CÓMO CONFIGURAR UN POOL DE CONEXIONES EN APACHE TOMCAT Y CONFIGURAR RECURSO JNDI

Vamos aprender ***como configurar un Pool de conexiones en Apache Tomcat*** y hacer un test de conexión desde una aplicación en Java, utilizando una configuración JNDI.

Hasta ahora hemos realizado algunas prácticas de como conectarse a MySQL, como por ejemplo: Cómo conectarse a MySQL desde Java, Operaciones CRUD utilizando el patrón DAO utilizando la clase *DriverManager.getConnection()*, y esto funciona correctamente, pero no es recomendable utilizarla en entornos de producción donde el número de usuarios es alto. Ya que el elevado consumo de memoria que se da a la hora de crear y cerrar conexiones limita el rendimiento del servidor.

QUE ES UN POOL DE CONEXIONES?

Un pool de conexiones, son un ***conjunto de conexiones*** a la base de datos que se encuentran listas para ser usadas y que son administradas por el servidor de aplicaciones.

CÓMO FUNCIONA?

Básicamente lo que se consigue con esto es la ***reutilización de conexiones existentes*** y consiste en que el servidor de aplicaciones se encarga de ***gestionar todas las conexiones a la base de datos***, verifica cuales están abiertas, cuales están inactivas (aquellas que ya no se están utilizando) y de acuerdo a esto asigna esas conexiones a nuevos usuarios que se quieren conectar a la base de datos, evitando cerrar y crear una nueva conexión, ya que esto supone mayor consumo tanto de memoria como de procesamiento en el servidor.

CÓMO CONFIGURAR UN POOL DE CONEXIONES EN APACHE TOMCAT

Para ***configurar un Pool de conexiones en Apache Tomcat*** lo primero que vamos hacer es editar el archivo context.xml, este archivo se encuentra dentro de Eclipse, junto con las carpetas de los proyectos, dentro de la carpeta Servers.

El fragmento de código que debes añadir dentro del archivo context.xml es el siguiente, a propósito a esta configuración se la conoce también como fuente de datos, entre algunos parámetros define el número de máximo de conexiones,

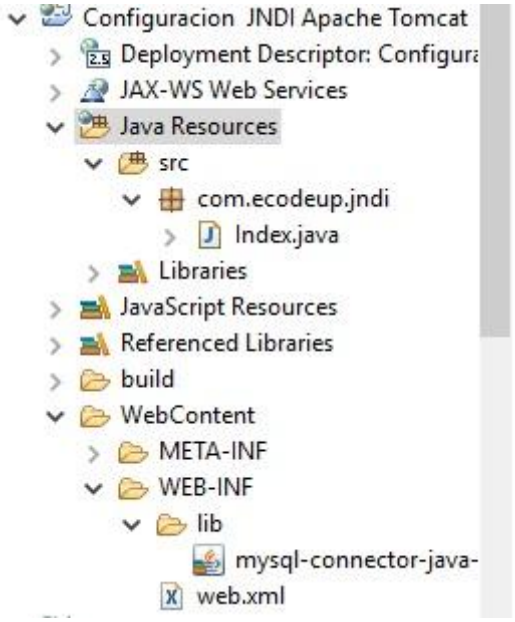
tiempo en el que una conexión se debe mantener inactiva, usuario de la base de datos, clave, puerto, servidor, la lista de parámetros completa la encuentras en el sitio oficial de Apache BasicDataSource Configuration Parameters.

```
1 <Resource name="jdbc/ecodeup" auth="Container" type="javax.sql.DataSource"
2                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
3                password="root" driverClassName="com.mysql.jdbc.Driver"
4                url="jdbc:mysql://localhost:3306/escuela" />
```

Las que vas a modificar son las siguientes:

- ***name***: Es el nombre con el que se identifica el recurso, cuando utilices la conexión dentro del código.
- ***username***: Debe ir el nombre del usuario de la base de datos.
- ***password***: Password correspondiente al usuario de la base de datos.
- ***url***: Define el servidor, el puerto y la base de datos a la que se conectará, para el ejemplo se llama escuela, este parámetro debe ser modificado de acuerdo al nombre de tu base de datos, para este ejemplo el nombre de mi base de datos se llama***escuela***.

ESTRUCTURA DEL PROYECTO



Recuerda que la versión del **Dynamic Web Project** debe ser 2.5, además que debes ***añadir el driver de conexión***

para MySQL puesto que vamos hacer una consulta a la base de datos.

