8장. URL 단축기 설계

Created	@October 27, 2022 11:29 AM
Progress	In Progress

📃 **학습 TODO list**

☐

> *tiny url* ***URL 단축기*** *설계하기*

8.1. 1단계: 문제 이해 및 설계 범위 확정

> *질문을 통해 모호함을 줄이고 요구사항을 알아내야 한다.*

👩‍💻 URL 단축기가 어떻게 동작해야 하는지 예제를 보여주실 수 있나요?

👨‍💻 https://www.systeminterview.com/1=chatsystem&c=loggedin&v=v3&l=long 입력으로 주어졌다고 해 봅시다. 이 서비스는 https://tinyurl.com/y7ke-ocwj와 같은 **단축 URL**로 제공해야 합니다. **이 URL에 접속하면 원래 URL로 갈 수도 있어야 하죠.**

👩‍💻 트래픽 규모는 어느 정도일까요?

👩‍💻 **매일 1억(100 million) 개의 단축 URL을 만들어 낼 수 있어야 합니다.**

👩‍💻 단축 URL의 길이는 어느 정도여야 하나요?

👩‍💻 짧으면 짧을수록 좋습니다.

👩‍💻 단축 URL에 포함될 문자에 제한이 있습니까?

👩‍💻 단축 URL에는 숫자(0부터 9까지)와 영문자(a부터 z, A부터 Z까지)만 사용할 수 있습니다.

👩‍💻 단축된 URL을 시스템에서 지우거나 갱신할 수 있습니까?

👩‍💻 시스템을 단순화하기 위해 삭제나 갱신은 할 수 없다고 가정합시다.

질의 응답을 통해 파악한 요구사항

- `URL 단축`: 주어진 긴 URL을 훨씬 짧게 줄인다.
- `URL 리디렉션(redirection)`: 축약된 URL로 HTTP 요청이 오면 원래 URL로 안내
- `높은 가용성`과 `규모 확장성`, 그리고 `장애 감내`가 요구됨

개략적 추정

- 쓰기 연산: 매일 1억개의 단축 URL 생성
  - 초당 쓰기 연산: 1억(100 million)/24/3600=1160
  - 초당 읽기 연산: 읽기 연산과 쓰기 연산 비율은 10:1이라고 하자. 그 경우 읽기 연산은 초당 11,600회 발생한다.(1160×10=11,600)
- URL 단축 서비스를 10년간 운영한다고 가정하면 1억(100 million)×365×10=3650억(365 billion) 개의 레코드를 보관해야 한다.
  - 축약 전 URL의 평균 길이는 100이라고 하자.
  - 따라서 10년 동안 필요한 저장 용량은 3650억(365 billion)×100바이트=36.5TB이다.

> *계산이 끝나면 결과를 면접관과 점검하여 합의한 후 진행한다.*

8.2. 2단계: 개략적 설계안 제시 및 동의 구하기

8.2.1. API 엔드포인트(endpoint)

- 클라이언트는 서버가 제공하는 `API 엔드포인트`를 통해 서버와 통신한다.

> *`엔드포인트`를 `RESTful API`로 설계해보자.*

1. `URL 단축`용 `엔드포인트` : 새 단축 URL을 생성하고자 하는 클라이언트는 이 `엔드포인트`에 단축할 URL을 인자로 실어서 `POST` 요청을 보내야 한다.

