SAÉ 1.03

Installation d'un Poste de Développement

Partie 2 - Utilitaires

Utilitaires

Comme cela a déjà été dit, développer ce n'est pas uniquement coder et c'est même souvent beaucoup d'autres choses que coder.

C'est aussi ce qui rend le métier passionnant… en principe.

Il nous faut souvent mettre en œuvre des outils pour façonner les données, pour préparer ce qui va alimenter les applications que l'on développe.

En aval, il est aussi fréquent de devoir remodeler ce que nos applications produisent pour qu'elles puissent s'intégrer à des environnements plus larges et hétérogènes que ceux prévus (ou pas) au départ.

Il n'est pas rare de devoir s'adapter, en cours de vie d'une application, à de nouveaux venus dans une infrastructure logicielle. On ne peut peut évidemment pas se satisfaire d'un "j'étais là le premier, aux autres de s'adapter à moi". C'est un jeu d'équipe où chacun, chaque application, doit pouvoir apporter ses meilleures pierres à l'édifice.

Pour satisfaire tous ces besoins, et bien d'autres encore qui surviendront dans votre vie professionnelle, vous aurez à mettre en place des outils, des **utilitaires**, pour vous simplifier la vie et faire en sorte que vos applications soient évolutives et facilement adaptables, intégrables, à des environnements hétérogènes. Comme pour les humains, vos applications doivent jouer les bons camarades pour bien s'intégrer à leur environnement.

Contexte

Vous travaillez pour un client qui lance un nouveau produit pour lequel il souhaite communiquer.

Il y aura un site Web pour parler de ce produit et une invitation de clients VIP à une présentation dans une salle de conférence. Un accès visio sur Teams permettra de suivre la présentation pour ceux qui ne pourront pas se déplacer.

Il est prévu que votre société réalise prochainement le site Web, mais votre tâche du jour est de traiter les invitations à la présentation.

Pour cela, vous avez reçu un fichier **`test_invit.csv`** (sur Moodle) qui contient les informations suivantes :

- Prénom de l'invité (colonne **`first_name`**)
- Nom de l'invité (colonne **`last_name`**)
- Nom de la société de l'invité (colonne **`company`**)
- Code unique de l'invitation (colonne **`id`**)
- e-mail de l'invité (colonne **`email`**)

Un fichier CSV[1], a généralement pour séparateur une **`,`** (virgule[2]). Le séparateur peut aussi être un **`;`** (point-virgule), voire plus rarement une tabulation ou d'autres symboles.

Dans un fichier CSV, les champs peuvent parfois être encadrés, individuellement, par des **`"`** (guillemets).

> Comme toujours, quand vous recevez un fichier de travail, jetez-y un œil avec un outil qui ne risque pas de le modifier.
>
> C'est-à-dire : évitez d'utiliser un éditeur de texte, utilisez plutôt **`less`** ou **`cat`** par exemple.

Objectif

Vous allez donc devoir produire des invitations imprimables et téléchargeables pour que votre client puisse les envoyer à ses invités.

Le format retenu est le PDF[3]. Il s'agit d'un format créé par la société Adobe[4] et considéré aujourd'hui comme un format universel, lisible sur la plupart des appareils et des systèmes sans nécessiter l'installation de logiciels spéciaux de lecture.

Notez que, même si c'est une pratique assez commune, envoyer un document MS Word en pièce jointe d'un mail, juste pour que le destinataire puisse avoir quelque chose de proprement formaté et imprimable, est une pratique à bannir pour plusieurs raisons dont les principales sont la nécessité de posséder un logiciel spécial (Word) qui est payant[5] et qui n'est pas forcément installé sur l'ordinateur du destinataire, ainsi que le fait que le document est modifiable facilement par le destinataire. Il en va de même pour tout autre

[1] **C**omma **S**eparated **V**alues
[2] *Comma* signifie *virgule* en anglais.
[3] **P**ortable **D**ocument **F**ormat
[4] Société connue aussi pour ses logiciels *Photoshop* et *Illustrator*, références des arts graphiques.
[5] Même s'il existe aussi des solutions gratuites telles que *Libre Office* ou *Open Office*.

type de document (Excel notamment) : un PDF est propre, léger, non modifiable[6], imprimable. Il a tout pour plaire, utilisez-le.

C'est la raison pour laquelle, en bons (futurs) professionnels, nous allons générer des PDF pour notre client.

Etapes

Avant de se lancer dans la création des 327 PDF, nous allons préparer le terrain en générant manuellement un QRCode, puis un PDF intégrant ce QRCode et enfin vous serez prêt pour la génération automatique de la série.

