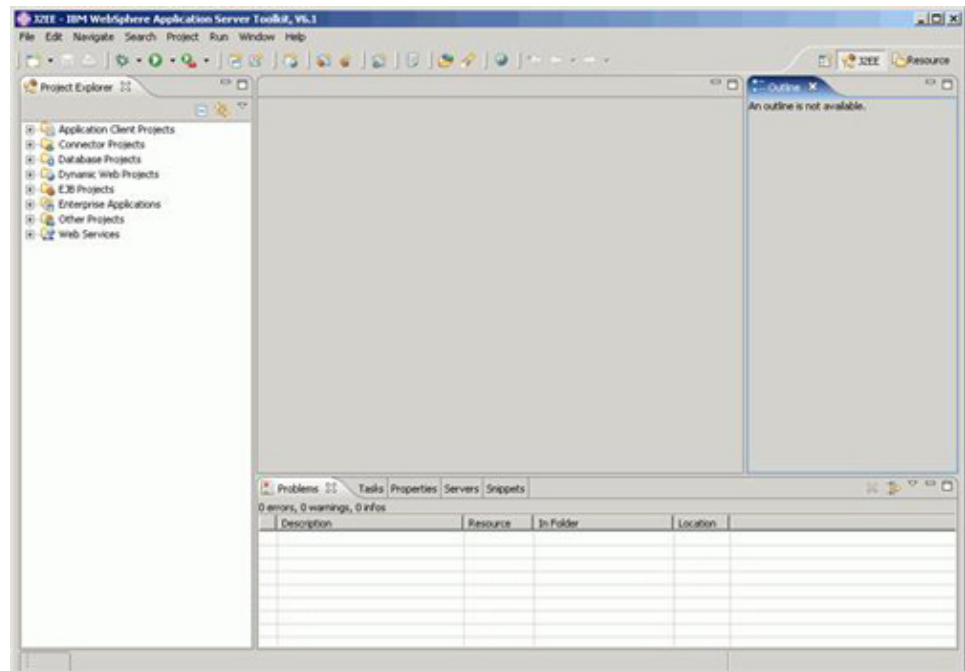
3. After specifying a workspace, the Welcome page opens. The Welcome page provides links to the product overview and what is new in Application Server Toolkit. Close the Welcome page when you are done exploring.

Figure 4. WebSphere Application Server Toolkit Welcome page



4. The Resource perspective displays by default. A *perspective* defines the initial set and layout of views in the Workbench. Each perspective provides a set of functionality aimed at accomplishing a specific type of task or works with specific types of resources. In this case, you are going to work with J2EE resources. Switch to the J2EE perspective.


- a. In the toolbar, click **Windows > Open Perspective > Other > J2EE**.
- b. Click **OK**. The J2EE perspective displays:
Figure 5. J2EE perspective



Section 6. Packaging a J2EE application

This exercise shows you how to create an enterprise application to contain an EJB module (HelloWorldEJB.jar) and a Web module (HelloWorldWeb.war). The HelloWorldEJB.jar and HelloWorldWeb.war files were provided to you from the development department of your company and you are now asked to package the application to publish on your company's server. The first step of this task is to package the files into a J2EE application.

Would you like to see these steps demonstrated for you?

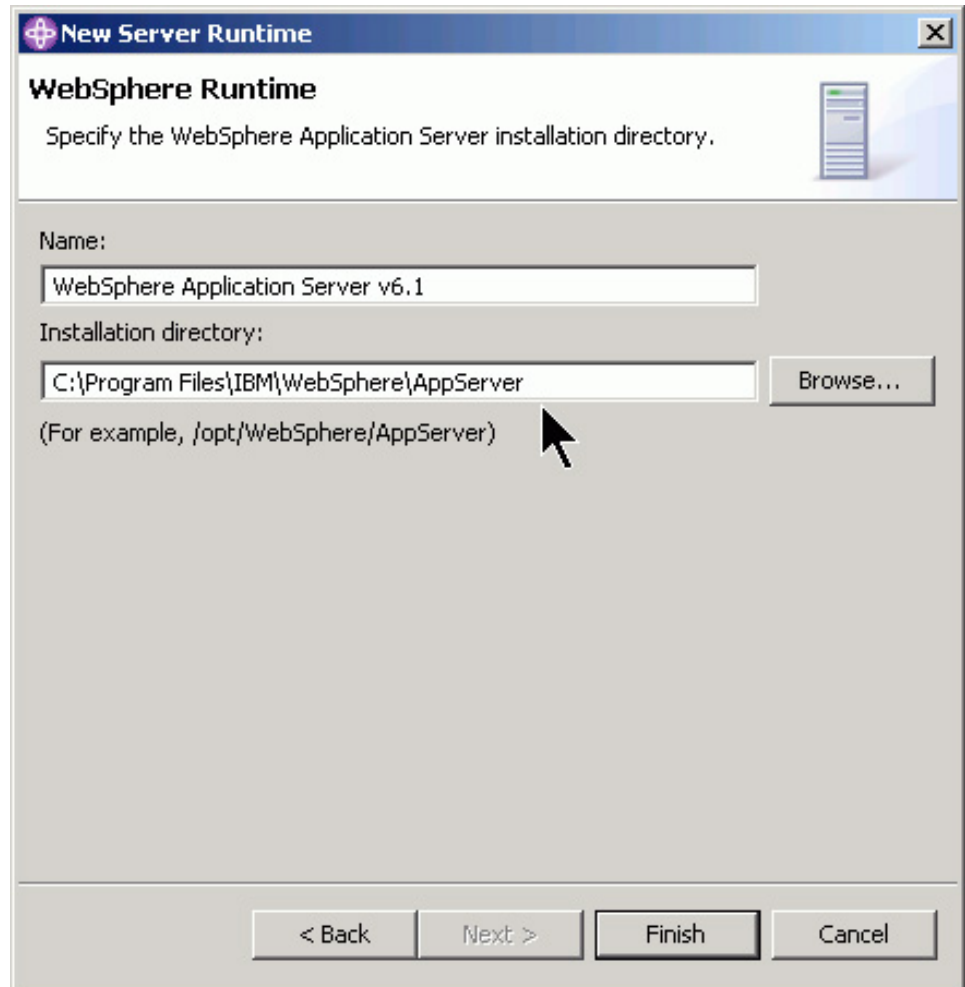
 [Show me](#)

Creating a new enterprise application

Create an enterprise application that contains the EJB and Web modules provided by your developers:

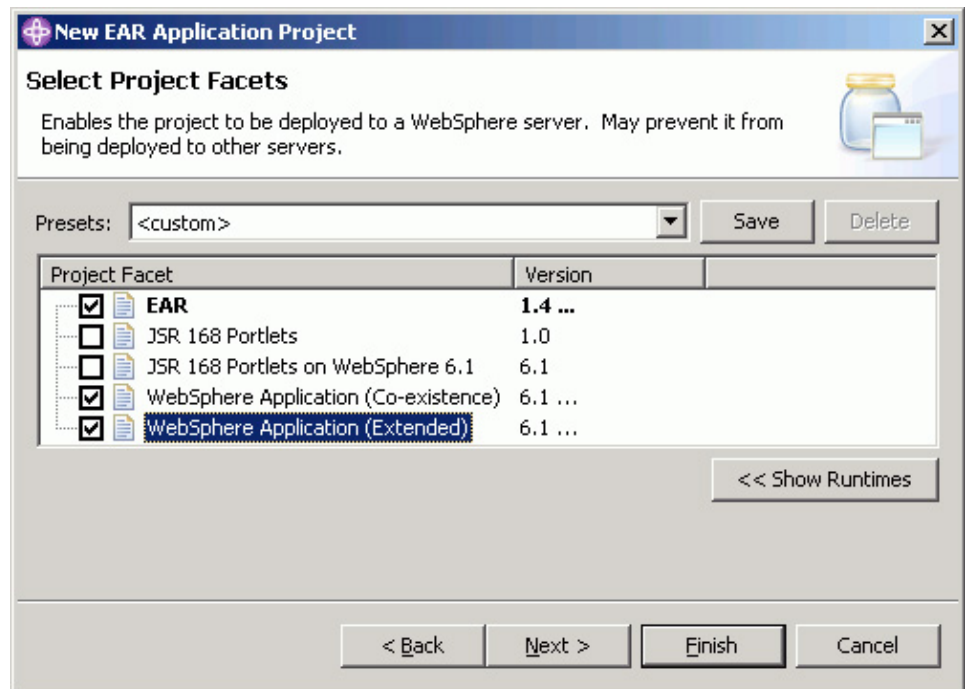
1. In the toolbar, select **File > New > Other**. The New wizard opens.
2. Select **J2EE > Enterprise Application Project**. Click **Next**.
3. In the Project Name field, type `HelloWorldEAR` as the project name.
4. Specify the server you plan to run your application on.
 - a. Next to the Target runtime field, click **New**. The target runtime setting is used to set the class path for J2EE projects, which sets the proper server runtime libraries and JDK for your application to compile against. The New Server Runtime wizard opens.
 - b. By default WebSphere Application Server V6.1 is selected. Click **Next**.
 - c. In the Installation directory field, specify `WAS_INSTALL_ROOT`, which is the directory where WebSphere Application Server is installed on your local machine. For example, `C:\Program Files\IBM\WebSphere\AppServer`. Click **Finish**.

Figure 6. Targeting the runtime to WebSphere Application Server



- d. Under the Target server list, select **WebSphere Application Server V6.1**. Click **Next**. The Select Project Facets page opens.
- e. Facet defines the characteristics of a project. When you add a facet to a project you are configuring the project to perform certain tasks:

Figure 7. Exploring facets added to the enterprise application project



The EAR project facet specifies to comply with J2EE V1.4 specification level. Selecting both WebSphere Application (Co-existence) and WebSphere application (extended) enables functionality specific to the WebSphere Application Server that may not be covered in the J2EE standards. As a result the application might not run on other server vendors.

If the WebSphere Application (Co-existence) option was selected and the WebSphere Application (Extended) option was cleared, this setting should allow the application to deploy onto other server vendors.

