TD1 SOC

▼ **Question 1** : Pourquoi utiliser le port 8080 pour tester notre serveur alors que le port standard pour un serveur WEB est le port 80?

Il y a deux choses à considérer ici:

1. Vos utilisateurs se souviendront-ils d'utiliser un port non standard dans le nom? Par défaut, le port 80 est le standard et vous n'avez donc pas à le saisir dans l'URL. Par exemple, `http://superuser.com` s'exécute sur le port 80 et votre navigateur suppose que 80 est le port que vous entendez lorsque vous le saisissez. Ce n'est pas différent de la saisie `http://superuser.com:80`. Si vous exécutez votre websever sur le port 8080, l'utilisateur *doit* taper `http://superuser.com:8080`. L'utilisateur moyen ne s'en souviendra probablement pas.
2. L'exécution d'un serveur Web sur un port non standard vous protège-t-elle des attaques DoS? Pas vraiment. Si quelqu'un voulait *vraiment* mettre votre site hors service, l'exécution sur un port non standard ne l'arrêtera pas. Les attaquants analysent tous les ports de votre IP et constatent rapidement que 8080 (ou tout ce que vous choisissez) est ouvert et répond aux requêtes HTTP.

Des méthodes telles que le changement de ports sont appelées « Sécurité par l'obscurité » et il est très douteux que le travail supplémentaire et les inconvénients apportent une sécurité précieuse.

Tester avant de commencer

```
telnet www.tigli.fr
GET /
```

▼ **Question 2** : Quel est le résultat obtenu ?

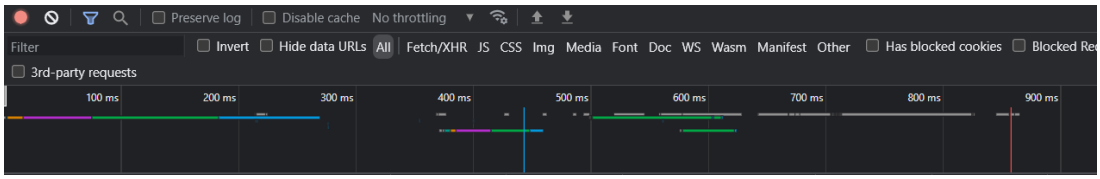
```
    <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
                                          <html><head>
                                                    <title>302 Found</title>
                                                              </head><body>
                                                                        <h1>Found</h1>

<p>The document has moved <a href="http://sparks.i3s.unice.fr/">here</a>.</p>
                                                    <hr>
                                                        <address>Apache/2.4.41 (Ubuntu) Server at ere
be-vm5.i3s.unice.fr Port 80</address>
                              </body></html>

Perte de la connexion à l'hôte.
```

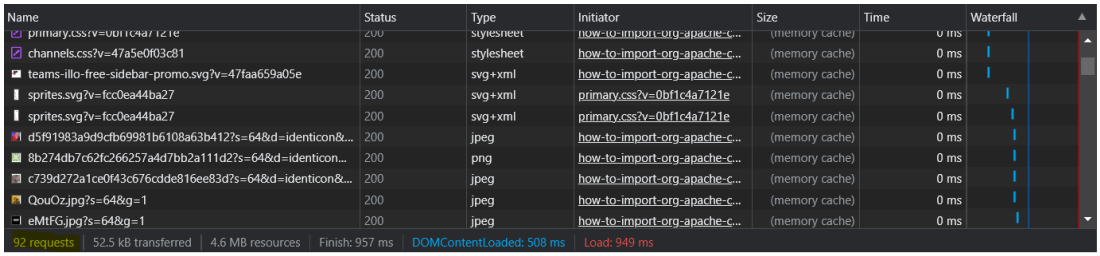
Le serveur envoie directement le contenu de la réponse, sans métadonnées.

▼ **Question 3** : Combien de requêtes HTTP sont déclenchées pour récupérer l'intégralité de la page ?

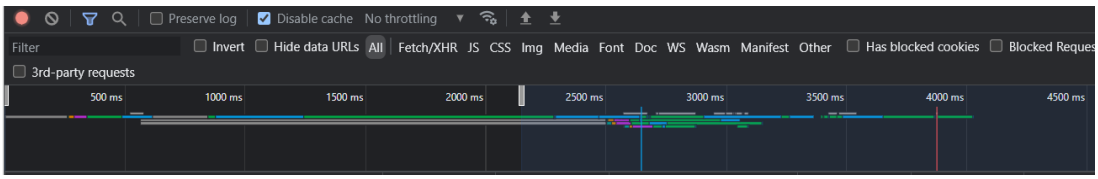
▼ Outils de développement > Network > StackOverflow

