Université de Montréal

IFT3225 (Techno. Web)
Projet 1
Shell/CSS/HTML5/Bootstrap/DOM pour le scraping d'images et de vidéos

Contact :
Philippe Langlais
RALI/DIRO
Université de Montréal

+1 514 343 61 11 ext: 47494
felipe@iro.umontreal.ca
http://www.iro.umontreal.ca/~felipe/

▮ dernière compilation : 9 février 2025 (11:47)

Contexte

Dans ce projet, votre but est 1) de lister et éventuellement récupérer les ressources (éléments `img` et `video`) d'une page HTML donnée et 2) de créer une page HTML 5 dynamique qui permet la visualisation de ces ressources. Vous allez pour cela développer deux scripts qui devront fonctionner sous l'environnement Linux du DIRO. Vous allez donc potentiellement manipuler le scriptage (csh, bash ou python) faisant ou non usage de librairies comme beautiful soup vue en démonstration et vous devrez produire des ressources HTML/CSS/javascript qui devront toutes être aux normes.

L'url `https://www-ens.iro.umontreal.ca/hiver/<usager>/index.html` est abritée pour mémoire dans le répertoire suivant sur votre compte au DIRO : `/home/www/ens/<usager>/public_html/index.html`. Plus à ce sujet ici.

Cahier des charges

1. Écrire une commande `extract` qui prend en argument l'url d'une page HTML et qui liste l'ensemble des ressources (images et vidéo) de cette page, Plusieurs options contrôlent le comportement de cette commande qui sont décrites à l'aide de ce synopsis :

```
extract [-r <regex>] % liste seulement les ressources
                       dont le nom  matche l'expression
                       régulière
        [-i]         % ne liste pas les éléments <img>
                             (default: liste)
        [-v]         % ne liste pas les éléments <video>
                             (default: liste)
        [-p <path>]  % liste et copie les ressources img
                       et/ou video  de <url> dans le path
                             (default: juste liste)
        [-h]         % affiche le synopsis de la commande
                       et les auteurs de la commande.
        <url>
```

   Vous pouvez utiliser le ou les languages de votre choix, mais nous lancerons votre commande comme indiqué dans les exemples qui suivent

et qui sont donc normatifs (ils définissent les attentes du cahier des charges). Vous pouvez ajouter des options (par exemple `-R <regex>` pourrait éliminer (filtrer) les ressources dont la source matche l'expression régulière). Dans les exemples qui suivent, une flèche `->` indique que la ligne est plus longue que la largeur de la page et l'affichage de cette ligne continue sur la ligne suivante et ... indique que la ligne a été tronquée.

1) `extract` -v "http://www-labs.iro.umontreal.ca/~felipe/brand_new_home/creative-design/public_html/index.php?lg=fr"
demande de lister les attributs `src` et `alt` (lorsqu'ils existent) des éléments `img` qui sont dans la page dont l'url est spécifiée. La sortie devrait correspondre à :

```
PATH http://www-labs.iro.umontreal.ca/~felipe/brand_new_home/ ->
   creative-design/public_html
IMAGE assets/imgs/logoRALI.png "to RALI"
IMAGE assets/imgs/university-of-montreal-squarelogo.png
IMAGE assets/imgs/sleepy-student.jpg "About us"
IMAGE assets/imgs/student.png "About us"
IMAGE assets/imgs/chilly-g.jpeg "Client Image"
IMAGE assets/imgs/jerome.jpeg "Client Image"
IMAGE assets/imgs/tigran.jpeg "Client Image"
IMAGE assets/imgs/felipe-2009.jpg
```

notez que la première ligne qui indique le path de l'url passée en argument. Ainsi, l'url de la première image est http://www-labs.iro.umontreal.ca/~felipe/brand_new_home/creative-design/public_html/assets/imgs/logoRALI.png.

2) `extract` -p mypath -v "http://www-labs.iro.umontreal.ca/~felipe/brand_new_home/creative-design/public_html/index.php?lg=fr"
demande de lister les attributs des éléments `img` qui sont dans la page dont l'url est spécifiée tout en recopiant localement dans le répertoire `mypath` les ressources (images) de cette page. La sortie devrait correspondre à :

```
PATH mypath
IMAGE logoRALI.png "to RALI"
```

```
IMAGE university-of-montreal-squarelogo.png
IMAGE sleepy-student.jpg "About us"
IMAGE student.png "About us"
IMAGE chilly-g.jpeg "Client Image"
IMAGE jerome.jpeg "Client Image"
IMAGE tigran.jpeg "Client Image"
IMAGE felipe-2009.jpg
```

notez qu'ici, la ligne PATH indique le répertoire dans lequel sont copiées les ressources et que le champ `src` (2è colonne) des lignes IMAGE est local à ce path.

