

architecte
logiciel

Guillaume Ponçon
Ouvrage dirigé par Libero Maesano

Best practices PHP 5

Préface de Cyril Pierre de Geyer

EYROLLES

© Groupe Eyrolles, 2006,
ISBN : 2-212-11676-4

Table des matières

CHAPITRE 1

PHP est-il adapté à vos besoins ? 1

Qu'est-ce que PHP ?	2
À quoi sert PHP ?	2
À qui s'adresse PHP ?	3
Qui développe PHP ?	4
Politique d'évolution	5
Pourquoi PHP est-il libre ?	6
Quels sont les apports de PHP en termes de productivité ?	6
<i>Sa simplicité</i>	6
<i>Sa similitude avec d'autres plates-formes populaires (C, Java)</i>	6
<i>Son adaptation aux débutants</i>	6
<i>Sa souplesse</i>	7
Quelles sont les garanties d'utilisation de PHP ?	7
À qui dois-je m'adresser en cas de problème ?	7
Le parcours de PHP	8
<i>Naissance de PHP</i>	8
<i>PHP/FI</i>	8
<i>PHP/FI 2</i>	8
<i>PHP 3</i>	9
<i>PHP 4</i>	9
<i>PHP 5</i>	9
Un outil simple pour résoudre des problèmes complexes	10
Une plate-forme intuitive, facile à assimiler	10
<i>Installer PHP, Apache et MySQL en moins de 10 minutes !</i>	10
Une syntaxe souple, qui requiert de la rigueur	11
<i>Une nécessité : un minimum de connaissances en génie logiciel</i>	13
De nombreuses fonctionnalités qui en font une plate-forme communicante	14
Souplesse d'adéquation à la complexité du problème et à la compétence des équipes	14

Une architecture simple, fiable et portable	15
Architecture minimale : PHP CLI et GTK	16
Architecture web : PHP + HTTP	17
Architecture web avancée	17
Richesse fonctionnelle et possibilités de PHP	18
Que peut faire PHP ?	18
Une puissance et une simplicité inégalées pour les front-ends	18
Une puissance et une simplicité inégalées pour les traitements XML	19
Un traitement simple et unifié de trois métastructures : objets/documents/tables	19
Performances et montée en charge	20
Le noyau de PHP : Zend Engine II	20
Limite des performances	20
Peut-on optimiser les performances de PHP ?	20
Peut-on déployer des applications PHP sur plusieurs serveurs ?	21
Qualité de développement	21
Rigueur et élégance avant tout	21
Les méthodes agiles pour le développement d'applications PHP	22

PREMIÈRE PARTIE

Organisation du projet : conventions et outils 23

CHAPITRE 2

Définir des conventions pour la conception d'applications PHP ... 25

Organisation du projet : conventions et outils	26
Utilité des conventions	26
Faciliter la collaboration entre les différents intervenants du projet	26
Assurer la pérennité des développements et faciliter les opérations de mises à jour	26
Permettre la réalisation de projets professionnels ambitieux	26
<i>En exploitant le potentiel de la dernière version de PHP</i>	26
<i>En connaissant bien les possibilités offertes par les ressources disponibles</i>	27
<i>En adoptant une architecture simple et performante</i>	28
Simplifier et réduire la maintenance	29
<i>Maintenance logicielle : de la rigueur avant tout</i>	29
<i>Maintenance des données</i>	30
<i>Maintenance technique</i>	30
Assurer une stabilité et des performances optimales	30
<i>Mettre à disposition un document écrit</i>	31
<i>Coachez votre équipe</i>	31

Adaptation des conventions à la méthode de gestion de projets	32
Les méthodes agiles	32
<i>Les 5 valeurs essentielles des méthodes agiles</i>	33
<i>L'eXtreme Programming (XP)</i>	34
<i>Aperçu des pratiques de programmation proposées par XP</i>	35
<i>XP et le travail d'équipe</i>	36
<i>Gérer un projet avec XP</i>	38
MVC	39
<i>MVC en pratique</i>	40
Bâtir sa propre méthode	44
Les lois du succès d'une méthode nouvelle	44
<i>Simple et cohérente</i>	44
<i>Documentée</i>	44
<i>Adaptée et travaillée</i>	44
Domaines d'application d'une méthode	45
Conventions et procédures liées à l'organisation du développement	45
Répartition des rôles au sein de l'équipe	46
<i>Des rôles et des responsabilités</i>	48
<i>Maîtrise d'ouvrage et maîtrise d'œuvre</i>	49
Des procédures à mettre en place	50
Qui collabore avec qui ?	51
Conventions liées à l'écriture du code source	52
Règles élémentaires d'écriture	53
<i>Formatage d'un code source</i>	53
<i>Composition du formatage</i>	53
<i>Conventions de formatage courantes</i>	54
Erreurs courantes	57
Règles de nommage	58
<i>Choisissez une langue</i>	58
Nommage des versions	60
<i>Quelques rappels sur les principes du versionning</i>	60
<i>Nommage des versions</i>	62
CHAPITRE 3	
Installer et utiliser un gestionnaire de versions.....	65
La gestion des versions en PHP	66
Utilité d'un gestionnaire de versions	66
Principe de fonctionnement	66
<i>Exemple de scénario</i>	66
<i>Versions de fichiers et versions d'applications</i>	68

