

## Programmation et projet encadré - L7TI005

Web: HTML, HTTP, récupérer des pages

Crédits supports : Serge Fleury

---

Yoann Dupont, Serge Fleury [prenom.nom@sorbonne-nouvelle.fr](mailto:prenom.nom@sorbonne-nouvelle.fr)

Pierre Magistry [pierre.magistry@inalco.fr](mailto:pierre.magistry@inalco.fr)

2022-2023

Université Sorbonne-Nouvelle

INALCO

Université Paris-Nanterre

## Les enjeux de cette portion de cours :

- comprendre ce qu'est une page web
- comment on la récupère et on gère les erreurs
- avec quels outils, quels protocoles
- démarrer les aspirations d'URL

# HTML



# HTML, c'est quoi

HTML (**H**yper**T**ext **M**arkup **L**anguage) est un langage de balisage pour représenter des pages web. Format reconnu par tous les navigateurs.

# HTML, c'est quoi

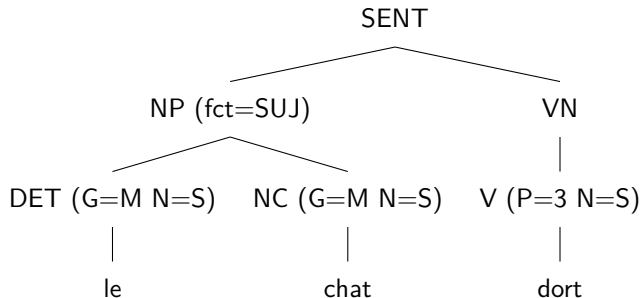
HTML (**H**yper**T**ext **M**arkup **L**anguage) est un langage de balisage pour représenter des pages web. Format reconnu par tous les navigateurs.

Permet de structurer l'information d'un page pour la rendre lisible :

- Dérivé du [SGML](#) (**S**tandard **G**eneralized **M**arkup **L**anguage) et "frère" du XML
- Permet de marquer des zones dans du contenu textuel
- Ces zones fournissent structure et enrichissements

## Du balisage, à quoi ça ressemble ? I

HTML définit des balises qui marquent explicitement le début et la fin d'une zone. On peut inclure des balises dans d'autres, mais pas de chevauchement : HTML se rapproche donc des constituants syntaxiques :



## Du balisage, à quoi ça ressemble ? II

Pour moins d'ambiguïté, les balises sont marquées explicitement. il y en a 3 types :

- Ouvrantes : `<balise>` → le début d'une zone
- Fermantes : `</balise>` → la fin d'une zone
- Ouvrantes et fermantes à la fois : `<balise/>` → "ancree"

## Du balisage, à quoi ça ressemble ? II

Pour moins d'ambiguïté, les balises sont marquées explicitement. il y en a 3 types :

- Ouvrantes : `<balise>` → le début d'une zone
- Fermantes : `</balise>` → la fin d'une zone
- Ouvrantes et fermantes à la fois : `<balise/>` → "ancres"

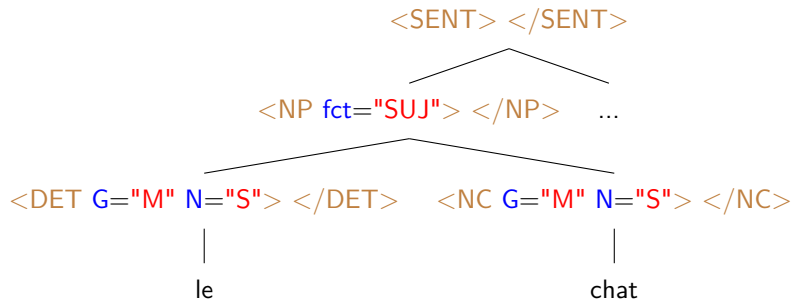
Les attributs d'une balise (donc du nœud) sont des couples clé/valeur renseignés sur la balise ouvrante :

- `<NP fct="SUJ">`
- `<DET G="M" N="S">`



## Du balisage, à quoi ça ressemble ? III

Si on reprend l'exemple mais en XML :



## Du balisage, à quoi ça ressemble ? IV

Le fichier tel qu'il sera écrit au format texte :

```
<SENT>
  <NP fct="SUJ">
    <DET G="M" N="S">Le</DET>
    <NC G="M" N="S">chat</NC>
  </NP>
  <VN>
    <V P="3" N="S">dort</V>
  </VN>
</SENT>
```

HTML reprend la construction globale du balisage, mais a sa propre syntaxe :

```
<html>  
  <head>...</head>  
  <body>...</body>  
</html>
```

HTML reprend la construction globale du balisage, mais a sa propre syntaxe :

```
<html>  
  <head>...</head>  
  <body>...</body>  
</html>
```

Où :

- head : l'entête du fichier (avec les métadonnées)
- body : le corps du fichier (avec le contenu textuel et la structure)

---

Bonne ressource pour approfondir HTML : <https://www.w3schools.com/html/default.asp>

# HTML : l'entête

L'entête `head`<sup>1</sup> contient beaucoup d'informations intéressantes. On reviendra plus en détail plus tard sur certaines.

