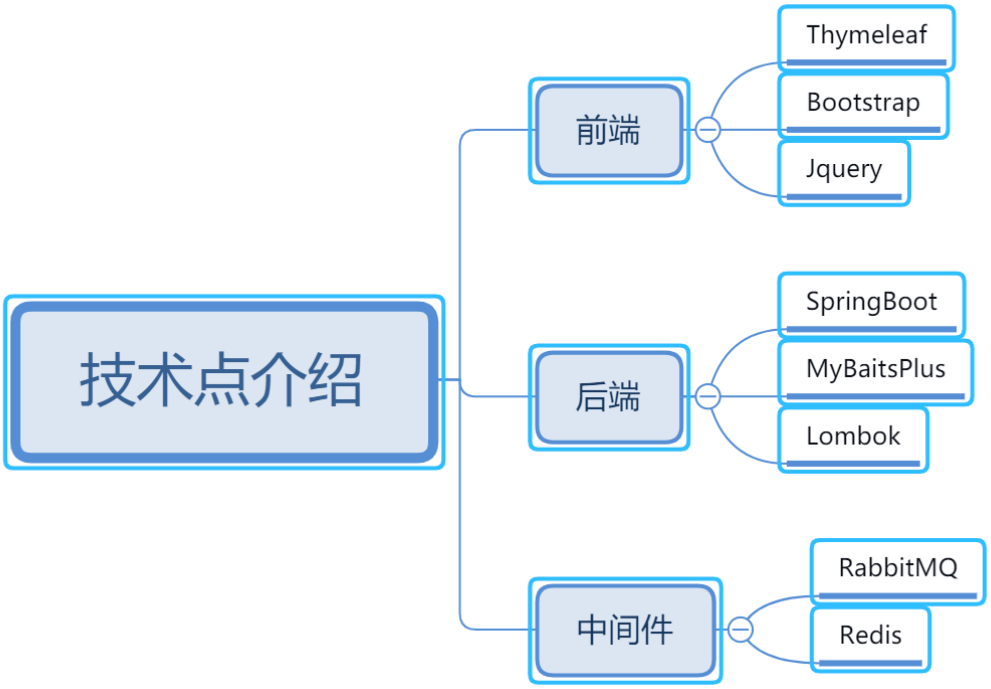

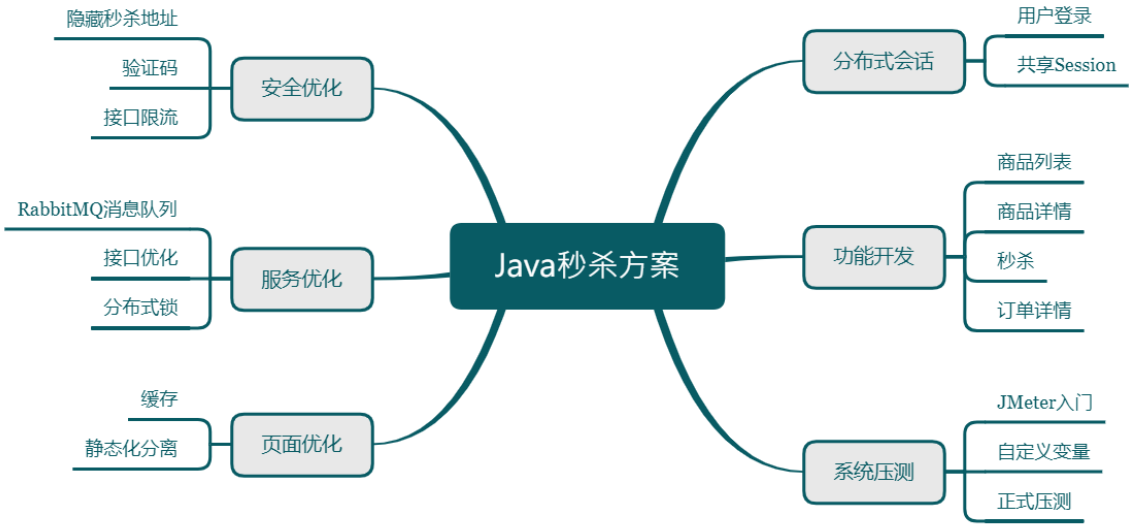

utf8和utf8mb4的区别

MySQL在5.5.3之后增加了这个utf8mb4的编码，mb4就是most bytes 4的意思，专门用来兼容四字节的unicode。好在utf8mb4是utf8的超集，除了将编码改为utf8mb4外不需要做其他转换。一般情况下使用utf8也就够了。

