

短域名服务

背景

技术方案

自增Id

摘要算法

普通随机数

实现方案

存储方案

测试方案

扩展思考

背景

- 什么是短域名：就是将一个长网址(长域名)缩短到一个很短的网址(短域名)，用户访问这个短网址就可以访问到原本的长网址，这样可以达到方便记忆以及转换的目的。

比如红杉中国的简介网址为：<https://www.sequoiacap.cn/china/people/introduction/>，使用新浪在线短域名服务生成的对应短网址为：<https://t.hk.uy/a87E>，通过该短域名也可以访问到红杉中国的主页。

- 短域名应用场景：
 - 微博字数限制：当我们发一些链接的时候，会超出字数限制，如果因为链接原因导致发布失败，用户体验会很差，因此短域名服务应运而生，在发布的时候会自动转换短域名，节省字数空间；
 - 短网址二维码：网址在转换成短网址时，也可以生成相应的短网址二维码，核心解决是跨平台、跨现实的数据传输问题；
 - 营销短信：在收到一些营销短信时，会发现短信中的链接一般都是短链接，这样也是为了提升用户体验，同时降低发送成本。

技术方案

短链接生成方式：短码一般是由 [a - z, A - Z, 0 - 9] 这62 个字母或数字组成，短码的长度也可以自定义，但一般不超过8位。比较常用的都是6位，6位的短码已经能有568亿种的组合： $(26+26+10)^6 = 56800235584$ ，已满足绝大多数的使用场景。目前比较流行的生成短码方式有：自增Id、摘要算法、普通随机数。

自增Id

- 实现方案：设置 id 自增，一个 10进制 id 对应一个 62进制的数值，1对1，也就不会出现重复的情况。这个利用的就是低进制转化为高进制时，字符数会减少的特性。十进制211对应的不同进制字符表

示：

21100

转换

进制	结果	解释
2	101001001101100	
8	51154	
10	21100	
16	526c	
26	bffo	小写字母
32	MKC	不包含 ILOU 字符
36	ga4	数字 + 小写字母
52	hPO	大写字母 + 小写字母
58	7gN	不包含 OOI 字符
62	Suk	数字 + 小写字母 + 大写字母

- 优点：比较容易理解，生成的短码永不重复；
- 缺点：由于是自增序列，生成的短码长度从1位开始递增，可以通过指定id初始值来实现短码长度固定；另一方面可能会存在安全问题，别人可以穷举生成的短链接来获取对应的原始链接。

摘要算法

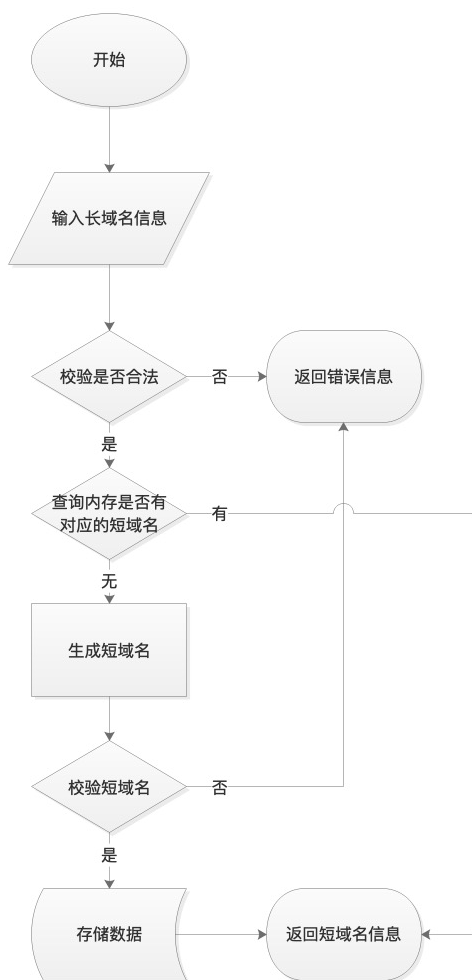
- 实现方案：摘要算法又称为哈希算法，表示输入任意长度的数据，输出固定长度的数据。具体算法过程如下：
 - a. 将长网址md5生成32位签名串,分为4段, 每段8个字节；
 - b. 对这四段循环处理, 取8个字节, 将他看成16进制串与0x3fffffff(30位1)与操作, 即超过30位的忽略处理；
 - c. 这30位分成6段, 每5位的数字作为字母表的索引取得特定字符, 依次进行获得6位字符串；
 - d. 总的md5串可以获得4个6位串；取里面的任意一个就可作为这个长url的短url地址；
- 优点：短码固定长度，不存在有序的情况，也不存在安全问题；
- 缺点：摘要算法会生成4个短码，存在碰撞的可能性，解决冲突比较麻烦。

