웹클라이언트실행프로그램 =웹브라우저 실행	웹서버실행프로그램 = 웹요청(url)+자바실행 =tomcat
html css javascript jquery	asp asp.net php cgi 자바작성 웹서버실행 servlet , jsp

1. 클라이언트1는 브라우저에서
HTTP://ip:port/프로젝트명/서블릿url 요청한다
2. 서버는 웹서버가 웹컨테이너로 위임한다.
3. 위임받은 웹컨테이너가 메모리 서블릿 객체 로딩되어 있지 않으면 서블릿 객체 메모리 로딩한다.
4. init 호출한다.-서블릿 처리 이전 초기화 구현
5. 서블릿 객체 멀티스레드 실행한다.
6. 요청 정보 HttpServletRequest 객체와 응답 정보 생성할 HttpServletResponse 객체를 생성한다.
7. doGet(HttpServletRequest, HttpServletResponse)/doPost/Service 호출하여 응답한다.
8. 클라이언트2는 브라우저에서
HTTP://ip:port/프로젝트명/서블릿url 요청한다
9. 서버는 웹서버가 웹어플리케이션 서버로 위임한다.
10. 요청 정보 HttpServletRequest 객체와 응답 정보 생성할 HttpServletResponse 객체를 생성한다.

11. doGet/doPost/Service 호출하여 응답한다.

.....

12. 서버 종료나 서블릿 소스 변경 – 재컴파일시에는
서블릿 객체 메모리 삭제한다.

13. destroy() 호출한다.- init 변수 메모리 삭제

14. 서블릿 메모리 제거한다.

15.