   `POST /api/v1/data/shorten`

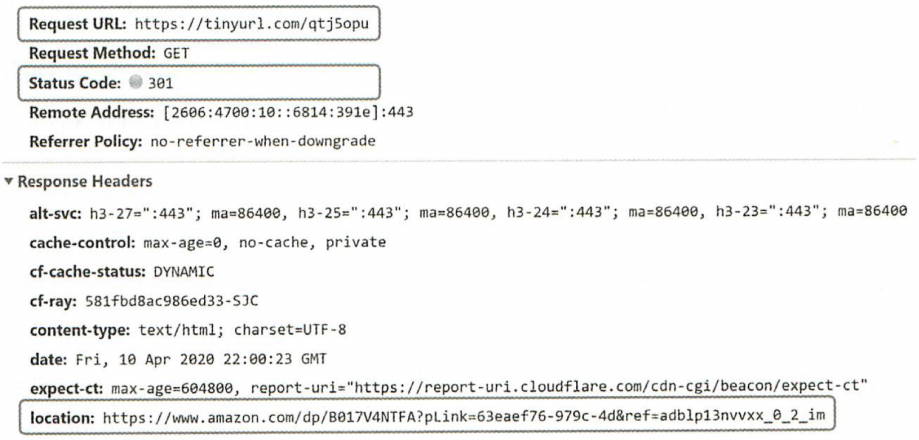
   - 인자: {longUrl: longURLstring}
   - 반환: 단축 URL

2. `URL 리디렉션`용 `엔드포인트`: 단축 URL에 대해서 `HTTP` 요청이 오면 원래 URL로 보내주기 위한 용도의 `엔드포인트`

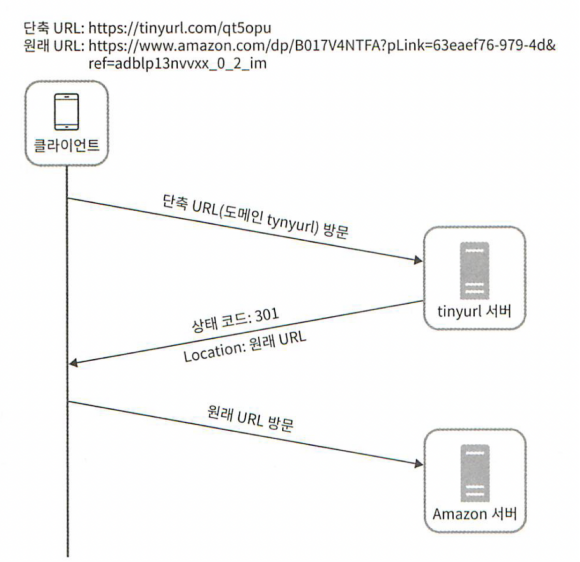
   `GET /api/v1/shortUrl`

   - 반환: HTTP 리디렉션 목적지가 될 원래 URL

8.2.2. URL 리디렉션



브라우저에 단축 URL을 입력할 경우 Response



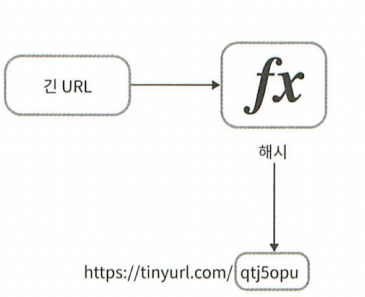
클라이언트와 서버 사이의 통신 절차

- 단축 URL을 받은 서버는 그 URL을 원래 URL로 바꾸어서 `301` 응답의 **`Location 헤더`**에 넣어 반환한다.
  - `301 Permanently Moved`: 해당 URL에 대한 **`HTTP`** 요청의 처리 책임이 **`영구적`**으로 **`Location 헤더`**에 반환된 URL로 이전되었다는 응답이다. 영구적으로 이전되었으므로, 브라우저는 이 응답을 **`캐시(cache)`**한다. → 추후 같은 단축 URL에 요청을 보낼 필요가 있을 때 브라우저는 캐시된 원래 URL로 요청을 보낸다.
    - **`서버 부하`**를 줄이는 것이 중요한 경우 이점을 갖는다.(첫 번째 요청만 단축 URL 서버로 전송된다.)
  - `302 Found`: 주어진 URL로의 요청이 **`일시적`**으로 **`Location 헤더`**가 지정하는 URL에 의해 처리되어야 한다는 응답이다. → 클라이언트의 요청은 언제나 단축 URL 서버에 먼저 보내진 후 원래 URL로 리디렉션 된다.
    - **`트래픽 분석(analytics)`**이 중요한 경우 이전을 갖는다.(클릭 발생률이나 발생 위치를 추적할 수 있다.)
- **`URL 리디렉션`**을 구현하는 방법: **`해시 테이블`**

