Spring Framework
- MyBatis 연동

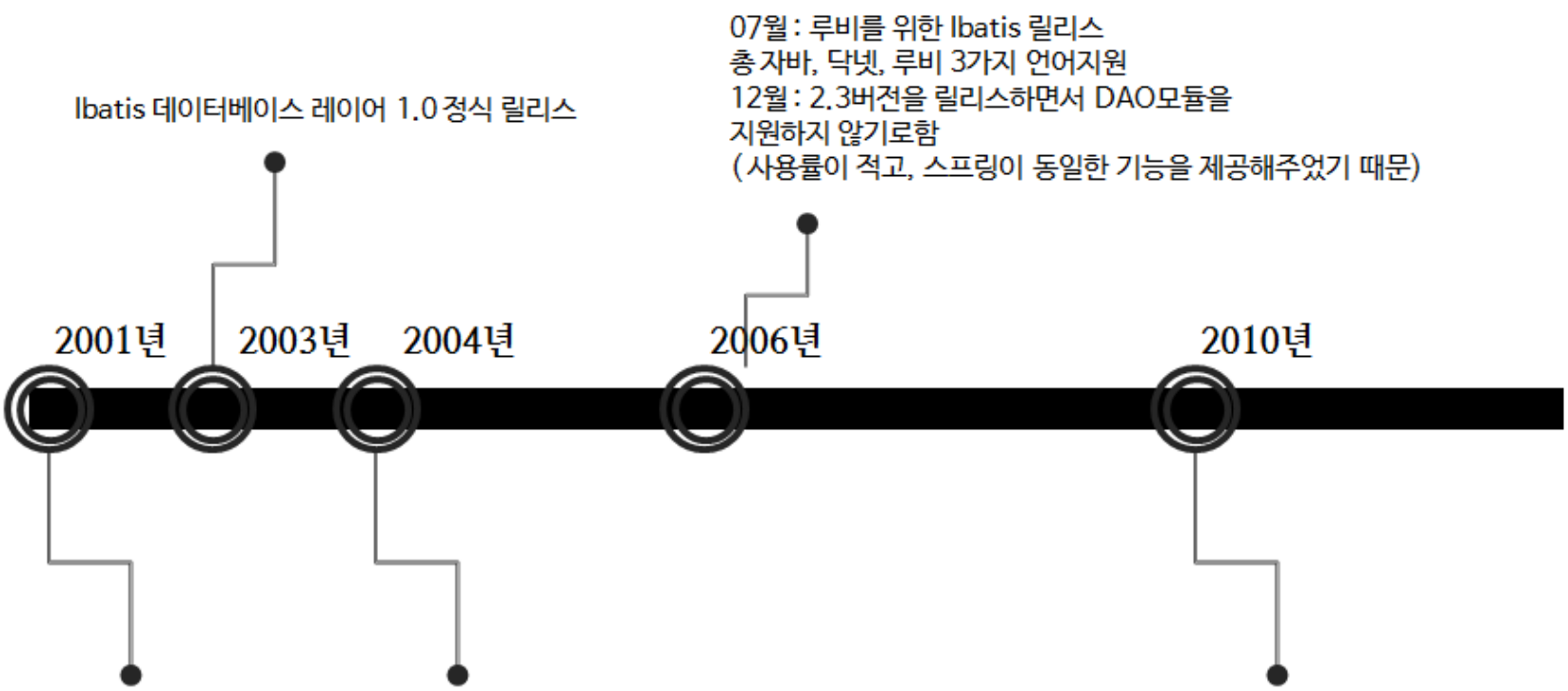
myBatis

- **JDBC 코드의 패턴**
  - Connection -> Statement ->쿼리전송->연결 close
  - 모든 JDBC 코드는 위의 패턴을 가진다.
  - 이 패턴을 캡슐화 하여 JDBC 코드를 간편하게 사용할 수 있도록 Framework화 가능
- **iBatis(MyBatis) 란**
  - SQL실행 결과를 자바 빈즈 혹은 Map 객체에 매핑해주는 Persistence 솔루션으로 SQL을 소스코드가 아닌 XML로 따로 분리해 관리하도록 지원
- **장점**
  - SQL 문장과 프로그래밍 코드의 분리
  - JDBC 라이브러리를 통해 매개변수를 전달하고 결과를 추출하는 일을 간단히 처리가능
  - 자주 쓰이는 데이터를 변경되지 않는 동안에 임시 보관(Cache) 가능
  - 트랜잭션처리 제공

MyBatis 새로운 기능

- 전반적으로 크게 달라진 점은 없어 기존에 ibatis를 사용하셨던 개발자들은 어려움 없이 사용가능.

