# JSP Board Paging

JSP Board Paging

# Paging

## Paging Algorithm

자료의 수가 많거나 한 화면에
다 보여줄 수 없는 경우
페이지를 분할하여
해당 내용을 보여주는 기법으로

통상적으로 Query문을 통하여 구현하지만
전체의 내용을 JavaScript로 분할하거나
java Object를 이용하여 분할하기도 한다.

**페이징을 처리하는 방식은 매우 다양한다.**

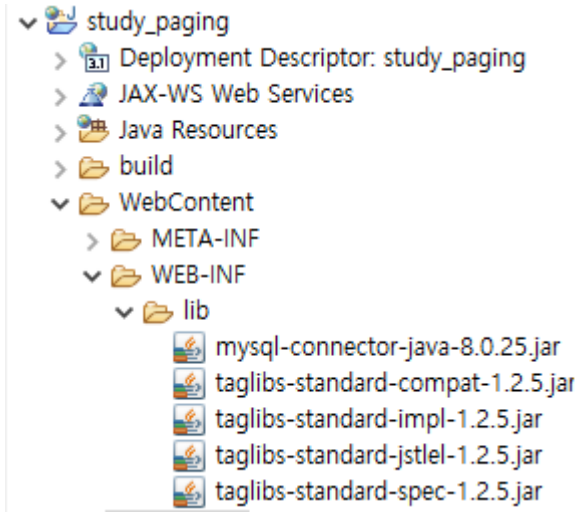
# 데이터베이스 입력

```
mysql> desc page;
+---------+------+------+-----+---------+----------------+
| Field   | Type | Null | Key | Default | Extra          |
+---------+------+------+-----+---------+----------------+
| num     | int  | NO   | PRI | NULL    | auto_increment |
| title   | text | YES  |     | NULL    |                |
| content | text | YES  |     | NULL    |                |
+---------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from page;
+-----+------------+-------------+
| num | title      | content     |
+-----+------------+-------------+
|   1 | 1번제목    | 1번 내용    |
|   2 | 2번제목    | 2번 내용    |
|   3 | 3번제목    | 3번 내용    |
|   4 | 4번제목    | 4번 내용    |
|   5 | 5번제목    | 5번 내용    |
|   6 | 6번제목    | 6번 내용    |
|   7 | 7번제목    | 7번 내용    |
|   8 | 8번제목    | 8번 내용    |
|   9 | 9번제목    | 9번 내용    |
|  10 | 10번제목   | 10번 내용   |
|  11 | 11번제목   | 11번 내용   |
|  12 | 12번제목   | 12번 내용   |
|  13 | 13번제목   | 13번 내용   |
|  14 | 14번제목   | 14번 내용   |
|  15 | 15번제목   | 15번 내용   |
|  16 | 16번제목   | 16번 내용   |
|  17 | 17번제목   | 17번 내용   |
|  18 | 18번제목   | 18번 내용   |
|  19 | 19번제목   | 19번 내용   |
|  20 | 20번제목   | 20번 내용   |
|  21 | 21번제목   | 21번 내용   |
|  22 | 22번제목   | 22번 내용   |
|  23 | 23번제목   | 23번 내용   |
|  24 | 24번제목   | 24번 내용   |
|  25 | 25번제목   | 25번 내용   |
|  26 | 26번제목   | 26번 내용   |
|  27 | 27번제목   | 27번 내용   |
|  28 | 28번제목   | 28번 내용   |
|  29 | 29번제목   | 29번 내용   |
|  30 | 30번제목   | 30번 내용   |
|  31 | 31번제목   | 31번 내용   |
|  32 | 32번제목   | 32번 내용   |
|  33 | 33번제목   | 33번 내용   |
|  34 | 34번제목   | 34번 내용   |
+-----+------------+-------------+
34 rows in set (0.00 sec)
```