Luego dentro de tu proyecto en el archivo web.xml debes añadir lo siguiente:

```
<resource-ref>
                    <description>DB Connection</description>
                    <res-ref-name>jdbc/ecodeup</res-ref-name>
                    <res-type>javax.sql.DataSource</res-type>
                    <res-auth>Container</res-auth>
          </resource-ref>
```

En el parámetro ***<description>*** puede ser cualquier nombre, en el parámetro ***<res-ref-name>***debe ir el mismo nombre declarado en el parámetro ***name*** del archivo context.xml.

IMPLEMENTADO UNA CONSULTA DE DATOS A TRAVÉS DE JNDI

Este es el Script que debes ejecutar para la creación de la base de datos y la tabla de la cual vamos hacer una consulta.

```
CREATE DATABASE  IF NOT EXISTS `escuela` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `escuela`;
-- MySQL dump 10.13  Distrib 5.7.12, for Win64 (x86_64)
--
-- Host: 127.0.0.1    Database: escuela
-- ------------------------------------------------------
-- Server version   5.7.16-log

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

```

```
20 --
21 -- Table structure for table `alumnos`
22 --
23
24 DROP TABLE IF EXISTS `alumnos`;
25 /*!40101 SET @saved_cs_client     = @@character_set_client */;
26 /*!40101 SET character_set_client = utf8 */;
27 CREATE TABLE `alumnos` (
28   `idAlumnos` int(11) NOT NULL,
29   `cedula` varchar(11) DEFAULT NULL,
30   `Nombres` varchar(45) DEFAULT NULL,
31   `Apellidos` varchar(45) DEFAULT NULL,
32   `Curso` varchar(45) DEFAULT NULL,
33   PRIMARY KEY (`idAlumnos`)
34 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
35 /*!40101 SET character_set_client = @saved_cs_client */;
36
37 --
38 -- Dumping data for table `alumnos`
39 --
40
41 LOCK TABLES `alumnos` WRITE;
42 /*!40000 ALTER TABLE `alumnos` DISABLE KEYS */;
43 INSERT INTO `alumnos` VALUES (1,'1717213183','Javier Ignacio','Molina Cano','Java
44 8'),(2,'1717456218','Lillian Eugenia','Gómez Álvarez','Java 8'),(3,'1717328901','Sixto
45 Naranjoe','Marín','Java 8'),(4,'1717567128','Gerardo Emilio','Duque Gutiérrez','Java
46 8'),(5,'1717902145','Jhony Alberto','Sáenz Hurtado','Java 8'),(6,'1717678456','Germán Antonio','Lotero
47 Upegui','Java 8'),(7,'1102156732','Oscar Darío','Murillo González','Java 8'),(8,'1103421907','Augusto
48 Osorno','Palacio Martínez','PHP'),(9,'1717297015','César Oswaldo','Alzate Agudelo','Java
49 8'),(10,'1717912056','Gloria Amparo','González Castaño','PHP'),(11,'1717912058','Jorge León','Ruiz
50 Ruiz','Python'),(12,'1717912985','John Jairo','Duque García','Java Script'),(13,'1717913851','Julio
51 Cesar','González Castaño','C Sharp'),(14,'1717986531','Gloria Amparo','Rodas
52 Monsalve','Ruby'),(15,'1717975232','Gabriel Jaime','Jiménez Gómez','Java Script');
53 /*!40000 ALTER TABLE `alumnos` ENABLE KEYS */;
54 UNLOCK TABLES;
55 /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
56
    /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
    /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
    /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
    /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
    /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
    /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
    /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

    -- Dump completed on 2016-12-09 20:24:42
```

Primero declaras las siguientes variables, las mismas que permiten obtener la conexión.

```
1     private Connection con;
2           private DataSource ds;
```

Dentro del servlet añades el siguiente código, este código permite a través de lo que se llama JNDI

utilizar el Pool de Conexiones para realizar sentencias SQL sobre la base de datos:

```
1 try {
2                       InitialContext ctx = new InitialContext();
3                       Context env = (Context) ctx.lookup("java:comp/env");
4                       dsource = (DataSource) env.lookup("jdbc/ecodeup");
5               } catch (NamingException e) {
6                       e.getMessage();
7               }
```

