



CHOOSE LANGUAGE

PRINTABLE VERSION

Docs & Support

[HOME](#) > [Docs & Support](#) > [NetBeans IDE 5.0](#)

NetBeans IDE 5.0 Quick Start Guide for J2EE Applications

[NetBeans IDE 5.5](#)

[NetBeans IDE 5.0
Using NetBeans
Installation Guide](#)

[NetBeans IDE 4.1](#)

[Older Releases](#)

[User FAQs](#)

[Articles](#)

[Flash Demos](#)

[Books](#)

[Learning Java](#)

[Module Developer's
Resources](#)

[NetBeans Support](#)

This tutorial will take you through the basic steps of creating entity Enterprise JavaBeans™ components with container-managed persistence (CMP entity beans). We will be using NetBeans™ IDE 5.0 to generate the CMP entity beans from an existing Apache Derby database. We will then create a simple session bean to access the database information and a Web module client.

This tutorial is meant to be a quick tour of the J2EE development features in NetBeans IDE 5.0 and not an instructional guide to programming enterprise applications. Although you can get through the tutorial without any prior knowledge of J2EE technology, a better starting point for beginners is the [J2EE Tutorial in NetBeans IDE](#).

Note: This tutorial is designed for use with the Sun Java™ System Application Server 8.2 and its bundled database server, Apache Derby. Earlier versions of the Sun Java System Application Server use PointBase as the bundled database server. To complete this tutorial with PointBase, see the [NetBeans IDE 4.1 Quick Start Guide for J2EE Applications](#).

This document covers the following topics:

- [Getting Started](#)
 - [Setting Up Your Environment](#)
 - [Configuring the Derby Database](#)
- [Coding the EJB Module](#)
 - [Creating the Enterprise Application Project](#)
 - [Generating the CMP Entity Beans](#)
 - [Coding the Session Bean](#)
- [Coding the Web Module](#)
 - [Creating a Customer Service Locator](#)
 - [Coding the Servlet](#)
- [Building and Deploying the Application](#)
 - [Setting the Default Web Page for the Application](#)
 - [Running the Application](#)
 - [Troubleshooting](#)
- [Next Steps](#)

Getting Started

Before you begin, you have to make sure you have all of the necessary software. You also have to configure the Derby database and populate the database tables that we will be using to generate our CMP entity beans.

Setting Up Your Environment

First, you need to install the Sun Java System Application Server Platform Edition 8.2 ([download](#)) on your computer.

Note: If the IDE runs on JDK 5.0, then the application server also needs to use the virtual machine from JDK 5.0. To set the JDK that the IDE uses, open `IDE_INSTALL_DIR/etc/netbeans.conf` and enter the path to the JDK in the `netbeans_jdkhome` property. To set the JDK that the application server uses, edit the `AS_INSTALL_DIR/config/asenv(.bat)` file and change the `AS_JAVA` environment variable.

Once you have installed the application server, you need to register it with NetBeans IDE. Note that if you downloaded and installed the version of NetBeans IDE that comes bundled with the application server, you do not have to perform this step. The IDE knows the location of the bundled application server.

1. Open NetBeans IDE 5.0 by going to the `NETBEANS_INSTALL_DIR/bin` directory and running the `netbeans` command. If `java` is not in your `PATH` variable start the IDE with the `--jdkhome /path/to/jdk` switch on the command line or add the line `netbeans_jdkhome="/path/to/jdk"` to your `NETBEANS_INSTALL_DIR/etc/netbeans.conf` file.
2. In the IDE, choose `Tools > Server Manager` from the main window.
3. Click `Add Server`. Select `Sun Java System Application Server` and give a name to the instance. Then click `Next`.
4. Specify the installation directory of the application server (for example, `C:\Sun\Appserver`).
5. Leave the `Register Local Default Domain` radio button selected and select a domain.
6. Optionally, click `Next` and enter your administrator username and password. If you do not want to store the username and password in your IDE user directory, you can leave these fields blank. The IDE will prompt you every time it needs the information.
- Note:** The default admin password is `adminadmin`.
7. Click `Finish`. The IDE registers the server and lists it under the `Servers` node in the `Runtime` window.

Finally, start the application server:

1. In the `Runtime` window, right-click the application server node and choose `Start Server`.
2. Expand the application server node. The node contains subnodes for all of the categories you would see in the application server's admin console. You can configure the application server by right-clicking any node and choosing `Properties`.

Sample Projects

Just want to play with some J2EE projects? In the IDE, choose `File > New Project`, then expand `Samples` folder. The IDE includes samples from the [Java BluePrints Solution Catalogue](#).

