

Installing the hsqldb database

As chapter 9 of *JSTL in Action* describes, JSTL's database tags are designed to work with nearly any relational database. If you or your organization already has a relational database set up, and if you have access to this database, you should be able to start using it with JSTL immediately. You'll simply need to find out from your database administrator (DBA) what credentials to use, where the database is located, and so on. (See chapter 9 for more details about connecting to a database.)

However, if you're an independent developer, or if you're new to databases, you might want to set up a sample database for experimental purposes. Fortunately, many high-quality databases—and, of course, some low-quality ones—are available for free. While some databases, like *Oracle*, are typically very expensive to purchase and maintain, systems like *PostgreSQL*, *MySQL*, and *hsqldb* are free, and they're also relatively easy to set up and manage.

For the purposes of simple experimentation, *hsqldb* is probably the lightest-weight and easiest free alternative to set up. *PostgreSQL* is a very high-quality product, suitable even for many demanding, mission-critical applications, and *MySQL* is also highly regarded by many developers. However, I've decided to discuss *hsqldb* because it's written entirely in Java and should run on any platform on which JSTL is available.

This tutorial discusses downloading, installing, and using *hsqldb* in its simplest mode, called “standalone” mode. The goal is not to teach you how to manage a public, robust database. That's an advanced topic beyond the scope of *JSTL in Action*. Instead, I just want to make sure you can learn JSTL's database tags without relying on a database administrator or an Oracle database!

Downloading hsqldb

Hsqldb is a free, open-source relational-database management system (RDBMS) written entirely in Java. It is the descendent of a product known as HypersonicSQL. When this tutorial was written, the main web site for *hsqldb* was <http://hsqldb.sourceforge.net>.

This tutorial discusses version 1.61 of *hsqldb*. You can download this version from the product's central web site. You are looking for the file called `hsqldb.jar`, which is available as part of a file named `hsqldb_v.1.61.zip`.

Once you download and extract `hsqldb.jar`, you should insert it into your application's `WEB-INF/lib` directory. As I discuss in my [tutorial on setting up Tomcat](#), this directory is used to store JAR files that provide supplemental Java logic for your application. The *hsqldb* database provides such supplemental logic. Therefore, once you've added the JAR file to `WEB-INF/lib`, your application is ready to use *hsqldb*. However, before you let your web application use *hsqldb*, you might want to set up some tables for the application to use.

Setting up tables in hsqldb

Hsqldb comes with a tool called *Database Manager* that lets you set up and manage databases. For instance, you can use Database Manager to create tables that your web application can use. In this section, we'll explore setting up and using the Database Manager. Along the way, we'll use a few SQL statements. For an introduction to using SQL in JSTL's tags, see appendix C in *JSTL in Action*; appendix D in *JSTL in Action* contains pointers to other database-related references.

Starting the Database Manager

To run this tool, you'll first need to add the `hsqldb.jar` file to your `CLASSPATH`. Note that this procedure is different from adding the JAR file to your web application's `WEB-INF/lib` directory. (Adding the JAR to `WEB-INF/lib` lets your application access hsqldb, but it doesn't help you run Database Manager directly. That is, Database Manager isn't a web-based tool; it's a Java application that will run right on your desktop.)

Let's walk through the procedure on Windows and Unix. On Windows, you'll first need to get to a command prompt. You can do this by going to the Start Menu, choosing Run, and typing `command` (for Windows 95, 98, and ME) or `cmd` (for Windows NT or 2000).

Once that the prompt, which on Windows will most likely look like `c:\>`, you will want to type the following two commands:

```
set CLASSPATH=path-to-hsqldb.jar
java org.hsqldb.util.DatabaseManager
```

where ***path-to-hsqldb.jar*** is the full file path to the `hsqldb.jar` file. For example, if you have put this file in `c:\temp`, then the first command would look like this

```
set CLASSPATH=c:\temp\hsqldb.jar
```

On Unix, the procedure is similar but not identical. In fact, depending on what shell—or command interpreter—you use, the procedure may be slightly different. The goal is first to set the `CLASSPATH` environment variable to include the `hsqldb.jar` file. In the Bourne shell (or its descendents, like `bash`, which is a common shell on Linux systems), command prompts typically end with the `$` character, and you will want to run the following commands:

```
CLASSPATH=path-to-hsqldb.jar
export CLASSPATH
```

where, again, ***path-to-hsqldb.jar*** points to the full file path for `hsqldb.jar` (for instance, `/home/user/hsqldb.jar`). On CSH-style shells (where command-prompts typically end with the `%` character), you will instead want to run the following command:

```
setenv CLASSPATH path-to-hsqldb.jar
```

Finally, once the `CLASSPATH` has been set appropriately, you may run the same `java` command as on Windows:

```
java org.hsqldb.util.DatabaseManager
```

Using the Database Manager

Once you run the Database Manager successfully, a window that looks like figure 1 will appear.

