

# mybatis-generator

## 1 概述

在使用mybaits框架的时候，有三种东西是我们在代码中需要特别注意的：

1. 实体类
2. 映射接口
3. 映射文件

默认情况下，这三种对象都需要我们自己手动的去创建，并进行编写，特别是映射文件。

为了简化项目中对应mybaits框架的使用，mybaits官方还提供了mybaits组件（实体类、映射接口、映射文件）自动生成的插件：[MyBatis Generator](#)


### MyBatis Generator



Last Published: 24 November 2019 | Version: 1.4.0

USER'S GUIDE	
Introduction	
What's New?	
Quick Start Guide	
Running MyBatis Generator	>
Tasks After Running MyBatis Generator	
Migrating from Ibatis	
Migrating from Abator	
XML Configuration Reference	>
Using the Generated Objects	>
Database Specific Information	>
Other Reference Information	>

## Introduction to MyBatis Generator

MyBatis Generator (MBG) is a code generator for MyBatis [MyBatis](#) . It will generate code for all versions of MyBatis. It will introspect a database table (or many tables) and will generate artifacts that can be used to access the table(s). This lessens the initial nuisance of setting up objects and configuration files to interact with database tables. MBG seeks to make a major impact on the large percentage of database operations that are simple CRUD (Create, Retrieve, Update, Delete). You will still need to hand code SQL and objects for join queries, or stored procedures.

MBG generates code in different styles, and for different languages, depending on how it is configured. For example, MBG can generate Java or Kotlin code. And MBG can generate MyBatis3 compatible XML - although that is now considered a legacy use of MBG. The newer styles of generated code do not require XML.

Depending on how it is configured, MyBatis Generator may generate:

- Java or Kotlin classes that match the table structure. This may include:
  - a class to match the primary key of the table (if there is a primary key)
  - a class to match the non-primary key fields of the table (except BLOB fields)
  - a class to include the BLOB fields of a table (if the table has BLOB fields)
  - a class to enable dynamic selects, updates, and deletes

注意，使用该插件，在项目中可以快速生成和数据库中表对应的实体类，以及基础的映射接口和映射文件

## 2 使用

在maven项目中，使用mybatis-generator插件，只需要简单的几步即可完成：

1. 在pom文件中引入该插件的坐标
2. 在配置文件中对插件进行配置（模板）
3. 执行该maven插件的goal目标（自动生成实体类、映射接口、映射文件）

### 1、在pom文件中映射该插件

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <groupId>com.briup</groupId>
6      <artifactId>estore</artifactId>
7      <version>0.0.1-SNAPSHOT</version>
8      <packaging>war</packaging>
9
10     <properties>
11         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12         <maven.compiler.source>1.8</maven.compiler.source>
13         <maven.compiler.target>1.8</maven.compiler.target>
14     </properties>
15
16     <dependencies>
17         <!-- 引入servlet3.1的依赖 -->
18         <dependency>
19             <groupId>javax.servlet</groupId>
20             <artifactId>javax.servlet-api</artifactId>
21             <version>3.1.0</version>
22             <scope>provided</scope>
23         </dependency>
24
25         <!-- 引入jstl的依赖，jsp页面中会使用到 -->
26         <dependency>
27             <groupId>javax.servlet</groupId>
28             <artifactId>jstl</artifactId>
29             <version>1.2</version>
30         </dependency>
31
32         <!-- 引入Mybatis的依赖 -->
33         <dependency>
34             <groupId>org.mybatis</groupId>
35             <artifactId>mybatis</artifactId>
36             <version>3.4.6</version>
37         </dependency>
38
39         <!-- 数据库mysql的驱动包依赖 -->
40         <dependency>
41             <groupId>mysql</groupId>
42             <artifactId>mysql-connector-java</artifactId>
43             <version>5.1.38</version>
44         </dependency>
45
46         <!--
47         <dependency>
48             <groupId>com.oracle</groupId>
49             <artifactId>ojdbc8</artifactId>
50             <version>8</version>
51         </dependency>
52         -->
53
54
55         <!-- 日志框架log4j的依赖 -->
56         <dependency>
57             <groupId>log4j</groupId>
58             <artifactId>log4j</artifactId>
59             <version>1.2.17</version>
```

```

60         </dependency>
61
62         <!-- junit的依赖 -->
63         <dependency>
64             <groupId>junit</groupId>
65             <artifactId>junit</artifactId>
66             <version>4.7</version>
67             <scope>test</scope>
68         </dependency>
69
70     </dependencies>
71
72     <build>
73         <plugins>
74             <plugin>
75                 <groupId>org.mybatis.generator</groupId>
76                 <artifactId>mybatis-generator-maven-plugin</artifactId>
77                 <version>1.4.0</version>
78                 <configuration>
79                     <!-- 配置文件的位置 -->
80
81                     <configurationFile>src/main/resources/generatorConfig.xml</configurationFile>
82                     <verbose>true</verbose>
83                     <overwrite>true</overwrite>
84                 </configuration>
85                 <dependencies>
86                     <dependency>
87                         <groupId>org.mybatis.generator</groupId>
88                         <artifactId>mybatis-generator-core</artifactId>
89                         <version>1.4.0</version>
90                     </dependency>
91                     <dependency>
92                         <groupId>mysql</groupId>
93                         <artifactId>mysql-connector-java</artifactId>
94                         <version>5.1.38</version>
95                     </dependency>
96                 </dependencies>
97             </plugin>
98         </plugins>
99     </build>
100 </project>

```

注意，这里添加了plugin插件部分的配置

## 2、在配置文件中对插件进行配置

在src/main/resources中，新建文件 `generatorConfig.xml`

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE generatorConfiguration
3     PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
4     "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
5