普通随机数

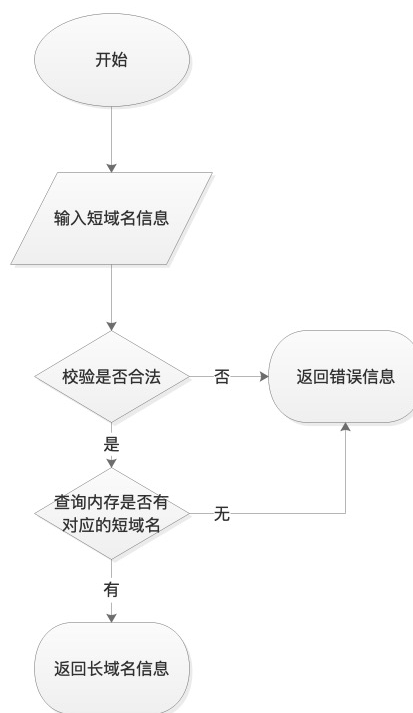
- 实现方案：从62个字符串中随机取出一个6位短码的组合，然后去数据库中查询该短码是否已存在。如果已存在，就继续循环该方法重新获取短码，否则就直接返回；
- 优点：简单容易实现，适用于业务量不大的场景
- 缺点：由于是伪随机数，碰撞的可能性较大，在数据比较多的情况下，可能需要循环多次才能生成一个不冲突的短码。

实现方案

结合各自的优缺点，采用方案二(摘要算法)的实现逻辑，流程图如下：



短域名存储接口流程图



短域名读取接口流程图

存储方案

由于题目要求映射数据存在JVM内存即可，因此采用HashMap作为存储结构，但要防止内存溢出，因此可以自定义map的初始化大小值(配置在application.properties)，并且采用LRU淘汰算法策略，保证map中的数据都是热点数据，自定义LRUCache代码如下

```

1 //自定义LRU算法存储数据
2 class LRUCache<K, V> extends LinkedHashMap<K, V> {
3     private int capacity;
4
5     public LRUCache(int capacity) {
6         super(capacity, 0.75F, true);
7         this.capacity = capacity;
8     }
9
10    @Override
11    public boolean removeEldestEntry(Map.Entry<K, V> eldest) {
12        return size() > capacity;
13    }
14 }
15
16 #短域名-长域名映射关系map大小
17 long2short.map.capacity=1000
18
19 #长域名-短域名映射关系map大小
20 short2long.map.capacity=1000