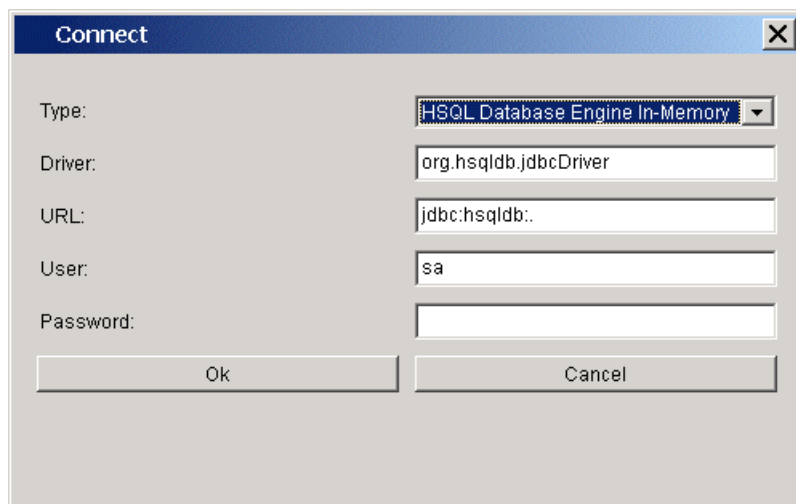


Figure 1 When Database Manager first runs, it displays a dialogue window that lets you describe your hsqldb operating environment.

This window requires that you describe your hsqldb operating environment. You can accept most of the default values this window presents, but to follow along with the examples here, you should make two changes. First, for the “Type” field, instead of leaving the default option—“HSQL Database Engine In-Memory”—you should use the second option: “HSQL Database Engine Standalone.”

Advanced note: Hsqldb supports several modes of operation. The default, “in-memory” mode simply keeps track of all data in memory. This is very fast, but it might be frustrating for all data to disappear each time you shut down your machine—or even just your JSP container. It also makes it harder to set up tables with tools like the Database Manager application. “Standalone” mode writes changes to disk, but it has an important restriction: only one application may use the database at once. Hsqldb supports other modes to get around this restriction, such as a general-purpose client/server mode. I do not describe these more advanced modes here, however.

The second change you need to make is a little trickier. When you change the “Type” to “HSQL Database Engine Standalone,” the “URL” field becomes `jdbc:hsqldb:test`. Instead of `test`, you want the final part of this URL to be a filename in a known location. The specific location doesn’t matter; it can be anywhere on your disk. But it should point to a valid directory and a file that doesn’t initially exist. On Windows, for example, the URL `jdbc:hsqldb:c:\temp\test` would work if you had a `c:\temp` directory. Similarly, on Unix, you might use `jdbc:hsqldb:/tmp/test`.

Warning: The URLs used here might look unfamiliar; they don’t begin with `http://`, like most URLs for the web. Don’t worry about this, however. These paths simply use a more general

URL syntax that's used by Java to access databases. It's only important that you remember the URL and use it consistently later.

Once you have decided on a URL, press the "OK" button. If any errors have occurred, the window will display them and give you an opportunity to try a different URL. If the URL is accepted, however, then the dialogue window will go away, and you'll see instead a window that looks like figure 2. The window in the upper right is used for entering SQL commands directly to the database; the window below it is used to display any results of your SQL commands.

