快速开始参考：http://mp.baomidou.com/guide/quick-start.html

测试项目： mybatis_plus

数据库：mybatis_plus

一、创建并初始化数据库

1、创建数据库：

mybatis_plus

2、创建 User 表

其表结构如下：

id	name	age	email
1	Jone	18	test1@baomidou.com
2	Jack	20	test2@baomidou.com
3	Tom	28	test3@baomidou.com
4	Sandy	21	test4@baomidou.com
5	Billie	24	test5@baomidou.com

其对应的数据库 Schema 脚本如下：

```
DROP TABLE IF EXISTS user;

CREATE TABLE user
(
    id BIGINT(20) NOT NULL COMMENT '主键ID',
    name VARCHAR(30) NULL DEFAULT NULL COMMENT '姓名',
    age INT(11) NULL DEFAULT NULL COMMENT '年龄',
    email VARCHAR(50) NULL DEFAULT NULL COMMENT '邮箱',
    PRIMARY KEY (id)
);
```

其对应的数据库 Data 脚本如下：

```sql
DELETE FROM user;

INSERT INTO user (id, name, age, email) VALUES
(1, 'Jone', 18, 'test1@baomidou.com'),
(2, 'Jack', 20, 'test2@baomidou.com'),
(3, 'Tom', 28, 'test3@baomidou.com'),
(4, 'Sandy', 21, 'test4@baomidou.com'),
(5, 'Billie', 24, 'test5@baomidou.com');
```

二、初始化工程

使用 Spring Initializr 快速初始化一个 Spring Boot 工程

Group：com.atguigu

Artifact：mybatis-plus

版本：2.2.1.RELEASE

三、添加依赖

1、引入依赖

spring-boot-starter、spring-boot-starter-test

添加：mybatis-plus-boot-starter、MySQL、lombok、

在项目中使用Lombok可以减少很多重复代码的书写。比如说getter/setter/toString等方法的编写

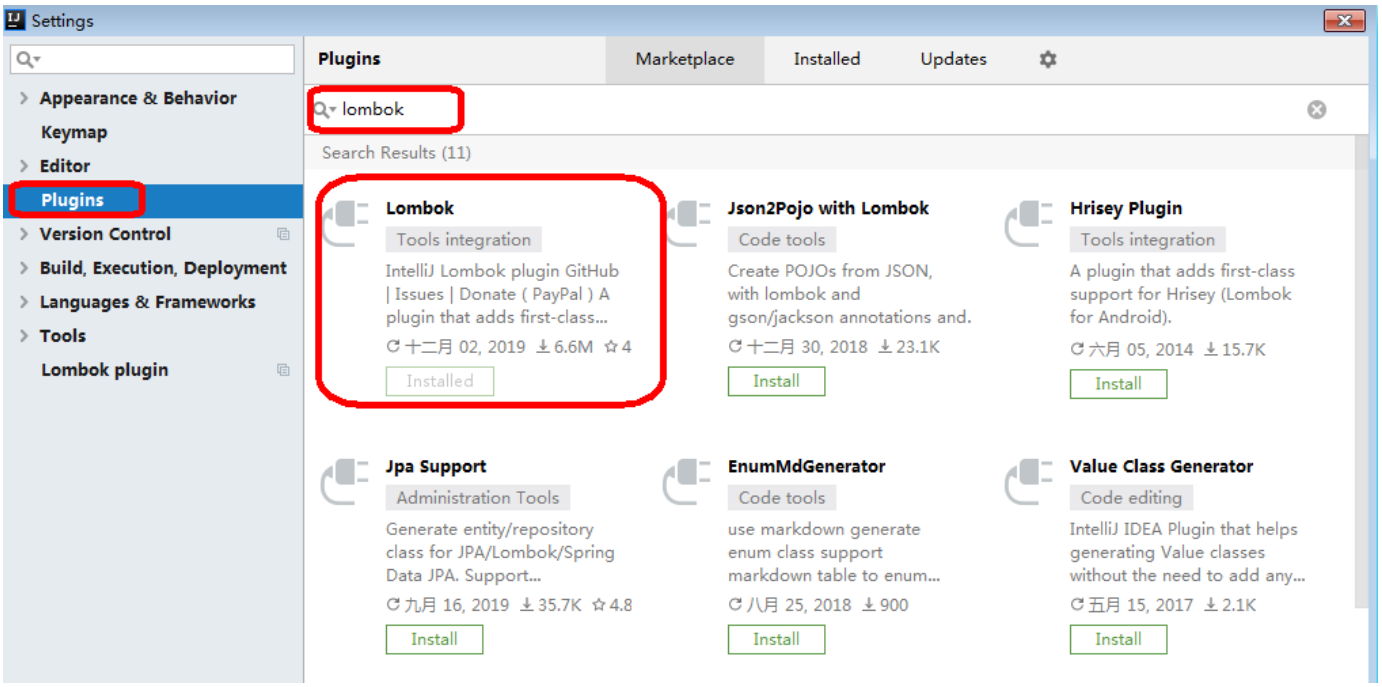
```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>
```

```
6
7       <dependency>
8           <groupId>org.springframework.boot</groupId>
9           <artifactId>spring-boot-starter-test</artifactId>
10          <scope>test</scope>
11       </dependency>
12
13      <!--mybatis-plus-->
14      <dependency>
15          <groupId>com.baomidou</groupId>
16          <artifactId>mybatis-plus-boot-starter</artifactId>
17          <version>3.3.1</version>
18      </dependency>
19
20      <!--mysql-->
21      <dependency>
22          <groupId>mysql</groupId>
23          <artifactId>mysql-connector-java</artifactId>
24      </dependency>
25
26      <!--lombok用来简化实体类-->
27      <dependency>
28          <groupId>org.projectlombok</groupId>
29          <artifactId>lombok</artifactId>
30      </dependency>
31  </dependencies>
```

