# 添加Maven依赖或引入Jar包

如果你使用Maven，只需要添加如下依赖：

<dependency>

<groupId>com.github.abel533</groupId>

<artifactId>mapper</artifactId>

<version>2.3.4</version>

</dependency>

如果你想引入Jar包，你可以从下面的地址下载：

<https://oss.sonatype.org/content/repositories/releases/com/github/abel533/mapper/>

<http://repo1.maven.org/maven2/com/github/abel533/mapper/>

由于通用Mapper依赖JPA，所以还需要下载persistence-api-1.0.jar：

<http://repo1.maven.org/maven2/javax/persistence/persistence-api/1.0/>

# 集成com.github.abel533.mapper.Mapper

对com.github.abel533.mapper.Mapper的集成，实际上是配置MapperHelper。

下面针对Java编码和Spring两种情况分别讲解配置方法。出这两种情况的配置外，还有一种拦截器的方式会作为第三种单独说明。

## 第一种、Java编码

对于单独使用Mybatis，通过如下方式创建sqlSessionFactory:

Reader reader **=** Resources**.**getResourceAsReader**(**"mybatis-config.xml"**);**

sqlSessionFactory **=** **new** SqlSessionFactoryBuilder**().**build**(**reader**);**

reader**.**close**();**

使用直接的JAVA编码方式，可以在初始化sqlSessionFactory的地方按照下面的方式操作：

*//从上面的sqlSessionFactory取出一个session*

session **=** sqlSessionFactory**.**openSession**();**

*//创建一个MapperHelper*

MapperHelper mapperHelper **=** **new** MapperHelper**();**

*// 设置UUID生成策略*

*// 配置UUID生成策略需要使用OGNL表达式*

*// 默认值32位长度:@java.util.UUID@randomUUID().toString().replace("-", "")*

mapperHelper**.**setUUID**(**""**);**

*// 主键自增回写方法,默认值MYSQL,详细说明请看文档*

mapperHelper**.**setIDENTITY**(**"HSQLDB"**);**

*// 序列的获取规则,使用{num}格式化参数，默认值为{0}.nextval，针对Oracle*

*// 可选参数一共3个，对应0,1,2,分别为SequenceName，ColumnName, PropertyName*

mapperHelper**.**setSeqFormat**(**"NEXT VALUE FOR {0}"**);**

*// 设置全局的catalog,默认为空，如果设置了值，操作表时的sql会是catalog.tablename*

mapperHelper**.**setCatalog**(**""**);**

*// 设置全局的schema,默认为空，如果设置了值，操作表时的sql会是schema.tablename*

*// 如果同时设置了catalog,优先使用catalog.tablename*

mapperHelper**.**setSchema**(**""**);**

*// 主键自增回写方法执行顺序,默认AFTER,可选值为(BEFORE|AFTER)*

mapperHelper**.**setOrder**(**"AFTER"**);**

*// 注册通用Mapper接口*

mapperHelper**.**registerMapper**(**Mapper**.**class**);**

mapperHelper**.**registerMapper**(**HsqldbMapper**.**class**);**

*//配置完成后，执行下面的操作*

mapperHelper**.**processConfiguration**(**session**.**getConfiguration**());**

*//OK - mapperHelper的任务已经完成，可以不管了*

上面配置参数的时候，是一个个调用set方法进行的，你还可以使用MapperHelper的MapperHelper(Properties properties)构造方法，或者调用setProperties(properties)方法，通过.properties配置文件来配置。

如果你的情况适用于这种方式，推荐你用JAVA编码的方式处理。

## 第二种、和Spring集成

在Spring中使用的时候，可以通过xml达到上面Java编码方式的效果。如下配置：

<bean class="com.github.abel533.mapperhelper.MapperHelper"

depends-on="sqlSession" init-method="initMapper" scope="singleton" lazy-init="false">

<property name="mappers">

<array>

*<!-- 可以配置多个 -->*

<value>com.isea533.mybatis.mapper.Mapper</value>

</array>

</property>

*<!-- 对于多数据源，这里也可以像上面这样配置多个 -->*

<property name="sqlSessions" ref="sqlSession"/>

</bean>

可以看到配置中依赖了sqlSession，所以使用这种方式，你还要在Spring的配置中保证sqlSession存在。一般情况下都会在Spring定义sqlSession。一般的定义方法如下：