Une métadonnée particulière nous sera intéressante ici : l'encodage (charset)<sup>2</sup>.

---

<sup>1</sup>Documentation entête HTML : [https://www.w3schools.com/html/html\\_head.asp](https://www.w3schools.com/html/html_head.asp)

<sup>2</sup>Documentation charset HTML : [https://www.w3schools.com/html/html\\_charset.asp](https://www.w3schools.com/html/html_charset.asp)

# HTML : l'entête

L'entête `head`<sup>1</sup> contient beaucoup d'informations intéressantes. On reviendra plus en détail plus tard sur certaines.

Une métadonnée particulière nous sera intéressante ici : l'encodage (`charset`)<sup>2</sup>.

```
<html>
  <head>
    [...]
    <meta charset="UTF-8" />
    [...]
  </head>
  <body>...</body>
</html>
```

---

<sup>1</sup>Documentation entête HTML : [https://www.w3schools.com/html/html\\_head.asp](https://www.w3schools.com/html/html_head.asp)

<sup>2</sup>Documentation charset HTML : [https://www.w3schools.com/html/html\\_charset.asp](https://www.w3schools.com/html/html_charset.asp)

# HTML : Créer un tableau

Pour créer un tableau en HTML, nous avons besoin de 4 balises :

- `table` : la balise racine du tableau
- `tr` : ***table row***, une ligne (se place dans `table`)
- `th` : ***table header***, une cellule d'entête (seulement la première ligne)
- `td` : ***table data***, une cellule classique (toutes les lignes pas entête)

# HTML : Créer un tableau

Pour créer un tableau en HTML, nous avons besoin de 4 balises :

- `table` : la balise racine du tableau
- `tr` : *table row*, une ligne (se place dans `table`)
- `th` : *table header*, une cellule d'entête (seulement la première ligne)
- `td` : *table data*, une cellule classique (toutes les lignes pas entête)

```
<table>
```

```
  <tr><th>livre</th><th>taille</th></tr>
```

```
  <tr>
```

```
    <td>Du côté de chez Swann</td><td>1.0Mo</td>
```

```
  </tr>
```

```
  <tr>
```

```
    <td>L'Assommoir</td><td>990 ko</td>
```

```
  </tr>
```

```
</table>
```



wget, cURL



wget et cURL (*client URL Request Library*) sont deux commandes qui vont pouvoir nous permettre de récupérer des pages web sans passer par un navigateur. Les deux commandes ont des différences qui les rendent intéressantes, même si on privilégiera cURL.

- wget est déjà installé
- cURL devra être installé

`wget` et `cURL` (*client URL Request Library*) sont deux commandes qui vont pouvoir nous permettre de récupérer des pages web sans passer par un navigateur. Les deux commandes ont des différences qui les rendent intéressantes, même si on privilégiera `cURL`.

- `wget` est déjà installé
- `cURL` devra être installé

Dans notre cas, la différence principale entre les deux commandes est que `wget` écrit dans un fichier et `cURL` écrit dans le terminal.

Via le terminal :

```
sudo apt install curl
```

Via le terminal :

```
sudo apt install curl
```

Lancer la commande cURL :

```
curl <URL>
```

Où <URL> est une URL vers une page sur le web.

Quelques options utiles :

- -i : va donner des informations sur l'interaction avec le serveur
- -L : suit les redirections
- -O <fichier> : indique un <fichier> de sortie
- d'autres à voir par vous-même : -I (i), -w, -s

## Quelques tests intéressants avec cURL |

```
curl www.perdu.com  
curl www.google.com  
curl www.youtube.com  
curl www.github.com
```

## Quelques tests intéressants avec cURL II

Regardons déjà les cas où ça va (ou ça fait semblant d'aller ?) :

```
curl -i www.perdu.com
```

```
curl -i www.google.com
```

## Quelques tests intéressants avec cURL II

Regardons déjà les cas où ça va (ou ça fait semblant d'aller ?) :

```
curl -i www.perdu.com  
curl -i www.google.com
```

Et les cas où ça ne va pas :

```
curl -iL www.youtube.com  
curl -i www.github.com
```



## Quelques tests intéressants avec cURL II

Regardons déjà les cas où ça va (ou ça fait semblant d'aller ?) :

```
curl -i www.perdu.com  
curl -i www.google.com
```

Et les cas où ça ne va pas :

```
curl -iL www.youtube.com  
curl -i www.github.com
```

Ce qu'on voit avec ces commandes, ce sont des informations sur la réponse du serveur. Deux lignes nous intéressent tout particulièrement :

- La première de chaque bloc d'entête : HTTP/1.1 XZY <message>
- Celle souvent juste après : content-type: <informations>

## Quelques tests intéressants avec cURL II

Regardons déjà les cas où ça va (ou ça fait semblant d'aller ?) :

```
curl -i www.perdu.com  
curl -i www.google.com
```

