1. 프레임워크(framework)

■ 프레임워크란?

- 뼈대, 골격이라는 뜻
- 동적인 웹 페이지나, 웹 애플리케이션, 웹 서비스를 개발하기 쉽게 해 주는 구성요소 모음
- API와 비슷하나, 프로그램의 흐름이나 객체의 라이프사이클을 관리함

■ 대표적인 JAVA 프레임워크

- struts, spring, struts2, MyBatis, Hibernate 등

2. ORM 프레임워크

■ ORM이란?

- object-relation mapping의 준말로 객체-관계 맵핑 툴
- ORM 프레임워크에는 대표적으로 hibernate가 있음
- 장점 : 자동화, ER/클래스 다이어그램에 의한 자동 JOIN/Cascade
- 단점 : 기술적 난이도 높음(아키텍쳐 이해, 설계, 트랜젝션, 캐쉬구성 등)

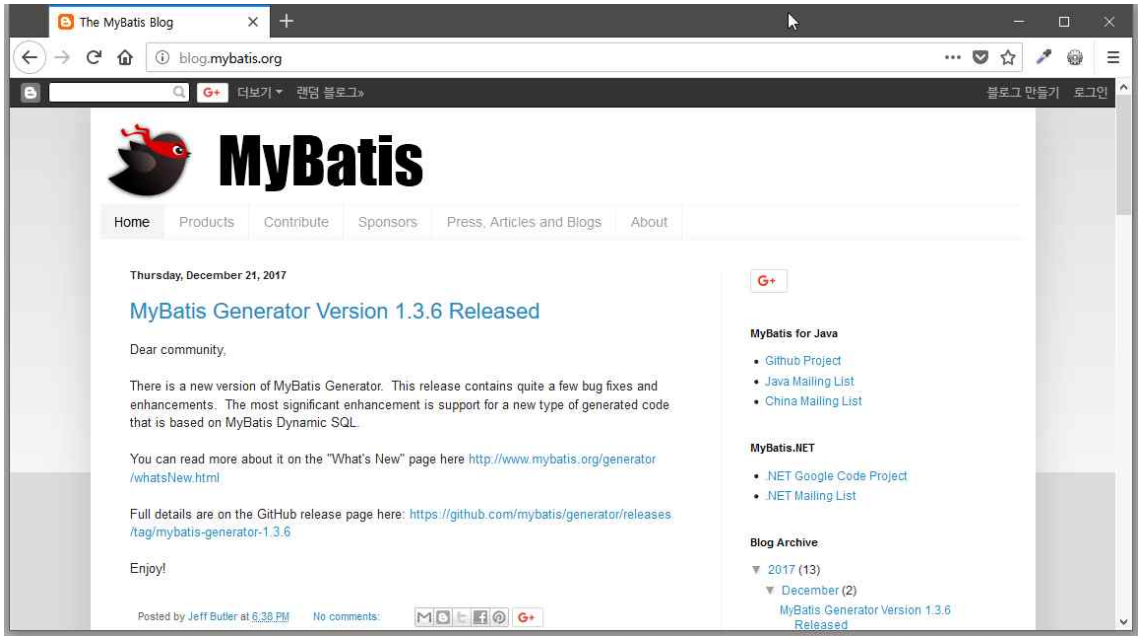
■ SQL Mapping 프레임워크

- SQL 구문 파라미터와 결과를 VO 객체에 맵핑
- 대표적으로 MyBatis가 있음
- 장점 : 편리함, 간결함, 쉬움
- 단점 : 효율이 좋다고 말할 수 없음