[Feedback](#)

Search

Related Documents

Importing Existing Enterprise Applications into NetBeans 5.0

A short guide to getting your existing enterprise application and EJB modules into NetBeans IDE 5.0.

J2EE Tutorial in NetBeans IDE

A version of selected chapters from the `java.sun.com` J2EE 1.4 Tutorial adapted for NetBeans IDE.

NetBeans IDE 5.0 Quick Start Guide

Quickly create, build, and execute a simple J2EE application.

NetBeans IDE 5.0 Quick Start Guide for Web Applications

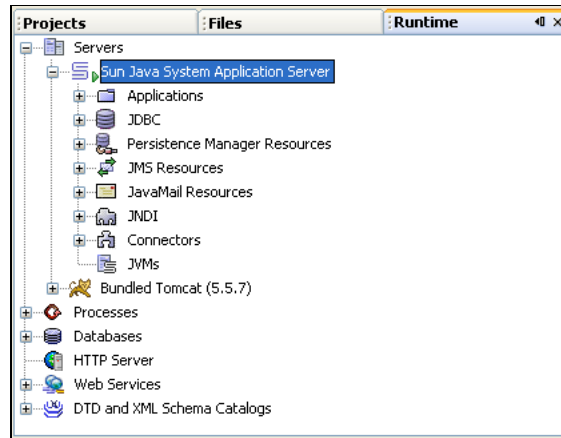
Create and deploy a simple web application.

NetBeans IDE 5.0 Quick Start Guide for Web Services

Create, expose, and consume a simple web service.

Documentation List for NetBeans IDE 5.0

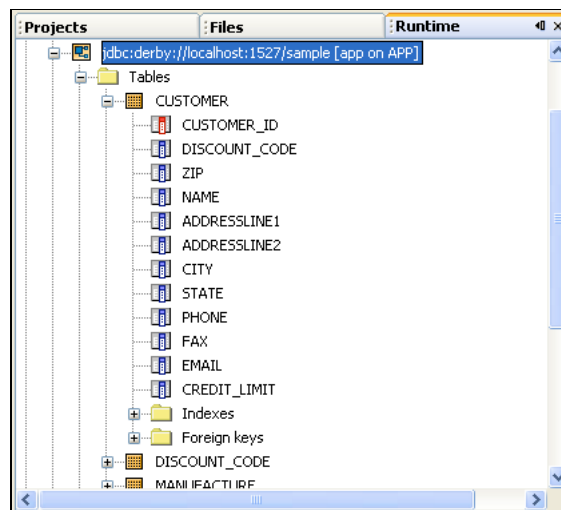
Complete list of documentation for NetBeans IDE 5.0.



Configuring the Derby Database

In this tutorial, you will generate CMP beans from one of the tables that are included with the Derby database server. The Derby database server is included in the Sun Java System Application Server 8.2 download.

1. In the Runtime window, expand the Databases node. You should see a node for the `jdbc:derby://localhost:1527/sample` database.
2. Right-click the `jdbc:derby://localhost:1527/sample` database and choose Connect. If prompted for a password, type `app` for the user and password and click OK.
3. Expand the `sample` database node and the Tables node. Make sure that the `CUSTOMER` node is present, as shown below.



Note: If you have altered the `sample` database, you can repopulate it using [this SQL script](#).

Coding the EJB Module

Coding enterprise beans is easy. The IDE takes care of all the implementation details for you, so you can concentrate on coding the business logic of your EJB module.

Creating the Enterprise Application Project

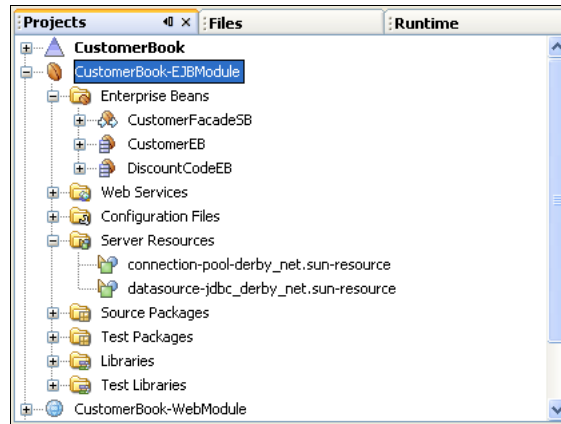
For this example, we will create an Enterprise Application project as a container for our Web module and EJB module. The Enterprise Application template lets you create the projects for your Web module and EJB module automatically.

1. Choose File > New Project (Ctrl-Shift-N) and select the Enterprise Application template from the Enterprise category. Click Next.
2. Name the application `CustomerBook` and specify a location for the project. Leave the rest of the settings at their default values and click Finish.