http://ip:9090/servlettest/test 입력

서버컴퓨터/servlettest/servlet/servlettest.TestServlet 실행

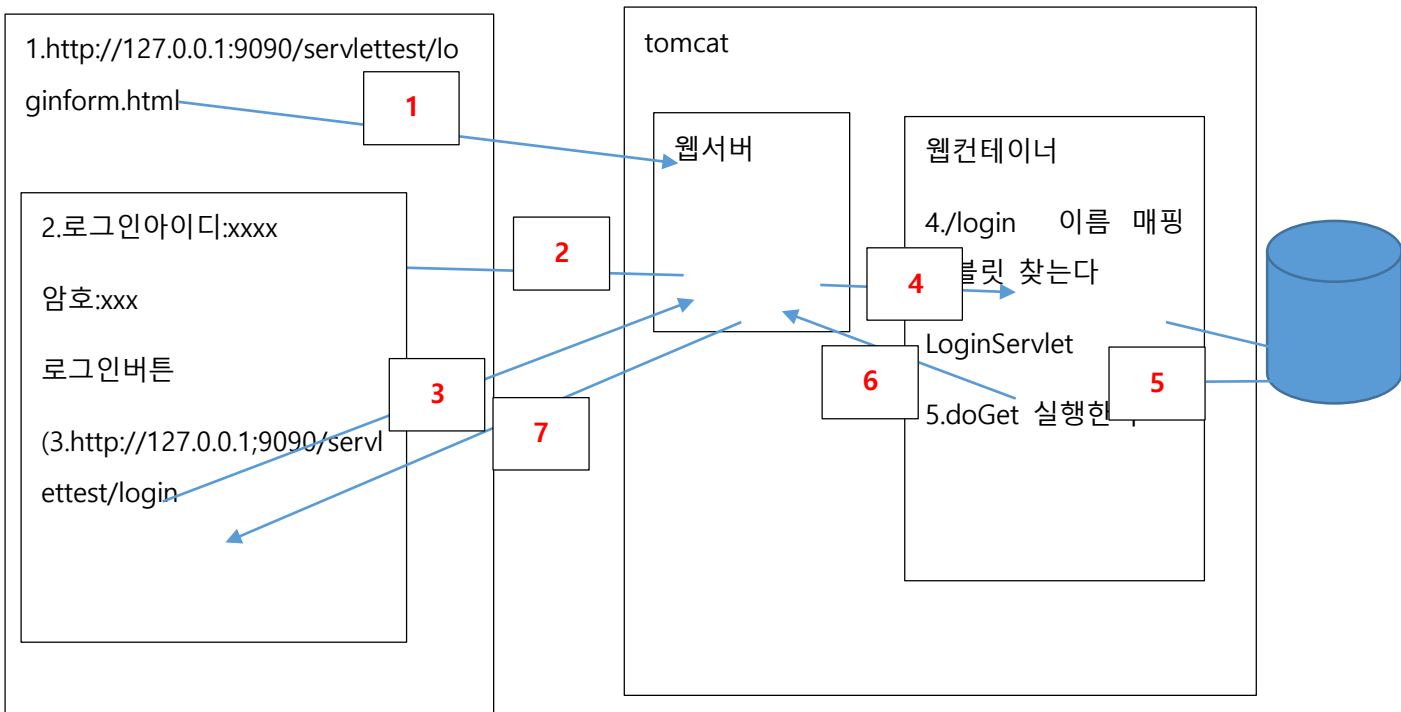
servlettest.TestServlet 클래스 정의

/test url 매핑 호출

web.xml

dynamic web project마다 1개씩 존재 파일- 각 프로젝트 환경설정

@WebServlet("/test")



get(실습 확인....)	post
url뒤에 데이터 전송 url 에서 ? 뒷부분 name1=value1 & name2=value2... 전송 데이터 길이 250자 1.<form action="login" > 2.<form action="login" method=get > 3.주소 입력 : http:..... 4.<a href="login"> 서블릿 이동 get - 기본 방식 doGet(....)	url 과 별도 분리 데이터 전송 url 뒤 안보인다 (url + ip + post데이터들) 보안 강하다 파일 업로드 길이 긴 데이터 전송 유리하다 서블릿에서는 속도 느리다 1.<form action="login" method=post> doPost(....)

7장 서블릿 : 요청 – 처리(jdbc , io) – 응답

비지니스 로직 처리

BoardMain 메뉴 글쓰기 게시물리스트	BoardInsertView input() 제목 내용 작성자	BoardDAO insertBoard(BoardDTO) ArrayList<BoardDTO> getBoardList()	BoardDTO =BoardVO

boardlist.html 게시물리스트	BoardServlet doGet()	BoardDAO insertBoard(BoardDTO) ArrayList<BoardDTO> getBoardList()	BoardDTO =BoardVO

- DataSource 사용하기

1.connection 생성 시간 오래 걸린다.

(sql 전송 결과 리턴 시간보다)

2.connection이 필요없을 때 메모리에서 삭제하고

3. 필요하면 생성한다.

