[**DB연동: DBCP, DataSource, JNDI**](https://nastyle.tistory.com/15)

DB에 연결하는 방법으로 **JDBC**와 **DBCP**를 사용하는 방식이 있다.

DBCP는 JDBC의 약점을 보완한 방식으로 DastaSource & JNDI를 이용한다.

이에 JDBC & DBCP의 개념과 사용 방법에 대해 자세히 알아보겠습니다.

**1. JDBC(Java Data Base Connectivity)**

(To-Do)

**2. DBCP(Data Base Connection Pool)**

 2.1 DBCP를 사용하는 이유

  DB와 연결하여 작동하는 app는 매번 구동시 연결하여 시간 및 부하가 걸림

  ->이를 극복하기 위해 매번 접속하지 않고 하는 방법 DBCP(DataBase Connection Pool) 개발

 2.2 Connection Pool이란?

   client의 요청이 들어올 때마다 Connection을 생성하는 것이 아니라

   미리 생성하여 Pool에서 미리 생성된  Connetion을 관리하고 있다가 요청시 객체를 할당하고 작업 종료시 다시 Pool에서 관리한다.

 2.3

   DBCP사용을 위해서는 JNDI와 DataSource에 대한 설정이 필요하다

3.DataSource

3. JNDI란?

 DB의 정보는 Java code에 직접 하드코딩하면 불편한 점이 많다.(유지보수 어려움)

 즉, 계정정보와 같은 DB 정보가 바뀐다면 수정하고 재 컴파일하여 서버에 업로드해야 하기 떄문이다

 DB 정보를 외부(was)에서 관리하고 Source에서는 외부정보를 가져다쓸 수 있는 이름 값을 사용하면

 더욱 좋은 서비스가 그낭하다.

 이를 위한 방법이 JNDI이다.

 3.1 JNDI의 정의

 3.2

4. Process

 1)사용자가 요청을 보낸다.

 2)JNDI에 등록된 DB정보 객체(Type: DataSource)를 검색한다.

 3)JNDI를 통해 찾은 정보객체로 부터 커넥션을 획득한다.

 4)DB 조작 작업이 끝난 후 획득한 커넥션을 반환한다.

#############################################################################

\* Connection Pool의 생성배경

웹 어플리케이션 소스단에서 직접 DB를 연결하여 작동하는 어플리케이션의 경우

 -> 어플리케이션 구동시 DB와 커넥션을 맺는 부분에서 상당히 많은 시간이 걸리게 됩니다.

 -> 따라서 어플리케이션에서 DB에 접속하면 할수록, 쿼리를 사용하면 할수록, 어플리케이션은 급격히 느려지게 됩니다.

이러한 속도 저하를 막기 위해서 한번 DB와 맺은 Connection을 제거하지 않고 다른 쿼리들도 사용할 수 있도록 재활용 하도록 도와주는 라이브러리(DBCP)입니다

매번 접속마다 Connection을 열닫는 절차가 줄어듬으로써 속도가 빨라짐

\*Connection Pooling이란

\* WAS에 DataSource를 설정 하는 이유

그전에 WAS에 Datasource를 설정 하는 이유에 대해서 간략하게 말씀 드리겠습니다.

보통 이클립스에서 개발을 할때는 순수 JDBC 또는 Spring에서 제공하는 기본적인 JDBC(DBCP)설정을 사용하서 개발을 합니다.

하지만 웹 어플리케이션을 운영 서버로 deploy할 경우 JDBC 설정이 달라집니다.

그 이유는

첫째. 운영을 할 경우 개발은 웹 어플리케이션을 만드는 분(개발자)들이 하시지만 실제 서비스 운영(WAS Admin)은 님(개발자)들이 만드신 웹 어플리케이션을 잘 모르는 사람들이 운영을 하게 됩니다.

그렇기 때문에 아무래도 소스 레벨에서 설정되어 있는 것보다 WAS에 설정이 되어 있는 것을 선호 합니다.

둘째. 하나의 WAS에 하나의 웹 어플리케이션을 올리기도 하지만 대부분은 그렇지 않습니다.

즉 <WAS:웹 어플리케이션의 관계>가 <1:n 관계>가 많을 것입니다.

WAS가 관리하지 않고 n개의 어플리케이션들이 동일 db를 본다면, DB pool도 n개가 나오게 되겠죠.

그렇게 되면 관리도 어렵고, 자원 또한 낭비가 됩니다.

그렇기 떄문에, WAS에 설정되어 있는 것이 자원낭비를 줄일 수 있습니다.

셋째. 장애가 나거나 성능이 정상적이지 못할 때 다른 한 서버가 대신 일을 해주는 failover 기능입니다.

