狂神说SpringBoot09：整合MyBatis

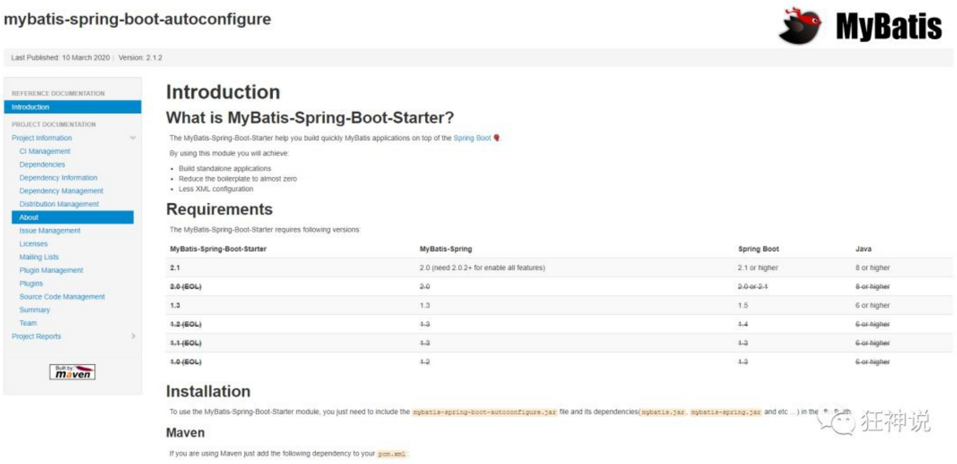
秦疆 狂神说 2020-03-16

狂神说SpringBoot系列连载课程，通俗易懂，基于SpringBoot2.2.5版本，欢迎各位狂粉转发关注学习。

整合MyBatis

官方文档：http://mybatis.org/spring-boot-starter/mybatis-spring-boot-autoconfigure/

Maven仓库地址：https://mvnrepository.com/artifact/org.mybatis.spring.boot/mybatis-spring-boot-starter/2.1.1



mybatis-spring-boot-autoconfigure

Last Published: 10 March 2020 | Version: 2.1.2

REFERENCE DOCUMENTATION
- Introduction

PROJECT DOCUMENTATION
- Project Information
  - CI Management
  - Dependencies
  - Dependency Information
  - Dependency Management
  - Distribution Management
  - About
  - Issue Management
  - Licenses
  - Mailing Lists
  - Plugin Management
  - Plugins
  - Source Code Management
  - Summary
  - Team
- Project Reports

Introduction

What is MyBatis-Spring-Boot-Starter?

The MyBatis-Spring-Boot-Starter help you build quickly MyBatis applications on top of the Spring Boot.

By using this module you will achieve:

- Build standalone applications
- Reduce the boilerplate to almost zero
- Less XML configuration

Requirements

The MyBatis-Spring-Boot-Starter requires following versions:

MyBatis-Spring-Boot-Starter	MyBatis-Spring	Spring Boot	Java
2.1	2.0 (need 2.0.2+ for enable all features)	2.1 or higher	8 or higher
~~2.0 (EOL)~~	~~2.0~~	~~2.0 or 2.1~~	~~8 or higher~~
1.3	1.3	1.5	6 or higher
~~1.2 (EOL)~~	~~1.3~~	~~1.4~~	~~6 or higher~~
~~1.1 (EOL)~~	~~1.3~~	~~1.3~~	~~6 or higher~~
~~1.0 (EOL)~~	~~1.2~~	~~1.3~~	~~6 or higher~~

Installation

To use the MyBatis-Spring-Boot-Starter module, you just need to include the `mybatis-spring-boot-autoconfigure.jar` file and its dependencies(`mybatis.jar`, `mybatis-spring.jar` and etc ...) in the

Maven

If you are using Maven just add the following dependency to your `pom.xml`:

整合测试

1、导入 MyBatis 所需要的依赖

```
<dependency>    <groupId>org.mybatis.spring.boot</groupId>    <artifactId>mybatis-spring-boot-starter</artifactId>    <version>2.1.1</version></dependency>
```

2、配置数据库连接信息（不变）

```
spring:  datasource:    username: root    password: 123456    #?serverTimez
one=UTC解决时区的报错    url: jdbc:mysql://localhost:3306/springboot?serverTim
ezone=UTC&useUnicode=true&characterEncoding=utf-8    driver-class-name: com
.mysql.cj.jdbc.Driver    type: com.alibaba.druid.pool.DruidDataSource
    #Spring Boot 默认是不注入这些属性值的，需要自己绑定    #druid 数据源专有配置    i
nitialSize: 5    minIdle: 5    maxActive: 20    maxWait: 60000    timeBetwe
enEvictionRunsMillis: 60000    minEvictableIdleTimeMillis: 300000    valida
tionQuery: SELECT 1 FROM DUAL    testWhileIdle: true    testOnBorrow: false
    testOnReturn: false    poolPreparedStatements: true
    #配置监控统计拦截的filters，stat:监控统计、log4j：日志记录、wall：防御sql注入
#如果允许时报错  java.lang.ClassNotFoundException: org.apache.log4j.Priority
  #则导入 log4j 依赖即可，Maven 地址：https://mvnrepository.com/artifact/log4j/
log4j    filters: stat,wall,log4j    maxPoolPreparedStatementPerConnectionS
ize: 20    useGlobalDataSourceStat: true    connectionProperties: druid.sta
t.mergeSql=true;druid.stat.slowSqlMillis=500
```