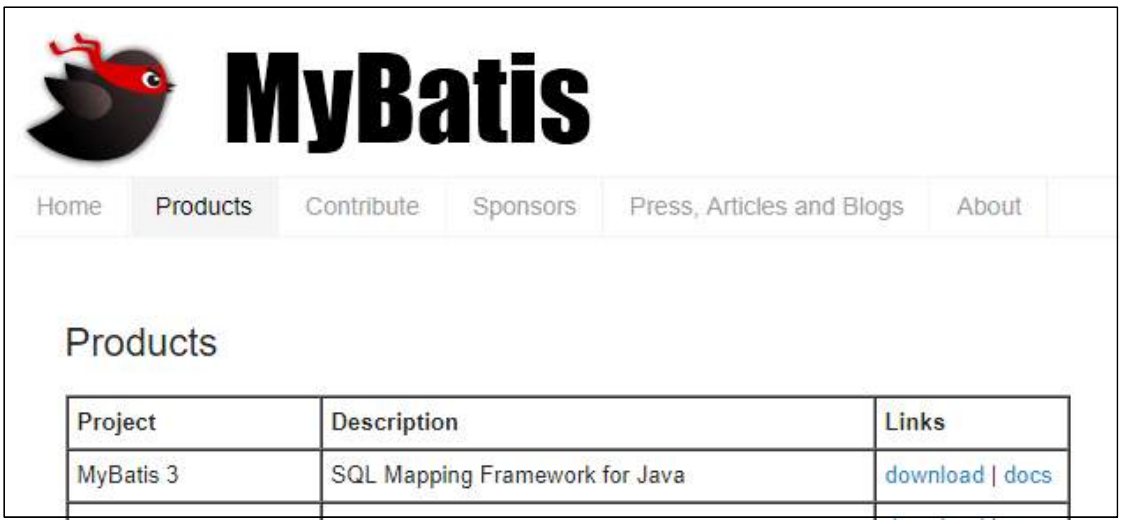
■ 현재는 SQL Mapping 프레임워크인 MyBatis도 ORM으로 인정하고 있음

3. MyBatis의 설치

■ http://blog.mybatis.org/ 접속

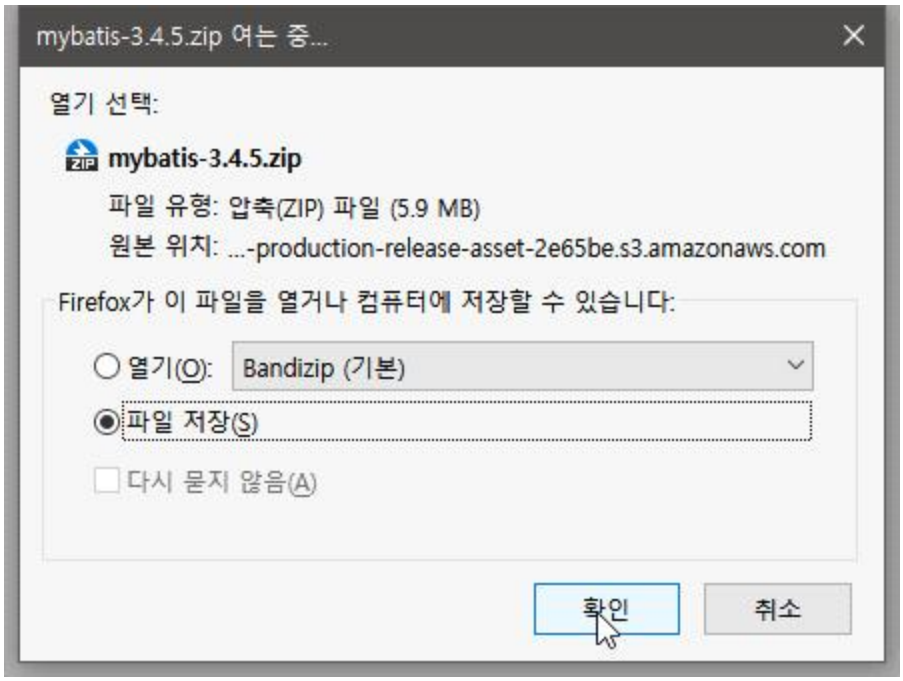


■ 메뉴중 Products 선택 -> MyBatis 3의 download 선택



■ MyBatis 3.4.5 선택

■ 다운로드



4. MyBatis의 사용이유

■ 간단함 : 가장 간단한 퍼시스턴스 프레임워크

> **퍼시스턴스(persistence) 란?**
> 영속, 지속성이란 뜻으로, 메모리상에 로딩되어 있는 데이터는 지속적이거나 영속적이지 못하다. 영속성을 유지하기 위해서는 파일에 저장하거나 데이터베이스에 저장하는 것이다. 그렇기 때문에 퍼시스턴스 계층, 혹은 퍼시스턴스 프레임워크라고 말하면, 데이터베이스관련 계층과 프레임워크라고 이해하면 된다.

