# 商品详情的方案

## **网页静态化方案**

1、创建商品详情的thymeleaf模板

2、创建一个RabbitMQ的消费者，收到消息后创建静态页面

3、搭建nginx服务器，返回静态页面

## **redis缓存商品信息方案**

使用redis做缓存，业务逻辑：

1、根据商品id到redis中查询

​ 查得到，直接返回

2、查不到，查询mysql，

​ 数据放到redis中

​ 设置缓存的有效期一天的时间，可以根据实际情况调整。

需要使用String类型来保存商品数据，可以加前缀对redis中的key进行归类：

ITEM\_INFO:123456:BASE

ITEM\_INFO:123456:DESC

ITEM\_INFO:123456:PARAM

# **查询商品详情**

## **创建usian\_detail\_web**

### **pom.xml**

<dependency>

<groupId>com.usian</groupId>

<artifactId>usian\_item\_feign</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

### **application.yml**

spring:

application:

name: usian-detail-web

server:

port: 8097

eureka:

client:

service-url:

defaultZone: http://127.0.0.1:8761/eureka/

### **logback.xml**

### **创建启动类**

@SpringBootApplication

@EnableDiscoveryClient

@EnableFeignClients

public class DetailWebApp {

public static void main(String[] args) {

SpringApplication.run(DetailWebApp.class, args);

}

}

## **查询商品信息**

### **usian\_item\_service**

#### **2.2.1.1application.yml**

#商品详情

ITEM\_INFO: ITEM\_INFO

BASE: BASE

DESC: DESC

PARAM: PARAM

ITEM\_INFO\_EXPIRE: 86400

#### **2.2.1.2.service**

修改ItemServiceImpl

@Value("${ITEM\_INFO}")

private String ITEM\_INFO;

@Value("${BASE}")

private String BASE;

@Value("${DESC}")

private String DESC;

@Value("${ITEM\_INFO\_EXPIRE}")

private Integer ITEM\_INFO\_EXPIRE;

@Autowired

private RedisClient redisClient;

/\*\*

\* 查询商品信息

\* @param itemId

\* @return

\*/

@Override

public TbItem selectItemInfo(Long itemId) {

//查询缓存

TbItem tbItem = (TbItem) redisClient.get(ITEM\_INFO + ":" + itemId + ":"+ BASE);

if(tbItem!=null){

return tbItem;

}

tbItem = tbItemMapper.selectByPrimaryKey(itemId);

//把数据保存到缓存

redisClient.set(ITEM\_INFO + ":" + itemId + ":"+ BASE,tbItem);

//设置缓存的有效期

redisClient.expire(ITEM\_INFO + ":" + itemId + ":"+ BASE,ITEM\_INFO\_EXPIRE);

return tbItem;

}

/\*\*

\* 根据商品 ID 查询商品描述

\* @param itemId

\* @return

\*/

@Override

public TbItemDesc selectItemDescByItemId(Long itemId) {

//查询缓存

TbItemDesc tbItemDesc = (TbItemDesc) redisClient.get(

ITEM\_INFO + ":" + itemId + ":"+ DESC);

if(tbItemDesc!=null){

return tbItemDesc;

}

TbItemDescExample example = new TbItemDescExample();

TbItemDescExample.Criteria criteria = example.createCriteria();

criteria.andItemIdEqualTo(itemId);

List<TbItemDesc> itemDescList =

this.tbItemDescMapper.selectByExampleWithBLOBs(example);

if(itemDescList!=null && itemDescList.size()>0){

//把数据保存到缓存

redisClient.set(ITEM\_INFO + ":" + itemId + ":"+ DESC,itemDescList.get(0));

//设置缓存的有效期

redisClient.expire(ITEM\_INFO + ":" + itemId + ":"+ DESC,ITEM\_INFO\_EXPIRE);

return itemDescList.get(0);

}

return null;

}

#### 修改ItemParamServiceImpl

@Value("${ITEM\_INFO}")

private String ITEM\_INFO;

@Value("${PARAM}")

private String PARAM;

@Value("${ITEM\_INFO\_EXPIRE}")

private Integer ITEM\_INFO\_EXPIRE;

@Autowired

private RedisClient redisClient;

/\*\*

\* 根据商品id查询商品规格

\* @param itemId

\* @return

\*/

@Override

public TbItemParamItem selectTbItemParamItemByItemId(Long itemId) {

//查询缓存

TbItemParamItem tbItemParamItem = (TbItemParamItem) redisClient.get(

ITEM\_INFO + ":" + itemId + ":"+ PARAM);

if(tbItemParamItem!=null){

return tbItemParamItem;