Règles de bonne conduite	69
Un dépôt ne supprime rien	69
<i>Mais pourquoi un dépôt ne supprime rien ?</i>	69
Ne valider que du code sans erreur	70
Tester avant de figer une version	70
Éviter les renommages et les déplacements en masse	71
Chacun son compte	72
Quand et pourquoi créer une branche ?	72
Subversion (SVN)	74
Apports de Subversion par rapport CVS	74
Installation	75
Création d'un dépôt de données	76
Configuration pour PHP avec Apache/WebDAV	76
Import de données	78
Opérations de base	78
Clients graphiques	78
Concurrent Version System (CVS)	79
Installation	79
Création d'un dépôt de données	80
Configuration du dépôt pour PHP	80
<i>Le fichier cvswrappers</i>	81
<i>Le fichier cvsignore</i>	81
Création de modules et import de données	82
Opérations de base	83
Utiliser les clés de substitution	83
Clients graphiques	84

CHAPITRE 4

Mettre en place l'environnement d'exécution

pour le développement 85

Qu'est-ce qu'un environnement d'exécution ?	86
Définition d'un environnement d'exécution pour PHP	86
<i>L'environnement minimal</i>	86
<i>Un environnement standard individuel</i>	87
<i>L'environnement standard en équipe</i>	88
<i>Un environnement agile complet</i>	89
Paramètres utiles d'un environnement d'exécution	90
<i>À la compilation</i>	90
<i>Après l'installation</i>	92
Le trio « développement/recette/production »	97

Apports et contraintes d'un environnement d'exécution	98
Des outils et des paramètres communs	98
Une simplification des développements en équipe	99
Un gain de temps en cas de crash	99
Une complexité relative	99
<i>Choix des outils</i>	100
<i>Stratégies d'installation</i>	100
<i>Un environnement au service des équipes projet</i>	101
Construire un environnement sur mesure	101
Architecture de l'environnement	101
<i>Déterminer les besoins</i>	102
<i>Sélectionner les outils</i>	102
<i>Évaluer les interactions entre les différents outils</i>	103
<i>Passer à l'acte</i>	104
Place du framework dans l'environnement	104
Outils utiles à l'installation de l'environnement	104
Intégrer l'environnement à l'intranet de l'entreprise	105
Développer des fonctionnalités pour l'environnement	106
Suivi planifié de la qualité/nightly build	107
À quoi servent les tâches planifiées ?	108
Contrôles automatisés et lancement des tests	108
Génération du rapport de qualité	109
Opérations de sauvegarde	110
<i>Exemple de procédure de sauvegarde</i>	110
Génération des archives	114
Tâches de maintenance	116
<i>Quelques tâches automatisées</i>	116
<i>Quelques tâches semi-automatisées</i>	117
Un environnement clé en main : la Zend Platform	117

CHAPITRE 5

Choisir un éditeur	119
Comment choisir un éditeur adapté à ses besoins ?	120
L'éditeur que l'on maîtrise	120
Un éditeur complet	121
Un éditeur adapté à la taille et à la nature des développements	123
Faut-il homogénéiser les outils d'édition ?	123
Éditeurs pour applications lourdes	124
Eclipse	124
Zend Studio	124

Maguma Open Studio/Maguma Studio	125
Éditeurs polyvalents	125
Komodo Professional	125
Anjuta	125
PHP Designer	126
Emacs	126
Dreamweaver	126
WebExpert	127
PHPEdit	127
PHPEd	127
UltraEdit	128
Crimson Editor	128
Quanta Plus	128
Éditeurs pour petites applications et travaux ponctuels	129
VIM	129
Side	129
Edit Plus	129
Scite	130
jEdit	130
Kate	130
gPHPEdit	130
Un test pour choisir son éditeur	131
Le questionnaire	131
Les réponses	132

CHAPITRE 6

Choisir les outils d'administration..... 135

Qu'est-ce qu'un outil d'administration ?	136
Simplifier les développements	137
Se débarrasser des tâches contraignantes	137
Éditeurs de bases de données	139
À quoi servent-ils ?	139
Éditeurs courants	139
<i>PhpMyAdmin</i>	139
<i>PhpPgAdmin</i>	140
<i>SQLiteManager</i>	140
<i>Et pour les autres SGBD</i>	141
Gestionnaires de sources de données (LDAP, Flux, etc.)	141
Utilité des éditeurs de sources de données	141
Éditeurs courants	142

