**001、从SqlSessionFactory开始**

和Hibernate的SessionFactory类似，在MyBatis中，一切都以**SqlSessionFactory**为起点。构造SqlSessionFactory的类是**SqlSessionFactoryBuilder**，该类支持XML和注解方式。

和Spring类似，MyBatis也提供了自己的**Resource**来帮助解析配置文件。创建一个简单的SqlSessionFactory的语句如下：

String resource = "org/mybatis/example/mybatis-config.xml";

InputStream inputStream = Resources.getResourceAsStream(resource);

sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

SqlSessionFactory的配置文件包含了很多设置，比如：数据源、事务管理器、mappers等。

**002、以SqlSession为中心**

有了SqlSessionFactory，现在可以获取**SqlSession**了，它提供了一切操作SQL的方法。

SqlSession session = sqlSessionFactory.openSession();

session.selectOne(“...selectBlog”,101);

**Mapper**：上面提供的方法缺乏语义性。Mapper只是一个用作配置的XML文件而已。让Mapper发挥更大威力的是将MapperXML和MapperClass对应起来。

BlogMapper mapper = session.getMapper(BlogMapper.class); //是否想起SpringBean？

mapper.selectBlog(101);

**003、作用域和生命周期(Scope & LifeCycle)**

SqlSessionFactoryBuilder：创建完SqlSessionFactory后即丢弃；

SqlSessionFactory：全局单例；

SqlSession：用完就关闭。它不是线程安全的。

Mapper：方法内使用。与Spring集成后，可以当做Bean使用。

**004、配置(Configuration)：基于XML**

配置元素的层次结构如下：

configuration

properties

settings

typeAliases

typeHandlers

objectFactory

plugins

environments

environment

transactionManager

dataSource

databaseIdProvider

mappers

下面分别说明：

**properties**：相当于Spring中的PropertyPlaceHolder。读取顺序：该元素的子元素---properties文件---通过Java Method传入，后面的覆盖前面的。

**settings**：控制MyBatis的运行行为。它提供的选项很多，如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 作用描述 | 可用值 | 默认值 |
| cacheEnabled | 全局开启或禁用mapper中的缓存配置 | true|false | true |
| lazyLoadingEnabled | 全局延迟加载 | true|false | true |
| aggressiveLazyLoading |  | true|false | true |
| multipleResultSetsEnabled | mapper是否可返回  multiple ResultSet | true|false | true |
| useColumnLabel | 使用column标签而不是名称 | true|false | true |
| useGeneratedKeys | 使用生成的key | true|false | false |
| autoMappingBehavior | 自动映射。NONE不映射，PARTIAL不提供嵌套映射，FULL都支持 | NONE|PARTIAL|FULL | PARTIAL |
| defaultExecutorType | REUSE重复使用prepared Statement，BATCH重复使用和批量修改 | SIMPLE|REUSE|BATCH | SIMPLE |
| defaultStatementTimeout | 超时时间 | 任意正数 | NOT SET |
| safeRowBoundsEnabled | 允许在嵌套语句上使用RowBounds | true|false | false |
| mapUnderscoreToCamelCase | 表列和POJO字段使用驼峰命名法自动映射 | true|false | false |
| localCacheScope | 缓存作用域 | SESSION|STATEMENT | SESSION |
| jdbcTypeForNull | Null处理 | JdbcType enumeration | OTHER |
| lazyLoadTriggerMethods | 触发延迟加载的方法名 | 逗号分隔的字符串 |  |

**typeAliases**：别名。

**typeHandlers**：为PreparedStatement设置参数和从ResultSet中获取值时都使用TypeHandler进行转换。如想自定义，扩展org.apache.ibatis.type.BaseTypeHandler即可。注意MyBatis是不知道数据类型的，因此需要在TypeHandler类上使用注解@MappedJdbcTypes()来说明该类是对那些JdbcType进行转换。

**objectFactory**：MyBatis创建Result的类，可以自行扩展，继承DefaultObjectFactory即可。

**plugins**：提供拦截，可供拦截的类和方法列表如下：

Executor (update, query, flushStatements, commit, rollback, getTransaction, close, isClosed)；

ParameterHandler (getParameterObject, setParameters)；

ResultSetHandler (handleResultSets, handleOutputParameters)；

StatementHandler (prepare, parameterize, batch, update, query)。

**environment**：配置多个数据库环境，分别用作开发、测试、生产。

**005、SQL Mapper详解**

可配置的元素：

cache，cache-ref：前者配置当前mapper的缓存(对应一个命名空间)，后者引用其他命名空间的cache。

resultMap：相当于Spring中的RowMapper。

parameterMap：已经deprecated了。

sql：可重用的sql片段，可被其他statement引用。

select、update、delete、inset：普通DML语句。

#{ }和${ }，前者用于设置PreparedStatement的参数，后者用于直接替代(要注意注入攻击)。

**006、动态SQL**

使用OGNL表达式消除了很多配置字元素。

**<if>**

<select id="findActiveBlogLike" parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

<if test="title != null">

AND title like #{title}

</if>

<if test="author != null and author.name != null">

AND author\_name like #{author.name}

</if>

</select> // test和OGNL的使用

**choose、when、otherwise**

<select id="findActiveBlogLike" parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

<choose>

<when test="title != null">

AND title like #{title}

</when>

<when test="author != null and author.name != null">

AND author\_name like #{author.name}

</when>

<otherwise>

AND featured = 1

</otherwise>

</choose>

</select>