**第一个springboot项目**

1. 打开https://start.spring.io/
2. 填写相关的项目信息，下载生成的压缩包
3. 使用idea打开，会根据pom.xml自动下载依赖包
4. 引入web项目相关依赖 ，复制以下内容到 pom.xml中

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId></dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope></dependency>

引入后，idea会自动下载对应依赖包（如果没有自动下载，可以尝试重新启动idea）

1. 新建package ,名称为controller, 在包下新建类 HelloWorldController.java，复制如下代码

@RestController

@RequestMapping("/hello")

public class HelloWorldController {

@RequestMapping("/say")

public String say(){

return "Hello World";

}

}

1. 修改 web服务对应的端口

在 resources\application.properties 添加 server.port=8999

1. 以FirstSampleApplication 下的main函数作为项目的入口，启动项目
2. 在浏览器中输入 <http://localhost:8999/hello/say> 来验证项目是否成功

参考：https://www.cnblogs.com/fishpro/p/spring-boot-study-helloworld.html

**集成mysql**

1. pom中添加mysql 相关依赖

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<!--添加JDBC依赖-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

1. Properties 文件中添加 数据库连接配置

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://192.168.125.200:3306/testdb?serverTimezone=UTC&useUnicode=true&characterEncoding=utf8

spring.datasource.username=root

spring.datasource.password=1010

并在对应的数据库服务器中新建 testdb 数据库，并新建表

CREATE TABLE `student` (

`id` int NOT NULL,

`name` varchar(45) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`age` int DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

1. 新建访问数据库的 service接口和实现

新建 包service，在其下新建imp 子包

在 service 下新建 IUserService 接口，

public interface IUserService {

public List<Map<String,Object>> getUserName();

}

在 imp 子包下新建 UserServiceImp 实现上面的接口

@Service

public class UserServiceImp implements IUserService {

@Autowired

private JdbcTemplate jdbcTemplate;

@Override

public List<Map<String,Object>> getUserName(){

List<Map<String,Object>> result =jdbcTemplate.queryForList("select \* from student");

return result;

}

}

1. 在controller中添加 IUserService的引用

@Autowired

IUserService service;

5.新建方法

@RequestMapping("/queryUserName")

public List<Map<String,Object>> queryUserName(){

return service.getUserName();

}

6.通过 <http://localhost:8999/hello/queryUserName> 访问

参考 https://blog.csdn.net/ymeddmn/article/details/95992463

**集成mongodb**

1.添加依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

2.properties文件中添加连接的配置 (前提是对应的用户在对应的数据库里有权限)

spring.data.mongodb.uri=mongodb://xl1:1010@192.168.125.200:27017/testdb

并通过客户端工具在mongodb服务器上建立表和库

db.testtable.insert({"id":1,"name":"aaa","age":10}) 标准

1. 在上一节中的service的实现中

通过

@Autowired

private MongoTemplate mongoTemplate;

注入 MongoTemplate

1. 在上一节中的service的接口和实现中添加

@Override

public String queryMongodb()

{

Query query = new Query(Criteria.where("id").is(1));

// 查询一条满足条件的数据

Map result = mongoTemplate.findOne(query, Map.class, "testtable");

return result.toString();

}

1. 在上一节的controller中添加 方法

@RequestMapping("/queryMongodb")

public String queryMongodb() {

return service.queryMongodb();

}

通过 <http://localhost:8999/hello/queryMongodb> 访问数据，返回结果

**集成redis**

1.添加依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-redis</artifactId>

</dependency>

2.properties文件中添加连接的配置 (需要预先设置好密码)

spring.redis.host=192.168.125.200

spring.redis.password=1010

spring.redis.port=6379

3.在上一节中的service的实现中

通过

@Autowired

private RedisTemplate<String, String> redisTemplate;

注入 redisTemplate

1. 在上一节中的service的接口和实现中添加

@Override

public String redisTest()

{

redisTemplate.opsForValue().set("aa","bb");

String getValue = redisTemplate.opsForValue().get("aa");

return getValue;

}

1. 在上一节的controller中添加 方法

@RequestMapping("/redisTest")

public String redisTest() {

return service.redisTest();

}

通过 http://localhost:8999/hello/redisTest 访问数据，返回结果

**集成mybatis**

1. 接第一个个springboot项目

1.添加依赖

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.3.0</version>

</dependency>

2.properties文件中添配置 (对应的package 和连接字符串要和代码数据库匹配)

mybatis.type-aliases-package=com.example.mybatisdemo

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://192.168.125.200:3306/testdb?serverTimezone=UTC&useUnicode=true&characterEncoding=utf8

spring.datasource.username=xl

spring.datasource.password=1010

3.在启动类中加注解

@MapperScan("com.example.mybatisdemo.mapper")

新建package mapper (此package需和上一行配置的完全对应，层级不能多也不能少) 并新建StudentMapper.java 的接口，并添加方法

public interface StudentMapper {

int getCount();

}

4.在properties文件中添加 配置项

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

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

5.在resources目录下新建 mybatis目录并新建mybatis-config.xml 和mapper子目录，并新建student.xml文件，和上一步的配置对应

mybatis-config.xml的内容如下

<?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>

<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>

student.xml 的内容如下（需要和上面的StudentMapper 接口对应）

<?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.example.mybatisdemo.mapper.StudentMapper">

<select id="getCount" resultType="java.lang.Integer">

SELECT count(\*)

FROM student

</select>

</mapper>

1. 新建 package service 以及子package imp

新建TestService 接口机器 实现类 TestServiceImp

public interface TestService {

int getTestCount();

}

@Service

public class TestServiceImp implements TestService {

@Autowired

private StudentMapper studentMapper;

@Override

public int getTestCount( ) {

return studentMapper.getCount();

}

}

6.新建package controller,并在其中新建TestController 类

@RestController

@RequestMapping("/test")

public class TestController {

@Autowired

private TestService testService;

@RequestMapping("/getCount")

public int getCount() {

return testService.getTestCount();

}

}

通过 localhost:port//test/getCount 来访问

7.返回实体列表

新建package bean 以及与表对应的实体类Student.java

public class Student {

public int id;

public String name;

public int age;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

在StudentMapper.java中添加方法 List<Student> getAll();

在student.xml文件中添加 对应的实体映射和方法映射

<resultMap id="studentMap" type="com.example.mybatisdemo.bean.Student">

<id property="id" column="id"/>

<result property="name" column="name "/>

<result property="age" column="age"/>

</resultMap>

<select id="getAll" resultMap="studentMap">

SELECT \*

FROM student

</select>

在 TestService 和 TestServiceImp 中添加对应的方法和方法实现

List<Student> getAll();

@Override

public List<Student> getAll()

{

return studentMapper.getAll();

}

在Contoller中添加对应的方法

@RequestMapping("/getAll")

public List<Student> getAll() {

return testService.getAll();

}

通过 <http://localhost:8929/test/getAll> 访问