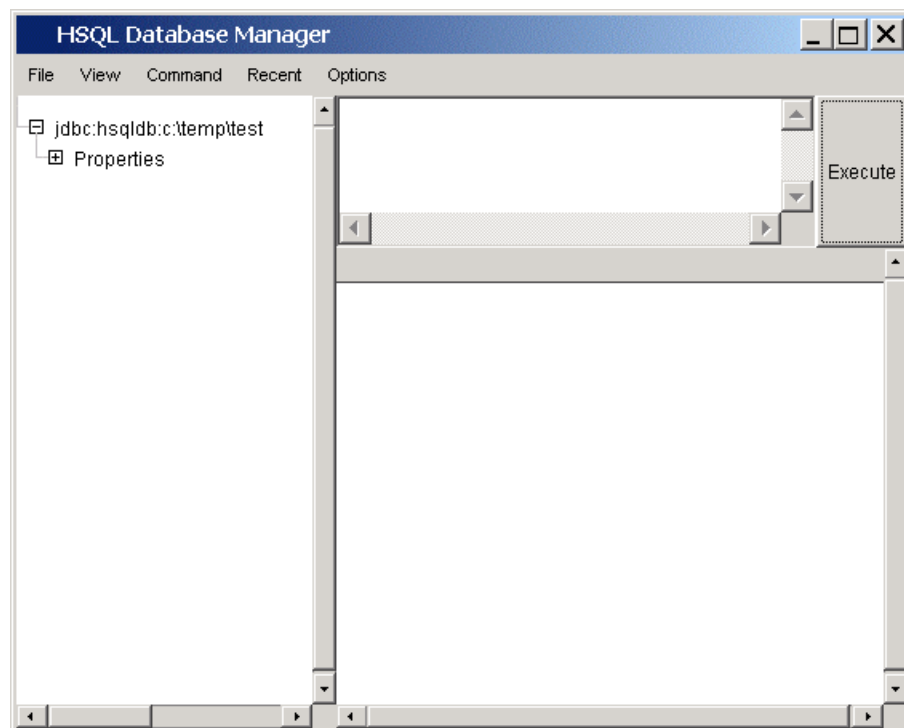


Figure 2 Database Manager's main operational window.

Creating a table

Let's use this upper-right box to create a table using SQL. Recall from appendix C that the following is a valid SQL command to create a simple table:

```
CREATE TABLE PEOPLE (
  NAME VARCHAR(255),
  AGE INTEGER,
  WEIGHT INTEGER
)
```

If you type this command into the upper-right box of Database Manager and click the "Execute" button, a table called `PEOPLE` will be created. This table would now be ready to use—that is, ready to accept rows of information.

Modifying data

Let's make use of this new `PEOPLE` table by inserting some data into it. Recall from appendix C that the following SQL command inserts a new row into the `PEOPLE` table:

```
INSERT INTO PEOPLE(NAME, AGE, WEIGHT)
VALUES('John Smith', '59', '170')
```

If we enter this SQL command into the Database Manager and press “Execute,” hsqldb will add the new row to our database. Note that as figure 3 shows, in the window on the bottom right, the Database Manager will respond with the number of rows that were modified (in this case, just one, for the one new row that was added).

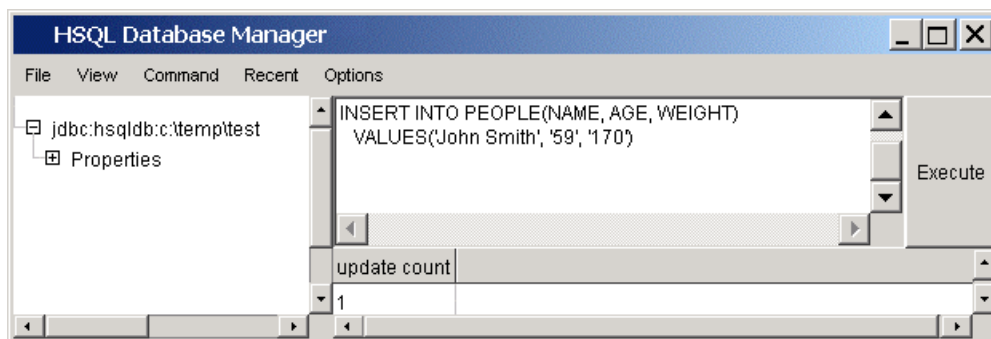


Figure 3 When Database Manager updates a table, it displays the number of rows that were modified in a pseudo-column called “update count.”

Updating and deleting data work just like inserting data. You can simply enter the `UPDATE` or `DELETE` statement and press “Execute.”

Querying data

You can also use the Database Manager to query data in your database's tables. This is useful primarily for testing purposes. For example, if you're trying to track down a problem with one of your JSP pages, you might want to look directly at the database's data to make sure that that's not the source of the problem. You can inspect hsqldb data simply by entering a SQL `SELECT` statement into the upper-right window in Database Manager. See figure 4 for an example of how data is displayed.

