Chap09

비즈니스 계층

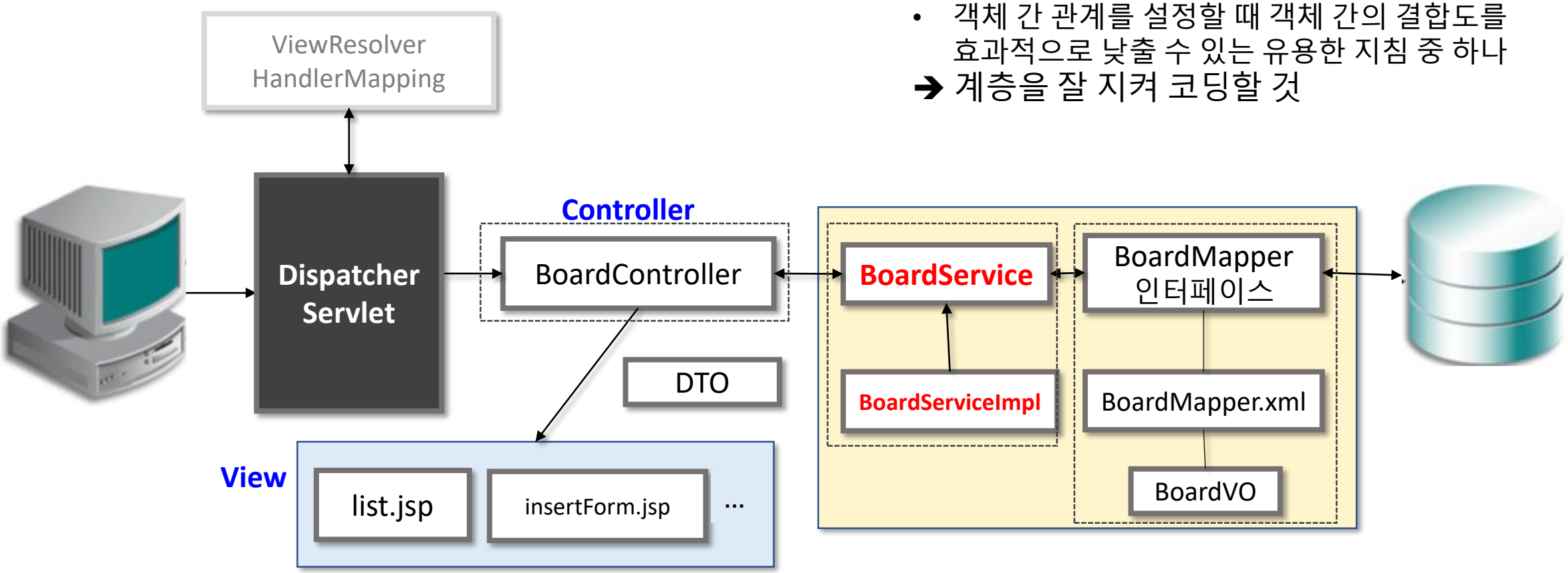
동의과학대학교
컴퓨터정보과
김진숙

구성도

[디미터 법칙(Law of Demeter)]

- 객체 구조의 경로를 따라 멀리 떨어져 있는 낯선 객체에 메시지를 보내는 설계는 피하라는 것
- 객체 간 관계를 설정할 때 객체 간의 결합도를 효과적으로 낮출 수 있는 유용한 지침 중 하나

