전자정부 표준프레임워크

실행환경 (공통기반) 실습

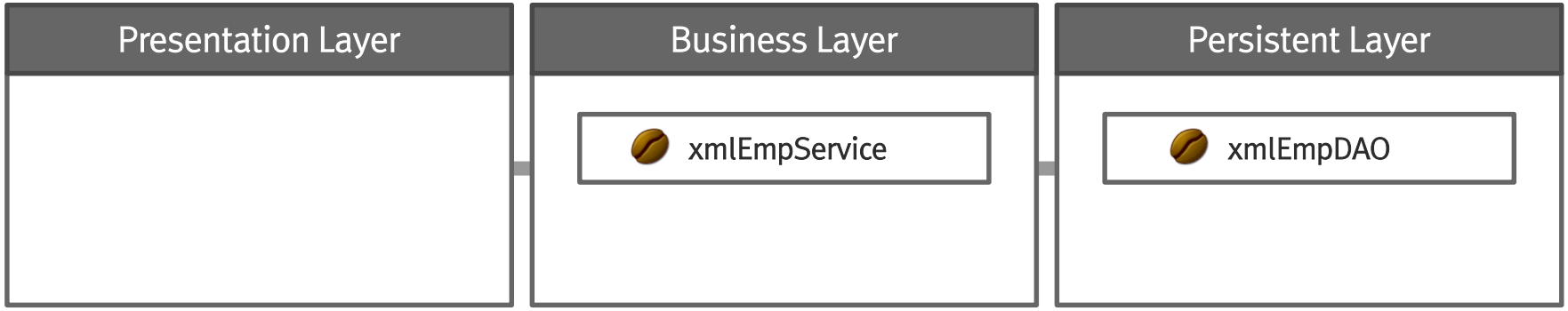
eGovFrame

Contents

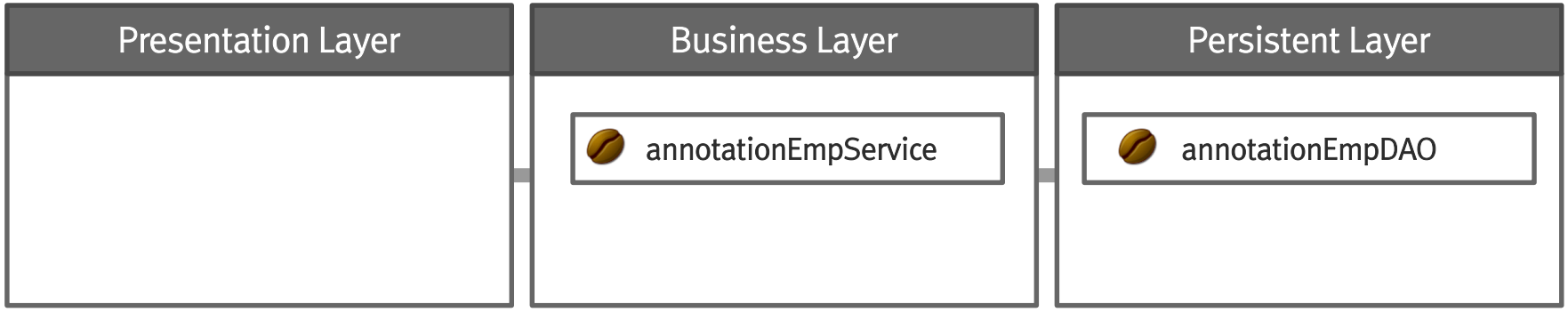
1. LAB 201-IOC 실습(1)
: XML 설정 방식의 Spring bean 서비스 작성

eGovFrame

- XML 설정방식에서 설정할 beans



- Annotation 설정방식에서 설정할 beans



❑ 1. Interface 작성

- /lab201-ioc/src/main/java/egovframework/lab/ioc/service/EmpService.java 를 작성한다.

```
public interface EmpService {

    public void insertEmp(EmpVO empVO) throws Exception;

    public void updateEmp(EmpVO empVO) throws Exception;

    public void deleteEmp(EmpVO empVO) throws Exception;

    public EmpVO selectEmp(EmpVO empVO) throws Exception;

    public List<EmpVO> selectEmpList() throws Exception;

}
```

- Ctrl + Shift + O (source 〉 Organize Imports) 를 수행하여 자동 import 를 수행한다

  cf.) 아직 EmpVO 를 작성하지 않아 컴파일 에러 상태일 것이다.

❑ 2. VO 작성(1/2)

(현재 실습과정에서 사용하는 DAO 에서 DB 구현 없이 자바 class 영역에 데이터를 임시 관리할 때 데이터 비교를 위해 Comparable 구현 (Generic 스타일로))

- /lab201-ioc/src/main/java/egovframework/lab/ioc/service/EmpVO.java 를 작성한다.

```
public class EmpVO implements Serializable {

        private int empNo;

        private String empName;

        private String job;
}
```

- Getter, setter 생성하기 : 마우스 우클릭 〉 Source 〉 Generate Getters and Setters 를 실행하여 Select All 한 다음 OK 실행
- Serializable을 implements함. -〉 Ctrl + Shift + O (자동 import)
- EmpVO 의 마커바 상에서 quick fix 로 제공되는 기능 중 Add generated serial version ID 추가 하기를 권고함.

