

# Université **IBM i**

## 19 et 20 novembre 2024

### IBM Innovation Studio Paris

#### **S23 – PHP sur IBM i : 30 trucs et astuces en 60 mn**

19 novembre 16:00 - 17:00

**Gautier Dumas**

CFD-Innovation

*gdumas@cfd-innovation.fr*



uui2024

#ibmi

#uui2024



common  
FRANCE



INNOVATION



Conseil



Formation



Développement



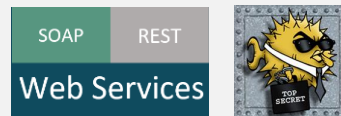
A  
S  
S  
I  
S  
T  
A  
N  
C  
E



Web  
et  
Open Source



Communication



Valorisation des  
données





# L'agenda de la session

1. Les « à-côtés » qui comptent
2. Les trucs et astuces de développement
3. Les trucs et astuces d'administration



# Pour commencer

## Les « à-côtés » qui comptent

# #1 – Installation & déploiement PHP

- Plusieurs solutions PHP disponibles
  - CommunityPlus+ PHP by Seiden Group
  - Zend Server by Perforce
  - ZendPHP by Perforce
- <https://www.ibm.com/support/pages/php-ibm-i>
- Installation via **yum** et paquets au format **rpm**
  - Rapide et facile à installer et maintenir
    - **yum install "php-\***
  - Installations et déploiement automatisables



# #1 – Installation & déploiement PHP

- Versions PHP actuellement disponibles avec le CP+
  - 7.4, 8.0, 8.1, 8.2, **8.3**
  - Mise à disposition rapide pour IBM i des nouvelles versions dès leur sortie
- Un grand nombre d'extensions disponibles
  - <https://www.seidengroup.com/php-documentation/whats-included-seiden-php/>
  - ibm\_db2
  - LDAP
  - ssh2
  - Soap
  - ...



## #2 – Cohabitation de plusieurs versions PHP

- Sur une même LPAR, il est possible d'avoir autant de versions et configurations PHP que nécessaires
  - Pour des environnements séparés (dev, test, prod...)
  - Pour des applications spécifiques ayant des prérequis de version PHP
  - Pour préparer les montées de versions

➔ A l'aide des **containers chroot**

- Création container : `$ chroot_setup /QOpenSys/containers_chroot/dev`
- Installation PHP dans container :

```
$ yum install --installroot=/QOpenSys/containers_chroot/dev "php-*
```

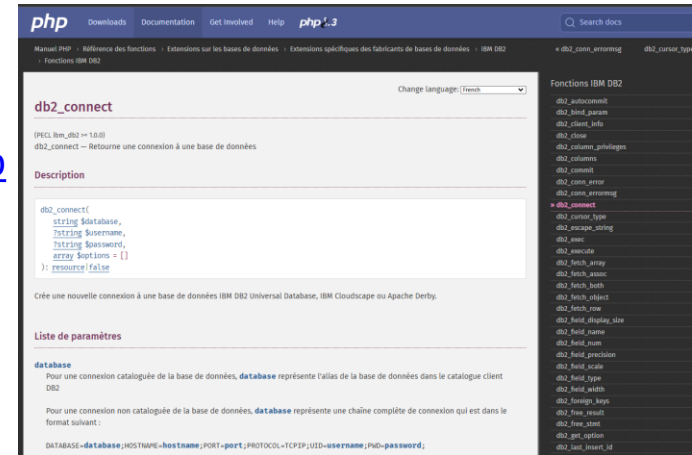
## #2 – Cohabitation de plusieurs versions PHP

- L'outil Open Source **siteadd** de Seiden Group facilite la création d'une instance Apache utilisant le PHP d'un container chroot
    - `$ addsite -c /QOpenSys/containers_chroot/dev -n PHP83 -p 1988 -l`
      - `-c` pour indiquer le container chroot à utiliser pour PHP
      - `-n` pour indiquer le nom de l'instance Apache
      - `-p` pour indiquer le port d'écoute
      - `-l` pour l'utilisation des configurations PHP au même niveau que l'instance Apache
- Exemple : `/www/PHP83/phpconf` contiendra le `php.ini` et les configurations propres à cette instance sans avoir besoin d'aller dans le container



