

Database Connectivity with JDBC

Advanced Topics in Java

Khalid Azim Mughal
khalid@ii.uib.no
<http://www.ii.uib.no/~khalid/atij/>

Version date: 2006-09-04

Overview

- Using the DriverManager to obtain database connections.
- Using JNDI to obtain a DataSource for database connections.

Practical Considerations

- The database *postgreSQL* (<http://www.postgresql.org/>) is installed.
- The web server/container *Tomcat* is installed.
- The postgres driver `postgresql-8.0-312.jdbc3.jar` is in the `%TOMCAT_HOME%/common/lib` directory.

Using the DriverManager (java.sql package)

- Example: the *horoscope* application
 - Files: `HoroscopeUsingDriverManager.java`, `HoroscopeUsingDriverManager.html` and `web.xml`
- 1. Loading the postgresSQL driver in the `init()` method of the servlet.

```
public void init() {
    ...
    try {
        // Load the driver
        String driver = getInitParameter("driver");
        Class.forName(driver).newInstance();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (InstantiationException e) {
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    }
}
```

2. Making the connection.

```
/** Making the connection using the DriverManager */
protected Connection getConnection() throws SQLException {
    String db_url = getInitParameter("db_url");
    String user   = getInitParameter("user");
    String pw    = getInitParameter("pw");
    Connection conn = DriverManager.getConnection(db_url, user, pw);
    return conn;
}
```

3. Executing queries.

```
protected String getHoroscope(String sign) {
    String horoStr = null;
    try {
        // Make the connection
        Connection conn = getConnection();

        // Create a query
        Statement stmt = conn.createStatement();
        // Executing the query
        String tableName = getInitParameter("db_table");
        ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName
            + " WHERE SIGN='" + sign + "'");
        // Handle the result set.
        if (rs.next())
            horoStr = rs.getString(2);
            System.out.println(horoStr);
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return horoStr;
}
```

4. Servlet specification in the application deployment descriptor file web.xml.

```
...
<servlet>
  <servlet-name>HoroscopeUsingDriverManager</servlet-name>
  <servlet-class>HoroServletUsingDriverManager</servlet-class>
  <init-param>
    <param-name>driver</param-name>
    <param-value>org.postgresql.Driver</param-value>
  </init-param>
  <init-param>
    <param-name>db_url</param-name>
    <param-value>jdbc:postgresql://localhost/horodb</param-value>
  </init-param>
  <init-param>
    <param-name>user</param-name>
    <param-value>javauser</param-value>
  </init-param>
  <init-param>
    <param-name>pw</param-name>s
    <param-value>javadude</param-value>
  </init-param>
  <init-param>
    <param-name>db_table</param-name>
```

```
    <param-value>horoTable</param-value>
  </init-param>
</servlet>
<!-- Define the servlet mapping -->
...
```

Running the HoroSrvletUsingDriverManager servlet

The screenshot shows two browser windows. The left window displays the main page titled "Get Your Daily Horoscope (using DriverManager)". It includes a "Read the stars!" button and a list of zodiac signs. The right window displays the "Simple Horoscope IV (Using DriverManager)" page. The "Initialization Parameters" section is circled in red and contains the following code:

```
user: javauser
db_url: jdbc:mysql://localhost:3306/horodb
db_table: horoTable
pwd: javadude
driver: com.mysql.jdbc.Driver
```

Using the DataSource (javax.sql package) through JNDI

- Optimization of opening and closing database connections can be crucial in an application.
- A DataSource is a factory for database connections, providing *connection pools*.
- A *connection pool* is a set of open connections that are maintained by a DataSource.
- A web application uses a DataSource to obtain a connection to a database.

JNDI: Java Naming and Directory Interface

- An application server/container like Tomcat can provide access to shared resources (like a connection pool) through JNDI for applications running under it.
- JNDI stores Java objects (i.e. resources) in a hierarchy that can be navigated and searched for registered objects.
- All resources are placed in the `java:comp/env` portion of the JNDI namespace.

Configuring Tomcat to use a DataSource

- The database resource is configured outside of the web applications.
- Example: the *horoscope* application
 - Files: HoroscopeUsingDB.java, HoroscopeUsingDB.html, web.xml and horoscope.xml.
- 1. Register the data base resource with the application server, and thus in the JNDI service.

A JNDI object will represent the database resource that can be used by the web applications.

Three ways of registering such a resource, using the <Resource> subelement under the <Context> element:

- In the server.xml file of the server
 - In the web.xml file of the web application
 - In a context file under the directory %TOMCAT_HOME%/conf/Catalina/localhost.
2. Specifying by a <resource-ref> element in the web.xml file how a web application can access the database resource.