2. VO 작성(2/2)

- Comparable 를 implements 토록 추가 – 여기서는 generic 스타일로 Comparable<EmpVO> 로 한정
- compareTo 메서드 추가 – 여기서는 EmpVO 의 empNo 속성의 크기를 비교하여 판단토록 하였음.

```
public class EmpVO implements Serializable, Comparable<EmpVO> {

    . . .
    public int compareTo(EmpVO o) {
        if (empNo > o.getEmpNo()) {
            return 1;
        } else if (empNo < o.getEmpNo()) {
            return -1;
        } else {
            return 0;
        }
    }
}
```

- EmpService 소스에서 Ctrl+Shift+O 하여 EmpVO 자동 import

3. Service Impl작성 (1/2)

- /lab201-ioc/src/main/java/egovframework/lab/ioc/service/impl/XmlEmpServiceImpl.java 를 작성한다.

```
public class XmlEmpServiceImpl implements EmpService {

	private XmlEmpDAO empDAO;

	public void setEmpDAO(XmlEmpDAO empDAO) {
		this.empDAO = empDAO;
	}

	public void insertEmp(EmpVO empVO) throws Exception {
		empDAO.insertEmp(empVO);
	}

	public void updateEmp(EmpVO empVO) throws Exception {
		empDAO.updateEmp(empVO);
	}

	public void deleteEmp(EmpVO empVO) throws Exception {
		empDAO.deleteEmp(empVO);
	}

	public EmpVO selectEmp(EmpVO empVO) throws Exception {
		return empDAO.selectEmp(empVO);
	}

	public List<EmpVO> selectEmpList() throws Exception {
		return empDAO.selectEmpList();
	}
}
```

3. Service Impl작성 (2/2)

- 위에서 dependency 객체로 XmlEmpDAO 를 setEmpDAO 메서드를 통해 Container 로부터 주입받아 동작하게 되며 EmpService 자체에 복잡한 비지니스 로직이 필요치 않은 경우로 DAO 에 단순 CRUD 기능을 위임해 처리하고 있음을 확인할 수 있음
- 목록조회 메서드에서 확인할 수 있듯이 JDK 1.5 이상의 Generic 스타일로 구현하는 것을 권고함

4. DAO 작성(1/3)

- /lab201-ioc/src/main/java/egovframework/lab/ioc/service/impl/XmlEmpDAO.java 를 작성한다.

```
public class XmlEmpDAO {

	static List<EmpVO> list;

	static {
		list = new ArrayList<EmpVO>();
		EmpVO empVO;
		for (int i = 1; i <= 100; i++) {
			empVO = new EmpVO();
			empVO.setEmpNo(i);
			empVO.setEmpName("EmpName " + i);
			empVO.setJob("SALESMAN");
			list.add(empVO);
		}
	}

	public void insertEmp(EmpVO empVO) throws Exception {
		list.add(empVO);
		Collections.sort(list);
	}

	. . . (계속)

}
```

4. DAO 작성(2/3)

```
	. . .(이어서)
	public void updateEmp(EmpVO empVO) throws Exception {
		int index = Collections.binarySearch(list, empVO);
		// 해당 데이터가 없는 경우 여기서는 ArrayIndexOutOfBoundsException 발생할 것임
		EmpVO orgEmpVO = list.get(index);

		orgEmpVO.setEmpName(empVO.getEmpName());
		orgEmpVO.setJob(empVO.getJob());
	}

	public void deleteEmp(EmpVO empVO) throws Exception {
		list.remove(Collections.binarySearch(list, empVO));
		Collections.sort(list);
	}

	public EmpVO selectEmp(EmpVO empVO) throws Exception {
		int index = Collections.binarySearch(list, empVO);

		// list search 결과 해당값을 찾을 수 없으면 음수값
		// (-(insertion point) - 1) 이 되돌려짐
		return index < 0 ? null : list.get(index);
	}

	public List<EmpVO> selectEmpList() throws Exception {
		return list;
	}

}
```

❑ 4. DAO 작성(3/3)

- 현 실습과정의 위 DAO 에서는 DB 연동/구현 없이 static 영역에 100 개의 EmpVO 에 대한 리스트를 생성해 두고 insert/update/delete 시에 static 하게 관리하고 있는 데이터에 대해 추가/변경/삭제가 일어나도록 간략히 구현한 예임.
- DB 가 아니므로 duplcated key 체크 등 번잡한 기능은 고려치 않았고, 데이터의 변경시에는 항상 sorting 을 새로 하여 select 시 binarySearch 로 빨리 찾을 수 있도록 하였음(EmpVO 는 Comparable 을 구현하였음).
- 목록조회는 검색조건 없이 전체 데이터를 return 하는 것으로 작성하였음.

