Server Installation and Configuration

Sun® Microsystems provides to developers without charge a reference implementation of its enterprise software development specification, the *Java™ 2 SDK, Enterprise Edition* (*J2EE™ SDK* here, for brevity). Deployment of any application components must be done on a commercial J2EE-compliant application server. This workbook uses Release 1.3_01 of the J2EE SDK for all the EJB 2.0 examples from the O'Reilly book, *Enterprise Java Beans, 3rd Edition*, by Richard Monson-Haefel (here, "the EJB book").

As of the writing of this workbook, Version 1.3_01 is available from Sun's *Java Developer Connection*. This chapter will describe the basic steps of installing the server software, along with the software that is prerequisite for running the examples.

Even though all the required software is platform-independent and the examples can run on any platform, this workbook describes the setup procedures only for Microsoft Windows NT/2000 platforms. Linux and Solaris users are directed to Sun's site for download and installation instructions. More information is available at *http://java.sun.com/j2ee/sdk_1.3/index.html*.

Software Setup

The J2EE SDK relies on Java 2 Standard Edition (J2SE™) version 1.3.1_01. Following is a list of URLs for the software you will need to download before you can install the J2EE SDK. A Java IDE is preferable but not needed. In the spirit of future integration of J2EE SDK v 1.3_01 with Forte for Java, this workbook refers to the early-access version of Forte for Java, CE version 3.0.

J2SE™ version 1.3.1_01 — *j2sdk-1_3_1_01-win.exe*

http://java.sun.com/j2se/1.3/

J2EE SDK FCS Version 1.3_01 — *j2sdkee-1_3_01-win.exe*

http://java.sun.com/j2ee/sdk_1.3/index.html (requires login to Developer Connection)

Jakarta Ant (build tool: v 1.4.1) — *jakarta-ant-1.4.1-bin.zip*

http://jakarta.apache.org/builds/jakarta-ant/release/v1.4.1/bin/

Forte For Java CE 3.0 EAP — *forte_ce_3_ea.exe*

http://www.sun.com/forte/ffj (requires login)

Installing J2SE™ v1.3.1

Download the executable for the Microsoft Windows platform from the location indicated above and run the install application. This is a conventional self-extracting installation wizard; just follow the prompts as usual.

Figure 1: Installing the J2SE SDK

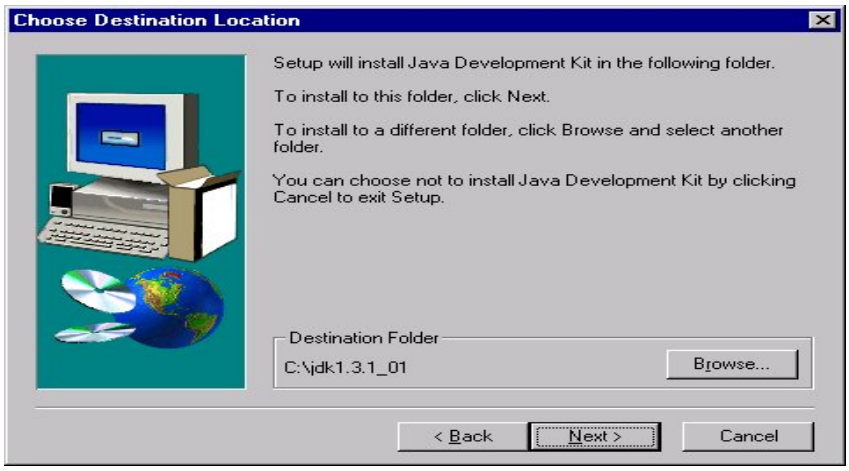
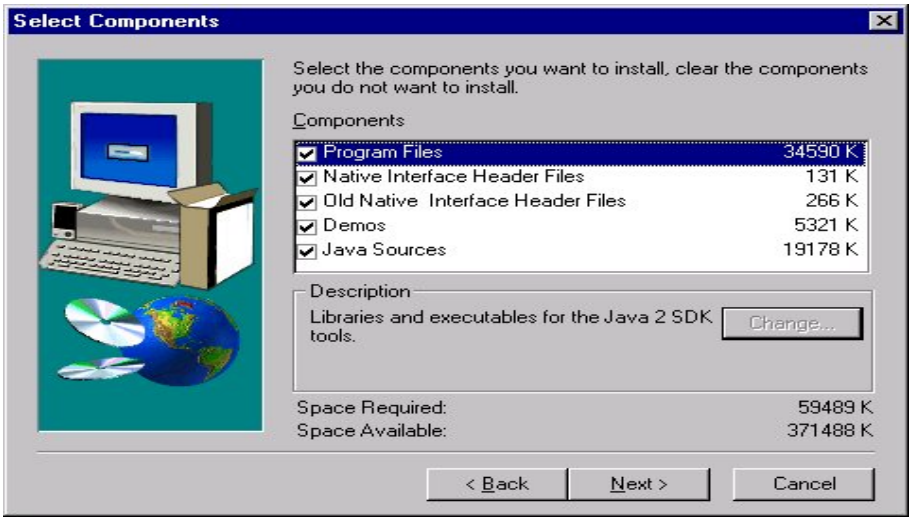


