kybetter RP 360
2018-03-29 发布

spring-boot 框架整合 MyBatis

> 本文讨论使用 mybatis-spring-boot-starter 的方式整合进 spring-boot 框架中
> 本文也只详细讨论基于 xml 的配置，基于注解的方式比 xml 要简单，不再做详细讲解。

1、将 MyBatis 加入 pom.xml 配置

```
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>1.3.1</version>
</dependency>
```

2、application.yml 配置

```
spring:
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/用哪个数据库?useUnicode=true&characterEncoding=utf-8
    username: 用户名
    password: 密码

server:
  port: 8080

mybatis:
  config-location: classpath:config/mybatis-config.xml
  mapper-locations: classpath:mapper/*.xml
```

MyBatis 配置项解读：

- config-location：指定 MyBatis 主配置文件的位置
- mapper-locations：指定 mapper 文件的位置。如果在项目中你的 mapper 文件是按目录来放置，那么对应的配置就变成：`mapper-locations: classpath:mapper/*/*.xml`

这时候假设我们的 resources 结构是这样的：

```
|-resources
|--config
|---application.yml
|---mybatis-config.xml
|--mapper
|---CityMapper.xml
```

3、mybatis-config.xml 配置

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <typeAliases>
        <package name="com.mybatis.domain"/>
    </typeAliases>
    <!--<mappers>-->
        <!--<mapper resource="sample/mybatis/mapper/CityMapper.xml"/>-->
        <!--<mapper resource="sample/mybatis/mapper/HotelMapper.xml"/>-->
```

```
    <!--</mappers>-->
</configuration>
```

这个配置见仁见智，在它里面我就配置了一个 `typeAliases`。不了解的同学可以移步文档查看相关解释。

你也可以把 mapper 配置在此处，有多少个 mapper 就配置多少次，当然，我们已经在 `application.yml` 中批量指定了，很方便，就不用在此处一个个写。

4、接下来就是业务代码部分了

假设我们的目录结构是这样的：

```
|-com.mybatis
|--controller
|---CityRestController.java (控制器)
|--domain
|---City.java (实体类)
|--mapper
|---CityMapper.java (mybatis的mapper)
|--service
|---CityService.java (service 接口)
|---CityServiceImpl.java (service 实现)
|--MyApplication.java (入口)
```

首先我们定义城市这个实体类：

```
        return provinceId;
    }

    public void setProvinceId(Long provinceId) {
        this.provinceId = provinceId;
    }

    public String getCityName() {
        return cityName;
    }

    public void setCityName(String cityName) {
        this.cityName = cityName;
    }

    @Override
    public String toString() {
        return "City{" +
                "id=" + id +
                ", provinceId=" + provinceId +
                ", cityName='" + cityName + '\'' +
               '}';
    }
}
```

接着我们来定义mapper：

```
//@Mapper
public interface CityMapper {

    City findByProvinceId(Long provinceId);

    List<City> findAll();
}
```

定义这个 mapper 的作用是用来跟数据库进行交互的。

请注意，这里我把 `@Mapper` 注解给注释掉了，大家先把这个注释打开，待会再来解释。

另外，如果你想使用**注解**的方式来操作数据库，那么可以这样来定义 mapper：

```
//@Mapper
public interface CityMapper {
    @Select("select * from city where province_id = #{provinceId}")
```

```
    // 返回结果实体属性与数据库字段转换
    @Results({
            @Result(property = "provinceId", column = "province_id"),
            @Result(property = "cityName", column = "city_name")
    })
    City findByProvinceId(@Param("provinceId") Long provinceId);
}
```

对于选择使用**注解**还是使用 **xml** 的方式，大家可以灵活选择，比如简单的语句可以使用注解，复杂的语句使用 xml，当然，还是需要跟团队保持一致。

还记得我们 resources 的结构吗？

里面有一个mapper目录，我们配置了在这个目录里面寻找 mapper 文件。

这里我定义了一个 `CityMapper.xml` 文件，用来跟数据库进行交互：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.mybatis.mapper.CityMapper">
    <resultMap id="BaseResultMap" type="com.mybatis.domain.City" >
        <result column="province_id" property="provinceId" />
        <result column="city_name" property="cityName" />
    </resultMap>
    <select id="findByProvinceId" resultMap="BaseResultMap">
        select * from city where province_id = #{provinceId}
    </select>
    <select id="findAll" resultMap="BaseResultMap">
        select * from city
    </select>
</mapper>
```

文件解读：

- namespace 属性用于跟咱们业务中的那个 mapper 进行关联
- resultMap 标签用来定义字段映射和结果字段返回类型
- select 中的 id 属性用来跟咱们业中 mapper 的方法进行关联，查询到的结果就会返回给该方法
- `#{}` 是占位符，表示需要动态获取的数据

最后定义 service 来处理咱们的业务：

```
public interface CityService {

    City getByProvinceId(Long provinceId);

    List<City> getAll();

}
```

```
@Service
public class CityServiceImpl implements CityService {

    @Resource
    private CityMapper cityMapper;

    @Override
    public City getByProvinceId(Long provinceId) {
        return cityMapper.findByProvinceId(provinceId);
    }

    @Override
    public List<City> getAll() {
        return cityMapper.findAll();
    }
}
```

在controller里面进行验证：

```
@RestController
@RequestMapping("/api")
public class CityRestController {

    @Resource
    private CityService cityService;

    @GetMapping("/cities")
    public List<City> cities () {
        return cityService.getAll();
    }

    @GetMapping("/city/{provinceId}")
    public City city(@PathVariable long provinceId) {
        return cityService.getByProvinceId(provinceId);
    }

}
```

最后

还记得我定义 `CityMapper.java` 的时候，为什么把 `@Mapper` 给注释掉了吗？是因为我在入口文件这里定义了到哪里去找 mapper 文件，所以就不用再在每个 mapper 文件中再写个注解了：

```
@SpringBootApplication
@MapperScan("com.mybatis.mapper") // 定义了在哪里扫描mapper文件
public class MyApplication {

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

至此，spring-boot 与 mybatis 整合完毕。

5、附上三个 MyBatis 链接，分别是：

1、mybatis-spring-boot-starter 配置项手册

2、GitHub仓库地址，包含代码示例

3、中文官方文档

你可能感兴趣的

- **mybatis 拓展 -- 通用mapper 和 动态 resultMap** liulu spring springboot mybatis
- **Mybatis：一种 Redis 缓存实现** xingpingz java mybatis redis 缓存设计
- **MyBatis入门介绍** scu酱油仔 mybatis java
- **7.平凡之路-动态SQL语句** pangsir8983 orm java-ee java mybatis
- **Spring Boot（三）Spring Boot 和 MyBatis 整合** liaosilzu2007 springboot mybatis
- **mybatis的statement的解析与加载** codecraft mybatis
- **Spring Boot学习笔记（六）结合MyBatis实现较为复杂的RESTful API** Bug生活2048 后端 程序员 mybatis spring-boot java
- **SpringBoot集成mybatis** codecraft mybatis springboot