❑ 5. XML 설정 파일 작성

- /lab201-ioc/src/test/resources/META-INF/spring/context-emp.xml 를 작성한다.

```
<!-- xml 형식 bean 정의 -->

<bean id="xmlEmpService" class="egovframework.lab.ioc.service.impl.XmlEmpServiceImpl">
<property name="empDAO" ref="xmlEmpDAO" />
</bean>

<bean id="xmlEmpDAO" class="egovframework.lab.ioc.service.impl.XmlEmpDAO" />
```

- xmlEmpService 와 xmlEmpDAO 에 대한 bean 설정을 확인할 수 있으며 xmlEmpService 의 property 설정 요소 (setter injection 방식) 로 xmlEmpDAO 를 연결하고 있음을 확인 가능.
- Spring IDE 기반의 bean 설정파일에 대한 다양한 code assist 가 지원되므로 대상 클래스에 Ctrl + 마우스 오버 --> 클릭시 대상 소스 열림 또는 class="" 속성, property name="" 속성 내에서 [일부typing] Ctrl + Space 등을 사용하여 자동 완성되는 코드를 사용하는 것이 오타 가능성을 줄일 수 있음.

❑ 6. Testcase 작성(1/5)

- /lab201-ioc/src/test/java/egovframework/lab/ioc/service/EmpServiceTest.java 를 작성한다.

```
package egovframework.lab.ioc.service;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertNull;
import static org.junit.Assert.assertTrue;

import java.util.List;

import javax.annotation.Resource;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {
        "classpath*:META-INF/spring/context-common.xml",
        "classpath*:META-INF/spring/context-emp.xml"
        // ,"classpath*:META-INF/spring/context-postprocessor.xml" // 이 주석을 풀고 테스트 시
        // annotationEmpService 에 대해서는 delete 메서드에 @Debug 를 설정하였으므로 trace 로그가 출력될 것임.
})
public class EmpServiceTest {

. . . (계속)
```

❑ 6. Testcase 작성(2/5)

```
. . .(이어서)
// xml 형식으로 bean 설정한 경우 – 주석을 변경해 가며 xml, annotation 에 대해 테스트 할것
@Resource(name = "xmlEmpService")
EmpService empService;

// annotation 형식으로 bean 설정한 경우
// @Resource(name = "annotationEmpService")
// EmpService empService;

public EmpVO makeVO() {
        // DAO 확인 – static 하게 관리하는 100 개 기본 데이터 있음
        return makeVO(101);
}

public EmpVO makeVO(int empNo) {
        EmpVO vo = new EmpVO();
        vo.setEmpNo(empNo);
        vo.setEmpName("홍길동" + empNo);
        vo.setJob("개발자");
        return vo;
}

public void checkResult(EmpVO vo, EmpVO resultVO) {
        assertNotNull(resultVO);
        assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
        assertEquals(vo.getEmpName(), resultVO.getEmpName());
        assertEquals(vo.getJob(), resultVO.getJob());
}
. . . (계속)
```

6. Testcase 작성(3/5)

```
. . .(이어서)
@Test
public void testInsertEmp() throws Exception {
        EmpVO vo = makeVO();

        // insert
        empService.insertEmp(vo);
        // select
        EmpVO resultVO = empService.selectEmp(vo);
        // check
        checkResult(vo, resultVO);
}

@Test
public void testUpdateEmp() throws Exception {
        EmpVO vo = makeVO(102);

        // insert
        empService.insertEmp(vo);
        // data change
        vo.setEmpName("홍길순");
        vo.setJob("설계자");

        // update
        empService.updateEmp(vo);
        // select
        EmpVO resultVO = empService.selectEmp(vo);
        // check
        checkResult(vo, resultVO);
}
. . .(계속)
```

6. Testcase 작성(4/5)

```
...(이어서)
    @Test
    public void testDeleteEmp() throws Exception {
            EmpVO vo = makeVO(103);
            // insert
            empService.insertEmp(vo);

            // delete
            empService.deleteEmp(vo);
            // select
            EmpVO resultVO = empService.selectEmp(vo);
            // null 이어야 함
            assertNull(resultVO);
    }

    @Test
    public void testSelectEmpList() throws Exception {

            // select list
            List<EmpVO> resultList = empService.selectEmpList();
            // check
            int firstListSize = resultList.size();
            // DAO 에서 Emp 데이터를 관리할 때 항상 sorted 된 상태임
            assertEquals(1, resultList.get(0).getEmpNo());

            // delete first data
            EmpVO empVO = new EmpVO();
            empVO.setEmpNo(1);
            ...(계속)
```

❑ 6. Testcase 작성(5/5)