**자료의 양이 많을 수록 테스트가 편리하다.**

# 개발 세팅




index.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>자료 목록</title>
    </head>
    <body>
        <form action="ReadPage" method="get">
            <input type="hidden" name="currentPage" value="1">
            한 페이지에 보여줄 수를 골라주세요
            <select name="recordsPerPage">
                <option value="5">5</option>
                <option value="10" selected>10</option>
                <option value="15">15</option>
            </select>
            <input type="submit" value="보기">
        </form>
    </body>
</html>
```

JDBC와 taglib들을 저장해준뒤 index.jsp를 작성

# Paging 처리의 개념

## SQL limit의 활용

**select * from table limit n1, n2;**
**해당 자료를 보여주되**
**n1번부터 시작하여 n2개를 보여준다.**

```
mysql> select * from page limit 0, 10;
+-----+------------+-------------+
| num | title      | content     |
+-----+------------+-------------+
|   1 | 1번제목    | 1번 내용    |
|   2 | 2번제목    | 2번 내용    |
|   3 | 3번제목    | 3번 내용    |
|   4 | 4번제목    | 4번 내용    |
|   5 | 5번제목    | 5번 내용    |
|   6 | 6번제목    | 6번 내용    |
|   7 | 7번제목    | 7번 내용    |
|   8 | 8번제목    | 8번 내용    |
|   9 | 9번제목    | 9번 내용    |
|  10 | 10번제목   | 10번 내용   |
+-----+------------+-------------+
10 rows in set (0.00 sec)
```

```
mysql> select * from page limit 10, 10;
+-----+------------+-------------+
| num | title      | content     |
+-----+------------+-------------+
|  11 | 11번제목   | 11번 내용   |
|  12 | 12번제목   | 12번 내용   |
|  13 | 13번제목   | 13번 내용   |
|  14 | 14번제목   | 14번 내용   |
|  15 | 15번제목   | 15번 내용   |
|  16 | 16번제목   | 16번 내용   |
|  17 | 17번제목   | 17번 내용   |
|  18 | 18번제목   | 18번 내용   |
|  19 | 19번제목   | 19번 내용   |
|  20 | 20번제목   | 20번 내용   |
+-----+------------+-------------+
10 rows in set (0.00 sec)
```

```
mysql> select * from page limit 20, 10;
+-----+------------+-------------+
| num | title      | content     |
+-----+------------+-------------+
|  21 | 21번제목   | 21번 내용   |
|  22 | 22번제목   | 22번 내용   |
|  23 | 23번제목   | 23번 내용   |
|  24 | 24번제목   | 24번 내용   |
|  25 | 25번제목   | 25번 내용   |
|  26 | 26번제목   | 26번 내용   |
|  27 | 27번제목   | 27번 내용   |
|  28 | 28번제목   | 28번 내용   |
|  29 | 29번제목   | 29번 내용   |
|  30 | 30번제목   | 30번 내용   |
+-----+------------+-------------+
10 rows in set (0.00 sec)
```