MyBatis 역사



- **현재 Mybatis에서 제공하는 최신버전은 3.3.0버전**
- **http://blog.mybatis.org/p/products.html 에서 다운받을 수 있도록 제공**

■ Mybatis와 Ibatis의 차이점

구분	Ibatis(아이바티스)	Mybatis(마이바티스)
네임스페이스	선택사항	필수사항
매핑구문정의	xml만 사용	xml과 에노테이션을 사용
동적 SQL	XML 엘리먼트만 사용 동적 SQL을 위한 XML 엘리먼트는 16개 내외	XML 엘리먼트 및 구문 빌더 사용 동적SQL을 위한 XML엘리먼트는 4개 내외 (if, choose, trim, foreach,set)
스프링 연동	스프링 자체 구현체 사용	마이바티스 별도 모듈 사용
지원계획	향후 아이바티스에 대한 공식적인 릴리스는 없음	향후 계속 릴리스 될 예정

XML과 에노테이션

- **XML**
  - 사용방법은 기존과 동일하나 용어들이 변경

이전용어	변경된 용어
SqlMapConfig	Configuration
sqlMap	mapper

  - XML 엘리먼트가 축소

if	조건문
choose(when otherwise)	반복문
trim(where)	공백제거
foreach	반복문(IN절 사용시 유용)
set	update에서 마지막으로 명시된 칼럼 표기에서 쉼표제거

Mybatis

- Mybatis는 직접 스프링과 연동하기 위한 모듈을 제공하고 있어 이 모듈을 이용해서 스프링이 제공하는 DataSource 및 트랜잭션 관리 기능을 MyBais에 적용이 가능하다.
- **Spring 과 MyBatis와 연동 방법**
  - MyBatis-Spring 모듈 추가
  - SqlSessionFactoryBean을 이용해서 SqlSessionFactory 설정
  - 트랜젝션 설정
  - MyBatis 를 이용해서 DAO 구현
  - SqlSession 을 이용해서 구현
  - 매퍼 동적 생성을 이용해서 구현

Mybatis Spring 모듈 추가

```
<!-- spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>${org.springframework-version}</version>
</dependency>

<!-- spring-tx -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-tx</artifactId>
<version>${org.springframework-version}</version>
</dependency>

<!-- mybatis -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.5.6</version>
</dependency>

<!-- mybatis-spring -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis-spring</artifactId>
<version>2.0.6</version>
</dependency>
```

SqlSessionFactoryBean 과 트랜잭션 관리자 설정

- mybatis-spring 모듈이 제공하는 SqlSessionFactoryBean을 이용해서 mybatis의 SqlSessionFactory를 생성

```
<beans:bean id="sqlSessionFactory"
                class="org.mybatis.spring.SqlSessionFactoryBean">
        <beans:property name="dataSource" ref="dataSource" />
        <beans:property name="mapperLocations"
        value="classpath:com/bitcamp/openproject/mapper/mybatis/*.xml" />
</beans:bean>
```

SqlSessionFactoryBean 과 트랜잭션 관리자 설정

- Mapper 작성 (MemberMapper.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="com.bitcamp.memberboard.mapper.mybatis.MemberMapper">

<select id="selectById"
resultType="com.bitcamp.openproject.member.model.Member">
select * from member where member_id = #{id}
</select>

<insert id="insertMember"
parameterType="com.bitcamp.openproject.member.model.Member">

insert into member
values( #{member_id}, #{member_name}, #{password}, #{regdate}, #{photo})

</insert>
</mapper>
```

SqlSessionFactoryBean 과 트랜잭션 관리자 설정

- MyBatis 트랜잭션 관리자는 DataSourceTransactionManager를 사용
  servlet-context.xml 아래 코드 적용

