Springboot项目可以直接在官网生成，也可以用开发工具idea来生成。

Springboot启动通过SpringApplication.run方法来启动

参数都配置在application.properties中，程序中需要使用通过注解@Value来引用

Bean的生成通配置@Configuration，和@Bean注解来实现，在引用的地方通过@Autowire

数据源 事务管理 sqlsessionfactory等bean都可以通过这样来处理。

Springboot程序的代码一定都要放在和启动类同一个包路径下面，不然会有问题。

我在springboot官网建的springboot程序，没有webapp文件夹，咋回事？

application.properties 内置属性，自定义属性

server.port=8888

server.context-path=/helloword

mysql.jdbcName=com.mysql.jdbc.Driver

mysql.dbUrl=jdbc:mysql://localhost:3306/dbName

mysql.userName=root

mysql.password=123456

@Componet

@ConfigurationProperties(prefix=”mysql”) 可以将属性封装在对象中

@RestContoller = @Controller + @ResponseBody 响应ajax请求 返回的只有数据

@Controller 返回的 视图+数据

@PathVariable 实现rest风格的请求

@RequestParam

@RequestBody

Spring JPA 使用的是hibernate，但是一般不用ddl，mybatis更灵活

在pojo的do上，使用@Entity @Table(name=”t\_book”)

在属性上面加注解：@Id @GeneratedValue @Column(length=50)

JPA封装了很多基本数据库操作的方法。只需要继承 JPARepository接口

如果需要特殊的数据库dml操作，怎么操作？

自定义查询 @Query 可以使用hql语言（默认使用），和sql语言 nativequery=true

要实现mybatis里面的动态sql，需要使用动态查询Specification

需要继承JPARepositorySpecification

Springboot的事物 @Transactional

Springboot表单验证

我以前用过hibernatevalid

BindingResult

import org.springframework.validation.BindingResult;

@RequestMapping("/add")

public String add(@Valid Student student,BindingResult bindingResult){

if(bindingResult.hasErrors()){

return bindingResult.getFieldError().getDefaultMessage();

}else{

studentService.add(student);

return "添加成功";

}

}

Springboot AOP 统一处理（日志，异常等等）

|  |
| --- |
| package com.java1234.aspect;  import javax.servlet.http.HttpServletRequest;  import org.apache.log4j.Logger;  import org.aspectj.lang.JoinPoint;  import org.aspectj.lang.annotation.After;  import org.aspectj.lang.annotation.AfterReturning;  import org.aspectj.lang.annotation.Aspect;  import org.aspectj.lang.annotation.Before;  import org.aspectj.lang.annotation.Pointcut;  import org.springframework.stereotype.Component;  import org.springframework.web.context.request.RequestContextHolder;  import org.springframework.web.context.request.ServletRequestAttributes;  import com.java1234.entity.Student;  @Aspect  @Component  public class RequestAspect {  private Logger logger=Logger.getLogger(RequestAspect.class);    @Pointcut("execution(public \* com.java1234.controller.\*.\*(..))")  public void log(){  }    @Before("log()")  public void doBefore(JoinPoint joinPoint){  logger.info("方法执行前...");  ServletRequestAttributes sra=(ServletRequestAttributes) RequestContextHolder.getRequestAttributes();  HttpServletRequest request=sra.getRequest();  logger.info("url:"+request.getRequestURI());  logger.info("ip:"+request.getRemoteHost());  logger.info("method:"+request.getMethod());  logger.info("class\_method:"+joinPoint.getSignature().getDeclaringTypeName()+"."+joinPoint.getSignature().getName());  logger.info("args:"+joinPoint.getArgs());  Student student=(Student) joinPoint.getArgs()[0];  System.out.println(student);  }    @After("log()")  public void doAfter(JoinPoint joinPoint){  logger.info("方法执行后...");  }    @AfterReturning(returning="result",pointcut="log()")  public void doAfterReturning(Object result){  logger.info("方法返回值："+result);  }  } |