SAÉ 1.03
Installation d'un poste de développement

Phase B - Chaîne de traitement

Introduction

Note : toutes les images (sauf **`html2pdf`**) contiennent **`PHP`** en version **`8.3 CLI`**[1]. Si besoin, vous pouvez en faire usage. **Vous ne devez pas utiliser le PHP installé sur vos postes de travail**. L'idée est qu'à l'exception de Docker, votre chaîne de traitement ne doit pas nécessiter la présence d'un outil particulier (tel que PHP par exemple) pour exécuter la chaîne. Tout est packagé dans des conteneurs. Les filtres Unix de base et le Bash de votre machine locale sont acceptés dans votre chaîne.

Etape 3 - La chaîne

Conseils

Vous devez réaliser la chaîne de traitement automatisé de tous les fichiers que vous recevez. Voici quelques conseils pour vous aiguiller sur la méthode :

- Vos scripts sont à écrire en Bash ou en PHP
- A l'exception du script qui déroule tout l'enchaînement et qui doit être exécuté sur votre machine locale, tous vos autres scripts doivent être exécutés dans des conteneurs dont on vous a fourni les images soit en Phase A soit ci-après dans ce document.
- L'option **`--rm`** d'un **`docker run`** vous permettra d'avoir des conteneurs éphémères qui ne laissent pas de traces derrière eux.
- L'option **`-d`** d'un **`docker run`** vous permet de laisser tourner un conteneur en arrière plan et de pouvoir continuer l'exécution de votre chaîne de traitement.
- L'option **`--name`** d'un **`docker run`** permet de donner un nom à un conteneur, ce qui est plus pratique pour le manipuler que d'utiliser son ID

[1] CLI : **C**ommand **L**ine **I**nterface, (Ligne de commande, dans le Terminal)

- Pour envoyer vos fichiers de travail et vos scripts de transformation dans un conteneur ainsi que pour récupérer le résultat produit depuis ce conteneur, vous avez la commande **`docker container cp`**

Surprise !

Vous allez donc créer la chaîne de traitement automatisées des fichiers, mais vous venez juste de recevoir un nouveau lot de fichiers dans lequel l'un d'entre eux a évolué (sur Moodle). C'est la dure vie de développ·eur·euse ! Voici les changements :

- Le tableau des médailles comporte désormais un titre mais l'équipe de développement du site Web n'en a pas besoin. Vous devez l'ôter de manière automatique, ainsi que les en-têtes des colonnes.
- Il faut opérer le calcul du total des médailles par pays (Or + Argent + Bronze)
- Il faut trier les pays par ordre décroissant de médailles d'Or, puis d'Argent pour les pays ayant le même nombre de médailles d'Or, puis de Bronze pour les départager en cas d'égalité sur l'Or et l'Argent.
- Les pays ayant le même nombre de médailles d'Or, d'Argent et de Bronze doivent avoir le même classement (dans ce cas, les ex-æquo doivent être triés par ordre alphabétique) et le rang des ex-æquo (à partir du second) doit afficher un tiret (**`-`**) au lieu du rang. Exemple :

```
44,Afrique du Sud,1,3,2,6
-,Jamaïque,1,3,2,6
-,Thaïlande,1,3,2,6
```

Ce n'est pas tout, un client qui se respecte a toujours des bonnes idées et des besoins qui évoluent !

Il a désormais besoin d'afficher des drapeaux miniatures devant chaque pays.

L'équipe de développement du site Web a besoin que vous lui fournissiez, en même temps que le fichier des médailles, un dossier complet avec tous les drapeaux des pays présents dans le tableau des médailles (et uniquement ceux-là), et ce en deux formats : largeur 20 pixels et hauteur variable pour tous les drapeaux et en 80x60 pixels en style "flottant". Les noms doivent être le code ISO Alpha 2 du pays (https://fr.wikipedia.org/wiki/ISO_3166-1)

Pour ce faire, vous devez télécharger dynamiquement chaque drapeau depuis le site https://flagcdn.com.

Consultez https://flagcdn.com et https://flagpedia.net/download/api qui détaillent le format d'URL permettant de télécharger chaque drapeau dans le format souhaité. Attention, il est possible de télécharger des packs de toutes les images. Ce n'est pas ce qui est attendu, vous devez scripter le téléchargement de chaque drapeau individuel.

Inspirez-vous de la section "Embed on your website" à l'URL https://flagpedia.net/download/api pour savoir comment formater une URL permettant de récupérer des images individuelles (note : **`wget`** qui est évoqué ci-après permet de faire ce qu'un navigateur fait, notamment quand il récupère les images associées aux balises **`<img...>`**)

Vous disposez aussi d'une nouvelle image Docker pour vous y aider : **`sae103-wget`**

sae103-wget

La commande **`wget`** est un outil Unix permettant de récupérer n'importe quelle ressource Web (une page HTML, une image, un fichier CSS, un script JS, etc.) comme le fait un navigateur, mais en ligne de commande. Évidemment ce qui est ainsi récupéré n'est pas affiché (**`wget`** n'a pas cette capacité) mais simplement stocké sur disque. Faites des essais.

La commande principale de cette image est **`wget`**

✏️ **Rendu** : une archive nommée **`etape3-<nom_équipe>.zip`** qui doit être la compression d'un dossier **`etape3/`** contenant tout le nécessaire pour effectuer les traitements de la chaîne.

🕓 **Délai** : la date qu'on fixera ensemble pour le rendu final

📂 **Dépôt** : Moodle

📋 **Attendus** :

Les scripts nécessaires pour effectuer les traitements de la chaîne, ainsi que le script mettant en œuvre la chaîne.
Vous devez aussi fournir une documentation PDF sur la manière de lancer la chaîne pour quelqu'un ne travaillant pas dans votre équipe (votre enseignant par exemple 😉)

N'oubliez pas de détailler qui a fait quoi et en quelle proportion.

Etape 4 - Le tableau des médailles en PDF

Le client veut mettre en ligne sur son site, un PDF d'une seule page affichant le tableau des médailles avec les noms des pays, leur classement, le décompte des médailles de chaque métal et un petit drapeau du pays devant son nom.

Il veut aussi ajouter une dernière colonne représentant le pourcentage de médailles que chaque pays à obtenus. C'est un pourcentage global, tous métaux confondus.

Ce tableau doit tenir sur une unique page A4 et peut (doit) donc être dessiné sur plusieurs colonnes. Prévoir un titre et un logo JO Paris 2024.

Comme au TP 2 sur Docker, votre script doit générer un fichier PDF à partir d'un HTML et de fichiers compagnons (les images). Mais avant de pouvoir en arriver là, vous devez générer le HTML et injecter le tout dans un conteneur de conversion HTML ➡ PDF.

Vous avez l'image Docker **`sae103-html2pdf`** (cf sujet Partie 1) qui doit vous servir à générer ce PDF. Ce n'est pas le même que celui du TP2 mais son utilisation est assez proche.

La commande principale de cette image est **`weasyprint`**

✏️ **Rendu** : une archive nommée **`etape4-<nom_équipe>.zip`** qui doit être la compression d'un dossier **`etape4/`** contenant ce qui est décrit dans les **Attendus**.

🕓 **Délai** : la date qu'on fixera ensemble pour le rendu final

📂 **Dépôt** : Moodle

📋 **Attendus** :

Les scripts de génération du PDF et de lancement de cette génération, ainsi qu'un exemple de PDF produit.