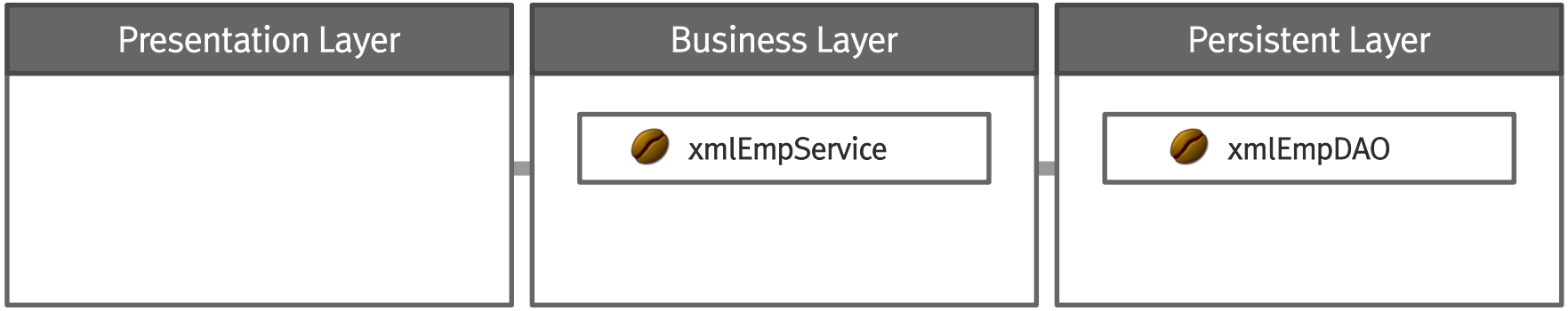
```
        . . .(이어서)

        empService.deleteEmp(empVO);

        // select List again
        resultList = empService.selectEmpList();

        assertEquals(firstListSize - 1, resultList.size());
        // DAO 에서 Emp 데이터를 관리할 때 항상 sorted 된 상태임
        assertEquals(2, resultList.get(0).getEmpNo());
        assertEquals("EmpName 2", resultList.get(0).getEmpName());
        assertEquals("SALESMAN", resultList.get(0).getJob());

        }
}
```

- Spring 연동을 위해 제공하는 @RunWith(SpringJUnit4ClassRunner.class), @ContextConfiguration (...) 설정에 유의한다. 테스트에 필요한 Spring Bean 설정 파일만으로 제한하는 것이 바람직함.
- 테스트에 필요한 Spring Bean 들은 annotation 형태(여기서는 @Resource)로 injection 하여 사용한다.
- JUnit 4.4 의 **Assert** 관련 기능은 Ctrl+Shift+O 로 **자동 import 되지 않음**. static import 사용해야 함. --〉 에러로 표시되는 asssertXX 사용 위치에 마우스 오버 하면 Add static imports ... 와 같은 quick fix 가 나타나 활용 가능함.

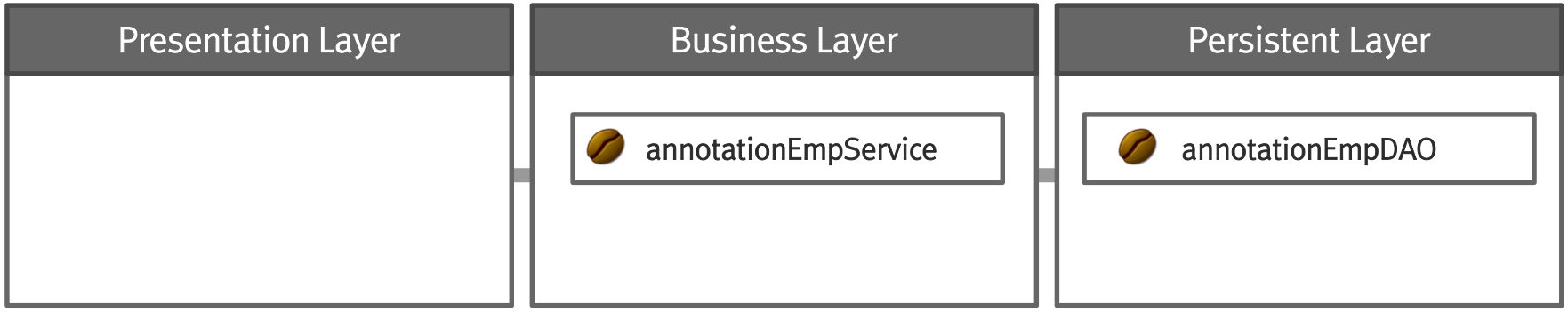
2. LAB 201-IOC 실습(2)
: Annotation 설정 방식의 Spring bean 서비스 작성

eGovFrame

- XML 설정방식에서 설정할 beans



- Annotation 설정방식에서 설정할 beans



2. Annotation 설정 방식의 Spring bean 서비스 작성

❑ 1. 동일한 Interface

- EmpService

❑ 2. 동일한 VO

- EmpVO

❑ 3. Annotation 을 적용한 Impl (1/2)

- /lab201-ioc/src/main/java/egovframework/lab/ioc/service/impl/AnnotationEmpServiceImpl.java 를 작성한다.