# #3 – La documentation php.net

- La documentation de référence pour PHP (*à privilégier par rapport à Stackoverflow*)
  - <https://www.php.net/>
- À jour et permettant de switcher sur les différentes versions PHP
- Multi-langues (dont le français)
- L'extension DB2 est très bien documentée
  - <https://www.php.net/manual/fr/function.db2-connect.php>
- Versions LTS
  - <https://www.php.net/supported-versions.php>
- Guides de migration
  - <https://www.php.net/manual/en/migration84.php>



# #4 – Rester à jour !

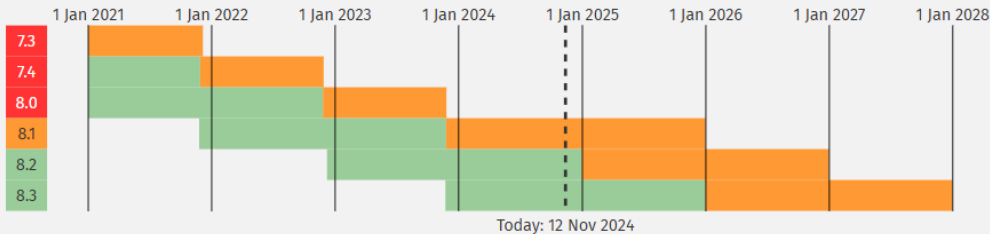
- PHP n'est pas RPG
  - Seul RPG supporte 65 ans de compatibilité ascendante
- Le PHP a besoin de mises à jour régulières
  - Pour intégrer des patchs de sécurité
  - Pour intégrer des nouvelles fonctionnalités
  - Pour se séparer de fonctions dépréciées (et avoir le temps de préparer leur suppression)
    - Une fonction dépréciée est retirée deux versions après son passage en dépréciée
- PHP est une technologie de pointe et cela nécessite des mises à jour fréquentes
- Pour éviter la dette technique et encourager la veille technologique

# #4 – Rester à jour !

## Currently Supported Versions

Branch	Initial Release		Active Support Until		Security Support Until	
<a href="#">8.1</a>	25 Nov 2021	2 years, 11 months ago	25 Nov 2023	11 months ago	31 Dec 2025	in 1 year, 1 month
<a href="#">8.2</a>	8 Dec 2022	1 year, 11 months ago	31 Dec 2024	in 1 month	31 Dec 2026	in 2 years, 1 month
<a href="#">8.3</a>	23 Nov 2023	11 months ago	31 Dec 2025	in 1 year, 1 month	31 Dec 2027	in 3 years, 1 month

Or, visualised as a calendar:



### Key

Active support	A release that is being actively supported. Reported bugs and security issues are fixed and regular point releases are made.
Security fixes only	A release that is supported for critical security issues only. Releases are only made on an as-needed basis.
End of life	A release that is no longer supported. Users of this release should upgrade as soon as possible, as they may be exposed to unpatched security vulnerabilities.

*Tableau des versions actuellement supportées (à date de novembre 2024) :*

Source <https://www.php.net/supported-versions.php>

# #5 – Utiliser un bon éditeur de code

- Eviter notepad++ & bloc note
  - A comparer à SEU -> old school
- **VSCoDe & Code for IBM i**
  - Intégration et connectivité IBM i (ssh)
  - Fonctionnalité de déploiement
  - Gestion de git intégrée
  - Pour aller plus loin, différentes extensions à choisir et à configurer



- **PHP Storm**

- Conçu spécifiquement pour le développement PHP
- Pour booster la productivité des développeurs
- Des outils directement intégrés (déploiement, versioning, débogage, tests unitaires ...)
- Fonctionnalités avancées de refactoring et d'analyse de code (pour identifier les erreurs, les mauvaises pratiques et les améliorations potentielles en temps réel)



