

😊 纯洁的微笑 (/)

[Home \(/\)](#) [Spring-Boot \(/spring-boot.html\)](#)

[mybatis-spring-boot-starter](#)

[Spring-Cloud \(/spring-cloud.html\)](#) [Archives \(/archives.html\)](#)

[配置文件注解版](#)

- 1 添加相关maven文件

[Link \(/link.html\)](#) [About \(/about.html\)](#)

- 2、application.properties 添加相

关配置

- 3、开发Mapper

- 4、使用

springboot(六): 如何优雅的使用mybatis

[极简xml版本](#)

📅 2016/11/06

- 1、配置

- 2、添加User的映射文件

- 3、编写Dao层的代码

- 4、使用

[如何选择](#)

这两天启动了一个新项目因为项目组成员一直都使用的是mybatis，虽然个人比较喜欢jpa这种极简的模式，但是为了项目保持统一性技术选型还是定了 mybatis。到网上找了一下关于spring boot和mybatis组合的相关资料，各种各样的形式都有，看的人心累，结合了mybatis的官方demo和文档终于找到了最简的两种模式，花了一天时间总结后分享出来。

orm框架的本质是简化编程中操作数据库的编码，发展到现在基本上就剩两家了，一个是宣称可以不用写一句SQL的hibernate，一个是可以灵活调试动态sql的mybatis,两者各有特点，在企业级系统开发中可以根据需求灵活使用。发现一个有趣的现象：传统企业大都喜欢使用hibernate,互联网行业通常使用mybatis。

hibernate特点就是所有的sql都用Java代码来生成,不用跳出程序去写(看)sql,有着编程的完整性,发展到最顶端就是spring data jpa这种模式了,基本上根据方法名就可以生成对应的sql了,有不太了解的可以看我的上篇文章springboot(五): spring data jpa的使用(<http://www.ityouknow.com/springboot/2016/08/20/spring-boo-jpa.html>)。

mybatis-spring-boot-starter

mybatis初期使用比较麻烦,需要各种配置文件、实体类、dao层映射关联、还有一大堆其它配置。当然mybatis也发现了这种弊端,初期开发了generator (<https://github.com/mybatis/generator>)可以根据表结果自动生产实体类、配置文件和dao层代码,可以减轻一部分开发量;后期也进行了大量的优化可以使用注解了,自动管理dao层和配置文件等,发展到最顶端就是今天要讲的这种模式了,mybatis-spring-boot-starter就是springboot+mybatis可以完全注解不用配置文件,也可以简单配置轻松上手

极简xml版本

现在想想Spring boot 就是牛逼呀,任何东西只要关联到spring boot都是化繁为简。

- 1、配置

- 2、添加User的映射文件

mybatis-spring-boot-starter

- 3、编写Dao层的代码

- 4、使用

官方说明: MyBatis Spring-Boot-Starter will help you use MyBatis with Spring Boot

如何选择

其实就是myBatis看spring boot这么火热也开发出一套解决方案来凑凑热闹,但这一凑确实解决了很多问题,使用起来确实顺畅了许多。mybatis-spring-boot-starter主要有两种解决方案,一种是使用注解解决一切问题,一种是简化后的老传统。

当然任何模式都需要首先引入mybatis-spring-boot-starter的pom文件,现在最新版本是1.1.1 (刚好快到双11了 :))

```
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>1.1.1</version>
</dependency>
```

好了下来分别介绍两种开发模式

无配置文件注解版

就是使用注解搞定。

无配置文件注解版

添加相关maven文件

- 2、application.properties 添加相关配置
- 3、开发Mapper
- 4、使用

极简xml版本

- 1、配置
- 2、添加User的映射文件
- 3、编写Dao层的代码
- 4、使用

如何选择

```

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>1.1.1</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>

