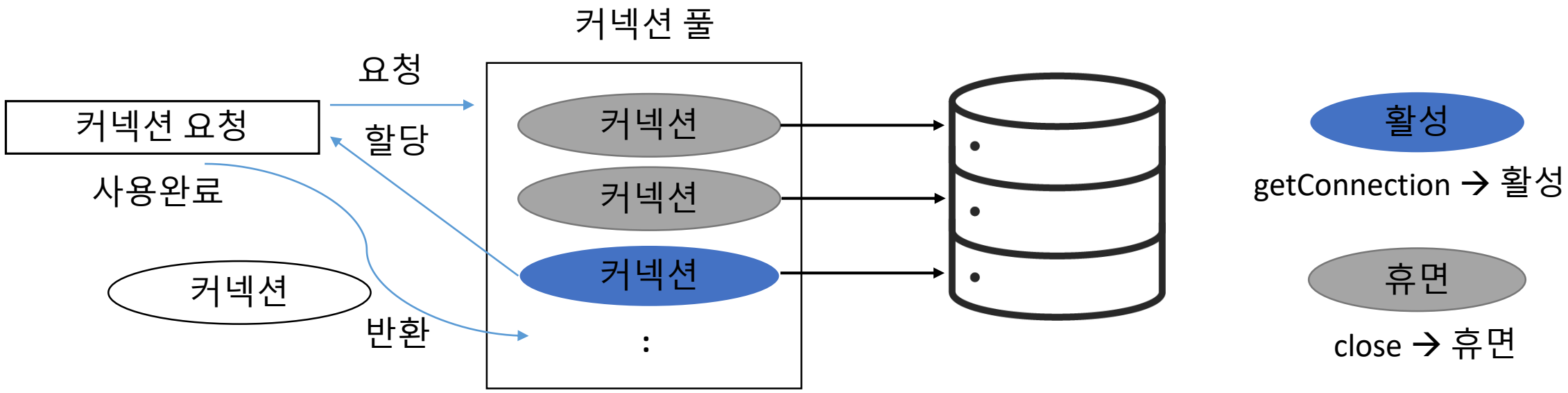
Connection pool
커넥션 풀

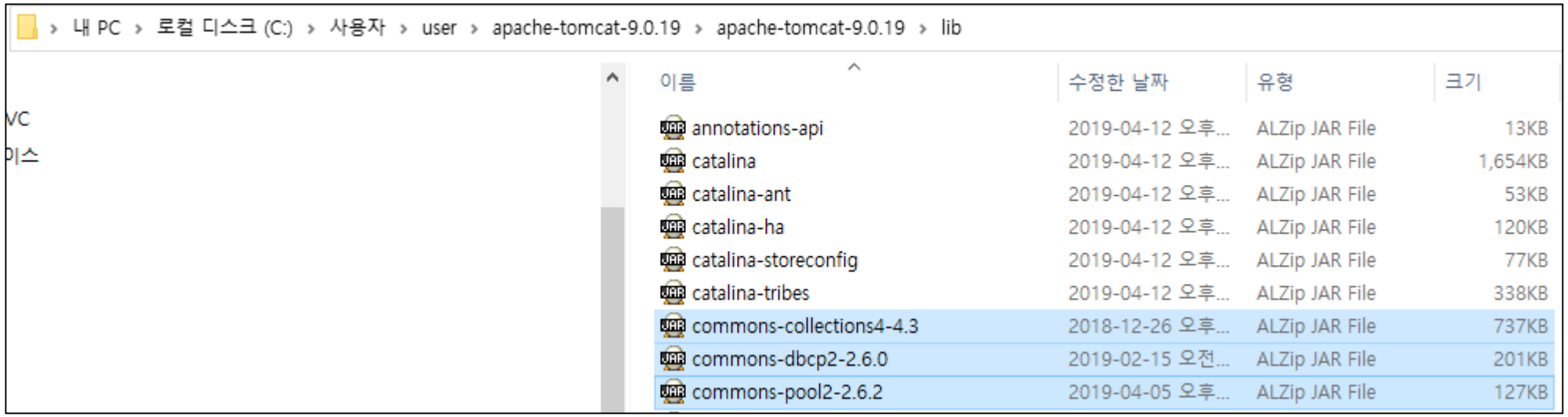
커넥션 풀

- **데이터베이스 커넥션 풀(DataBase Connection Pool, DBCP)**
  - DB에 접근할 때마다 Connection을 수행하는 것은 서버에서 큰 부담
  - 미리 여러 개의 데이터베이스 Connection을 생성해서 보관
  - 요청 마다 하나씩 Connection을 꺼내서 사용하고 사용이 끝나면 다시 Pool로 반환



커넥션 풀

- **데이터베이스 커넥션 풀을 사용하기 위해 필요한 바이너리 파일 다운**
  - http://commons.apache.org/collections/
  - http://commons.apache.org/dbcp/
  - http://commons.apache.org/pool/
- 각 압축파일 해제 후, jar파일을 apache-tomcat설치 경로의 lib폴더에 복사(커넥션 풀은 이클립스 안에서 관리하는 것이 아니라 서버에서 관리)



커넥션 풀

- **Servers 폴더 안에 있는 context.xml파일 끝에 resource추가(아래의 코드에서 4개 수정)**
  - 빨강색 박스로 표시한 부분을 자신의 환경에 맞게 수정

```
<Resource
 name="jdbc/MySQL"
 type="javax.sql.DataSource"
 auth="Container"
 maxActive="30"
 maxIdle="3"
 maxWait="3000"
 username="root"
 password="1234"
 testOnBorrow="true"
 driverClassName="com.mysql.jdbc.Driver"
 url="jdbc:mysql://localhost/mydb?characterEncoding=utf8&amp;serverTimezone=UTC&amp;useSSL=false"/>
</Context>
```