# #6 – Connectivité ssh

- Privilégier `ssh` / `sftp` pour interagir avec
  - Les scripts (le code) PHP, HTML, CSS, JS ...
  - Les configurations Apache / PHP
  - Les logs Apache / PHP
- Rapide, sécurisé, standard
- De plus en plus utilisé sur IBM i
  - Gestion des packages Open Source
  - Code for IBM i
  - CI / CD
  - ...
- Eviter les partages Windows / FTP



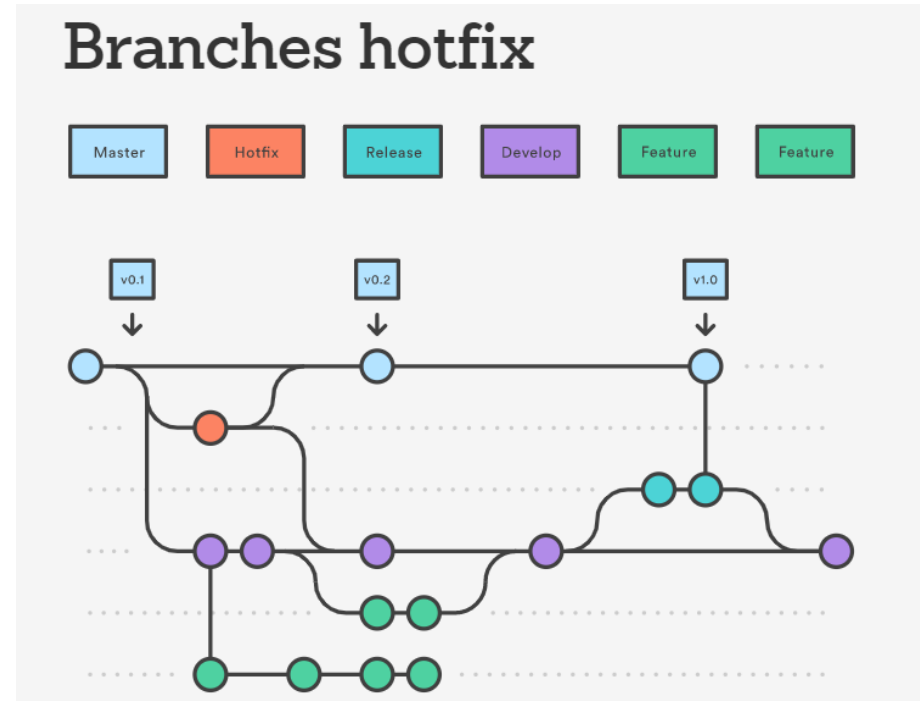
# #7 – Composer

- Pour gérer les dépendances PHP
  - Gestion des référentiels
  - Téléchargement des packages
  - Installation des packages dans les versions compatibles avec votre environnement
  - Upgrade et downgrade des versions simplifiées
  - `$ composer require PHPMailer/PHPMailer`
- Pour gérer l'autoload des classes
  - `require_once('../vendor/autoload.php');`
- Référentiel de packages : <https://packagist.org/>