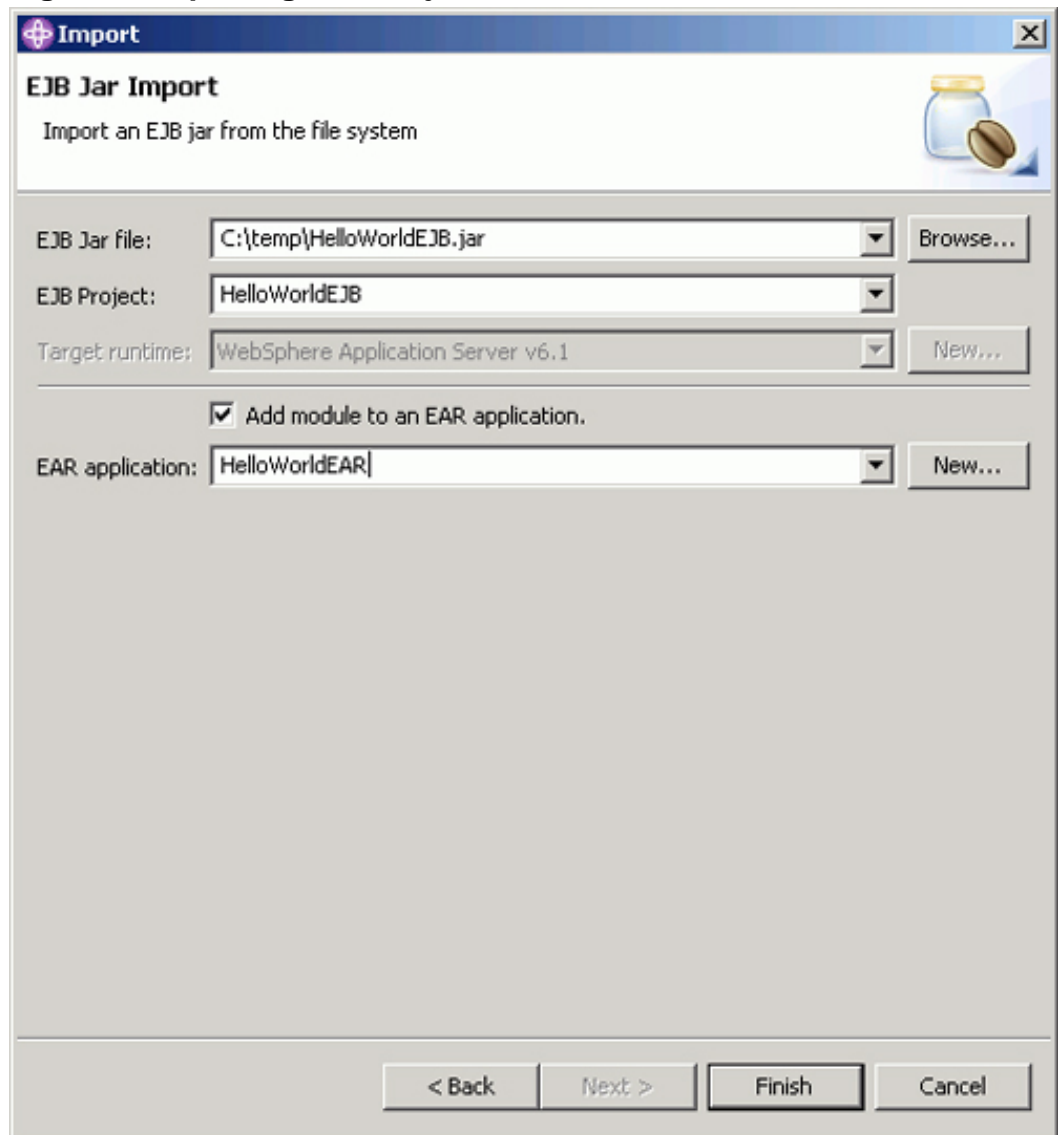
- f. Click **Next**. The J2EE Modules to Add to the EAR page opens. In the next steps, you are going to add the J2EE modules to your newly created enterprise application; this page can remain blank. Click **Finish**.
6. In the Problems view, you might see an error. It is safe to ignore this error for now because you have created an enterprise application containing no J2EE modules. In the next steps, you are going to import Web and EJB modules, which resolves this problem.

Importing the EJB module

Import the EJB module into your newly created HelloWorldEAR enterprise application.

1. In the toolbar, select **File > Import > EJB JAR file**. Click **Next**.
2. In the EJB Jar file field, specify the location where you downloaded the HelloWorldEJB.jar on your local file system.
3. Verify the **Add module to an EAR application** check box is selected.
4. In the EAR application list, select the previously created HelloWorldEAR enterprise application and click **Finish**.

Figure 8. Importing an EJB jar file

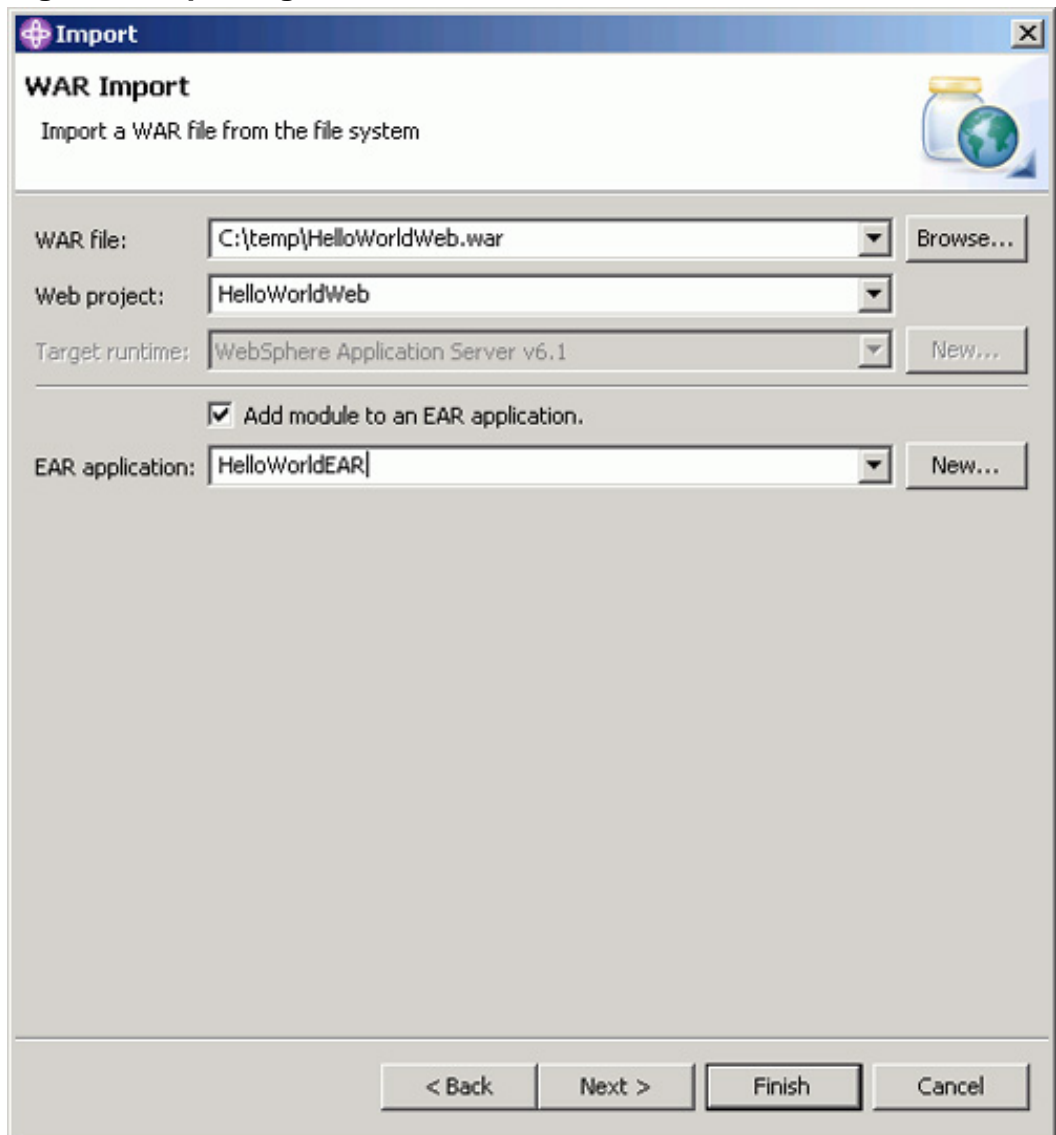


Importing the Web module

Import the Web module into the HelloWorldEAR enterprise application.

1. In the toolbar, select **File > Import > WAR file**. Click **Next**.
2. In the WAR file field, specify the location where you downloaded the HelloWorldWeb.war on your local file system.
3. Verify the **Add module to an EAR application** check box is selected.
4. In the EAR application list, select the previously created **HelloWorldEAR** enterprise application and click **Finish**.

Figure 9. Importing a WAR file



5. In the Problems view, there are several errors due to Java Build dependencies in the application. To resolve these problems, define that the Web module has a dependency on the EJB module:
 - a. In the Project Explorer view, expand **Dynamic Web Projects**.
 - b. Select the **HelloWorldWeb** Web project. Select **File > Properties** from the menu bar.
 - c. Select **Java Build Path**, and on the right pane select the **Projects** page.
 - d. Select **Add** and select the **HelloWorldEJB** check box.
 - e. Click **OK** in the Required Project Selection wizard and click **OK** in the Properties for HelloWorldWeb wizard.
 6. Congratulations, you have now packaged the HelloWorld application into a J2EE application.
-

Section 7. Exploring the deployment descriptor file and set up the application for deployment

In this exercise, explore the deployment descriptor files that were packaged by the developers in the EJB and Web modules. The *deployment descriptor* files describe how to deploy a module or application as defined in the configuration and container options.

Would you like to see these steps demonstrated for you?

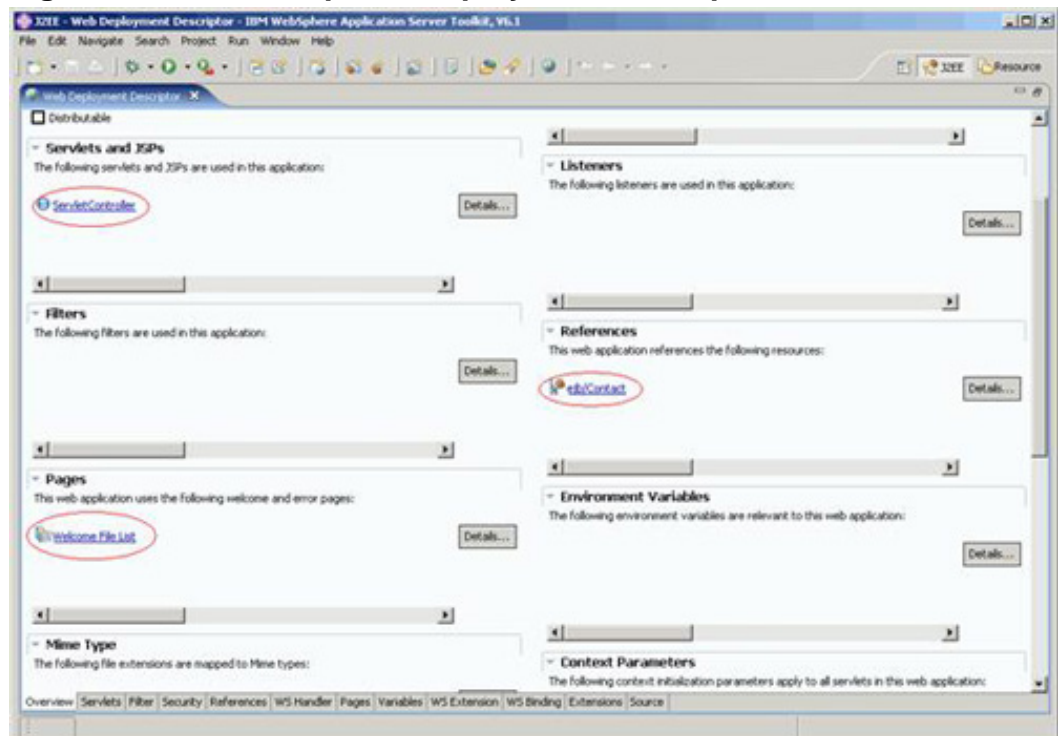


Look at the deployment descriptor files to discover details about the EJB and Web modules and determine what resources are required to publish the HelloWorld application successfully onto the server.

Exploring the Web deployment descriptor

1. Open the Web Deployment Descriptor:
 - a. In the Project Explorer view, expand **Dynamic Web Projects > HelloWorldWeb** project.
 - b. Under the HelloWorldWeb project, there is another HelloWorldWeb node. Right-click this second HelloWorldWeb node and select **Open with > Deployment Descriptor Editor**. The Web Deployment Descriptor editor opens.
2. In the Overview page under the Servlet and JSPs section, you discover that this Web module contains a servlet called `ServletController`. In addition, under the References section, this Web module has a reference to an EJB called `Contact`.

