**基本配置**

本部分配置包含了大部分用户的常用配置，其中一部分为 MyBatis 原生所支持的配置。

**configLocation**

* 类型：String
* 默认值：null

MyBatis 配置文件位置，如果您有单独的 MyBatis 配置，请将其路径配置到 configLocation 中。 MyBatis Configuration 的具体内容请参考[MyBatis 官方文档](http://www.mybatis.org/mybatis-3/zh/configuration.html)

Spring Boot：

mybatis-plus:

config-location: classpath:mybatis-config.xml

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configLocation" value="classpath:mybatis-config.xml"/>

</bean>

**mapperLocations**

* 类型：String[]
* 默认值：[]

MyBatis Mapper 所对应的 XML 文件位置，如果您在 Mapper 中有自定义方法（XML 中有自定义实现），需要进行该配置，告诉 Mapper 所对应的 XML 文件位置。

Spring Boot：

mybatis-plus:

mapper-locations: classpath\*:mybatis/\*.xml

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="mapperLocations" value="classpath\*:mybatis/\*.xml"/>

</bean>

::: warning Maven 多模块项目的扫描路径需以 classpath\*: 开头 （即加载多个 jar 包下的 XML 文件） :::

**typeAliasesPackage**

* 类型：String
* 默认值：null

MyBaits 别名包扫描路径，通过该属性可以给包中的类注册别名，注册后在 Mapper 对应的 XML 文件中可以直接使用类名，而不用使用全限定的类名（即 XML 中调用的时候不用包含包名）。

Spring Boot：

mybatis-plus:

type-aliases-package: com.baomidou.mybatisplus.samples.quickstart.entity

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="typeAliasesPackage" value="com.baomidou.mybatisplus.samples.quickstart.entity"/>

</bean>

**typeAliasesSuperType**

* 类型：Class<?>
* 默认值：null

该配置请和 [typeAliasesPackage](https://github.com/baomidou/mybatis-plus-doc/tree/master/config#typealiasespackage) 一起使用，如果配置了该属性，则仅仅会扫描路径下以该类作为父类的域对象 。

Spring Boot：

mybatis-plus:

type-aliases-package: com.baomidou.mybatisplus.samples.quickstart.entity

type-aliases-super-type: java.lang.Object

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="typeAliasesPackage" value="com.baomidou.mybatisplus.samples.quickstart.entity"/>

<property name="typeAliasesSuperType" value="java.lang.Object"/>

</bean>

**typeHandlersPackage**

* 类型：String
* 默认值：null

TypeHandler 扫描路径，如果配置了该属性，SqlSessionFactoryBean 会把该包下面的类注册为对应的 TypeHandler。

::: tip TypeHandler 通常用于自定义类型转换。 :::

Spring Boot：

mybatis-plus:

type-handlers-package: com.baomidou.mybatisplus.samples.quickstart.handler

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="typeHandlersPackage" value="com.baomidou.mybatisplus.samples.quickstart.handler"/>

</bean>

**typeEnumsPackage**

* 类型：String
* 默认值：null

枚举类 扫描路径，如果配置了该属性，会将路径下的枚举类进行注入，让实体类字段能够简单快捷的使用枚举属性，具体使用请结合 [枚举注入](https://github.com/baomidou/mybatis-plus-doc/blob/master/guide/enum.md) 查看。

Spring Boot：

mybatis-plus:

type-enums-package: com.baomidou.mybatisplus.samples.quickstart.enums

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="typeEnumsPackage" value="com.baomidou.mybatisplus.samples.quickstart.enums"/>

</bean>

**checkConfigLocation**

* 类型：boolean
* 默认值：false

启动时是否检查 MyBatis XML 文件的存在，默认不检查。

Spring Boot：

mybatis-plus:

check-config-location: false

**executorType**

* 类型：ExecutorType
* 默认值：sample

通过该属性可指定 MyBatis 的执行器，MyBatis 的执行器总共有三种：

* ExecutorType.SIMPLE：该执行器类型不做特殊的事情，为每个语句的执行创建一个新的预处理语句（PreparedStatement）
* ExecutorType.REUSE：该执行器类型会复用预处理语句（PreparedStatement）
* ExecutorType.BATCH：该执行器类型会批量执行所有的更新语句