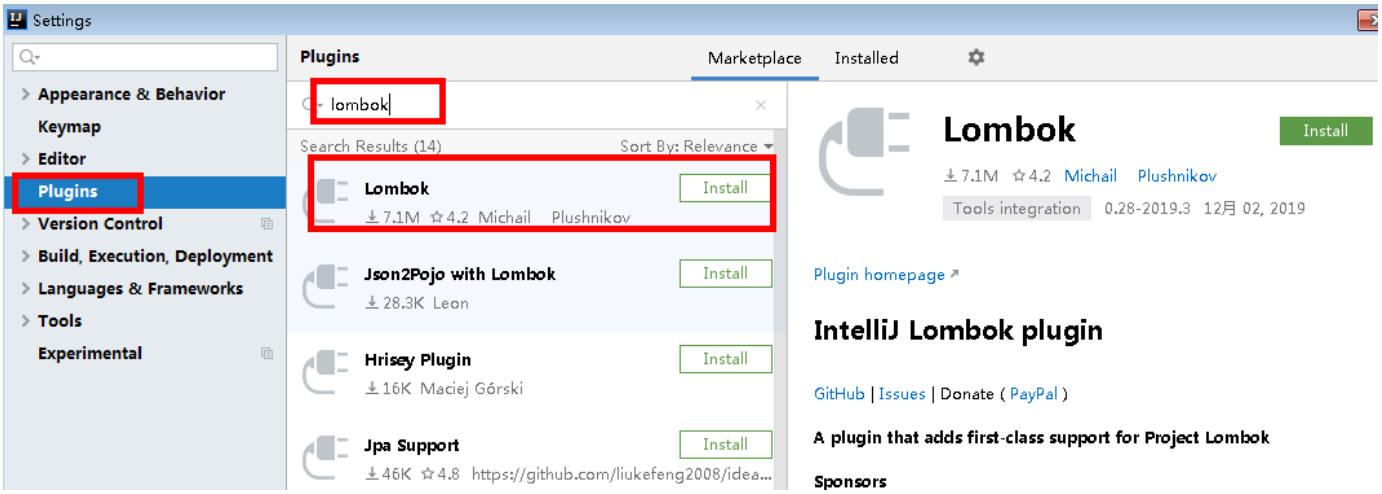
注意：引入 MyBatis-Plus 之后请不要再次引入 MyBatis 以及 MyBatis-Spring，以避免因版本差异导致的问题。

2、idea中安装lombok插件

（1）idea2018版本



（2）idea2019版本



四、配置

在 application.properties 配置文件中添加 MySQL 数据库的相关配置：

mysql5

```
#mysql数据库连接
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mybatis_plus
spring.datasource.username=root
spring.datasource.password=123456
```