Figure 2: Installation continued



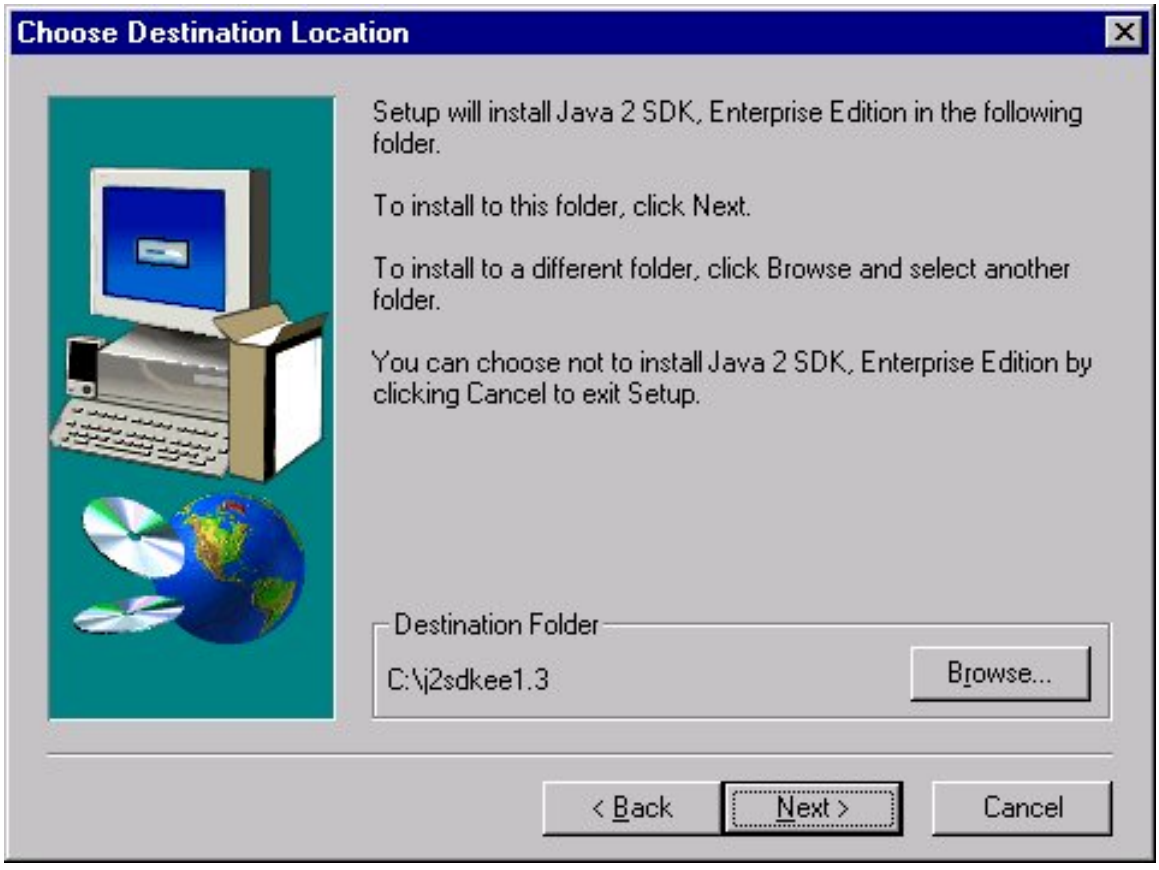
Installing Jakarta Ant v1.4.1

Ant is an open source platform-independent build tool available from the Apache Software Foundation. *Ant* is developed as a sub-project under the Jakarta Project specification from Apache. Extract the binary release of *Ant*, available in *.zip* format, to a local directory. For example, specifying **c:** will install *Ant* to the directory *c:\Jakarta-ant-1.4.1* (`$ANT_HOME`). For your use of this workbook you won't need any "optional tasks." If optional tasks for J2EE SDK become available later, consult the corresponding documentation and place the optional *.jar* files in the *$ANT_HOME\lib* directory.

Installing J2EE SDK FCS Release v1.3_01

Run the self-extracting install application *j2sdkee-1_3_01-win.exe* and accept the licensing agreement.

Figure 3: J2EE SDK installation wizard



This specification for the Destination Folder installs the reference implementation to the directory *c:\J2SDKEE1.3*. This workbook assumes that on the Windows platform you will use Drive C as the mount partition for all these installs. You can install the software in any available partitions. The configuration section below talks about required settings.

Installation of Forte for Java CE v 3.0 is left as an exercise for the reader.

Tools Configuration

Before running the J2EE SDK you must set up a few environment variables to ensure that the *Ant* build as well as the J2EE SDK will work for the examples.

In Windows NT/2000 open **Properties** for **My Computer** and select the **Environment** tab (NT) or the **Advanced** tab (Win 2K), then select **Environment Variables**, and specify the following:

- `JAVA_HOME=c:\JDK1.3.1_01`: home directory of J2SE™ v1.3.1
- `ANT_HOME=C:\JAKARTA-ANT-1.4.1`: home directory of Jakarta *Ant* build 1.3
- `J2EE_HOME=C:\J2SDKEE1.3`: home directory of J2EE SDK

Make changes to the path settings by pre-pending the following to the existing path:

```
PATH=%JAVA_HOME%\bin;%J2EE_HOME%\bin;%ANT_HOME%\bin;%path%
```

You have completed the initial configuration needed to run the J2EE SDK.

Verifying Installation of Required Software Tools

You will run most of the tools provided by the reference implementation from the command line. Details of each of the tools are provided later in the chapter.

Verifying J2SE™ Version

Open a command prompt and type

```
java -version
```

You should see the response:

```
java version "1.3.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1_01)
Java HotSpot(TM) Client VM (build 1.3.1_01, mixed mode)
```

Verifying Ant Installation

Open a command prompt and type

```
Ant
```

You should see the response:

```
Buildfile: build.xml does not exist!
Build failed
```

Verifying J2EE SDK Installation

Open a command prompt and type

```
j2ee -version
```

You should see the response

```
Java 2 Enterprise Edition version 1.3, build 1.3_01-b01
```

You have now verified the tools and completed installation of the required toolset.

J2EE SDK Environment

Sun Microsystems provides on-line documentation and a configuration guide for the J2EE SDK tools. Typically these documents will be available from *http://java.sun.com/j2ee/docs.html#sdk* for various releases. As of the writing of this workbook these documents are not available on-line but are available as part of the installation.