```

```

6 <generatorConfiguration>
7
8 <!--
9     MyBatis3可以生成动态sql
10     <context id="MyBatis3" targetRuntime="MyBatis3" defaultModelType="flat">
11
12     MyBatis3Simple可以生成简单的sql
13     <context id="MyBatis3Simple" targetRuntime="MyBatis3Simple"
defaultModelType="flat">
14         -->
15     <context id="MyBatis3" targetRuntime="MyBatis3" defaultModelType="flat">
16
17         <!-- 防止xml追加，默认再次生成的xml文件会追加内容 -->
18         <plugin type="org.mybatis.generator.plugins.UnmergeableXmlMappersPlugin">
</plugin>
19
20         <!-- 取消所有注释 -->
21         <commentGenerator>
22             <property name="suppressAllComments" value="true" />
23             <property name="suppressDate" value="true" />
24         </commentGenerator>
25
26         <!-- 配置连接数据的相关属性 -->
27         <jdbcConnection driverClass="com.mysql.jdbc.Driver"
connectionURL="jdbc:mysql://localhost:3306/estore?
useUnicode=true&characterEncoding=utf8&useSSL=false" userId="root"
password="root" />
28
29         <!-- 指定生成的类型为java类型，避免数据库中number等类型字段 -->
30         <javaTypeResolver>
31             <property name="forceBigDecimals" value="false" />
32         </javaTypeResolver>
33
34         <!-- 对应的实体类 -->
35         <javaModelGenerator targetPackage="com.briup.generator.bean"
targetProject="src/main/java">
36             </javaModelGenerator>
37
38         <!--对应的XXXMapper.xml文件 -->
39         <sqlMapGenerator targetPackage="com.briup.generator.mapper.basic"
targetProject="src/main/java">
40             </sqlMapGenerator>
41
42         <!-- 对应的XXXMapper.java文件 -->
43         <javaClientGenerator type="XMLMAPPER"
targetPackage="com.briup.generator.dao.basic" targetProject="src/main/java">
44             </javaClientGenerator>
45
46         <!-- 查找数据库中es_开头的表，并生成对应的实体类、映射接口、映射文件 -->
47         <table tableName="es_%" enableCountByExample="false"
enableDeleteByExample="false" enableSelectByExample="false"
enableUpdateByExample="false">
48             <!-- 生成的文件名字中去掉Es开头的前缀，注意这个首字母是大写的 -->
49             <domainObjectRenamingRule searchString="^Es" replaceString="" />
50         </table>
51
52         <!-- 也可以单独一张一张表的进行生成，并指定生成的实体类的名字 -->
53         <!--<table tableName="t_user" domainObjectName="User" ></table>-->

```

```
54     </context>
55     </generatorConfiguration>
56
```

此文件中主要配置的内容有：

- mybatis-generator插件的配置，例如生成的模式、注释的处理方式、xml文件是否追加、生成的字段类型等
- 连接数据的基本信息
- 生成实体类、映射接口、映射文件的位置
- 对数据库中哪些表进行解析并生成