```

mybatis-spring-boot-starter

无配置文件注解版

- 1 添加相关maven文件
- 2、application.properties 添加相关配置
- 3、开发Mapper
- 4、使用

极简xml版本

- 1、配置
- 2、添加User的映射文件
- 3、编写Dao层的代码
- 4、使用

如何选择

完整的pom包这里就不贴了，大家直接看源码

2、 application.properties 添加相关配置

```
mybatis.type-aliases-package=com.neo.entity
```

```
spring.datasource.driverClassName = com.mysql.jdbc.Driver
```

```
spring.datasource.url = jdbc:mysql://localhost:3306/test1?useUnicode=true&characterEncoding=utf-8
```

```
spring.datasource.username = root
```

```
spring.datasource.password = root
```

mybatis-spring-boot-starter

无配置文件注解版

- 1 添加相关maven文件

springboot会自动加载spring.datasource.*相关配置, 数据源就会自动注入到sqlSessionFactory中,

- 2、application.properties添加相

sqlSessionFactory会自动注入到Mapper中, 对了你一切都不用管了, 直接拿起来使用就行了。

- 3、开发Mapper

在启动类中添加对mapper包扫描 @MapperScan

- 4、使用

极简xml版本

@SpringBootApplication

@MapperScan("com.neo.mapper")

public class Application {

- 3、编写Dao层的代码

- 4、使用 **public static void main(String[] args) {**

SpringApplication.run(Application.class, args);

如何选择

}

}

或者直接在Mapper类上面添加注解 @Mapper ,建议使用上面那种, 不然每个mapper加个注解也挺麻烦的

3、开发Mapper

第三步是最关键的一块, sql生产都在这里

```

public interface UserMapper {

    @Select("SELECT * FROM users")
    @Results({
        @Result(property = "userSex", column = "user_sex", javaType = UserSexEnum.class),
        @Result(property = "nickName", column = "nick_name")
    })
    List<UserEntity> getAll();

    @Select("SELECT * FROM users WHERE id = #{id}")
    @Results({
        @Result(property = "userSex", column = "user_sex", javaType = UserSexEnum.class),
        @Result(property = "nickName", column = "nick_name")
    })
    UserEntity getOne(Long id);

    @Insert("INSERT INTO users(userName,passWord,user_sex) VALUES(#{userName}, #{passWord}, #{userSex})")
    void insert(UserEntity user);

    @Update("UPDATE users SET userName=#{userName},nick_name=#{nickName} WHERE id =#{id}")
    void update(UserEntity user);

    @Delete("DELETE FROM users WHERE id =#{id}")
    void delete(Long id);

}

```

为了更接近生产我特地将user_sex、nick_name两个属性在数据库加了下划线和实体类属性名不一致，另外user_sex使用了枚举

- @Select 是查询类的注解，所有的查询均使用这个
- @Result 修饰返回的结果集，关联实体类属性和数据库字段——对应，如果实体类属性和数据库属性名保持一致，就不需要这个属性来修饰。
- @Insert 插入数据库使用，直接传入实体类会自动解析属性到对应的值

mybatis-spring-boot-starter

• @Update 负责修改，也可以直接传入对象

无配置文件注解版

• @delete 负责删除

- 1 添加相关maven文件

了解更多属性参考[这里\(https://mybatis.org/mybatis-3/zh/java-api.html\)](https://mybatis.org/mybatis-3/zh/java-api.html)

关配置

注意，使用#符号和\$符号的不同：

- 4、使用

极简xml版本

*// This example creates a prepared statement, something like select * from teacher where name = ?;*

@Select("select * from teacher where name = #{name}")

Teacher selectTeachForGivenName(@Param("name") String name);

- 2、添加User的映射文件

3、编写Dao层的代码

*// This example creates an inlined statement, something like select * from teacher where name = 'someName';*

@Select("select * from teacher where name = '\${name}'")

Teacher selectTeachForGivenName(@Param("name") String name);

如何选择

4、使用

上面三步就基本完成了相关dao层开发，使用的时候当作普通的类注入进入就可以了

```

@RunWith(SpringRunner.class)
@SpringBootTest
public class UserMapperTest {

    @Autowired
    mybatis-spring-boot-starter
    private UserMapper userMapper;

    无配置文件注解版
    @Test
    public void testInsert() throws Exception {
        - 1 添加@Mapper
        - 2、application.properties 添加相关配置
        - 3、开发Mapper
        - 4、使用
        userMapper.insert(new UserEntity("aa", "a123456", UserSexEnum.MAN));
        userMapper.insert(new UserEntity("bb", "b123456", UserSexEnum.WOMAN));
        userMapper.insert(new UserEntity("cc", "b123456", UserSexEnum.WOMAN));

        极简xml版本
        Assert.assertEquals(3, userMapper.getAll().size());

        - 1、配置
        - 2、添加User的映射文件
        - 3、编写DAO层的代码
        @Test
        public void testQuery() throws Exception {
            - 4、使用
            List<UserEntity> users = userMapper.getAll();
            System.out.println(users.toString());
        }

        如何选择

        @Test
        public void testUpdate() throws Exception {
            UserEntity user = userMapper.getOne(31);
            System.out.println(user.toString());
            user.setNickName("neo");
            userMapper.update(user);
            Assert.assertTrue(("neo".equals(userMapper.getOne(31).getNickName())));
        }
    }
}