mysql8以上（spring boot 2.1）

注意：driver和url的变化

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mybatis_plus?serverTimezone=GMT%2B8
spring.datasource.username=root
spring.datasource.password=123456
```

注意：

1、这里的 url 使用了 ?serverTimezone=GMT%2B8 后缀，因为Spring Boot 2.1 集成了 8.0版本的jdbc驱动，这个版本的 jdbc 驱动需要添加这个后缀，否则运行测试用例报告如下错误：

java.sql.SQLException: The server time zone value 'ÖÐ¹ú±ê×¼Ê±¼ä' is unrecognized or represents more

2、这里的 driver-class-name 使用了 com.mysql.cj.jdbc.Driver ，在 jdbc 8 中 建议使用这个驱动，之前的 com.mysql.jdbc.Driver 已经被废弃，否则运行测试用例的时候会有 WARN 信息

五、编写代码

1、主类

在 Spring Boot 启动类中添加 @MapperScan 注解，扫描 Mapper 文件夹

注意：扫描的包名根据实际情况修改

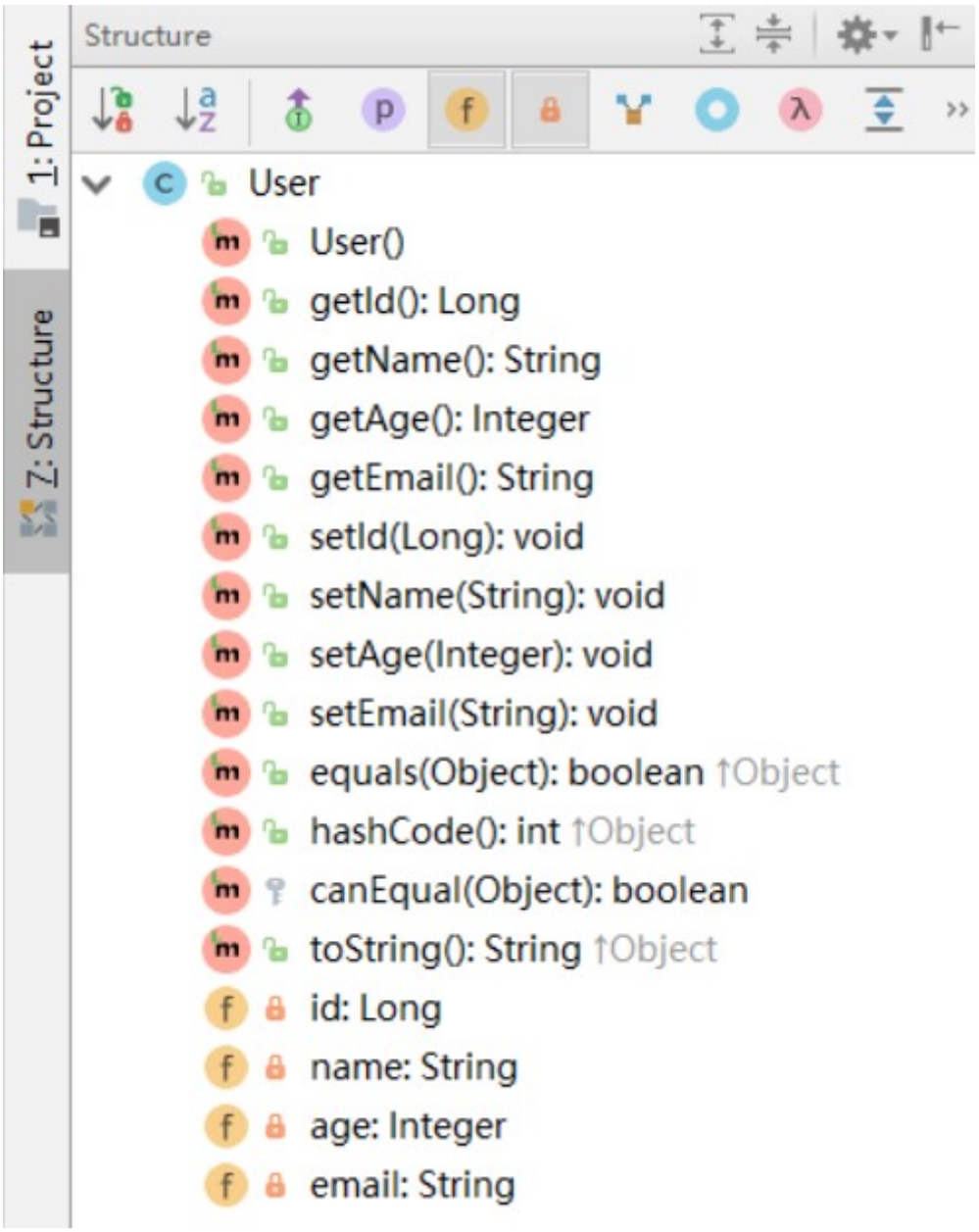
```
@SpringBootApplication
@MapperScan("com.atguigu.mybatisplus.mapper")
public class MybatisPlusApplication {
    ......
}
```

2、实体

创建包 entity 编写实体类 User.java（此处使用了 Lombok 简化代码）

```java
@Data
public class User {
    private Long id;
    private String name;
    private Integer age;
    private String email;
}
```

查看编译结果



Lombok使用参考：

https://blog.csdn.net/motui/article/details/79012846

3、mapper

创建包 mapper 编写Mapper 接口： UserMapper.java

```
public interface UserMapper extends BaseMapper<User> {

}
```

六、开始使用

添加测试类，进行功能测试：

```
public class MybatisPlusApplicationTests {

    @Autowired
    private UserMapper userMapper;

    @Test
    public void testSelectList() {
        System.out.println(("----- selectAll method test ------"));
        //UserMapper 中的 selectList() 方法的参数为 MP 内置的条件封装器 Wrapper
        //所以不填写就是无任何条件
        List<User> users = userMapper.selectList(null);
        users.forEach(System.out::println);
    }
}
```

注意：

IDEA在 userMapper 处报错，因为找不到注入的对象，因为类是动态创建的，但是程序可以正确的执行。

为了避免报错，可以在 dao 层 的接口上添加 @Repository 注

控制台输出：

```
User(id=1, name=Jone, age=18, email=test1@baomidou.com)
User(id=2, name=Jack, age=20, email=test2@baomidou.com)
User(id=3, name=Tom, age=28, email=test3@baomidou.com)
```

```
User(id=4, name=Sandy, age=21, email=test4@baomidou.com)
User(id=5, name=Billie, age=24, email=test5@baomidou.com)
```

通过以上几个简单的步骤，我们就实现了 User 表的 CRUD 功能，甚至连 XML 文件都不用编写！

七、配置日志

查看sql输出日志

```
#mybatis日志
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```