```

测试方案

- Jacoco单元测试用例执行结果如下：其中分支覆盖率85%，行覆盖率为77/78=98%。

demo

demo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods	Missed Classes
com.example.baiyang.demo.utils		98%		80%	3 17	1 41	1 12	0 3
com.example.baiyang.demo.controller		100%		83%	2 9	0 19	0 3	0 1
com.example.baiyang.demo.service.impl		100%		100%	0 5	0 12	0 3	0 1
com.example.baiyang.demo.validate		100%		100%	0 3	0 6	0 2	0 1
Total	5 of 593	99%	4 of 28	85%	5 34	1 78	1 20	0 6

```

[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 26, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:report (post-test) @ demo ---
[INFO] Loading execution data file /Users/frank/interview/target/jacoco.exec
[INFO] Analyzed bundle 'demo' with 6 classes
[INFO]

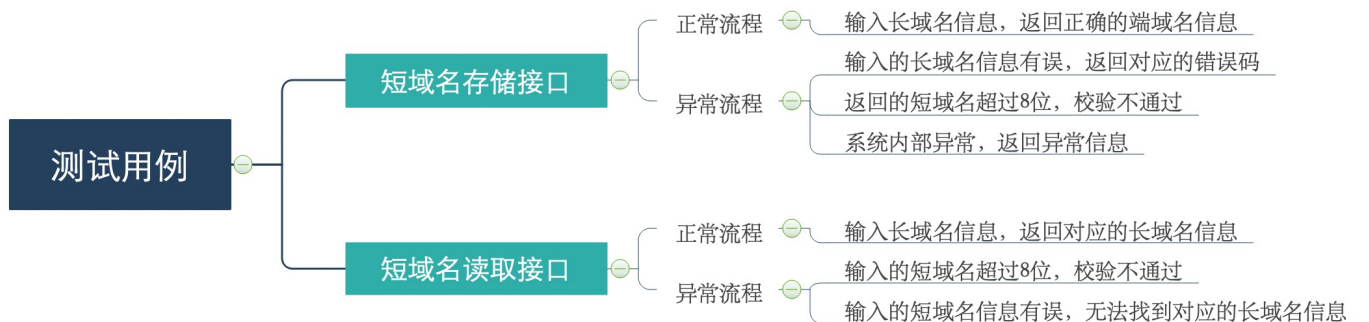
```

NOTE: 记录junit使用的一个坑：本地执行单元测试都能执行成功，但是用mvn test就无法执行单元测试，通过网上搜索发现可能是由于 maven-surefire-plugin doesn't have full support of junit5. There is an open issue about adding this support in [SUREFIRE-1206](#).但是项目中用的

是junit4, 因此打印出依赖树, 发现spring-boot-starter-test引入了junit5, 需要在pom文件里面排除掉, 问题解决。

```
└─ org.springframework:spring-expression:jar:5.3.4:compile
+- org.springframework.boot:spring-boot-starter-test:jar:2.4.3:compile
+- org.springframework.boot:spring-boot-test:jar:2.4.3:compile
+- org.springframework.boot:spring-boot-test-autoconfigure:jar:2.4.3:compile
+- com.jayway.jsonpath:json-path:jar:2.4.0:compile
|   └─ net.minidev:json-smart:jar:2.3:compile
|       └─ net.minidev:accessors-smart:jar:1.2:compile
+- jakarta.xml.bind:jakarta.xml.bind-api:jar:2.3.3:compile
|   └─ jakarta.activation:jakarta.activation-api:jar:1.2.2:compile
+- org.assertj:assertj-core:jar:3.18.1:compile
+- org.hamcrest:hamcrest:jar:2.2:compile
+- org.junit.jupiter:junit-jupiter:jar:5.7.1:compile
|   +- org.junit.jupiter:junit-jupiter-api:jar:5.7.1:compile
|   |   +- org.apiguardian:apiguardian-api:jar:1.1.0:compile
|   |   +- org.opentest4j:opentest4j:jar:1.2.0:compile
|   |   └─ org.junit.platform:junit-platform-commons:jar:1.7.1:compile
|   +- org.junit.jupiter:junit-jupiter-params:jar:5.7.1:compile
|   └─ org.junit.jupiter:junit-jupiter-engine:jar:5.7.1:runtime
|       └─ org.junit.platform:junit-platform-engine:jar:1.7.1:runtime
```

- 测试用例:



- 测试结果:
 - 正常返回结果

POST

/api/getShortUrl

根据长域名获取短域名信息

Response Class (Status 200)

OK

Model

Example Value

```
{
  "errorCode": "string",
  "errorMessage": "string",
  "response": {
    "feature": "string",
    "longUrl": "string",
    "shortUrl": "string"
  },
  "success": true
}
```

Response Content Type

application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
requestDTO	<pre>{ "userId": 211, "longUrl": "http://www.sequoiacap.cn/china/people/introduction/", "shortUrl": "", "traceId": "8000211", "feature": "" }</pre>	requestDTO	body	RequestDTO