3) `extract` -p mypath -r jpeg -v "http://www-labs.iro.umontreal.ca/~felipe/brand_new_home/creative-design/public_html/index.php?lg=fr"
comme dans l'exemple précédent, mais seules les ressources dont le nom matche `jpeg` seront copiées/listées. La sortie devrait donc correspondre à :

```
PATH mypath
IMAGE chilly-g.jpeg "Client Image"
IMAGE jerome.jpeg "Client Image"
IMAGE tigran.jpeg "Client Image"
```

on peut considérer que l'expression régulière fournie à l'option -r est une séquence de caractères (pas de caractères spéciaux du langage des expressions régulières comme `*`).

4) `extract` "https://zone.votresite.ca/site-web-blogue/photo-video-audio/photos-et-videos:-quand-en-integrer-sur-votre-site-et-ou-vous-les-procurerr gCsl7aSdzo/"
demande de lister les attributs (`src` et éventuellement `alt`) des éléments `img` et `video` qui sont dans la page dont l'url est spécifiée. La sortie devrait correspondre à :

```
PATH https://zone.votresite.ca/site-web-blogue/photo-video-audio/ ...
IMAGE /img/logo-zone-votresite3.png "http://zone.votresite.ca"
IMAGE /images/icon-section-web.png
IMAGE /images/generale.jpg
VIDEO https://videos.francoischarron.com/~francois/medias.francois ...
```

```
IMAGE http://zone.votresite.ca/datascontent/pixabay.jpg "Pixabay"
IMAGE http://zone.votresite.ca/datascontent/istockphoto.jpg "iStockphoto"
```

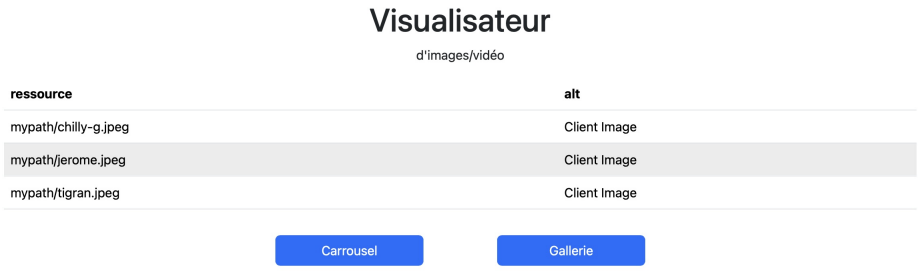
dans le cas d'éléments `video`, la source affichée est soit la valeur de l'attribut `src` s'il existe, l'attribut `src` du premier élément `source` sinon.

2. Écrire une commande `genere` qui lit sur l'entrée standard une liste de ressources (images et/ou video) au format généré par l'extracteur et produit sur la sortie standard une page HTML5 qui permet de visualiser toutes les ressources listées. En voici un synopsis, vous pouvez ajouter des options :

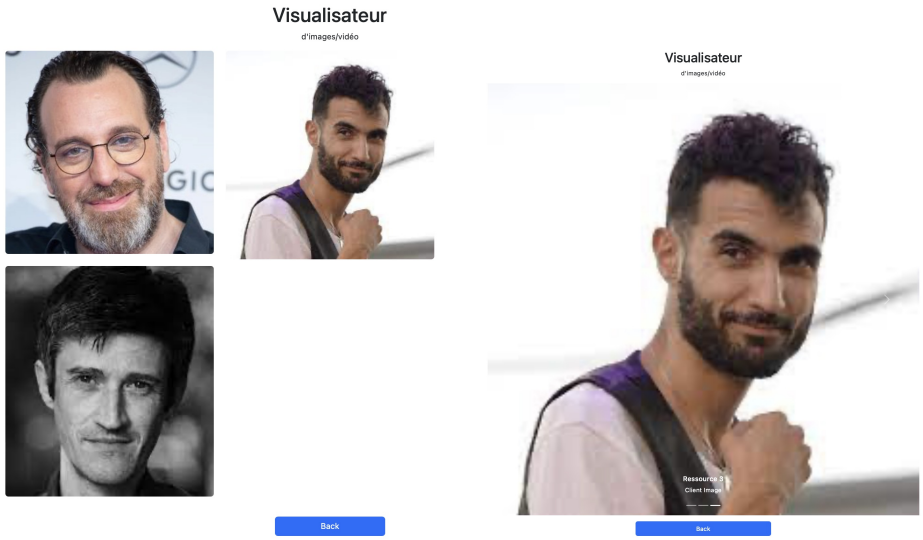
```
genere [  -h  ]    % affiche le synopsis de la commande et
                     les auteurs de la commande.
```

Là encore, vous pouvez utiliser le ou les languages de votre choix, mais nous lancerons votre commande tel qu'indiqué dans l'exemple qui suit (qui est donc normatif).

