Programmation et projet encadré - L7TI005

mini-projet : préparation au travail en groupe

Yoann Dupont yoann.dupont@sorbonne-nouvelle.fr
Pierre Magistry pierre.magistry@inalco.fr

2022-2023

Université Sorbonne-Nouvelle
INALCO
Université Paris-Nanterre

Les enjeux de cette portion de cours :

- débuter l'écriture de la chaîne de traitement
  - étape 1 : faire des aspirations de pages web à partir d'une liste d'URL et en extraire le contenu textuel
- écrire une sortie en tableau
  - d'abord au format "texte" tabulaire
  - puis au format "web" HTML

Début du mini-projet : commencer la récolte sur des données préparées

Mini-projet

Le mini-projet va permettre de voir (au moins en partie) les différentes étapes du projet de groupe, mais individuellement.

Cela a plusieurs intérêts :

- tout le monde fait le travail essentiel à chaque projet
- permet de vous lancer en douceur sur le travail en groupe

Mini-projet

Le mini-projet va permettre de voir (au moins en partie) les différentes étapes du projet de groupe, mais individuellement.

Cela a plusieurs intérêts :

- tout le monde fait le travail essentiel à chaque projet
- permet de vous lancer en douceur sur le travail en groupe

On démarrera d'une liste d'URL déjà faite, à récupérer sur le dépôt git / icampus du cours.

