Connection Pooling?
  1)Database와 연결된 connection을 미리 일정 갯수만큼 생성하여 pool속에 저장해 놓고 필요할 때마다 이 pool에 접근하여 Connection 객체를 사용하고, 작업이 끝나면 다시 반환하는 것
  2)사용자가 connection이 필요할 때마다 Connection 객체를 생성하여 연결한다는 것은 매우 비효율적
  3)이 pool을 사용하면 pool 속에 미리 connection이 생성되어 있기 때문에 connection을 생성하는데 드는 시간이 소비되지 않는다.
  4)재 사용이 가능하기 때문에 사용자가 접속할 때마다 계속해서 connection을 생성할 필요가 없다.
  5)Program의 효율과 성능 개선의 효과

  6)JDBC 방식

```
    Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root",
    "1234");
```

  7)Connection Pool 방식

```
    Class.forName("org.apache.commons.dbcp.PoolingDriver");
    Connection conn = DriverManager.getConnection("jdbc:apache:commons:dbcp:/pool");
```

  8)Connection Pool의 대표적인 Open Source에는 DBCP와 C3P0가 있다.

DBCP2 2.6 API for JDBC 4.1 의 사용방법
-refer to : https://hsp1116.tistory.com/8
            https://sjh836.tistory.com/148

1. DBCP 관련 Jar file 및 JDBC driver Jar file 설치하기
  1)Homepage
    -DBCP : http://commons.apache.org/proper/commons-dbcp/
    -Pool : http://commons.apache.org/proper/commons-pool/
    -Logging : http://commons.apache.org/proper/commons-logging/

  2)Downloads
    -DBCP API 관련 jar file : commons-dbcp2-2.6.0-bin.zip or commons-dbcp2-2.6.0-bin.tar.gz
    -Pool API 의 jar file : commons-pool2-2.6.2-bin.zip or commons-pool2-2.6.2-bin.tar.gz
    -Logging API의 jar file : commons-logging-1.2.bin.zip or commons-logging-1.2-bin.tar.gz

  3)위의 file의 압축을 풀고 각각의 jar file을 WEB-INF/lib folder에 import 한다.
    -commons-dbcp2-2.6.0.jar, commons-pool2-2.6.2.jar, commons-logging-1.2.jar

2. Connection Pool 관련 설정 file 및 Connection Pool 관련 driver loading하기
  -src/com.example.utils.DBCPInit.java

```
    package com.example.utils;

    import java.sql.DriverManager;

    import javax.servlet.ServletException;
    import javax.servlet.http.HttpServlet;

    import org.apache.commons.dbcp2.ConnectionFactory;
    import org.apache.commons.dbcp2.DriverManagerConnectionFactory;
    import org.apache.commons.dbcp2.PoolableConnection;
    import org.apache.commons.dbcp2.PoolableConnectionFactory;
    import org.apache.commons.dbcp2.PoolingDriver;
```

```
import org.apache.commons.pool2.impl.GenericObjectPool;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;

public class DBCPInit extends HttpServlet {
  private final String driverClass = "oracle.jdbc.driver.OracleDriver";
  private final String url = "jdbc:oracle:thin:@localhost:1521:XE";
  private final String username = "hr";
  private final String password = "hr";

  @Override
  public void init() throws ServletException{
    loadJDBCDriver();
    initconnectionPool();
  }

  private void loadJDBCDriver() {
    try {
      //Connection Pool에서 사용할 JDBC Driver Loading
      Class.forName(this.driverClass);
    }catch(ClassNotFoundException ex) {
      throw new RuntimeException("Driver Loading Failure");
    }
  }

  private void initconnectionPool() {
    try {
      //ConnectionFactory 생성, ConnectionFactory는 새로운 Connection을 생성할 때 사용.
      ConnectionFactory connFactory = new DriverManagerConnectionFactory(this.url,
      this.username, this.password);

      //DBCP가 Connection Pool에 connection을 보관할 때 사용하는 PoolableConnectionFactory 생성
      //실제로 내부적으로 connection을 담고 있고, connection을 관리하는데 기능을 제공한다.
      //Connection을 close하면 종료하지 않고 Connection Pool에 반환한다.
      PoolableConnectionFactory poolableConnFactory = new
      PoolableConnectionFactory(connFactory, null);
      //Connection이 유효한지 확인할 때 사용하는 query를 설정한다.
      poolableConnFactory.setValidationQuery("select 1 from dual");

      //Connection Pool의 설정 정보를 생성한다.
      GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
      //유휴 connection 검사 주기
      poolConfig.setTimeBetweenEvictionRunsMillis(1000L * 60 * 1L);
      //Pool에 있는 connection이 유효한지 검사 유무 설정
      poolConfig.setTestWhileIdle(true);
      //Connection 최소 갯수 설정
      poolConfig.setMinIdle(4);
      //Connection 최대 갯수 설정
      poolConfig.setMaxTotal(50);

      //Connection Pool 생성, parameter는 위에서 생성한 PoolableConnectionFactory와
      GenericObjectPoolConfig를 사용
      GenericObjectPool<PoolableConnection> connectionPool = new
      GenericObjectPool<>(poolableConnFactory, poolConfig);