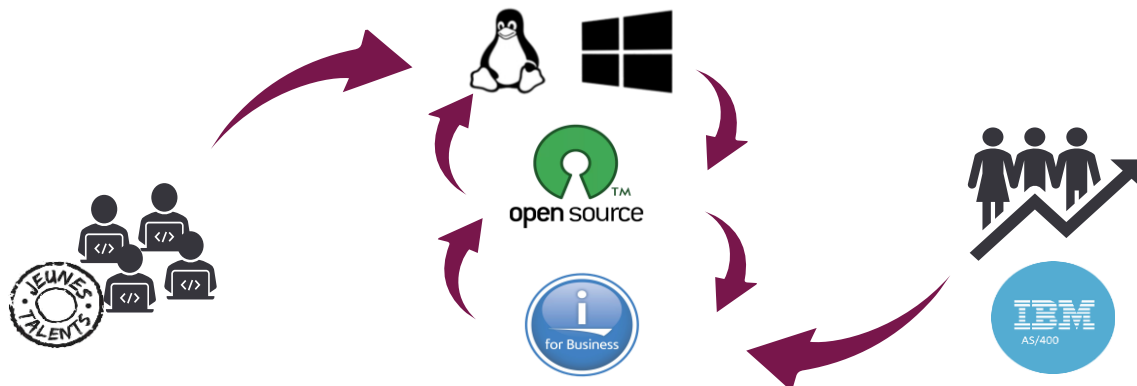
# #8 – Versioning avec git

- Le développement web se prête particulièrement bien au versioning avec Git
- Utilisation standard de git
- Pourquoi versionner son code ?
  - Pour gérer les versions du code au fur et à mesure des modifications
  - Pour faciliter les rollbacks
  - Pour garder un historique des modifications
  - Pour faciliter le travail en équipe et paralléliser les développements
  - Pour répondre à des exigences légales
  - Pour pouvoir mettre en place du CI / CD



# #9 – Une porte d'entrée sur l'IBM i

- Pour faire face à la pénurie de compétences IBM i
  - Intégration des équipes web sur l'IBM i
  - Trouver plus facilement de nouveaux talents à intégrer dans vos équipes IBM i
- Faire monter en compétences les développeurs IBM i traditionnels
- Favoriser les échanges entre les équipes micro et IBM i.







# Trucs et astuces de développement

# #10 – Gestion des exceptions

- Gérer vos erreurs ! Elles font parties de l'application
- Eviter d'utiliser l'opérateur @ devant vos fonctions
  - Suppression / masquage des erreurs
  - Equivalent de MONMSG CPF0000
- Apprendre à gérer les erreurs
  - Try & Catch <https://www.php.net/manual/fr/language.exceptions.php>

```
$file = @file_get_contents("fichier_inexistant.txt");
```

```
function lireFichier($chemin)
{
    if (!file_exists($chemin)) {
        // Si le fichier n'existe pas, lancer une exception
        throw new Exception("Le fichier $chemin n'existe pas.");
    }

    return file_get_contents($chemin);
}
```

```
try {
    // Essayer de lire un fichier qui n'existe pas
    $contenu = lireFichier("fichier_inexistant.txt");
    echo $contenu;
} catch (Exception $e) {
    // Attraper et gérer l'erreur
    echo "Erreur : " . $e->getMessage();
}
```

# #11 – Bien choisir la méthode HTTP

- On peut s'appuyer sur la norme RESTFull pour des API Backend
  - **GET** => Pour les opérations de lecture (SELECT SQL)
  - **POST** => Pour les opérations de création (INSERT SQL)
  - **PUT** => Pour les opérations de mise à jour (UPDATE SQL)
  - **DELETE** => Pour les opérations de suppression (DELETE SQL)
  
- Pour la partie Front et navigation dans une application
  - **GET** est idéal pour les URL intégrées
    - Lien vers une page détails par Id
    - Routage d'une application
    - Faire transiter des données non sensibles
  - **POST**
    - Pour faire transiter des informations sensibles ou complexes
    - A coupler avec du HTTPS



# #12 – Connectivité base de données

- De nombreux connecteurs base de données sont disponibles
  - DB2
    - ODBC / PDO\_ODBC
    - ibm\_db2 / PDO\_IBM
  - MySQL
    - Mysqli, pdo\_mysql
  - SQL Serveur
    - Pdo\_dblib (FreeTDS)



➔ A privilégier : **PDO\_ODBC**

- ➔ ODBC : Portabilité des applications IBM i / Windows / Linux
- ➔ PDO (*PHP Data Objects*), portabilité des bases de données (DB2, MySQL ...)
- ➔ Utilisation du package rpm **ibmi-access** (odbc sur IBM i)

# #13 – Préparer les requêtes

- Utiliser seulement des requêtes préparées
  - Aident à se protéger contre les attaques par **injections SQL**
  - Les données utilisateur sont correctement échappées, ce qui rend le code plus sûr sans avoir à manipuler manuellement les inputs

```
$requete = $pdo->prepare("SELECT * FROM utilisateurs WHERE email = :email");  
$requete->execute(['email' => $email]);  
$resultat = $requete->fetchAll();
```