```
mysql> select * from page limit 30, 10;
+-----+------------+-------------+
| num | title      | content     |
+-----+------------+-------------+
|  31 | 31번제목   | 31번 내용   |
|  32 | 32번제목   | 32번 내용   |
|  33 | 33번제목   | 33번 내용   |
|  34 | 34번제목   | 34번 내용   |
+-----+------------+-------------+
4 rows in set (0.00 sec)
```

**n1과 n2값을 변경하여 페이징 구성이 가능**

# Paging 처리의 개념

## Current Page 변수

현재 확인중인 페이지를 나타낸다.
확인중인 페이지는
n1으로 부터 시작하여 n2만큼의 자료를 보여준다.

현재 페이지에 보여주는 자료의 수는 n2개이며
다음 페이지에서 시작되는 자료는 n1+n2+1부터이다.

P1 : 0~10 (n1=0, n2=10)
P2 : 11~20 (n1=10, n2=10)
P3 : 21~30 (n1=20, n2=10)
P4 : 31~34 (n1=30, n2=10)

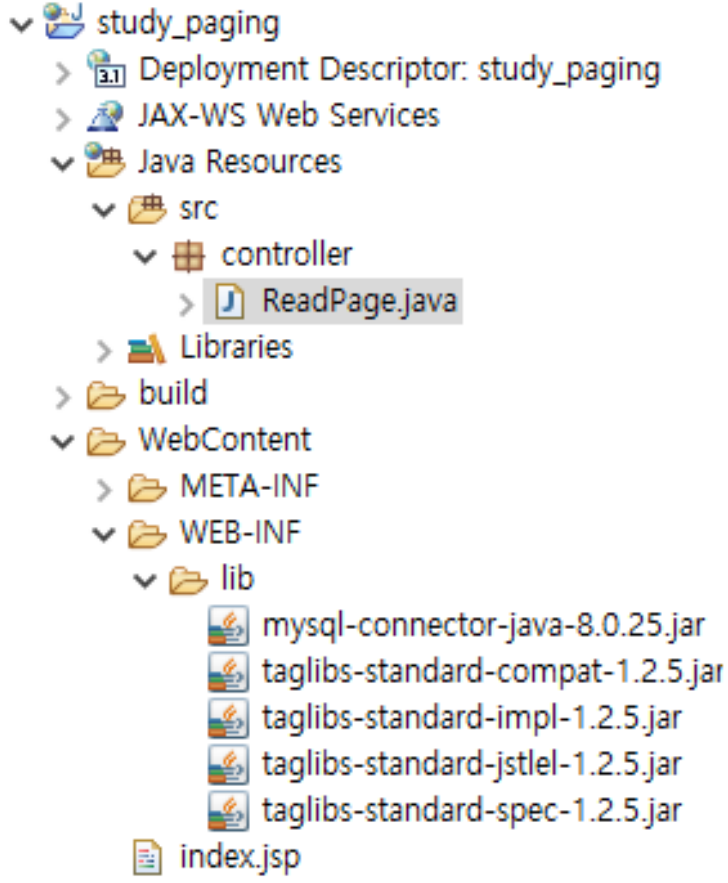
다음 페이지의 n1을 결정하는 것은
현재 페이지의 n1과 n20이다.

```
mysql> select * from page limit 10, 10
+-----+------------+--------------+
| num | title      | content      |
+-----+------------+--------------+
|  11 | 11번제목   | 11번 내용    |
|  12 | 12번제목   | 12번 내용용  |
|  13 | 13번제목   | 13번 내용용용|
|  14 | 14번제목   | 14번 내용용용|
|  15 | 15번제목   | 15번 내용용용|
|  16 | 16번제목   | 16번 내용용용|
|  17 | 17번제목   | 17번 내용용용|
|  18 | 18번제목   | 18번 내용용용|
|  19 | 19번제목   | 19번 내용용  |
|  20 | 20번제목   | 20번 내용    |
+-----+------------+--------------+
10 rows in set (0.00 sec)
```

```
mysql> select * from page limit 20, 5;
+-----+------------+--------------+
| num | title      | content      |
+-----+------------+--------------+
|  21 | 21번제목   | 21번 내용    |
|  22 | 22번제목   | 22번 내용용  |
|  23 | 23번제목   | 23번 내용용용|
|  24 | 24번제목   | 24번 내용용용|
|  25 | 25번제목   | 25번 내용    |
+-----+------------+--------------+
5 rows in set (0.00 sec)
```

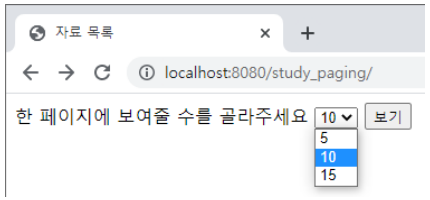
**auto_increment의 index와 limit의 index가 다름**

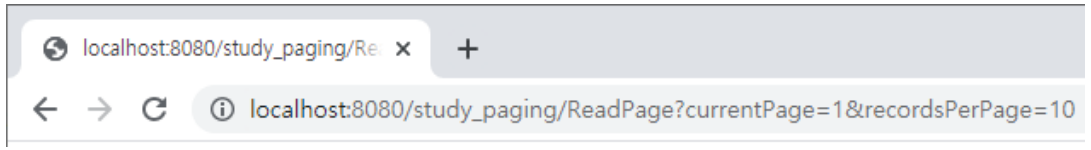
# 서블릿 제작

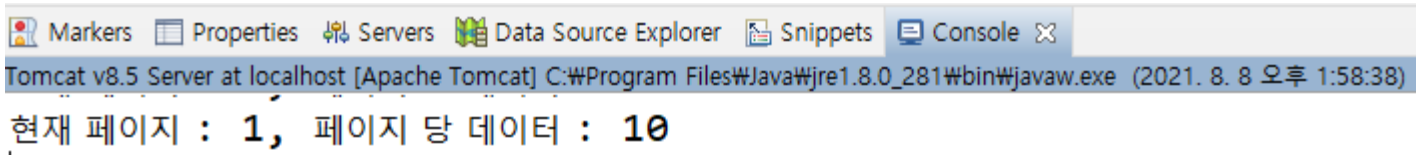



ReadPage.java