Et les cas où ça ne va pas :

```
curl -iL www.youtube.com  
curl -i www.github.com
```

Ce qu'on voit avec ces commandes, ce sont des informations sur la réponse du serveur. Deux lignes nous intéressent tout particulièrement :

- La première de chaque bloc d'entête : HTTP/1.1 XZY <message>
- Celle souvent juste après : content-type: <informations>

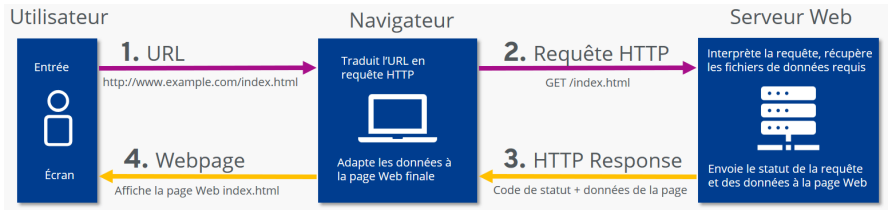
Il s'agit d'un résumé de la communication entre le client (nous) et le serveur (le site hébergé quelque part sur le net). Ces informations sont issues du protocole HTTP.

# HTTP



# Présentation

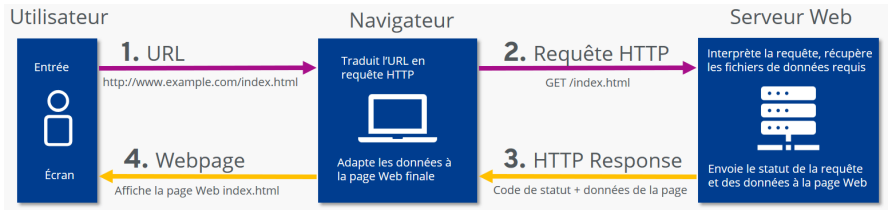
**Hypertext Transfer Protocol.** Protocole de communication entre un client et un serveur pour transmettre pages, média, etc.



**Figure 1:** Schématisation de HTTP (source : [www.ionos.fr](http://www.ionos.fr))

# Présentation

**Hypertext Transfer Protocol.** Protocole de communication entre un client et un serveur pour transmettre pages, média, etc.



**Figure 1:** Schématisation de HTTP (source : [www.ionos.fr](http://www.ionos.fr))

Lorsqu'on utilise des outils comme cURL, le code de statut permet de connaître rapidement le résultat d'une requête.

Permettent d'avoir le statut de la réponse :

- 1xx : information
- 200 : réussite
- 3xx : redirections
- 4xx : erreurs du client
- 5xx : erreurs du serveur

On utilisera ces codes pour (in)valider les requêtes dans les fichiers d'URL.

**Utiliser le tout pour commencer  
la récolte**

---

# L'allure générale du travail de récolte

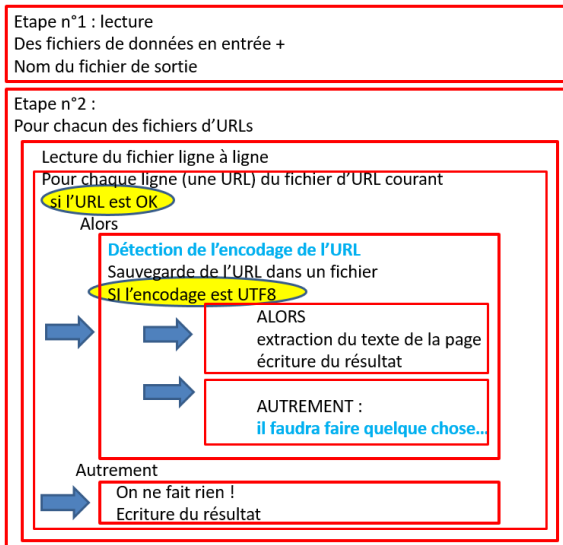


Figure 2: crédits : Serge Fleury



Votre script devra faire plusieurs choses :

- récupérer les URL d'un fichier texte
- écrire un tableau dans un fichier HTML

On a donc deux arguments : le fichier d'URL et le fichier de sortie où on écrit le tableau.

# Lire les lignes d'un fichier en bash

pour lire les lignes du fichier "urls.txt", on peut utiliser le code suivant :

```
while read -r line;
do
    echo $line;
done < "urls.txt";
```

Questions :

1. Pourquoi ne pas utiliser cat ?
2. Comment transformer "urls.txt" en paramètre ?
3. Comment écrire un tableau d'URL en HTML dans le fichier de sortie ?
  - 3.1 Le HTML devra être encodé en UTF-8 et le mentionner dans l'entête.
4. Comment y rajouter le numéro de ligne pour chaque URL ?

# Transformer le script pour lire un dossier

À partir de votre code basé sur :

```
while read -r line;
do
    echo $line;
done < "urls.txt";
```

Toujours en gardant deux paramètres pour votre script :

1. Comment lire non pas un fichier, mais un dossier d'URL ?
2. Comment séparer les résultats dans un dossier de sortie par langue ?