For more information see

%J2EE_HOME%\doc\release\ToolsOnly.html

$%J2EE_HOME%\doc\release\ConfigGuide.html

This workbook doesn't try to duplicate the information already available in these published documents, but rather describes the use of these tools to configure and run the examples in the EJB book. We also encourage the readers to visit the FAQ pages for the beta version of the reference implementation. The FAQ is available at

http://java.sun.com/j2ee/sdk_1.3/faq.html#ri

The J2EE SDK is built using a CORBA ORB. The directory structure of the J2EE SDK is as follows on a Microsoft Windows platform:

%J2EE_HOME% (e.g., *c:\j2sdkee1.3):* home directory for the J2EE SDK

- *\bin:* all batch scripts and tools required to run the SDK
- *\cloudscape:* Cloudscape databases
- *\conf:* DTDs, application environment settings, policy files, etc.
- *\config:* property files for the J2EE SDK environment configuration
- *\connector:* adapters for different connectors
- *\doc:* documentation
  - *\api:* javadocs for J2EE APIs
  - *\release:* various useful HTML documents about tools, configuration, CMP etc.

samples: bundled examples

help; help files: Java help for deploy tool

images: various image files

lib: all required Java classes, .jar files, client-.jars for Cloudscape, etc.

logs: system log directory

nativelib

public_html: where application web-archives are deployed in exploded fashion

repository

\<*Machine_Name*>

applications: custom application .jars deployment directory

db: JMS persistence components

gnrtrTMP

\<*enterprise archive folder*>*:* files related to each *.ear* file

objects

web: compiled JSPs and servlets, web-container utility java classes etc.

The *bin* directory contains the primary tools needed to run the reference implementation.

Starting J2EE SDK Tools

Start the J2EE reference implementation server and verify that the configuration is valid.

J2EE Server Startup

At a command prompt type

```
j2ee -verbose
```

The system should respond with the following information:

```
J2EE server listen port: 1050
Naming service started:1050
Binding DataSource, name = jdbc/DB2, url =
jdbc:cloudscape:rmi:CloudscapeDB;create=true
Binding DataSource, name = jdbc/InventoryDB, url =
jdbc:cloudscape:rmi:CloudscapeDB;create=true
Binding DataSource, name = jdbc/Cloudscape, url =
jdbc:cloudscape:rmi:CloudscapeDB;create=true
```

```
Binding DataSource, name = jdbc/EstoreDB, url =
jdbc:cloudscape:rmi:CloudscapeDB;create=true
Binding DataSource, name = jdbc/DB1, url =
jdbc:cloudscape:rmi:CloudscapeDB;create=true
Binding DataSource, name = jdbc/XACloudscape, url =
jdbc/XACloudscape__xa
Binding DataSource, name = jdbc/XACloudscape__xa, dataSource =
COM.cloudscape.core.RemoteX
aDataSource@77eaf8
Starting JMS service...
Initialization complete - waiting for client requests
Binding: < JMS Destination : jms/Topic , javax.jms.Topic >
Binding: < JMS Destination : jms/Queue , javax.jms.Queue >
Binding: < JMS Cnx Factory : QueueConnectionFactory , Queue , No
properties >
Binding: < JMS Cnx Factory : jms/TopicConnectionFactory , Topic , No
properties >
Binding: < JMS Cnx Factory : TopicConnectionFactory , Topic , No
properties >
Binding: < JMS Cnx Factory : jms/QueueConnectionFactory , Queue , No
properties >
Starting web service at port: 8000
Starting secure web service at port: 7000
J2EE SDK/1.3
Starting web service at port: 9191
J2EE SDK/1.3
J2EE server startup complete.
```

This output reports a clean installation of the J2EE SDK. If any startup problems arise, refer to the release notes for the version. One possible source of problems: a user running Microsoft Outlook or another MAPI client may encounter port conflicts.

- **Caution:** The server will permit you to configure it for a non-default Listen port, but the Deployer tool is more restrictive. It will deploy components only if the server is configured for port 1050.

Verify your configuration by bringing up the J2EE default home page: open a web browser and point it to *http://<J2EEServerHost>:8000*; for example, *http://localhost:8000*.