```
<!-- 트랜젝션 처리 시작 -->

<beans:bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <beans:property name="dataSource">
        <beans:ref bean="dataSource" />
        </beans:property>
</beans:bean>


<tx:annotation-driven transaction-manager="transactionManager" />


<!-- 트랜젝션 처리 끝 -->
```

MyBatis를 이용한 DAO 구현

- SqlSessionFactoryBean을 이용해서 mybatis의 SqlSessionFactory를 생성하면 MyBatis를 이용해서 DAO 구성 가능
- DAO 구현 방법은 두 가지로 구분
  - **SqlSessionTemplate을 이용한 DAO 구현**
  - **자동 매퍼 생성 기능을 이용한 DAO 구현**

SqlSessionTemplate을 이용한 DAO 구현

- SqlSessionTemplate 클래스는 SqlSession 을 위한 스프링 연동 부분을 구현하고 있고, DAO 는 SqlSessionTemplate를 이용해서 구현

```
<beans:bean id="sqlSessionFactory"
        class="org.mybatis.spring.SqlSessionFactoryBean">
        <beans:property name="dataSource" ref="dataSource" />
        <beans:property name="mapperLocations"
value="classpath:com/bitcamp/memberboard/mapper/mybatis/*.xml" />
</beans:bean>


<beans:bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
        <beans:constructor-arg index="0" ref="sqlSessionFactory" />
</beans:bean>


<beans:bean id="mybatisDao"
        class="com.bitcamp.memberboard.member.dao.MemberMyBatisDao" />
```

SqlSessionTemplate을 이용한 DAO 구현

com.bitcamp.openproject.member.dao

```
public class MemberMyBatisDao {

    @Autowired
    private SqlSessionTemplate sqlSessionTemplate;

    public void insert(Member member) {
        String str =
        "com.bitcamp.openproject.mapper.mybatis.MemberMapper.insertMember";
        int num = sqlSessionTemplate.update(str, member);
        System.out.println(num);
    }

    public Member selectById(String userid) {
        String str =
        "com.bitcamp.openproject.mapper.mybatis.MemberMapper.selectById";
        return (Member) sqlSessionTemplate.selectOne(str, userid);
    }

}
```

자동 매퍼 생성 기능을 이용한 DAO 구현

- MyBatis를 이용해서 DAO를 구현 할 때 대부분의 코드는 단순희 SqlSession의 메서드를 호출하는 것으로 끝난다.
  이러한 단순 코드 작업을 줄여주기 위해서 MyBatis는 인터페이스를 이용해서 런타임에 매퍼 객체를 생성하는 기능을 제공.

자동 매퍼 생성 기능을 이용한 DAO 구현

com.bitcamp.openproject.member.dao.LoginDao.java

```
package com.bitcamp.openproject.member.dao;

import com.bitcamp.openproject.member.model.Member;

public interface LoginDao {

	public Member selectById(String userId);
	public void insertMember(Member member);
}
```

자동 매퍼 생성 기능을 이용한 DAO 구현

com.bitcamp.openproject.member.dao.LoginService.java

```
public class LoginService {

        // @Autowired
        // private MemberMyBatisDao dao;

        @Autowired
        private SqlSessionTemplate sqlSessionTemplate;

        private LoginDao dao;

        @Transactional
        public User login(
                String userid,
                String password,
                HttpServletRequest req) throws SQLException {

                dao = sqlSessionTemplate.getMapper(LoginDao.class);
```

자동 매퍼 생성 기능을 이용한 DAO 구현

com.bitcamp.openproject.member.dao.LoginService.java

```
                Member member = dao.selectById(userid);
                System.out.println(member);

                if (member == null) {
                        throw new Exception();
                }
                if (!member.matchPassword(password)) {
                        throw new Exception();
                }
                User user = new User();
                user.setName(member.getName());
                user.setUserid(member.getMemberid());
                user.setPhoto(member.getPhoto());

                return user;
        }

}
```

XML과 에노테이션

- **참고사이트**
  - http://mybatis.github.io/mybatis-3/ko/index.html

[과제] 회원관리프로그램

- 데이터베이스 처리부분을 Mybatis를 이용해서 구현