저번주 수업 Command handler 방식 정리

1. 서블릿 :: Controller.java
   모든 '요청' 받는 역할. 실제 '기능 구현'은 Command 추상클래스를 상속받아서 작동하게끔.
   - init() 메서드로 properties 파일에 저장해놓은 command들을 'cmds' 변수에 저장.
   - doGet() 메서드로 cmd값을 읽어오면 해당 값에 mapping 되어 있는 class fullname을 가지고 Command 객체를 생성. 해당 객체를 'command' 변수에 저장.

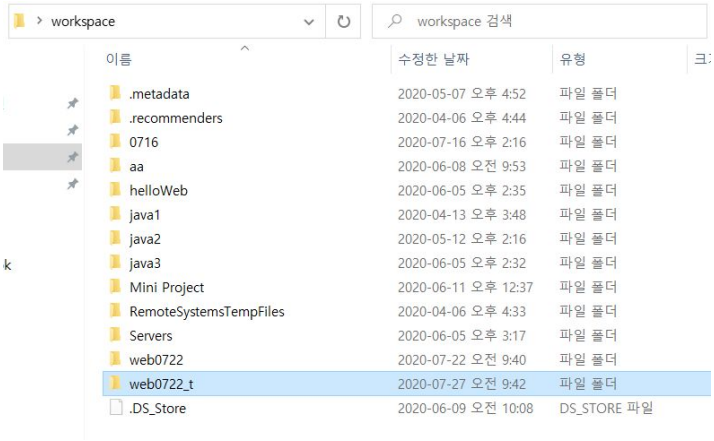
   (Spring에서는 해당 역할 숨겨져 있음)
2. 추상클래스 :: Command.java
   Command를 처리하는 역할.
   '명령어 수행(execute)', '페이지 이동(forward/sendredirect)'을 담당하는 코드의 껍데기.
3. Command.java의 자식클래스
   껍데기를 상속받아 각 명령의 기능에 적합하게 execute를 구현.
   a. ListCommand(글 목록 출력)
   b. WriteFormCommand(글 작성 폼)
   c. WriteCommand
   d. EditFormCommand
   e. EditCommand
   f. DelCommand
4. Properties 파일 :: cmd.properties
   'name = class full name' *class full name: package name + file name
   페이지 처음 실행될 때(Controller.java) init 메서드가 해당 파일을 읽음.
   이후에 name으로 class full name 접근해서 해당 객체를 생성함.

- cf1. 서블릿/jsp 파일의 생명주기 함수(웹 컨테이너를 거치면 jsp파일도 java 파일로 변경되기 때문에 jsp = 서블릿)

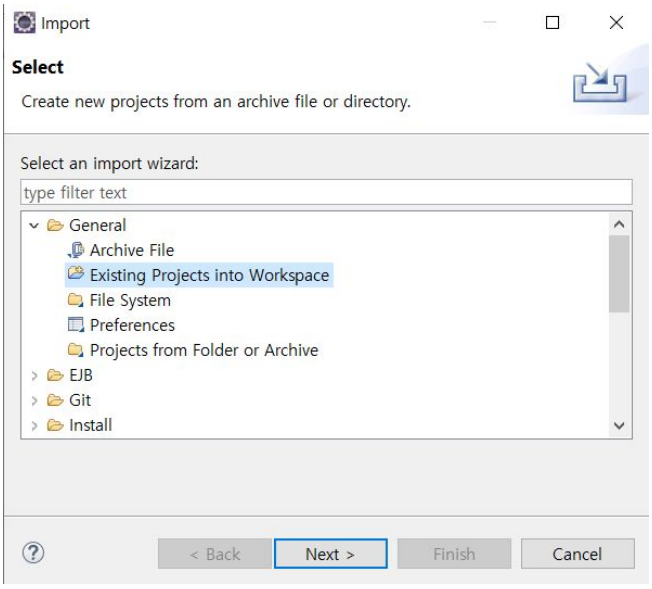
- init(): 해당 페이지를 실행할때 단 한번만 실행됨.
  ex. 많은 클라이언트가 접속하는 경우에도, 맨 처음 단 한번만 실행
- service(): 클라이언트가 페이지를 요청할 때마다 실행됨.
  즉, 클라이언트의 요청을 처리하는 메서드
  ex. 많은 클라이언트가 접속하는 경우에, 매 요청마다 실행.
- destroy(): 메모리에서 소멸될 때 실행됨.