Interfaces de débogage	143
Limites du débogage sans outil adéquat	143
Définir une stratégie de débogage globale	144
Utilisation d'outils existants pour le débogage d'applications PHP	144
Quelques configurations utiles pour le débogage	146
Monitoring du développement	146
Le rapport de qualité	146
Le rapport de performances	147
Les résultats des tests	147

CHAPITRE 7

Choisir les ressources et les supports de données 149

Les extensions C pour PHP	150
Qu'est-ce qu'une extension ?	150
Quand et pourquoi développer une extension en langage C ?	151
Choix d'un framework de développement	151
Utilité d'un framework	152
Choix d'un framework existant	153
Construire son propre framework	157
Utilisation de PEAR	158
Autres ressources (scripts et applications)	160
Comment choisir des ressources fiables ?	160
Note concernant les licences	161
Choix du SGBD	161
Qu'est-ce qu'un SGBD ?	161
MySQL	162
<i>Caractéristiques de MySQL</i>	162
<i>Pourquoi choisir MySQL ?</i>	163
PostgreSQL	164
<i>Caractéristiques de PostgreSQL</i>	164
<i>Pourquoi choisir PostgreSQL ?</i>	164
Oracle	164
<i>Caractéristiques d'Oracle</i>	164
<i>Pourquoi choisir Oracle ?</i>	165
SQLite	165
<i>Caractéristiques de SQLite</i>	165
<i>Pourquoi choisir SQLite ?</i>	165
Comparatif des SGBD supportés par PHP	166
Outils d'abstraction de bases de données	167
DBX	167

<i>PDO</i>	167
<i>ODBC</i>	168
Création du modèle de base de données	169
Modèle conceptuel de données (MCD)	169
Modèle physique de données (MPD)	170
Écriture des requêtes de création	170
Outils de design et de génération	172
Choix d'un format de données normalisé	173
XML	173
<i>XML et ses applications</i>	174
<i>Protocoles et applications basés sur XML</i>	175
LDAP	175
<i>Organisation des données avec LDAP</i>	175
<i>Schéma et classes LDAP</i>	176
Fichiers texte structurés (.inietc.)	176
Formats spécifiques (HTML, PDF, etc.)	178

DEUXIÈME PARTIE

Modélisation en UML pour PHP 179

CHAPITRE 8

Éléments de modélisation utiles à PHP 181

Les étapes de la modélisation	182
Trois axes de modélisation englobant différentes actions	182
Le sujet de nos exemples	183
L'analyse fonctionnelle	183
Expression des besoins, exigences et contraintes	183
<i>Quelques questions à se poser</i>	183
<i>Rédaction de l'expression des besoins</i>	183
Exigences et contraintes	184
Le diagramme des cas d'utilisation	184
<i>Identification des acteurs</i>	184
<i>Diagramme des cas d'utilisation</i>	184
Analyse technique statique	188
Les différents types de classes	189
<i>La classe métier</i>	189
<i>Le stéréotype</i>	189
L'identification des objets métier	190
Le diagramme de classes	190

Le diagramme de classes de conception	192
Analyse technique dynamique	195
Le diagramme de séquence	195
Le diagramme d'activités	195
Le diagramme de collaboration	198
Du modèle à l'implémentation	198
Utilisation d'un générateur de code	198
<i>Qu'est-ce qu'un générateur de code ?</i>	198
<i>UML2PHP5</i>	199
MDA : la voie du futur ?	200

CHAPITRE 9

Optimiser le modèle pour PHP	201
Pratiques de modélisation agile	202
Qu'est-ce que la modélisation agile ?	202
Modélisation agile pour PHP	202
Particularités et limites de la POO avec PHP 5	205
Fonctionnalités objet disponibles avec PHP 5	205
<i>L'auto-chargement de classes</i>	205
<i>La surcharge de propriétés et de méthodes</i>	206
Conseils d'usage pour améliorer la performance des objets	209
<i>L'instanciation d'une classe est-elle utile ?</i>	210
<i>Accélérer l'accès aux objets persistants</i>	210
Le « tout objet » n'est pas une bonne pratique pour PHP	212
S'adapter aux caractéristiques de PHP	213
Limitation du code à parser	213
Limitation des instanciations et appels	213
Exploiter les fonctions natives fournies par PHP	213
Favoriser l'interopérabilité et la pérennité du modèle	214
Les couches d'abstraction	214
<i>Avantages et inconvénients des couches d'abstraction</i>	215
Éviter d'encombrer les objets métier	215
Jouer avec la généricité	218
<i>Première étape : prévoir</i>	219
<i>Deuxième étape : une première évolution</i>	219
<i>Troisième étape : évolution et adaptation</i>	219
Adopter les standards et s'adapter à l'existant	219