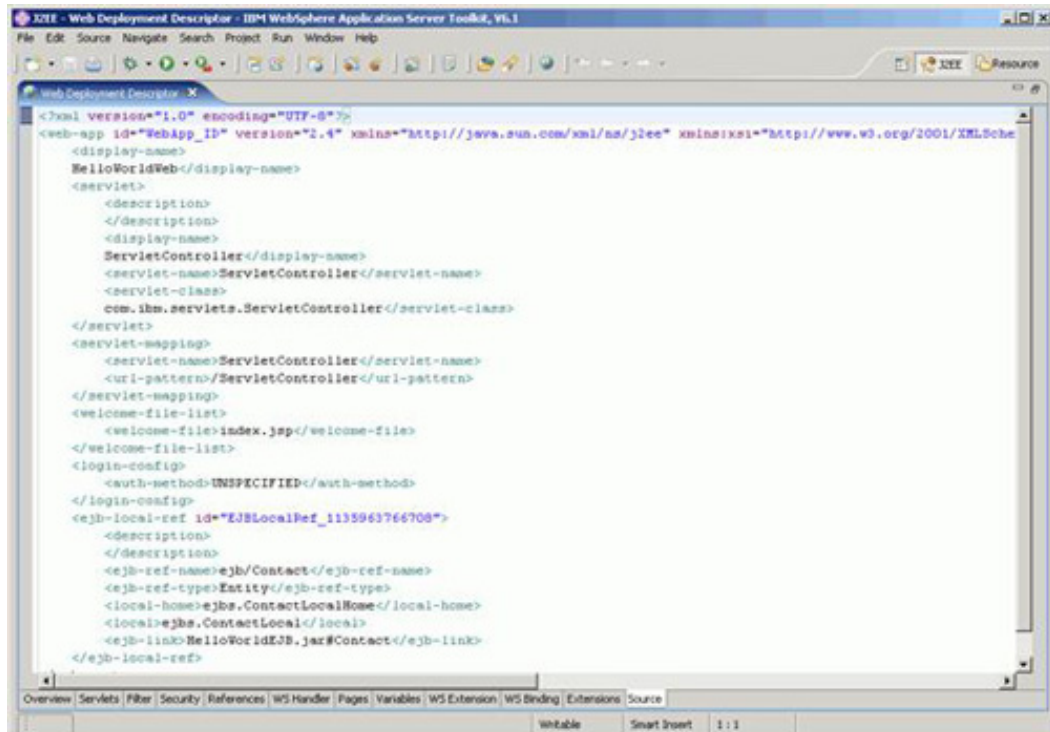
Figure 10. The WebSphere Deployment Descriptor editor



3. Discover the entry point into the application. Under Pages, click the **Welcome File List** link. The Pages section of the Web Deployment Descriptor editor opens.
4. Under the Welcome Pages section, you discover that this application uses an `index.jsp` page as the entry point. The following is the URL format to access the application:
`http://<machineName>:<port>/<WebContextRoot>/index.jsp`

5. All the above configuration options for the Web deployment descriptor are stored in an XML file called `web.xml`. To see the source of the Web deployment descriptor, click the **Source** page of the Web Deployment Descriptor editor.
6. Close the Web Deployment Descriptor editor when you are done exploring.

Figure 11. Source code of the Web deployment descriptor (web.xml file)