… (계속)

2. Annotation 설정 방식의 Spring bean 서비스 작성

…(이어서)

```
@Service("annotationEmpService")
public class AnnotationEmpServiceImpl implements EmpService {

@Resource(name = "annotationEmpDAO")
private AnnotationEmpDAO empDAO;

	public void insertEmp(EmpVO empVO) throws Exception {
		empDAO.insertEmp(empVO);
	}

	public void updateEmp(EmpVO empVO) throws Exception {
		empDAO.updateEmp(empVO);
	}

	public void deleteEmp(EmpVO empVO) throws Exception {
		empDAO.deleteEmp(empVO);
	}

	public EmpVO selectEmp(EmpVO empVO) throws Exception {
		return empDAO.selectEmp(empVO);
	}

	public List<EmpVO> selectEmpList() throws Exception {
		return empDAO.selectEmpList();
	}
}
```

- @Service 스테레오 타입 Annotation 을 사용하여 bean 설정하였음.
- @Resource (JSR250 표준) Annotation 을 사용하여 Dependency Bean(여기서는 AnnotationEmpDAO) 를 injection 하였음.
- 기타 CRUD 관련 비지니스 메서드는 동일함.

❑ 4. Annotation을 적용한 DAO (1/3)

```
@Repository("annotationEmpDAO")
public class AnnotationEmpDAO {

	static List<EmpVO> list;

	static {
		list = new ArrayList<EmpVO>();
		EmpVO empVO;
		for (int i = 1; i <= 100; i++) {
			empVO = new EmpVO();
			empVO.setEmpNo(i);
			empVO.setEmpName("EmpName " + i);
			empVO.setJob("SALESMAN");
			list.add(empVO);
		}
	}

	public void insertEmp(EmpVO empVO) throws Exception {
		list.add(empVO);
		Collections.sort(list);
	}

	... (계속)
```

❑ 4. Annotation을 적용한 DAO(2/3)

```
        ... (이어서)
        public void updateEmp(EmpVO empVO) throws Exception {
                int index = Collections.binarySearch(list, empVO);
                // 해당 데이터가 없는 경우 여기서는 ArrayIndexOutOfBoundsException 발생할 것임
                EmpVO orgEmpVO = list.get(index);

                orgEmpVO.setEmpName(empVO.getEmpName());
                orgEmpVO.setJob(empVO.getJob());
        }

        public void deleteEmp(EmpVO empVO) throws Exception {
                list.remove(Collections.binarySearch(list, empVO));
                Collections.sort(list);
        }

        public EmpVO selectEmp(EmpVO empVO) throws Exception {
                int index = Collections.binarySearch(list, empVO);

                // list search 결과 해당값을 찾을 수 없으면 음수값
                // (-(insertion point) - 1) 이 되돌려짐
                return index < 0 ? null : list.get(index);
        }

        public List<EmpVO> selectEmpList() throws Exception {
                return list;
        }

}
```

❑ 4. Annotation을 적용한 DAO(3/3)

- xml 설정 방식의 예와 마찬가지로 DB 연동 없이 테스트를 위한 static 한 내부 데이터를 관리하며 CRUD 하는 예임
- @Repository 스테레오 타입 Annotation 을 사용하여 bean 설정 하였음. (DAO 인 경우)

❑ 5. common 설정 파일 – component scan

- /lab201-ioc/src/test/resources/META-INF/spring/context-common.xml 를 작성한다.

```
<!-- annotation 형식 bean 정의한 것에 대해 자동적으로 scan 하여 등록함. 여기서는 AnnotationEmpServiceImpl, AnnotationEmpDAO -->
<context:component-scan base-package="egovframework" />
```

❑ 6. Testcase 작성 (기존 Testcase 에서 DI 하는 서비스만 변경)

- /lab201-ioc/src/test/java/egovframework/lab/ioc/service/EmpServiceTest.java 를 작성한다. (이미 작성하

```
// annotation 형식으로 bean 설정한 경우
@Resource(name = "annotationEmpService")
EmpService empService;
```

- annotation 형식으로 설정한 annotationEmpService 를 테스트 대상 서비스로 사용토록 주석 변경하였음.