  `<단축 URL, 원래 URL>`
  - `원래 URL=hashTable.get(단축 URL)`
  - `301` 또는 `302` 응답 **`Location 헤더`**에 원래 URL을 넣은 후 전송

8.2.3. URL 단축 플로

> *긴 URL을 해시 값으로 대응시킬 해시 함수 fx를 찾아야 한다.*



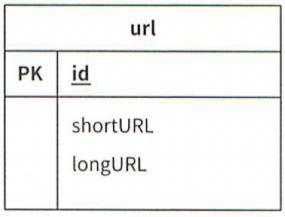
단축 URL: www.tinyurl.com/{hashValue}

> *해시 함수* *요구사항*

- 입력으로 주어지는 긴 URL이 다른 값이면 `해시 값`도 달라야 한다.
- 계산된 해시 값은 원래 입력으로 주어졌던 긴 URL로 복원될 수 있어야 한다.

8.3. 3단계: 상세 설계

8.3.1. 데이터 모델



테이블의 간단한 설계 사례

- 메모리는 유한하고 비싸므로 모든 것을 해시 테이블에 둘 수 없다.
- 그러므로 `<단축 URL, 원래 URL>`의 순서쌍을 관계형 데이터베이스에 저장한다.

8.3.2. 해시 함수

- `해시 함수(hash function)`은 원래 URL을 단축 URL(`hashValue`)로 변환하는 데 쓰인다.

> *해시 값 길이*

- hashValue는 `[0-9, a-z, A-Z]`의 문자들로 구성된다. → 사용할 수 있는 문자의 개수: 10+26+26=62
  - 개략적으로 계산했던 추정치에 따르면 이 시스템은 3650억 개의 URL을 만들어 낼 수 있어야 한다. → $62^n \geq$3650억(265 billion)인 `n`(`hashValue` 길이)의 최솟값을 찾아야 하므로 n=3이다.(62^3=238,328)

> *해시 함수 구현 방법: 해시 후 충돌 해소, base-62 변환*

해시 후 충돌 해소 전략	base-62 변환
단축 URL의 길이가 고정됨	단축 URL의 길이가 가변적. ID 값이 커지면 같이 길어짐
유일성이 보장되는 ID 생성기가 필요치 않음	유일성 보장 ID 생성기가 필요
충돌이 가능해서 해소 전략이 필요	ID의 유일성이 보장된 후에야 적용 가능한 전략이라 충돌이 아예 불가능
ID로부터 단축 URL을 계산하는 방식이 아니라서 다음에 쓸 수 있는 URL을 알아내는 것이 불가능	ID가 1씩 증가하는 값이라고 가정하면 다음에 쓸 수 있는 단축 URL이 무엇인지 쉽게 알아낼 수 있어서 보안상 문제가 될 소지가 있음

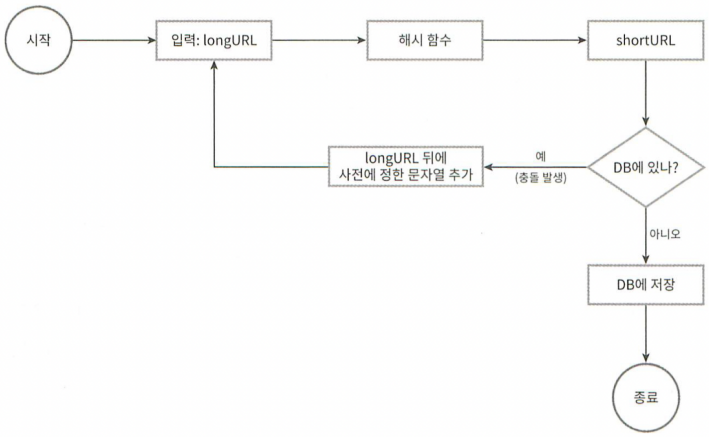
1. 해시 후 충돌 해소

- 잘 알려진 `해시 함수`: `CRC32`, `MD5`, `SHA-1`

해시 함수	해시 결과 (16진수)
CRC32	5cb54054
MD5	5a62509a84df9ee03fe1230b9df8b84e
SHA-1	0eeae7916c06853901d9ccbefbfcaf4de57ed85b