■ 생산성

- 퍼시스턴스 계층에서 62%에 달하는 코드의 양을 줄이는 결과
- 필요없는 JDBC 코드를 줄임(sql은 사용)

■ 성 능

- 잘못 작성된 JDBC 코드보다는 성능이 향상됨

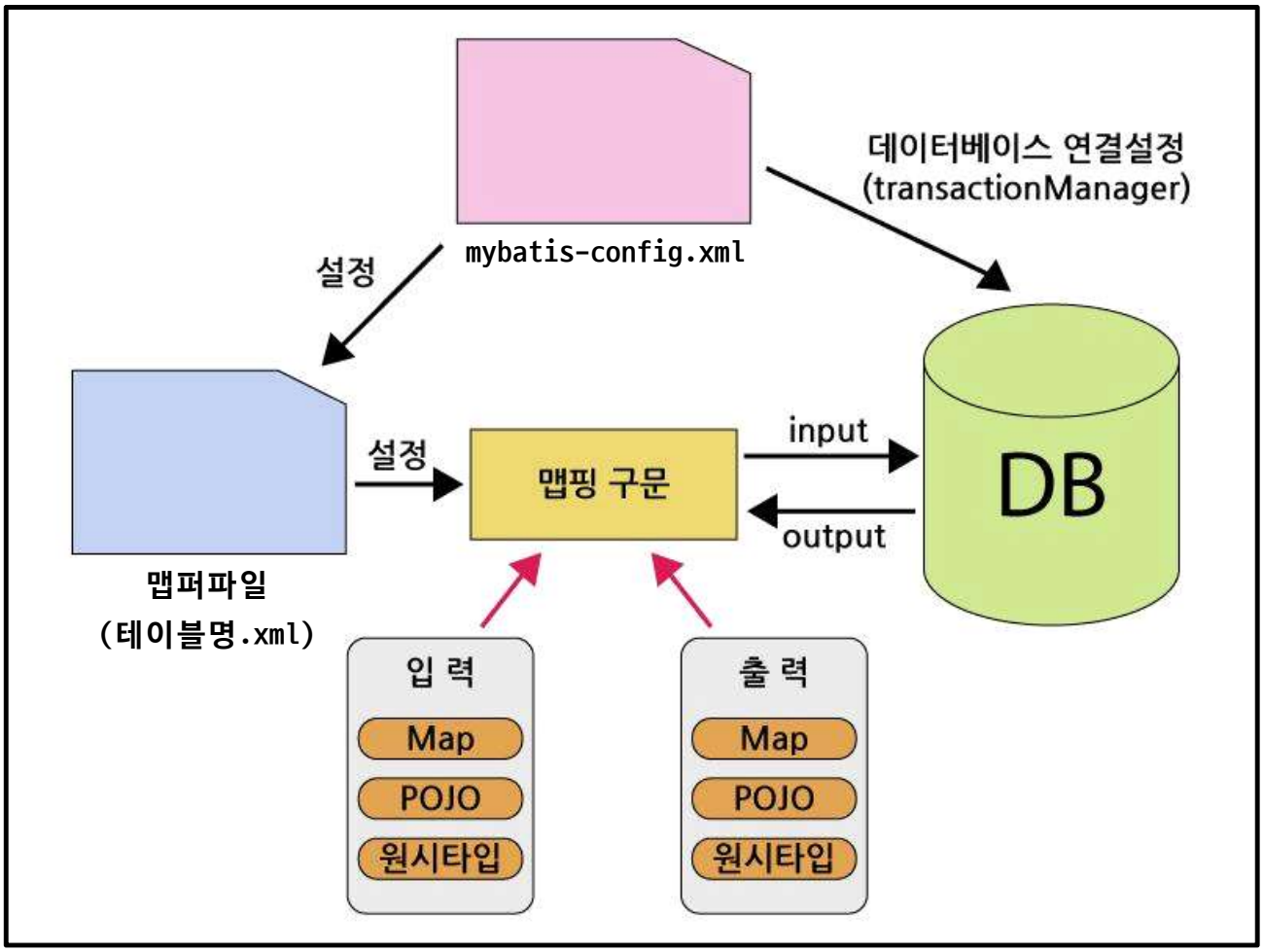
■ 관심사의 분리

- SQL 구문, 커넥션, 결과셋이 같은 코드에 작성되면 관심사가 분리되지 않아 개발이나 유지보수시 하나에 집중하기 힘듦
- 작업의 분배가 가능(sql을 DBA가 작성 가능)

