9_1_2022_SPRING

ModelAndView

```
@RequestMapping("/member/list")
   public ModelAndView list() {
     //view page 에 전달할 모델(data)
     List<MemberDto> list=dao.getList();
     //Model과 view page의 정보를 동시에 담을 객체를 생성해서
     ModelAndView mView=new ModelAndView();
     //view page 에 전달할 모델을 list라는 키값으로 담고
     mView.addObject("list",list);
     //view page의 정보도 담고
     mView.setViewName("member/list");
     //ModelAndView 객체를 리턴해주면 동일하게 동작한다.
     return mView;
   }
```

```
@RequestMapping("/member/updateform")
   public ModelAndView updatrform(int num, ModelAndView mView) {
    MemberDto dto = dao.getData(num);
    mView.addObject("dto", dto);
    mView.setViewName("member/updateform");
    return mView;
   }
```

```
@RequestMapping("/member/insertform")
   public ModelAndView insertform(ModelAndView mView) {
     //모델은 담지 않고 view page정보만 담아서 리턴할 수도 있다.
     mView.setViewName("member/insert");
     return mView;
   }
```

redirect

```
@RequestMapping("/member/delete")
   public String delete(int num) {
     //MemberDao 객체를 이용해서 DB에서 삭제하고
     dao.delete(num);
     /* /member/list.do 요청을 다시 하도록 리다이렉트 이동시킨다.
      *  리다이렉트 이동은 특정 경로로 요청을 다시 하라고 강요
      *  "redirect: context 경로를 제외한 요청을 다시 할 절대경로"
```

```
     */
    return "redirect:/member/delete";
  }
```

의존성 줄이기

컨트롤러에 대한 이해

- 컨트롤러에서는 dao를 이용한 비즈니스 로직 혹은 복잡한 비즈니스 로직 처리는 하지 않느게 원칙이다.
- 따라서 컨트롤러에서 dao 에 의존하고 있다는 것은 바람직한 구조가 아니다.
- 그럼 dao를 활용한 비즈니스 로직처리는 어떻게 해야 하나?
  - 답 : 서비스 객체를 이용해서 비즈니스 로직을 처리해야 한다.
- 따라서 컨트롤러는 dao에 의존하지 말고 서비스에 의존하도록 해야 한다.
- 그럼 컨트롤러에서는 무엇을 해야 하나?
  - 답 : 클라이언트의 어떤 경로 요청에 대해서 어떤 서비스로 비즈니스 로직을 처리하고
    어디로 어떻게 이동해서 응답해야 할 지에 대한 작업만 하면 된다.

▼ .member.service 패키지

▼ MemberService.java interface

```
package com.livelikesloth.step03.member.service;

import org.springframework.web.servlet.ModelAndView;

import com.livelikesloth.step03.member.dto.MemberDto;

public interface MemberService {
  //회원정보를 추가해 주는 메소드
  public void addMember(MemberDto dto);
  //회원정보를 수정 해 주는 메소드
  public void updateMember(MemberDto dto);
  //회원정보를 삭제 해 주는 메소드
  public void deleteMember(int num);
  //회원 한명의 정보를 인자로 전달함 ModelAndView 객체에 담아주는 메소드
  public void getMember(int num, ModelAndView mView);
  //회원 전체의 정보를 인자로 전달한 ModelAndView 객체에 담아주는 메소드
```

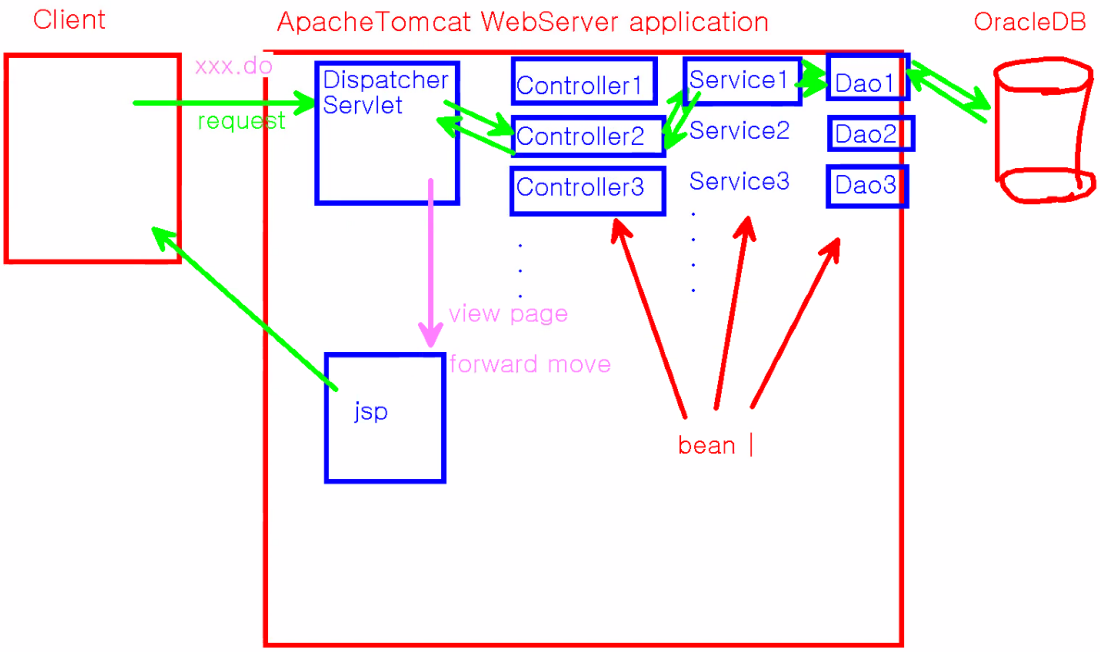
```
  public void getListMember(ModelAndView mView);

}
```