해시 함수로 https://en.wikipedia.org/wiki/Systems_design을 축약한 결과

- 계산된 해시 값을 7글자 문자열로 줄이는 방법: 계산된 `해시 값`에서 처음 7개 글자만 이용한다.
  - 해시 결과가 서로 충돌할 확률이 높아진다.
  - 충돌이 실제로 발생하면 충돌이 해소될 때까지 사전에 정한 문자열을 `해시 값`에 덧붙인다.



해시 후 충돌 해소 절차

  - 단축 URL을 생성할 때 한 번 이상 데이터베이스 query를 날려야 한다. → `오버헤드`가 크다.
  - 데이터베이스 대신 `블룸 필터`를 사용하면 성능을 높일 수 있다.
    - `블룸 필터`: 어떤 집합에 특정 원소가 있는지 검사할 수 있도록 하는 확률론에 기초한 공간 효율이 좋은 기술이다.

2. base-62 변환

- `진법 변환(base conversion)`은 `URL 단축기`를 구현할 때 흔히 사용되는 접근법이다.
  - 표현 방식이 다른 두 시스템이 같은 수를 공유하여야 하는 경우에 유용하다.
- `62진법`을 사용하는 이유: `hashValue`에서 사용할 수 있는 문자(character) 개수가 62개이다.(10-35: a-z, 36-61: A-Z)

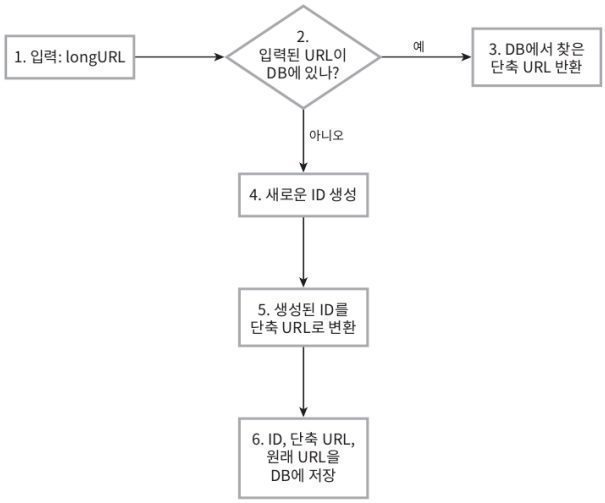
		나머지	62진수 표현
62	11157	59	X
62	179	55	T
62	2	2	2
	0		

$$11157_{10} = 2 \times 62^2 + 55 \times 62^1 + 59 \times 62^0 = 2TX_{62}$$

8.3.3. URL 단축기 상세 설계

> *시스템의 핵심 컴포넌트이므로, 처리 흐름이 논리적으로 단순해야 하고 기능적으로 언제나 동작하는 상태로 유지되어야 한다.*

> *`62진법 변환` 기법을 사용해 설계해보자.*



처리 흐름 순서도

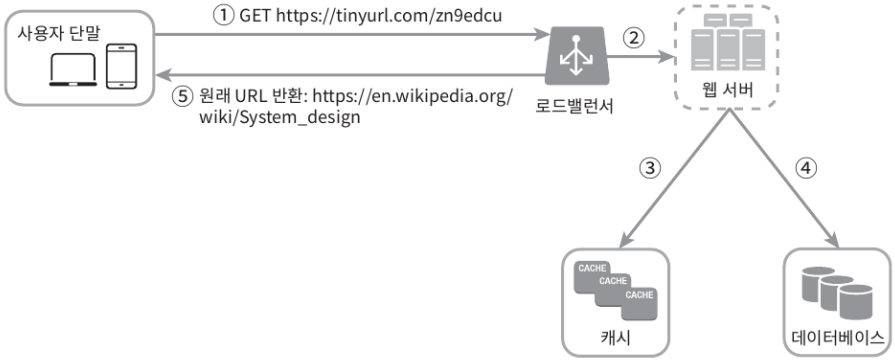
1. 입력으로 긴 URL을 받는다.
   - ex. 입력된 URL이 https://en.wikipedia.org/wiki/Systems_design이라고 하자.
2. 데이터베이스에 해당 URL이 있는지 검사한다.
3. 데이터베이스에 있다면 해당 URL에 대한 단축 URL을 만든 적이 있는 것이다. 따라서 데이터베이스에서 해당 단축 URL을 가져와서 클라이언트에게 반환한다.