}

TbItemParamItemExample example = new TbItemParamItemExample();

TbItemParamItemExample.Criteria criteria = example.createCriteria();

criteria.andItemIdEqualTo(itemId);

List<TbItemParamItem> tbItemParamItemList =

tbItemParamItemMapper.selectByExampleWithBLOBs(example);

if(tbItemParamItemList!=null && tbItemParamItemList.size()>0){

//把数据保存到缓存

redisClient.set(ITEM\_INFO + ":" + itemId + ":"+

PARAM,tbItemParamItemList.get(0));

//设置缓存的有效期

redisClient.expire(ITEM\_INFO + ":" + itemId + ":"+ PARAM,ITEM\_INFO\_EXPIRE);

return tbItemParamItemList.get(0);

}

return null;

}

#### **controller**

修改ItemController

/\*\*查询商品信息

\* 根据商品id

\* @param itemId

\* @return

\*/

@RequestMapping("/selectItemInfo")

public TbItem selectItemInfo(Long itemId){

return this.itemService.selectItemInfo(itemId);

}

/\*\*

\* 根据商品 ID 查询商品描述

\*/

@RequestMapping("/selectItemDescByItemId")

public TbItemDesc selectItemDescByItemId(Long itemId){

return this.itemService.selectItemDescByItemId(itemId);

}

修改ItemParamController

/\*\*

\* 根据商品 ID 查询商品规格

\*/

@RequestMapping("/selectTbItemParamItemByItemId")

public TbItemParamItem selectTbItemParamItemByItemId(@RequestParam Long itemId){

return itemParamService.selectTbItemParamItemByItemId(itemId);

}

### **usian\_detail\_web**

#### **Controller**

@RestController

@RequestMapping("/frontend/detail")

public class DetailController {

@Autowired

private ItemServiceFeign itemServiceFeign;

/\*\*

\* 查询商品基本信息

\*/

@RequestMapping("/selectItemInfo")

public Result selectItemInfo(Long itemId) {

TbItem tbItem = itemServiceFeign.selectItemInfo(itemId);

if (tbItem != null) {

return Result.ok(tbItem);

}

return Result.error("查无结果");

}

/\*\*

\* 查询商品介绍

\*/

@RequestMapping("/selectItemDescByItemId")

public Result selectItemDescByItemId(Long itemId){

TbItemDesc tbItemDesc = itemServiceFeign.selectItemDescByItemId(itemId);

if(tbItemDesc != null){

return Result. ok (tbItemDesc);

}

return Result.error ("查无结果");

}

/\*\*

\* 根据商品 ID 查询商品规格参数

\*/

@RequestMapping("/selectTbItemParamItemByItemId")

public Result selectTbItemParamItemByItemId(Long itemId){

TbItemParamItem tbItemParamItem =

itemServiceFeign.selectTbItemParamItemByItemId(itemId);

if(tbItemParamItem != null){

return Result.ok(tbItemParamItem);

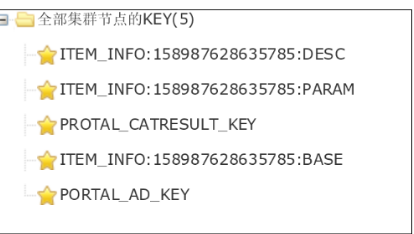
}

return Result. error ("查无结果");

}

}

测试结果：

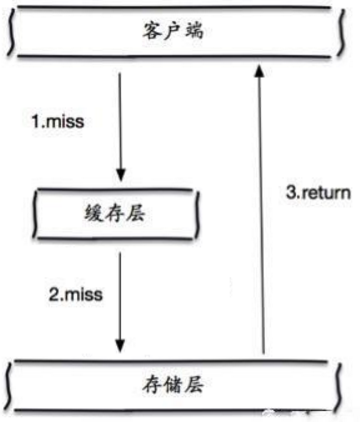


# **缓存同步**

# **后台修改或删除商品时直接删除redis中的商品即可完成Redis中的缓存同步。**

# **缓存穿透**

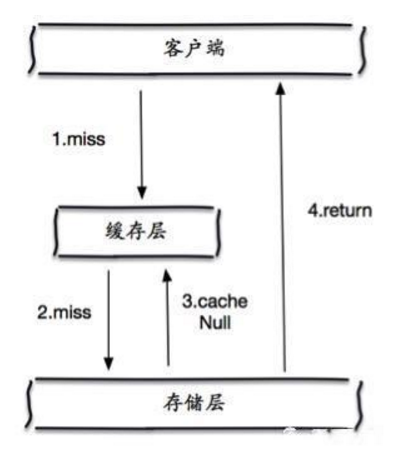
**缓存穿透是指缓存和数据库中都没有数据，而用户不断发起请求则这些请求会穿过缓存直接访问数据库，如发起为id为“-1”的数据或id为特别大不存在的数据。假如有恶意攻击，就可以利用这个漏洞，对数据库造成压力，甚至压垮数据库。**