3. LAB 202-AOP 실습
: XML 설정 방식의 AOP 테스트 서비스 작성

eGovFrame

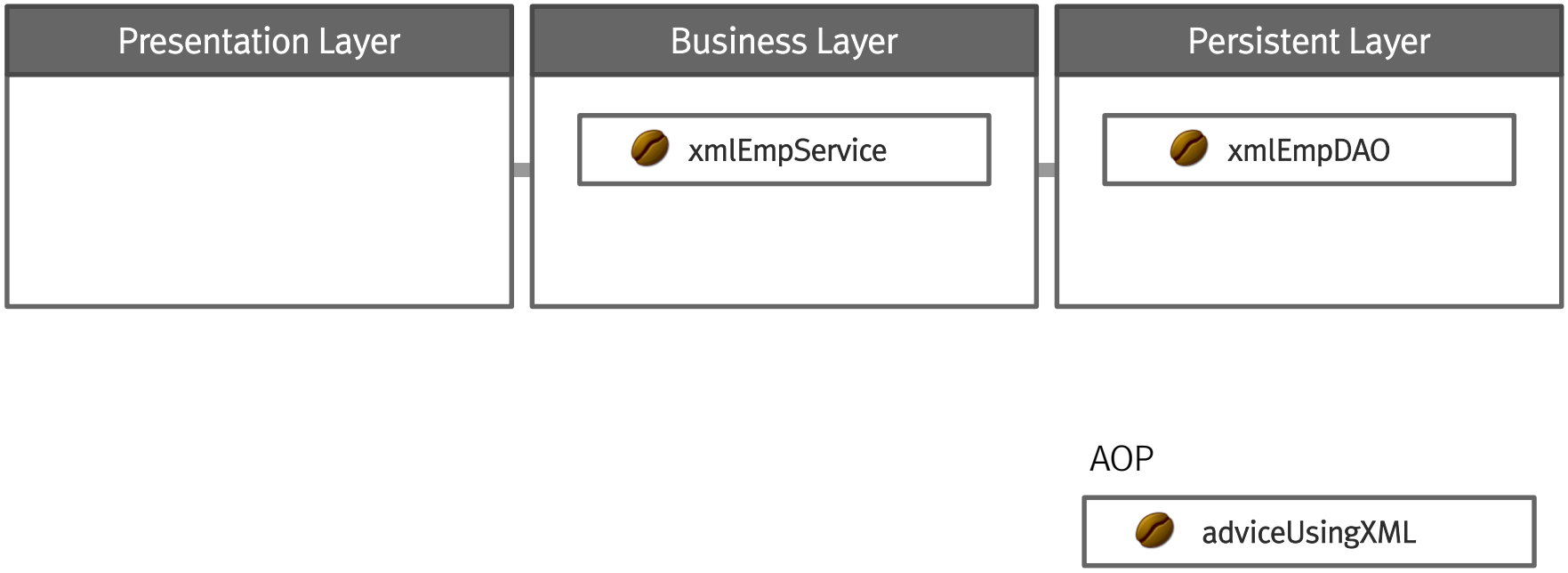
❑ 실습을 통해 AOP에 대하여 살펴본다.

❑ 실습 순서

- Lab 202-aop 프로젝트

  XML 설정 방식의 AOP 테스트 서비스 작성

- lab202-aop프로젝트의 beans



1. XML 설정 방식의 AOP 테스트 서비스 작성

❑ 1. Interface 작성

- /lab202-aop/src/main/java/egovframework/lab/aop/service/EmpService.java 를 확인한다.

(lab201-ioc 와 동일함)

❑ 2. VO 작성

- /lab202-aop/src/main/java/egovframework/lab/aop/service/EmpVO.java 를 확인한다.

(lab201-ioc 의 VO 와 동일함)

❑ 3. Impl 작성

- /lab202-aop/src/main/java/egovframework/lab/aop/service/impl/XmlEmpServiceImpl.java 를 확인한다.

```
. . .
public EmpVO selectEmp(EmpVO empVO) throws Exception {
        EmpVO resultVO;
        resultVO = empDAO.selectEmp(empVO);

        if(resultVO == null) {
            throw new Exception("no data found!");
        }

        return resultVO;
    }
. . .
```

- lab201-ioc 와 대부분 동일하며 selectEmp 메서드에서 해당 데이터가 없는 경우 exception 을 throw 하도록 비지니스 로직을 추가하였음.

❑ 4. DAO 작성

- /lab202-aop/src/main/java/egovframework/lab/aop/service/impl/XmlEmpDAO.java 를 확인한다. (lab201-ioc 의 DAO 와 동일함.)

❑ 5. xml 설정 파일 작성

- /lab202-aop/src/test/resources/META-INF/spring/context-emp.xml 를 확인한다. (lab201-ioc 의 설정과 동일함.)

6. Advice 작성 (1/3)

- /lab202-aop/src/main/java/egovframework/lab/aop/xml/AdviceUsingXML.java 를 작성한다.

```
public class AdviceUsingXML {

    private final Log sysoutLogger = LogFactory.getLog(AdviceUsingXML.class);

    public void beforeTargetMethod(JoinPoint thisJoinPoint) {
        sysoutLogger.debug("\nAdviceUsingXML.beforeTargetMethod executed.");

        @SuppressWarnings("unchecked")
        Class clazz = thisJoinPoint.getTarget().getClass();
        String className = thisJoinPoint.getTarget().getClass().getSimpleName();
        String methodName = thisJoinPoint.getSignature().getName();

        // 현재 class, method 정보 및 method arguments 로깅
        StringBuffer buf = new StringBuffer();
        buf.append("\n== AdviceUsingXML.beforeTargetMethod : [" + className
            + "." + methodName + "()] ==");
        Object[] arguments = thisJoinPoint.getArgs();
        int argCount = 0;
        for (Object obj : arguments) {
            buf.append("\n - arg ");
            buf.append(argCount++);
            buf.append(" : ");
            // commons-lang 의 ToStringBuilder 를
            // 통해(reflection 을 이용)한 VO 정보 출력
            buf.append(ToStringBuilder.reflectionToString(obj));
        }

        // 대상 클래스의 logger 를 사용하여 method arguments 로깅하였음.
        Log logger = LogFactory.getLog(clazz);
        if (logger.isDebugEnabled())
            logger.debug(buf.toString());
    }
...(계속)
```