CREANDO UNA CONSULTA A LA BASE DE DATOS

Antes que nada recordarte que antes de crear la consulta debiste ejecutar el script sql anterior, en cuánto a la consulta, es el mismo formato que se ha venido utilizando en otros tutoriales.

```
1  String sql = "SELECT * FROM ALUMNOS";
2                       try {
3                               con = ds.getConnection();
4                               PreparedStatement pstm = con.prepareStatement(sql);
5                               ResultSet rset = pstm.executeQuery();
6                               System.out.println("ID | CÉDULA | NOMBRES  | APELLIDOS | CURSO  ");
7                               while (rset.next()) {
8                                       System.out.println(rset.getInt(1)+" | "+rset.getString(2)+" |
9  "+rset.getString(3)+" | "+rset.getString(4)+" | "+rset.getString(5));
10                              }
11                      } catch (SQLException e) {
12                              e.printStackTrace();
                        }
```

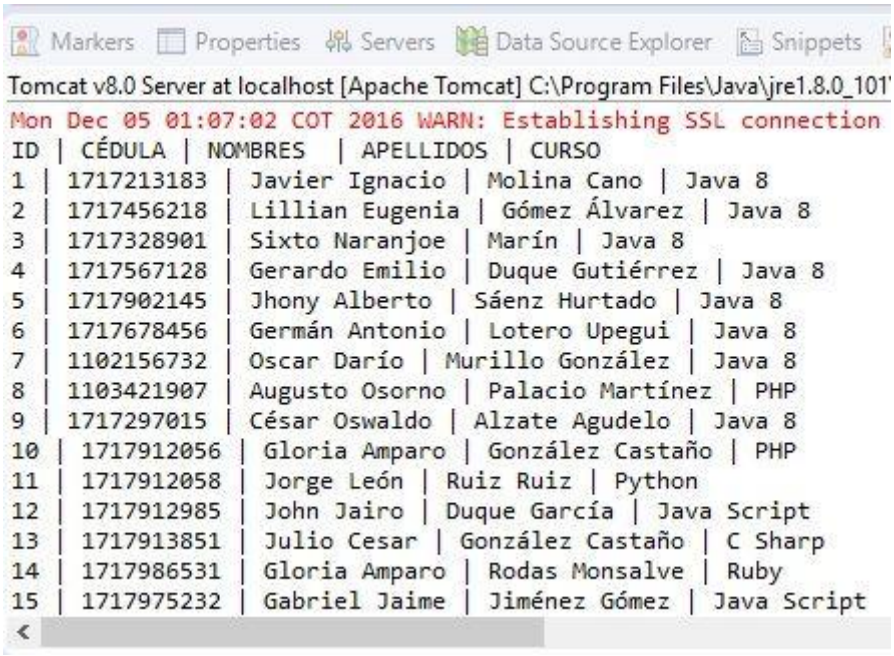
A continuación dejo el código completo del servlet.

```
1  package com.ecodeup.jndi;
2
3  import java.io.IOException;
4  import java.sql.Connection;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
```

```
import java.sql.SQLException;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/**
 * Servlet implementation class Index
 */
public class Index extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private Connection con;
	private DataSource ds;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public Index() {

	}

	@Override
	public void init(ServletConfig config) throws ServletException {
		super.init(config);
		try {
			InitialContext ctx = new InitialContext();
			Context env = (Context) ctx.lookup("java:comp/env");
			// nombre del recurso en el context.xml
			ds = (DataSource) env.lookup("jdbc/ecodeup");
		} catch (NamingException e) {
			e.printStackTrace();
		}
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		String sql = "SELECT * FROM ALUMNOS";
		try {
			con = ds.getConnection();
			PreparedStatement pstm = con.prepareStatement(sql);
			ResultSet rset = pstm.executeQuery();
			System.out.println("ID | CÉDULA | NOMBRES  | APELLIDOS | CURSO  ");
			while (rset.next()) {
				System.out.println(rset.getInt(1)+" | "+rset.getString(2)+" |
"+rset.getString(3)+" | "+rset.getString(4)+" | "+rset.getString(5));
			}
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
```

```
72          protected void doPost(HttpServletRequest request, HttpServletResponse response)
73                                  throws ServletException, IOException {
74                  // TODO Auto-generated method stub
75                  doGet(request, response);
76          }
77
    }
```

La presentación queda de la siguiente manera:



Si deseas profundizar en el tema puedes visitar el sitio oficial de Apache Tomcat donde encontrarás información más a detalle JNDI Datasource HOW-TO.