```

源码中controller层有完整的增删改查，这里就不贴了

极简xml版本

极简xml版本保持映射文件的老传统, 优化主要体现在不需要实现dao的是实现层, 系统会自动根据方法名在映射文件中找对应的sql.

mybatis-spring-boot-starter

1. 配置

- 1 添加相关maven文件

配置文件和上个版本一样, 只是在application.properties 新增以下配置

相关配置

3. 开发Mapper

mybatis.config-location=classpath:mybatis/mybatis-config.xml

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

极简xml版本

1. 配置

指定了mybatis基础配置文件和实体类映射文件的地址

- 2、添加User的映射文件

mybatis编写配置文件

- 4、使用

如何选择

```
<configuration>
  <typeAliases>
    <typeAlias alias="Integer" type="java.lang.Integer" />
    <typeAlias alias="Long" type="java.lang.Long" />
    <typeAlias alias="HashMap" type="java.util.HashMap" />
    <typeAlias alias="LinkedHashMap" type="java.util.LinkedHashMap" />
    <typeAlias alias="ArrayList" type="java.util.ArrayList" />
    <typeAlias alias="LinkedList" type="java.util.LinkedList" />
  </typeAliases>
</configuration>
```

这里也可以添加一些mybatis基础的配置

2、添加User的映射文件

```

<mapper namespace="com.neo.mapper.UserMapper" >
    <resultMap id="BaseResultMap" type="com.neo.entity.UserEntity" >
        <id column="id" property="id" jdbcType="BIGINT" />
        <result column="userName" property="userName" jdbcType="VARCHAR" />
        <result column="passWord" property="passWord" jdbcType="VARCHAR" />
        <result column="user_sex" property="userSex" javaType="com.neo.enums.UserSexEnum"/>
        <result column="nick_name" property="nickName" jdbcType="VARCHAR" />
    </resultMap>

```

mybatis-spring-boot-starter

无配置文件注解版

- 1、添加相关pom文件
- 2、application.properties 添加相关配置
- 3、开发Mapper
- 4、使用

```

<sql id="Base_Column_List" >
    SELECT
        id, userName, passWord, user_sex, nick_name
    </sql>

```

极简xml版本

- 1、配置
- 2、添加User的映射文件
- 3、编写Dao层的代码
- 4、使用

```

<select id="getOne" parameterType="java.lang.Long" resultMap="BaseResultMap" >
    SELECT

```

如何选择

```

        <include refid="Base_Column_List" />
        FROM users
        WHERE id = #{id}
    </select>

```

```

<insert id="insert" parameterType="com.neo.entity.UserEntity" >
    INSERT INTO
        users
        (userName,passWord,user_sex)
    VALUES
        (#{userName}, #{passWord}, #{userSex})
</insert>

```

```

<update id="update" parameterType="com.neo.entity.UserEntity" >
    UPDATE

```

```

        users
        SET
        <if test="userName != null">userName = #{userName}, </if>
        <if test="passWord != null">passWord = #{passWord}, </if>
        nick_name = #{nickName}
    WHERE
    id = #{id}
</update>
<delete id="delete" parameterType="java.lang.Long" >
    DELETE FROM
    users
    WHERE
    id =#{id}
</delete>
</mapper>

```

mybatis-spring-boot-starter

无配置文件注解版

- 1 添加相关maven文件
- 2、application.properties 添加相关配置
- 3、开发Mapper
- 4、使用

极简xml版本

- 1、配置
- 2、添加User的映射文件

其实就是把上个版本中mapper的sql搬到了这里的xml中了

- 4、使用

3、编写Dao层的代码

```

public interface UserMapper {

    List<UserEntity> getAll();

    UserEntity getOne(Long id);

    void insert(UserEntity user);

    void update(UserEntity user);

    void delete(Long id);

}

