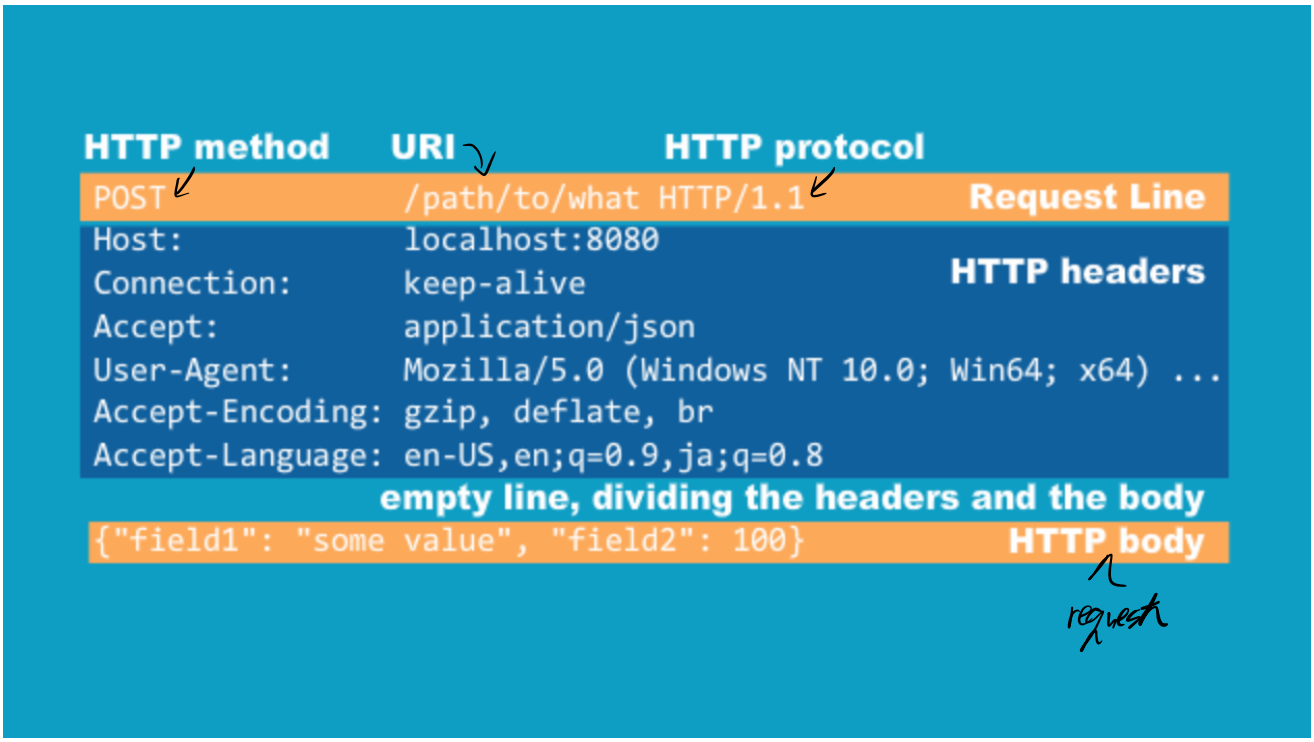

Request Line

1. 전송 메소드

위 그림의 Request Line에서 가장 좌측 'POST'가 여기에 해당합니다. GET/POST 등 어떤 전송 방식으로 구성되어 있는지를 나타냅니다.

2.요청 내용에 대한 경로

위 그림의 Request Line에서 전송 메소드 오른쪽에 있는 내용입니다. '/path/to/what' 이렇게 요청 내용이 서버상의 어느 위치에 있는지를 나타냅니다.

3.요청 형식에 대한 버전 정보

위 그림의 Request Line에서 '요청 내용에 대한 경로' 오른쪽에 있는 내용입니다. HTTP는 통신을 위한 약속을 정의한 규약입니다. 이 내용은 어떤 버전의 HTTP 문법을 준수하고 있는가를 나타냅니다. 예를 들어 1.1, 2.0 , 3.0이 있겠네요.

Header

HTTP 헤더는 요청 라인 아래에 작성된 내용부터 한 줄 공백이 나오기 전까지 모두를 포함합니다. HTTP 헤더는 클라이언트와 서버가 요청 또는 응답으로 **부가적인 정보**를 전송할 수 있도록 합니다. 헤더는 모두 키 : 값으로 이루어집니다. 위 그림의 각각의 내용은 다음을 의미합니다.

#host:서버의 도메인명과 서버가 리스닝중인 TCP 포트 번호를 명시합니다.

#Connection:현재 트랜잭션이 끝난후에 네트워크 연결을 열린 상태로 둘지 여부를 제어합니다.

#Accept: Return할 데이터 타입에 대해 서버에 알립니다.

#User-Agent:네트워크 프로토콜 피어가 요청하는 사용자 에이전트의 애플리케이션 타입, 운영 체제, 소프트웨어 벤더 또는 소프트웨어 버전을 식별할 수 있는 문자열을 포함합니다.

#Accept-Encoding:인코딩 알고리즘에 대해 서버에 알립니다.

#Accept-Langue:서버가 Return하기로 예상된 언어에 대해 서버에 알립니다.

Body

한 줄 공백 아래에 작성된 메시지 본문을 HTTP Body 라고 합니다.

추가내용

지금까지 살펴본 HTTP 요청 메시지는 POST 방식일 경우입니다. 만약 전송 방식이 GET이라면 메시지 구조는 달라집니다.

```
GET /userAccount/login?
account=swift@swift.com&amp;amp;passwd=1234&amp;amp;grant_type=password HTTP/1.1
Host: swiftapi.rubypaper.co.kr:2029
cache-Control: no-cache
```

GET 메소드를 사용할 경우 위 내용과 같이 메시지 본문(Body)이 사라지고, 본문에 있어야 할 값이 첫 번째 라인의 경로 뒤에 '?' 문자열과 함께 연결되었습니다. GET 방식에서는 파라미터를 모두 URL 뒤에 연결해서 전달하기 때문입니다. 이렇게 연결된 파라미터를 쿼리 스트링이라고 합니다.

비교적 간결하게 정보를 전달할 수 있다는 장점이 있지만, **URL 경로는 1024 Byte까지만 허용**되기 때문에, 긴 값을 GET 방식으로 전송할 수는 없습니다. 때문에 데이터를 전송하는 경우보다 **필요한 정보를 요청할 때 주로 GET 방식이 사용**됩니다.