- cf2. 외부 project 내 eclipse에 가져오기

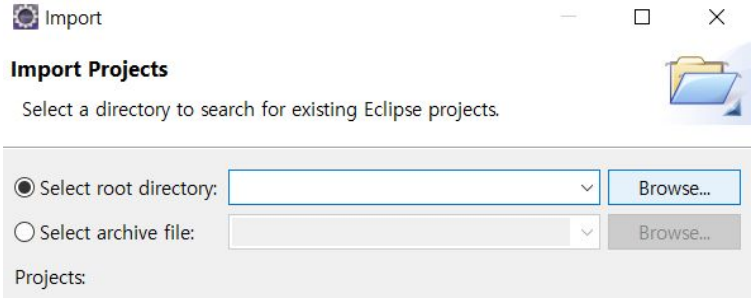
1) 해당 디렉토리 내 workspace에 넣기

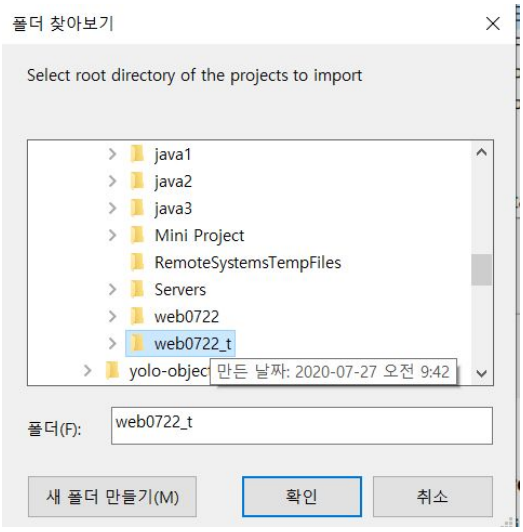


2) 이클립스 오픈 => 마우스 오른쪽 import=>general의 existing projects into workspace

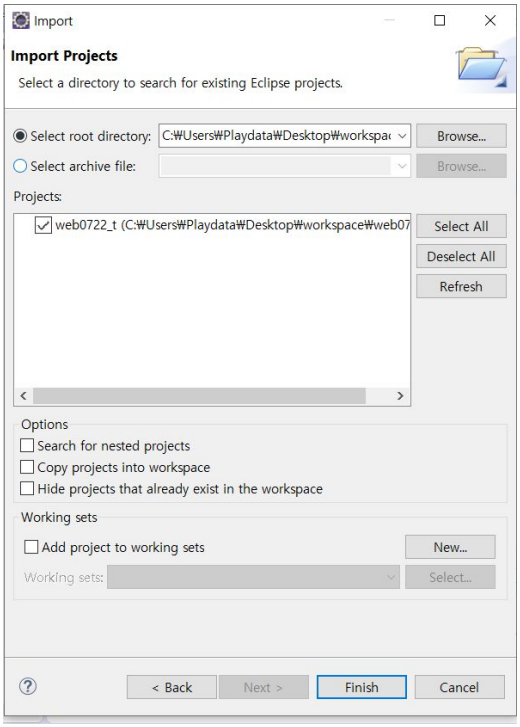


3) import projects => browse에서 내 workspace안에 옮기고 싶은 디렉토리 선택



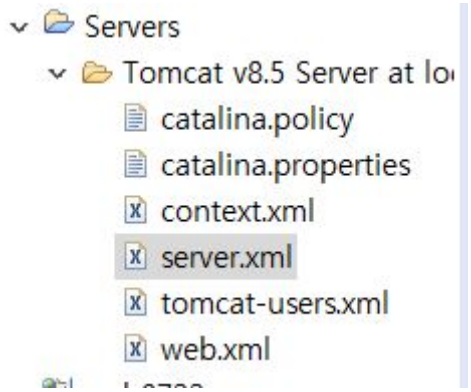


4) finish



***외부 프로젝트 내 이클립스에 붙여넣을 때, 동일한 프로젝트명때문에 충돌 날 경우 해결방법

1) 붙여넣을 웹 프로젝트 이클립스에서 실행 => 오류 뜨더라도 server.xml 파일에 해당 웹프로젝트 정보 update됨
2) Servers의 server.xml 파일 open



3) 하단에 프로젝트 path 검토하기. 두 개의 web project가 동일한 path를 쓸 경우 서버 포트 충돌 발생.

```
<Context docBase="web0722" path="/web0722" reloadable="true" source="org.eclipse.j
<Context docBase="web0722_t" path="/web0722" reloadable="true" source="org.eclipse
```

4) path 변경하기

```
<Context docBase="web0722" path="/web0722" relo
<Context docBase="web0722_t" path="/web0722_t"
```

5) jsp파일 내에 url 모두 변경
or 처음부터 el 표현식 사용 (일일히 바꿀 필요 없음)

```