Spring Boot：

mybatis-plus:

executor-type: simple

**configurationProperties**

* 类型：Properties
* 默认值：null

指定外部化 MyBatis Properties 配置，通过该配置可以抽离配置，实现不同环境的配置部署。

Spring Boot：

mybatis-plus:

configuration-properties: classpath:mybatis/config.properties

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configurationProperties" value="classpath:mybatis/config.properties"/>

</bean>

**configuration**

* 类型：Configuration
* 默认值：null

原生 MyBatis 所支持的配置，具体请查看 [进阶配置](https://github.com/baomidou/mybatis-plus-doc/tree/master/config#%E8%BF%9B%E9%98%B6%E9%85%8D%E7%BD%AE)

**globalConfig**

* 类型：GlobalConfig
* 默认值：null

MyBatis-Plus 全局策略配置，具体请查看 [全局策略配置](https://github.com/baomidou/mybatis-plus-doc/tree/master/config#%E5%85%A8%E5%B1%80%E7%AD%96%E7%95%A5%E9%85%8D%E7%BD%AE)

**进阶配置**

本部分（Configuration）的配置大都为 MyBatis 原生支持的配置，这意味着您可以通过 MyBatis XML 配置文件的形式进行配置。

**mapUnderscoreToCamelCase**

* 类型：boolean
* 默认值：true

是否开启自动驼峰命名规则（camel case）映射，即从经典数据库列名 A\_COLUMN（下划线命名） 到经典 Java 属性名 aColumn（驼峰命名） 的类似映射。

::: tip 注意 此属性在 MyBatis 中原默认值为 false，在 MyBatis-Plus 中，此属性也将用于生成最终的 SQL 的 select body

如果您的数据库命名符合规则无需使用 @TableField 注解指定数据库字段名 :::

**aggressiveLazyLoading**

* 类型：boolean
* 默认值：true

当设置为 true 的时候，懒加载的对象可能被任何懒属性全部加载，否则，每个属性都按需加载。需要和 [lazyLoadingEnabled](https://github.com/baomidou/mybatis-plus-doc/blob/master/config) 一起使用。

Spring Boot：

mybatis-plus:

configuration:

aggressive-lazy-loading: true

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configuration">

<bean class="com.baomidou.mybatisplus.core.MybatisConfiguration">

<property name="aggressiveLazyLoading" value="true"/>

</bean>

</property>

</bean>

**autoMappingBehavior**

* 类型：AutoMappingBehavior
* 默认值：partial

MyBatis 自动映射策略，通过该配置可指定 MyBatis 是否并且如何来自动映射数据表字段与对象的属性，总共有 3 种可选值：

* AutoMappingBehavior.NONE：不启用自动映射
* AutoMappingBehavior.PARTIAL：只对非嵌套的 resultMap 进行自动映射
* AutoMappingBehavior.FULL：对所有的 resultMap 都进行自动映射

Spring Boot：

mybatis-plus:

configuration:

auto-mapping-behavior: partial

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configuration">

<bean class="com.baomidou.mybatisplus.core.MybatisConfiguration">

<property name="autoMappingBehavior" value="PARTIAL"/>

</bean>

</property>

</bean>

**autoMappingUnknownColumnBehavior**

* 类型：AutoMappingUnknownColumnBehavior
* 默认值：NONE

MyBatis 自动映射时未知列或未知属性处理策略，通过该配置可指定 MyBatis 在自动映射过程中遇到未知列或者未知属性时如何处理，总共有 3 种可选值：

* AutoMappingUnknownColumnBehavior.NONE：不做任何处理 (默认值)
* AutoMappingUnknownColumnBehavior.WARNING：以日志的形式打印相关警告信息
* AutoMappingUnknownColumnBehavior.FAILING：当作映射失败处理，并抛出异常和详细信息

