链滴

Spring Boot 整合 MyBatis

作者：mtkx

原文链接：https://ld246.com/article/1556436468836

来源网站：链滴

MyBatis 是一款优秀的持久层框架，原名叫作iBaits，2010年由 ApacheSoftwareFoundation

迁移到 Google Code 并改名为 MyBatis, 2013年又迁移到 GitHub 上。MyBatis 支持定制化 SQL、储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。在传的 SSM 框架整合中，使用 MyBatis 需要大量 XML 配置，而 Spring Boot 中， MyBatis 官方提供一套自动化配置方案，可以做到 MyBatis 开箱即用。具体使用步骤如下：

1.新建Spring Boot工程，然后添加 spring-boot-starter-web ， mysql-connector-java ， mybatis-spring-boot-starter ， druid 依赖，代码如下：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- 本实例使用mysql 8.x 版本-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
    <version>8.0.15</version>
</dependency>
<!-- mybatis依赖 -->
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>2.0.0</version>
</dependency>
<!-- 引入第三方数据源 地址:https://github.com/alibaba/druid-->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>druid</artifactId>
    <version>1.1.9</version>
</dependency>
```

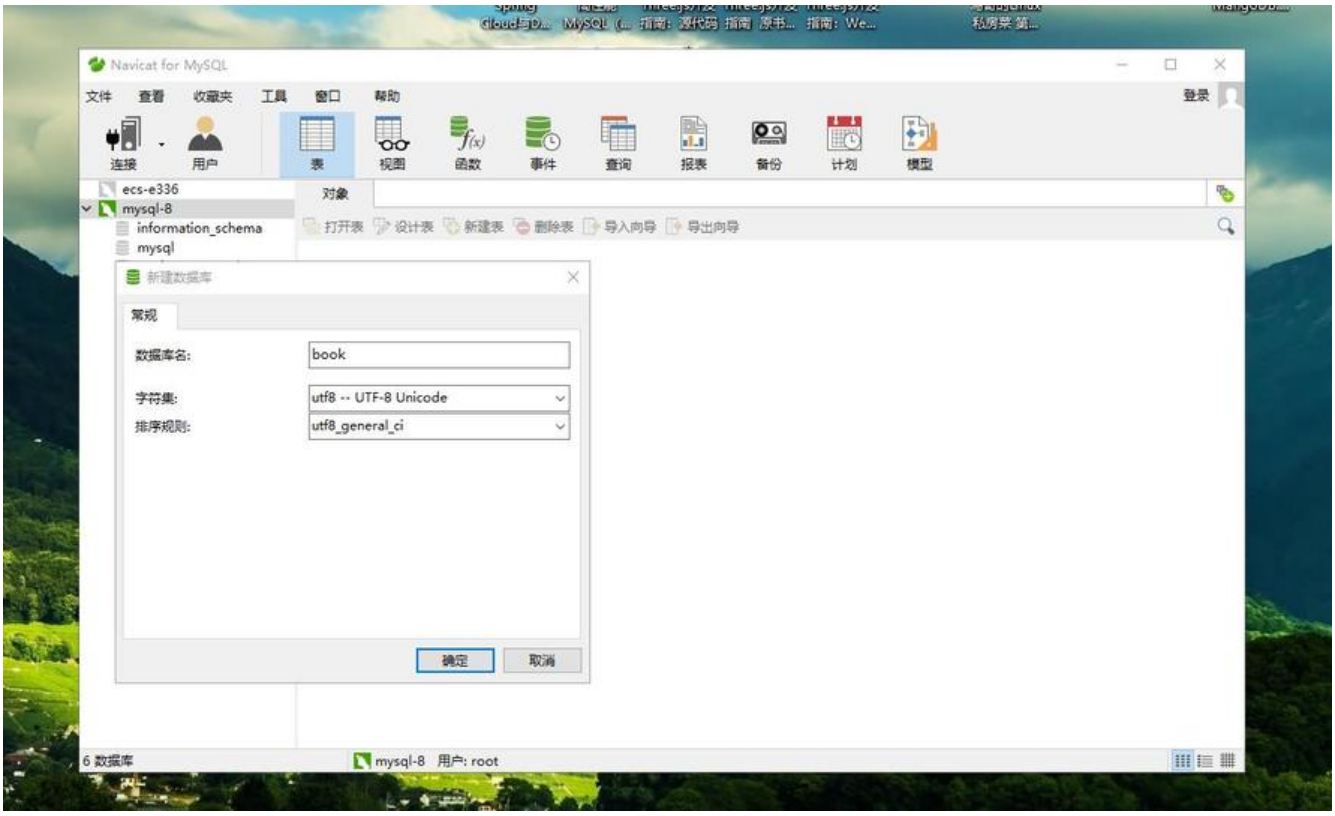
2.在 application.properties 中配置 Mysql 基本连接信息，并启用 druid 据源，代码如下：