```java
package controller;

import java.io.IOException;

/**
 * Servlet implementation class ReadPage
 */
@WebServlet("/ReadPage")
public class ReadPage extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String currentPage=request.getParameter("currentPage");
        String recordsPerPage=request.getParameter("recordsPerPage");
        System.out.println("현재 페이지 : "+currentPage+", 페이지 당 데이터 : "+recordsPerPage);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

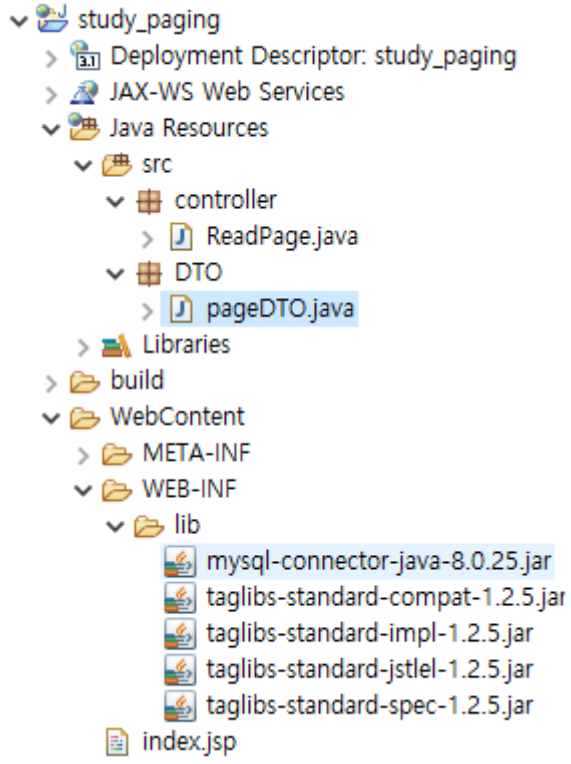



# get을 통하여 페이징을 구현할 패러미터 전달

# DTO 제작



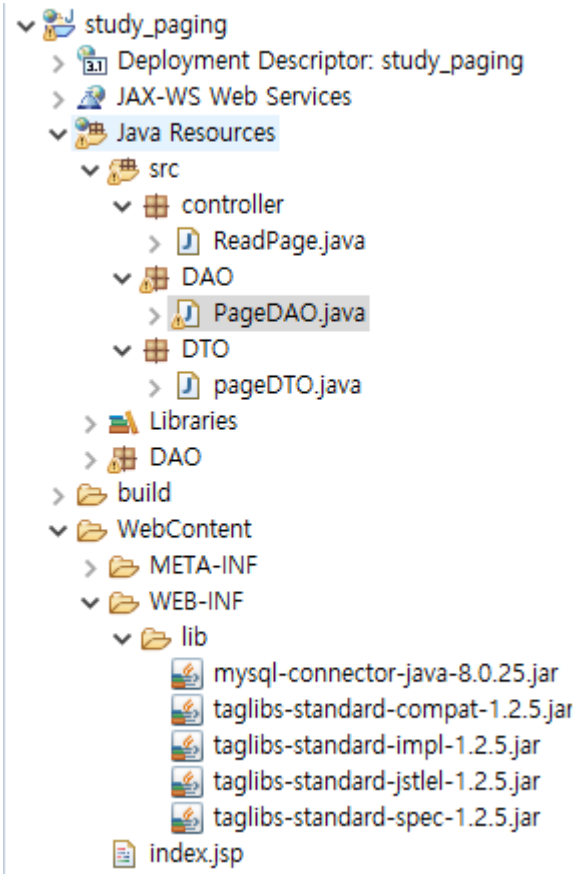

pageDTO.java

```java
package DTO;

public class pageDTO {
    private int num;
    private String title;
    private String content;
    public int getNum() {
        return num;
    }
    public void setNum(int num) {
        this.num = num;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
}
```

```
mysql> desc page;
+---------+------+------+-----+---------+----------------+
| Field   | Type | Null | Key | Default | Extra          |
+---------+------+------+-----+---------+----------------+
| num     | int  | NO   | PRI | NULL    | auto_increment |
| title   | text | YES  |     | NULL    |                |
| content | text | YES  |     | NULL    |                |
+---------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

## 데이터를 보관하기 위한 클래스 제작

# DAO 제작




PageDAO.java