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" scope="prototype">

<constructor-arg index="0" ref="sqlSessionFactory"/>

</bean>

在Spring中使用这种方式的时候，Spring启动完成的时候，所有的通用Mapper都已经处理完成了。后面就可以直接使用通用方法，不需要拦截器来执行了。

**重要:**目前和Spring集成的时候存在一个bug，这个bug产生的原因如下：

如果你的Mapper没有在项目启动的时候通过@Autowired注入到Service或者其他类中，那么这个Mapper是一个还没有注册到MybatisSqlSession中的Mapper,MapperHelper在启动过程中没有处理该Mapper，这就会导致dynamicSQL**无法实例化一类的异常**。

这种情况只能使用拦截器处理，如果没有更好的解决版本，后面会恢复拦截器的方式。

## 第三种、拦截器

首先这里提到的三种配置方式不要重复进行，只需要按其中一种方式即可。

配置方式分为mybatis配置方式和纯spring集成方式。

### Mybatis配置文件方式

在mybatis-config.xml中添加如下配置:

<plugins>

<plugin interceptor="com.github.abel533.mapperhelper.MapperInterceptor">

*<!--================================================-->*

*<!--可配置参数说明(一般无需修改)-->*

*<!--================================================-->*

*<!--UUID生成策略-->*

*<!--配置UUID生成策略需要使用OGNL表达式-->*

*<!--默认值32位长度:@java.util.UUID@randomUUID().toString().replace("-", "")-->*

*<!--<property name="UUID" value="@java.util.UUID@randomUUID().toString()"/>-->*

*<!--主键自增回写方法,默认值MYSQL,详细说明请看文档-->*

<property name="IDENTITY" value="HSQLDB"/>

*<!--序列的获取规则,使用{num}格式化参数，默认值为{0}.nextval，针对Oracle-->*

*<!--可选参数一共3个，对应0,1,2,分别为SequenceName，ColumnName,PropertyName-->*

<property name="seqFormat" value="{0}.nextval"/>

*<!--主键自增回写方法执行顺序,默认AFTER,可选值为(BEFORE|AFTER)-->*

*<!--<property name="ORDER" value="AFTER"/>-->*

*<!--通用Mapper接口，多个通用接口用逗号隔开-->*

<property name="mappers" value="com.github.abel533.mapper.Mapper"/>

</plugin>

</plugins>

### 纯Spring配置方式

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="mapperLocations">

<array>

<value>classpath:mapper/\*.xml</value>

<value>classpath:com/isea533/mybatis/mapper/\*.xml</value>

</array>

</property>

<property name="typeAliasesPackage" value="com.isea533.mybatis.model"/>

<property name="plugins">

<array>

<bean class="com.github.abel533.mapperhelper.MapperInterceptor">

<property name="properties">

*<!-- 属性一行一个，具体属性参考mybatis-config.xml中的属性 -->*

<value>

mappers=com.github.abel533.mapper.Mapper

</value>

</property>

</bean>

</array>

</property>

</bean>

如果你同时使用了其他的插件，请将其他插件配置通用Mapper拦截器的下面。

# IDENTITY参数配置（仅对 insert 有用）

对于不同的数据库，需要配置不同的参数，这些参数如下：

* **DB2**: VALUES IDENTITY\_VAL\_LOCAL()
* **MYSQL**: SELECT LAST\_INSERT\_ID()
* **SQLSERVER**: SELECT SCOPE\_IDENTITY()
* **CLOUDSCAPE**: VALUES IDENTITY\_VAL\_LOCAL()
* **DERBY**: VALUES IDENTITY\_VAL\_LOCAL()
* **HSQLDB**: CALL IDENTITY()
* **SYBASE**: SELECT @@IDENTITY
* **DB2\_MF**: SELECT IDENTITY\_VAL\_LOCAL() FROM SYSIBM.SYSDUMMY1
* **INFORMIX**: select dbinfo('sqlca.sqlerrd1') from systables where tabid=1
* **JDBC**:这会令 MyBatis 使用 JDBC 的 getGeneratedKeys 方法来取出由数据库内部生成的主键（比如：像 MySQL 和 SQL Server 这样的关系数据库管理系统的自动递增字段）。

JAVA编码方式使用：

*// 主键自增回写方法,默认值MYSQL,详细说明请看文档*

mapperHelper**.**setIDENTITY**(**"HSQLDB"**);**

Spring中可以属性注入：

<property name="IDENTITY" value="MYSQL"/>