既然utf8能够存下大部分中文汉字,那为什么还要使用utf8mb4呢? 原来mysql支持的 utf8 编码最大字符长度为 3 字节，如果遇到 4 字节的宽字符就会插入异常了。三个字节的 UTF-8 最大能编码的 Unicode 字符是 0xffff，也就是 Unicode 中的基本多文种平面(BMP)。**utf8 不支持任**

何不在基本多文本平面的 Unicode字符。包括 Emoji 表情(Emoji 是一种特殊的 Unicode 手机上常见)、生僻字等等。

```
/*
 Navicat Premium Data Transfer

 Source Server : 本地数据库
 Source Server Type : MySQL
 Source Server Version : 80022
 Source Host : localhost:3306
 Source Schema : seckill

 Target Server Type : MySQL
 Target Server Version : 80022
 File Encoding : 65001

 Date: 18/01/2021 10:38:24
*/

SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for t_user
-- ----------------------------
DROP TABLE IF EXISTS `t_user`;
CREATE TABLE `t_user` (
  `id` bigint NOT NULL COMMENT '用户id，手机号码',
  `nickname` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `password` varchar(32) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT 'Md5加密 两次',
  `slat` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT ' 加盐',
  `head` varchar(128) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '头像',
  `register_date` datetime NULL DEFAULT NULL COMMENT '注册时间',
  `last_login_date` datetime NULL DEFAULT NULL,
  `login_count` int NULL DEFAULT 0 COMMENT '登录次数',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
```

```
35
36 -- ----------------------------
37 -- Records of t_user
38 -- ----------------------------
39
40 SET FOREIGN_KEY_CHECKS = 1;
41
```

Md5加密Util

```
1  package com.xxxx.seckill.util;
2
3
4  import org.apache.commons.codec.digest.DigestUtils;
5
6  /*
7   *@program:seckill-demo
8   *@author: Tomasonlee
9   *@Time: 2021/1/18 10:43
10  */
11 public class Md5Util {
12  public static String md5(String args) {
13  return DigestUtils.md5Hex(args);
14  }
15  private static final String salt="1a2b3c4d";
16
17  public static String inputPassToFormPass(String inputPass){
18  String str = salt.charAt(0)+salt.charAt(2)+inputPass+salt.charAt(5)+sal
t.charAt(4);
19  return md5(str);
20  }
21
22  //二次加密 后端密码到数据库密码
23  public static String fromPassToDBPass(String formPass,String salt){
24  String str =
salt.charAt(0)+salt.charAt(2)+formPass+salt.charAt(5)+salt.charAt(4);
25  return md5(str);
26  }
27
28  public static String inputPassToDBPass(String inputPass,String salt){
29  String fromPass = inputPassToFormPass(inputPass);
30  String dbPass = fromPassToDBPass(fromPass,salt);
31  return dbPass;
```

```
 }
// public static void main(String[] args) {
// System.out.println(inputPassToFormPass("123456"));
// System.out.println(fromPassToDBPass("ce21b747de5af71ab5c2e20ff0a60eea","1a2b3c4d"));
// System.out.println(inputPassToDBPass("123456","1a2b3c4d"));
// }

}
```

代码JSR303参数校验简化代码

1. 依赖

```
<!-- validation组件 -->
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

2. @Valid注解

```
public RespBean doLogin(@Valid LoginVo loginVo){
 log.info("输入的登录信息为{}",loginVo);
 return userService.doLogin(loginVo);
}
```

3. @notnull等注解 + 自定义注解

```
@Data
public class LoginVo {
 @NotNull
 @IsMobile
 private String mobile;
 @NotNull
 @Length(min=32)
 private String password;
}
```

4. 自定义注解@IsMobile

```
package com.xxxx.seckill.validator;

import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.*;

/*
```

```
 *@program:seckill-demo
 *@author: Tomasonlee
 *@Time: 2021/1/18 19:16
 * lee写的第一个自定义注解
 */
@Target({ElementType.METHOD, ElementType.FIELD, ElementType.ANNOTATION_TYPE, ElementType.CONSTRUCTOR, ElementType.PARAMETER, ElementType.TYPE_USE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Constraint(
 validatedBy = {IsMobileValidator.class}
)
public @interface IsMobile {

 boolean required() default true;

 String message() default "手机号码格式错误";

 Class<?>[] groups() default {};