- Principe et syntaxe existante avec le SQLRPG

# #14 – Utiliser des procédures stockées

- Les procédures stockées sont comme des programmes SQL
  - Procédures stockées [SQL](#)
  - Procédures stockées [externes](#) : fait le mapping entre un \*PGM et une procédure SQL
- Invocables par PHP par l'intermédiaire d'une simple connexion BDD
  - Gestion des paramètres input / output
  - Récupération d'un ou plusieurs result sets
- Permet :
  - De centraliser les règles métiers (business logic) dans la base de données ou dans les programmes RPG existants
  - La réutilisation du code existant

# #15 – Typage des paramètres / constantes

- **Avant PHP 7** : PHP ne supportait pas le typage strict des paramètres ; les types de données étaient souvent convertis automatiquement
- **Depuis PHP 7+** : Introduction du typage pour les paramètres et le retour des fonctions, permettant un code plus robuste et moins sujet aux erreurs ; types scalaires ; typage des propriétés ; gestion du type null ; unions de types (ex. int|float) ; gestion des arguments nommés ; typage des constantes
- PHP devient plus strict et sécurisé avec chaque version, se rapprochant des langages fortement typés.
- **Bonne pratique** : adopter le typage pour un code plus maintenable, fiable et facile à comprendre.

# #16 – Xdebug

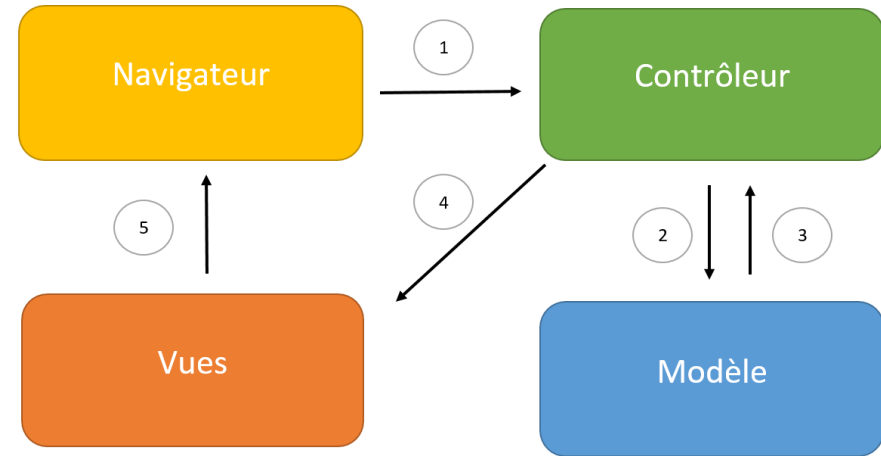
- Déboguer avec `var_dump` / `echo exit`; c'est bien
- Avec **Xdebug**, c'est mieux !
- Plus puissant et plus précis
  - Débogueur intégré avec breakpoints, évolution des variables, apparition des messages d'erreurs...
- Connectivité du poste développeur avec l'IBM i
- Mise en œuvre : <https://www.seidengroup.com/how-to-set-up-remote-debugging-over-ssh-using-xdebug-on-ibm-i/>





# #17 – Architecture MVC

- Adopter une architecture **MVC** (Modèle-Vue-Controller) permet :
  - Une séparation des responsabilités de chaque partie de l'application
    - Rend le code plus **lisible** et **organisé**, facilitant la maintenance du projet et l'intégration de nouveaux développeurs
  - Une bonne collaboration en équipe
    - Où chaque développeur peut se concentrer sur une partie :
      - Un développeur sur le **backend** (logique métier et modèle)
      - Un autre sur le **frontend** (vue et interface utilisateur)
  - Une évolutivité et réutilisabilité
    - Rend le code plus modulaire
    - Chaque partie peut être réutilisée dans différentes parties de l'application