QRCodes

Pour générer des QRCodes, vous pourriez installer un logiciel spécifique sur votre ordinateur. Peut-être en existe-t-il un déjà présent mais évidemment, vous l'avez compris, nous allons chercher à standardiser nos process de développement et à faire en sorte de dépendre au minimum d'outils à installer afin de simplifier les choses si un membre de l'équipe devait, par exemple, reprendre à son compte votre travail sur son propre environnement de développement.

On pourrait aussi utiliser des services sur Internet mais ils sont bien pour générer un ou deux QRCodes, or nous allons avoir besoin de traiter des volumes conséquents de données et il serait compliqué de trouver un outil en ligne qui satisfasse à nos besoins.

Nous vous avons préparé une image Docker, bien nommée **`qrcode`**, qui va faire exactement ce qu'on souhaite. Enfin, plus précisément, elle va être le point de départ pour faire ce qu'on souhaite au final, car on va devoir adapter un peu l'outil.

Etape 1 - Dans l'hôte

Récupérez l'image depuis le dépôt, comme vous avez appris à la faire en 1ère séance de SAÉ. Vous avez certainement consigné, dans un document de synthèse, la syntaxe à utiliser.

Etape 2 - Dans l'hôte

Vous allez créer un conteneur depuis cette image (**`qrcode`**), et voici quelques explications pour en déterminer les caractéristiques. Lisez intégralement les caractéristiques avant de lancer quoi que ce soit :

[6] Il est possible, avec certains logiciels spécialisés, de modifier un PDF, mais ça demande quand même un certain effort.

Mode Interactif

Vous allez devoir exécuter des commandes dans le conteneur pour générer un qrcode. Cela signifie que le conteneur doit être lancé en mode interactif, sinon il s'arrêtera immédiatement après son lancement.

Pour créer un conteneur dans ce mode interactif, vous devez utiliser les options courtes **`-t`** (pour *attaché au Terminal*) et **`-i`** (pour *Interactif*) qu'on résume généralement en **`-ti`**.

Mode éphémère

Il n'est pas absolument nécessaire de créer un conteneur éphémère mais comme il ne servira qu'une seule fois ou, plus précisément, qu'on ne sait pas si on en aura encore besoin, ni quand, et qu'une fois utilisé on peut s'en débarrasser car on peut le re-créer très facilement, on décide donc de créer le conteneur en mode éphémère, c'est à dire en mode auto-destruction une fois son travail terminé.

Pour créer un conteneur dans ce mode éphémère, vous devez utiliser l'option longue **`--rm`** (comme **r**e**m**ove).

Il ne faut pas hésiter à créer ce type de conteneur quand l'occasion se présente, ça évite de polluer sa machine de conteneurs inutiles qui resteront en état **`STOP`** indéfiniment. Même s'ils ne consomment pas de RAM, ils occupent de l'espace disque inutilement.

Lancement du conteneur qrcode

Voilà, vous disposez des conseils et des informations nécessaires à la création du conteneur **`qrcode`**, à vous de jouer.

Vous devez obtenir un prompt **`bash`**.

Notez le nom du répertoire dans lequel vous êtes arrivé dans le conteneur, vous allez travailler uniquement dans ce répertoire, sinon les commandes ne fonctionneront pas correctement !

Etape 3 - Dans le conteneur

Sans vous déplacer, lancez une commande :

```
qrencode --help
```

C'est votre source de documentation unique. Revenez-y si besoin.

Etape 4 - Dans le conteneur

En vous aidant de l'aide en ligne fournie par la commande **`qrencode`**, créez un QRCode nommé **`code.png`** codant le texte : **`Essai de QRCode ABC`**

Attention, le texte à coder contient des espaces. Vous devez faire en sorte que ce texte soit pris comme un unique paramètre pour la commande qrencode et pas comme 4 paramètres.

Vérifiez que le fichier **`code.png`** a bien été créé dans le conteneur.

Le problème à ce stade est que le fichier est dans le conteneur qrcode et qu'en aimerait le récupérer sur l'hôte.

Etape 5 - Dans l'hôte (2ème Terminal)

Retrouvez l'**`ID`** du conteneur **`qrcode`**.

Vous allez maintenant pouvoir copier le fichier **`code.png`** depuis le conteneur vers l'hôte. Pour ce faire, il y a une commande qui permet de copier des fichiers entre l'hôte et le conteneur. Il s'agit de la commande suivante :

```
docker container cp ID:chemin_source chemin_dest
```

où :

- **`ID`** est celui du conteneur. Il n'est pas nécessaire de donner tous les chiffres hexa, les 2-3 premiers sont largement suffisants. Vous pouvez ainsi les taper à la main.
- Ne pas oublier le **`:`**
- **`chemin_source`** est le chemin absolu (dans le conteneur) de votre fichier à récupérer. Ici ce devrait donc être **`/work/code.png`**
- **`chemin_dest`** est le chemin absolu ou relatif (dans l'hôte) où vous souhaitez placer le fichier récupéré. Il n'est pas nécessaire de donner un nom de fichier, dans ce cas il conservera le même nom. Vous pouvez aussi utiliser **`.`** (point) pour indiquer que vous souhaitez le récupérer dans le dossier courant, le dossier où vous vous trouvez actuellement, au moment de taper la commande.