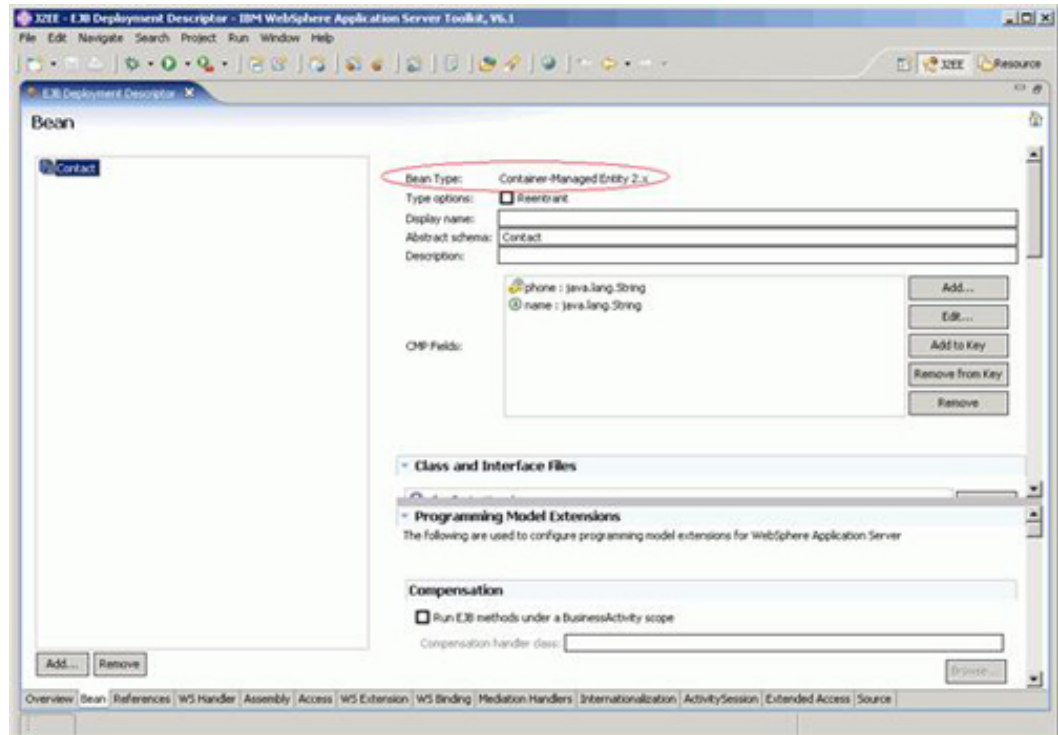
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>
    HelloWorldWeb</display-name>
  <servlet>
    <description>
    </description>
    <display-name>
      ServletController</display-name>
    <servlet-name>ServletController</servlet-name>
    <servlet-class>
      com.ibm.servlets.ServletController</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>ServletController</servlet-name>
      <url-pattern>/ServletController</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <login-config>
      <auth-method>UNSPECIFIED</auth-method>
    </login-config>
    <ejb-local-ref id="EJBLocalRef_1135963766708">
      <description>
      </description>
      <ejb-ref-name>ejb/Contact</ejb-ref-name>
      <ejb-ref-type>Entity</ejb-ref-type>
      <local-home>ejbs.ContactLocalHome</local-home>
      <local>ejbs.ContactLocal</local>
      <ejb-link>HelloWorldEJB.jar#Contact</ejb-link>
    </ejb-local-ref>
  </web-app>
```

Exploring the EJB deployment descriptor

1. Open the EJB deployment descriptor:
 - a. In the Project Explorer view, expand **EJB Projects > HelloWorldEJB** project.
 - b. Under the HelloWorldEJB project, there is another HelloWorldEJB node. Right-click this second HelloWorldEJB node and select **Open with > Deployment Descriptor Editor**. The EJB Deployment Descriptor editor opens.
2. In the Overview page, under the Enterprise JavaBeans section, you see

that this EJB module contains an EJB called `Contact`. Click **Contact** to learn more about this `Contact` EJB. The Bean page of the EJB Deployment Descriptor editor opens.

Figure 12. The EJB deployment descriptor editor



3. In the Bean Type field, notice that the `Contact` EJB is a `Container-Managed Entity 2.x`. The following text box explains what `Container-Managed Entity 2.x` means:

There are three types of enterprise beans: entity beans, session beans, and message-driven beans. In this case, the `Contact` EJB is an entity bean. *Entity beans* store permanent data, so they require connections to a form of persistent storage. This storage might be a database, an existing legacy application, or another type of persistent storage. Entity beans that manage their own persistence are called *bean-managed persistence (BMP)* entity beans. Entity beans that delegate their persistence to their EJB container are called *container-managed persistence (CMP)* entity beans. In this case, the `Contact` EJB is a container-managed persistence entity bean.

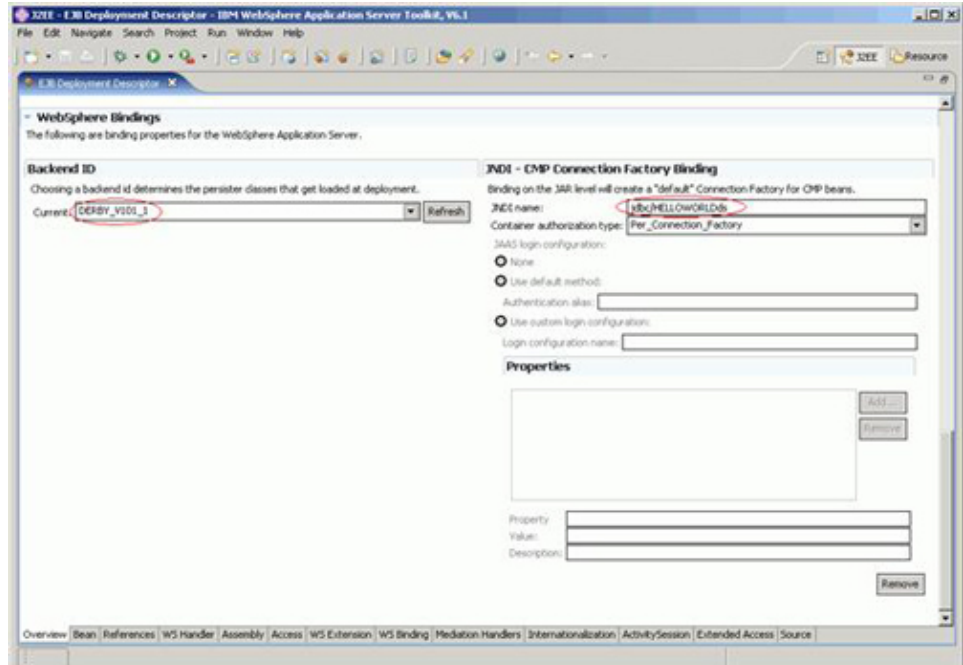
The `Contact` EJB complies with the 2.0 and 2.1 EJB specification-level which are standards set by Sun Microsystems J2EE architectures.

- For more details on enterprise beans, including information on session and message-driven beans, see [EJB architecture](#).
- For more details on EJB specification levels, see <http://java.sun.com/products/ejb/docs.html#specs>

4. Now that you've learned the `Contact` EJB is a CMP bean, you understand that CMP beans require database access. Beans that require database access use *data sources*, which are administrative resources that define pools of connections to databases. As an administrator, you need to create this data source to successfully deploy this HelloWorld application. Before you can create the data source, you need to find out two pieces of information:
 - **Current back-end ID**
The current back-end ID specifies the database type that you are deploying to and determines the persister classes that get loaded at deployment.
 - **Java Naming and Directory Interface (JNDI) name**
A *JNDI* is a naming and lookup service used to bind the EJB to a data source.
5. To find the JNDI name and the current back-end ID:
 - a. Select the **Overview** page of the EJB Deployment Descriptor editor.
 - b. Scroll down on the Overview page until you reach the WebSphere Bindings section.
 - c. The area below the **Backend ID** section shows the current database ID is `DERBY_V101_1`.
This back-end ID specifies the application is ready to be deployed against an IBM Cloudscape v10.1 database vendor.

Cloudscape is a commercial release of Derby. Derby is the Apache Software Foundation's (ASF) open source relational database project. Cloudscape support for complex SQL transactions and JDBC allows your applications to migrate to other SQL databases, such as IBM DB2 Universal Database (UDB), when they need to grow.
For more information about IBM Cloudscape and Derby, see <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0408anderson/>
 - d. The area below the **JNDI - CMP Connection Factory Binding** section shows the JNDI name is: `jdbc/HELLOWORLDds`

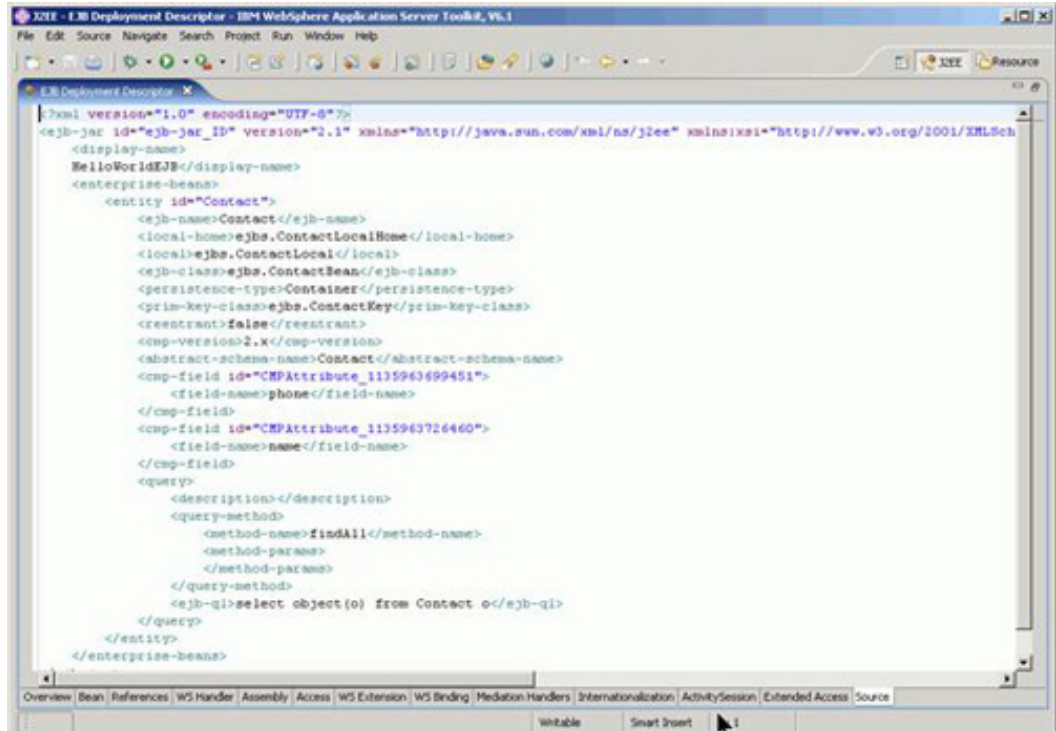
Figure 13. The EJB deployment descriptor editor



The CMP factory connection binding and the current back-end ID are settings specific to WebSphere Application Server. Binding configurations specific to WebSphere Application Server are stored in the `ibm-<module>-bnd.xmi` files of an application, where `<module>` is either `ejb-jar`, `Web`, `application` or `application-client`.

6. The configuration options for the EJB deployment descriptor are stored in an XML file called `ejb-jar.xml`. To see the source of the EJB deployment descriptor, click the **Source** page of the EJB Deployment Descriptor editor.

Figure 14. Source code of the EJB deployment descriptor (ejb-jar.xml file)



7. Go back to [Figure 2](#) to understand how the `ibm-ejb-bnd.xml`, `ejb-jar.xml`, and `web.xml` files fit in the J2EE architecture deployed on a WebSphere Application Server.
8. Close the EJB Deployment Descriptor editor.

Creating table and data source

During your exploration of the EJB deployment descriptor, you discovered the HelloWorld Application is ready to deploy against an IBM Cloudscape v10.1 database vendor. However, the persistent storage resources such as the database and table need to be created. Typically, this step would have already been performed by the database administrator. Because the database administrator is not available, you need to create the database and table required for this HelloWorld application to run successfully.

In addition, you need to create a data source which is used for database access to a vendor set to IBM Cloudscape v10.1 and use `jdbc/HELLOWORLDds` as its JNDI name.

To create the database, table and data source, use the table and data source creator wizard, which is a tool used for binding these data resources to a WebSphere Application Server. For more details on the table and data source

creator wizard, see [Creating tables and data sources automatically to test CMP beans for WebSphere Application Server V6.1](#) .

1. To use the table and data source creator wizard, first create a server connection between the workbench and the server:
 - a. In the Servers view, right-click and select **New > Server**. The New Server wizard opens.
 - b. In the Server's host name field, specify the local machine name where you installed WebSphere Application Server. Assuming you have installed the WebSphere Application Server on your local machine, keep the Server's host name field as localhost. Otherwise, specify the machine name or the IP address of the computer where you installed WebSphere Application Server.
 - c. Under the Select the server type, select **WebSphere V6.1 Server**. Click **Next**.
 - d. If the WebSphere Runtime page opens, in the Installation directory field specify `WAS_INSTALL_ROOT`, which is the directory where WebSphere Application Server is installed on your local machine. For example, `C:\Program Files\IBM\WebSphere\AppServer`. Click **Next**. The WebSphere Server Settings page opens.
 - e. Notice the WebSphere profile name field contains the value `AppSrv01`. Use this default profile to manage your server processes.

When WebSphere Application Server V6.1 is installed, the installation procedure creates an initial profile named `AppSrv01` in an application server named `server1`. A *profile* is the set of files that define a single server runtime environment. WebSphere profiles allow you to create and run multiple servers from the same WebSphere Application Server installation. Use the Profile Management tool to create additional profiles.

For details on creating additional WebSphere profiles see, [Creating an application server profile](#) .

- f. Under the Server connection type and admin port section, specify either the remote method invocation (RMI) or SOAP connector port to make Java Management Extensions (JMX) connections with the server. In other words, these ports are used for communication between the workbench and the server. For more details on the difference between the RMI and SOAP connector port see, [Setting](#)

the connection to the WebSphere Application Server V6.x .

Select the **SOAP** radio button as this connector port is designed to be more firewall compatible. The default setting of the SOAP port is 8880.