```
#==========================================数据库相关=========
=====================================

# 请自行安装 8.x mysql
spring.datasource.url=jdbc:mysql://localhost:3306/book?useUnicode=true&characterEncodin
=utf-8&serverTimezone=GMT%2B8
spring.datasource.username =root
spring.datasource.password =xinxin
#如果不使用默认的数据源 （com.zaxxer.hikari.HikariDataSource）
spring.datasource.type =com.alibaba.druid.pool.DruidDataSource

#增加打印sql语句，一般用于本地开发测试
mybatis.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

3.创建数据库：



4.创建 Book 实体类，代码如下：

```
public class Book {

    private Integer id;
    private String name;
    private String author;
```

```
    public Book() {
    }

    public Book(String name, String author) {
        this.name = name;
        this.author = author;
    }

    @Override
    public String toString() {
        return "Book{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", author='" + author + '\'' +
                '}';
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
}
```

4.创建数据访问层 BookMapper 与 BookProvider 并在启动类中加入 @MaperScan 注解具体代码如下:

```
public interface BookMapper {

    @Select("select * from book")

    List<Book> findAll();

    @Select("SELECT * FROM book WHERE id = #{id}")
    List<Book> findById(int id);
```

```
    //@Update("UPDATE book set name=#{name} WHERE id = #{id}")
    @UpdateProvider(type = BookProvider.class,method = "updateBook")
    int update(Book book);

    @Delete("DELETE FROM book WHERE id =#{id}")
    int delete(int id);

    @Insert("INSERT INTO `book`.`book` (`name`, `author`)" +
            " VALUES " +
            "(#{name},#{author});")
    @Options(useGeneratedKeys = true,keyProperty = "id",keyColumn = "id")
    int save(Book book);
}

/**

 * book构建动态sql语句
 * Created by Mtkx on 2019/4/28.
 */
public class BookProvider {
    /**
     * 更新Book动态sql语句
     * @param book
     * @return
     */
    public String updateBook(final Book book){
        return new SQL(){{
            UPDATE("book");

            //条件写法.
            if(book.getName()!=null){
                SET("name=#{name}");
            }
            if(book.getAuthor()!=null){
                SET("author=#{author}");
            }

            WHERE("id=#{id}");
        }}.toString();
    }
}

@SpringBootApplication

@MapperScan("xyz.mtkx.mybatis.mapper")`//BookMapper所在包名`

public class MybatisApplication {

    public static void main(String[] args) {
        SpringApplication.run(MybatisApplication.class, args);
    }

}
```

5.创建 BookService 与 BookServiceImpl 具体代码如下:

```
public interface BookService {
    List<Book> findAll();

    List<Book> findById(int id);

    int update(Book book);

    int delete(int id);

    int save(Book book);
}

@Service
public class BookServiceImpl implements BookService {
    @Autowired
    private BookMapper bookMapper;
    @Override
    public List<Book> findAll() {
        return bookMapper.findAll();
    }

    @Override
    public List<Book> findById(int id) {
        return bookMapper.findById(id);
    }

    @Override
    public int update(Book book) {
        return bookMapper.update(book);
    }

    @Override
    public int delete(int id) {
        return bookMapper.delete(id);
    }

    @Override
    public int save(Book book) {
        int rows=bookMapper.save(book);
        System.out.println("新插入id="+book.getId());
        return rows;
    }
}
```

5.创建 BookController 具体代码如下:

```
@RestController
@RequestMapping("/api/v1/book")
public class BookController {
    @Autowired
    private BookService bookService;
```

```
/**
 * 根据id 查询图书列表
 * @return
 */
@GetMapping("findAll")
public Object findAll(){
    return bookService.findAll();
}

/**
 * 根据id 查询图书
 * @param book_id
 * @return
 */
@GetMapping("findById")
public Object findById(@RequestParam(value = "book_id",required = true) int book_id){
    return bookService.findById(book_id);
}

/**
 * 根据id 删除图书
 * @param book_id
 * @return
 */
@DeleteMapping("del_by_id")
public Object delById(@RequestParam(value = "book_id",required = true) int book_id){
    return bookService.delete(book_id);
}

/**
 * 更新
 * @param book
 * @return
 */
@PutMapping("update_by_id")
public Object update(@RequestBody Book book){
    return bookService.update(book);
}

/**
 * 保存book对象
 * @param book
 * @return
 */
@PostMapping("save")
public Object save(@RequestBody Book book){
    return bookService.save(book);
}

/**
 * 测试
 * @return
 */
```