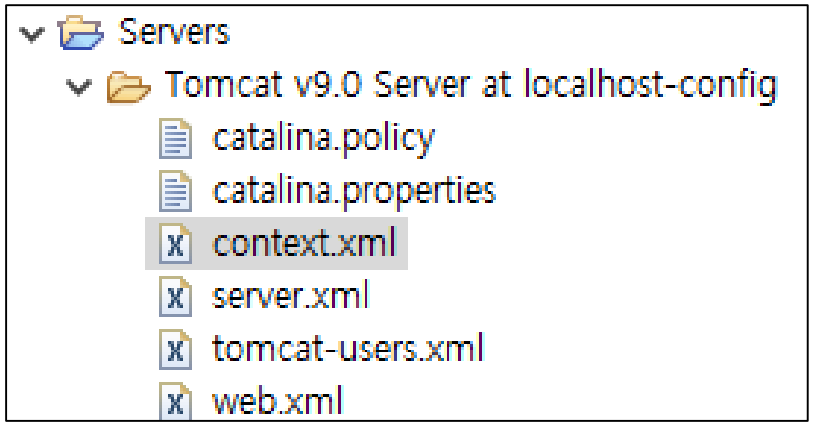
리소스의 이름 기억

사용자 이름

비밀번호

DB명

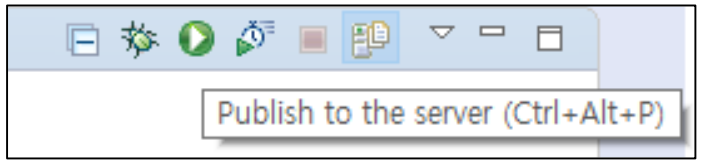
이전 예제에서는 &만 사용



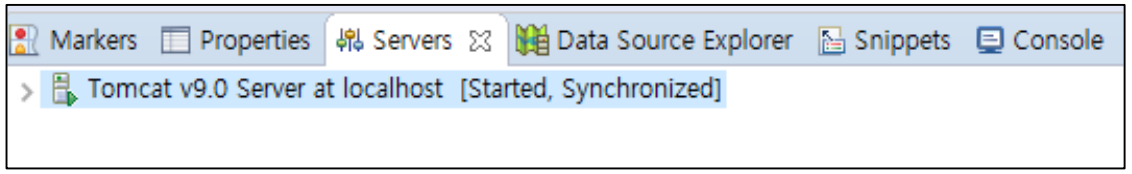
강의자료에 있는 내용을 복사 붙여넣기 할 경우, 문제가 발생할 수 있으니 직접 입력(오타 주의)

커넥션 풀

- **서버 끄고 동기화 수행 후 다시 서버 실행**
  - Connection pool은 서버에서 관리하는 것이지 이클립스 안에 있는 파일에서 관리하는 것이 아니므로 이클립스에서 수정한 context.xml파일 정보를 톰캣 서버에 반영해야 함(동기화, synchronization)



  - 서버 시작



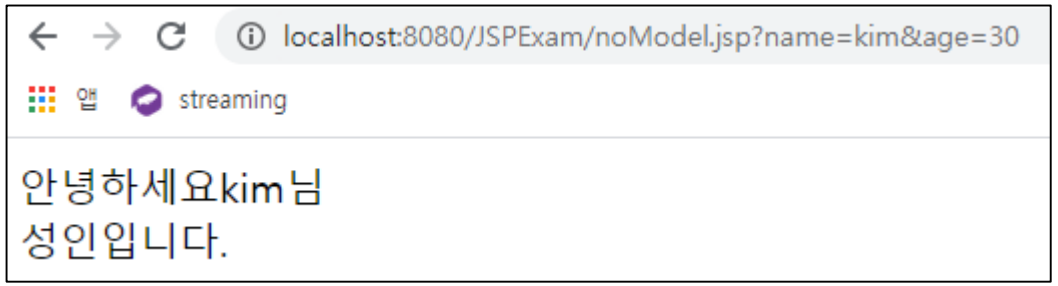
MVC모델
(DTO, DAO, Dispatcher)

MVC모델

- **스파게티 코드**
  - 컴퓨터 프로그램의 흐름이 복잡하게 뒤엉킨 모습을 스파게티가 엉킨 모습에 비유한 표현

```
<%
int intAge = 0;
String name = request.getParameter("name");
String age = request.getParameter("age");

if(age !=null && !age.equals(""))
intAge = Integer.parseInt(age);
%>
```




```
<body>
안녕하세요<%out.print(name);%>님
<br>
<%if(intAge>=20){%>
성인입니다.
<% }
else{%>
미성년자입니다.
<%}%>
</body>
```

MVC모델

- **MVC model1**

```jsp
<%
    int intAge = 0;
    String adult = "미성년자";
    String name = request.getParameter("name");
    String age = request.getParameter("age");

    if(age !=null && !age.equals("")){
        intAge = Integer.parseInt(age);
        if(intAge>=20){
            adult = "성인";
        }
    }
%>
```