注意，了解文件中每部分的内容，将来使用的时候直接把配置模板复制并进行修改即可

可以指定任意位置来存放生成的实体类、映射接口、映射文件，当前例子中指定的位置如下：

```

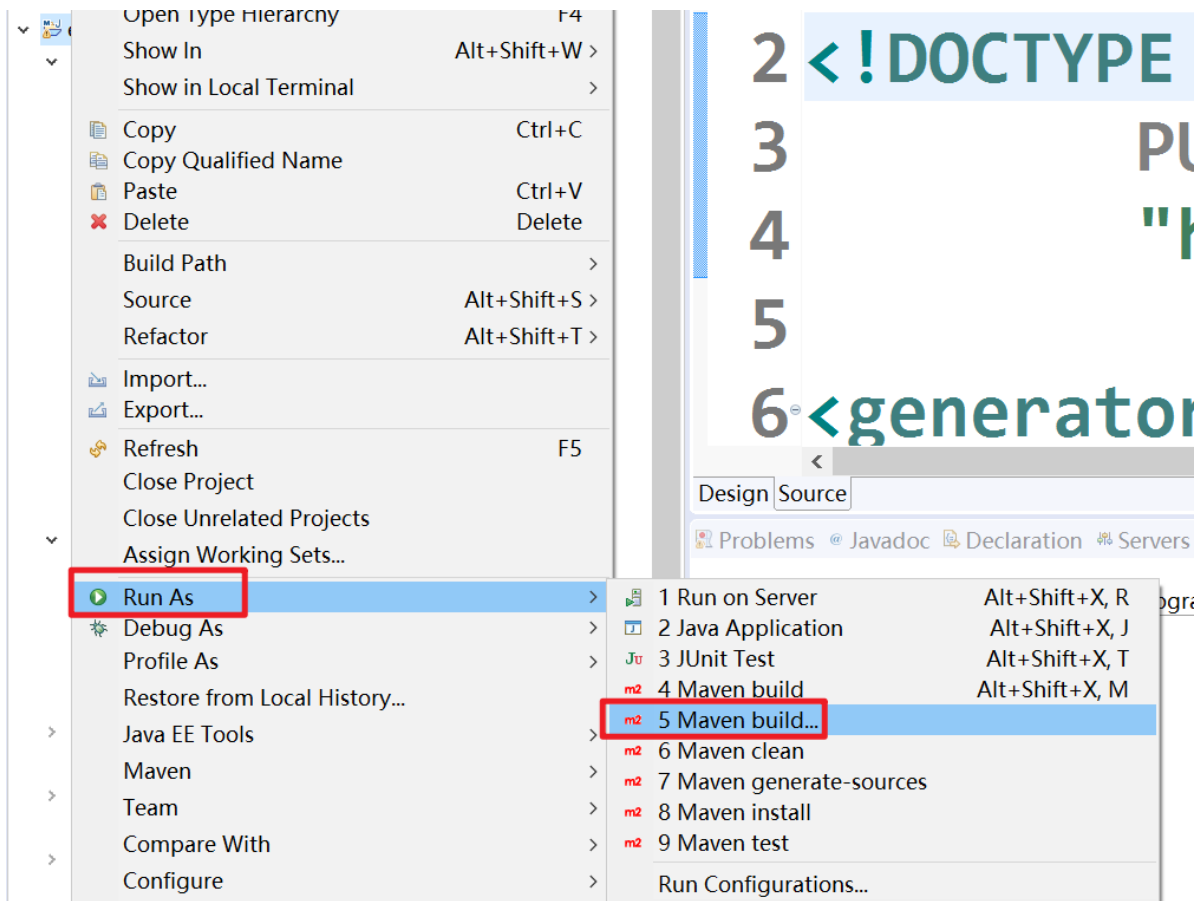
  ▾ 包 generator
    ▾ 包 bean
    ▾ 包 dao
      ▾ 包 basic
      ▾ 包 extend
    ▾ 包 mapper
      ▾ 包 basic
      ▾ 包 extend
    .
```

注意，这里指定了一个叫basic的包，另外还有一个extend的包，这是为了把自动生成的映射文件、映射接口和将来我们定义的映射文件和接口分开，方便维护

### 3、执行该maven插件的goal目标

执行mybatis-generator插件的目录，读取数据库中表的信息，然后反向生成对应的实体类、映射接口、映射文件。

```
mybatis-generator:generate
```



Base directory:

Goals:

Profiles:

User settings:

☐ Offline ☐ Update Snapshots

☐ Debug Output ☐ Skip Tests ☐ Non-recursive

☐ Resolve Workspace artifacts

Threads

Parameter ...	Value

Maven Runtime:

执行目标后，刷新项目，即可看到生成的实体类、映射接口、映射文件：

- ▼ 包 generator
  - ▼ 包 bean
    - > 文件 Book.java
    - > 文件 Category.java
    - > 文件 Customer.java
    - > 文件 Orderform.java
    - > 文件 Orderline.java
    - > 文件 Shopaddress.java
  - ▼ 包 dao
    - ▼ 包 basic
      - > 文件 BookMapper.java
      - > 文件 CategoryMapper.java
      - > 文件 CustomerMapper.java
      - > 文件 OrderformMapper.java
      - > 文件 OrderlineMapper.java
      - > 文件 ShopaddressMapper.java
    - 包 extend
  - ▼ 包 mapper
    - ▼ 包 basic
      - 文件 BookMapper.xml
      - 文件 CategoryMapper.xml
      - 文件 CustomerMapper.xml
      - 文件 OrderformMapper.xml
      - 文件 OrderlineMapper.xml
      - 文件 ShopaddressMapper.xml
    - 包 extend

### 3 补充

---

mybatis-generator插件自动生成的映射文件和映射接口，只能完成对某张表中数据的基本操作，很难完成所有的业务功能的需求。

除了之外，我们还应该编写自己的映射接口和映射文件，用来满足系统中其他功能的需要。

所以，可以将自动生成的文件和接口存放到basic目录中，将自己编写的文件和接口存放到extend目录中。按照这个思路，你也可以指定自己需要的目录来存放自动生成和单独扩展定义的文件和接口。