➔ 계층을 잘 지켜 코딩할 것

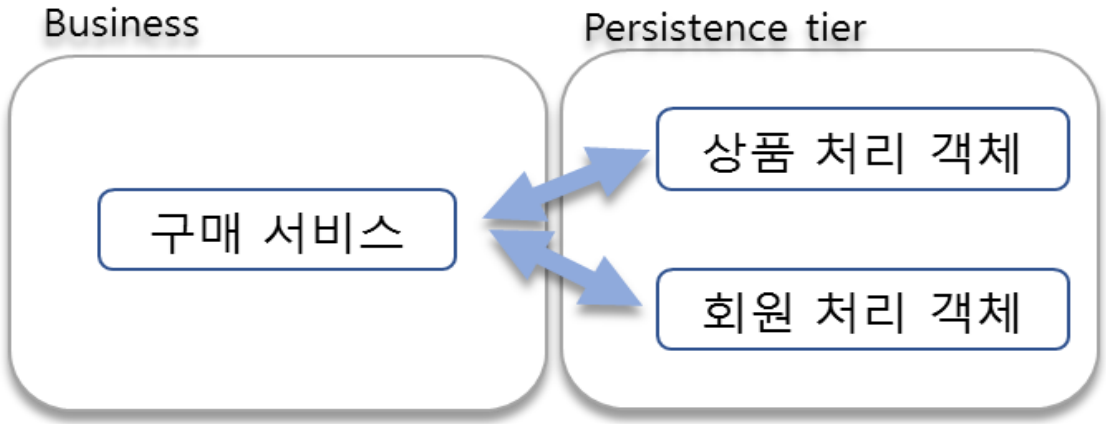


구현 순서

4. 화면 구현 3. 컨트롤러 계층 2. 서비스 계층 1. 영속 계층

비즈니스(서비스) 계층 처리

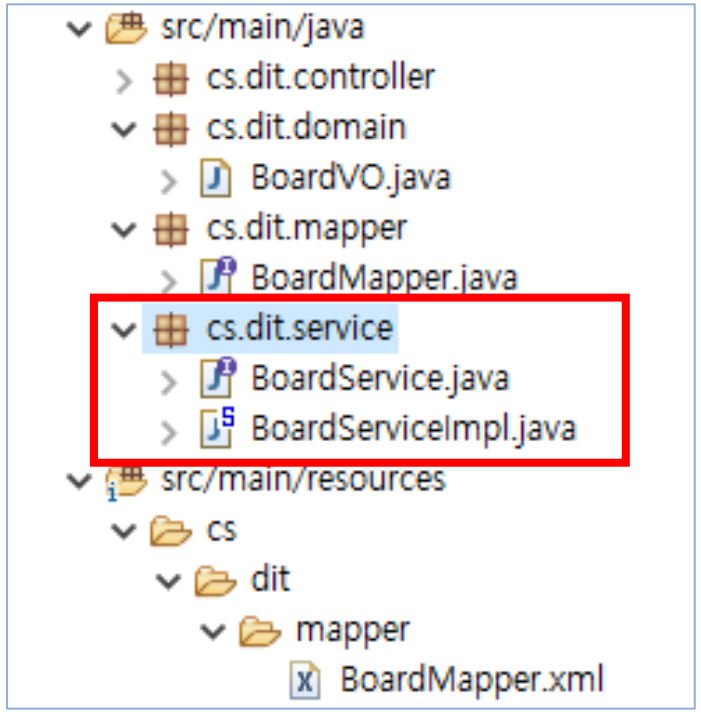
- 고객의 **요구사항**을 반영하는 계층
- 영속 계층과 프레젠테이션 계층의 중간 다리 역할
- **영속 계층**은 **DB 기준**으로 설계를 나누어 구현
- **비즈니스 계층**은 **업무의 로직 단위**로 설계
  - 트랜잭션의 단위
- 여러 개의 Mapper나 DAO를 사용하는 경우가 존재함
- xxxService의 형태로 작성
- 비즈니스 용어를 사용(register, modify, remove, get, getList 등)



영속계층과 중복되는 것처럼 보일 수 있으나 확장성을 고려하면 이 단계를 만드는 것이 유용하다.

서비스 패키지 설정

- 인터페이스와 클래스를 설정하고, root-context.xml에 등록



root-context.xml의 일부

```
<context:component-scan base-package="cs.dit.service"/>
```

- 느슨한 연결(결합도가 낮은)인 인터페이스로 설계하는 이유
  - ✓ 인터페이스를 사용하면 실제 구현 객체를 알지 못해도 코딩이 가능하여 의존성 주입과 더불어 유연한 코딩이 가능해짐

servlet-context.xml 은 웹과 관련된 내용만 설정하고 나머지는 root-context.xml 에서 다룸

게시물 조회

BoardService.java

```
public interface BoardService {

  public List<BoardVO> getList();

}
```

BoardServiceImpl.java

```
@Log4j
@Service
@RequiredArgsConstructor
public class BoardServiceImpl implements BoardService {
  //spring 4.3 이상에서 자동 처리
  private final BoardMapper mapper;

  @Override
  public List<BoardVO> getList() {
      log.info("getList ----------");
      return mapper.getList();
  }
}
```

final 필드 자동 주입 방법

- @Service는 스프링에 빈으로 등록되는 서비스객체의 어노테이션
- XML의 경우에는 <component-scan>에서 조사하는 패키지의 클래스들 중에 @Service가 있는 클래스의 인스턴스를 스프링의 빈으로 설정(root-context.xml)

[참고]의존성 주입 방법

1. 필드 주입

```
 8 @Component
 9 @ToString
10 public class Restaurant {
11
12     @Autowired
13     private Chef chef;
14
15 }
```

2. Setter 주입

```
 9 @Component
10 @ToString
11 public class Restaurant {
12
13     @Setter(onMethod_ = {@Autowired})
14     private Chef chef;
15
16 }
```

3. 생성자 주입

```
 8 @Component
 9 @ToString
10 @AllArgsConstructor
11 public class Restaurant {
12
13     private Chef chef;
14
15 }
```

4. Final 필드 자동 주입

```
 5 import lombok.RequiredArgsConstructor;
 6 import lombok.ToString;
 7
 8 @Component
 9 @ToString
10 @RequiredArgsConstructor
11 public class Restaurant {
12
13     private final Chef chef;
14
15 }
```