Allez-y, récupérez votre **`code.png`** du conteneur vers votre hôte, puis visualisez l'image **`code.png`** dans l'hôte.

A l'aide de votre smartphone[7], vérifiez que vous parvenez à lire le QRCode et qu'il contient bien le texte attendu !

[7] En principe l'appareil photo d'un smartphone sait détecter et interpréter les QRCodes qu'il voit.

Etape 6 - Dans le conteneur

Arrêtez votre conteneur **qrcode**, soit par en tapant **exit** ou **CTRL+D** dans le 1er Terminal, soit en l'arrêtant depuis le 2ème Terminal (SAÉ 1.03 - Partie 1 - p 11).

Comme il s'agissait d'un conteneur éphémère (**--rm**), il a disparu à jamais, avec le fichier **code.png** qu'il contenait, mais ce n'est pas grave puisque vous en avez récupéré une copie à l'étape précédente.

Conclusion

Bravo, vous venez d'expérimenter l'image **qrcode** sans rien avoir installé sur votre machine hôte. Maintenant que le conteneur **qrcode** est supprimé, vous avez retrouvé l'état originel de votre hôte. C'est important, parfois on installe des outils qui ne servent qu'une fois et restent là à vie. Mais plus important encore, vous avez une solution reproductible partout rapidement et très facilement.

Justement, à ce stade et dans un réel projet professionnel, vous pourriez inclure cette image (et la documentation afférente) à la documentation de développement pour que n'importe quel membre de l'équipe puisse se substituer à vous et puisse aussi générer des QRCodes, en quelques secondes.

Poursuivons notre travail car pour le moment on est encore loin de générer les **PDF** d'invitations pour notre client.

Astuce de nettoyage

Pour information, si vous voulez purger les conteneurs inutiles (arrêtés), lancez la commande suivante :

```
docker container prune
```

et répondez **Y** à la question.

Ca marche aussi pour les images :

```
docker image prune
```

HTML2PDF

Nous allons maintenant opérer de façon similaire (conteneur éphémère) pour produire un **PDF** à partir d'une page **HTML**.

Nous vous avons préparé une image **html2pdf** pour ça.

Nous n'allons pas détailler toutes les étapes. Vous commencez à connaître le mécanisme, à vous de jouer.

Voici quand même quelques explications :

- Image **html2pdf**
- Conteneur éphémère et interactif à partir de cette image
- Préparez un fichier **HTML** (**invit.html**) avec ces éléments :
  - Un titre : <**H1>Invitation</H1>**
  - Très précisément ceci : ** %PRENOM% %NOM%, **
  - 3 paragraphes **<p>** de faux texte (lorem ou ce que vous voulez)
  - Une image **code.png** (utilisant l'image générée dans l'exercice précédent)
  - Un lien **<a href>** menant vers le site Web de l'IUT par exemple.
- Envoyez les éléments (**invit.html** et **code.png**) dans le conteneur.
- La commande de conversion HTML ➜ PDF est :

> **html2pdf Nom_invité Prénom_invité *fic_html fic_pdf***

  Évidemment vous devez adapter **Nom_invité**, **Prénom_invité**, ***fic_html*** et ***fic_pdf*** ! Par soucis de simplicité, évitez les espaces dans les paramètres.

- Récupérez le **PDF** généré depuis le conteneur vers l'hôte.
- Ouvrez votre **PDF** et vérifiez que tout y est, que le QRCode fonctionne et que le lien est cliquable. Vérifiez aussi que le nom de l'invité s'affiche à la place du ** %PRENOM% %NOM%, **

Conclusion

Re-bravo, ça avance, le client est (bientôt) content !

Automatisons tout ça maintenant.

Automatisation

Si ce n'est pas le cas, les conteneurs éphémères et interactifs **`qrcode`** et **`html2pdf`** doivent être lancés, chacun dans une fenêtre de Terminal.

Etape 1 - Dans l'hôte

A partir du fichier **`test_invit.csv`**, créez un fichier **`invit`** contenant uniquement les champs suivants sur chaque ligne :

- Champ 1 - Le nom de l'invité (**`first_name`**)
- Champ 2 - Le nom de l'invité (**`last_name`**)
- Champ 3 - Le nom de la société de l'invité (**`company`**)
- Champ 4 - L'identifiant unique de l'invité (**`ID`**)

Attention, si des espaces sont présents dans le **`Nom`**, le **`Prénom`** ou la **`Société`**, ils doivent être remplacés par des **`_`** (underscores), il y a des filtres pour faire ce genre de choses.