L'allure générale du travail de récolte

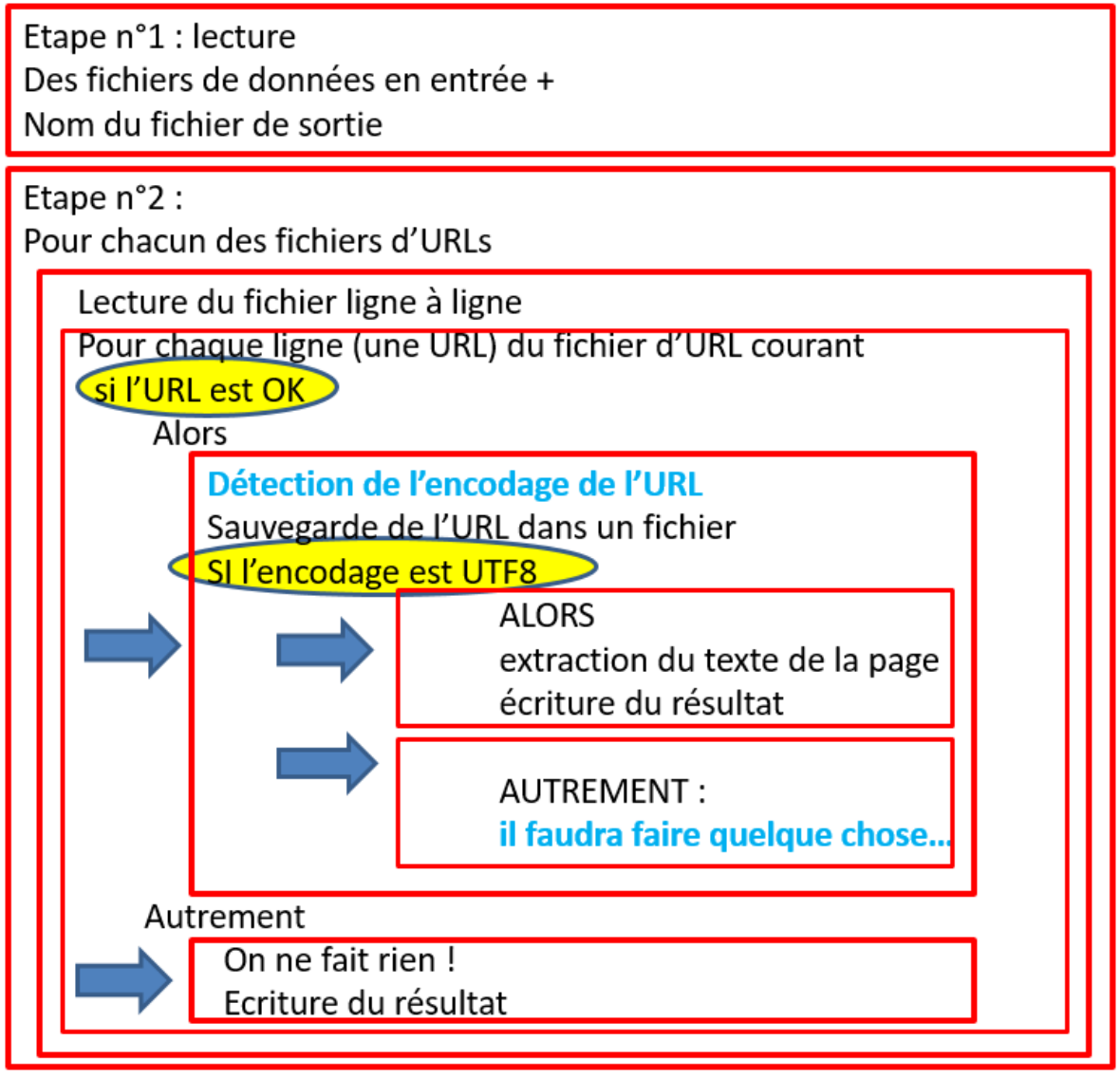


Figure 1: crédits : Serge Fleury

Construction du script

Votre script devra faire plusieurs choses :

- récupérer les URL contenues dans un fichier texte
- écrire sur le terminal des informations séparées par des tabulations
    - on transformera le tout plus tard en page web

On a donc un argument : le fichier d'URL.

Travail à faire

Exercice 1 : lire les lignes d'un fichier en bash

Dans le dépôt vous trouverez le dossier "mini-projet" avec le code suivant :

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

Questions :

1. Pourquoi ne pas utiliser cat ?
2. Comment transformer "urls/fr.txt" en paramètre du script ?
   2.1 S'assurer qu'on donne bien un argument au script, sinon on s'arrête
3. Comment afficher le numéro de ligne avant chaque URL (sur la même ligne) ?
   - Bien séparer les valeurs par des tabulations

Exercice 2 : récupérer les métadonnées de collecte

Après l'exercice 1 fait, on va rajouter des informations à chaque ligne, toujours séparées par des tabulations :

1. le code HTTP de réponse à la requête
   1.1 certaines erreurs peuvent être corrigées
2. l'encodage de la page, s'il est présent

Structure attendue pour le mini-projet

En supposant que votre dépôt s'appelle **PPE1-2023**, l'arborescence de vos fichiers devra ressembler à cela une fois tous les exercices finis :

```
PPE1-2023
├── journal.md
├── exercices
│   └── ...
└── miniprojet
    ├── programmes
    │   └── miniprojet.sh
    ├── urls
    │   └── fr.txt
    └── tableaux
        └── tableau-fr.html
```

