# 注解集合

* @pathvariable 获取url参数
* @requestparam 获取request参数,带问好的也获得,defaultvalue可以设置默认值
* @transactional 事务管理
* @value表单验证

# jpa

## maven

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
</dependency>

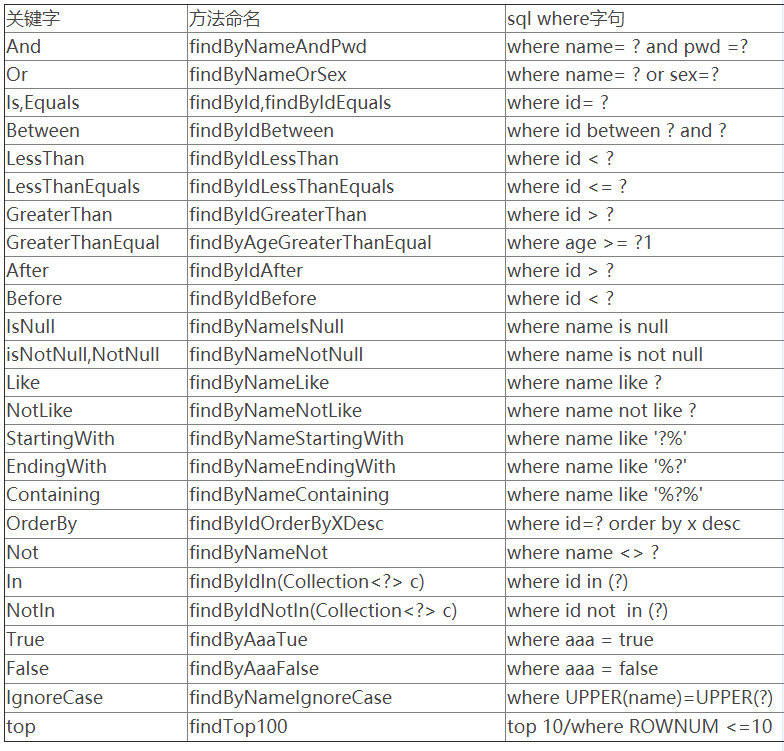
## 配置

spring:  
 profiles:  
 active: dev  
 datasource:  
 driver-class-name: com.mysql.jdbc.Driver  
 url: jdbc:mysql://localhost:3306/girl  
 username: root  
 password: root  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true

# 数据库命名

不要骆驼峰,用下划线

## mapper命名



# 事务管理

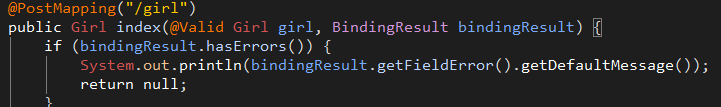
@transactional 事务管理,主要用于修改数据库数据的,监视如果其中有一条失败其他的都会失败,比如同时插入两条数据,一条失败了 ,则另外一条也会失败.

Girl save(Girl girl) {  
 return mapper.save(girl);  
}

@Test  
 @Transactional  
 public void save() throws Exception {  
  
 service.save(new Girl("yang", "z", 23));  
 service.save(new Girl("shi", "bb", 23));  
// Assert.assertNotNull(result);  
  
 }

# 表单验证





@value 表示要表单验证的po,bindingresult是错误的返回对象.

# aop

<http://blog.csdn.net/liuchuanhong1/article/details/55099753>

|  |
| --- |
| @Aspect @Component |
| @Around("log()") public Object doAround(ProceedingJoinPoint pjp) throws Throwable {  RequestAttributes attributes = RequestContextHolder.getRequestAttributes();  ServletRequestAttributes attribute = (ServletRequestAttributes) attributes;  HttpServletRequest request = attribute.getRequest();  System.out.println("模拟事物开始"); Object[] args = pjp.getArgs();  for (Object arg : args) {  if (arg instanceof Girl) {  ((Girl) arg).setName("新增" + ((Girl) arg).getName());  } } Object result = pjp.proceed(); System.out.println("执行方法后..."); return result; |

# 统一异常

## 自定义异常

public class GirlException extends RuntimeException {  
  
 private int code;  
  
 public GirlException(String message, int code) {  
 super(message);  
 this.code = code;  
 }

## control外捕获异常

@ControllerAdvice  
public class ExceptionHandle {  
  
 @ExceptionHandler  
 @ResponseBody  
 public Result handle(Exception e) {  
 if (e instanceof GirlException) {  
 return new Result(((GirlException) e).getCode(), e.getMessage(), null);  
 }  
  
 return null;  
 }  
}

## 常用枚举

public enum ResultEnum {  
  
 UNKONW\_ERROR(-1, "未知错误"), SUCCESS(0, "成功");  
  
 private int code;  
 private String msg;  
  
 public int getCode() {  
 return code;  
 }  
  
  
 public String getMsg() {  
 return msg;  
 }  
  
  
 ResultEnum(int code, String msg) {  
  
 this.code = code;  
 this.msg = msg;  
 }  
}

# 单元测试

## 一般测试

@RunWith(SpringRunner.class)  
@SpringBootTest

## controller测试

后面添加@AutoConfigureMockMvc

# thymeleaf

因为他是严格解析html文件所以取消严格模式,得改成

mode: LEGACYHTML5

同时添加

<dependency>  
 <groupId>net.sourceforge.nekohtml</groupId>  
 <artifactId>nekohtml</artifactId>  
 <version>1.9.22</version>  
</dependency>