URL

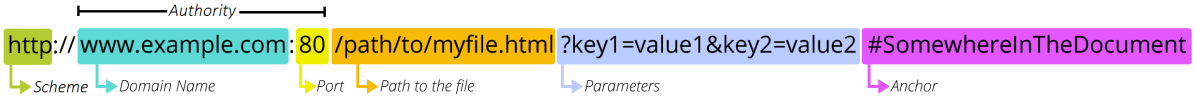
강의자	배한조
일자	@2023년 10월 28일

URL?

Uniform Resource Locator의 약자로 웹에서 주어진 고유 리소스 주소를 의미한다.

일반적으로 각각 유효한 URL은 고유한 리소스를 가리킨다.

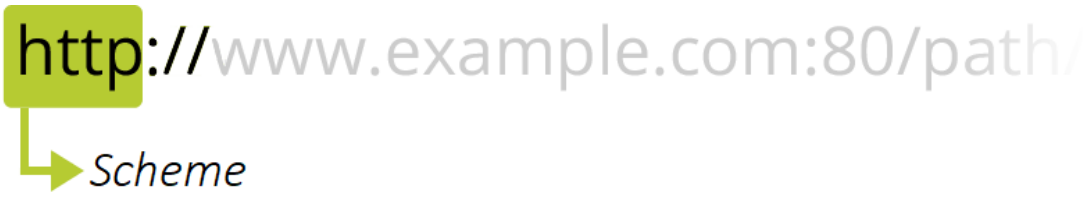
URL 구조



URL의 구조

전체적인 구조는 위와 같으며 일부는 필수이고 일부는 선택 사항이다.

Scheme



URL의 첫 번째 부분은 브라우저가 리소스를 요청하는데 사용해야하는 프로토콜을 나타낸다.

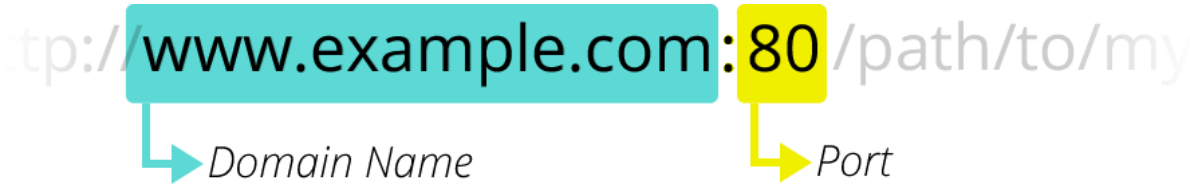
보통 웹사이트는 HTTPS 또는 HTTP를 사용한다. 이외에 다른 프로토콜 혹은 스키마는 아래와 같다.

스키마	설명
data	Data URL
file	호스트별 파일 이름
ftp	File Transfer Protocol
http/https	하이퍼텍스트 전송 프로토콜
javascript	URL 내 JavaScript 코드

스키마	설명
mailto	메일 주소
ssh	보안 쉘
tel	전화
urn	통합 자원 이름
view-source	리소스의 소스코드
ws/wss	웹 소켓 연결

자세한 내용은 여기를 참고

Domain 이름과 Port



Domain

- 요청하는 웹 서버를 나타낸다.
- 도메인은 일반적으로 도메인 이름을 사용하지만 IP 주소도 사용될 수 있다.

Port

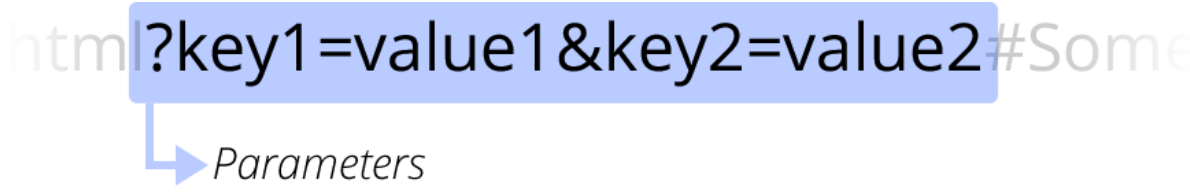
- 웹 서버의 리소스에 접근하는 데 사용되는 기술적인 게이트를 나타낸다.
- 웹 서버가 HTTP 프로토콜의 표준 포트(HTTP = 80, HTTPS = 443)를 사용하는 경우엔 일반적으로 생략한다. 그렇지 않다면 필수다.

Resource Path

웹 서버에 있는 리소스의 경로를 의미한다.

최근에는 대부분 물리적 실체가 없는 웹 서버가 추상적으로 처리한다.

Query String



웹 서버에 제공되는 추가 매개변수이다.

`?`로 시작되며 매개변수가 여러 개라면 `&` 기호로 구분되는 키/값 쌍 목록이다.

Anchor

리소스 내부에서 책갈피와 같은 역할을 하며 브라우저에 해당 책갈피 지점의 콘텐츠를 표시하도록 지시한다. `#` 뒤의 부분은 서버로 전송되지 않는다.

HTML Anchor 태그로 이를 만들 수 있다.