Controller

adult

Model

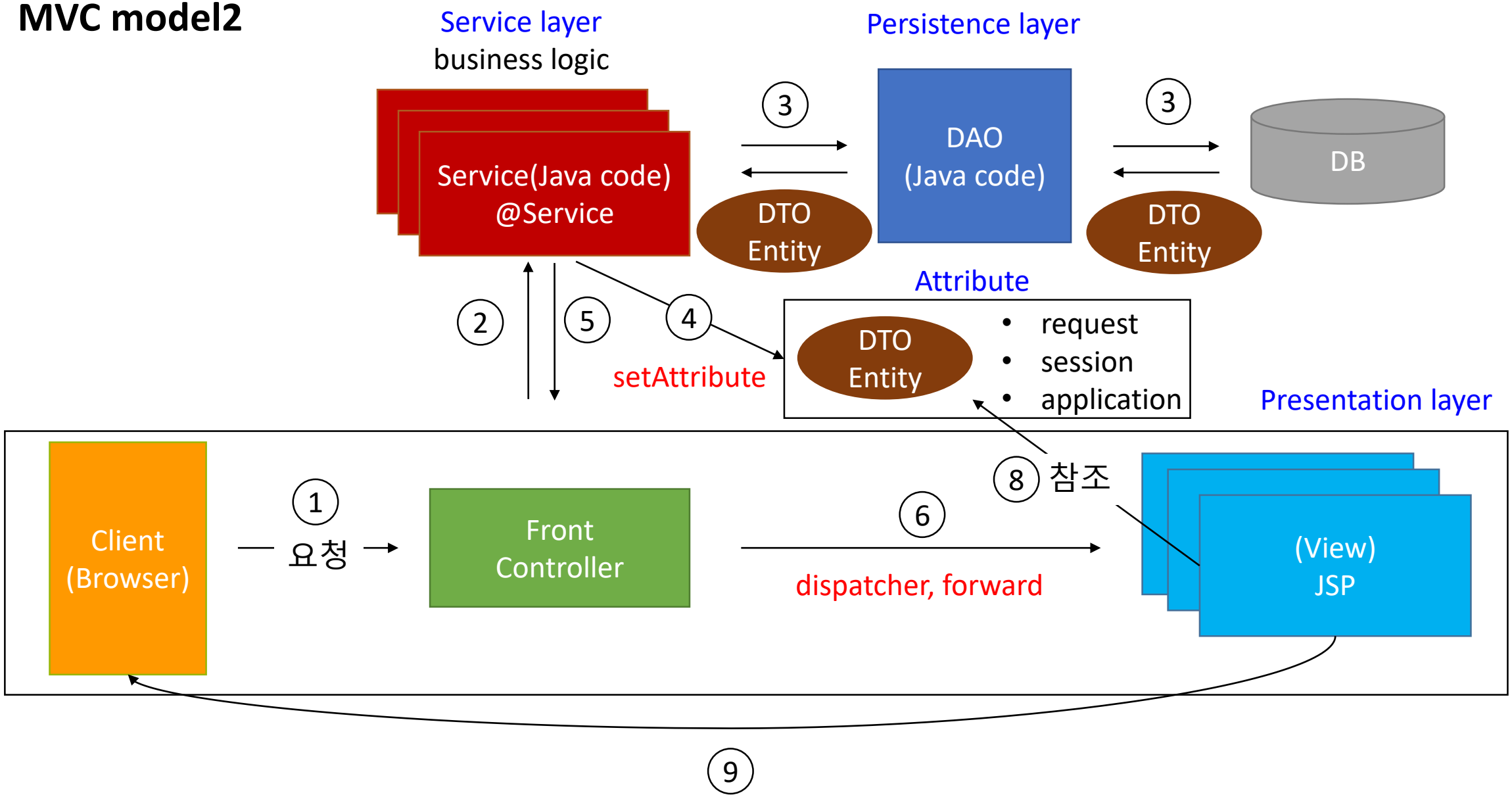
```jsp
<body>
안녕하세요<%=name%>님
<br>
<%=adult%>입니다.
</body>
```