Registering the Database Resource in JNDI

- Configuring the resource in the context file horoscope.xml under the directory %TOMCAT_HOME%/conf/Catalina/localhost.

```
<?xml version='1.0' encoding='utf-8'?>
```

```
<Context path="/horoscope" ... >
```

```
...
```

```
<!-- The Resource element must specify the following: -->
```

```
<!-- The name of the resource to be created,  
relative to the java:comp/env context.  
Name must be same as in the web.xml -->
```

```
<!-- Who does the authentication?
```

```
The value of this attribute must be Application or Container. -->
```

```
<!-- The fully qualified Java class name expected by the web application  
when it performs a lookup for this resource. -->
```

```
<!-- Class name for the official postgresSQL driver -->
```

```
<!-- The JDBC connection url for connecting to the postgresSQL dB. -->
```

```
<!-- The postgresSQL dB username and password for dB connections -->
```

```
<!-- Maximum number of dB connections in pool. -->
```

```
<!-- Maximum number of idle dB connections to retain in pool. -->
```

```
<!-- Maximum time to wait for a dB connection to become available  
in ms. Set to -1 to wait indefinitely. -->
```

```

<Resource
  name="jdbc/TestHoro"
  auth="Container"
  type="javax.sql.DataSource"
  driverClassName="org.postgresql.Driver"
  url="jdbc:postgresql://localhost/horodb"
  username="javauser"
  password="javadude"
  maxActive="20"
  maxIdle="10"
  maxWait="-1" />
</Context>

```

Associating the web application with the Database Resource

- In the web.xml file, the <resource-ref> element is specified for this purpose.

```

<web-app ...>
  ...
  <resource-ref>
    <description>HoroDB Connection</description> <!-- Optional description -->
    <res-ref-name>jdbc/TestHoro</res-ref-name> <!-- Unique path for the resource -->
    <res-type>javax.sql.DataSource</res-type> <!-- Fully-qualified class name
    for the resource -->
    <res-auth>Container</res-auth> <!-- The container takes care of
    authentication -->
  </resource-ref>
</web-app>

```

Using a DataSource in a Servlet

- See source file `HoroServletUsingDB.java`.

```
...
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;           // NB! Using the javax.sql package

public class HoroServletUsingDB extends SimpleHoroServlet {

    DataSource pool; // Pool of database connections

    /** Retrieve the DataSource */
    public void init() {

        servletName = "Simple Horoscope V (Using DataSource)";
        try {
            // Obtain the context. This is the same for all servlets.
            Context ctx = new InitialContext();

            // Look up the required data source.
            String resourceRefName = getInitParameter("resourceRefName"); // "jdbc/TestHoro"
            pool = (DataSource) ctx.lookup("java:comp/env/" + resourceRefName);
        }
    }
}
```

```
    } catch (NamingException e) {
        e.printStackTrace();
    }
}

protected String getHoroscope(String sign) {
    String horoStr = null;
    try {
        // Make the connection
        Connection conn = pool.getConnection();
        ...
        return horoStr;
    }
}
```

- The servlet can specify the DataSource reference name (i.e path) as an `init-param` in the `web.xml` file.

```

<web-app ...>
  ...
  <servlet>
    <servlet-name>HoroscopeUsingDatabase</servlet-name>
    <servlet-class>HoroServletUsingDB</servlet-class>
    <init-param>
      <param-name>resourceRefName</param-name>
      <param-value>jdbc/TestHoro</param-value>
    </init-param>
    <init-param>
      <param-name>db_table</param-name>
      <param-value>horoTable</param-value>
    </init-param>
  </servlet>
</web-app>

```

Running the HoroServletUsingDB servlet

The screenshot shows a web browser window with the URL `http://localhost:8080/horoscope/HoroscopeUsingDatabase.html`. The page content includes:

- Get Your Daily Horoscope (using DataSource)**
- Learn about all the clamaties th
- Of course, it's all FREE! Includ
- Choose your sign:
- Radio buttons for zodiac signs: Aquarius (Jan 21 - Feb 19), Pisces (Feb 20 - Mar 20), Aries (Mar 21 - Apr 20), Taurus (Apr 21 - May 21), Gemini (May 22 - Jun 21), Cancer (Jun 22 - Jul 23), Leo (Jul 24 - Aug 23) (selected), Virgo (Aug 24 - Sept 23), Libra (Sept 24 - Oct 23), Scorpio (Oct 24 - Nov 22), Sagittarius (Nov 23 - Dec 21), Capricorn (Dec 22 - Jan 20).
- Read the stars!

An inset window shows the URL `http://localhost:8080/horoscope/HoroscopeUsingDatabase - Microsoft Internet Explo...` and displays the following parameters:

- Request Parameters**: sign: leo
- Initialization Parameters**: db_table: horoTable, resourceRefName: jdbc/TestHoro (circled in red)
- Your horoscope, LEO:** Telling secrets will bring about an unexpected turn of events. You can trust someone who is making you an offer that is too good to be true. Do not rely on yourself and your own abilities.