URL 인코딩 (Percent Encoding)

문자나 특수문자를 웹 서버나 브라우저에서 보편적으로 허용되는 형식으로 변화하는 메커니즘

URL에서 사용할 수 없는 문자나 URL로 사용할 수는 있지만 의미가 왜곡될 수 있는 문자들을 `%xx` 형태로 변환한다. (이를 안전하지 않은 문자로 취급)

```
// URL 인코딩 예시
(공백) => %20 혹은 +
| (파이프문자) => %7C
, (쉼표) => %2C
```

또한, URL에 ascii code가 아닌 문자의 경우 유니코드로 인코딩 후 `%+16진수`로 변환한다.

```
// 한글
하이

// 변환
'하' = U+D55C = ED 95 98
'이' = U+C774 = EC 9D B4

// 결과
%ed%95%98%ec%9d%b4
```

URL 인코딩은 이러한 안전하지 않은 문자를 `%다음 두 개의 16진수`로 대체한다.

문자	인코딩
':'	%3A
'/'	%2F
'?'	%3F
'#'	%23
'['	%5B
']'	%5D
'@'	%40
'!'	%21
'$'	%24
'&'	%26
"'"	%27
'('	%28
')'	%29
'*'	%2A
'+'	%2B
','	%2C
';'	%3B
'='	%3D
'%'	%25
' '	%20 or +

안전하지 않은 문자 인코딩 표

URL 인코딩 하는 이유

- 특수 문자와 공백 처리
  - 특수 문자나 공백을 URL에 포함하 수 있지만 URL 구조와 혼동을 줄 수 있다.
  - 대표적으로 &, ? 등이 그 예
  - 이러한 문자를 인코딩하여 URL의 일부로 해석되지 않고 안전하게 전달 가능
- ASCII 이외의 문자 처리
  - URL은 기본적으로 ASCII 코드를 사용한다. 그러나 한글과 같이 ASCII 코드가 없는 문자나 기호는 URL 인코딩을 사용해 처리할 수 있다.
- 일관성, 호환성

Semantic URL

시맨틱 URL은 클린 URL과 같은 의미를 가진다.

질의어 없이 경로만 가진 간단한 구조의 URL을 지칭하며 사용자 친화적 URL, 검색엔진 친화적 URL이라고도 불린다.

사용자가 쉽게 기억할 수 있으며 누구나 브러우저 주소창에 입력할 수 있다.

시맨틱 URL의 이점은 아래와 같다.

- URL을 쉽게 조작할 수 있다.
- 사용자가 어디에 있는지, 무엇을 하고 있는지, 웹에서 무엇을 읽거나 상호 작용하는지에 대해 사용자가 명확하게 설명한다.
- 일부 검색 엔진은 URL을 분석해 웹 페이지 내용을 평가하므로, SEO(검색 엔진 최적화)가 가능하다.

예시

원본 URL	시맨틱 URL
http://example.com/about.html	http://example.com/about
http://example.com/user.php?id=1	http://example.com/user/1
http://example.com/index.php?page=name	http://example.com/name
http://example.com/kb/index.php?cat=1&id=23	http://example.com/kb/1/23
http://en.wikipedia.org/w/index.php?title=Clean_URL	http://en.wikipedia.org/wiki/Clean_URL

EncodeURL과 EncodeURLComponent 차이

```
// 원본
'http://username:password@www.example.com:80/path/to/file.php?foo=316&bar=this+has+spaces#anchor'

// encodeURI
encodeURI('http://username:password@www.example.com:80/path/to/file.php?foo=316&bar=this+has+spaces#anchor')
'http://username:password@www.example.com:80/path/to/file.php?foo=316&bar=this+has+spaces#anchor'

// encodeURIComponent
encodeURIComponent('http://username:password@www.example.com:80/path/to/file.php?foo=316&bar=this+has+spaces#anchor')
'http%3A%2F%2Fusername%3Apassword%40www.example.com%3A80%2Fpath%2Fto%2Ffile.php%3Ffoo%3D316%26bar%3Dthis%2Bhas%2Bspaces%23anchor'
```

```
var set1 = ";,/?:@&=+$#"; // 예약 문자
var set2 = "-_.!~*'()"; // 비예약 표식
var set3 = "ABC abc 123"; // 알파벳 및 숫자, 공백

console.log(encodeURI(set1)); // ;,/?:@&=+$#
console.log(encodeURI(set2)); // -_.!~*'()
console.log(encodeURI(set3)); // ABC%20abc%20123 (공백은 %20으로 인코딩)

console.log(encodeURIComponent(set1)); // %3B%2C%2F%3F%3A%40%26%3D%2B%24%23
console.log(encodeURIComponent(set2)); // -_.!~*'()
console.log(encodeURIComponent(set3)); // ABC%20abc%20123 (공백은 %20으로 인코딩)
```

API	예약문자	비예약표식	알파벳 및 숫자, 공백
encodeURI	X	X	O
encodeURIComponent	O	X	O

URI `전체`를 인코딩 ⇒ `encodeURI`
URI `파라미터` 인코딩 ⇒ `encodeURIComponent`

이렇게 기억하자.

URLSearchParams

URL의 Query 문자열을 조작할 수 있도록 도와주는 JavaScript 네이티브 유틸리티 메서드다.

많이 쓰는 메서드

메서드명	설명
~.append()	주어진 키/값 쌍을 새로운 검색 매개변수로 추가

~.delete()	주어진 검색 매개변수와 그 값을 모두 삭제
~.entries()	객체의 모든 키/값 쌍을 쿼리 문자열과 같은 순서로 순회할 수 있는 순회기를 반환
~.get()	주어진 검색 매개변수에 연결된 첫 번째 값을 반환
~.has()	주어진 검색 매개변수에 연결된 모든 값을 반환
~.sort()	모든 키/값 쌍을 키의 순서로 정렬

더 자세한 내용으로는 MDN을 참고