View

control logic과 view logic이 하나의 파일에 존재 → MVC model2로 발전

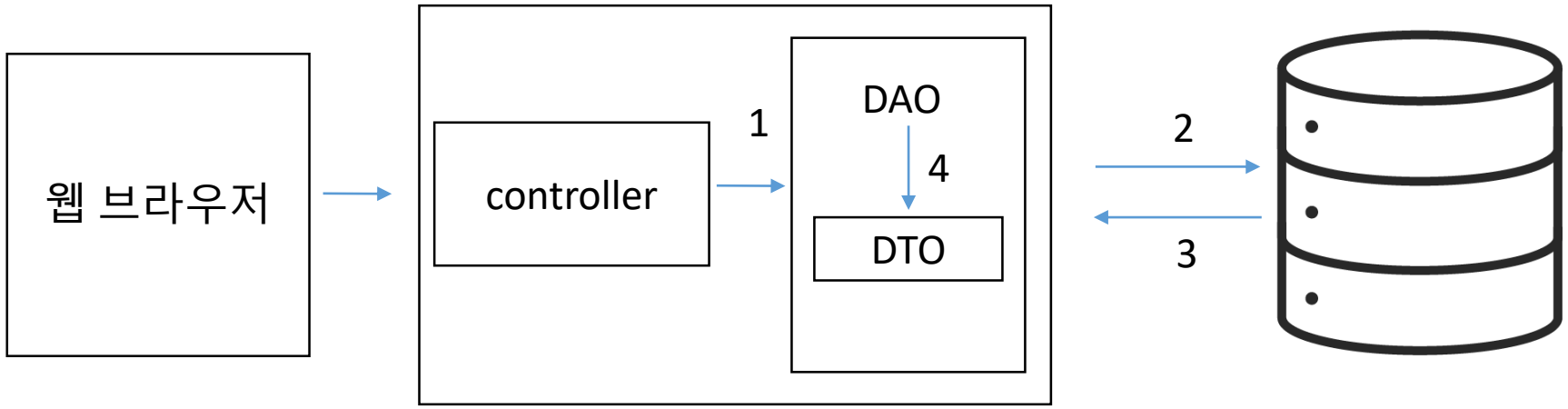
MVC모델

MVC model2



DTO, DAO

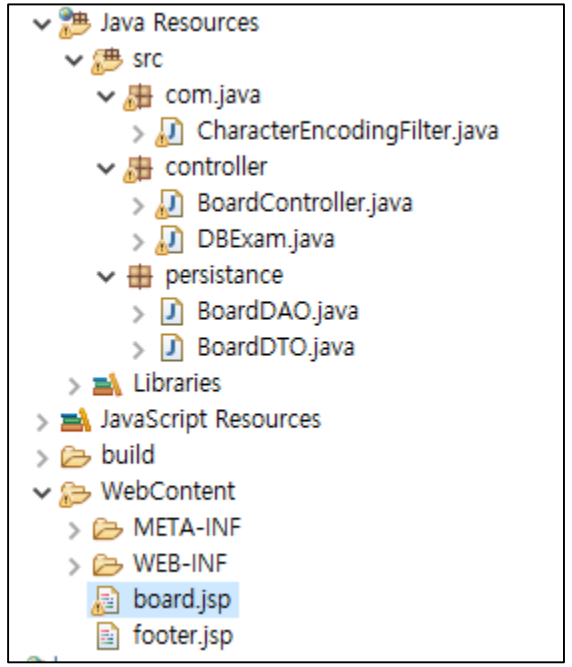
- **DTO(Data Transfer Object)**
  - DTO클래스 = JavaBean(간단하게 여러 데이터를 하나로 묶고 이를 getter와 setter로 조작하는 클래스)
  - VO(Value Object)는 유사한 개념이지만 VO는 read only 프로퍼티만을 가짐
- **DAO(Data Access Object)**
  - DB연결 및 작업 후 연결 해제 담당 → 데이터베이스에 연결하여 수행한 결과를 받아오는 객체
  - Connection pool에서 하나의 connection만 가져와서 이를 통해 DB와 연동
  - 쿼리 결과는 DTO에 넣음(자동적으로 그렇게 된다는 의미는 아니고 우리가 setter나 생성자로 넣어줌)



DTO

- **DTO(Data Transfer Object)**

```
public class BoardDTO {
    private String id;
    private String writer;
    private String title;
    private String contents;
    private LocalDateTime regdate;
    private String hit;
    Getter, Setter
}
```