---> "미리" 생성 (정한 갯수) / 보관 다시 재사용 – ConnectionPooling 기능 api

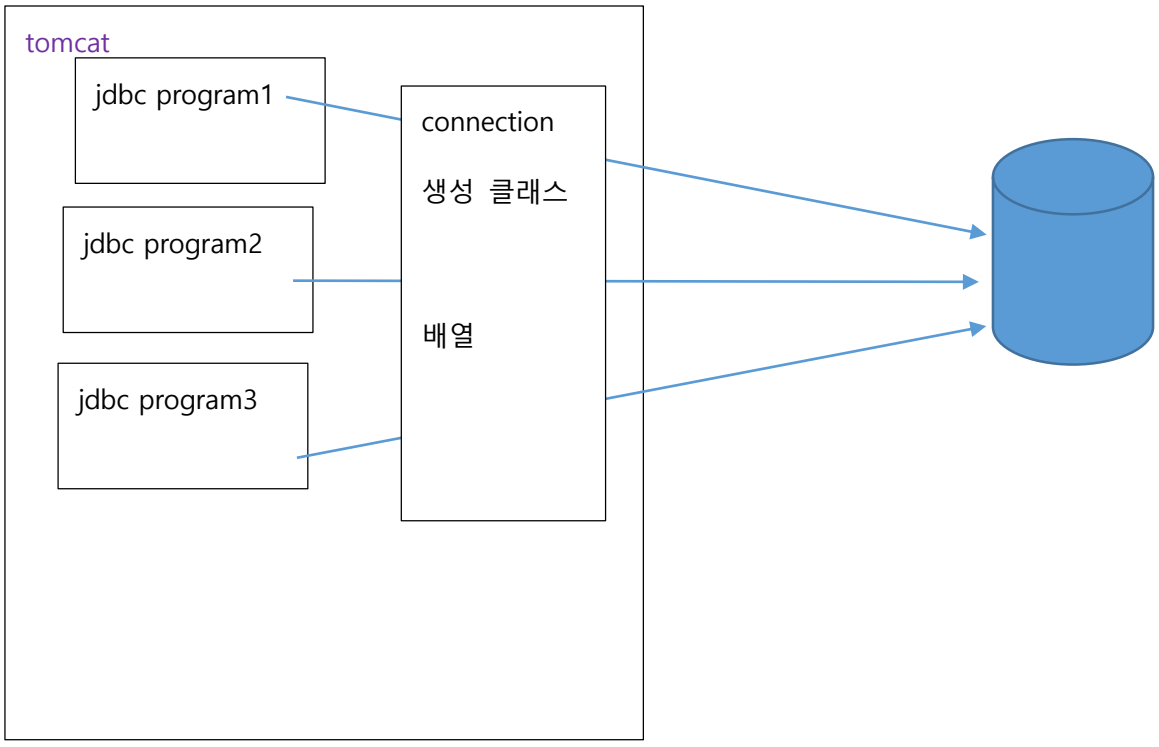
4. 직접 구현 / 서버 미리 구현 제공

5. tomcat server.xml 파일 설정

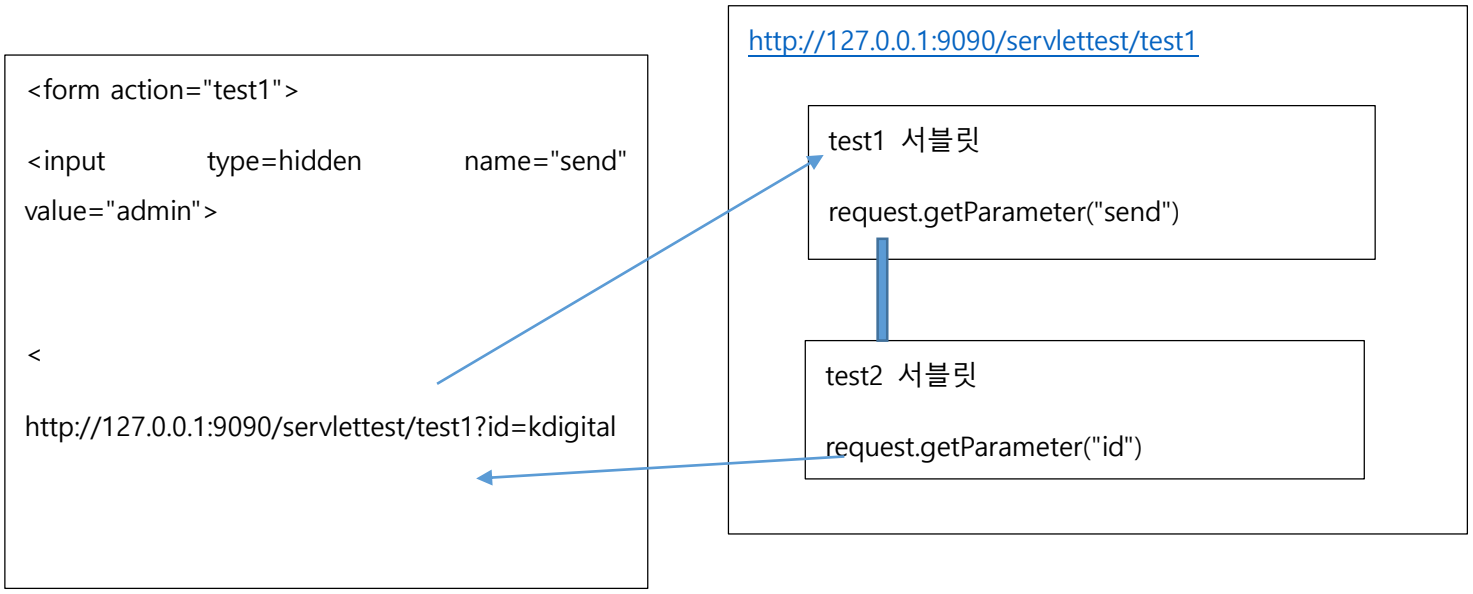
(server.xml 파일 편집 -)

6. 서블릿, dao – 데이터소스 이용 코드

--->



8장



RequestDispatcher – 다른 서블릿 파일 호출 연동

RequestDispatcher dis = request.getRequestDispatcher("test2");

request.setAttribute("", Object들);/ ()request.getAttribute("'")

dis.forward(request, response);

RequestDispatcher dis = request.getRequestDispatcher("test2");

dis.include(request, response);

session.setAttribute("", Object들);/ ()session.getAttribute("'")

context.setAttribute("", Object들);/ ()context.getAttribute("'")