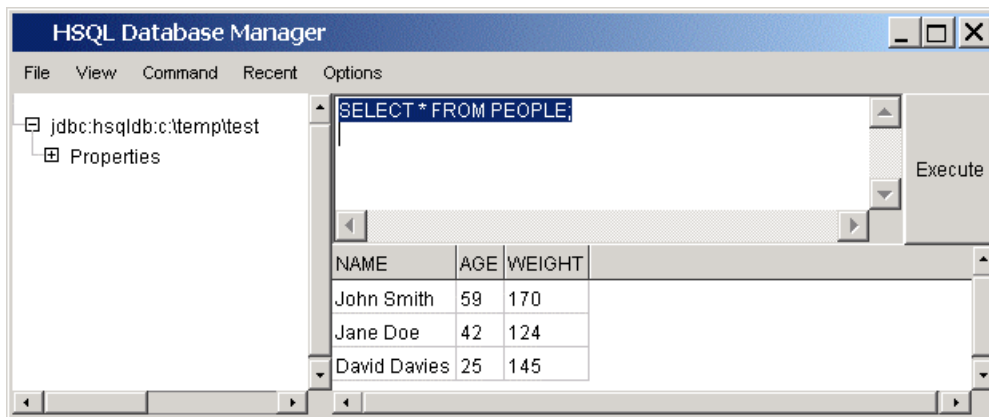


Figure 4 A sample query and response in the Database Manager.

Using an hsqldb database in your application

Database Manager is just an administrative tool. You use it when you want to intervene personally in your hsqldb database. Separately, JSTL's database tags can access your hsqldb database from within your web application. To make sure things work the way you want them to, you'll need to keep a few things in mind.

Different databases, different URLs

Even in the simple mode we've described here, hsqldb can support an essentially unlimited number of databases; that is, hsqldb can set up numerous, isolated collections of tables that have nothing to do with each other. This is useful primarily when you want to use hsqldb on the same machine but for different, perhaps unrelated applications.

As we saw earlier, when you start Database Manager, you need to enter a URL that describes the database you'll be working with. When using tags like `<sql:driver>`, which we discuss in chapter 9 of *JSTL in Action*, you will need to enter the URL for the database that you want to expose to your JSP pages. This URL needs to match the one you entered into Database Manager.

For the simple use of hsqldb described here, databases are local to the specific machine on which you create them. That is, for all practical purposes, a "standalone" hsqldb database is accessible only from the machine where it was created. Normally, databases are accessible over the network, but again, our goal is experimentation and learning, not deployment. Deploying database applications from scratch is an advanced topic not covered by this book.

Simultaneous access is forbidden

One of the limitations of the simple hsqldb mode we discuss here is that it doesn't support simultaneous access to the same database from different applications. For example, you can't access the same database URL from two different programs at once. This is not only a problem for production environments; it presents inconveniences for simple experimental applications as well. For instance, you must exit the Database Manager (which you can do simply by closing its window) before starting up your JSP container and using JSTL tags to access it. Database Manager should not run at the same time as any web applications that access it.

Tip: There's no problem accessing *different* databases simultaneously. This is one reason you might want to create multiple databases. That way, you'll have no problem testing two web

applications at the same time. You can just use one database URL from one application and a different one from another.

One example of an acceptable order of operations is as follows:

1. Run Database Manager and create tables in a new database (that is, a new URL).
2. Close Database Manager.
3. Start Tomcat and run pages that use JSTL tags.

Then, if you need to run Database Manager again, you would go through the following steps:

1. Stop Tomcat.
2. Run Database Manager again.
3. Close Database Manager.
4. Start Tomcat again.

If this procedure seems too cumbersome, you have two choices. First, you can use `hsqldb` in a more complicated mode, called *client/server*. This mode is trickier to run but does allow for simultaneous access. `Hsqldb`'s documentation describes the procedure for running client/server mode.

Your other choice is to avoid using Database Manager altogether. One somewhat inelegant but perhaps useful trick is to simply create your tables from within a JSP page that uses the `<sql:update>` JSTL tag. Nothing stops you from sending a `CREATE TABLE` statement using this tag; for instance, the following is perfectly valid:

```
<sql:update>
CREATE TABLE PEOPLE (
  NAME VARCHAR(255) ,
  AGE INTEGER,
  WEIGHT INTEGER
)
</sql:update>
```

When this tag runs, it attempts to create a new `PEOPLE` table. You probably wouldn't want to make pages containing tags like this public, but you shouldn't have any problem if you use such pages experimentally. Chances are you won't be rolling out a public application that uses `hsqldb`'s "standalone," non-simultaneous mode anyway; as I said before, it's useful primarily for experimentation. primarily for experimentation.