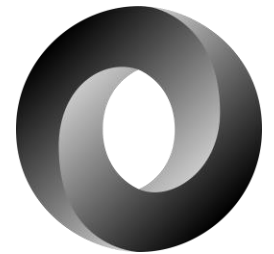
# #18 – Etudier les design patterns

- A la base de beaucoup de frameworks
- Permettent de structurer le code, le rendre plus flexible et plus maintenable.
- Exemples :
  - **Pattern Singleton** : Garantit qu'une classe n'a qu'une seule instance Utilisé généralement pour des composants nécessitant une instance unique dans l'application, comme une connexion à la base de données.
  - **Pattern Factory** : permet de créer des objets sans exposer le processus de création exact à l'utilisateur. Il fournit une **méthode pour instancier des objets** de différentes classes, sans avoir à spécifier la classe exacte.

```
class AnimalFactory {  
    public static function createAnimal(string $type): Animal {  
        return match ($type) {  
            'chien' => new Chien(),  
            'chat' => new Chat(),  
            default => throw new Exception("Type d'animal  
inconnu."),  
        };  
    }  
}
```

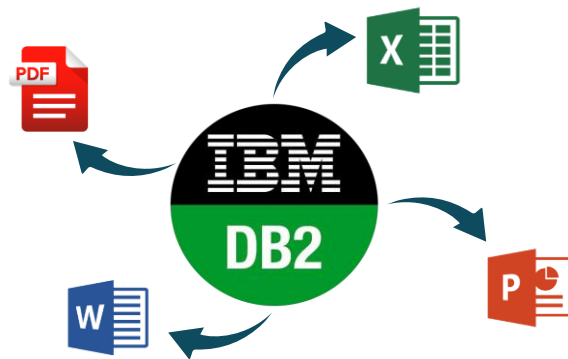
# #19 – Gestion des chaînes JSON

- Les objets et les tableaux PHP sont très souples et peuvent représenter des structures simples comme complexes
- Le **JSON** (*JavaScript Object Notation*) est fortement utilisé dans les échanges sur le web
- Deux fonctions très utiles permettent de passer d'un objet / tableau PHP à une chaîne JSON et inversement
  - **json\_encode** : tableau / objet -> JSON :  
<https://www.php.net/manual/en/function.json-encode.php>
  - **json\_decode** : JSON -> tableau / objet :  
<https://www.php.net/manual/en/function.json-decode.php>
- Validation json avec un schéma JSON :
  - <https://packagist.org/packages/justinrainbow/json-schema>



# #20 – Des fonctionnalités bureautiques

- PHP permet d'intégrer et de gérer facilement des projets intégrant de la bureautique :
  - Création, lecture, écriture, envoi de documents :
    - PDF : fpdf/fpdi, tcpdf, dompdf, mpdf ...
      - <https://packagist.org/packages/setasign/fpdf>
      - <https://packagist.org/packages/setasign/fpdi>
    - Excel : PHPSpreadsheet
      - <https://packagist.org/packages/phpoffice/phpspreadsheet>
    - Word et Power Point : PHPWord
      - <https://packagist.org/packages/phpoffice/phpword>
      - <https://packagist.org/packages/phpoffice/phppresentation>

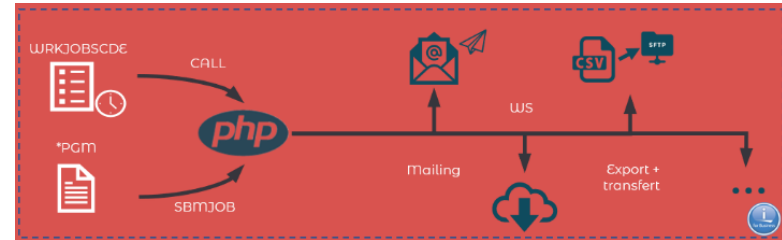


# #21 – Complet pour tous les projets Web Services

- Quoi de mieux qu'un langage web pour faire des Web Services ?
- PHP dispose de quoi implémenter les **consommations** et les **expositions** WS
  - Que ce soit du **SOAP** ou du **REST**
- Avec les possibilités :
  - De gérer la sécurité (par AuthBasic, Bearer Token, Token JWT ...)
  - D'accès DB2
    - Pour les données
    - Pour les logiques métiers
  - De définir les routes comme bon nous semble (en respectant les bonnes pratiques)
  - De gérer les messages échangés (JSON, XML, CSV ...)