Figure 4: J2EE default home page

Cloudscape™ Server Startup

IBM's Cloudscape DBMS comes bundled with the J2EE SDK to support development of databases and related software. The workbook examples archive includes an additional script to run Cloudview, a graphical user interface for creating and managing Cloudscape databases. This script, called *cloudview.cmd,* is in the *$EXAMPLES_ROOT(EJBBook)\installscripts* directory. Cloudview cannot open a database with the default driver if the Cloudscape server is already connected to the database. Before opening the database in Cloudview, make sure that the Cloudscape server is not connected to the database. You can open Cloudview with another open connection using the RMI JDBC driver.

Note that this version of the J2EE SDK does not include the *tools.jar* required for Cloudview to run, but the EJB book's downloaded code includes two *.jar* files needed to run the Cloudview utility: *cloudview.jar* and *jh.jar,* in the *EJBbook\cloudscapejars* directory. Optionally you may download these *.jar* files from http://www.cloudscape.com/support/downloads.jsp

Place these two *.jar* files in *%J2EE_HOME%\lib\system*.

Place the *cloudview.cmd* file in *%J2EE_HOME\bin*. Start and shut down the Cloudscape server.

To start the Cloudscape server, at a command prompt type

```
cloudscape -start
```

You should see:

```
Sun Oct 28 23:15:58 MST 2001: [RmiJdbc] Starting Cloudscape RmiJdbc
Server Version 1.7.2 ...
Sun Oct 28 23:15:59 MST 2001: [RmiJdbc]
COM.cloudscape.core.JDBCDriver registered in DriverManager
Sun Oct 28 23:15:59 MST 2001: [RmiJdbc] Binding RmiJdbcServer...
Sun Oct 28 23:15:59 MST 2001: [RmiJdbc] No installation of RMI
Security Manager...
Sun Oct 28 23:15:59 MST 2001: [RmiJdbc] RmiJdbcServer bound in rmi
registry
```

To shut down the server, type the command:

```
cloudscape -stop
```

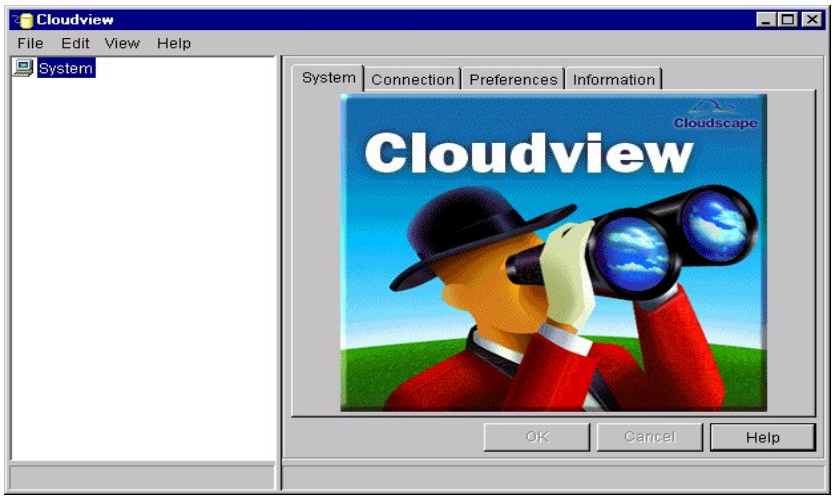
The system will respond with:

```
URL=[jdbc:rmi:jdbc:cloudscape:]

Attempting to shutdown RmiJdbc server
RmiJdbc Server RmiAddr is: //w157452/RmiJdbcServer

WARNING: Shutdown was successful!
```

To run the Cloudview console shown in the next figure, type the command:

```
cloudview
```

Figure 5: Cloudview console



Cloudscape also comes with a command-line utility for running SQL scripts and commands. This utility is bundled in the `-isql` option of the Cloudscape script. This connects your server/Cloudview to the default database called *CloudscapeDB* with the RMI JDBC driver. The Cloudscape server should be running for this command-line utility to work.