■ 이식성

- myBatis는 어떤 프로그래밍 언어로도 구현 가능

5. MyBatis 개념



■ mybatis-config.xml

- myBatis 설정의 중심
- 데이터베이스 접속에서부터 SqlMap 파일들의 설정이 포함

■ 맵퍼(mapper) 파일

- 맵핑구문이 들어가 있는 파일

6. 기본예제

■ select 예제1

① table 생성

```
CREATE TABLE  "EMPLOYEES"(
        "ID" VARCHAR2(20) NOT NULL,
       "PASSWORD" VARCHAR2(20) NOT NULL,
       "NAME" VARCHAR2(30) NOT NULL,
       "LVL" CHAR(1) NOT NULL,
       "HIREDATE" DATE NOT NULL,
        CONSTRAINT "EMPLOYEES_PK" PRIMARY KEY ("ID")
   )
```

② mybatis-config.xml 작성

mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <!-- db 연결 -->
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC" />
            <dataSource type="POOLED">
                <property name="driver" value="oracle.jdbc.OracleDriver" />
                <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"
/>
                <property name="username" value="timo" />
                <property name="password" value="1111" />
            </dataSource>
        </environment>
    </environments>

    <!-- sql 맵퍼 -->
    <mappers>
        <mapper resource="sqlmapper/employees.xml" />
    </mappers>
```

```
</configuration>
```

③ SqlMap(employees.xml) 작성

employees.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="emp">
     <select id="selectAll" resultType="ex.vo.EmployeesVO">
        SELECT * FROM employees
    </select>
</mapper>
```

④ VO 객체 생성

EmployeesVO.java

```
package ex.vo;

import java.util.Date;

public class EmployeesVO {
    private String id;
    private String password;
    private String name;
    private String lvl;
    private Date hiredate;

    public Date getHiredate() {
        return hiredate;
    }

    public void setHiredate(Date hiredate) {
        this.hiredate = hiredate;
    }
```

```
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getLvl() {
        return lvl;
    }

    public void setLvl(String lvl) {
        this.lvl = lvl;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

⑤ EmployeesClient.java 작성

EmployeesClient.java

```
package ex;
```

```
import java.io.IOException;
import java.io.Reader;
import java.sql.Date;
import java.util.List;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import ex.vo.EmployeesVO;

public class EmployeesClient {

	public static void main(String[] args) {

		//config파일의 경로
		String resource = "config/mybatis-config.xml";
		SqlSession session = null;
		//mybatis-config.xml로딩
		try {
			Reader reader = Resources.getResourceAsReader(resource);
			SqlSessionFactory sqlMapper =
 new SqlSessionFactoryBuilder().build(reader);

			session = sqlMapper.openSession();
			System.out.println("세션 생성 완료!");

		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		List<EmployeesVO> list = session.selectList("emp.selectAll");

		for(EmployeesVO emp : list) {
			System.out.println(emp);
		}
```

```
      }

}
```

■ select 예제2 (검색)

- 아이디가 “admin” 인 사원 검색

① employees.xml 수정

employees.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="emp">
     <select id="selectAll" resultType="ex.vo.EmployeesVO">
        SELECT * FROM employees
    </select>

    <select id="selectById" parameterType="string" resultType="ex.vo.EmployeesVO">
       SELECT * FROM employees WHERE id = #{id}
    </select>
</mapper>
```