❑ 6. Advice 작성 (2/3)

```
...(이어서)
    public void afterTargetMethod(JoinPoint thisJoinPoint) {
        sysoutLogger.debug("AdviceUsingXML.afterTargetMethod executed.");
    }

    @SuppressWarnings("unchecked")
    public void afterReturningTargetMethod(JoinPoint thisJoinPoint,
            Object retVal) {
        sysoutLogger
            .debug("AdviceUsingXML.afterReturningTargetMethod executed.");

        Class clazz = thisJoinPoint.getTarget().getClass();
        String className = thisJoinPoint.getTarget().getClass().getSimpleName();
        String methodName = thisJoinPoint.getSignature().getName();

        // 현재 class, method 정보 및 method arguments 로깅
        StringBuffer buf = new StringBuffer();
        buf.append("\n== AdviceUsingXML.afterReturningTargetMethod : ["
            + className + "." + methodName + "()] ==");
        buf.append("\n");

        // 결과값이 List 이면 size 와 전체 List 데이터를 풀어 reflection 으로 출력 - 성능상 사용않는 것이 좋음
        if (retVal instanceof List) {
            List resultList = (List) retVal;
            buf.append("resultList size : " + resultList.size() + "\n");
            for (Object oneRow : resultList) {
                buf.append(ToStringBuilder.reflectionToString(oneRow));
                buf.append("\n");
            }
        } else {
            buf.append(ToStringBuilder.reflectionToString(retVal));
        }
...(계속)
```

6. Advice 작성 (3/3)

```
...(이어서)
        // 대상 클래스의 logger 를 사용하여 결과값 로깅하였음.
        Log logger = LogFactory.getLog(clazz);
        if (logger.isDebugEnabled())
            logger.debug(buf.toString());
        // return value 의 변경은 불가함에 유의!
    }

    public void afterThrowingTargetMethod(JoinPoint thisJoinPoint,
            Exception exception) throws Exception {
        sysoutLogger.debug("AdviceUsingXML.afterThrowingTargetMethod executed.");
        sysoutLogger.error("에러가 발생했습니다.", exception);

        // 원본 exception 을 wrapping 하고 user-friendly 한
        // 메시지를 설정하여 새로운 Exception 으로 re-throw
        throw new BizException("에러가 발생했습니다.", exception);
        // 여기서는 간단하게 작성하였지만 일반적으로 messageSource 를 사용한
        // locale 에 따른 다국어 처리 및 egov. exceptionHandler
        // 를 확장한 Biz. (ex. email 공지 등) 기능 적용이 가능함.
    }

    public Object aroundTargetMethod(ProceedingJoinPoint thisJoinPoint)
            throws Throwable {
        sysoutLogger.debug("AdviceUsingXML.aroundTargetMethod start.");
        long time1 = System.currentTimeMillis();
        Object retVal = thisJoinPoint.proceed();

        // Around advice 의 경우 결과값을 변경할 수도 있음!
        // 위의 retVal 을 가공하거나 심지어 전혀 다른 결과값을 대체하여 caller 에 되돌려줄 수 있음
        long time2 = System.currentTimeMillis();
        sysoutLogger.debug("AdviceUsingXML.aroundTargetMethod end. Time("
            + (time2 - time1) + ")");

        return retVal;
    }
}
```

❑ 7. xml 설정 방식의 AOP정의 작성

- /lab202-aop/src/test/resources/META-INF/spring/context-advice-xml.xml 를 작성한다.

```
<!-- TODO [Step 1-7] xml 설정 방식의 AOP 설정 -->
<bean id="adviceUsingXML" class="egovframework.lab.aop.xml.AdviceUsingXML" />

<aop:config>
        <aop:pointcut id="targetMethod"
        expression="execution(* egovframework.lab.aop..Xml*Impl.*(..))" />
        <aop:aspect ref="adviceUsingXML">
                <aop:before pointcut-ref="targetMethod" method="beforeTargetMethod" />
                <aop:after-returning pointcut-ref="targetMethod"
                                method="afterReturningTargetMethod" returning="retVal" />
                <aop:after-throwing pointcut-ref="targetMethod"
                                method="afterThrowingTargetMethod" throwing="exception" />
                <aop:after pointcut-ref="targetMethod" method="afterTargetMethod" />
                <aop:around pointcut-ref="targetMethod" method="aroundTargetMethod" />
        </aop:aspect>
</aop:config>
```

- 별도의 클래스로 작성한 Advice bean 을 등록하고, aop 네임스페이스 빈 설정 태그(aop:config, aop:pointcut, aop:aspect, before/after-returning/after-throwing/after/around 정의 등) 를 사용하여 AOP 설정. 여기서는 모든 비지니스 메서드(Impl 로 끝나는 모든 class 의 모든 메서드)에 대해 다양한 Advice 기능을 동시에 적용하였음에 유의할 것.