Generating the CMP Entity Beans

Now we are going to generate the entity beans that will access the `sample` database. We need one entity bean for each of the tables our program will access.

1. In the Projects window, right-click the EJB module's node (`CustomerBook-EJBModule`) and choose New > CMP Entity Beans from Database.
2. Select JDBC Connection as the database source and choose the `jdbc:derby://localhost:1527/sample` connection from the combo box.
3. For the package name, type `ejb` and click Next. The wizard shows you all of the tables in the selected database.
4. Select the `CUSTOMER` table, click Add, then click Finish.
The IDE creates the CMP bean for the `CUSTOMER` table and for any tables that are referenced from `CUSTOMER` (`DISCOUNT_CODE`). The IDE also creates a new JDBC connection pool and data source which it will register on the application server when you deploy the project.



Coding the Session Bean

Now let's create a session bean that will handle access between the Web application client and the information in the entity beans. We will create an empty session bean, generate a call to the entity bean, and add some business methods.

1. In the Projects window, right-click the CustomerBook-EJBModule project node and choose New > Session Bean. Name the session bean `CustomerFacade`, and place it in the `ejb` package. Set the bean to be stateless and to have only remote interfaces. Then click Finish. The IDE creates the bean and opens its bean class in the Source Editor.
2. In the Source Editor, right-click anywhere in the bean class and choose Enterprise Resources > Call Enterprise Bean. Select the CustomerEB bean and click OK. The IDE generates a `lookupCustomerBean` method at the bottom of the source file.
3. Add the following to the variable declaration to the beginning of the bean class:

```
private ejb.CustomerLocalHome custHome;
```

4. Now let's obtain a reference to the home interface of the entity bean in the session bean's create method. Delete the comments in `ejbCreate()` and add the following statement:

```
custHome = lookupCustomerBean();
```

5. Right-click anywhere in the bean class' body and choose EJB Methods > Add Business Method. Specify the following information:

Element	Value
Method name	<code>getCustomerInfo</code>
Return type	<code>String</code>
Paramater	<code>int custId</code>
Exception	<code>javax.ejb.FinderException</code>

6. In the Source Editor, edit the `getCustomerInfo` business method to look like the following:

```
public String getCustomerInfo(int custId) throws javax.ejb.FinderException {
    ejb.CustomerLocal customer = custHome.findByPrimaryKey(new Integer(custId));
    return "Name: " + customer.getName() + ", E-mail: " +customer.getEmail();
}
```

The final `CustomerFacadeBean.java` file should look like [this](#).

Coding the Web Module

Now we need to code a Web module that will provide the user interface for our entity beans. The Web module contains one servlet that allows the user to search for customers by customer number.

Creating a Custom Service Locator

In the last section, we had the IDE generate inline lookup code when calling an enterprise bean. In this section, we will create a custom service locator that the IDE will use when generating calls to the enterprise bean. Our service locator doesn't do anything special, but you could customize the way your enterprise application locates enterprise beans.

1. In the Projects window, right-click the CustomerBook-WebModule node and choose New > File/Folder. In the wizard, select the Service Locator template from the Enterprise category. Click Next.
2. Type `ServiceLocator` as the Class Name and `web` as the Package, and click Finish.

Coding the Servlet

For the final step, we will add a servlet to the Web module that lets you look up and display the information for each of the customers in our `CUSTOMER` table.

1. In the Projects window, right-click the CustomerBook-WebModule node and choose New > Servlet. Name the servlet `CustomerDetail` and put it in the `web` package. Then click Finish.
2. In the Source Editor, right-click anywhere in the servlet class and choose Enterprise Resources > Call Enterprise Bean. Select the CustomerFacade session bean, set the `web.ServiceLocator` class as service locator strategy, and click OK. The IDE inserts the lookup method.
3. Edit the `processRequest` method to look like the following:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

```

response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet customerDetail</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Servlet customerDetail at " + request.getContextPath () + "</h1>");

String customerNr = request.getParameter("customer_nr");
if((customerNr != null) && !(customerNr.equals(""))){
    {
    try{
   .ejb.CustomerFacadeRemote custFacade = lookupCustomerFacadeBean();
    out.println("Customer's info for nr. " + customerNr + ": " + custFacade.getCustomerInfo(
        Integer.parseInt(customerNr));
    }catch(javax.ejb.FinderException ex){
        out.println("Customer with nr. " + customerNr + " not found");
    }
    }

    out.println("<form>");
    out.println("Customer number: <input type='text' name='customer_nr' />");
    out.println("<input type=submit value=Select />");
    out.println("</form>");

    out.println("</body>");
    out.println("</html>");
    out.close();
}
}

```

The final `CustomerDetail.java` servlet should look like [this](#).