In this exercise, use the default starting port numbers. However, if you have another WebSphere Application Server installation or created another WebSphere profile on your machine, the port numbers may have changed. If this is the case, determine and use the assigned port numbers to successfully communicate with the server. Instructions to determine the assigned port numbers are provided in the following text box:

Default port number settings are given when you first install WebSphere Application Server, or you can optionally change the starting port numbers. As additional installations of WebSphere Application Server or additional profiles are created on your system, the starting port number settings are incremented by one to avoid conflicts with other assigned ports.

A port number must be unique within a hostname. The port number is valid in the range of 0 and 65535, where 0 specifies that the server should bind to any ephemeral (short-lived) port available.

Determine the port values of a configured profile by referencing its server configuration files. The port values are stored in the serverindex.xml file located in the following directory:
 WAS_INSTALL_ROOT\profiles*<profileName>*\config\cells*<cellName>*\nodes*<nodeName>*,
 where WAS_INSTALL_ROOT is the directory WebSphere Application Server is installed. For example,
 C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\config\cells\larinatpadNode02Cell\nodes\larinatpadNode02\serverindex.xml

The following table lists the ports that are in this exercise. If the default starting port numbers have changed, fill out the following table for future reference:

Port name	Port description	Default port number	Your assigned port number
SOAP_CONNECTOR_ADDRESS	SOAP	8880	
BOOTSTRAP_ADDRESS	RMI	2809	
WC_adminhost	Administrative Console	9060	
WC_defaulthost	HTTP transport	9080	

If the assigned port number of the SOAP_CONNECTOR_ADDRESS is different from the default 8880, specify your assigned port number in the SOAP connector port field on the WebSphere Server Settings page.

- g. Verify the **Run server with resources within the workspace** check box is selected. This option lets you quickly run and debug applications directly from the workspace by reducing the files copied from the workbench to the server.
Caution: If you delete the workspace, the server can no longer find the resources and you may lose your application if it's not under source control management.

Later you will learn how to switch the publishing setting of the server to install the application into the directories of the server rather than leaving the application in the directories of the workspace.

Figure 15. Define a new server

New Server

WebSphere Server Settings

Input settings for the new WebSphere server.

WebSphere profile name: AppSrv01

Server connection type and admin port

RMI (Designed to improve communication with the server)

ORB bootstrap port: 2809

SOAP (Designed to be more firewall compatible)

SOAP connector port: 8880

Run server with resources within the workspace

Security is enabled on this server

Current active authentication settings:

User ID:

Password:

Server name: server1

Server type

BASE, Express or unmanaged Network Deployment server

Network Deployment server

Network Deployment server name:

The server name is in the form of:
<cell name>/<node name>/<server name>
For example, localhost/localhost/server1. In a cluster environment,
the server name is in the form of:
<cell name>/<cluster name>

Click this button to detect the server type.

< Back Next > Finish Cancel

- h. Click **Next**. The Add and Remove projects page opens.
 - i. Under the Available projects list, select **HelloWorldEAR** and click **Add** to add the HelloWorld application to the server.
 - j. Click **Finish**. A new server is created in the Servers view.
2. Create the database, tables, and data sources:

- a. In the Servers view, right-click the server.
- b. Select **Create tables and data sources**. The Table and Data Source Creator wizard opens.
- c. In the previous step when you added the HelloWorld application to the server, the top pane of the Table and Data Source Creator wizard automatically identifies that the application added to the server would like to create a table and data source against an IBM Cloudscape v10.1 database because its current backend ID is set to DERBY_V101_1.

Figure 16. Table and Data Source Creator wizard

Table and Data Source Creator

Create Tables and Data Source

Click Next to specify the database connection information.

Current back-end ID: DERBY_V101_1
Current EJB project: HelloWorldEJB (1 out of 1)
Current EJB JAR name: HelloWorldEJB
Current EJB JAR version: 2.1

Create database tables for Derby 10.1
 Drop the previously generated database tables

Create data source for Derby 10.1

Database server host name or IP address: not required
Database server port number: not required

Specify all data source connection information here

Database user ID: not required
Database user password: not required
Local path to JDBC drivers: not required

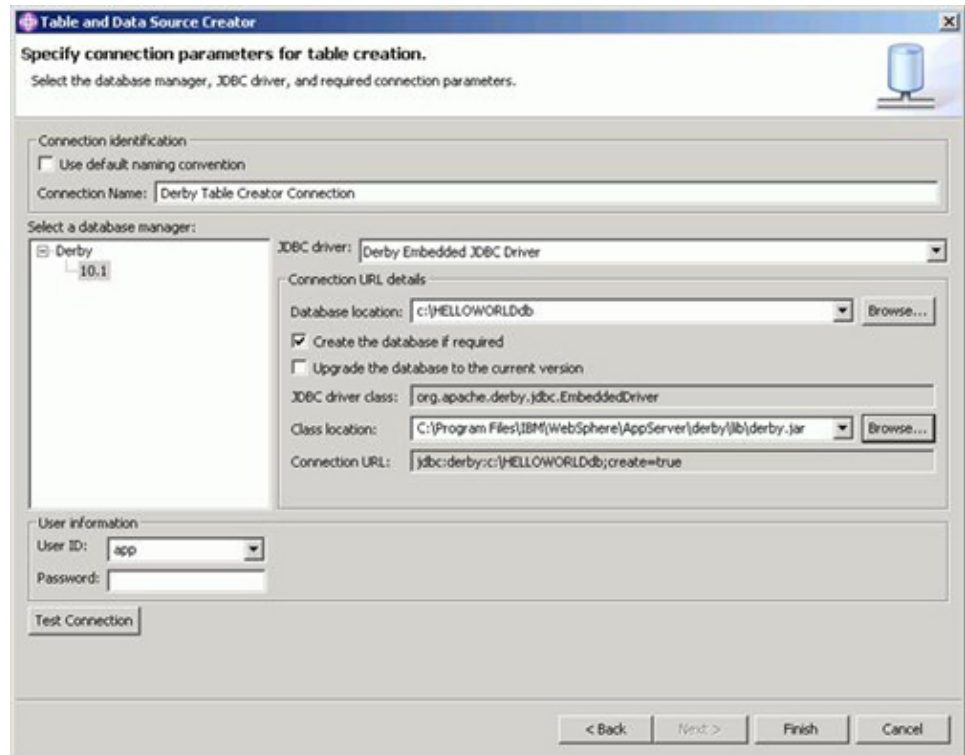
< Back Next > Finish Cancel

- d. Verify **Create database tables for Derby 10.1** is selected. This check box specifies to create a set of tables to support the Contact CMP entity bean inside your HelloWorldEJB project. In these tables, each column corresponds to a CMP field of the

enterprise bean, and the generated mapping maps the field to the column. This is known as *top-down mapping*, where it takes the design of the existing CMP entity bean to determine the database design. For more information on top-down mapping and other approaches for mapping enterprise beans to database tables, see [Approaches for mapping enterprise beans to database tables](#) .

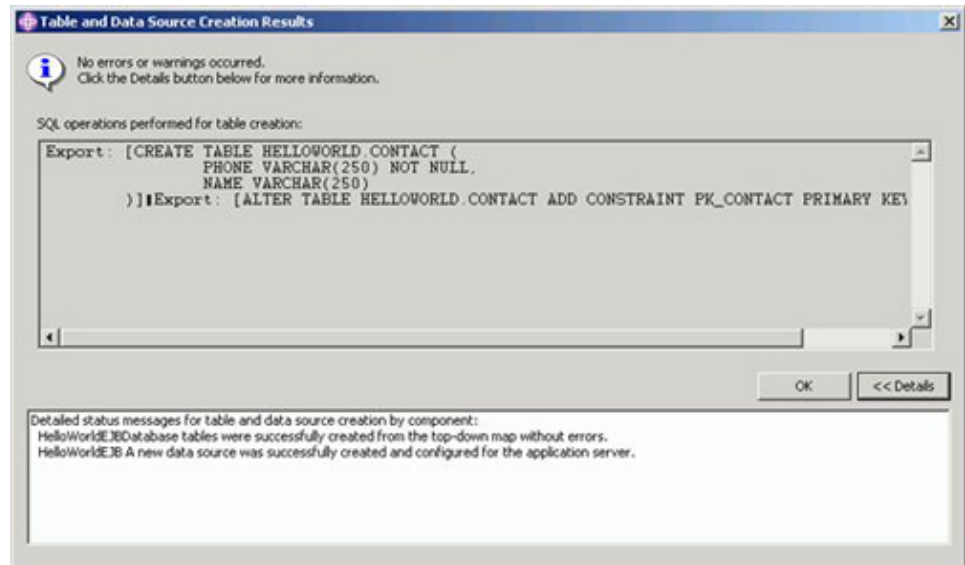
- e. Clear the **Drop the previously generated database tables** check box. Because you are creating the tables for the first time you do not need to remove any previously generated database tables.
- f. Verify **Create data source for Derby 10.1** is selected. This check box specifies to create the data source for database access to a vendor set to IBM Cloudscape v10.1.
- g. Click **Next**. The Select Connection page opens.
- h. Verify the **Create a new connection** radio button is selected and click **Next**. The Specify connection parameters for table creation page opens.
- i. Under the JDBC driver list, select **Derby Embedded JDBC Driver** as the Java database connectivity (JDBC) driver.
- j. In the Database location text field, type `c:\HELLOWORLDb` as the file location to create the database.
- k. Verify the **Create the database if required** check box is selected. This option is available only for Cloudscape. It creates the database if it does not exist.
- l. In the Class location field, specify the location where the IBM Cloudscape v10.1 JDBC drivers are located, such that the WebSphere Application Server can find and use them. Browse to `WAS_ROOT_INSTALL\derby\lib\derby.jar` where `WAS_ROOT_INSTALL` is the installation directly of WebSphere Application Server, for example, `C:\Program Files\IBM\WebSphere\AppServer\derby\lib\derby.jar`.

Figure 17. Specifying database connection settings for table creation



- m. Click **Finish**.
- n. The Table and Data Source Creations Results window opens and displays a list of SQL operations and any errors when performing the table and data source creation. The Details button is available for additional status on the table and data source creation. Click **OK** after you are finished looking at the results.

Figure 18. Results of the Table and Data Source Creation wizard



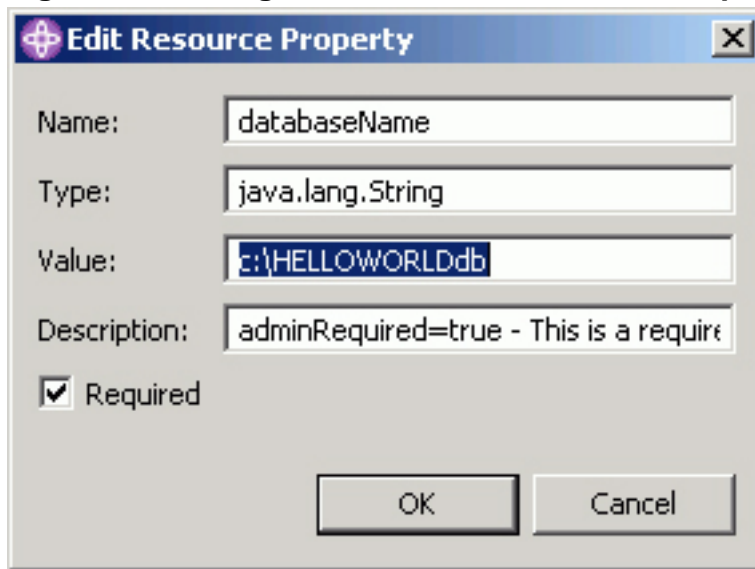
Verifying the creation of the data source

1. Open the Application Deployment Descriptor:
 - a. In the Project Explorer view of the J2EE perspective, expand **Enterprise Applications > HelloWorldEAR** project.
 - b. Under the HelloWorldEAR project, there is another HelloWorldEAR node. Right-click this second HelloWorldEAR node and select **Open with > Deployment Descriptor Editor**.
 - c. The Application Deployment Descriptor editor opens.
2. Select the **Deployment** page at the bottom of the editor. This Deployment page allows you to specify some server configurations specific to WebSphere Application Server V6.0 and above. The server configuration data that you specify in this editor gets embedded within the application itself and is made available to the server at deployment-time.

Alternatively, set the same configurations directly on the server using the WebSphere administrative console. However, if you decide to deploy the same application on multiple V6.1 WebSphere Application Servers, setting the server configuration in the Deployment page is a better approach. Using the Deployment page, you only need to create the configuration once at the application-level. Whereas, if you used the WebSphere Administrative console, you would have to create the configuration multiple times, depending on number of servers hosting the application. Therefore, using the Deployment page can help decrease the administration work.

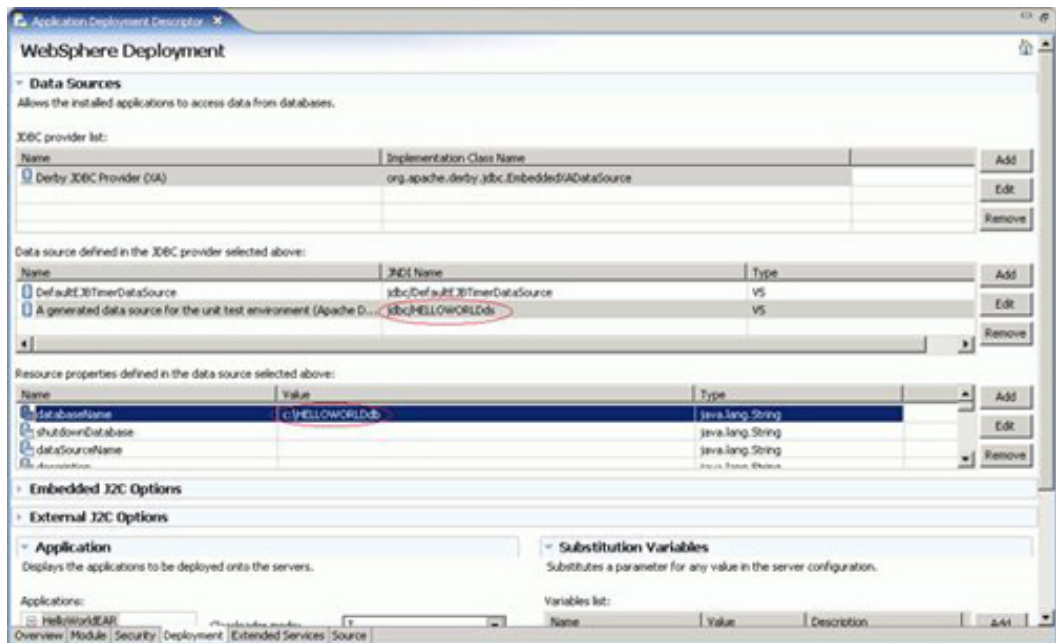