<a href="${pageContext.request.contextPath}/guestbook/Controller?cmd=writeForm">글작성</a><br/>
```

6) 서버 재시작

- scope 객체와 el표현식

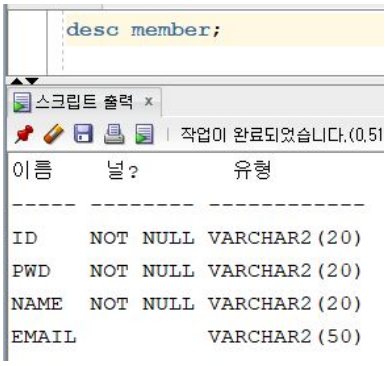
1) Scope 객체
   - 특징
     - 담을 수 있는 종류 page, request*, session*, application
     - 각 종류마다 범위, 유효시간이 다르다.
     - 모두 setAttribute(name,value), getAttribute(name),removeAttribute()를 가진다.
       *object 타입으로 저장됨. 다운캐스팅 필요
   - 종류
     - page: 현재 페이지에서만 유지
     - request: 클라이언트의 요청을 객체로 만든 것.
       해당 요청에 대한 응답을 보내기 전까지 유지
     - session: http 프로토콜은 요청이 하나 끝나면 정보 삭제. 따라서 로그아웃 시까지 유지시킬 정보가 있을 경우 사용. (정보를 서버에 저장.)
       로그아웃 할 때까지 유지
     - application(서버에 올라간 해당 프로젝트 하나): 현재 어플리케이션이 끝날 때(서버에서 내려갈 때)까지 계속 유지
2) el표현식
   - scope객체를 get하는 방식으로 사용하지 않고, 과정 생략 후에 내가 필요로 하는 값을 바로 get.
   - 용도: request나 session에 저장해놓은 값을 꺼내어 쓸 때 사용.

- 예시: OO Comand.java의 setAttribute사용 시 저장한 이름을 가지고, private 변수인 a를 바로 참조할 수 있다.
  ListCommand.java => setAttribute("a",a);
  list.jsp => ${a.num}, ${a.content}.....
- 원리: 내부에서 자동으로 setter, getter를 호출하여 기능.