❑ 8. Testcase 작성 (1/4)

- /lab202-aop/src/test/java/egovframework/lab/aop/service/EmpServiceTest.java 를 작성한다.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {
"classpath*:META-INF/spring/context-advice-xml.xml",
"classpath*:META-INF/spring/context-aspectj-annotation.xml",
"classpath*:META-INF/spring/context-common.xml",
"classpath*:META-INF/spring/context-emp.xml"})
public class EmpServiceTest {

	// xml 형식으로 bean 설정한 경우 - 주석을 변경해 가며 xml,
	// annotation 에 대해 테스트 할것
	@Resource(name = "xmlEmpService")
	EmpService empService;
	// annotation 형식으로 bean 설정한 경우
	//    @Resource(name = "annotationEmpService")
	//    EmpService empService;

	public EmpVO makeVO() {
		// DAO 확인 - static 하게 관리하는 100 개 기본 데이터 있음
		return makeVO(101);
	}

	public EmpVO makeVO(int empNo) {
		EmpVO vo = new EmpVO();
		vo.setEmpNo(empNo);
		vo.setEmpName("홍길동" + empNo);
		vo.setJob("개발자");
		return vo;
	}

	...(계속)
```

8. Testcase 작성 (2/4)

```
...(이어서)
public void checkResult(EmpVO vo, EmpVO resultVO) {
        assertNotNull(resultVO);
        assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
        assertEquals(vo.getEmpName(), resultVO.getEmpName());
        assertEquals(vo.getJob(), resultVO.getJob());
}

@Test
public void testInsertEmp() throws Exception {
        EmpVO vo = makeVO();
        // insert
        empService.insertEmp(vo);
        // select
        EmpVO resultVO = empService.selectEmp(vo);
        // check
        checkResult(vo, resultVO);
}

@Test
public void testUpdateEmp() throws Exception {
        EmpVO vo = makeVO(102);
        // insert
        empService.insertEmp(vo);
        // data change
        vo.setEmpName("홍길순");
        vo.setJob("설계자");
        // update
        empService.updateEmp(vo);
        // select
        EmpVO resultVO = empService.selectEmp(vo);
        // check
        checkResult(vo, resultVO);
}
...(계속)
```

8. Testcase 작성 (3/4)

```
...(이어서)
@Test
public void testDeleteEmp() throws Exception {
	EmpVO vo = makeVO(103);
	// insert
	empService.insertEmp(vo);
	// delete
	empService.deleteEmp(vo);
	// select
	try {
		@SuppressWarnings("unused")
		EmpVO resultVO = empService.selectEmp(vo);
		fail("Biz Exception 이 발생해야 합니다.");
	} catch (Exception e) {
		assertNotNull(e);
		// aop 를 적용하여 after-throwing 에서 가공한 BizException 으로 넘어올 것임.
		// cf.) ServiceImpl 원본에서는 그냥 Exception 이었음.
		assertTrue(e instanceof BizException);
		assertEquals("에러가 발생했습니다.", e.getMessage());
		assertEquals("no data found!", e.getCause().getMessage());
	}
}

@Test
public void testSelectEmpList() throws Exception {
	// select list
	List<EmpVO> resultList = empService.selectEmpList();
	// check
	int firstListSize = resultList.size();
	// DAO 에서 Emp 데이터를 관리할 때 항상 sorted 된 상태임
	assertEquals(1, resultList.get(0).getEmpNo());

	...(계속)
```

8. Testcase 작성 (4/4)

```
        ...(이어서)

                // delete first data
                EmpVO empVO = new EmpVO();
                empVO.setEmpNo(1);

                empService.deleteEmp(empVO);
                // select List again
                resultList = empService.selectEmpList();
                assertEquals(firstListSize - 1, resultList.size());
                // DAO 에서 Emp 데이터를 관리할 때 항상 sorted 된 상태임
                assertEquals(2, resultList.get(0).getEmpNo());
                assertEquals("EmpName 2", resultList.get(0).getEmpName());
                assertEquals("SALESMAN", resultList.get(0).getJob());
        }
}
```

- lab201-ioc 의 테스트케이스와 유사하며 ContextConfiguration 설정 파일 로딩에 유의할것.
- delete 테스트의 경우 삭제 후 재조회 시 null 인 경우 강제 Exception 처리를 하였고 이를 다시 after-throwing AOP 처리로 BizException 으로 재처리한 것을 테스트 하고 있음.