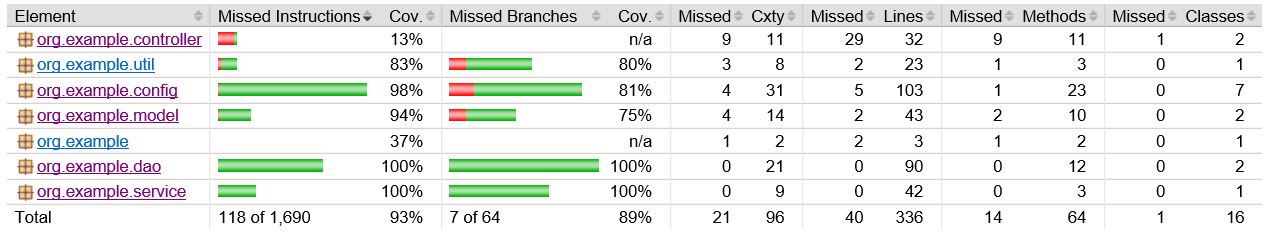
# 运行结果



# 整体说明

本系统实现两个功能，第一个是根据长域名生成一个唯一的8位短域名；第二个是根据8位短域名获取该短域名对应的长域名。

本系统服务层采用Spring-boot微服务。

缓存采用ConcurrntHashMap，既保证数据高效缓存，又保证高并发访问。

数据存储采用分布式存储数据库ElasticSearch，可以实现海量数据的存储。



## 第一个是根据长域名自动生成8位短域名映射。

域名的生成使用10位数字，26位小写字母，26位大写字母组合的62进制。这样总共能够生成的8位域名数量是62的8次幂，这是一个天文数字，相信能够满足未来的需要。

实现高性能，采用缓存设计，将长域名与8位短域名的映射存储到缓存中。长短域名映射实现快算响应。

防止缓存溢出，缓存的退出机制采用最近最少使用算法。即当缓存占满时，将最近最少使用的域名从缓存中剔除。

实现高并发，缓存采用的是ConcurrentHashMap。

## 第二个是根据8位短域名返回与之映射的长域名。

直接在缓存中根据8位短域名映射长域名，获取后返回长域名。

## 关于缓存击穿

对于没有映射到的数据，实际生产中应该将这些数据放入到数据库中。在数据查询时，先检索缓存，缓存没有查到再去查找数据库。

本实例是一个Demo，未进行这方面处理。

# 工作流程

## 根据长域名获取8位短域名

### 长域名获取8位短域名流程

(1)根据当前长域名在缓存中获取8位短域名映射,如果获取到，则直接返回8位短域名。

(2)如果没有获取到，系统到数据库中获取短域名。如果获取到，则将该短域名放到缓存中，返回短域名。

如果数据库中没有获取到短域名，则表明还未生成该长域名的短域名。系统会自动生成一个新的8位短域名。

如果缓存已满，则删除最近最少使用的8位短域名映射节点，将最新的节点插入到缓存中。

(3)最后系统返回该长域名对应的8位短域名。

图1长域名获取短域名流程图

### 最新8位短域名获取子流程

第一：设置一个8位字符串，组成一个 10位数字，26位小写数字，26位大写数字组成的62位进制数字数列---SHORT\_URL[8]。每一位的最小值位‘0’，最大值为’Z’。

第二：设置一个位置数字，初始为0，叫做当前位A。

1. 如果当前位的数值SHORT\_URL[A]为最大值，即位’Z’。则将SHORT\_URL[A]设为’0’, A的值+1。再重复(1)的操作。
2. 将最新的8位字符组成字符串。
3. 返回最新字符串。



图2获取最新短域名

## 8位短域名获取长域名

1. 在缓存中获取该8位短域名的长域名。如果找到则直接返回。
2. 如果没有找到则到数据库中查找该8位短域名对应的长域名。

如果找到，则将该长域名与8位短域名放到缓存中。然后返回长域名。

1. 如果数据库中未找到该8位短域名对应的长域名，说明该8位短域名没有对应的长域名，返回null.



# 算法

## 长域名与8位短域名

短域名采用自增方式获得。

8位短域名采用10个数字，26个小写英文字母，26个大写英文字母，一共是10 + 26 +26 = 62进制8位数字组成。

8位一共可以存储的数据量是62 的8次幂。 这是一个天文数字，相信够满足未来对短域名数量的要求。

具体算法为：

1. 生成一个8位字符数组ShortURL[8]，初始值位’0’;
2. 设置当前位CurrentIndex, 初始值为0
3. 判断ShortURL[CurrentIndex]是否等于’Z’
4. 如果等于’Z’,则ShortURL[CurrentIndex]=’0’;

CurrentIndex 值 加1

1. CurrentIndex是否大于7
2. CurrentIndex大于7，则耗尽所有短域名值，算法结束
3. CurrentIndex不大于7，返回(3)
4. ShortURL[CurrentIndex] 值+1
5. 由8位数组ShortURL[8]组成8位字符串
6. 返回该字符串，即为所需要的短域名



## 最近最少使用算法

最近最少使用算法，采用双链表实现。

1. 创建双链表，创建头节点，创建尾节点。
2. 将最新使用的节点放到转移到第一个节点。
3. 删除时，直接去掉最后一个节点。

# 缓存

ConcurrentHashMap

# 数据库

数据库采用分布式数据库ElasticSearch，实现海量长域名的存储。