- Under the Data source defined in the JDBC provider selected above section, select the **A generated data source for the unit test environment (Apache Derby v10)** entry. This is the data source created for the HelloWorld application by the Table and Data Source Creator wizard.
- Under the Resource properties defined in the data source selected above section, select **databaseName** and click **Edit**. The Edit Resource Property window opens. A full file path to the database needs to be specified. In the Value field, type `c:\HELLOWORLddb` and click **OK**.

Figure 19. Editing the databaseName resource property



- Select **File > Save** to save the changes in the Application Deployment Descriptor editor.
- The deployment page of the Application deployment descriptor editor should look like this:

Figure 20. Deployment page of the Application deployment descriptor editor



Notice the data source sets the JNDI name to `jdbc/HELLOWORLDDs`. When the data source was created, the JNDI name was provided by the EJB deployment descriptor files and the database name was provided by you when defining the Database location field in the Table and Data Source creator wizard.

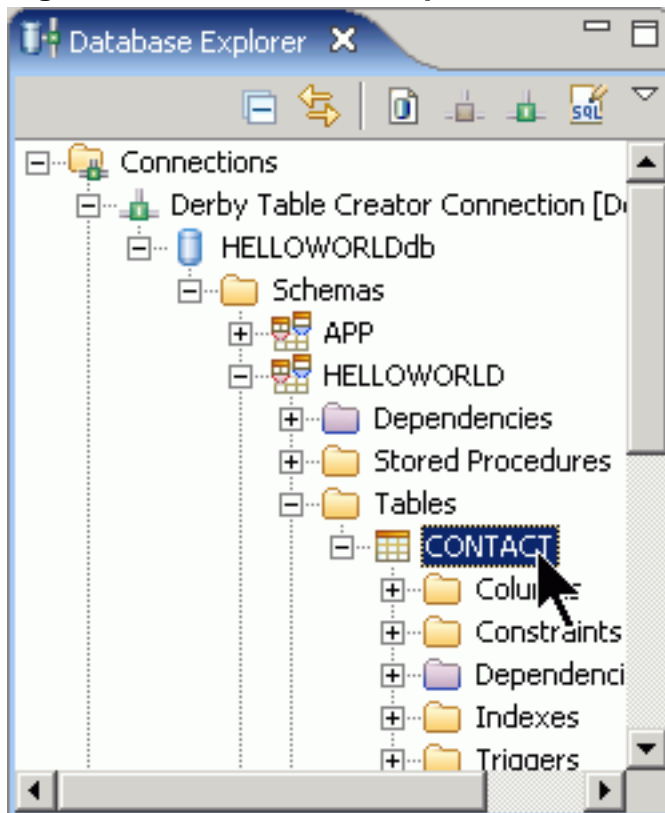
7. Close the Application Deployment Descriptor editor after you finish verifying that the data source has successfully been created for the HelloWorld application.
8. Because you modify the Application Deployment Descriptor by specifying the full path to the database, restart the application so the server can pick up the changes:
 - a. In the Servers view, expand **WebSphere V6.1 Server @ localhost**.
 - b. Right-click **HelloWorldEAR** and select **Restart HelloWorldEAR**.

Verifying the creation of the database and tables

1. Switch to the Database perspective:
 - a. In the toolbar, select **Window > Open Perspective > Other**.
 - b. In the Select Perspective wizard, select **Data > OK**.

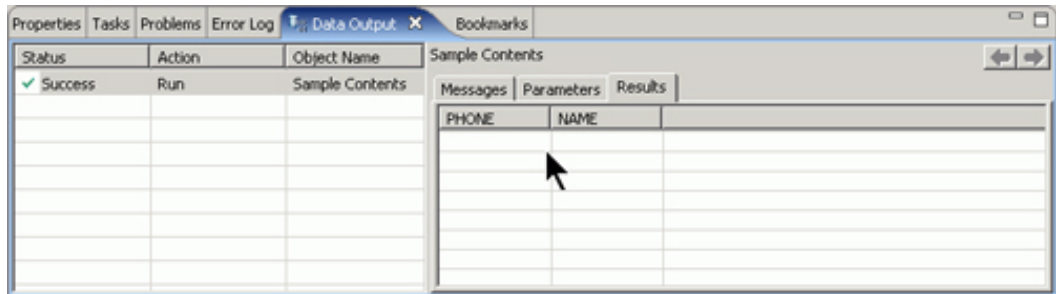
2. In the Data Explorer view, expand the **Connection** node and right-click the **Derby Table Creator Connection [Derby 10.1]**.
3. Select **Disconnect** from the pop-up menu. Once the database connection is disconnected, right-click the **Derby Table Creator Connection [Derby 10.1]** and select **Reconnect**.
4. The Database Authorization window opens requesting a user ID and password. A user ID and password is not required, so click **OK**. The connection icon turns green.
5. In the Data Explorer view, expand the **Connection > Derby Table Creator Connection [Derby 10.1] > HELLOWORLddb > Schemas > HELLOWORLD > Tables > CONTACT**.

Figure 21. The Database Explorer view



6. Right-click **CONTACT** and select **Data > Sample Contents** to see the contents of the table. In the DB Output view, notice the table contains a **PHONE** and **NAME** column.

Figure 22. The DB Output view




7. Multiple connections to a single Cloudscape database are not supported due to a Cloudscape configuration restriction. If you maintain the database connection to the database from the Database Explorer and a server tries to make another Cloudscape connection through a data source, the second connection fails. As a result, you need to close the connection from the Database Explorer before a server can establish a connection to the Cloudscape database:
 - a. In the Database Explorer view, right-click **Derby Table Creator Connection [Derby 10.1]** and select **Disconnect**.
8. Congratulations, you have successfully explored the deployment descriptors and created the required resources (data source, database and table) for running the HelloWorld application.

Section 8. Testing and publishing the J2EE application on the server

In this exercise, you are going to test the HelloWorld application that is running on the workbench prior to installing the application into the directories of the server.

Would you like to see these steps demonstrated for you?

 [Show me](#)

Testing the J2EE application on the server

1. Switch to the J2EE perspective:

- a. In the toolbar, select **Window > Open Perspective > Other**.
 - b. In the Select Perspective wizard, select **J2EE > OK**.
2. When you explored the Web deployment descriptor, you discovered that the entry point to the HelloWorld application is index.jsp. Try to test the HelloWorld application by running this file on the server:
- a. In the Project Explorer view, expand **Dynamic Web Projects > HelloWorldWeb > WebContent**.
 - b. Right-click the index.jsp file and select **Run As > Run on Server**. The Server Selection wizard opens.
 - c. Select the **Choose an existing server** radio button, as you are going to use the previously created server connection.
 - d. Click **Next**. The Add and Remove projects page opens.
 - e. If the HelloWorldEAR application is specified under the Configured projects list, the application is added to the server. Otherwise, complete the following:
 - Under the **Available projects** list, select the **HelloWorldEAR** and click the **Add** button to add the HelloWorld application to the server.
 - f. Click **Finish**. This step takes a bit of time for the system to process.

The **Run on Server** tool completes the following server tasks:

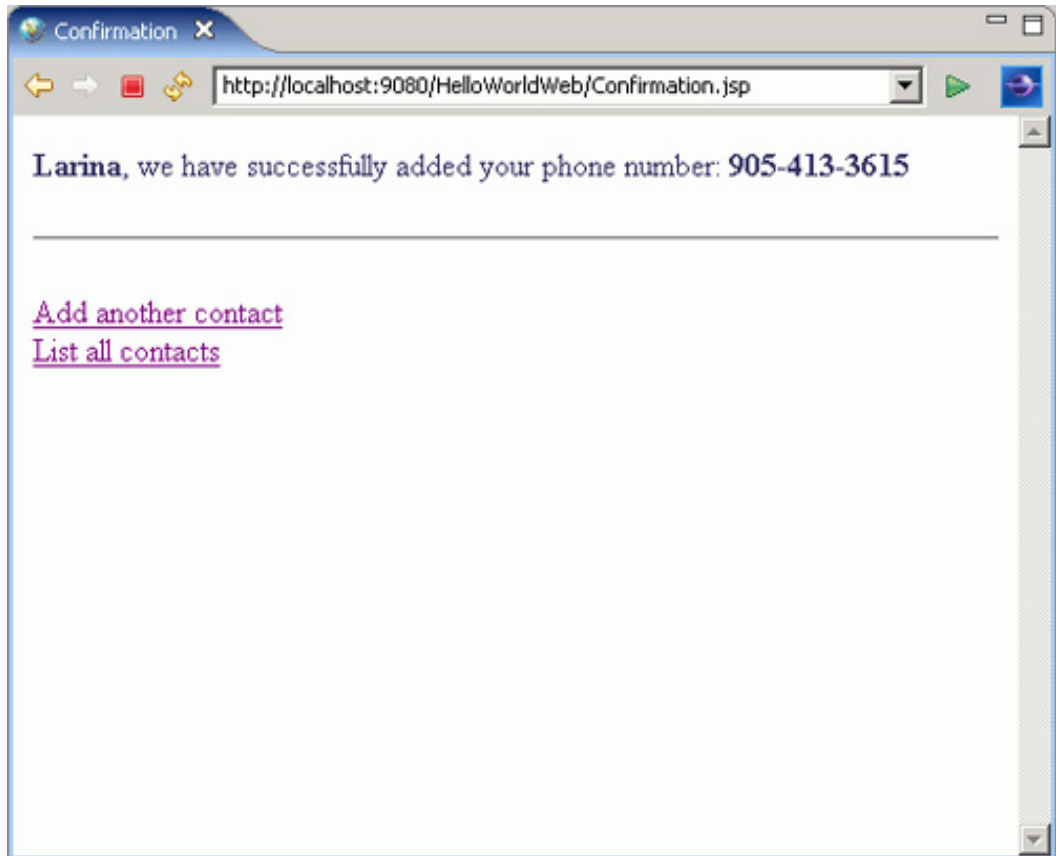
- Copies the application files to the correct location for the server to find and use them.
- If the Automatically publish before starting servers check box on the Server preferences page (**Window > Preferences > Server**) is selected, the workbench checks to see if your project and files on the server are synchronized. If they are not, the project and the files are automatically updated.

3. The HelloWorld application successfully runs at the following URL:
`http://localhost:9080/HelloWorldWeb/index.jsp`
Try adding a new user name and phone number. Click **Submit**.
- Figure 23. Testing the HelloWorld application**



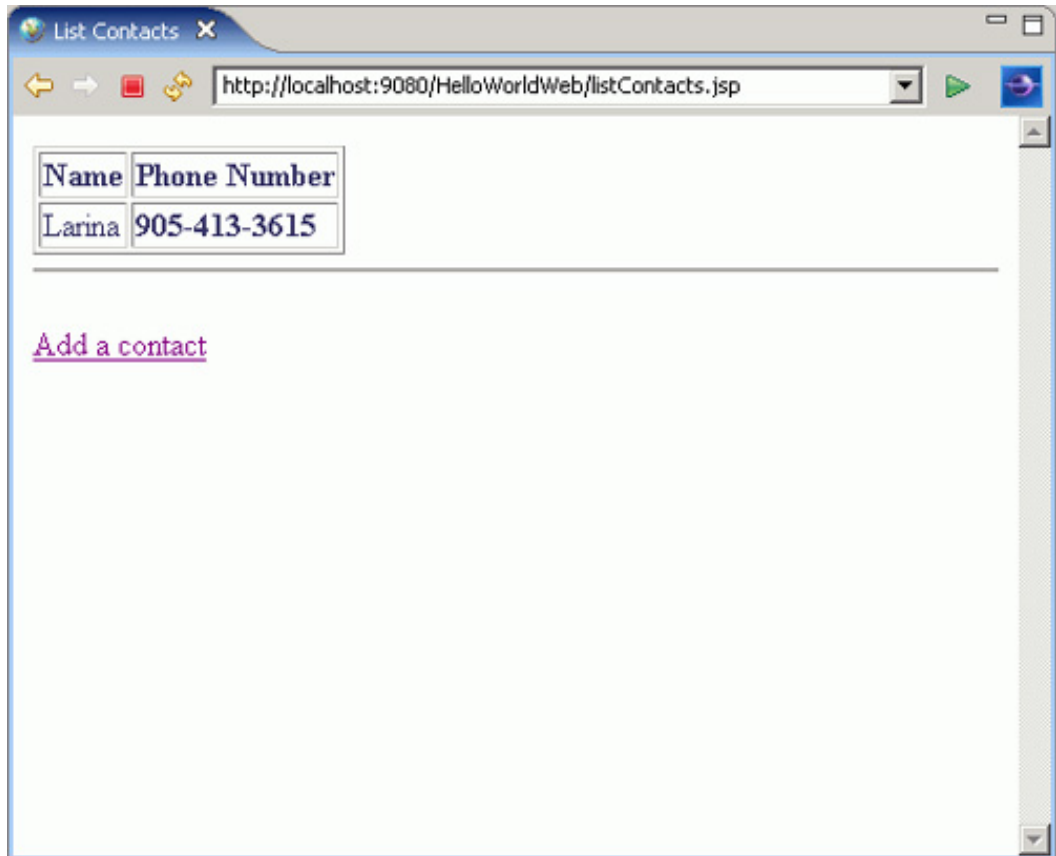
The screenshot shows a web browser window with the title "addContactInfo.jsp". The address bar displays "http://localhost:9080/HelloWorldWeb/index.jsp". The main content area features the text "Hello, World!" in a large, dark blue serif font. Below this is a horizontal line. The text "Add a name and phone number in the following fields:" is displayed in a dark blue serif font. There are two input fields: "Name:" with the value "Larina" and "Phone:" with the value "905-413-3615". Below the input fields are two buttons: "Submit" and "Clear". A horizontal line is positioned below the buttons. At the bottom of the form area, there is a blue underlined link that says "List all contacts".

4. You have successfully added a new name and phone number. Try listing all the user names and phone numbers by clicking **List all contacts**.
Figure 24. Adding a contact to the HelloWorld application



5. You have successfully listed all the user names and phone numbers entered by the application. Close the Web browser after you have completed your test of the HelloWorld application.

Figure 25. Listing all the contacts in the HelloWorld application



Changing the context root

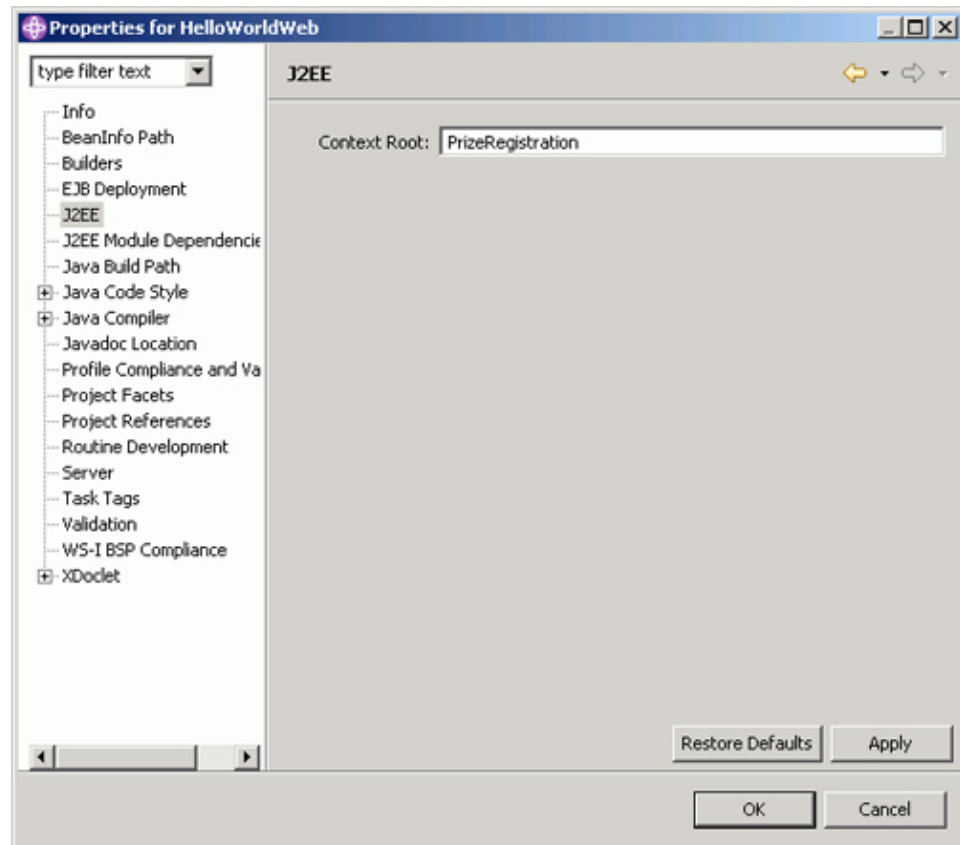
Now that you completed the testing of the HelloWorld application, you review your work with your manager on the testing environment. During the review, your manager requests to modify the current URL

`http://localhost:9080/HelloWorldWeb/index.jsp`. She would like you to replace any mention of `HelloWorldWeb` with `PrizeRegistration` when you deploy the application on the server. She explains that this HelloWorld application is used for an upcoming promotional prize giveaway. The purpose of the application is to capture the name and phone number of each participant that enters the online lucky draw.

1. To change the URL from `HelloWorldWeb` to `PrizeRegistration`, you need to change the context root. The *context root* is the Web application root, which is the top-level directory of your application when it is deployed to the server.
 - a. In the Project Explorer view, expand **Dynamic Web Projects** and select **HelloWorldWeb** project folder.

- b. In the menu bar, select **File > Properties**. The Properties for HelloWorldWeb window opens.
- c. Select **J2EE**. The J2EE page opens.
- d. In the Context Root field, type `PrizeRegistration`. Click **Apply > OK**.

Figure 26. Changing the context root of the HelloWorld application



2. To test this new context root on the server:
 - a. In the Project Explorer view, expand **Dynamic Web Projects > HelloWorldWeb > WebContent**.
 - b. Right-click the `index.jsp` file and select **Run As > Run on Server**. The Server Selection wizard opens.
 - c. Select **Choose an existing server** and click **Next**. The Add and Remove projects page opens.
 - d. Verify that the HelloWorldEAR application is specified under the

Configured projects list so it can be added to the server. Click **Finish**.

- e. A Web browser opens with the application running with the new context root: `http://localhost:9080/PrizeRegistration/index.jsp`.

Figure 27. Testing the HelloWorld application with the new context root