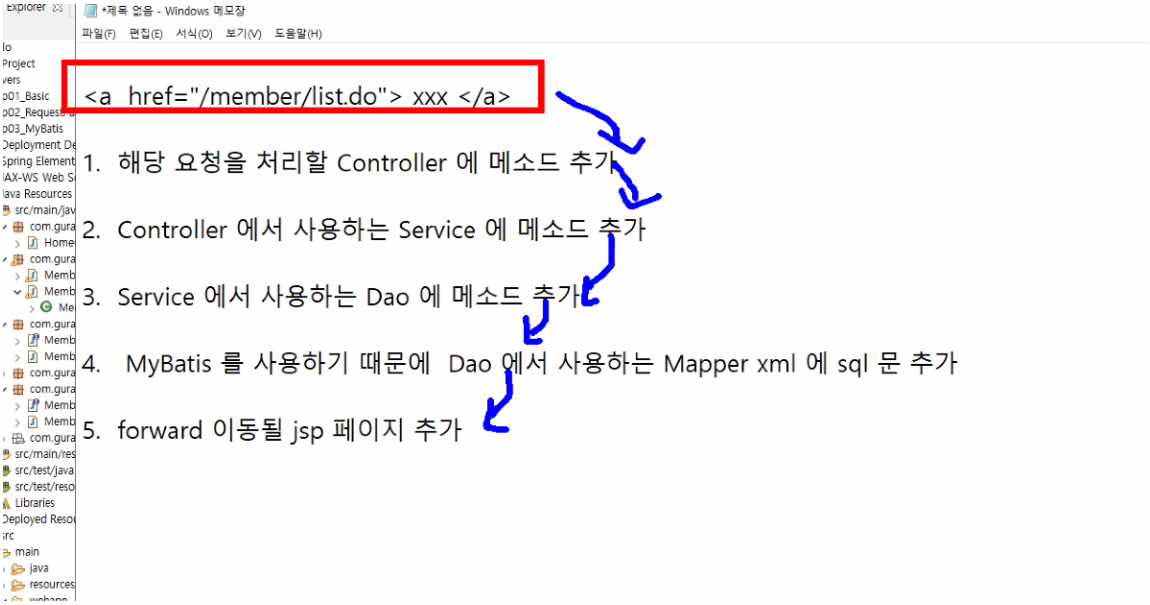
▼ MemberServiceImpl.java

```
package com.livelikesloth.step03.member.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.servlet.ModelAndView;

import com.livelikesloth.step03.member.dao.MemberDao;
import com.livelikesloth.step03.member.dto.MemberDto;

@Service
public class MemberServiceImpl implements MemberService {
  //의존객체 주입받기

  @Autowired
  private MemberDao dao;

  @Override
  public void addMember(MemberDto dto) {
    dao.insert(dto);

  }

  @Override
  public void updateMember(MemberDto dto) {
    dao.update(dto);

  }

  @Override
  public void deleteMember(int num) {
    dao.delete(num);

  }

  @Override
  public void getMember(int num, ModelAndView mView) {
    MemberDto dto = dao.getData(num);
    mView.addObject("dto", dto);

  }

  @Override
  public void getListMember(ModelAndView mView) {
    List<MemberDto> list = dao.getList();
    mView.addObject("list", list);

  }
```