Un peu plus loin

créer une page HTML avec les résultats

HTML

HTML, c'est quoi

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

HTML, c'est quoi

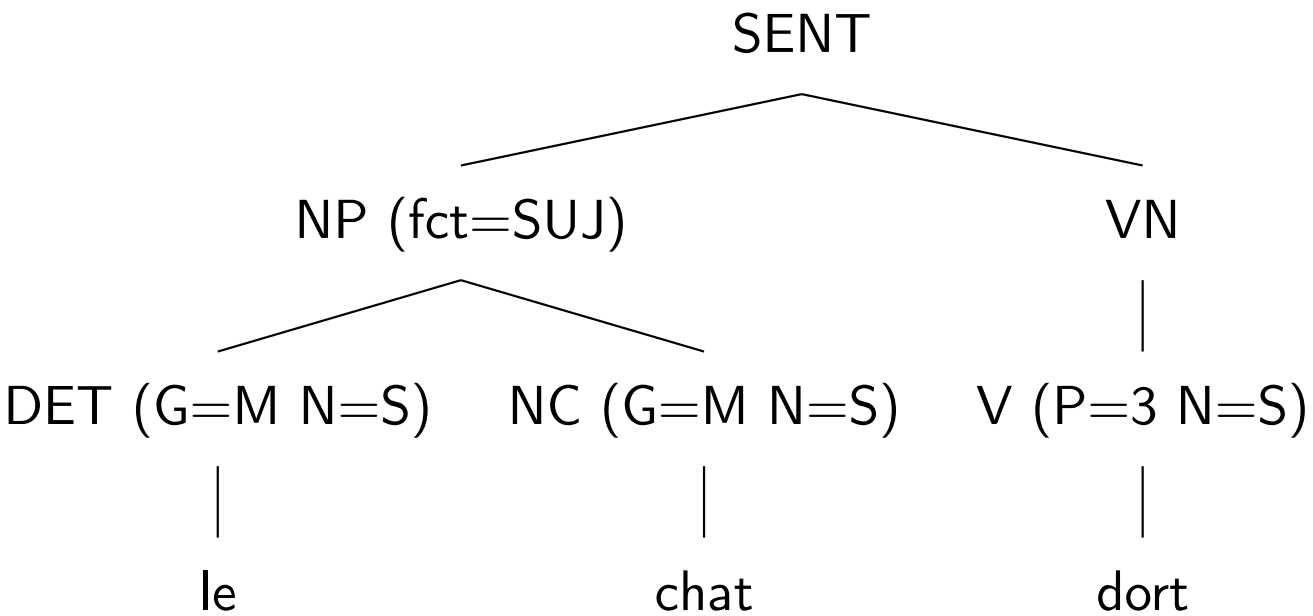
HTML (**H**yper**T**ext **M**arkup **Language**) est un langage de balisage pour représenter des pages web. Format reconnu par tous les navigateurs.

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
- Autofermantes ou vides : <balise/> → position dans le document[1]

[1] Techniquement, cette syntaxe n'est nécessaire qu'en XML. En HTML, il est préférable de s'en passer, comme pour
 par exemple. Source: https://developer.mozilla.org/fr/docs/Glossary/Void_element

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
- Autofermantes ou vides : <balise/> → position dans le document[1]

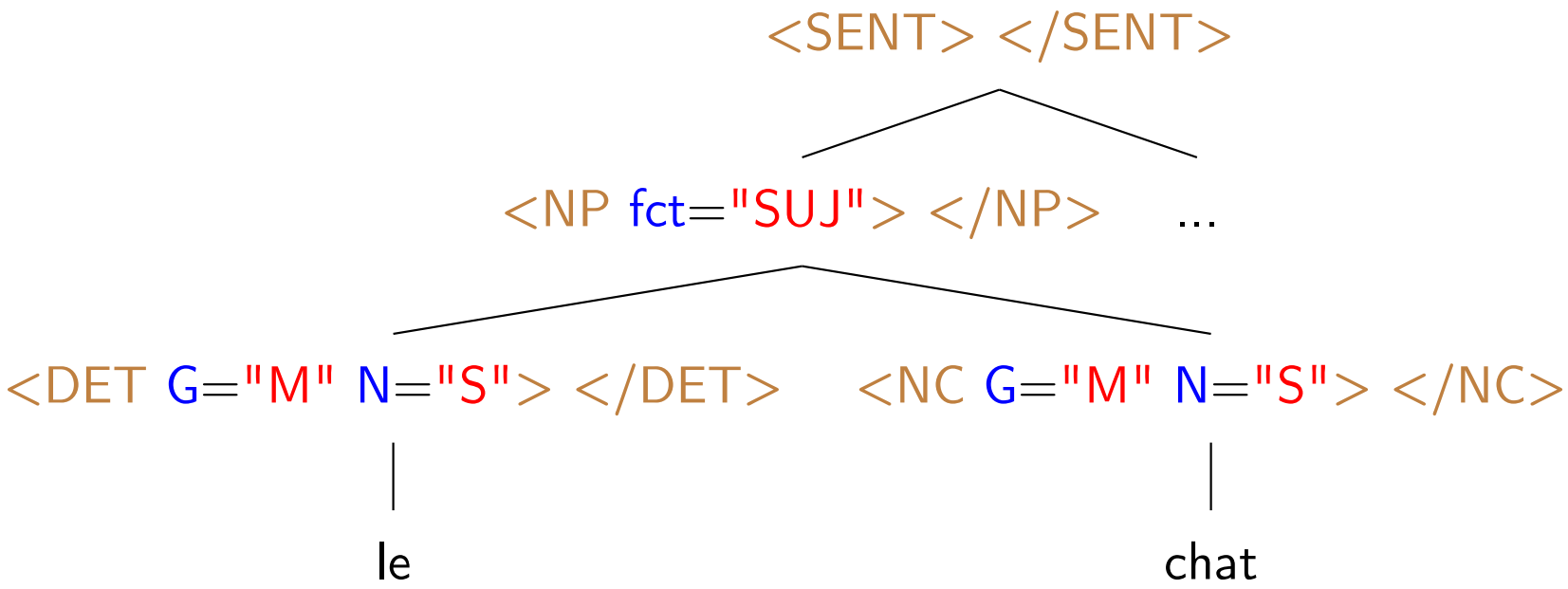
Les attributs d'une balise (donc du nœud) sont des couples clé/valeur renseignés sur la balise ouvrante ou autofermante :

