### 电商行业技术特点

技术新

技术范围广

分布式

高并发、集群、负载均衡、高可用

海量数据

业务复杂

系统安全

### 通用mapper条件查询

@Test

**public** **void** testSelectCountByExample() {

List idArr = **new** ArrayList();

idArr.add(4L);

idArr.add(5L);

//创建一个查询条件

Example example = **new** Example(User.**class**);

example.createCriteria().andIn("id", idArr);

List<User> list = **this**.commonUserMapper.selectByExample(example);

**for** (User user : list) {

System.***out***.println(user);

}

}

### RESTful风格处理器的返回类型ResponseEntity

@RequestMapping(method = RequestMethod.***GET***)

**public** ResponseEntity<List<ItemCat>>queryItemCatBy(@RequestParam(value="id",defaultValue="0")Long parentId){

ItemCat itemCat = **new** ItemCat();

itemCat.setParentId(parentId);;

List<ItemCat> list = itemCatService.queryListByWhere(itemCat);

//方法返回类型是ResponseEntity<Void>,即返回头信息，不返回响应体，也就是不需要响应数据。比如增、删、改不需要返回值时。

//ResponseEntity.ok().build();

//方法返回类型是ResponseEntity<Object>,即返回头信息，也返回响应体，也就是需要响应数据。比如返回查询的记录、返回插入数据时影响记录数。

//ResponseEntity.ok().body(list);

//有响应数据的简便写法。

**return** ResponseEntity.*ok*(list);

}

### 项目中的亮点1

**public** **abstract** **class** BaseService<T **extends** BasePojo> {

/\*\*

\* Spring4增加了对泛型注入的支持，这个特性对通用Mapper来说，非常的有用。

\* 此特性使得可以通过Mapper<T>来注入通用mapper。

\* Spring4以前，注入属性的类型，需要是具体的类或接口(ItemMapper)，现在可以是带泛型的类或接口(Mapper<T>)。

\*/

@Autowired

**private** Mapper<T> mapper;

/\*\*

\* 获得泛型的真实类型

\*/

**private** Class<T> clazz;

BaseService(){

ParameterizedType type = (ParameterizedType) **this**.getClass().getGenericSuperclass();

**this**.clazz = (Class<T>) type.getActualTypeArguments()[0];

}

}

### 项目中的亮点2

Niginx解决域名映射中端口的问题

实现负载均衡(实现负载均衡的策略：轮询访问、流量控制)

### 项目中的亮点3

商品信息表(Item)的数据库优化

商品价格在数据库中用bigint存储(单位从元变成分)，避免小数点问题。

把信息量大、修改频率低的“商品信息描述”属性单独设置成一个子表。可以使用Redis提高访问性能，减小网络流量压力

### 业务层异常要抛出不要捕获

因为pring事务管理中，事务传递机制设置的是**PROPAGATION\_REQUIRED : 如果A有事务,那么B也会使用A的事务.如果A没有事务.B会开启一个事务.将A包含进来.(保证这两个操作在一个事务中.)。**如果自己把异常捕获了，Spring就不知道业务中是否发生了异常，就不会回滚事务。

### @ResponseBody和ResponseEntity<T>

//如果设置了注解@ResponseBody，或者返回类型是ResponseEntity<T>，那么返回结果就会被转换成JSON格式

//并且返回值不再被视图解析器拦截，也就是说前台页面不会跳转。

@ResponseBody

**public** ResponseEntity<EasyUIResult> queryItemPageByWhere(

Item item,

@RequestParam(value="page",defaultValue="1")Integer page,

@RequestParam(value="rows",defaultValue="30")Integer rows

){

... ...

}

### 使用注解设置setContentType

//设置produces = MediaType.TEXT\_HTML\_VALUE，是因为富文本编辑器要求返回TEXT型结果。

//相当于response.setContentType(MediaType.TEXT\_HTML\_VALUE);

@RequestMapping(value = "upload", method = RequestMethod.***POST***, produces = MediaType.***TEXT\_HTML\_VALUE***)

@ResponseBody

**public** String upload(@RequestParam("uploadFile") MultipartFile uploadFile, HttpServletResponse response)

**throws** Exception {

... ...