예를 들어서 DB1에 장애가 발생하면 백업 DB2로 로드 밸런싱을 해야 합니다.

이런 기능들은 대부분 WAS에 제공 합니다.

즉, WAS에서 설정을 하게 되면 다양한 이점이 있습니다.

즉 운영,관리,최적화,장애대처 등 다양한 이점이 있기 때문에 WAS에 DB Source를 설정 하는 것입니다.

모든 WAS에서도 마찬가지지만 DataSource를 설정하는 기술에는 자바 스펙인 Java Naming and Directory Interface (JNDI)를 이용 합니다.

JNDI는 쉽게 말해서 WAS에 공유 오브젝트를 저장 후에 쉽게 가져다가 쓰는 방식 입니다.

\*JDBC & DBCP

JDBC: Java Data Base Connectivity, DB와 연결하는 Java 인터페이스

      1)DriverManger를 통한 DB Connection 객체 생성

      2)Connection 객체에서 statement 객체 생성 및 쿼리 실행

      3)ResultSet 사용

      4)Coonection close

DBCP: DataBase Connection Pool, DB와 연결하고 있는 객체를 관리하기 위한 Connection Pool

      Connection pool에서 오픈된 Connection을 가지고 있다가 필요한 곳에 Connection을 할당 및 관리

\*DB와 연결하기 위해서는 JDBC가 필요하고 DBCP는 Connection Pool을 이용해 효율성을 향상

###################################3

<JNDI>

Java Naming & Directory Interface

네이밍 서비스는 분산된 환경에서 특정한 이름으로 객체를 등록하고 관리할 수 있는 방법을 의미한다.

객체를 네이밍 서비스에 등록, 삭제 검색할 수 있는 방법을 제공하는 것이 JNDI이다.

http://acet.pe.kr/214의 그림 사용

이름을 객체로 매핑하는 것을 바인딩이라 부르고 명명서비스는 프로그램 런타임시에 이름을 객체로 바인딩하거나 해제하는 방법을 제공한다

Context는 이름을 객체로 바인딩한 집합이다.

Context는 이름 규칙과 search 기능을 갖고 있다.

<JNDI와 JDBC의 차이>

JNDI는 DB Pool을 미리 Naming 시켜두는 방법, JDBC필

JDBC는 API 패키지를 이용하여  DB 연결

\*\*Naming Service 종류

1)DNS(Domain Name Service)

<사용법>

0. 라이브러리 추가

Commons-dbcp.jar

Commons-pool.jar

Commons-collections.jar 필요

tomcat 6.0 이후에는 tomcat-dbjp.jar 파일 하나로 통합됨

1. conf/server.xml 설정(WAS)

 -jdbc 셋

<GlobalNamingResources>

    <!-- UserMemoryRealm을 삭제하고 JNDI resouce를 넣어준다. -->

    <Resource name="jdbc/mariadb"    //네임스페이스에 등록될 객체의 이름

            auth="container"                              //자원관리자

            type="javax.sql.DataSource"                   //(반환 객체타입)

            driverClassName="org.mariadb.jdbc.Driver"     //DB드라이버 클래스

            loginTimeout="10"

            maxWait="5000"

            username="servlet"

            password="servlet"

            testOnBorrow="true"

            url="jdbc:mariadb://localhost:3306/servlet"   //데이터베이스의 위치정보

  />

</GlobalNamingResources>

2. conf/context.xml 설정(WAS)

 <Context> </Context> 사이에 추가

 <ResourceLink global="jdbc이름?" name="jdbc이름?" type="javax.sql.DataSource"/>

3. WEB-INF/web.xml 설정(Application)

<Web-app></Web-app> 사이에 추가

<resource-ref>

    <description>DB Connection</description>

    <res-ref-name>pbf\_web</res-ref-name>

    <res-type>javax.sql.DataSource</res-type>

    <res-auth>Container</res-auth>

</resource-ref>

4. private javax.sql.DataSource cp;

   try{

    javax.naming.Context ctx = new InitialContext();  //naming 조작을 위한 초기 context 반환

    cp = (DataSource) ctx.lookup("[JDNI-NAME]");      //등록된 이름의 자원 반환

   }catch(Exception e){

      e.printStackTrace();

   }

   Connection con = cp.getConnection();                 //커넥션 얻어옴;

   Context는 Naming Service를 이용하기 위한 바인딩 정보를 가지고 있는 객체이다.

   이 객체를 통해 이름으로 등록된 DBCP 객체를 가져 올 수 있다. Type: DataSource

출처: <https://nastyle.tistory.com/15> [오 좋은 정보인걸]