Ensuite, les champs doivent être séparés par des espaces. Le fichier **`invit`** doit être créé à l'aide de filtres, en une seule ligne d'exécution, c'est-à-dire avec tous les filtres et les tubes que vous voulez mais en une seule étape.

Ceci met en œuvre des choses vues en TP/TD et qu'il est essentiel de maîtriser, en voici la preuve ! Cependant, si vous n'y parvenez pas, et pour ne pas y passer l'heure dessus, recopiez les 10-15 premières lignes du fichier **`test_invit.csv`** dans un fichier **`invit`** et modifiez manuellement les données de **`invit`** pour obtenir le résultat final.

Attention, vous devez omettre la 1ère ligne qui contient les entêtes de colonnes.

Si vous n'y parvenez pas, retirez-la manuellement du fichier final.

Vérifiez votre résultat. Il doit ressembler à quelque chose comme ceci :

```
Liza Whether Wehner-White 032e6f26-8752-44a2-ae44-c2f917221ba3
Camel Tatersale Marquardt_Group dd0156e7-06b4-4ced-942a-20d97d1148f7
```

Si ce n'est pas le cas, la suite ne fonctionnera pas.

Etape 2 - Dans l'hôte

Envoyez votre fichier **`invit`** dans le conteneur **`qrcode`**.

Etape 3 - Dans le conteneur qrcode

Dans le 1er exercice, vous aviez généré un **code.png** manuellement à l'aide d'une commande du genre :

```
qrencode -o code.png "Le texte du QRCode"
```

Cette fois-ci on a besoin de traiter plusieurs centaines de qrcodes (un par ligne du fichier **invit**). Vous vous en doutez, on ne va pas faire ça manuellement, un par un.

L'image docker **qrcode** contient une petite commande qui sait traiter une série en une seule fois. On appelle cela un traitement *par lot*, un *batch* en anglais.

La commande qu'on vous a préparée s'appelle **qrbatch**.

Sa syntaxe est la suivante :

```
qrbatch nom_fic_data
```

A vous de passer le bon ***nom_fic_data*** du fichier de données (c'est sans doute **invit** ?) pour générer les images de QRCodes. C'est l'**ID** de l'invité qui servira pour nommer chaque image QRCode.

Une archive contenant toutes les images de QRCodes sera créée à la fin du batch. Elle s'appellera **out.tgz**.

NB : chaque QR Codes généré contient le texte **Nom/Prénom/Société** de l'invité.

A vous de jouer pour générer vos QRCodes : 327 si vous avez réussi à créer votre fichier automatiquement ou 10-15 si vous y êtes allé manuellement.

Etape 4 - Dans l'hôte

Récupérez l'archive **out.tgz** depuis le conteneur **qrcode**.

Envoyez cette archive dans le conteneur **html2pdf**.

Etape 5 - Dans le conteneur html2pdf

Comme pour le traitement par lot des QRCodes à l'Étape 3, nous vous avons préparé un traitement par lot des **PDF**.

Sa syntaxe est la suivante :

> **`pdfbatch` *`nom_fic_data archive_png modele.html`***

où :

- ***`nom_fic_data`*** est le même fichier de données que vous avez préparé à l'Étape 1 (sans doute **`invit`** ?)
- ***`archive_png`*** est le fichier archive des images QRCodes récupéré à l'Étape 4 (sans doute **`out.tgz`** ?)
- ***`modele.html`*** est le nom du fichier **`HTML`** qui sert de modèle pour créer les **`PDF`** (sans doute **`invit.html`** ?)

Une archive contenant tous les **`PDF`** sera créée à la fin du batch. Elle s'appellera **`pdf.tgz`**.

Attention, traiter 327 invitations va prendre beaucoup trop de temps ! Supprimez des lignes pour ce 1er test et ne gardez que les 10-15 premières lignes. Vous pourrez vous amuser à tout traiter une fois que ce sera terminé et que tout fonctionnera parfaitement (en fin de TP).

A vous de jouer pour générer vos **`PDF`**.

Etape 6 - Dans l'hôte

Récupérez l'archive **`pdf.tgz`** depuis le conteneur **`html2pdf`**.

Extrayez son contenu et regardez votre belle production. Bravo, décidément c'est une séance pleine de succès !

Prochaine séance : on va apprendre à faire encore plus simple et efficace en remplaçant les **`docker container cp`** par des volumes.