File d’attente

Mon tuteur de stage m’a confié un projet intéressant. Le fichier « cron.php » stocke des informations de la base dans 4 fichiers textes différents, et les envois en ftp sur un serveur. Le client qui a fourni un cahier des charges, a déjà rencontré des problèmes avec ce script. En effet, il arrive que certains fichiers ne sont pas envoyés pour des raisons diverse, comme une perte de connexion, autant du coté serveur FTP, que base de données. Mais il n’y a qu’une seule tentative d’envoi, donc si elle échoue, si le client s’en rend compte, est obligé de le signaler à l’infogérant, qui doit lui-même faire l’envoie, souvent par mail. De plus comme il y a 4 fichiers, il arrivent que seuls certains fichiers ne soient pas envoyés. Puis cela utilise beaucoup de ressources. Alors le projet doit permettre compacter les 4 fichiers et de créer une file d’attente pour l’envoie de fichiers, tant que ces derniers ne sont pas correctement envoyés.

Il faut d’abord apporter une modification à la base de données. En effet c’est dans la base que va être la file d’attente. Dans le cahier des charges, une ébauche des modifications de la base de données sont fournis. Après étude de celui-ci, et réflexion avec mon tuteur de stage, on convient que des modifications plus simple, utilisant moins de ressources et plus logique peut être mit en œuvre pour la modification de la base de données. J’ai alors rédigé le script de modifications de la base de données. Les modifications apportées sont :

* L’ajout de la table « exportCommande » dans laquelle va être stocké :
  + Un identifiant de la file d’attente.
  + Le nom du fichier a exporté.
  + La date de création de l’occurrence.
  + La date de modification de l’occurrence.
  + Le statut du fichier (« En attente » ou « Envoyé »).
  + Le nombre de tentative d’envoie
* L’ajout de la table « statutExport », dans laquelle va être stocké :
  + Un identifiant du statut.
  + Le libellé du statut.
* L’ajout d’un champ clé étrangère à la table « Commande ».
* Mise en place des contraintes d’intégrités fonctionnelles entre :
  + « exportVente » et « statutExport »
  + « commande » et « exportVente »
* Et l’ajout des 2 statuts d’exportation dans la table « statutExport ».

L’ajout d’un champ dans la table « commande » génère une erreur sur le passage d’une commande sur le site lui-même. En effet, lors de la commande, une requête est exécutée pour insérer une commande dans la table. Mais la requête ne spécifie pas les champs sur lesquelles de nouvelle données vont être insérer. Ainsi une erreur sql 1136 survient, car il y a moins de valeurs passées que de champs dans la table. J’ai alors modifié cette requête dans le script « edenred.php » en spécifiant les champs concernés, c’est-à-dire tout ceux présent sauf celui que j’ai moi-même rajouté. Une erreur de syntaxe d’un des champs, créer une erreur surprenante, car le style de la page ne s’affiché plus. La correction de cette erreur rétablit le comportement du site à la normal.

J’ai ensuite codé une fonction permettant de créer une archive zip, pour compresser les 4 fichiers de l’exportation. J’ai entamé des recherches, et m’es inspirant d’exemple de code trouvé ici et là, je constitue rapidement cette fonction. Des fonctions natives de php permet de créer une archive .zip. Il suffit de créer une instance de classe « ZipArchive », et utilisé la fonction « open() » de cette classe, avec comme paramètres, le nom du fichier et le mode d’utilisation de l’archive (ZipArchive::CREATE) indiquant qu’il faut créer l’archive. Le nom de l’archive a été défini dans le cahier des charges par le client, il doit commencer par « FTP\_VEL\_ », suivi de l’identifiant dans la base de données, puis de la jour, mois, année, et heure, minute, seconde, comme ceci :

* FTP\_VEL\_661\_1313020202201420142014\_151549492424.zip.

Pour pouvoir récupérer l’identifiant dans la base de données, j’ai dû créer l’occurrence en utilisant la fonction de « codeigniter », « insert\_string », qui génère la requête SQL, avec les paramètres dans la fonction. Et grâce à la fonction « insert\_id », de « codeigniter », j’ai pu récupérer le dernier identifiant de la table, avec lequel j’utilisais pour le nom du fichier. Une fois que mon archive sera prête, je mettrais à jour l’enregistrement dans la base avec le nom du fichier, grâce à la fonction « update\_string », qui réalise la même tache que « insert\_string », sauf qu’elle génère un requête de mise à jour.