```

对比上一步这里全部只剩了接口方法

4、使用

使用和上个版本没有任何区别，大家就看代码吧

mybatis-spring-boot-starter

无配置文件注解版

如何选择

添加相关maven文件

- 2、application.properties 添加相

两种模式各有特点，注解版适合简单快速的模式，其实像现在流行的这种微服务模式，一个微服务就会对应三个自己的数据库，多表连接查询的需求会大大的降低，会越来越适合这种模式。

- 4、使用
老传统模式比适合大型项目，可以灵活的动态生成SQL，方便调整SQL，也有痛痛快快，洋洋洒洒的写SQL的感觉

- 1、配置

2、添加User的映射文件

示例代码-github (<https://github.com/ityouknow/spring-boot-examples>)

- 3、编写Dao层的代码

示例代码-码云 (<https://gitee.com/ityouknow/spring-boot-examples>)

如何选择

作者：纯洁的微笑

出处：www.ityouknow.com (<http://www.ityouknow.com>)

版权所有，欢迎保留原文链接进行转载：)



mybatis-spring-boot-starter

无配置文件注解版

- 1 添加相关maven文件
- 2、application.properties 添加相关配置
- 3、开发Mapper
- 4、使用

极简xml版本

- 1、配置
- 2、添加User的映射文件
- 3、编写Dao层的代码

扫码关注有惊喜

(转载本站文章请注明作者和出处 纯洁的微笑-ityouknow (<http://www.ityouknow.com>))

Show Disqus Comments

如何选择

8 (<https://github.com/ityouknow/blog-comments/issues/156>) 条评论

未登录用户 ▾



说点什么

① 支持 Markdown 语法 (<https://guides.github.com/features/mastering-markdown/>)


使用 Github 登录



wenbing90 (<https://github.com/wenbing90>) 发表于 3 个月前



用generator生成的自带基本查询,并且多生成一个类,但是在改字段的时候就很麻烦,用注解方式的还是不知道怎样动态sql



kuanshujiang (<https://github.com/kuanshujiang>) 发表于 3 个月前

mybatis-spring-boot-starter


你好,请问第二种方式,怎么打印sql语句在控制台,我用了网上的方法,配置了无配置文件注解版

- 1 添加logging.level.mapper包=debug 还是没用
- 2、application.properties 添加相关配置
- 3、开发MyMapper
- 4、使用低调小熊猫就是我,学完去装逼

极简xml版本

- 1、配置
- 2、添加User的映射文件
- 3、编写开发MyMapper 直接写在接口里面? 还需不需要写实现类?
- 4、使用

如何选择




missfmaster (<https://github.com/missfmaster>) 发表于 大约 2 个月前

惭愧,16年的文章现在才看到。

平时第二种方式更方便一些,第一种简单的同时,功能上也制约了。

打印SQL的那个,mybatis-config中配置 `<setting name="logImpl" value="STDOUT_LOGGING" />`



mfzhu (<https://github.com/mfzhu>) 发表于 大约 2 个月前

mybatis.config-locations=classpath:mybatis/mybatis-config.xml

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

这个报错不能同时使用



joeaniu (<https://github.com/joeaniu>) 发表于 大约 2 个月前



作者更正一下吧: 官方文档写的是“config-location”, 而不是“config-locations”, 见<http://www.mybatis.org/spring-boot-starter/mybatis-spring-boot-autoconfigure/> (<http://www.mybatis.org/spring-boot-starter/mybatis-spring-boot-autoconfigure/>)

mybatis-springboot-starter

无配置文件注解版



1. 添加相关maven文件

liuminda (<https://github.com/liuminda>) 发表于 大约 1 个月前



2. application.properties添加相关配置 谢谢大神的分享。

- 3、开发Mapper

- 4、使用

Post Directory

- 1、配置

- 2、添加User的映射文件

- 3、编写Dao层的代码

- 4、使用

如何选择

知乎 (<https://www.zhihu.com/people/ityouknow>) 微博 (<http://weibo.com/ityouknow>)

Github (<https://github.com/ityouknow>)

(/feed.xml) Power by Yummy Jekyll (<https://github.com/DONGChuan/Yummy-Jekyll>)

京ICP备15067287号-3 TOP

资源 (/share/2017/10/01/resource-sharing.html) 故事 (/life.html) 架构 (/arch.html) Jvm (/jvm.html) FastDFS (/fastdfs.html)

MongoDB (/mongodb.html) Docker (/docker.html) Code (/open-source.html)