3. Close the Web browser after you have tested the HelloWorld application using the PrizeRegistration context root.

If you exit WebSphere Application Server Toolkit, your HelloWorld application remains running as long as the WebSphere Application Server is up and running.

Deploying the application on the external server

You successfully tested the application and fulfilled your manager's requirements. Now it's time to put the application on the external server by installing the application into the directories of the server.

New in Version 6.0 of WebSphere Application Server, you can use the full WebSphere Application Server installation for both testing and as an external server. The publishing setting integrated in the workbench allows you to safely switch between an external server and test environments as you

use the same WebSphere Application Server installation. This guarantees that your application will run the same when testing and publishing the application on the server.

Note: The following IBM development tools or products can be used to publish or remove applications to any WebSphere Application Server that it supports:

- WebSphere Application Server Toolkit
- Rational Application Developer
- Rational Software Architect
- WebSphere Integration Developer

Although it is possible to deploy applications into production directly from your development environment, this is not a recommended practice. Access control should be enforced on production servers, and applications should deploy through controlled and repeatable processes.

There are three publishing settings:

Run server with resources on Server: This publishing option installs and copies the full application and its server-specific configuration from the workbench into the directories of the server. The default location where an application gets installed into server is `x:/profile/installedApps/cellName` directory, where `x:/profile` is the directory of your profile for the WebSphere Application Server. This option may take the longest time to publish because it involves the most files copied to the server. To run this publishing option, the server can be running either on the same or different machine as your workbench.

Run server with resources within the workspace: Requests the server to run your application from the workspace. This allows for testing and debugging as it is designed to operate faster because fewer files are involved when copied over to the server. This option is available only when running the server on the same machine as your workbench (this environment is known as running a local server) and not available when running a server on a different machine as your workbench (this environment is known as running a remote server).

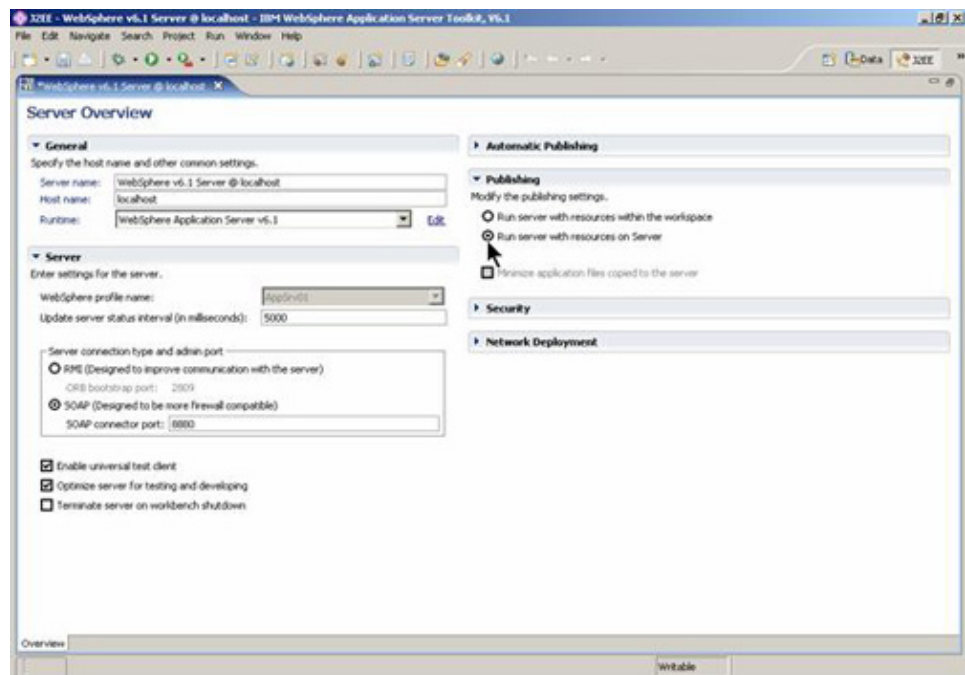
Minimize application files copied to the server: This publishing option becomes available only when the Run server with resources within the workspace option is selected. This is designed to publish much faster than when only the Run server with resources within the workspace option is selected because even fewer files are involved when copied over to the server. This option is available only when running a local server with resources within the workspace and not available when running a remote server.

Up to this point, you have been running on a local server environment using the **Run server with resources within the workspace** with the **Minimize application files copied to the server** publishing setting enabled. When you deploy your application in an external server, you should copy your application into the directories of the server, as opposed to running the application on the workbench. Otherwise, if you delete the workspace, the server no longer can find your application. And if you did not put your application under source control management, you might accidentally erase your application from your file system. Switch the publishing setting to **Run server with resources on Server** to install and run the application in the directories of the server.

For more details on the publishing settings, see [Setting publishing preferences for a WebSphere Application Server V6.x](#).

1. Prior to publishing your application on the external server, you need to remove the application from the test environment.
 - a. In the Servers view, right-click the server and select **Add and remove projects**. The Add and Remove projects wizard opens.
 - b. Under the Configured Projects list, select **HelloWorldEAR** and click **Remove**. The HelloWorldEAR application is moved under the Available Projects list.
 - c. Click **Finish**.
2. To switch to the Run server with resources on Server publishing setting:
 - a. In the Servers view, double-click the server. The server editor opens.
 - b. Expand the Publishing section.
 - c. Select **Run server with resources on Server**.

Figure 28. Changing publishing settings to Run server with resources on Server



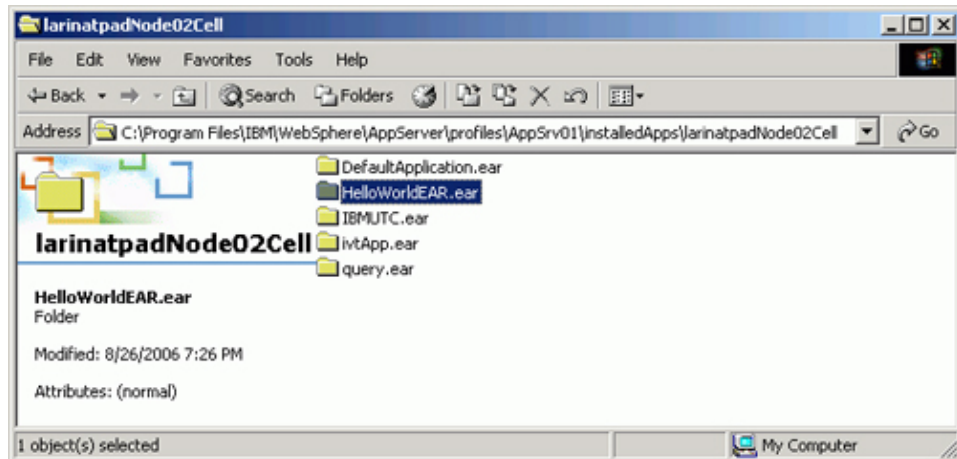
- d. In the toolbar, select **File > Save** to save the changes in the server editor.
- e. Close the server editor.

3. Now that you have switched the publishing setting to the external server, add the HelloWorld application to the server.
 - a. In the Servers view, right-click the server and select **Add and remove projects**. The Add and Remove projects wizard opens.
 - b. Under the Available Projects list, select **HelloWorldEAR** and click **Add**. The HelloWorldEAR application is moved under the Configured Projects list.
 - c. Click **Finish**.
The application automatically gets installed on the server as the application is copied from the workbench and placed in the `installedApps` directory of the server.

4. Verify the HelloWorld application is installed and published to the external server.
 - a. Open Windows Explorer to the following directory:
`WAS_INSTALL_ROOT\profiles\default\installedApps\cellName`,
where `WAS_INSTALL_ROOT` is the installation directory of WebSphere Application Server. For example,
`C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\installedApps\larinatpadNode`
directory.

 - b. If you see the HelloWorldEAR.ear folder, you have successfully installed the application on the external server. This folder contains a copy of the HelloWorld application, resource files, and deployment descriptors in the correct location for the server to find and use them.

Figure 29. Verify the HelloWorld application is installed into the directories of the server

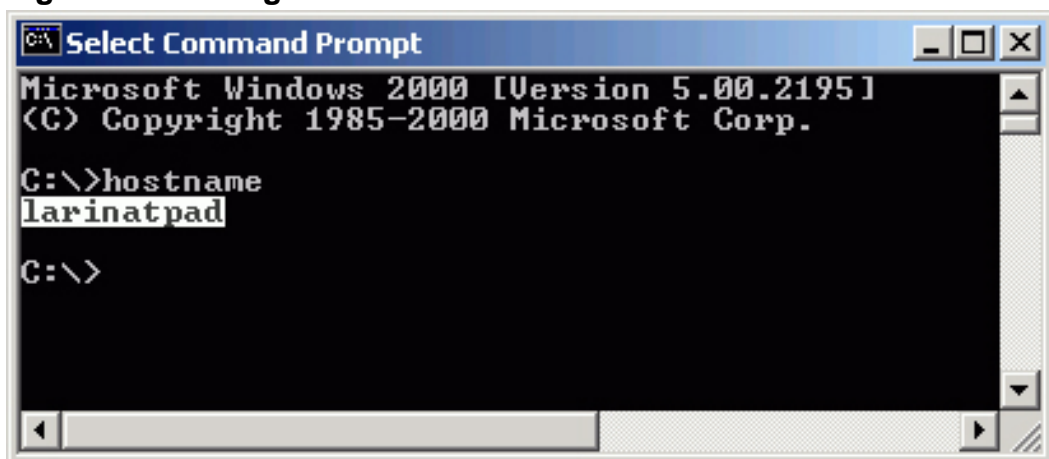


Run the application on another machine on the network

Test if the HelloWorld application is deployed successfully onto the server by verifying the application is accessible on another machine.

1. On the machine where your server is running, open a command prompt, and type `hostname`. Write down the hostname of the machine on which you are running the server. In this example, the hostname is `larinatpad`.

Figure 30. Finding the hostname



If your domain name system (DNS) server does not recognize the hostname, try the IP address. Find the IP address by typing `ipconfig` in a command prompt.

2. Open a Web browser on another machine and in the address bar specify: `http://<hostname>:9080/PrizeRegistration/index.jsp` where `<hostname>` is the name of the machine or the IP address where

you are running the server.

Figure 31. Testing the HelloWorld application on another machine




3. Congratulations, you have successfully tested and published the HelloWorld application on the external server.

Section 9. Maintaining the J2EE application on the server

This exercise shows you how to start, stop, and uninstall the application from the server. In addition, you learn some problem determination techniques.

Would you like to see these steps demonstrated for you?

 Show me

Managing the application on the server

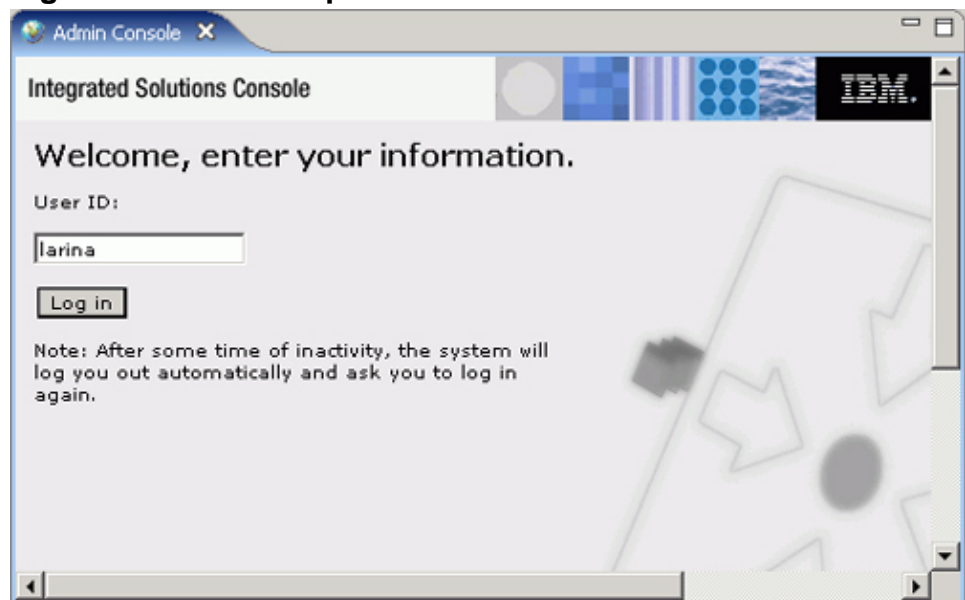
Suppose a month has passed and the due date of the prize giveaway has approached. It's time to stop the HelloWorld application from running on the server as your company is no longer accepting contestant entries. To stop the application on the server, you need to use the WebSphere administrative console. The workbench is integrated with the WebSphere administrative console.

1. In the Servers view, right-click the server and select **Run Administrative Console**. The Administrative Client logon window opens in the Web Browser view.

Alternatively, open the administrative console in a Web browser by specifying `http://localhost:9060/ibm/console/` in the address bar.

- a. Specify any text for the user ID as it is used for logging purposes when the server is not secured. Click **Log In**.

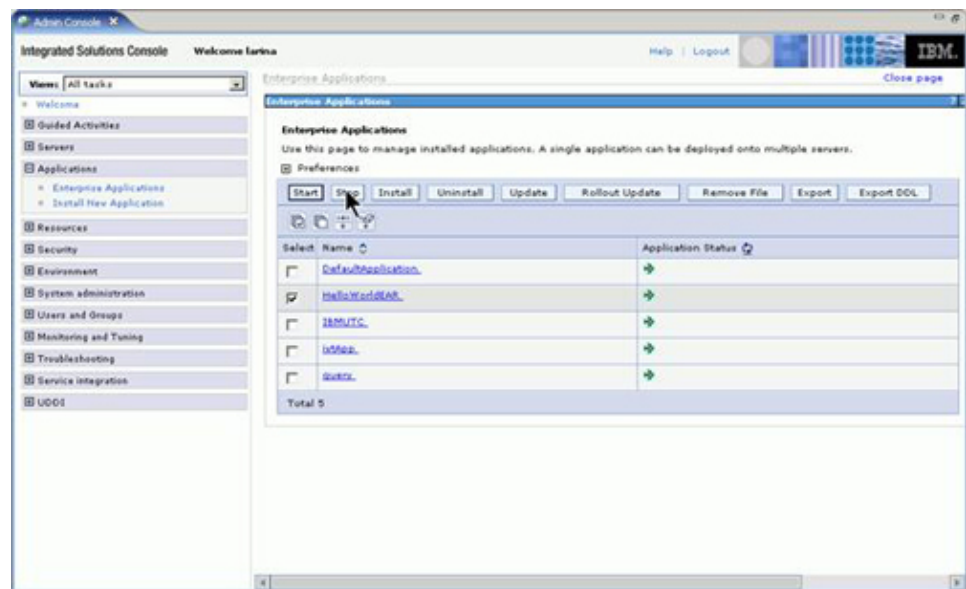
Figure 32. The WebSphere Administrative Console



- b. On the left pane, select **Applications > Enterprise Applications**. The Enterprise Applications page opens.
- c. Place a check mark beside HelloWorldEAR and click **Stop**. The

 stop icon.

Figure 33. Stopping an application in WebSphere Administrative Console



- d. Open a new Web browser and in the address bar specify:


