Anyframe MyBatis Plugin

Version 1.1.0

I.Installation

MyBatis Plugin은 MyBatis [http://www.mybatis.org/]와 Anyframe과 연계를 가이드하기위한 샘플코드와 이 오픈소스들을 활용하는데 필요한 참조 라이브러리들로 구성되어 있다.

Installation

Command 창에서 다음과 같이 명령어를 입력하여 MyBatis plugin을 설치한다.

```
mvn anyframe:install -Dname=mybatis
```

installed(mvn anyframe:installed) 혹은 jetty:run(mvn clean jetty:run) command를 이용하여 설치 결과를 확인해볼 수 있다.

Plugin Name	Version Range
Core [http://dev.anyframejava.org/docs/anyframe/plugin/essential/core/1.6.0/reference/htmlsingle/core.html]	2.0.0 > * > 1.4.0

II.MyBatis

MyBatis Plugin은 object-oriented application에서 relational database 사용을 쉽게 해주는 Data Mapper Framework 이다. iBatis가 version 3이 되고 source repository를 Apache에서 google code로 이전 하면서 이름을 MyBatis로 변경 하였다. MyBatis는 내부적으로 DataSource 서비스를 이용하고 있으므로, DataSource 서비스와 같이 배포되어야 함에 유의하도록 한다.

MyBatis 매뉴얼에서 제공하는 모든 테스트 코드는 HSQL DB를 기반으로 실행된다.

1.MyBatis-Spring

MyBatis-Spring은 MyBatis와 Spring Framework와 연계를 원활하게 해주는 오픈 소스이다. MyBatis-Spring 모듈을 사용하면 MyBatis factory와 session 클래스 등 을 스프링에서 load 한다. 또한 Spring Bean에서 MyBatis data mapper, SqlSession을 inject 할 수 있도록 제공해주며 transation을 Spring Framework으로 위임 할 수 있다.

Why MyBatis-Spring?

Spring version 2 에서는 iBatis version 2를 지원하고 있다. MyBatis를 Spring version 3 에서 지원하도록 노력하였으나 Spring version 3 개발완료 시점에 MyBatis는 릴리즈를 하지 못하였다. Spring 팀에서는 릴리즈가 되지 않은 MyBatis를 Spring과 연계 하는것을 원하지 않았고 이로 인하여 Spring version 3에서 MyBatis 지원은 연기 되었다. (Spring 3.1 에서 지원된다고 하였으나 그 역시 연기 되었고 언제가 될지는 확실하지 않다. 본 매뉴얼이 작성될 당시 Spring 최신 버전은 3.1.1 임) Spring 연계에 지속적으로 관심을 가지고 있던 MyBatis 커뮤니티에서는 관련있는 사람들을 모아서 MyBatis-Spring 이라는 서브 프로젝트를 만들어 Spring과의 연계를 지원하고 있다.

2.Configuration

본 절에서는 Anyframe과 MyBatis와 연계와 관련된 설정에 대해서만 다룰 것이다. MyBatis에 대한 자세한 내용은 MyBatis 매뉴얼 [http://www.mybatis.org/core/]을 참고 하도록 한다.

2.1.SqlSessionFactoryBean

MyBatis에서는 SqlSessionFactoryBuilder 클래스를 사용하여 session factory를 생성한다. MyBatis-Spring은 SqlSessionFactoryBuilder 클래스 대신 SqlSessionFactoryBean 클래스를 사용하여 생성한다.

2.1.1.Configuration

Factory Bean은 다음과 같이 Spring XML 설정 파일에 정의 한다.

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="configLocation" value="classpath:sql/mybatis/mybatis-config.xml" />
    <property name="mapperLocations" value="classpath:sql/mybatis/mapper-mybatis-*.xml" />
</bean>
```

2.1.2.Properties

SqlSessionFactoryBean 클래스의 속성들 주요 속성들에 대해서 설명한다. 이 속성들 중 필수적으로 정의 해야하는 속성은 dataSource 이다.

Property	Description	Required	Default Value
dataSource	JDBC Datasource를 설정한다.	Y	N/A
configLocation	MyBatis XML 설정 파일의 위치를 지정한다.	N	N/A
mapperLocations	MyBatis mapper 파일(쿼리 파일)의 위치를 지정한다.	N	N/A
transactionFactory	MyBatis TransactionFactory를 사용할 경우 설정한다.	N	N/A

2.2.Transactions

Mybatis-Spring을 사용하는 중요한 목적중에 한가지는 transaction을 Spring Framework에 위임하기 위해서 이다. MyBatis에서 새로운 transaction manager를 생성하는 것보다 Spring에서 제공하는 DataSourceTransactionManager를 활용한다. Transaction에 대한 설정은 @Transactional annotation 이나 AOP style 설정을 모두 지원한다.

2.2.1.Standard Configuration

Spring Framemwork에서 제공하는 transaction을 사용하기 위한 Spring XML 설정 파일은 다음과 같다.

```
<bean id="txManager"
 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
</bean>
```

Transaction Manager에 정의된 datasource와 SqlSessionFactoryBean에 정의된 datasource는 같은 bean 이어야 한다.

2.2.2.Container Managed Transaction

Spring Framework에서 제공하는 DataSourceTransactionManager 클래스를 사용하지 않고 JEE container(WAS)에 transaction을 위임하기 위한 Spring XML 설정은 다음과 같다.

