HTTP cookie

HTTP Cookie（也叫 Web Cookie 或浏览器 Cookie）是服务器发送到用户浏览器并保存在本地的一小块数据。服务器指定 Cookie 后，浏览器的每次请求都会携带 Cookie 数据，会带来额外的性能开销。

1. 创建 Cookie

服务器收到 HTTP 请求后，服务器可以在响应标头里面添加一个或多个 `Set-Cookie` 选项。浏览器收到响应后通常会保存下 Cookie，并将其放在 HTTP Cookie 标头内，向同一服务器发出请求时一起发送。

1.1 Set-Cookie 和 Cookie 标头

```
Set-Cookie: <cookie-name>=<cookie-value>
```

这指示服务器发送标头告知客户端存储一对 cookie：

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry
```

现在，对该服务器发起的每一次新请求，浏览器都会将之前保存的 Cookie 信息通过 Cookie 请求头部再发送给服务器。

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

在 Nodejs 中设置 Cookie 非常简单，只需要在响应对象上调用 `res.setHeader()` 方法即可。

```
response.setHeader('Set-Cookie', 'yummy_cookie=choco');
response.setHeader('Set-Cookie', 'tasty_cookie=strawberry');
```

1.2. 定义 Cookie 的生命周期

Cookie 的生命周期可以通过两种方式定义：

- 会话期 Cookie 会在当前的会话结束之后删除。浏览器定义了“当前会话”结束的时间，一些浏览器重启时会使用会话恢复。这可能导致会话 cookie 无限延长。
- 持久性 Cookie 在过期时间（Expires）指定的日期或有效期（Max-Age）指定的一段时间后被删除。

```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
```

当 Cookie 的过期时间（ Expires）被设定时，设定的日期和时间只与客户端相关，而不是服务端。

1.3. 限制访问 Cookie

有两种方法可以确保 Cookie 被安全发送，并且不会被意外的参与者或脚本访问：Secure 属性和 HttpOnly 属性。

标记为 Secure 的 Cookie 只应通过被 HTTPS 协议加密过的请求发送给服务端。

JavaScript Document.cookie API 无法访问带有 HttpOnly 属性的 cookie；此类 Cookie 仅作用于服务器。

```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT; Secure; HttpOnly
```

1.4. 定义 Cookie 发送的位置

Domain 和 Path 标识定义了 Cookie 的作用域：即允许 Cookie 应该发送给哪些 URL。

（1）Domain

`Domain` 指定了哪些主机可以接受 Cookie。如果不指定，该属性默认为同一 host 设置 cookie，不包含子域名。如果指定了 Domain，则一般包含子域名。

例如，如果设置 `Domain=mozilla.org`，则 Cookie 也包含在子域名中（如 `developer.mozilla.org`）。

（2）Path 属性

`Path` 属性指定了一个 URL 路径，该 URL 路径必须存在于请求的 URL 中，以便发送 Cookie 标头。

例如，设置 Path=/docs，则以下地址都会匹配：

- `/docs`
- `/docs/`
- `/docs/Web`
- `/docs/Web/HTTP`

（3）sameSite 属性

SameSite 属性允许服务器指定是否/何时通过跨站点请求发送。这提供了一些针对跨站点请求伪造攻击（cross site request forgery,CSRF）的保护。它采用三个可能的值：Strict、Lax 和 None。

使用 `Strict`，cookie 仅发送到它来源的站点。`Lax` 与 `Strict` 相似，只是在用户导航到 cookie 的源站点时发送 cookie。例如，通过跟踪来自外部站点的链接。`None` 指定浏览器会在同站请求和跨站请求下继续发送 cookie，但仅在安全的上下文中（即，如果 SameSite=None，且还必须设置 Secure 属性）。如果没有设置 SameSite 属性，则将 cookie 视为 `Lax`。

1.5. JavaScript 通过 Document.cookie 访问 Cookie

通过 `Document.cookie` 属性可创建新的 Cookie。如果未设置 `HttpOnly` 标记，可以从 JavaScript 访问现有的 Cookie。

```
document.cookie = "yummy_cookie=choco";
document.cookie = "tasty_cookie=strawberry";
// "yummy_cookie=choco; tasty_cookie=strawberry"

// 一次只能设置一组 cookie，不能设置多组
// 比如：document.cookie = "a=1; b=2"; 返回 "a=1; b=2"，但通过 document.cookie 获取时，只能获取到 "a=1"。
console.log(document.cookie);
```

通过 JavaScript 创建的 Cookie 不能包含 HttpOnly 标志。

2. 安全

缓解涉及 Cookie 的攻击的方法：

- 使用 HttpOnly 属性可防止通过 JavaScript 访问 cookie 值。
- 用于敏感信息（例如指示身份验证）的 Cookie 的生存期应较短，并且 SameSite 属性设置为 Strict 或 Lax。