② EmployeesClient2.java

EmployeesClient2.java

```
package ex;

import java.io.IOException;
import java.io.Reader;
import java.sql.Date;
import java.util.List;
```

```
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import ex.vo.EmployeesVO;

public class EmployeesClient2 {

	public static void main(String[] args) {

		//config파일의 경로
		String resource = "config/mybatis-config.xml";
		SqlSession session = null;
		//mybatis-config.xml로딩
		try {
			Reader reader = Resources.getResourceAsReader(resource);
			SqlSessionFactory sqlMapper = new
SqlSessionFactoryBuilder().build(reader);

			session = sqlMapper.openSession();
			System.out.println("세션 생성 완료!");

		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		EmployeesVO emp = session.selectOne("emp.selectById", "test2");

		System.out.println(emp);
	}
}
```

■ select 예제3 (검색)

- 성이 '김' 씨인 사람 검색

① employees.xml 수정

employees.xml
기존 코드에 추가 <select id="selectByLastName" resultType="ex.vo.EmployeesVO" parameterType="String"> SELECT * FROM employees WHERE name LIKE #{name} </select>

② EmployeesClient.java 수정

EmployeesClient.java

//기존 코드에 추가
 	
List<EmployeesVO> list = session.selectList("emp.selectByLastName", "김%");
 	
for (EmployeesVO emp : list) {
System.out.println("아이디 : " + emp.getId());
System.out.println("비밀번호 : " + emp.getPassword());
System.out.println("이름 : " + emp.getName());
System.out.println("레벨 : " + emp.getLvl());
System.out.println("입사일 : " + emp.getHiredate());
}

- 다른 방법

① employees.xml 수정

employees.xml
기존 코드에 추가 <select id="selectByLastName" resultType="ex.vo.EmployeesVO“ parameterType="String"> SELECT * FROM employees WHERE name LIKE #{lastname}¦¦'%' </select>

② EmployeesClient.java 수정

EmployeesClient.java

```
//기존 코드에 추가

List<EmployeesVO> list = session.selectList("emp.selectByLastName", "김");

        for (EmployeesVO emp : list) {
            System.out.println("아이디 : " + emp.getId());
            System.out.println("비밀번호 : " + emp.getPassword());
            System.out.println("이름 : " + emp.getName());
            System.out.println("레벨 : " + emp.getLvl());
            System.out.println("입사일 : " + emp.getHiredate());
        }
```

■ insert 예제

① employees.xml 수정

employees.xml

```
기존 코드에 추가

<insert id="insert" parameterType="ex.vo.EmployeesVO">
       INSERT INTO employees(id,password,name,lvl,hiredate)
       VALUES(#{id},#{password},#{name},#{lvl},sysdate)
</insert>
```

② EmployeesClient.java 수정

EmployeesClient.java

```
//기존 코드에 추가
EmployeesVO emp = new EmployeesVO();

emp.setId("test2");
emp.setLvl('A');
```

```
emp.setName("김필구");
emp.setPassword("1234");

int result = session.insert("emp.insert", emp);
System.out.println(result+"수행");

session.commit();
```

■ insert 예제2 (시퀀스의 사용)

① 테이블 및 시퀀스 생성

```
CREATE TABLE  "BOARD"
   (   "BOARDNO" NUMBER(6,0) NOT NULL ENABLE,
       "WRITER" VARCHAR2(20) NOT NULL ENABLE,
       "SUBJECT" VARCHAR2(100) NOT NULL ENABLE,
       "CONTENTS" VARCHAR2(4000) NOT NULL ENABLE,
       "REGDATE" DATE NOT NULL ENABLE,
        CONSTRAINT "BOARD_PK" PRIMARY KEY ("BOARDNO") ENABLE
   );

CREATE SEQUENCE   "BOARD_SEQ"  MINVALUE 1 MAXVALUE 999999 INCREMENT BY 1 START
 WITH 1 NOCACHE  NOORDER  NOCYCLE;
```

② mybatis-config.xml에 추가

```
<mapper resource="sqlmapper/board.xml" />
```

③ board.xml 작성

■ **연습문제**

① update를 실행해 보세요.(비밀번호, 이름 수정)

② delete도 실행해 보세요.(id로 검색하여 삭제)