- **Note:** On the **Preferences** tab, uncheck the **Save DDL as case insensitive** option.

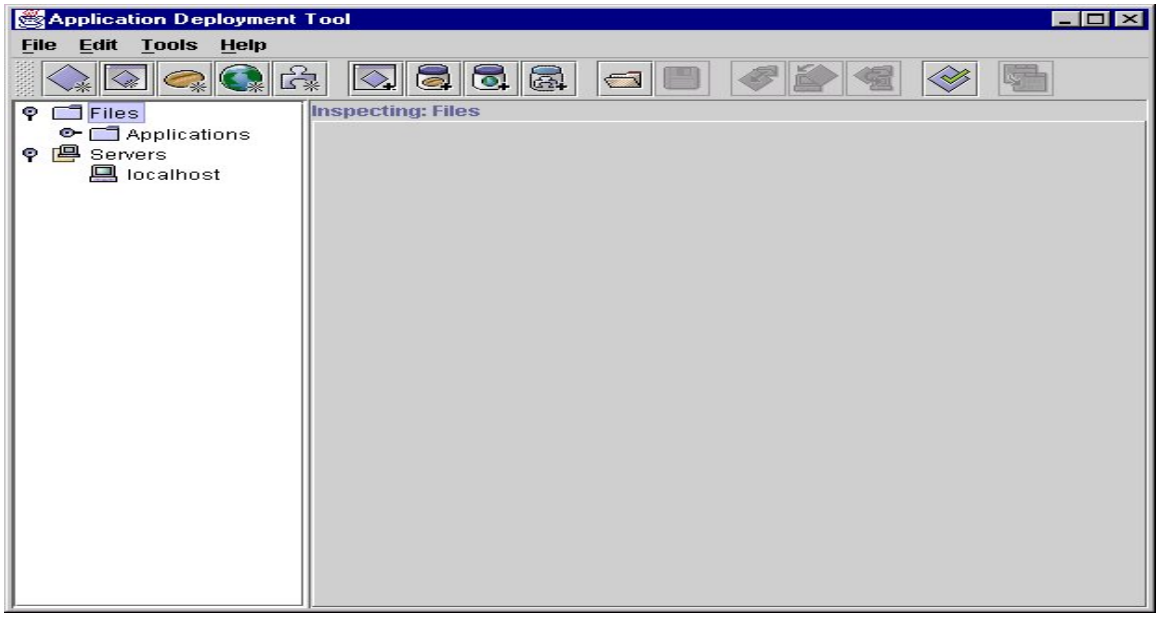
Deployer Tool Startup

For packaginng applications, the J2EE SK provides a graphical tool named *Deployer*. You will use this tool extensively throughout the workbook for packaging the components into EJBs (*.jar*), web application archives (*.war*) and enterprise archive (*.ear*) files. The *Ant* utility can perform these tasks, but in this workbook you will be using *Ant* only to compile the Java classes. Also, the Sun reference implementation follows certain extensive descriptive mechanisms in the XML descriptors. Wild-card mechanisms in the descriptor files make the deployment of components confusing. To simplify things, and to foster a better understanding of the packaging mechanism, this workbook encourages you to use the *Deployer* tool.

To start *Deployer* type at the command prompt:

```
deploytool
```

Figure 6: Deployer



Once you have a good grasp of the deployment process, you will want to move beyond the *Deployer* tool. When you develop real applications in a commercial application server environment, you will want to use *Make*, *Ant*, and other building tools to automate the task of compiling, packaging, and deploying components to the target environment. The J2EE SDK comes bundled with other command-line utilities for verifying and packaging the components. The *bin* directory under *$J2EE_HOME* contains *verifier*, *packager*, *realmtool*, *keytool*, *cleanup*, and others. Check the on-line documentation provided at the Sun site or at *$J2EE_HOME\doc\index.html*.