## #22 – Pour les traitements batch

- PHP est tout à fait adapté pour des traitements batch ou asynchrones ne nécessitant pas d'IHM.
- Possible de planifier et/ou soumettre des exécutions de ces traitements !
  - QSH / QP2SHELL depuis des \*PGM puis à l'aide du WRKJOBSCDE
  - En ligne de commande php script\_a\_executer.php puis à l'aide de cronie
- De nombreux exemples sont aujourd'hui en production :
  - Envois de mails / sms
  - Edition de factures au format PDF
  - Mise à jour quotidienne des tarifs fournisseurs
  - Exports CSV avec dépose SFTP chez un partenaire



<https://www.seidengroup.com/php-documentation/calling-php-from-cl/>



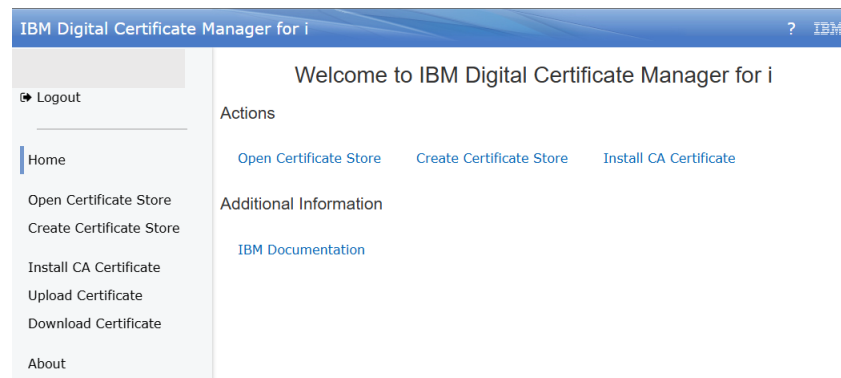
# Trucs et astuces d'administration

# #23 – HTTP / HTTPS



- Sécuriser vos flux TCP
  - Et notamment HTTP
- **DCM** (*Digital Certificate Manager*) est un outil intégré de l'IBM i permettant de gérer les certificats TLS (création, import, export, association à une application)

- Possible d'utiliser un certificat :
  - Auto-signé
  - Let's Encrypt
  - Acheté auprès d'un organisme tiers



- <https://www.ibm.com/docs/fr/i/7.5?topic=security-digital-certificate-manager>



# #24 – Optimisations Fastcgi Apache

- Optimiser l'instance Apache avec Fastcgi
  - /www/votre\_instance/conf/**fastcgi.conf**
- Ajustement du nombre de travaux pour accueillir plus de trafic (quand ça le nécessite)
  - Directive **PHP\_FCGI\_CHILDREN**
  - Valeur 10 par défaut
- Recyclage des travaux
  - Directive **PHP\_FCGI\_MAX\_REQUESTS**
  - Valeur 0 par défaut (pas de recyclage)
- Gestion des timeouts
  - Directives **ConnectionTimeout & RequestTimeout**
  - Valeurs 30 et 60 par défaut
- <https://www.seidengroup.com/2021/10/28/optimize-ibmi-web-application-fastcgi/>

# #25 – Intégrer des tests avec PHPUnit

- Les tests permettent
  - Une détection rapide des bogues
  - Une amélioration de la qualité de l'application
  - Une facilitation de la maintenance
  - D'obtenir une documentation vivante de l'application
  - D'éviter les regressions
- À intégrer dans un processus de CI / CD
  - On peut commencer par les jouer manuellement dans l'IDE du développeur
- Ressources PHPUnit
  - <https://cfd-innovation.fr/blog/optimiser-le-developpement-php-avec-phpunit-guide-complet-pour-une-premiere-integration-2/>
  - <https://cfd-innovation.fr/blog/les-tests-avec-phpunit/>
  - <https://packagist.org/packages/phpunit/phpunit>



# #26 – Intégrer les classes utilitaires comme dépendances

- A l'aide de Composer,
  - intégration de référentiels type **vcs**
- Pratique pour réutiliser :
  - Des classes utilitaires
  - Des classes métiers (Client, Commande, Facture...)