■ 프로젝트 프로세스

아이템 결정	→	분석	→	디자인	→	개발 / 제작	→	테스트 / 런칭
브레인스토밍 아이템 수집 업무분장		요구사항분석 벤치마킹 사용성테스트 페르소나 결정		인포메이션 아키텍쳐 콘텐츠 디자인 UI디자인 ER다이어그램 프로토타입핑		UI시나리오 DB모델링 기능 제작/구현		유닛테스트 부하테스트 종합테스트 런칭

■ guestbook_seq 테이블 생성

Column Name	Data Type	Nullable	Default	Primary Key
NO	NUMBER(4,0)	No	-	1
WRITER	VARCHAR2(100)	No	-	-
CONTENTS	VARCHAR2(420)	No	-	-
REGDATE	DATE	No	-	-
				1 - 4

■ guestbook_seq 시퀀스 생성

Min Value	1
Max Value	9999
Increment By	1
Cycle Flag	N
Order Flag	N
Cache Size	0
Last Number	1

■ WEB-INF → lib 폴더에 ojdbc / mybatis 라이브러리 가져다 놓기

■ META-INF폴더 → context.xml 에 커넥션풀 설정


```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
	<Resource name="jdbc/oracle"
		auth="Container"
		driverClassName="oracle.jdbc.OracleDriver"
		type="javax.sql.DataSource"
		url="jdbc:oracle:thin:@localhost:1521:xe"
		username="test" password="1111"
		maxIdle="10" maxActive="20" maxWait="-1"
	/>
</Context>
```

■ Guestbook VO생성

```
package vo;

import java.sql.Date;

public class Guestbook {

	private int no;
	private String writer;
	private String contents;
	private Date regdate;
```

```
	//기본생성자
	public Guestbook() {}

	//2개짜리
	public Guestbook(String writer, String contents) {

		this.writer = writer;
		this.contents = contents;

	}

	//toString() overriding
	@Override
	public String toString() {
		String guest = "no: " + no+" / 글쓴이 : " +
				writer +" / 내용 : " + contents
				+ " / 등록일 : " + regdate;

		return guest;
	}



	public int getNo() {
		return no;
	}
	public void setNo(int no) {
		this.no = no;
	}
	public String getWriter() {
		return writer;
	}
	public void setWriter(String writer) {
		this.writer = writer;
	}
	public String getContents() {
		return contents;
	}
	public void setContents(String contents) {
		this.contents = contents;
```

```
	}
	public Date getRegdate() {
		return regdate;
	}
	public void setRegdate(Date regdate) {
		this.regdate = regdate;
	}

}
```

■ FactoryUtil 제작

```
package util;

import java.io.Reader;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class FactoryUtil {

	private static SqlSessionFactory factory;
	static {
		try {
			//config의 경로
			String config = "config/mybatis-config.xml";

			Reader reader =
					Resources.getResourceAsReader(config);

			factory =
					new SqlSessionFactoryBuilder().build(reader);

		} catch (Exception e) {
```

```
                    System.out.println("SqlSessionFactory 만들때 익셉션!");
                    e.printStackTrace();
             }
      }
      public static SqlSessionFactory getFactory() {
             return factory;
      }
}
```

■ mybatis-config 파일 생성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC
"-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
      <environments default="development">
             <environment id="development">
                    <transactionManager type="JDBC"/>
                    <dataSource type="JNDI">
                           <property name="data_source"
                           value="java:comp/env/jdbc/oracle"/>
                    </dataSource>
             </environment>
      </environments>

      <mappers>
             <mapper resource="mapper/guestbooks.xml"/>
      </mappers>
</configuration>
```

■ mapper 파일 생성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC
```