3、测试数据库是否连接成功！

4、创建实体类，导入 Lombok！

Department.java

```
package com.kuang.pojo;
import lombok.AllArgsConstructor;import lombok.Data;import lombok.NoArgsCon
structor;
@Data@NoArgsConstructor@AllArgsConstructorpublic class Department {
    private Integer id;    private String departmentName;
}
```

5、创建mapper目录以及对应的 Mapper 接口

DepartmentMapper.java

```
//@Mapper : 表示本类是一个 MyBatis 的 Mapper@Mapper@Repositorypublic interface
DepartmentMapper {
    // 获取所有部门信息    List<Department> getDepartments();
    // 通过id获得部门    Department getDepartment(Integer id);
}
```

6、对应的Mapper映射文件

DepartmentMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE mapper        PUBLIC "-//m
ybatis.org//DTD Mapper 3.0//EN"        "http://mybatis.org/dtd/mybatis-3-ma
pper.dtd">
<mapper namespace="com.kuang.mapper.DepartmentMapper">
    <select id="getDepartments" resultType="Department">        select * fro
m department;    </select>
    <select id="getDepartment" resultType="Department" parameterType="int">
       select * from department where id = #{id};    </select>
</mapper>
```

7、maven配置资源过滤问题

```
<resources>    <resource>        <directory>src/main/java</directory>
  <includes>            <include>**/*.xml</include>        </includes>
   <filtering>true</filtering>    </resource></resources>
```

8、编写部门的 DepartmentController 进行测试！

```
@RestControllerpublic class DepartmentController {         @Autowired    DepartmentMapper departmentMapper;          // 查询全部部门    @GetMapping("/getDepartments")    public List<Department> getDepartments(){         return departmentMapper.getDepartments();     }
    // 查询全部部门    @GetMapping("/getDepartment/{id}")    public Department getDepartment(@PathVariable("id") Integer id){         return departmentMapper.getDepartment(id);     }     }
```

启动项目访问进行测试！

我们增加一个员工类再测试下，为之后做准备

1、新建一个pojo类 Employee；

```
@Data@AllArgsConstructor@NoArgsConstructorpublic class Employee {
    private Integer id;    private String lastName;    private String email;
    //1 male, 0 female    private Integer gender;    private Integer depart
ment;    private Date birth;
    private Department eDepartment; // 冗余设计
}
```

2、新建一个 EmployeeMapper 接口

```
//@Mapper : 表示本类是一个 MyBatis 的 Mapper@Mapper@Repositorypublic interface
EmployeeMapper {
    // 获取所有员工信息    List<Employee> getEmployees();
    // 新增一个员工    int save(Employee employee);
    // 通过id获得员工信息    Employee get(Integer id);
    // 通过id删除员工    int delete(Integer id);
}
```

3、编写 EmployeeMapper.xml 配置文件

```
<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE mapper        PUBLIC "-//m
ybatis.org//DTD Mapper 3.0//EN"        "http://mybatis.org/dtd/mybatis-3-ma
pper.dtd">
<mapper namespace="com.kuang.mapper.EmployeeMapper">
    <resultMap id="EmployeeMap" type="Employee">        <id property="id" c
olumn="eid"/>        <result property="lastName" column="last_name"/>
  <result property="email" column="email"/>        <result property="gender"
 column="gender"/>        <result property="birth" column="birth"/>        <
association property="eDepartment"  javaType="Department">            <id p
roperty="id" column="did"/>            <result property="departmentName" co
lumn="dname"/>        </association>    </resultMap>
    <select id="getEmployees" resultMap="EmployeeMap">        select e.id a
s eid,last_name,email,gender,birth,d.id as did,d.department_name as dname
```

```
      from department d,employee e        where d.id = e.department    </se
lect>
    <insert id="save" parameterType="Employee">        insert into employee
 (last_name,email,gender,department,birth)        values (#{lastName},#{ema
il},#{gender},#{department},#{birth});    </insert>
    <select id="get" resultType="Employee">        select * from employee w
here id = #{id}    </select>
    <delete id="delete" parameterType="int">        delete from employee wh
ere id = #{id}    </delete>
</mapper>
```

4、编写EmployeeController类进行测试

```
@RestControllerpublic class EmployeeController {
    @Autowired    EmployeeMapper employeeMapper;
    // 获取所有员工信息    @GetMapping("/getEmployees")    public List<Employe
e> getEmployees(){        return employeeMapper.getEmployees();    }
    @GetMapping("/save")    public int save(){        Employee employee = n
ew Employee();        employee.setLastName("kuangshen");        employee.se
tEmail("qinjiang@qq.com");        employee.setGender(1);        employee.se
tDepartment(101);        employee.setBirth(new Date());        return emplo
yeeMapper.save(employee);    }
    // 通过id获得员工信息    @GetMapping("/get/{id}")    public Employee get(@
PathVariable("id") Integer id){        return employeeMapper.get(id);    }
    // 通过id删除员工    @GetMapping("/delete/{id}")    public int delete(@Pa
thVariable("id") Integer id){        return employeeMapper.delete(id);    }
}
```

测试结果完成，搞定收工！

喜欢就在看！
狂神说