}

### 项目中的亮点4

使用Nginx搭建静态图片服务器(需要配合使用SwitchHost，修改DNS)

server {

listen 80;

server\_name image.haitao.p165.com; //静态资源服务器路径

#charset koi8-r;

#access\_log logs/host.access.log main;

proxy\_set\_header X-Forwarded-Host $host;

proxy\_set\_header X-Forwarded-Server $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

location / {

root E:/fileupload/img; //静态资源存储路径

}

}

### 项目中的亮点5

商品规格(模板方式)，描述比较复杂，自己组织描述语言，下面列出几个重要点：

商品规格参数的特点：

1、商品分类多

2、每种商品分类的规格参数不一样，就需要创建很多规格参数表(一千多张)

3、规格参数表的结构可能会变动，改动表结构工作量大

4、规格参数表结构简单，可以看作是单表操作（商品分类与规格参数一对一，商品分类ID可作为规格参数ID）

采用解决问题的办法：

创建两张表：

1. 规格参数模板表，与商品分类表是一对一关系。即一条商品分类记录对应一条规格参数模板记录。
2. 规格参数数据表，与商品信息表是一对一关系。即一条商品信息记录对应一条规格参数数据记录。

规格参数模板 和 规格参数数据 都采用可调整的数据格式：JSON数据格式

### 项目重点1

代码注释要足够多(1/3)

使用Logger记录程序运行状态

Logger LOGGER = LoggerFactory.getLogger(ItemController.class);

LOGGER.debug("新增商品成功：itemId = {}", item.getId());

### 项目重点2（？技术过老，是否还要讲解）

富文本编辑器kindeditor实现图片上传

### 项目重点3

Spring中的父子容器。Spring子容器可以访问父容器的bean，但是无法访问父容器中的properties资源。

即@Resource、@Autowire可以夸容器注入，但是@value注解只在当前的容器中查找资源内容

### 项目重点5

前、后台项目分离，前台项目不直接访问数据库，由后台项目提供接口（服务）来进行访问

a) 优点

i. 编码工作量较小，不用编写DAO层

ii. 不用学习数据库，减小学习成本

iii. 只有一个团队进行数据库管理，数据库变更，其他团队不用修改代码

b) 缺点

i. 访问速度较慢，性能较低

ii. 如果需要调用的服务很多，服务之间的调用会非常复杂

解决方案：服务注册中心

### 项目中的亮点6

内容管理系统（CMS）

如果每一个前端展示模块（大广告位、小广告位等等），单独建立表，进行CRUD操作，会有以下问题

• 页面制作无序，网站风格不统一，大量信息堆积，发布显得异常沉重；

• 内容繁杂，手工管理效率低下，手工链接视音频信息经常无法实现；

• 应用难度较高，许多工作需要技术人员配合才能完成，角色分工不明确；

• 改版工作量大，系统扩展能力差，集成其它应用时更是降低了灵活性；

数据库结构，创建两张表：

内容分类表：存储分类（模块）信息，重要字段：parent\_id，is\_parent，sort\_order(优先级，谁给的钱多，谁优先级高，谁排名靠前)(疑问：优先级不应该设置在内容表里面么？)

内容表：存储商品的详细信息，例如标题、描述、图片、商家链接

### 项目中的亮点7

HttpClient

### 项目中的亮点8

Redis缓存菜单、大广告位、商品详情、商品描述、商品规格等数据

缓存分为前台缓存和后台缓存

缓存的时候，KEY的值要用指定的分类明显、意义明确的前缀，因为缓存内容会很多，分类前缀使维护容易。

缓存的清理。后台修改商品信息后，调用前台接口对前台的缓存内容进行清理（谁的内容谁负责）。

### 项目重点4

缓存数据库RedisService方法的抽取

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*day81\_商品详情\_缓存\_单点登录SSO\_用户注册\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 项目中的亮点9

SSO单点登录中，用户登录时根据用户名(username)查询数据库，因为username不经常被改变，我们可以对其建立索引，提高查询效率，而用户密码会经常变化不适合做索引列。

用户名密码不能明文存放，使用MD5加密

springMVC解决中文乱码

后台向前台输出汉字乱码：在<mvc:annotation-driven>中配置StringHttpMessageConvertor

jsonp：自定义类型：public class CallbackMappingJackson2HttpMessageConverter extends MappingJackson2HttpMessageConverter {}

Hibernamt-validate，在bean的属性上添加校验注解，在Controller方法形参的Bean对象上使用@Validate

@Length(max=14,min=6,message="用户名4-16位")

@Email

jackon ObjectMapper 把Bean转换成json