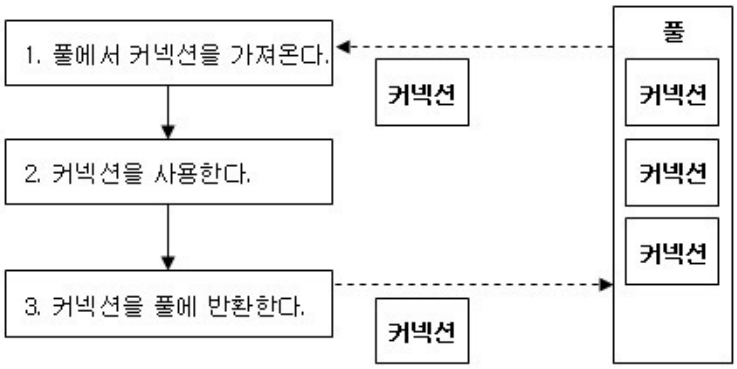
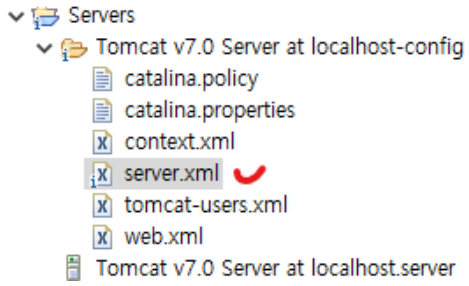
1. DBCP(Database Connection Pool) 개념도



2. 톰캣 설정: server.xml(or context.xml)



3. 선언문 작성: Context docBase

가. <Context docBase /></Host> -> <Context docBase ></Context></Host>

나. <Context docBase >

<**Resource** name="jdbc/myoracle(임의의이름)" auth="Container"
Type="javax.sql.Datasource"
driverClassName="oracle.jdbc.OracleDriver"(데이터베이스에 맞는 드라이버 이름)
url="jdbc:oracle:thin:@**localhost**:1521:xe"
username="kht123" password="kht123" maxActive="20" maxIdle="10"
maxWait="-1"/>

</Context></Host>

4. 커넥션 풀의 속성

속성	설명
maxActive	커넥션 풀이 제공할 최대 커넥션 개수
whenExhaustedAction	커넥션 풀에서 가져올 수 있는 커넥션이 없을 때 어떻게 동작할지를 지정. 0: 에러 발생 1: maxWait 속성에서 지정한 시간만큼 커넥션을 구할 때까지 대기 2: 일시적으로 커넥션을 생성해서 사용
maxWait	whenExhaustedAction 속성의 값이 1일 때 사용되는 대기 시간 단위는 1/1000초, 0보다 작을 경우 무한히 대기
maxIdle	사용되지 않고 풀에 저장될 수 있는 최대 커넥션 개수 음수일 경우 제한이 없음

5. web.xml 설정

가. server.xml에서 선언한 DataSource를 웹어플리케이션에서 사용하기 위해서는 web.xml에 해당 resource를 참조한다는 선언을 해야 함

나. <web-app> 태그안에 항목 추가

다. 추가 항목

```
<resource-ref>
   <res-ref-name>jdbc/myoracle</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
```

6. DBCP를 사용하는 예제

가. 유틸 예제

```
package util;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.sql.DataSource;

//DBCP: DataBase Connection Pool
public class DBManager {
    //db 연결
    public static Connection getConnection() {
            Connection conn=null;
            try {
                    //1. InitialContext 객체 생성
                    Context initContext=new InitialContext();
            //2. DataSource ds = (DataSource) initialContext.lookup('java:/comp/env/jdbc/myoracle');
                   Context  envContext=(Context)initContext.lookup("java:/comp/env");
                   DataSource ds=(DataSource)envContext.lookup("jdbc/myoracle");
            //3. lookup 메소드로 얻어낸 DataSource 객체로 getConnection()메소드를 호출하여 커넥션 객체를 얻어냄
                    conn = ds.getConnection();
            } catch (Exception e) {
                    e.printStackTrace();
            }
            return conn;
    }

            //select 수행 후 리소스 해제를 위한 메소드
    public static void close(Connection conn,Statement stmt, ResultSet rs) {
            try {
                    rs.close();
                    stmt.close();
                    conn.close();

            } catch (Exception e) {
            }
    }

            //insert, update, delete 작업을 수행한 후 리소스 해제를 위한 메소드
    public static void close(Connection conn, Statement stmt) {
            try {
                    stmt.close();
                    conn.close();
            } catch (Exception e) {
            }
    }

}
```

나. 활용 예제

```
public List<ProductEntity> selectAllProducts() {
    List<ProductEntity> list = new ArrayList<ProductEntity>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    conn = DBManager.getConnection();
    try {
            String sql = “”;
            pstmt = conn.preparedStatement(sql);
            Rs = psmt.executeQuery();
            while(rs.next()) {
                    ProductEntitiy vo = new ProductEntity();
                    …
                    list.add(vo);
            }
    } catch(Exception e) {
    } finally {
            DBManager.close(conn, psmt, rs);
    }
    return list;
}
```