```
<tx:jta-transaction-manager>
```

위와 같이 DataSourceTransactionManager 클래스를 사용하지 않고 JEE container(WAS)에 transaction을 위임하는 경우 SqlSesssionFactoryBean 설정에는 다음과 같이 ManagedTransactionFactory를 추가해줘야 한다.

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="transactionFactory">
        <bean class="org.apache.ibatis.transaction.managed.ManagedTransactionFactory" />
    </property>
</bean>
```

3.Usage

3.1.SqlSession

MyBatis에서는 SqlSessionFactory 클래스를 가지고 SqlSession을 생성한다. 생성된 session은 SQL 문을 실행, commit, rollback, 그리고 session close 역할을 한다. 그러나 MyBatis-Spring은 SqlSessionFactory 클래스를 직접 사용할 필요가 없다. Spring Bean에서 thread safe SqlSession을 inject 할 수 있기 때문이다. 또한 commit, rollback, 그리고 session close 모두 Spring transaction 설정을 통해 관리 된다.

3.1.1.SqlSessionTemplate

SqlSessionTemplate 클래스는 MyBatis-Spring의 핵심 기능이며 MyBatis SqlSessions 관리 및 MyBatis SQL method 호출을 담당한다. SqlSessionTemplate 클래스는 SqlSessionFactory를 생성자의 인자로 받을수 있다.

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

위에서 정의한 bean은 DAO 클래스에서 inject 해서 사용 할 수 있다.

```
public class MovieDao {
    @Inject
    @Named("sqlSession")
    public SqlSession sqlSession;

    public Movie getMovie(String movieId) {
        return (Movie)sqlSession.selectOne("Movie.getMovie", movieId);
    }
}
```

3.1.2.SqlSessionDaoSupport

SqlSessionDaoSupport 클래스는 SqlSession을 제공하는 abstract support class 이다. getSqlSession() 메소드를 호출하여 SqlSessionTemplate을 사용 할 수 있으며 MyBatis SQL method를 아래와 같이 호출 할 수 있다.

```
public class MovieDao extends SqlSessionDaoSupport {
    public Movie getMovie(String movieId) {
        return (Movie)getSqlSession().selectOne("Movie.getMovie", movieId);
    }
}
```

3.2.Injecting Mappers

SqlSessionDaoSupport 혹은 SqlSessionTemplate 클래스를 활용항 DAO를 작성하는 방법이외에도 MyBatis-Spring에서 제공하는 proxy factory인 MapperFactoryBean를 이용할 수 도 있다. MapperFactoryBean 클래스는 service bean에서 data mapper interface(dao interface)를 inject 할 수 있도록 해준다. mapper interface(dao interface)를 이용하여 Dao를 호출 할 경우 Dao에 대한 구현체를 만들지 않아도 된다.

3.2.1.MapperFactoryBean

Data Mapper(Dao Interface)를 Spring Bean으로 등록 하는 방법은 아래와 같다.

```
<bean id="movieMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
    <property name="mapperInterface"
 value="org.anyframe.sample.mybatis.moviefinder.dao.MovieDao" />
    <property name="sqlSessionFactory" ref="sqlSessionFactory" />
</bean>
```

MovieDao(MovieMapper)에 대한 proxy 객체를 생성하고 해당 어플리케이션에서는 proxy 객체를 inject 해서 사용한다. proxy 객체는 런타임 시에 생성이 되므로 Mapper(Dao)는 반드시 Interface로 작성되어야 하고 구현체는 필요하지 않다. Spring Bean에서 mapper(dao)를 inject 해서 사용하는 방법은 다음과 같다.

```
@Service("mybatisMovieService")
public class MovieServiceImpl implements MovieService {

    @Inject
    @Named("movieDao")
    private MovieDao movieDao;

    public Movie get(String movieId) throws Exception {
        return movieDao.getMovie(movieId);
    }
}
```

```
public interface MovieDao {
    Movie getMovie(String movieId);
    ...
}
```

3.2.2.MapperScannerConfigurer

Mapper(Dao)를 Spring XML 설정 파일에 등록 하는 대신 MapperScannerConfigurer 클래스를 사용하여 classpath내에 있는 mapper(dao) interface를 자동으로 등록 할 수 있다. MapperScannerConfigurer에 대한 설정은 다음과 같다.

```
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="org.anyframe.sample.mybatis.moviefinder.dao" />
</bean>
```

위와 같이 설정 할 경우 지정된 basePackage 하위에 존재 하는 Mapper(Dao) Interface를 검색하여 등록 한다.

4.Resources

- 다운로드

  다음에서 테스트 DB를 포함하고 있는 hsqldb.zip과 sample 코드를 포함하고 있는 anyframe-sample-mybatis.zip 파일을 다운받은 후, 압축을 해제한다. 그리고 hsqldb 폴더 내의 start.cmd (or start.sh) 파일을 실행시켜 테스트 DB를 시작시켜 놓는다.

  - Maven 기반 실행

    Command 창에서 압축 해제 폴더로 이동한 후, mvn compile exec:java -Dexec.mainClass=...이라는 명령어를 실행시켜 결과를 확인한다. 각 Eclipse 프로젝트 내에 포함된 Main 클래스의 JavaDoc을 참고하도록 한다.

  - Eclipse 기반 실행

    Eclipse에서 압축 해제 프로젝트를 import한 후, src/main/java 폴더 하위의 Main.java를 선택하고 마우스 오른쪽 버튼 클릭하여 컨텍스트 메뉴에서 Run As > Java Application을 클릭한다. 그리고 실행 결과를 확인한다.

표 4.1. Download List

Name	Download
hsqldb.zip	Download [http://dev.anyframejava.org/docs/anyframe/plugin/optional/mybatis/1.1.0/reference/sample/hsqldb.zip]
anyframe-sample-mybatis.zip	Download [http://dev.anyframejava.org/docs/anyframe/plugin/optional/mybatis/1.1.0/reference/sample/anyframe-sample-mybatis.zip]

- 참고자료

  - MyBatis [http://www.mybatis.org/core/]