 Class<? extends Payload>[] payload() default {};
}
```

IsMobileValidator

```
package com.xxxx.seckill.validator;

import com.xxxx.seckill.util.ValidatorUtil;
import org.springframework.util.StringUtils;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

/*
 *@program:seckill-demo
 *@author: Tomasonlee
 *@Time: 2021/1/18 19:19
 */
public class IsMobileValidator implements ConstraintValidator<IsMobile, String> {
 private boolean required = false;

 @Override
```

```
18   public void initialize(IsMobile constraintAnnotation) {
19   required = constraintAnnotation.required();
20   }
21
22   @Override
23   public boolean isValid(String value, ConstraintValidatorContext
context) {
24   if (required) {
25   return ValidatorUtil.isMobile(value);
26   } else {
27   if (StringUtils.isEmpty(value)) {
28   return true;
29   } else {
30   return ValidatorUtil.isMobile(value);
31   }
32   }
33   }
34  }
```

ValidatorUtil

```
1  package com.xxxx.seckill.util;
2
3  import org.springframework.util.StringUtils;
4
5  import java.util.regex.Matcher;
6  import java.util.regex.Pattern;
7
8  /*
9   *@program:seckill-demo
10   *@author: Tomasonlee
11   *@Time: 2021/1/18 19:23
12   */
13  public class ValidatorUtil {
14   private static final Pattern mobile_pattern = Pattern.compile("[1]([3-
9])[0-9]{9}$");
15
16   public static boolean isMobile(String mobile) {
17   if (StringUtils.isEmpty(mobile)) {
18   return false;
19   }
20   Matcher matcher = mobile_pattern.matcher(mobile);
21   return matcher.matches();
```

```
22    }
23 }
```

异常处理

springboot的全局异常处理方式两种

- @ExceptionHandler和@RestControllerAdvice**控制器异常（自由度高）**
- ErrorController类 **所有的异常（未进入控制器的也可以）**

1. 定义Pojo异常处理类GlobalException

分布式Session

问题复现：一台Tomcat上，没问题。部署多台，配合Nginx时用户登录失败问题。

问题原因：由于 Nginx 使用默认负载均衡策略（轮询），请求将会按照时间顺序逐一分发到后端应用上。也就是说刚开始我们在 Tomcat1 登录之后，用户信息放在 Tomcat1 的 Session 里。过了一会，请求又被 Nginx 分发到了 Tomcat2 上，这时 Tomcat2 上 Session 里还没有用户信息，就要重新登录。

解决方案：

- Session复制

无需修改代码，只需要修改Tomcat配置

Session同步传输占用内网带宽

多台Tomcat同步性能指数级下降

Session占用内存，无法有效水平扩展

- 前端存储

不占用服务端内存

存在安全风险

数据大小受cookie限制

占用外网带宽

- Session粘滞

无需修改代码

服务端可以水平扩展

增加新机器，会重新Hash，导致重新登录

应用重启，需要重新登录

- 后端集中存储

安全

容易水平扩展

增加复杂度

需要修改代码

Redis引入

```
redis-cli 客户端 -p端口-h地址-a密码
select 3 按下标号选择数据库 一共15自己设置的
操作string hash list set sortset
1 string:  set get mset mget
2 hash:
```

- 分布式Session

1. 引入依赖

```
<!--分布式Session-->
<!-- spring data redis 依赖 -->
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<!-- commons-pool2 对象池依赖 -->
<dependency>
 <groupId>org.apache.commons</groupId>
  <artifactId>commons-pool2</artifactId>
 </dependency>
 <!-- spring-session 依赖 -->
 <dependency>
  <groupId>org.springframework.session</groupId>
  <artifactId>spring-session-data-redis</artifactId>
 </dependency>
```

2. 添加配置

```
spring:
redis:
   #超时时间
  timeout: 10000ms
   #服务器地址
  host: 192.168.10.100
   #服务器端口
  port: 6379
   #数据库
  database: 0
    #密码
  password: root
  lettuce:
   pool:
     #最大连接数，默认8
```

```
    max-active: 1024
     #最大连接阻塞等待时间，默认-1
    max-wait: 10000ms
     #最大空闲连接
    max-idle: 200
     #最小空闲连接
    min-idle: 5
```