```

```
            //PoolableConnectionFactory에도 Connection Pool 연결
            poolableConnFactory.setPool(connectionPool);

            //Connection Pool을 제공하는 JDBC Driver 등록.
            Class.forName("org.apache.commons.dbcp2.PoolingDriver");

            PoolingDriver driver = (PoolingDriver) DriverManager.getDriver("jdbc:apache:commons:dbcp:");

            //위에서 Connection Pool Driver에 생성한 Connection Pool을 등록한다.
            //이름은 cp이다.
            driver.registerPool("cp", connectionPool);
          }catch(Exception ex) {
            throw new RuntimeException(ex);
          }
        }
      }
```

3. Web Application이 시작될 때 DBCPInit Servlet class가 시작될 수 있도록 지정하기
   -WEB-INF/web.xml

```
      <?xml version="1.0" encoding="UTF-8"?>
      ....
      ....
        <servlet>
          <servlet-name>DBCPInit</servlet-name>
          <servlet-class>com.example.utils.DBCPInit</servlet-class>
          <load-on-startup>1</load-on-startup>
        </servlet>
      </web-app>
```

4. Connection을 가져오는 class
   -Connection을 구하는 class는 별도의 DBConnection class를 작성하는것이 개발하는데 편리하다.
   -src/com.example.utils.DBConnection.java

```
      package com.example.utils;

      import java.sql.Connection;
      import java.sql.DriverManager;
      import java.sql.SQLException;

      public class DBConnection {
        public static Connection getConnection() throws SQLException {
          return DriverManager.getConnection("jdbc:apache:commons:dbcp:cp");
        }
      }
```

5. 사용방법
   -Connection을 구하는 곳에 다음과 같이 해주면 된다.

```
      try {
        conn = DBConnection.getConnection();

```

6. DBCP Configuration 정보 : https://commons.apache.org/proper/commons-dbcp/configuration.html
  -Refer to : https://sjh836.tistory.com/148

7. Lab
  1)Create Stored Procedure sp_select
```
    CREATE OR REPLACE PROCEDURE sp_select
    (
       v_deptno   IN    employees.department_id%TYPE,
       employee_records    OUT SYS_REFCURSOR
    )
    AS
    BEGIN
       OPEN employee_records FOR
       SELECT employee_id, first_name, salary,
             TO_CHAR(hire_date, 'YYYY-MM-DD') AS hiredate,
             department_name, city, e.department_id AS deptno
       FROM employees e INNER JOIN departments d ON e.department_id = d.department_id
                    INNER JOIN locations l ON d.location_id = l.location_id
       WHERE e.department_id = v_deptno;
    END;
    /
```

  2)Build Path에 oracle driver 추가하기
    -project > right-click > Build Path > Configure Build Path...
    -Libraries tab > Click [Add External JARs...]
    -Select ojdbc6.jar > Click [Apply and Close]

  3)WebContent/WEB-INF/lib에 jar file 추가
    -ojdbc6.jar
    -taglibs-standard-impl-1.2.5.jar
    -taglibs-standard-spec-1.2.5.jar
    -commons-dbcp2-2.6.0.jar
    -commons-logging-1.2.jar
    -commons-pool2-2.6.2.jar