Running J2EEAdmintool

J2EEAdmintool is another command-line utility provided by the J2EE SDK, to configure the resources for the server. Type the following command to see a list of available options:

```
j2eeadmin –help
```

The response will be:

```
Usage:
-addJdbcDriver <class name>
-addJdbcDatasource <jndi name> <url>
-addJdbcXADatasource <jndi name> <class name> [<xa user name> <xa
password>] [-props (<name>=<value>)+]
-addJmsDestination <jndi name> (queue|topic)
-addJmsFactory <jndi name> (queue|topic) [-props (<name>=<value>)+]
-addConnectorFactory <jndi name> [<app name>:]<rar filename> [<xa
user name> <xa password>] [-props (<name>=<value>)+]
-list<resource type>
-remove<resource type> <name>
-removeAll<resource type>
```

You now have at least a passing acquaintance with most of the tools needed for compiling, packaging, deploying, and running the Sun J2EE reference implementation.

Exercise Code Setup and Configuration

So far we have discussed the tools needed and the server configuration for the J2EE SDK toolkit. Before you start working with the examples you need to prepare your environment for running the applications. For all the exercises, you will issue two commands to start up first the Cloudscape server, then the J2EE reference implementation server:

```
cloudscape –start
j2ee –verbose
```

Once the servers are started you can work with the component deployments. This workbook assumes the use of a separate JDBC data source and a separate Cloudscape database, which you will create now. You could use Cloudview to create the database, but here you will create it on the fly, from the first example.

Adding the Data Source and the Database

You'll begin by adding a new JDBC data source called *jdbc/Titan* that will point to a database named *TitanDB*. You will use this database for the workbook examples for the Titan Cruise application.

First list all the existing JDBC DataSource objects that are bound to JNDI.

```
j2eeadmin -listJdbcDatasource
```

```
JdbcDatasource
--------------
< JDBC Resource : jdbc/EstoreDB ,
<dbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/DB1 ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/InventoryDB ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/DB2 ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/Cloudscape ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
```

This lists the current available data source objects bound to JNDI.

Add a new data source called *Titan* to the JNDI for a database called *TitanDB* with a `j2eeadmin` command that uses the `–addJdbcDatasource` option and specifies the JNDI name and URL:

```
j2eeadmin -addJdbcDataSource jdbc/Titan
   jdbc:cloudscape:rmi:TitanDB;create=true
```

List the data sources again to verify that your newly created data source exists:

```
j2eeadmin –listJdbcDatasource
```