- Côté application (composer.json) :
  - déclaration d'un repository
  - Intégration du package souhaité

```
"repositories": [  
  {  
    "type": "vcs",  
    "url": "git@gitlab.com:cfid-innovation/toolkit.git"  
  }  
]  
}  
  
"require": {  
  "cfid-innovation/toolkit": "dev-master",  
},
```

- Côté librairie : déclaration du package dans composer.json

```
{  
  "name": "cfid-innovation/toolkit",  
  "description": "Boîte à outils CFD-Innovation",  
  "autoload" : {  
    "psr-4" : {  
      "cfid-innovation\\toolkit\\" : "/"  
    }  
  }  
}
```

# #27 – Gestion des logs

- Log PHP du CP+ dans
  - `/www/site_name_here/logs/php_error.log` si instance Apache créée par siteadd
  - Sinon : `/QOpenSys/var/log/php_error.log`
  - Possible de configurer son emplacement avec la directive `error_log` du `php.ini`
- Pour simplifier la gestion du log (et éviter les gros fichiers), mettre en place une rotation du log PHP
  - À l'aide d'un petit programme CL qui consiste à le renommer en l'horodatant à une fréquence régulière
  - À l'aide `logrotate` + `cronie` (deux packages rpm)
- Connecter les logs à un SIEM pour un suivi dans le temps
- **Surveiller les logs !** Même les NOTICES et attention aux **deprecated** : à remplacer dès que possible
- <https://www.seidengroup.com/php-documentation/how-to-configure-php-error-logging-on-ibm-i/>

# #28 – Gestion des logs par vhost

- Sur une partition de développement IBM i, il est utile de configurer un virtual host (vhost) par développeur
  - Pour garantir à chaque développeur un environnement indépendant
  - Pour avoir un fichier error\_log propre à chaque environnement par vhost

Template vhost (dans httpd.conf)

```
# Port développeur dev1  
Listen *:100xx
```

```
# vhost développeur dev1  
<VirtualHost *:100xx>
```

```
DocumentRoot "/www/PHPDEV/dev1"  
<Directory "/www/PHPDEV/dev1">  
SetEnv PHP_VALUE "user_ini.filename = .user.ini"  
<Files ".user.ini">  
    Require all denied  
</Files>  
    Options +Indexes +FollowSymLinks  
    DirectoryIndex index.php  
    Order allow,deny  
    Allow from all  
    AllowOverride All  
</Directory>
```

```
</VirtualHost>
```

.user.ini (à mettre à la racine du vhost)  
error\_log = /www/PHPDEV/logs/php\_error\_dev1.log

# #29 – Analyse de code statique

- Ces solutions permettent d'analyser du code pour en faire ressortir les erreurs sans même les exécuter
  - PHPCodesniffer (OSS)
    - [https://github.com/PHPCSStandards/PHP\\_CodeSniffer/](https://github.com/PHPCSStandards/PHP_CodeSniffer/)
  - Phpstan (Freemium)
    - <https://github.com/phpstan/phpstan>
- Pratique pour analyser du code legacy ou en prévision d'une montée de version
- Phploc : Pour avoir une idée de la taille d'un projet PHP
  - <https://github.com/sebastianbergmann/phploc>



## #30 – Qualité du code

- Intégration dans votre process CI/CD d'une **Quality Gate**
- Exemple avec **SonarQube** pour les apps PHP qui permet de faire ressortir :
  - Les failles de sécurité potentielles
  - La qualité du code
  - La duplication de code
  - Le taux de code coverage (par les tests)



MERCS