- 반드시 Getter와 Setter를 작성해야 함
- 이클립스의 경우 소스코드에서 우클릭→source→Generate getters and setters로 자동 생성 가능

DAO

- **DAO(Data Access Object)**

```
public class BoardDAO {
    private DataSource ds;

    public BoardDAO() {
        try {
            Context context = new InitialContext();
            ds = (DataSource) context.lookup("java:comp/env/jdbc/MySQL");
        } catch (NamingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

  - DAO는 데이터베이스에 직접 접근하는 클래스이므로 DataSource를 member로 가짐
  - context.lookup함수에는 context.xml에 명시한 resource의 이름 기입
  - DAO에는 일반적으로 DataSource설정 및 DB연동과 관련된 동작인 SELECT, INSERT, UPDATE, DELETE 함수를 제공하도록 작성

Controller

- **BoardController**

```
@WebServlet("/board/*")
public class BoardController extends HttpServlet{
    private BoardDAO boardDAO = new BoardDAO();
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        ArrayList<BoardDTO> boardDto;
        boardDto = boardDAO.select();
        req.setAttribute("boardDto", boardDto);
        ServletContext context = req.getServletContext();
        RequestDispatcher dispatcher = req.getRequestDispatcher("/board.jsp");
        dispatcher.forward(req, resp);
        . . .
```

  - 결과를 보여 줄 적절한 jsp페이지로 forward
  - 만약 view라는 폴더 밑에 board.jsp가 있었다면 "/view/board.jsp"로 명시

Controller

- **board.jsp**

```
<table width="700" border="3" bordercolor="lightgray" align="center">
    <thead>
    <tr>
    <td>no</td><td>제 목</td><td>글쓴이</td><td>내용</td><td>작성일</td><td>조회수</td>
    </tr>
    </thead>
    <%
    List<BoardDTO> list = (List<BoardDTO>) request.getAttribute("boardDto");
    for (BoardDTO dto : list) {
    pageContext.setAttribute("dto", dto);
    %>
    <tr>
        <td>${dto.id}</td>
        <td>${dto.title}</td>
        <td>${dto.writer}</td>
        <td>${dto.contents}</td>
        <td>${dto.regdate}</td>
        <td>${dto.hit}</td>
    </tr>
    <%}%>
</table>
```

Dispatcher

- 클라이언트 요청을 받은 서블릿(또는 JSP)는 현재 작업중인 페이지에서 다른 페이지로 이동
- 두 가지 방식
  - Dispatcher
    - forward
    - include
  - Redirect
    - sendRedirect

Dispatcher

- **dispatcher 획득 방법**
  - ServletContext 이용

web.xml에 정의된 servlet이름


```
ServletContext context = request.getServletContext();
RequestDispatcher dispatcher = context.getNamedDispatcher("helloServlet");
```

```
ServletContext context = request.getServletContext();
RequestDispatcher dispatcher = context.getRequestDispatcher("/hello");
```

  - request를 이용

```
RequestDispatcher dispatcher = request.getRequestDispatcher("/hello");
```

  - JSP페이지에서

```
RequestDispatcher dispatcher = application.getRequestDispatcher("/hello");
```

Dispatcher

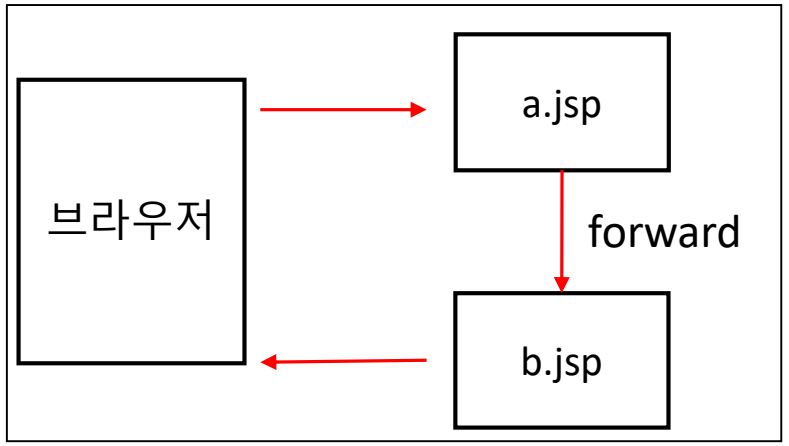
- Dispatcher를 이용해 요청을 보내는 방법(forward)