4. 데이터베이스에 없는 경우에는 해당 URL은 새로 접수된 것이므로 유일한 ID를 생성한다. 이 ID는 데이터베이스의 기본 키로 사용된다.
    - ex. 이 URL에 대해 `ID 생성기`가 반환한 ID는 `2009215674938`이다.
5. `62진법 변환`을 적용하여 ID를 단축 URL로 만든다.
    - ex. 62진수로 변환하면 `zn9edcu`를 얻는다.
6. ID, 단축 URL, 원래 URL로 새 데이터베이스 레코드를 만든 후 단축 URL을 클라이언트에 전달한다.
    - ex. 생성된 데이터베이스 레코드 `<ID, shortURL, longURL>`: `<2009215674938, zn9edcu, https://en.wikipedia.org/wiki/Systems_design>`

> *ID 생성기는 단축 URL을 만들 때 사용할 ID를 만드는 것이고, **`전역적 유일성 (globally unique)`**이 보장되는 ID여야 한다.*

8.3.4. URL 리디렉션 상세 설계

- 쓰기보다 읽기를 더 자주 하는 시스템이라, `<단축 URL, 원래 URL>`의 쌍을 `캐시`에 저장하여 성능을 높인다.



URL 리디렉션(redirection) 메커니즘의 상세한 설계(로드밸런서의 동작 흐름)

1. 사용자가 단축 URL을 클릭한다.
2. `로드밸런서`가 해당 클릭으로 발생한 요청을 `웹 서버`에 전달한다.
3. 단축 URL이 이미 `캐시`에 있는 경우에는 원래 URL을 바로 꺼내서 클라이언트에게 전달한다.
4. `캐시`에 해당 단축 URL이 없는 경우에는 데이터베이스에서 꺼낸다. 데이터베이스에 없다면 아마 사용자가 잘못된 단축 URL을 입력한 경우일 것이다.
5. 데이터베이스에서 꺼낸 URL을 `캐시`에 넣은 후 사용자에게 반환한다.

8.4. 4단계: 마무리

👩‍💻 추가 논의 사항

- `처리율 제한 장치(rate limiter)`: 앞서 살펴본 시스템은 엄청난 양의 URL 단축 요청이 밀려들 경우 무력화될 수 있다.(잠재적 보안 결함)
  - `처리율 제한 장치`로 IP 주소를 비롯한 `필터링 규칙(filtering rule)`들을 이용해 요청을 걸러낼 수 있다.
- `웹 서버의 규모 확장`: 앞서 설계한 웹 계층은 `무상태(stateless) 계층`이므로, 웹 서버를 자유롭게 증설하거나 삭제할 수 있다.
- `데이터베이스의 규모 확장`: 데이터베이스를 다중화하거나 `샤딩(sharding)`하여 규모 확장성을 달성할 수 있다.
- `데이터 분석 솔루션(analytics)`: `URL 단축기`에 데이터분석 솔루션을 통합해 두면 어떤 링크를 얼마나 많은 사용자가 클릭했는지, 언제 주로 클릭했는지 등 중요한 정보를 알아낼 수 있다.
- `가용성`, `데이터 일관성`, `안정성`: 대규모 시스템이 성공적으로 운영되기 위해서는 반드시 갖추어야 할 속성들이다.