**解决方案：**

**缓存空对象：**

**当存储层不命中后，即使返回的空对象也将其缓存起来，同时会设置一个过期时间（避免控制占用更多的存储空间），之后再访问这个数据将会从缓存中获取，保护了后端数据源；**



## **usian\_item\_service**

修改ItemServiceImpl

/\*\*

\* 查询商品信息

\* @param itemId

\* @return

\*/

@Override

public TbItem selectItemInfo(Long itemId) {

//查询缓存

TbItem tbItem = (TbItem) redisClient.get(ITEM\_INFO + ":" + itemId + ":"+ BASE);

if(tbItem!=null){

return tbItem;

}

tbItem = tbItemMapper.selectByPrimaryKey(itemId);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*解决缓存穿透\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(tbItem == null){

//把空对象保存到缓存

redisClient.set(ITEM\_INFO + ":" + itemId + ":"+ BASE,null);

//设置缓存的有效期

redisClient.expire(ITEM\_INFO + ":" + itemId + ":"+ BASE,30);

return tbItem;

}

//把数据保存到缓存

redisClient.set(ITEM\_INFO + ":" + itemId + ":"+ BASE,tbItem);

//设置缓存的有效期

redisClient.expire(ITEM\_INFO + ":" + itemId + ":"+ BASE,ITEM\_INFO\_EXPIRE);

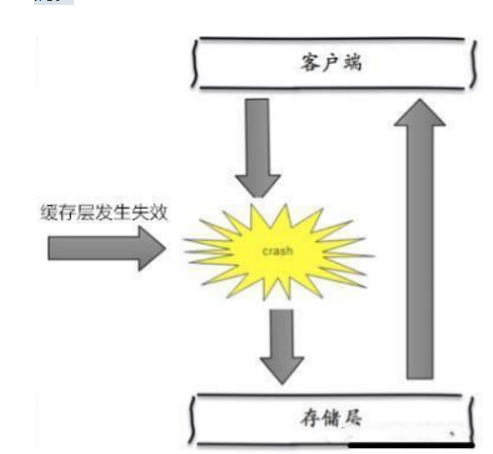
return tbItem;

}

**同理，商品描述和商品规格参数信息是同样的操作**

# 缓存击穿

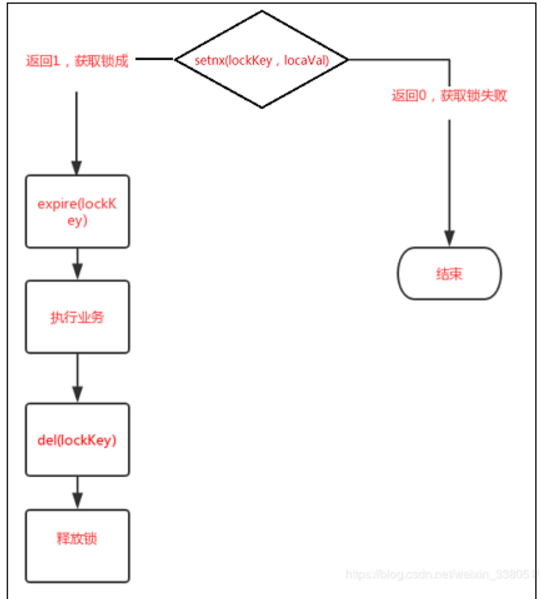
缓存击穿，是指一个key非常热点，在不停的扛着大并发，大并发集中对这一个key不停进行访问，当这个key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库，就像在一个屏障上凿开了一个洞。



**解决方案：**

**1、设置热点数据永远不过期**

**2、加分布式锁**



1、如何释放锁？del

2、业务处理失败？expire

## **common\_redis**

**修改RedisClient，添加方法：**

/\*\*

\* 分布式锁

\* @param key

\* @param count

\* @param value

\* @return

\*/

public Boolean setnx(String key, Object value, long time) {

try {

return redisTemplate.opsForValue().setIfAbsent(key, value, time,

TimeUnit.SECONDS);