Puis l’utilisation du de la fonction « addFile() », pour ajouter des fichiers. Ses paramètre sont le chemin absolu du fichier et un seconde paramètre facultatif permet d’indiquer le nom du fichier à inscrire dans l’archive. Sans cela, c’est le premier paramètre qui sert pour le nom du fichier, ainsi dans l’archive se créer tous les répertoires successifs ou se trouve les fichiers, car c’est le chemin absolu. Ainsi en indiquant le nom du fichier, il se trouve à la « racine » de l’archive, sans la création de répertoire.

Je me suis ensuite attelé à la fonction ajoutant l’archive dans la file d’attente. Cette fonction doit ajouter une occurrence dans la base de données, dans la table (exportCommande) créée spécialement. Elle met également à jour les commandes concernées par l’export. En effet, grâce à la contrainte d’intégrité, chaque commande se trouve dans un et unique archive prêt à l’exportation.

Ensuite, suite à la demande de mon tuteur de stage j’ai créé un second fichier. Dans celui-ci j’ai créé une fonction qui extrait de la base de données, tous les exportations qui sont en attente d’envoi. Fonction rapidement créée, qui ne fait qu’exécuter une requête, et qui boucle autant de fois qu’il y a de fichier à envoyer.

J’ai donc entamé la création d’une fonction qui envoie les archives en SFTP. Pour cela, je me suis appuyé sur un code déjà existant dans le script « cron.php », qui envoyé les 4 fichiers textes. Ce code utilise une librairie déjà installé sur le serveur. J’ai retravaillé ce code en envoyant qu’un seul fichier, en supprimant les instructions superflues, et en mettant en place des journalisations d’erreurs et des alertes mails. Si l’envoi en SFTP échoue, une requête de mise à jour incrémente le nombre d’essais. Par contre si l’envoi réussi, j’incrémente le nombre d’essais, mais je passe l’export dans la base de données en statut numéro 2, soit statut de réussite d’envoi.

Malheureusement, je puis réaliser des tests sur le serveur FTP du client, car mon tuteur de stage, et je le comprends, ne souhaite pas que je réalise ces tests sur le serveur du client, afin de ne pas polluer le serveur. Alors il m’a donné l’accès à son serveur SFTP personnel, quelques jours plus tard, pour que je puisse réaliser mes tests. J’ai rencontré un léger problème lors de l’envoie des fichiers en SFTP. En effet le chemin de destination des fichiers sur le serveur distant n’était pas bon. J’ai essayé de les placer dans le répertoire « /IN », hors je n’ai premièrement pas accès à la racine du serveur et de plus il y a de très grande chance pour que le serveur n’est pas de répertoire à sa racine se nommant « IN ». Il m’a suffit de mettre « IN/ », car ce répertoire se trouve à la racine de l’utilisateur SFTP.

Pour assainir mon code j’ai créé une fonction pour « logger » dans un fichier texte. Ainsi chaque fonction n’a plus besoins d’ouvrir le fichier de log, de faire appel à la fonction « fwrite » et de refermer le fichier de log. Un simple appel à cette fonction en passant en paramètre le message qu’on souhaite journaliser suffit.

Une fois cela fait mon tuteur de stage souhaite que ma journalisation soit plus précise quand une erreur survient. En effet il faudrait savoir si c’est à cause du manque du fichier source, ou un problème avec le répertoire destination sur le serveur distant, un problème de droit ou d’espace disque. J’ai alors commencé par vérifier l’existence de l’archive sur le serveur, en utilisant une fonction native du PHP, qui est « file\_exists » dans laquelle je passe le chemin absolu des archives. Cette fonction retourne « TRUE » si le fichier existe bien, sinon « FALSE ». Si la fonction retourne « FALSE », la journalisation le consigne, et une alerte mail est déclenchée.