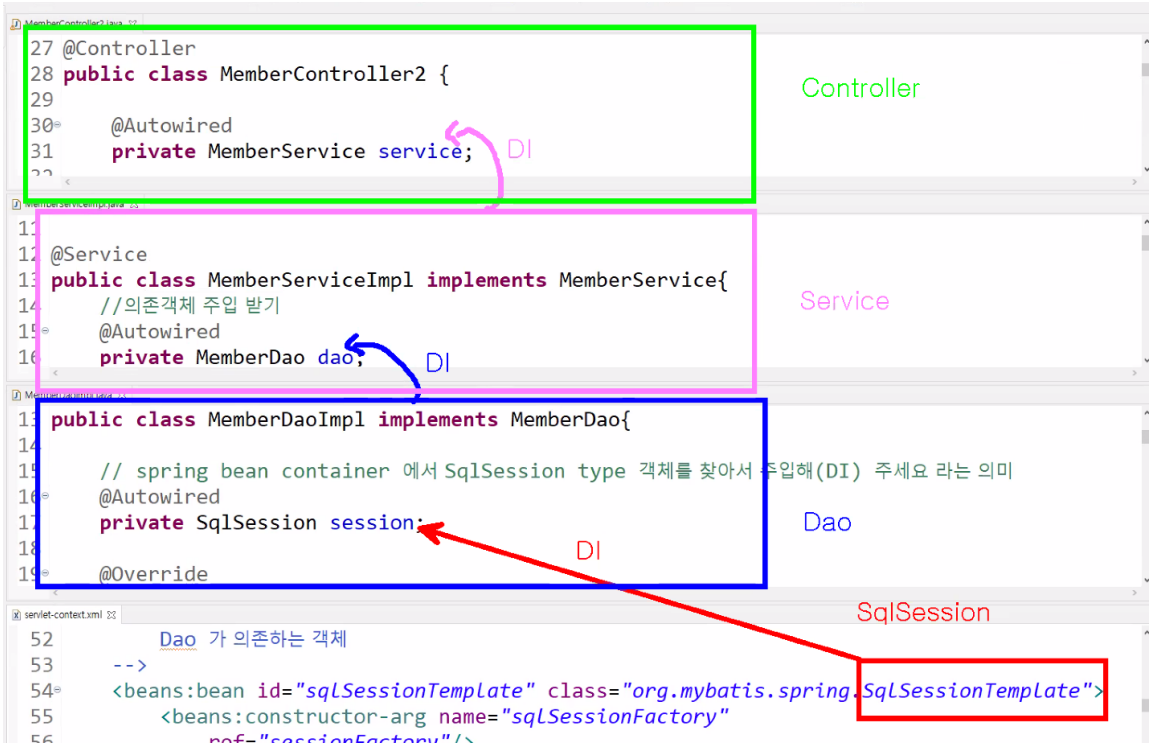
```
}
```

응답순서

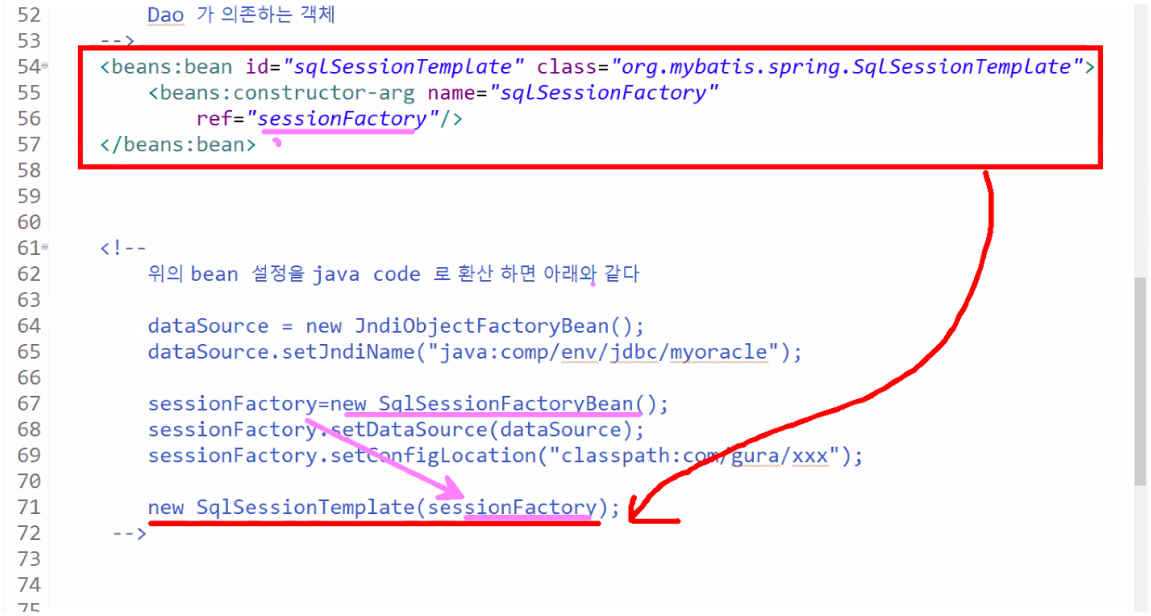




의존관계가 느슨해 유지보수가 용의



db 연동을 잘 하려면 sqlSessionTemplate잘 연동 해야함



JSON응답하는 방법

▼ JSONTestController.java

```
package com.livelikesloth.step03;

import java.util.HashMap;
import java.util.Map;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

/*
 *  1.Jackson-databind 라이브러리가 Dependency에 명시되어 있고
 *  2.servlet-context.xml에 <annotation0driven/> 이 명시 되어있고
 *  3.컨트롤러의 메소드에@ResponseBody 어노테이션이 붙어있으면
 *  Map or dto or List 객체에 담긴 내용이 json 문자열로 변환되어 응답한다.
 */
@Controller
public class JSONTestController {

  @RequestMapping("/send")
  @ResponseBody
  public Map<String, Object> send(String msg) {
    Map<String, Object> map=new HashMap<>();
    map.put("num", 1);
    map.put("name", "두부");
    map.put("isMale", true);

  }
}
```

▼