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

## **usian\_item\_service**

**修改ItemServiceImpl：**

/\*\*

\* 查询商品信息

\* @param itemId

\* @return

\*/

@Override

public TbItem selectItemInfo(Long itemId){

//1、先查询redis,如果有直接返回

TbItem tbItem = (TbItem) redisClient.get(ITEM\_INFO+":"+itemId+":"+BASE);

if(tbItem!=null){

return tbItem;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*解决缓存击穿\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(redisClient.setnx(SETNX\_BASC\_LOCK\_KEY+":"+itemId,itemId,30L)){

//2、再查询mysql,并把查询结果缓存到redis,并设置失效时间

tbItem = tbItemMapper.selectByPrimaryKey(itemId);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*解决缓存穿透\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if(tbItem!=null){

redisClient.set(ITEM\_INFO+":"+itemId+":"+BASE,tbItem);

redisClient.expire(ITEM\_INFO+":"+itemId+":"+BASE,ITEM\_INFO\_EXPIRE);

}else{

redisClient.set(ITEM\_INFO+":"+itemId+":"+BASE,null);

redisClient.expire(ITEM\_INFO+":"+itemId+":"+BASE,30L);

}

redisClient.del(SETNX\_BASC\_LOCK\_KEY+":"+itemId);

return tbItem;

}else{

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

return selectItemInfo(itemId);

}

}

/\*\*

\* 根据商品 ID 查询商品介绍

\* @param itemId

\* @return

\*/

@Override

public TbItemDesc selectItemDescByItemId(Long itemId) {

//1、先查询redis,如果有直接返回

TbItemDesc tbItemDesc = (TbItemDesc) redisClient.get(ITEM\_INFO + ":" + itemId +

":" + DESC);

if(tbItemDesc!=null){

return tbItemDesc;

}

if(redisClient.setnx(SETNX\_DESC\_LOCK\_KEY+":"+itemId,itemId,30L)){

//2、再查询mysql,并把查询结果缓存到redis,并设置失效时间

tbItemDesc = tbItemDescMapper.selectByPrimaryKey(itemId);

if(tbItemDesc!=null){

redisClient.set(ITEM\_INFO + ":" + itemId + ":" + DESC,tbItemDesc);

redisClient.expire(ITEM\_INFO + ":" + itemId + ":" +

DESC,ITEM\_INFO\_EXPIRE);

}else{

redisClient.set(ITEM\_INFO + ":" + itemId + ":" + DESC,null);

redisClient.expire(ITEM\_INFO + ":" + itemId + ":" + DESC,30L);

}

redisClient.del(SETNX\_DESC\_LOCK\_KEY+":"+itemId);

return tbItemDesc;

}else{

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

return selectItemDescByItemId(itemId);

}

}

**修改ItemParamServiceImpl**

/\*\*

\* 根据商品id查询商品规格

\* @param itemId

\* @return

\*/

@Override

public TbItemParamItem selectTbItemParamItemByItemId(Long itemId) {

//1、先查询redis,如果有直接返回

TbItemParamItem tbItemParamItem = (TbItemParamItem) redisClient.get(ITEM\_INFO +

d":" + itemId + ":" + PARAM);

if(tbItemParamItem!=null){

return tbItemParamItem;

}

if(redisClient.setnx(SETNX\_PARAM\_LOCK\_KEY+":"+itemId,itemId,30L)){

//2、再查询mysql,并把查询结果缓存到redis,并设置失效时间

TbItemParamItemExample tbItemParamItemExample = new TbItemParamItemExample();

TbItemParamItemExample.Criteria criteria =

tbItemParamItemExample.createCriteria();

criteria.andItemIdEqualTo(itemId);

List<TbItemParamItem> tbItemParamItems =

tbItemParamItemMapper.selectByExampleWithBLOBs(tbItemParamItemExample);

if(tbItemParamItems!=null && tbItemParamItems.size()>0){

tbItemParamItem = tbItemParamItems.get(0);

redisClient.set(ITEM\_INFO + ":" + itemId + ":" + PARAM,tbItemParamItem);

redisClient.expire(ITEM\_INFO + ":" + itemId + ":" +

PARAM,ITEM\_INFO\_EXPIRE);

}else{

redisClient.set(ITEM\_INFO + ":" + itemId + ":" + PARAM,null);

redisClient.expire(ITEM\_INFO + ":" + itemId + ":" + PARAM,30L);

}

redisClient.del(SETNX\_PARAM\_LOCK\_KEY+":"+itemId);

return tbItemParamItem;

}else{

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

return selectTbItemParamItemByItemId(itemId);

}

}