```java
package DAO;

import java.sql.*;
import java.util.*;

public class PageDAO {
    private PageDAO() {

    }

    private static PageDAO instance=new PageDAO();

    public static PageDAO getInstance() {
        return instance;
    }

    //커넥션 기능 처리
    public Connection getConnection() throws Exception{
        Connection conn=null;
        String url="jdbc:mysql://127.0.0.1:3306/test";
        String id="root";
        String pass="iotiot";

        Class.forName("com.mysql.cj.jdbc.Driver");
        conn=DriverManager.getConnection(url, id, pass);
        return conn;
    }
}
```

## 데이터 접근을 위한 DAO 제작

# 조회기능 제작

```
30      //리스트 출력을 위한 정보처리
31      public List<pageDTO> findList() {
32          List<pageDTO> List=new ArrayList<pageDTO>();
33          String sql="select * from page";
34          Connection conn=null;
35          PreparedStatement pstmt=null;
36          ResultSet rs=null;
37
38          try {
39              conn=getConnection();
40              pstmt=conn.prepareStatement(sql);
41              rs=pstmt.executeQuery();
42              while(rs.next()) {
43                  pageDTO p=new pageDTO();
44                  p.setNum(rs.getInt("num"));
45                  p.setTitle(rs.getString("title"));
46                  p.setContent(rs.getString("content"));
47                  List.add(p);
48              }
49          }catch(Exception e) {
50              System.out.println("접속중 오류발생 : "+e);
51          }finally {
52              try {
53                  if(rs!=null) {rs.close();}
54                  if(pstmt!=null) {pstmt.close();}
55                  if(conn!=null) {conn.close();}
56              }catch(Exception e){
57                  System.out.println("접속 종료 중 오류발생");
58              }
59          }
60          return List;
61      }
```

# DAO 내부에 제작

# servlet 처리

ReadPage.java

```java
package controller;

import java.io.IOException;
import java.util.List;

import DTO.pageDTO;
import DAO.PageDAO;

import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

@WebServlet("/ReadPage")
public class ReadPage extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String currentPage=request.getParameter("currentPage");
        String recordsPerPage=request.getParameter("recordsPerPage");
        System.out.println("현재 페이지 : "+currentPage+", 페이지 당 데이터 : "+recordsPerPage);

        PageDAO BDO=PageDAO.getInstance();

        List<pageDTO> data=BDO.findList();
        request.setAttribute("data", data);

        request.setAttribute("currentPage", currentPage);
        request.setAttribute("recordsPerPage", recordsPerPage);

        RequestDispatcher dispatcher=request.getRequestDispatcher("list.jsp");
        dispatcher.forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

## 만들어진 DAO의 doGet을 변경

# 리스트 제작

list.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>도시 리스트</title>
        <link rel="stylesheet" href="page.css"/>
    </head>
    <body>
        <h1>
            currentPage : ${currentPage}<br>
            recordsPerPage : ${recordsPerPage}<br>
        </h1>
        <table>
            <tr>
                <th>num</th>
                <th>name</th>
                <th>population</th>
            </tr>
            <c:forEach items="${data }" var="data">
                <tr>
                    <td>${data.getNum() }</td>
                    <td>${data.getTitle() }</td>
                    <td>${data.getContent() }</td>
                </tr>
            </c:forEach>
        </table>
    </body>
</html>
```

page.css