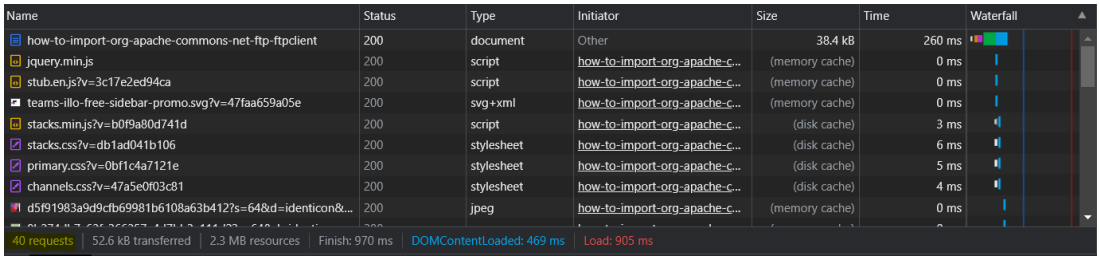
- Avec cache → Les informations de connexion sont connues par le cache ce qui évite une connexion au serveur pour faire la recherche.





- Sans cache → La réponse prend 4 fois plus de temps sans le cache à cause de la recherche sur le serveur.





Name	Status	Type	Initiator	Size	Time	Waterfall
how-to-import-org-apache-commons-net-ftp-ftpclient	200	document	Other	38.4 kB	260 ms	
jquery.min.js	200	script	how-to-import-org-apache-c...	(memory cache)	0 ms	
stub.en.js?v=3c17e2ed94ca	200	script	how-to-import-org-apache-c...	(memory cache)	0 ms	
teams-illo-free-sidebar-promo.svg?v=47faa659a05e	200	svg+xml	how-to-import-org-apache-c...	(memory cache)	0 ms	
stacks.min.js?v=b0f9a80d741d	200	script	how-to-import-org-apache-c...	(disk cache)	3 ms	
stacks.css?v=db1ad041b106	200	stylesheet	how-to-import-org-apache-c...	(disk cache)	6 ms	
primary.css?v=0bf1c4a7121e	200	stylesheet	how-to-import-org-apache-c...	(disk cache)	5 ms	
channels.css?v=47a5e0f03c81	200	stylesheet	how-to-import-org-apache-c...	(disk cache)	4 ms	
d5f91983a9d9cfb69981b6108a63b412?s=64&d=identicon&...	200	jpeg	how-to-import-org-apache-c...	(memory cache)	0 ms	

40 requests | 52.6 kB transferred | 2.3 MB resources | Finish: 970 ms | DOMContentLoaded: 469 ms | Load: 905 ms

⛔ Toutes les requêtes HTTP effectuées par le navigateur sont d'abord acheminées vers le cache du navigateur pour vérifier s'il existe une réponse valide dans le cache qui peut être utilisée pour répondre à la requête. S'il y a une correspondance, la réponse est lue dans le cache, ce qui élimine à la fois la latence du réseau et les coûts de données que le transfert engendre.

▼ **Question 4** : Quel est le % de gain de temps ? De bande passante ?

Avec les données trouvées, on peut voir un gain de temps de 1.44% environ quand on a un cache par rapport à quand nous n'en avons pas ce qui peut paraître assez négligeable.

Pour ce qui est de la bande passante, on passe de 150kb/s avec cache à 147.7kb/s environ sans, soit une baisse de 1.54% quand on n'a plus de cache. Cette plus grande bande passante trouvée quand nous avons un cache peut être liée au fait que plus de requêtes HTTP sont émises que sans.

Mise en œuvre d'un serveur HTTP classique

```
<title>L'exemple HTML le plus simple</title>
<h1>Ceci est un sous-titre de niveau 1</h1>
Bienvenue dans le monde HTML. Ceci est un paragraphe.
<p>Et ceci en est un second.</p>
<a href="index.html">cliquez ici</a> pour réafficher.
```

Mise en œuvre d’un serveur HTTP avec la classe HttpListener

▼ **Question 5** : Affichez (dans la console du serveur par exemple), les champs d'en-tête qui vous semblent les plus intéressants / pertinents. L'important n'est pas (pour l'instant) la pertinence de vos choix, mais de montrer que vous pouvez récupérer les champs que vous souhaitez en cas de besoin.

```
private static async void ResponseInfo(Stream output, int code, Stream content)
    {
        await using var sw = new StreamWriter(output);
        // Code de retour de la requête HTTP
        await sw.WriteLineAsync($"HTTP/1.1 {code} OK");
        // Écriture de données dans l'en-tête
        await sw.WriteLineAsync("Content-Type: text/html");
        await sw.WriteLineAsync($"Content-Length: {content.Length}");
        await sw.WriteLineAsync("Content-Encoding: UTF-8");
        await sw.WriteLineAsync("Connection: close");
        // Saut de ligne pour séparer l'en-tête du message créé par la partie Echo
        await sw.WriteLineAsync();
        // On pousse le contenu de ce qui a été écrit auparavant (code retour + en-tête)
        await sw.FlushAsync();
        // Copie sur le stream de sortie output du content qu'on a flush
        await content.CopyToAsync(output);
    }
```

La récupération de données se fait avec la syntaxe {ce qui est recherché}.