```
PS:在图形界面使用 ctrl+alt+F2切换到dos界面
dos界面 ctrl+alt+F2切换回图形界面
在命令上 输入 init 3 命令 切换到dos界面
输入 init 5命令 切换到图形界面
如果想系统默认 以某种方式启动
使用systemd创建符号链接指向默认运行级别。
 修改方法为：
1.首先删除已经存在的符号链接
-------------------------------------------------------------------------------
 rm /etc/systemd/system/default.target
-------------------------------------------------------------------------------
 2.默认级别转换为3(文本模式)
-------------------------------------------------------------------------------
 ln -sf /lib/systemd/system/multi-user.target
/etc/systemd/system/default.target
-------------------------------------------------------------------------------
 或者默认级别转换为5(图形模式)
-------------------------------------------------------------------------------
 ln -sf /lib/systemd/system/graphical.target
/etc/systemd/system/default.target
-------------------------------------------------------------------------------
 3.重启 reboot
-------------------------------------------------------------------------------
 注：ps -ef|grep xxx命
 grep命令是查找
 中间的|是管道命令 是指ps命令与grep同时执行
 PS是LINUX下最常用的也是非常强大的进程查看命令
```

```
26 grep命令是查找，是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。
27 grep全称是Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。
28 以下这条命令是检查redis进程是否存在：ps -ef |grep redis
29
30 字段含义分别如下：
31 UID ：程序被该 UID 所拥有
32 PID ：就是这个程序的 ID
33 PPID ：则是其上级父程序的ID
34 C ：CPU使用的资源百分比
35 STIME ：系统启动时间
36 TTY ：登入者的终端机位置
```

遇到问题，连接不上虚拟机redis 6379

解决办法：

查看虚拟机是否开启6379端口(注意：firewall命令只支持centos7及以上版本才可使用)

1. 查看开放端口：firewall-cmd --list-ports
2. 未添加，添加：firewall-cmd --zone=public --add-port=6379/tcp --permanent
3. 添加后记得重新加载防火墙：firewall-cmd --reload

命令简单介绍：

–zone #作用域

–add-port=80/tcp #添加端口，格式为：端口/通讯协议

–permanent #永久生效，没有此参数重启后失效

每一个方法都要判断用户信息，有咩有登录

接受参数之前 获取user做校验。用到MVC

开始：参数校验麻烦 用了@validation组件 全局异常处理

分布式session问题 我们选择把登录信息存到服务器 springsession 或者redis存储

然后 每个接口都去校验ticket空不空和用户信息 就加入mvc登录块

秒杀开始！！！

设计表：商品表、订单表、秒杀表、秒杀订单

```
/*
 Navicat Premium Data Transfer

 Source Server : 本地数据库
 Source Server Type : MySQL
 Source Server Version : 80022
 Source Host : localhost:3306
 Source Schema : seckill

 Target Server Type : MySQL
 Target Server Version : 80022
 File Encoding : 65001

 Date: 25/01/2021 14:51:22
*/

SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for goods
-- ----------------------------
DROP TABLE IF EXISTS `goods`;
CREATE TABLE `t_goods` (
  `id` bigint NOT NULL AUTO_INCREMENT COMMENT '商品ID',
  `goods_name` varchar(16) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '商品名称',
  `goods_title` varchar(64) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '商品标题',
  `goods_img` varchar(64) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL DEFAULT NULL COMMENT '商品图片',
  `goods_detail` longtext CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NULL COMMENT '商品详情',
  `goods_price` decimal(10, 0) NULL DEFAULT 0 COMMENT '商品价格',
  `goods_stock` int NULL DEFAULT 0 COMMENT '库存，-1表示不限量',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;

-- ----------------------------
-- Records of goods
```

```
-- ----------------------------

SET FOREIGN_KEY_CHECKS = 1;

```

int和Integer的区别

1、Integer是int的包装类，int则是java的八种基本数据类型中的一种(byte,short,int,long,float,double,boolean,char)

2、Integer变量必须实例化后才能使用，而int变量不需要

3、Integer实际是对象的引用，当new一个Integer时，实际上是生成一个指针指向此对象；而int则是直接存储数据值

4、Integer的默认值是null，int的默认值是0

做springboot项目中mapper.xml映射文件的时候，当你在做插入操作的时候变得尤为重要的两点知识

a.定义属性值int类型的时候，在数据库中默认null，当插入操作的时候会把默认值变成0

b.定义属性值Integer类型的时候，在数据库中默认也是null,但是当插入操作的的时候默认值会变成null。

springboot静态资源 自动配置 约定大于配置!

如果出现图片无法访问的情况需要修改MvcConfig配置类。否则无需修改此配置类

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
registry.addResourceHandler("/**").addResourceLocations("classpath:/static/");
}
```