  4)src/com.example.utils.DBCPInit.java
    -위 참조

  5)src/com.example.utils.DBConnection.java
    -위 참조

  6)src/com.example.vo.EmployeeVO.java
```
    package com.example.vo;

    public class EmployeeVO {
      private int employee_id;
      private String first_name;
      private double salary;
      private String hiredate;
      private String department_name;
      private String city;
```

```
        private int departno;

        public EmployeeVO() {}

        public int getEmployee_id() {
            return employee_id;
        }

        public void setEmployee_id(int employee_id) {
            this.employee_id = employee_id;
        }

        public String getFirst_name() {
            return first_name;
        }

        public void setFirst_name(String first_name) {
            this.first_name = first_name;
        }

        public double getSalary() {
            return salary;
        }

        public void setSalary(double salary) {
            this.salary = salary;
        }

        public String getHiredate() {
            return hiredate;
        }

        public void setHiredate(String hiredate) {
            this.hiredate = hiredate;
        }

        public String getDepartment_name() {
            return department_name;
        }

        public void setDepartment_name(String department_name) {
            this.department_name = department_name;
        }

        public String getCity() {
            return city;
        }

        public void setCity(String city) {
            this.city = city;
        }

        public int getDepartno() {
            return departno;
        }

        public void setDepartno(int departno) {
            this.departno = departno;
        }

        @Override
        public String toString() {
            return "EmployeeVO [employee_id=" + employee_id + ", first_name=" + first_name + ", salary=" + salary
                    + ", hiredate=" + hiredate + ", department_name=" + department_name + ", city=" + city
                    + ", department_id=" + departno + "]";
        }

    }

7)src/com.example.dao.EmployeeDao.java
    package com.example.dao;

    import java.sql.CallableStatement;
    import java.sql.Connection;
    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.util.ArrayList;

    import com.example.utils.DBConnection;
    import com.example.vo.EmployeeVO;

    public class EmployeeDao {
        public static ArrayList<EmployeeVO> selectAll(int deptno) throws SQLException{
            ArrayList<EmployeeVO> list = new ArrayList<EmployeeVO>();
            Connection conn = DBConnection.getConnection();
            CallableStatement cstmt = conn.prepareCall("{ call sp_select(?, ?) }");
            cstmt.setInt(1, deptno);
            cstmt.registerOutParameter(2, oracle.jdbc.OracleTypes.CURSOR);
            cstmt.executeUpdate();
            ResultSet rs = (ResultSet)cstmt.getObject(2);
            while(rs.next()) {
                EmployeeVO emp = new EmployeeVO();
                emp.setEmployee_id(rs.getInt("employee_id"));
                emp.setFirst_name(rs.getString("first_name"));
                emp.setSalary(rs.getDouble("salary"));
                emp.setHiredate(rs.getString("hiredate"));
                emp.setDepartment_name(rs.getString("department_name"));
                emp.setCity(rs.getString("city"));
                emp.setDepartno(rs.getInt("deptno"));
                list.add(emp);
            }
            if(rs != null) rs.close();
            if(cstmt != null) cstmt.close();
            return list;
        }
    }
```

8)src/com.example.service.EmployeeService.java
```
  package com.example.service;

  import java.sql.SQLException;
  import java.util.ArrayList;

  import com.example.dao.EmployeeDao;
  import com.example.vo.EmployeeVO;

  public class EmployeeService {
    private int deptno;
    private ArrayList<EmployeeVO> list;

    public void setDeptno(int deptno) {
      this.deptno = deptno;
    }

    public ArrayList<EmployeeVO> getList() {
      ArrayList<EmployeeVO> list = null;
      try {
        list = EmployeeDao.selectAll(this.deptno);
      }catch(SQLException ex) {
        System.out.println(ex);
      }
      return list;
    }
  }
```

9)WebContent/dbtest.jsp
```
  <%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
  <%@ page import="java.util.ArrayList, com.example.vo.EmployeeVO" %>
  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
  <jsp:useBean id="service" class="com.example.service.EmployeeService" />
  <c:set target="${service}" property="deptno" value="${empty param.deptno ? 10 :
  param.deptno}"/>
  <!DOCTYPE html>
  <html>
  <head>
  <meta charset="UTF-8">
  <title>사원명단</title>
  </head>
  <body>
    <h1>사원 명단(부서번호 : <c:out value="${empty param.deptno ? 10 : param.deptno}"
    />)</h1>
    <table border="1">
      <thead>
        <tr>
          <th>사원번호</th><th>사원이름</th><th>봉급</th><th>입사일자</th><th>부서이름
          </th><th>부서위치</th><th>부서번호</th>
        </tr>
      </thead>
      <tbody>
```

```
            <c:forEach items="${service.list}" var="emp">
            <tr>
              <td>${emp['employee_id']}</td><td>${emp['first_name']}</td><td>${emp.salary}</td>
              <td>${emp.hiredate}</td><td>${emp['department_name']}</td><td>${emp.city}</td>
              <td>${emp.departno}</td>
            </tr>
            </c:forEach>
          </tbody>
        </table>
      </body>
      </html>
```