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

[1] Techniquement, cette syntaxe n'est nécessaire qu'en XML. En HTML, il est préférable de s'en passer, comme pour
 par exemple. Source: https://developer.mozilla.org/fr/docs/Glossary/Void_element

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 : les spécificités

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

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

HTML : les spécificités

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`[2] contient beaucoup d'informations intéressantes. On reviendra plus en détail plus tard sur certaines.

Une métadonnée particulière nous sera intéressante ici : l'encodage (charset)[3].

[2] Documentation entête HTML : https://www.w3schools.com/html/html_head.asp
[3] Documentation charset HTML : https://www.w3schools.com/html/html_charset.asp

HTML : l'entête

L'entête `head`[2] contient beaucoup d'informations intéressantes. On reviendra plus en détail plus tard sur certaines.

Une métadonnée particulière nous sera intéressante ici : l'encodage (charset)[3].

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

[2] Documentation entête HTML : https://www.w3schools.com/html/html_head.asp
[3] Documentation charset HTML : 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` : ***t****able* ***r****ow*, une ligne (se place dans table)
- `th` : ***t****able* ***h****eader*, une cellule d'entête (seulement la première ligne)
- `td` : ***t****able* ***d****ata*, une cellule classique (toutes les lignes pas entête)

---

https://www.w3schools.com/html//html_tables.asp

HTML : Créer un tableau

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

- `table` : la balise racine du tableau
- `tr` : ***t**able **r**ow*, une ligne (se place dans table)
- `th` : ***t**able **h**eader*, une cellule d'entête (seulement la première ligne)
- `td` : ***t**able **d**ata*, 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>
```

https://www.w3schools.com/html//html_tables.asp

Exercice 3 : transformer la sortie tabulaire en HTML

Après l'exercice 2, la dernière étape du mini-projet est de transformer la sortie tabulaire au format HTML. La page produite devra contenir :

1. un entête
2. un corps
    - devra contenir au moins le tableau des données récupérées, avec une ligne d'entête et les résultats pour chaque URL.

Ce code HTML devra être écrit dans un fichier `.html` qui devra être lisible par un navigateur web quelconque.

Consignes de rendu

- faire les exercices du miniprojet sur son dépôt individuel
- créer le tag **miniprojet** à la fin du travail et le push sur Github
- créer un fichier texte avec uniquement le lien github vers le tag **miniprojet**
- le déposer jusqu'au lundi 06/11 à 23h59
  - retards acceptés jusqu'au mardi 07/11 à 12h
  - pas de rendu après ça !