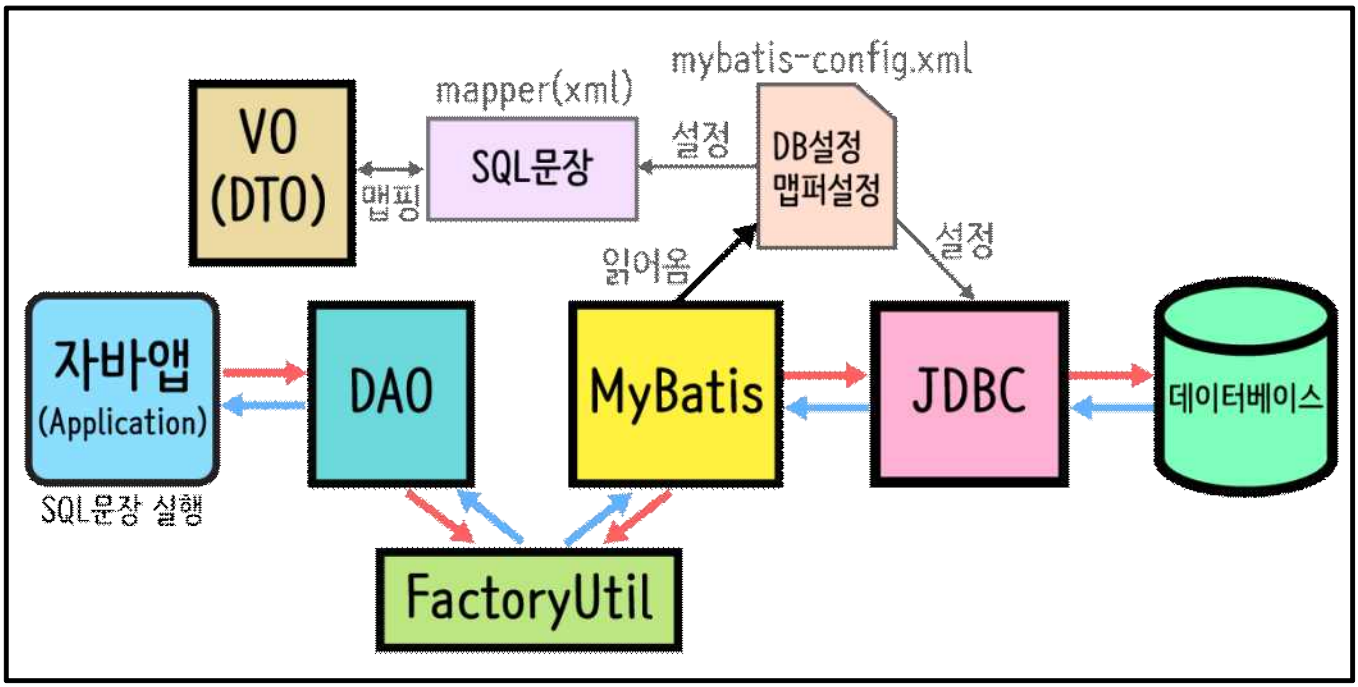
```
"-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="guestbooks">

	<insert id="insert" parameterType="vo.Guestbook">
		INSERT INTO guestbooks(no,writer,contents,regdate)
		VALUES(guestbooks_seq.nextval,#{writer},#{contents},sysdate)
	</insert>

	<select id="selectAll" resultType="vo.Guestbook">
		SELECT no,writer,contents,regdate
		FROM guestbooks
		ORDER BY regdate DESC
	</select>

</mapper>
```

■ GuestbookDAO 생성

```
package dao;

import java.util.List;

import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;

import util.FactoryUtil;
import vo.Guestbook;

public class GuestbooksDAO {

	private static GuestbooksDAO dao;

	//DB와 연결하는 factory가 필요합니다.
	private SqlSessionFactory factory;

	//외부에서 DAO객체를 생성하지 못하게 합니다.
	private GuestbooksDAO() {
```

```
		factory = FactoryUtil.getFactory();
	}

	//메서드를 호출해서 DAO객체를 얻습니다.
	public static GuestbooksDAO getDAO() {
		if(dao==null) {
			dao = new GuestbooksDAO();
		}
		return dao;
	}



	//index.jsp에서 호출하고
	//mapper의 selectAll을 수행하는 메서드
	public List<Guestbook> selectList() {

		return factory.openSession(true).selectList("guestbooks.selectAll");

	}

	//guestbooks.insert구문을 수행하는 메서드
	//글쓴이와 내용을 입력받아서 데이터베이스에
	//입력하는 메서드

	public void insert(Guestbook guestbook) {

		SqlSession session = factory.openSession(true);

		session.insert("guestbooks.insert", guestbook);

	}

}
```

■ MyBatis의 구조



■ DAO의 개념