```css
@charset "utf-8";

ul li{
    display:inline-block;
}
```

currentPage : 1
recordsPerPage : 10

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |
| 6 | 6번제목 | 6번 내용 |
| 7 | 7번제목 | 7번 내용 |
| 8 | 8번제목 | 8번 내용 |
| 9 | 9번제목 | 9번 내용 |
| 10 | 10번제목 | 10번 내용 |
| 11 | 11번제목 | 11번 내용 |
| 12 | 12번제목 | 12번 내용 |
| 13 | 13번제목 | 13번 내용 |
| 14 | 14번제목 | 14번 내용 |
| 15 | 15번제목 | 15번 내용 |
| 16 | 16번제목 | 16번 내용 |
| 17 | 17번제목 | 17번 내용 |
| 18 | 18번제목 | 18번 내용 |
| 19 | 19번제목 | 19번 내용 |
| 20 | 20번제목 | 20번 내용 |
| 21 | 21번제목 | 21번 내용 |
| 22 | 22번제목 | 22번 내용 |
| 23 | 23번제목 | 23번 내용 |
| 24 | 24번제목 | 24번 내용 |
| 25 | 25번제목 | 25번 내용 |
| 26 | 26번제목 | 26번 내용 |
| 27 | 27번제목 | 27번 내용 |
| 28 | 28번제목 | 28번 내용 |
| 29 | 29번제목 | 29번 내용 |
| 30 | 30번제목 | 30번 내용 |
| 31 | 31번제목 | 31번 내용 |
| 32 | 32번제목 | 32번 내용 |
| 33 | 33번제목 | 33번 내용 |
| 34 | 34번제목 | 34번 내용 |

# 간략한 css까지 제작

# 페이지 알고리즘 제작

```java
    //리스트 출력을 위한 정보처리
    //현재 페이지가 몇 페이지인지, 한 페이지 당 보여줄 자료의 양을 매개변수로 전달
    public List<pageDTO> findList(int currentPage, int recordsPerPage) {
        List<pageDTO> List=new ArrayList<pageDTO>();
        int start=currentPage*recordsPerPage-recordsPerPage;
        //1페이지가 1~9
        //2페이지가 10~19
        //3페이지가 20~29
        //인 경우 현재 페이지 값(1)*한 페이지 당 자료 수(10)을 진행한 뒤 페이지 당 자료 수를 빼주면 시작값을 찾아낼 수 있다.
        String sql="select * from page limit ?, ?";//쿼리문 수정
        Connection conn=null;
        PreparedStatement pstmt=null;
        ResultSet rs=null;

        try {
            conn=getConnection();
            pstmt=conn.prepareStatement(sql);
            pstmt.setInt(1, start);
            pstmt.setInt(2, recordsPerPage);
            rs=pstmt.executeQuery();
            while(rs.next()) {
                pageDTO p=new pageDTO();
                p.setNum(rs.getInt("num"));
                p.setTitle(rs.getString("title"));
                p.setContent(rs.getString("content"));
                List.add(p);
            }
        }catch(Exception e) {
            System.out.println("접속중 오류발생 : "+e);
        }finally {
            try {
                if(rs!=null) {rs.close();}
                if(pstmt!=null) {pstmt.close();}
                if(conn!=null) {conn.close();}
            }catch(Exception e){
                System.out.println("접속 종료 중 오류발생");
            }
        }
        return List;
    }
```

# DAO.findList() 부분을 수정

# 페이지 알고리즘 제작

```java
import java.io.IOException;
import java.util.List;

import DTO.pageDTO;
import DAO.PageDAO;

import javax.servlet.*;
import javax.servlet.annotation.*;
import javax.servlet.http.*;

@WebServlet("/ReadPage")
public class ReadPage extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int currentPage=Integer.parseInt(request.getParameter("currentPage"));//계산이 가능하도록 int로 변경
        int recordsPerPage=Integer.parseInt(request.getParameter("recordsPerPage"));
        System.out.println("현재 페이지 : "+currentPage+", 페이지 당 데이터 : "+recordsPerPage);

        PageDAO BDO=PageDAO.getInstance();

        List<pageDTO> data=BDO.findList(currentPage, recordsPerPage);//메서드에게 해당 내용을 전달
        request.setAttribute("data", data);

        request.setAttribute("currentPage", currentPage);
        request.setAttribute("recordsPerPage", recordsPerPage);

        RequestDispatcher dispatcher=request.getRequestDispatcher("list.jsp");
        dispatcher.forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