Parameter content type:

application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "userId": 211,
  "longUrl": "http://www.sequoiacap.cn/china/people/introduction/",
  "shortUrl": "",
  "traceId": "8000211",
  "feature": ""
}' http://localhost:8080/api/getShortUrl'
```

Request URL

http://localhost:8080/api/getShortUrl

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "success": true,
  "response": {
    "longUrl": "http://www.sequoiacap.cn/china/people/introduction/",
    "shortUrl": "Zveqyq",
    "feature": null
  },
  "errorMessage": null,
}
```

POST

/api/getLongUrl

根据短域名获取长域名信息

Response Class (Status 200)

OK

Model

Example Value

```
{
  "errorCode": "string",
  "errorMessage": "string",
  "response": {
    "feature": "string",
    "longUrl": "string",
    "shortUrl": "string"
  },
  "success": true
}
```

Response Content Type

application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
requestDTO	<pre>{ "userId": 211, "longUrl": "", "shortUrl": "Zveqyq", "traceId": "8000211", "feature": "" }</pre>	requestDTO	body	RequestDTO

Parameter content type:

application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "userId": 211,
  "longUrl": "",
  "shortUrl": "Zveqyq",
  "traceId": "8000211",
  "feature": ""
}' http://localhost:8080/api/getLongUrl'
```

Request URL

http://localhost:8080/api/getLongUrl

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
```

```
"success": true,
"response": {
  "longUrl": "http://www.sequoiacap.cn/china/people/introduction/",
  "shortUrl": "Zveqyq",
  "feature": null
},
```

异常返回结果

requestDTO

```
{
  "userId": 211,
  "longUrl": "http://www.sequoiacap.cn/china/people/introduction/",
  "shortUrl": "Zveqyq",
  "traceId": "10000211",
  "feature": ""
}
```

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! [Hide Response](#)

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "userId": 211,
  "longUrl": "http://www.sequoiacap.cn/china/people/introduction/",
  "shortUrl": "Zveqyq",
  "traceId": "10000211",
  "feature": ""
}' http://localhost:8080/api/getShortUrl
```

Request URL

http://localhost:8080/api/getShortUrl

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "timestamp": "2021-12-17T08:10:58.154+00:00",
  "status": 400,
  "error": "Bad Request",
  "message": "",
  "path": "/api/getShortUrl"
}
```

Response Code

400

Response Headers

```
{
  "connection": "close",
  "content-type": "application/json",
  "date": "Fri, 17 Dec 2021 08:10:58 GMT",
  "transfer-encoding": "chunked"
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
requestDTO	{ "userId": 211, "longUrl": "http://www.sequoiacap.cn/china/people/introduction/", "shortUrl": "Zveqyq", "traceId": "10000211", "feature": "" }	requestDTO	body	RequestDTO

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! [Hide Response](#)

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "userId": 211,
  "longUrl": "http://www.sequoiacap.cn/china/people/introduction/",
  "shortUrl": "Zveqyq",
  "traceId": "10000211",
  "feature": ""
}' http://localhost:8080/api/getLongUrl
```

Request URL

http://localhost:8080/api/getLongUrl

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "success": false,
  "response": null,
  "errorMessage": "2001",
  "errorCode": "域名信息不存在"
}
```

Response Code

200

Response Headers

```
{
  "connection": "keep-alive",
  "content-type": "application/json",
  "date": "Fri, 17 Dec 2021 08:11:54 GMT",
  "keep-alive": "timeout=60",
  "transfer-encoding": "chunked"
}
```

- 性能测试：详见单独报告。

扩展思考

由于数据是存储在内存的，并没有做持久化处理，如果考虑搭建一套完整的服务，我的系统搭建方案如下：

- 采用Nginx做负载均衡以及流量分发；
- 采用redis做缓存服务，承接高并发的流量；
- 采用Mysql做数据持久化
- 服务器都采用集群部署，redis和mysql主从同步机制保证高可用