```
http://<hostname>:9080/PrizeRegistration/index.jsp
```

 where, <hostname> is the name of the machine where you are running the server. An error 404: FileNotFoundException is displayed.
 - e. Congratulations, you have successfully stopped the application from running on the server.
2. If your manager decides to extend the promotional giveaway, allowing for more contestant entries past the due date, repeat the preceding steps and select **Start** to restart the application on the server.
 3. If your manager decides that the promotional giveaway will no longer continue, select **Uninstall** to remove the application from the installedApps directory of the external server.
 4. Logout and close the WebSphere administrative console.

Troubleshooting and problem determination

1. The WebSphere Application Server generates many log files that assist in

problem determination. All WebSphere Application Server log files are under the `WAS_INSTALL_ROOT/profile/log` directory, where `WAS_INSTALL_ROOT/profile` is the directory of your profile for the WebSphere Application Server. For example, `C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs` directory. The following table describes some of the main log files available for troubleshooting the server:

Log file	Description
SystemOut.log	Standard JVM output log that indicates if code running in the server started and stopped successfully.
SystemErr.log	Standard JVM error log that contain exceptions thrown by code running in the server.
startServer.log	Logs the startup of the server. If the server has successfully started, the last two lines of this log file read: Server launched. Waiting for initialization status. Server server1 open for e-business; process id is 1932.
stopServer.log	Logs the shutdown of the server. If the server has successfully stopped, the last two lines of this log file read: Server stop request issued. Waiting for stop status. Server server1 stop completed.
activity.log	A binary file that logs events that show a history of activities. A Log Analyzer tool is available to read the output from this file.

- In WebSphere Application Server Toolkit , you might have noticed that most of the messages in these log files were written in the Console view during the runtime of the server.
- Interpreting log messages. A WebSphere Application Server log entry has the following format:

Tin Sta	Thi Id	Col	Me Ty	Me Rel	Message Description
[8/16/06 9:37:20:380 EDT]	0000000a	WsServerImpl	A	WSVR0001I	Server server1 open for

Time Stamp: The timestamp is formatted using the locale of the process where it is formatted. It includes a fully qualified date (YYMMDD), 24-hour time with millisecond precision, and the time zone.

Thread Id: An 8-character hexadecimal value generated from the hash code of the thread that issued the trace event.

Component: The abbreviated name of the logging component that issued the log event. This is typically the class name for WebSphere Application Server internal components, but might be some other identifier for user applications.

Message Type: A one-character field that indicates the type of the log event:

>	Entry to a method
<	Exit a method
A	Audit
W	Warning
X	Error
E	Event
D	Debug
T	Terminate (exits process)
F	Fatal (exits process)
I	Information
O	Program output

Message Reference: The message identifier can be either 8 or 9 characters in length and has the form, for example, CCCC1234X, where CCCC is a four-character alphabetic component or application identifier, 1234 is a four-character numeric identifier used to identify the specific message for that component, and X is an optional alphabetic severity indicator (I=Informational, W=Warning, E=Error). For details on the message reference on a particular message identifier, search on the message identifier in the [WebSphere Information Center](#).

Message Description: The message output issued.

Section 10. Summary

Congratulations! You have now completed Part 4 of the Hello, World! series. You've learned how to assemble and publish J2EE applications on a WebSphere Application Server using WebSphere Application Server Toolkit, manage the J2EE application on the server using the WebSphere Application Server administrative console, and read WebSphere Application Server log files to assist in problem determination of the server.

To keep up with all the Hello World tutorials and articles, check the [Hello World overview page](#).

Downloads

Description	Name	Size	Download method
Hello World EJB module	HelloWorldEJB.zip	39KB	HTTP
Hello World Web module	HelloWorldWeb.war	11KB	HTTP

[Information about download methods](#)

Resources

Learn

- [WebSphere Application Server Information Center](#)
- [Service Oriented Architecture \(SOA\)](#)
- [IBM SOA Foundation](#)
- [Redbook: WebSphere Application Server V6 Technical Overview](#)

Get products and technologies

- Download a free trial version of [WebSphere Application Server, Version 6.1](#).
- Download Apache HTTP Server (<http://apache.org>).
- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Participate in [Global WebSphere Community blogs](#) and get involved in the community.

About the author

Larina Vera



Larina Vera is an information developer for IBM Rational Application Developer and WebSphere Application Server Toolkit at the IBM Toronto Lab. As an information developer, she provides help documentation to the products she represents. Previously, she was a knowledge engineer and served as a support analyst on the WebSphere Studio support team. As a knowledge engineer, she worked towards delivering support information to the Web and the product help system. As a support analyst, she had direct customer interaction to solve problems and served as an advocate for customer concerns such as defects and feature requests. Larina received an honors bachelor in Computer Science from the University of Toronto in 2003.