**currentPage : 1**
**recordsPerPage : 10**

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |
| 6 | 6번제목 | 6번 내용 |
| 7 | 7번제목 | 7번 내용 |
| 8 | 8번제목 | 8번 내용 |
| 9 | 9번제목 | 9번 내용 |
| 10 | 10번제목 | 10번 내용 |

**currentPage : 1**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |

# ReadPage.doGet 부분을 수정

# 페이지 개수 처리

```java
    //전체 자료의 갯수를 가져오는 메서드
    public int getNumberOfRows() {
        String sql="select count(num) from page";
        int numOfRows=0;//몇개의 페이지가 존재하는지 확인하는 부분
        Connection conn=null;
        PreparedStatement pstmt=null;
        ResultSet rs=null;

        try {
            conn=getConnection();
            pstmt=conn.prepareStatement(sql);
            rs=pstmt.executeQuery();
            rs.next();
            numOfRows=Integer.parseInt(rs.getString(1));
        }catch(Exception e) {
            System.out.println("접속중 오류발생");
        }finally {
            try {
                if(rs!=null) {rs.close();}
                if(pstmt!=null) {pstmt.close();}
                if(conn!=null) {conn.close();}
            }catch(Exception e){
                System.out.println("접속 종료 중 오류발생");
            }
        }
        return numOfRows;
    }
```

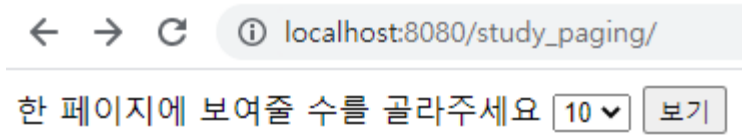
```java
@WebServlet("/ReadPage")
public class ReadPage extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int currentPage=Integer.parseInt(request.getParameter("currentPage"));
        int recordsPerPage=Integer.parseInt(request.getParameter("recordsPerPage"));
        System.out.println("현재 페이지 : "+currentPage+", 페이지 당 데이터 : "+recordsPerPage);

        PageDAO BDO=PageDAO.getInstance();

        List<pageDTO> data=BDO.findList(currentPage, recordsPerPage);
        request.setAttribute("data", data);
        int row=BDO.getNumberOfRows();       //페이지 수를 가져오는 메서드를 실행
        int nOfPages=row/recordsPerPage;     //전체자료수/페이지당자료수=페이지의 수
        //34/10인 경우 나머지가 4가 남는다, 이 경우 하나의 별도 페이지를 구성해야 하므로 1개의 페이지를 추가로 늘려준다.
        if(nOfPages%recordsPerPage>0) {
            nOfPages++;
        }
        request.setAttribute("nOfPages", nOfPages); //총 몇페이지인지 데이터를 request에 저장
        request.setAttribute("currentPage", currentPage);
        request.setAttribute("recordsPerPage", recordsPerPage);

        RequestDispatcher dispatcher=request.getRequestDispatcher("list.jsp");
        dispatcher.forward(request, response);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }
}
```

**전체 자료 수를 페이지 당 자료수로 나누어준다.**
**DAO 내부에 메서드를 작성한 뒤 servlet에서 처리한다.**

# 페이지 표시

list.jsp

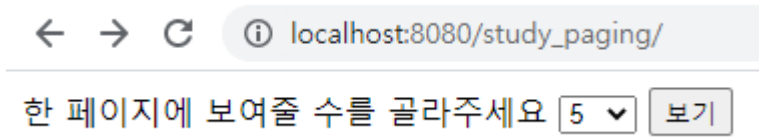
```jsp
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>도시 리스트</title>
        <link rel="stylesheet" href="page.css"/>
    </head>
    <body>
        <h1>
            nOfPages : ${nOfPages}</br>
            currentPage : ${currentPage}<br>
            recordsPerPage : ${recordsPerPage}<br>
        </h1>
        <table>
            <tr>
                <th>num</th>
                <th>name</th>
                <th>population</th>
            </tr>
            <c:forEach items="${data }" var="data">
                <tr>
                    <td>${data.getNum() }</td>
                    <td>${data.getTitle() }</td>
                    <td>${data.getContent() }</td>
                </tr>
            </c:forEach>
        </table>

        <ul>
            <c:forEach begin="1" end="${nOfPages}" var="i">
                <li>${i }</li>
            </c:forEach>
        </ul>
    </body>
</html>
```