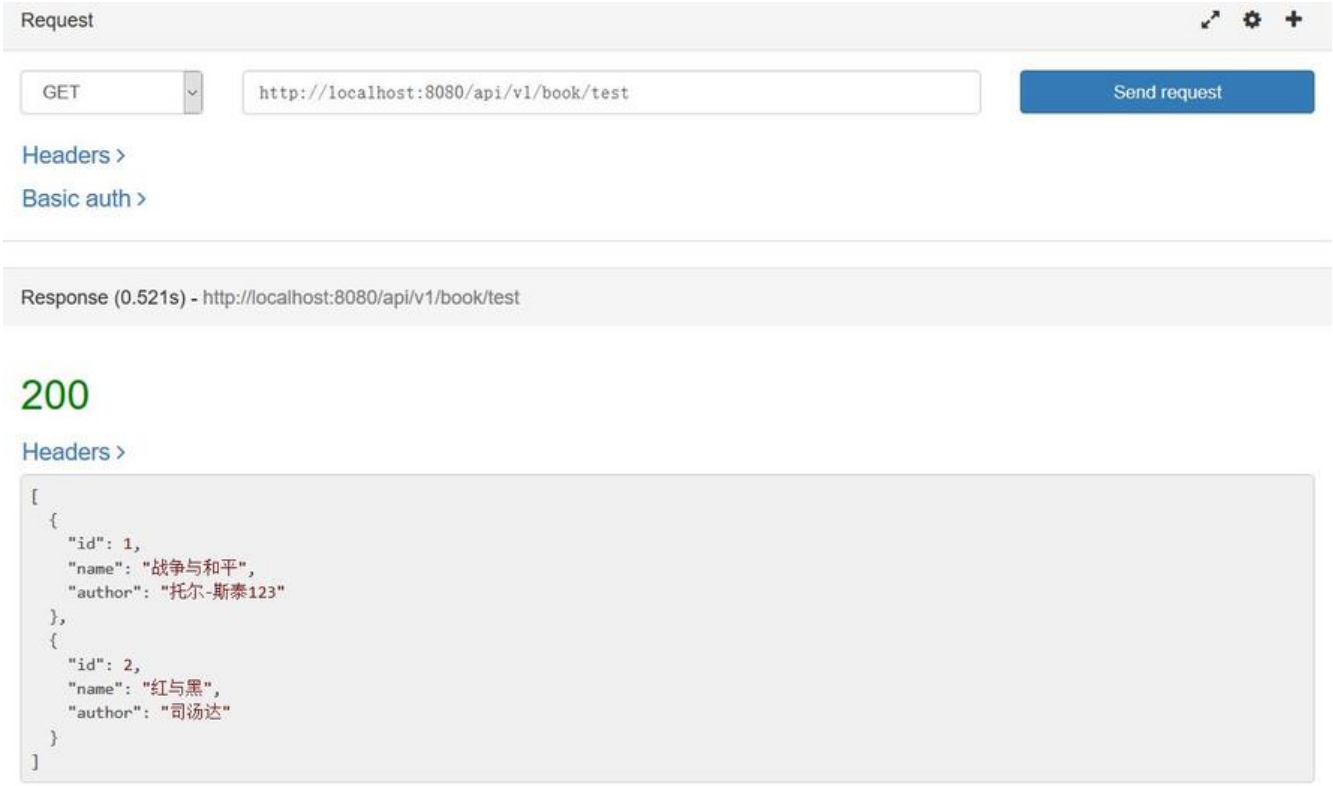
```
    @GetMapping("test")
    public Object test(){
        //添加数据（单条）
        Book book=new Book("战争与和平","托尔斯泰");
        bookService.save(book);
        Book book1=new Book("红与黑","司汤达");
        bookService.save(book1);
        Book book2=new Book("悲惨世界","雨果");
        bookService.save(book2);
        System.out.println("1-添加后的数据-->"+bookService.findAll());

        //更新
        Book uBook=new Book();
        uBook.setId(1);
        uBook.setAuthor("托尔-斯泰123");
        bookService.update(uBook);
        System.out.println("2-更新后的数据-->"+bookService.findById(1));

        //删除
        bookService.delete(3);
        System.out.println("3-删除后的数据-->"+bookService.findAll());

        //查询数据
        return bookService.findAll();
    }
}
```

6.运行项目，在浏览器中输入 http://localhost:8080/api/v1/book/test 即看到运行结果，如图所示。



原文链接：Spring Boot 整合 MyBatis

通过上面的例子可以看到 MyBatis 基本上实现了开箱即用的特性。自动化配置将开发者从繁杂的配置文件中解脱出来，以专注于业务逻辑的开发。

原文链接：Spring Boot 整合 MyBatis