Spring Boot：

mybatis-plus:

configuration:

auto-mapping-unknown-column-behavior: none

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configuration">

<bean class="com.baomidou.mybatisplus.core.MybatisConfiguration">

<property name="autoMappingUnknownColumnBehavior" value="NONE"/>

</bean>

</property>

</bean>

**cacheEnabled**

* 类型：boolean
* 默认值：true

全局地开启或关闭配置文件中的所有映射器已经配置的任何缓存，默认为 true。

Spring Boot：

mybatis-plus:

configuration:

cache-enabled: true

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configuration">

<bean class="com.baomidou.mybatisplus.core.MybatisConfiguration">

<property name="cacheEnabled" value="true"/>

</bean>

</property>

</bean>

**callSettersOnNulls**

* 类型：boolean
* 默认值：false

指定当结果集中值为 null 的时候是否调用映射对象的 Setter（Map 对象时为 put）方法，通常运用于有 Map.keySet() 依赖或 null 值初始化的情况。

通俗的讲，即 MyBatis 在使用 resultMap 来映射查询结果中的列，如果查询结果中包含空值的列，则 MyBatis 在映射的时候，不会映射这个字段，这就导致在调用到该字段的时候由于没有映射，取不到而报空指针异常。

当您遇到类似的情况，请针对该属性进行相关配置以解决以上问题。

::: warning 基本类型（int、boolean 等）是不能设置成 null 的。 :::

Spring Boot：

mybatis-plus:

configuration:

call-setters-on-nulls: false

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="configuration">

<bean class="com.baomidou.mybatisplus.core.MybatisConfiguration">

<property name="callSettersOnNulls" value="false"/>

</bean>

</property>

</bean>

**configurationFactory**

* 类型：Class<?>
* 默认值：null

指定一个提供 Configuration 实例的工厂类。该工厂生产的实例将用来加载已经被反序列化对象的懒加载属性值，其必须包含一个签名方法static Configuration getConfiguration()。（从 3.2.3 版本开始）

Spring Boot：

mybatis-plus:

configuration:

configuration-factory: com.xxx.SampleConfigurationFactory

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="configurationFactory" value="com.xxx.SampleConfigurationFactory"/>

</bean>

</property>

</bean>

**全局策略配置**

**refresh**

* 类型：boolean
* 默认值：false

是否自动刷新 Mapper 对应的 XML 文件，默认不自动刷新。如果配置了该属性，Mapper 对应的 XML 文件会自动刷新，更改 XML 文件后，无需再次重启工程，由此节省大量时间。

::: warning 该配置不建议在生产环境打开！ :::

Spring Boot：

mybatis-plus:

global-config:

refresh: true

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="refresh" value="true"/>

</bean>

</property>

</bean>

**sqlParserCache**

* 类型：boolean
* 默认值：false

是否缓存 Sql 解析，默认不缓存。

Spring Boot：

mybatis-plus:

global-config:

sql-parser-cache: true

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="sqlParserCache" value="true"/>

</bean>

</property>

</bean>

**sqlSession**

* 类型：SqlSession
* 默认值：null

单例重用 SqlSession。

Spring Boot：

mybatis-plus:

global-config:

sql-session: com.xxx.SqlSession

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="sqlSession" value="com.xxx.SqlSession"/>

</bean>

</property>

</bean>

**sqlSessionFactory**

* 类型：SqlSessionFactory
* 默认值：null

缓存当前 Configuration 的 SqlSessionFactory。

Spring Boot：

mybatis-plus:

global-config:

sql-session-factory: com.xxx.SqlSessionFactory

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="sqlSessionFactory" value="com.xxx.SqlSessionFactory"/>

</bean>

</property>

</bean>

**dbConfig**

* 类型：com.baomidou.mybatisplus.annotation.DbConfig
* 默认值：null

MyBatis-Plus 全局策略中的 DB 策略配置，具体请查看 [DB 策略配置](https://github.com/baomidou/mybatis-plus-doc/tree/master/config#DB%E7%AD%96%E7%95%A5%E9%85%8D%E7%BD%AE)