1. 회원가입

a. 기본 세팅

1) oracle db member table 확인



2) src/Controllers 패키지 생성
3) src/Controllers에 MemController서블릿 생성
4) cmd 패키지 생성, cmd.member 패키지 생성
   cmd>>cmd.member(회원가입용 명령)
5) cmd에 guestbook패키지의 추상클래스 Command.java 붙여넣기
6) WEB-INF에 cmdprop 폴더 생성
7) /WEB-INF/cmdprop에 memcmd.properties 파일 생성

b. MemController.java

1) WebServlet 속성 변경

```
@WebServlet(urlPatterns = "/MemController", initParams = {
        @WebInitParam(name = "mapping", value = "/WEB-INF/cmdprop/cmd.properties") })
```

2) 멤버 변수 cmds 선언

```
private Properties cmds;
```

3) override 메서드의 generic => init(servlet config) 생성
4) init 메서드 수정

```
@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
    String path = config.getInitParameter("mapping");// 이름이 mapping인 초기화
    // 파라메터의 값을 읽음
    InputStream is = getServletContext().getResourceAsStream(path);
    cmds = new Properties();
    try {
    cmds.load(is);
    is.close();
    } catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    }
}
```

5) doGet 메서드 수정

```
protected void doGet(HttpServletRequest request, HttpServletResponse respon
    String cmd = request.getParameter("cmd");

    if (cmd == null || cmd.equals("")) {
        cmd = "list";
    }

    String className = cmds.getProperty(cmd);
    try {
        Command command = (Command) Class.forName(className).newInstance();
        command.execute(request, response);
    } catch (InstantiationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IllegalAccessException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

c. cmd.member 패키지

1) JoinFormCommand.java (가입JoinForm.jsp로 페이지 이동-f)
2) JoinCommand.java (db에 새 멤버 정보 저장. LoginFormCommand로 이동-r)
3) LoginFormCommand.java (로그인loginForm.jsp로 페이지 이동-f)
4) LoginCommand.java (로그인. 세션에 id 저장-f)
5) EditFormCommand.java
6) EditCommand.java
7) LogoutCommand.java(세션 remove, 무효화, loginForm.jsp로 페이지 이동-f)
8) OutCommand.java

d. webContent/views/member 폴더 (뷰 페이지)

1) JoinForm.jsp
2) LoginForm.jsp
3) result.jsp
4) editForm.jsp *<script>'{a.pwd}' => string이면 따옴표로 묶어주기

e. memcmd.properties

```
joinForm=cmd.member.JoinFormCommand
join=cmd.member.JoinCommand
loginForm=cmd.member.LoginFormCommand
login=cmd.member.LoginCommand
editForm=cmd.member.EditFormCommand
edit=cmd.member.EditCommand
logout=cmd.member.LogoutCommand
out=cmd.member.OutCommand
```

2. 게시판

a. 기본세팅

1) oracle db board table 확인

```
이름      널?       유형
------- -------- -------------
NUM     NOT NULL NUMBER
WRITER           VARCHAR2(20)
TITLE            VARCHAR2(20)
W_DATE           DATE
CONTENT          VARCHAR2(100)
```

2) board 패키지 생성
   - dto 클래스 Board 클래스 생성
   - dao 인터페이스 생성
   - daoImpl
   - service 인터페이스 생성
   - serviceImpl
3) boardcmd.properties 파일 생성
4) /WebContent/views에 board 디렉터리 생성
5) /controllers에 BoardController.java 생성
   - webServlet 속성 변경
   - 멤버변수 추가
   - init()메서드 수정
   - do get() 메서드 수정
6) 오라클에서 예시글 insert 하기

b. ListCommand.java

c. list.jsp

- jstl-1.2jar 파일 lib에 저장

*jstl 사용을 위해서는 jsp파일 상단에 taglib를 넣어줘야 한다.
for(Board b:list){}와 동일
<c:forEach var="b" items="${list}"></c:forEach>

```
<c:forEach var="b" items="${list}">
    <tr>
        <td>${b.num}</td>
        <td>${b.writer}</td>
        <td>${b.w_date}</td>
        <td>${b.content}</td>
    </tr>
</c:forEach>
```

d. result.jsp

```
<a href="${pageContext.request.contextPath }/BoardController?cmd=list">게시판</a><br/>
```