Building and Deploying the Application

Our enterprise application is all ready to be deployed to the application server. There's no more configuration of deployment descriptors necessary. The IDE has already configured the deployment descriptors and prepared a new connection pool and data source for our enterprise application.

Setting the Default Web Page for the Program

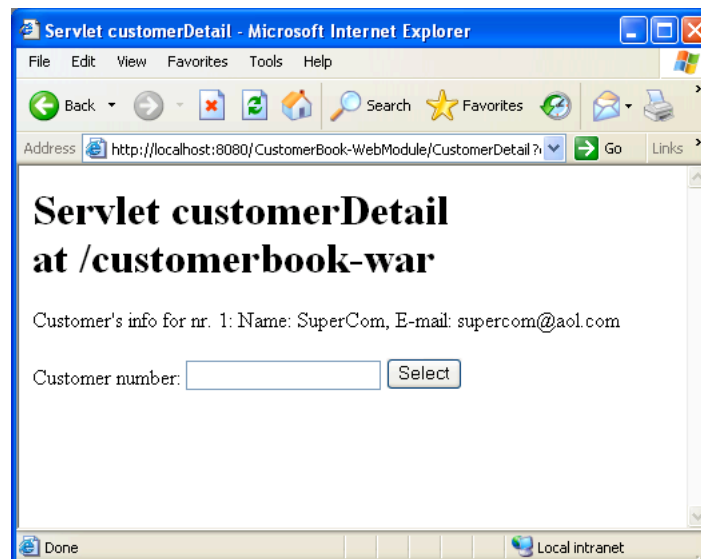
By default, a Web application displays its `index.jsp` page when it is run. Since our `index.jsp` is blank, we want the `CustomerBook` project to display the `CustomerDetail` servlet instead.

1. In the Projects window, right-click the `CustomerBook` project and choose Properties. Then click Run in the left panel.
2. Type `/CustomerDetail` in the Relative URL field.

Running the Application

1. Right-click the `CustomerBook` project and choose Run Project. The IDE does all of the following:
 - Builds the enterprise application project and all of its subprojects (the Web module project and EJB module project).
 - Starts the server if it is not already running.
 - Undeploys the enterprise application if it is already deployed to the application server.
 - Deploys the enterprise application to the application server.
 - Opens the Web module at the specified relative URL.

You should see the page below in your external browser. When you type a customer number and press Enter, the page should display the information for the customer.



Troubleshooting

Some common problems are:

- Opening `CustomerDetail` gives you an HTTP Status 404 error. This means that the application server could not find the `CustomerDetail` servlet. Make sure that the context root (`/CustomerBook-WebModule`) matches what is set for the

- Web module (right-click Web module project node > Properties > Run > Context Path).
- Submitting a number in the CustomerDetail form gives you an `javax.transaction.TransactionRolledbackException: CORBA TRANSACTION_ROLLEDBACK` error. This usually means that the server cannot access the database. If this occurs, do the following:
 - Make sure that the Derby database server is running. To start Derby, choose Tools > Derby > Start Derby Server.
 - Make sure that the resource pool and data source were correctly registered on the application server. To check this, go to the Runtime window and expand the application server's JDBC node. The JDBC Resources node should contain a `jdbc/derby_net` and the Connection Pools node should contain a `derby_netConnectionPool` node. To register missing connection pools and data sources:
 1. Expand the CustomerBook-EJBModule project's Server Resources node.
 2. Right click the `connection-pool-derby_net.sun-resource` node and choose Register.
 3. Follow the same procedure for the `datasource-jdbc_derby_net.sun-resource` node.

Note: You can view the server log by going to the Runtime window, right-clicking the application server node, and choosing View Server Log.

Next Steps

For more information about using NetBeans IDE 5.0, see the following resources:

- NetBeans IDE 5.0 Quick Start Guides:
 - [For Basic Java Applications](#)
 - [For Web Applications](#)
 - [For Web Services](#)
 - [For J2ME MIDP Applications](#)
- NetBeans IDE 5.0 Import Guides:
 - [For Basic Java Applications](#)
 - [For Web Applications](#)
 - [For J2EE Applications](#)
 - [For J2ME MIDP Applications](#)
- [NetBeans IDE Support and Docs page](#)

To send comments and suggestions, get support, and keep informed on the latest developments on the NetBeans IDE J2EE development features, [join the nbj2ee@netbeans.org mailing list](mailto:nbj2ee@netbeans.org).