Pour m’assurer de l’existence du répertoire sur le serveur distant j’ai fait appel à une fonction de libraire permettant d’envoyer SFTP les archives. La fonction se nomme « lstat », dans laquelle je passe en paramètre le répertoire du serveur distant. Cette fonction retourne un tableau contenant diverse informations sur ce répertoire. Mais s’il n’existe pas, alors la fonction ne renvoie rien. Ainsi je n’ai qu’à vérifier si elle retourne le tableau ou pas. Si elle ne renvoie rien, je log et déclenche une alerte mail.

Pour ce qui est des erreurs concernant le serveur distant, mon tuteur m’indique que le serveur doit très certainement me retourner des messages concernant les erreurs. Des messages que l’on peut retrouver dans la console de FileZilla. Alors j’ai essayé de récupérer le message que me renvoie le serveur, pour cela mon tuteur m’indique d’aller dans la fonction qui permet d’envoyer des fichiers en SFTP. J’ai tout d’abord fait une recherche dans les fichiers pour trouver la fonction « put ». Une fois cela fait, je cherche un moyen de récupérer le message du serveur. Il se trouve que dans la fonction, il se trouve une variable qui récupère d’une les messages serveur. J’affiche cette variable dans un « echo », et je fais des tests. Quand l’envoie de fichier est réussi le message « Success » est retourné par le serveur, sinon selon la nature de l’erreur, le message délivré par le serveur diffère. Mon tuteur me signale qu’il n’y a pas besoins que je personnalise le message dans le journal et dans le mail d’alerte. Il suffit de récupérer ce que me retourne le serveur et de « logger » et envoyer un mail. Cependant mon tuteur ne souhaite pas que je modifier la fonction. Pour plusieurs raisons, tout d’abord pour ne pas modifier la fonction et donc la librairie. La variable que j’utilisais, récupéré le message du serveur grâce à une fonction. Cette fonction commence par un under-score « \_ », ce qui signifie qu’elle est utilisée uniquement par d’autre fonction de la librairie mais n’avait pas d’être appelé dans des scripts hors de la librairie. Lors de l’appel de cette fonction, cela généré une erreur conséquente. En effet non seulement la page ne se chargé pas et le navigateur était en attente un long moment jusqu’à retourner une erreur 503 (site en maintenance ou surcharge du trafic). De plus j’étais obligé de fermer mon navigateur pour ré-accéder au site internet. J’ai donc réaliser des recherche sur internet, lesquelles mon menés à la fonction « getSFTPErrors() ». C’est fonction retourne un tableau d’erreurs rencontrées. Une structure conditionnel placé après de l’envoie du fichier en SFTP, vérifiant si le tableau n’est pas vide, permet « logger » l’erreur tel que retournée par le serveur et déclencher une alerte mail. J’extrais le contenu du tableau, donc l’erreur retournée pour la consigner dans le journal et dans le mail d’alerte. Si une erreur survient, la page est tuée.

Suite à la visite de mon professeur de suivi technique, il m’a conseillé d’alerter par mail, un responsable si il y a un problème lors de la consignation dans le journal de log. Pour cela, dans la fonction que j’ai créé pour consigner dans le journal, je vérifie dans des structures conditionnelles l’ouverture du fichier (fonction native PHP « fopen »), et l’écriture dedans (fonction native PHP « fwrite »). S’il y a un problème, alors un mail est envoyé en stipulant l’erreur.

Je n’ai pas pu respecter le planning que je m’étais fixé. En effet mon tuteur de stage étant énormément pris pour des tâches plus importantes, la mise en place du serveur SFTP sur lesquelles je dois réaliser mes tests n’a été faite que quelques jours plus tard. Je n’avais pas anticipé cette contrainte avant de réaliser ce projet. J’aurai du répartir les tâches sur une série de jours non chronologique, et prévoir des jours ou j’aurais réalisé des tâche étrangère à ce projet, comme si ce qu’il s’est passé en réalité.

Mon tuteur de stage, m’a demandé de passer ce projet sur le serveur de recette. Cela implique donc l’injection du script SQL dans la base de données, copier les fichiers sur le serveur, et apporté quelques modifications dans le fichier « constants.php ». Il ne faut pas non plus oublier de changer la requête dans le fichier « edenred.php », qui, à cause de la modification de la base, et notamment de la table « commande », génère une erreur (voir plus haut). Une fois cela fait, je réalise un test, qui marcha du premier coup.