**nOfPages : 4**
**currentPage : 1**
**recordsPerPage : 10**

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |
| 6 | 6번제목 | 6번 내용 |
| 7 | 7번제목 | 7번 내용 |
| 8 | 8번제목 | 8번 내용 |
| 9 | 9번제목 | 9번 내용 |
| 10 | 10번제목 | 10번 내용 |

1 2 3 4




**nOfPages : 7**
**currentPage : 1**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |

1 2 3 4 5 6 7

# list.jsp에서 데이터를 통해 페이지를 표시

# 페이지 링크처리

```
<ul>
    <c:forEach begin="1" end="${nOfPages}" var="i">
        <c:choose>
            <c:when test="${currentPage eq i }">
                <li><a>${i }(현재)</a></li>
            </c:when>
            <c:otherwise>
                <li><a href="ReadPage?recordsPerPage=${recordsPerPage }&currentPage=${i}">${i }</a></li>
            </c:otherwise>
        </c:choose>
    </c:forEach>
</ul>
```

**nOfPages : 7**
**currentPage : 1**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |

1(현재) 2 3 4 5 6 7

**nOfPages : 7**
**currentPage : 4**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 16 | 16번제목 | 16번 내용 |
| 17 | 17번제목 | 17번 내용 |
| 18 | 18번제목 | 18번 내용 |
| 19 | 19번제목 | 19번 내용 |
| 20 | 20번제목 | 20번 내용 |

1 2 3 4(현재) 5 6 7

**nOfPages : 7**
**currentPage : 7**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 31 | 31번제목 | 31번 내용 |
| 32 | 32번제목 | 32번 내용 |
| 33 | 33번제목 | 33번 내용 |
| 34 | 34번제목 | 34번 내용 |

1 2 3 4 5 6 7(현재)

# servlet에 currentPage값을 전달하여 이동

# 부가기능 처리

```
<ul>
    <c:if test="${currentPage !=1 }">
        <li><a href="ReadPage?recordsPerPage=${recordsPerPage }&currentPage=${currentPage-1}">이전페이지</a></li>
    </c:if>
    <c:forEach begin="1" end="${nOfPages}" var="i">
        <c:choose>
            <c:when test="${currentPage eq i }">
                <li><a>${i }(현재)</a></li>
            </c:when>
            <c:otherwise>
                <li><a href="ReadPage?recordsPerPage=${recordsPerPage }&currentPage=${i}">${i }</a></li>
            </c:otherwise>
        </c:choose>
    </c:forEach>
    <c:if test="${currentPage lt nOfPages }">
        <li><a href="ReadPage?recordsPerPage=${recordsPerPage }&currentPage=${currentPage+1}">다음페이지</a></li>
    </c:if>
</ul>
```

**nOfPages : 7**
**currentPage : 1**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 1 | 1번제목 | 1번 내용 |
| 2 | 2번제목 | 2번 내용 |
| 3 | 3번제목 | 3번 내용 |
| 4 | 4번제목 | 4번 내용 |
| 5 | 5번제목 | 5번 내용 |

1(현재) 2 3 4 5 6 7 다음페이지

**nOfPages : 7**
**currentPage : 4**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 16 | 16번제목 | 16번 내용 |
| 17 | 17번제목 | 17번 내용 |
| 18 | 18번제목 | 18번 내용 |
| 19 | 19번제목 | 19번 내용 |
| 20 | 20번제목 | 20번 내용 |

이전페이지 1 2 3 4(현재) 5 6 7 다음페이지

**nOfPages : 7**
**currentPage : 7**
**recordsPerPage : 5**

| num | name | population |
|---|---|---|
| 31 | 31번제목 | 31번 내용 |
| 32 | 32번제목 | 32번 내용 |
| 33 | 33번제목 | 33번 내용 |
| 34 | 34번제목 | 34번 내용 |

이전페이지 1 2 3 4 5 6 7(현재)

# 첫 페이지와 마지막인 경우는 없음