5) `extract` -p mypath -r jpeg -v "http://www-labs.iro.umontreal.ca/~felipe/brand_new_home/creative-design/public_html/index.php?lg=fr" | `genere >!` mapage.html
produit une page `mapage.html` qui doit ressembler à ceci, c'est à dire une table HTML indiquant chaque ressource et son attribut `alt` si disponible :



la page doit contenir deux boutons : `Carrousel` et `Galerie` qui permettent de visualiser les images de la page à l'aide d'un carrousel (droite) ou d'une galerie (gauche) :



chacune de ces vues contient un bouton `Back` qui permet de revenir à la page initiale (la visualisation sous forme de table).

3. Vous devez produire un rapport au **format pdf** d'au plus 3 pages. Ce rapport doit au moins contenir :
   — le nom et le rôle de chaque membre de l'équipe.
   — 3 urls de pages avec des images et/ou vidéo ainsi que les 3 pages HTML (abritées au DIRO) permettant de visualiser ces 3 pages. Les liens doivent être cliquables afin de faciliter la consultation.
   — la liste des langages et librairies utilisés pour coder votre projet.
   — les traitements particuliers (qu'avez vous fait par exemple avec les images données sous la forme svg ?) que vous avez réalisé.
   — si votre solution n'est pas entièrement compatible avec les contraintes énoncées dans le sujet, indiquez comment lancer vos scripts afin que Théo n'ait pas à deviner comment le faire.

Contraintes

— La page générée par `genere` propose par défaut une vue table. Le fait de cliquer sur les différents boutons doit modifier la vue en cours à l'aide de scriptage DOM : il n'existe qu'une seule page HTML (une par url fournie à la commande `extract`) et cette page ne doit pas contenir

les trois vues (table, galerie, carrousel) que l'on rendrait visible ou cacherait en fonction des clics : seule la table doit être encodée dans votre page HTML.
— De même, lorsque l'on clique sur le bouton `Back` de la vue carrousel ou galerie, c'est par scriptage que l'on doit repasser à la vue table.
— Lorsque vous cliquez sur une ligne de la table (première figure), cela doit faire apparaitre une fenêtre (popup/bulle) avec la photo correspondant à la ligne cliquée, cette photo doit disparaître lorsqu'on relâche le clic.
— Le HTML ne doit contenir aucune commande css, de même que le code javascript doit être entièrement séparé de la page HTML.
— Vos programmes doivent fonctionner sur les machines du DIRO, sans imposer au correcteur une quelconque installation (librairie ou ressource).

Notation

La note de 25 points dépendra de différents critères, comme :
— Adéquation au cahier des charges (deux commandes + des ressources à remettre)
— Ergonomie des pages générées par la commande HTML5
— Bonne gestion de l'aspect dynamique de vos pages
— Qualité du code (lisibilité, élégance, bon choix des languages de programmation)
— Validité des ressources générées (HTML/CSS)
— Le respect des consignes

Mises en garde

Dans la mesure où je vous demande de créer des ressources abritées au DIRO, vous n'êtes pas à l'abri d'un plagiat (sans même le savoir) car les ressources web servies par le DIRO doivent être en lecture pour tous (`chmod og+r`). Ce faisant, rien n'empêcherait une personne de récupérer vos pages. De même les commandes demandées doivent être exécutables par tous. (mais pas en lecture). Plus en cours à ce sujet. Afin de réduire les risques,

je vous recommande de créer un `path` depuis l'endroit où sont abritées vos pages au DIRO avec un ou plusieurs répertoires au nom improbable (ex : `yEW5g93BahmCcKVB`[1]) et dont les seuls droits d'accès sont en eXécution pour les autres (`o`) et le groupe (`g`) (ex : `chmod 711 yEW5g93BahmCcKVB`).

Remise

Le projet est à faire en groupe de deux personnes[2] et être remis sur Studium sous le libellé `projet1` au plus tard le **dimanche 9 mars 2025 à 23h59**. Vous devez remettre votre code et votre rapport (format pdf) dans une archive (gzip, tar, tar.gz) dont le nom est préfixé de `projet1-noms`, où `noms` est à remplacer par l'identité des personnes impliquées dans le projet (`prénom1_nom1+prénom2_nom2`). Cette archive doit comporter votre rapport, le code de vos commandes (et dépendances). Chaque élément de votre archive doit porter les noms des personnes impliquées. Par exemple, votre rapport portera le nom : `prénom1_nom1+prénom2_nom2-rapport.pdf`.

Questions

Posez vos questions sur le canal `projet1` de discord. Prenez le temps de lire le sujet avant de poser des questions. Nous ne répondrons plus aux questions de compréhension trois jours avant la date butoir.

1. N'utilisez pas ce nom ou un nom proche !
2. Si vous êtes sans binôme, même après avoir recherché sur discord, contactez Théo !