```
RequestDispatcher dispatcher = request.getRequestDispatcher("/hello");
dispatcher.forward(request, response);
```

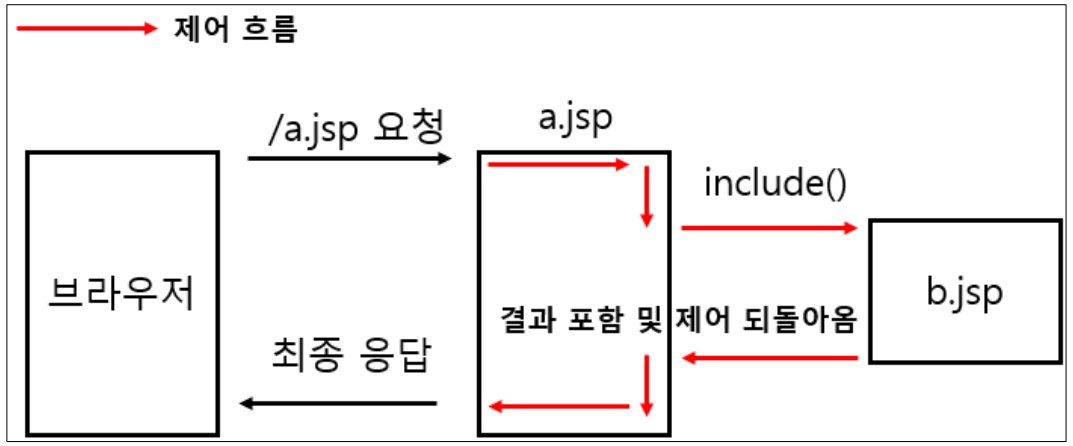
- forward() 사용시 주의할 점
  - forward()는 제어를 넘기기 이전에 출력 버퍼를 비움
  - source.jsp -> destination.jsp로 호출 시 source.jsp에서 수행한 버퍼출력은 완전히 무시되며 제어가 넘어간 destination.jsp의 출력 내용만 브라우저에게 전달

Dispatcher

- Dispatcher를 이용해 요청을 보내는 방법(include)
  - dispatcher.include(request, response) : 처리 흐름 제어를 잠시 특정 대상에게 넘기고 대상이 처리한 결과를 현재 페이지에 포함시키는 기능
  - include()는 출력버퍼를 비우지 않기때문에 호출하는 페이지에서 출력하는 내용에 이어서 호출되는 페이지에서 출력하는 내용까지 포함시킬 수 있음



[forward]



[include]

Dispatcher

- Redirect

```
response.sendRedirect("이동할 페이지");
```

  - 웹 서버 측에서 웹 브라우저에게 어떤 페이지로 이동하라고 지정하는 것
    - 게시판에 글을 저장한 후에 목록 페이지로 이동하는 경우
    - 로그인이 필요한 요청을 했을 때 로그인 페이지로 이동하는 경우
  - 주의할 사항
    - Response 객체에 쓰기 작업을 한 뒤에는 sendRedirect()를 할 수 없음
    - sendRedirect() 메소드는 매개변수를 URL 객체가 아닌 String 객체를 받음
  - SendRedirect로 페이지를 이동할 때, 데이터 공유가 필요하다면
    - session 혹은 application객체의 setAttribute이용
    - Query string이용