**DB 策略配置**

**capitalMode**

* 类型：boolean
* 默认值：false

是否开启大写命名，默认不开启。

Spring Boot：

mybatis-plus:

global-config:

db-config:

capital-mode: false

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="capitalMode" value="false"/>

</bean>

</property>

</bean>

</property>

</bean>

**columnLike**

* 类型：boolean
* 默认值：false

是否开启 LIKE 查询，即根据 entity 自动生成的 where 条件中 String 类型字段 是否使用 LIKE，默认不开启。

Spring Boot：

mybatis-plus:

global-config:

db-config:

column-like: false

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="columnLike" value="false"/>

</bean>

</property>

</bean>

</property>

</bean>

**columnUnderline**

::: danger 注意 此属性存在于 2.x 版本上,现同 [mapUnderscoreToCamelCase](https://github.com/baomidou/mybatis-plus-doc/tree/master/config#mapunderscoretocamelcase) 融合 :::

**dbType**

* 类型：com.baomidou.mybatisplus.annotation.DbType
* 默认值：OTHER

数据库类型,默认值为未知的数据库类型 如果值为OTHER,启动时会根据数据库连接 url 获取数据库类型;如果不是OTHER则不会自动获取数据库类型

Spring Boot：

mybatis-plus:

global-config:

db-config:

db-type: mysql

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="dbType" value="mysql"/>

</bean>

</property>

</bean>

</property>

</bean>

**fieldStrategy**

* 类型：com.baomidou.mybatisplus.annotation.FieldStrategy
* 默认值：NOT\_NULL

字段验证策略

::: tip 说明: 该策略约定了如何产出注入的sql,涉及insert,update以及wrapper内部的entity属性生成的 where 条件 :::

Spring Boot：

mybatis-plus:

global-config:

db-config:

field-strategy: not\_null

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="fieldStrategy" value="NOT\_NULL"/>

</bean>

</property>

</bean>

</property>

</bean>

**idType**

* 类型：com.baomidou.mybatisplus.annotation.IdType
* 默认值：ID\_WORKER

全局默认主键类型

Spring Boot：

mybatis-plus:

global-config:

db-config:

id-type: id\_worker

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="idType" value="ID\_WORKER"/>

</bean>

</property>

</bean>

</property>

</bean>

**logicDeleteValue**

* 类型：String
* 默认值：1

逻辑已删除值,([逻辑删除](https://github.com/baomidou/mybatis-plus-doc/blob/master/guide/logic-delete.md)下有效)

Spring Boot：

mybatis-plus:

global-config:

db-config:

logic-delete-value: 1

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="logicDeleteValue" value="1"/>

</bean>

</property>

</bean>

</property>

</bean>

**logicNotDeleteValue**

* 类型：String
* 默认值：0

逻辑未删除值,([逻辑删除](https://github.com/baomidou/mybatis-plus-doc/blob/master/guide/logic-delete.md)下有效)

Spring Boot：

mybatis-plus:

global-config:

db-config:

logic-not-delete-value: 0

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="logicNotDeleteValue" value="0"/>

</bean>

</property>

</bean>

</property>

</bean>

**tablePrefix**

* 类型：String
* 默认值：null

表名前缀

Spring Boot：

mybatis-plus:

global-config:

db-config:

table-prefix: xx\_

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="tablePrefix" value="xx\_"/>

</bean>

</property>

</bean>

</property>

</bean>

**tableUnderline**

* 类型：boolean
* 默认值：true

表名、是否使用下划线命名，默认数据库表使用下划线命名

Spring Boot：

mybatis-plus:

global-config:

db-config:

table-underline: true

Spring MVC：

<bean id="sqlSessionFactory" class="com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean">

<property name="globalConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig">

<property name="dbConfig">

<bean class="com.baomidou.mybatisplus.core.config.GlobalConfig.DbConfig">

<property name="tableUnderline" value="true"/>

</bean>

</property>

</bean>

</property>

</bean>