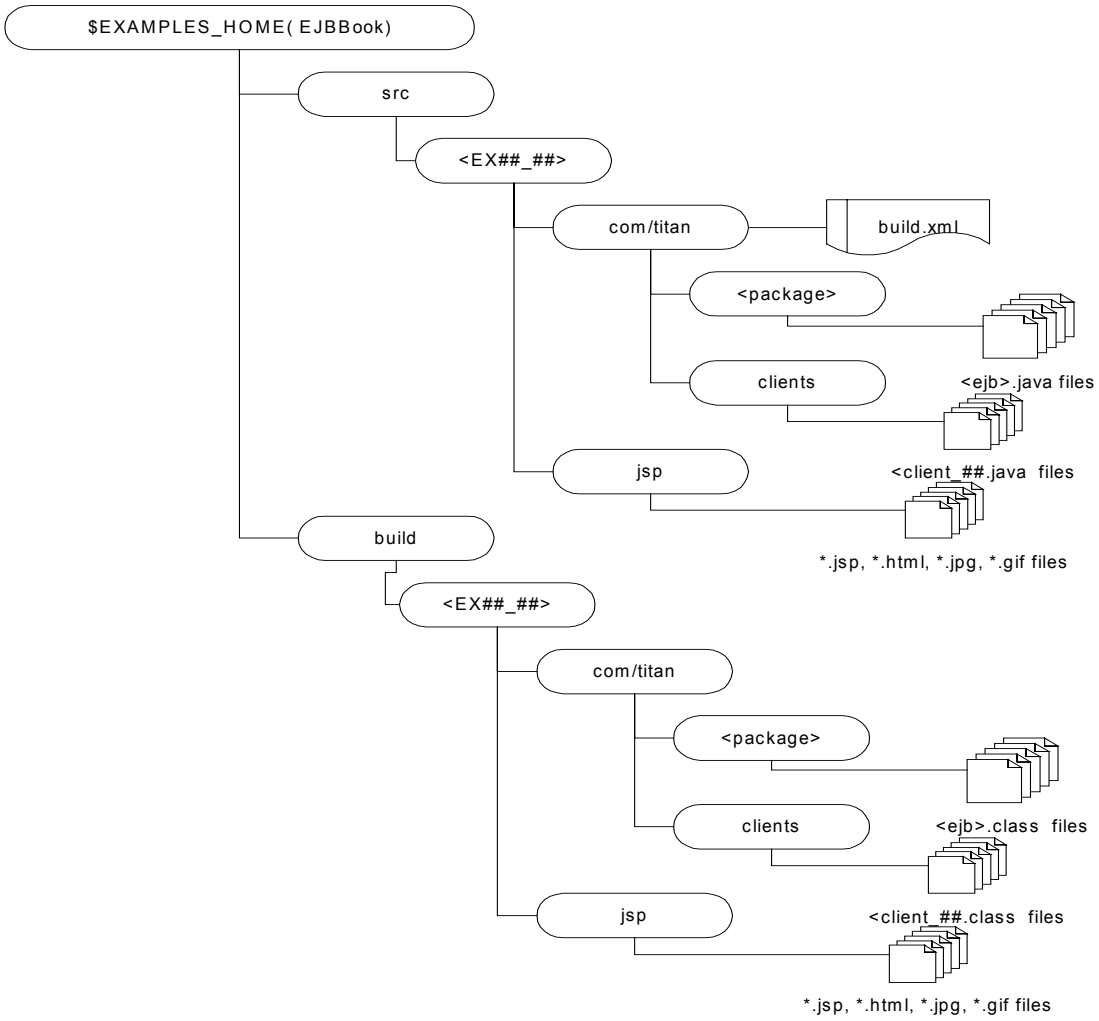
```
JDBCDatasource
--------------
< JDBC Resource : jdbc/Cloudscape ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/EstoreDB ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/Titan ,
jdbc:cloudscape:rmi:TitanDB;create=true >
< JDBC Resource : jdbc/DB2 ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/InventoryDB ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
< JDBC Resource : jdbc/DB1 ,
jdbc:cloudscape:rmi:CloudscapeDB;create=true >
```

This report shows the newly added *Titan* JDBCDatasource bound to JNDI under the name *jdbc/Titan*.

Exercise Code Configuration

The code for the EJB book examples is organized in a hierarchical directory structure. Once downloaded, the examples archive should be expanded. Each directory represents coding examples for a given chapter. This workbook will assume the following directory structure from a given Home directory. While building the examples, `c:\EJBBook` has been used as the `$EXAMPLES_HOME`.

Figure 7: Example code structure



Each *ex##_##* directory represents the root directory for an exercise. Some of the EJBs are created in one exercise and used in others. In further exercises they are modified to extend their functionality. To simplify the process and further the reader's understanding, this workbook obliges the reader to rebuild and deploy each exercise as a unit of work. Experienced users can eliminate the steps of re-deploying existing EJBs. Examples for Chapters 4 through 6 are packaged into one single *.ear* file, while the rest are packaged individually.

The download site for this workbook contains individual *.jar* or *.zip* archives for each exercise in the EJB book. Each downloaded exercise typically contains the following files:

- *build.xml*: *Ant* build script for compiling the Java classes and moving them to a staging directory
- *ejb-jar.xml*: decriptor file for the EJBs
- Source code for the EJBs
- Source code for application clients
- JSP pages for the web application client.

You are finished with the preparation, setup, and configuration of the J2EE SDK tools, the examples directory structure, and other needed tools. Now you can build, deploy the examples, and enjoy.

Time for a tea break!