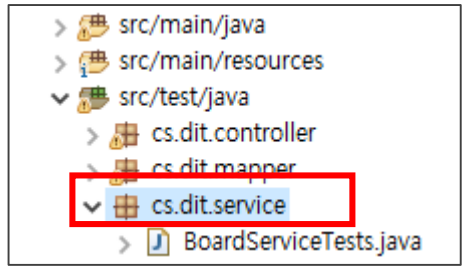
게시물 조회 테스트

BoardServiceTests.java

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardMapperTests {
	@Autowired
	private BoardService service;

	@Test
	public void testGetList() {
		log.info("GetList-------------------------");

		//service.getList().forEach(board-> log.info(board)); //람다식 표현
		List<BoardVO> list = service.getList();
		for(BoardVO board : list) {
			log.info(board);
		}
	}
}
```




```
INFO : jdbc.resultsettable -
|----|------------|------------|----------|----------------------|----------------------|
|bno |title       |content     |writer    |regdate               |updatedate            |
|----|------------|------------|----------|----------------------|----------------------|
|101 |Test 테스트   |Content     |user01    |2021-11-24 15:45:24.0 |2021-11-24 15:45:24.0 |
|62  |방가방가       |진짜 졸린다!!!!  |춘향이      |2021-11-24 00:54:47.0 |2021-12-09 17:18:10.0 |
|181 |테스트        |테스트테스트     |테스트      |2021-11-27 01:08:46.0 |2021-11-27 01:08:46.0 |
|201 |asdf        |asdfasdf    |asdfasd   |2021-11-27 01:21:02.0 |2021-11-27 01:21:02.0 |
|263 |반가워요       |정말          |성춘향      |2021-12-03 13:58:24.0 |2021-12-03 13:58:24.0 |
|281 |테스트        |합니다         |gildong   |2021-12-06 00:26:52.0 |2021-12-06 00:26:52.0 |
|321 |테스트 코드에서 입력 |테스트 코드에서 입력 |홍길동      |2021-12-09 14:37:09.0 |2021-12-09 14:37:09
```

참고

- 람다식 함수(익명함수 : Anonymous function)
  - 람다의 근간은 수학과 기초 컴퓨터과학 분야에서의 람다 대수
  - 람다 대수는 수학에서 사용하는 함수를 보다 단순하게 표현하는 방법
  - 함수명이 필요 없음
  - 함수를 값으로 사용 할 수도 있으며 파라메터로 전달 및 변수에 대입 하기와 같은 연산들이 가능
- 람다의 표현식
  - 매개변수 화살표(->) 함수 몸체로 사용
  - 함수 몸체가 단일 실행문이면 { } 생략가능
  - 함수 몸체가 return으로만 구성되어 있는 경우에는 { } 생략 불가능
  - ( ) -> { }
  - ( ) -> 1
  - ( ) -> { return 1;}
  - (int x) -> x+1

service.getList().forEach(board-> log.info(board));

BoardService 인터페이스

```
package cs.dit.service;
import java.util.List;
import cs.dit.domain.BoardVO;

public interface BoardService {
	//Service의 메서드를 설계할 때는 비즈니스에서 사용되는  로직명 사용

	public List<BoardVO> getList();

	public BoardVO get(Long bno);

	public int register(BoardVO board);

	public int modify(BoardVO board);

	public int remove(Long bno);

}
```

BoardServiceImpl

```
package cs.dit.service;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import cs.dit.domain.BoardVO;
import cs.dit.mapper.BoardMapper;
import lombok.extern.log4j.Log4j;

@Log4j
@Service
@RequiredArgsConstructor
public class BoardServiceImpl implements BoardService {

	private final BoardMapper mapper;

	@Override
	public List<BoardVO> getList() {

		return mapper.getList();
	}
```

```
	@Override
	public int register(BoardVO board) {

		return mapper.insert(board);
	}

	@Override
	public BoardVO get(Long bno) {

		return mapper.read(bno);
	}

	@Override
	public int modify(BoardVO board) {

		return mapper.update(board);
	}

	@Override
	public int remove(Long bno) {

		return mapper.delete(bno);
	}
}
```

BoardServiceTests

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardServiceTests {

	@Autowired
	private BoardService service;

	@Test
	public void testGetList() {
		log.info("GetList--------------------------");

		//service.getList().forEach(board-> log.info(board));
		List<BoardVO> list = service.getList();
		for(BoardVO board : list) {
			log.info(board);
		}
	}
```

```
	@Test
	public void testGet() {
		log.info("GET--------------------------");

		BoardVO board = service.get(32L);
		log.info(board);
	}

	@Test
	public void testRemove() {
		log.info("Remove--------------------------");

		log.info("remove result : " + service.remove(4L));
	}

	@Test
	public void testModify() {
		log.info("Modify--------------------------");
		BoardVO board = service.get(62L);

		board.setContent("진짜 졸린다!!!!");
		log.info("Modify result : " + service.modify(board));
	}
}
```

서비스 계층의 구현과 테스트 진행

- 원칙적으로는 서비스 계층 역시 Mapper나 DAO와 같이 별도로 테스트를 진행하는 것이 바람직
- 하나의 Mapper나 DAO를 이용하는 경우에는 테스트를 생략하는 경우도 많은 편