CHAPITRE 10

Les motifs de conception (Design Patterns)..... 221

À quoi servent les motifs de conception ?	222
Les motifs de création	222
La fabrique (the Factory method)	222
<i>Principe de la fabrique</i>	223
<i>Mise en pratique</i>	224
La fabrique abstraite (Abstract Factory)	227
<i>Principe de la fabrique abstraite</i>	227
<i>Remarque sur l'utilisation de la fabrique abstraite avec PHP</i>	228
Le monteur (Builder)	228
<i>Principe du monteur</i>	228
<i>Mise en pratique</i>	228
Le prototype (Prototype)	229
<i>Principe du prototype</i>	229
<i>Le prototype en PHP</i>	229
Le singleton (Singleton)	229
<i>Principe du singleton</i>	229
<i>Mise en pratique</i>	231
Les motifs de structuration	232
L'adaptateur (Adapter)	232
<i>Principe de l'adaptateur</i>	233
MVC (Model View Controller)	233
Le pont (Bridge)	233
<i>Principe du pont</i>	233
<i>Mise en pratique</i>	234
Le composite (Composite)	234
<i>Principe du composite</i>	234
Le décorateur (Decorator)	235
<i>Principe du décorateur</i>	235
<i>Mise en pratique</i>	235
La façade	236
<i>Principe de la façade</i>	236
<i>Mise en pratique</i>	236
Le proxy (Proxy)	236
<i>Principe du proxy</i>	237
<i>Idée de mise en pratique</i>	237
Les motifs de comportement	238
La chaîne de responsabilité (Chain of responsibility)	238
<i>Principe de la chaîne de responsabilité</i>	238

<i>Mise en pratique</i>	238
La commande (Command)	239
<i>Principe de la commande</i>	239
<i>Mise en pratique</i>	239
<i>Utilisation</i>	239
L'itérateur (Iterator)	240
<i>Principe de l'itérateur</i>	240
<i>Utilisation avec PHP</i>	240
Le médiateur (Mediator)	242
<i>Principe du médiateur</i>	242
<i>Mise en pratique</i>	242
Le memento	242
<i>Principe du memento</i>	242
<i>Mise en pratique</i>	243
L'observateur (Observer)	243
<i>Principe de l'observateur</i>	243
<i>Mise en pratique</i>	243
L'état (State)	244
<i>Principe de l'état</i>	244
<i>Mise en pratique</i>	244
La stratégie (Strategy)	245
Le patron de méthode (Template of Method)	245
<i>Principe du patron de méthode</i>	245
<i>Mise en pratique</i>	246

TROISIÈME PARTIE

Bonnes pratiques de développement en PHP249

CHAPITRE 11

Exploiter les points forts de PHP : les méta-structures 251

Les trois méta-structures de base	252
Les tableaux	253
Quand et comment utiliser des tableaux ?	253
Exploiter la souplesse et la simplicité des tableaux PHP	253
<i>Convertir des données en tableau</i>	253
<i>Exploiter la souplesse des tableaux</i>	254
Opérations utiles pour manipuler des tableaux	255
Les documents XML	255
Quand et comment utiliser des documents XML ?	255

Concevoir et manipuler des documents XML avec PHP	257
SimpleXML	258
SAX	259
DOM	260
XSLT	261
Les objets	264
Qu'est-ce qu'un objet ?	264
Quand et comment utiliser des objets ?	265
<i>La logique métier</i>	265
<i>Les objets « contrôles »</i>	267
<i>Les fonctionnalités similaires</i>	267
Concevoir des objets performants	268
<i>Spécificité des objets PHP</i>	268
Concevoir des objets propres et réutilisables	269
<i>La mauvaise solution</i>	269
<i>La bonne solution</i>	269
<i>Bonnes pratiques de développement des objets</i>	270
<i>Pratiques à bannir</i>	271
Concevoir une bibliothèque d'objets homogènes	272
Utilisation avancée des classes pour PHP	274
Passer d'une méta-structure à une autre	274
Des objets aux documents XML	274
<i>Utilité</i>	274
<i>Outils existants</i>	275
<i>Implémentations possibles</i>	275
Des objets aux tableaux	277
<i>Utilité</i>	277
<i>Outils existants</i>	277
<i>Implémentations possibles</i>	278
Des documents XML aux tableaux	278
<i>Utilité</i>	278
<i>Outils existants</i>	279
<i>Implémentations possibles</i>	279
Des documents XML aux objets	280
<i>Utilité</i>	280
<i>Outils existants</i>	280
<i>Implémentations possibles</i>	281
Des tableaux aux objets	281
<i>Utilité</i>	281
<i>Outils existants</i>	281

<i>Implémentations possibles</i>	282
Des tableaux aux documents XML	283
<i>Utilité</i>	283
<i>Outils existants</i>	283
<i>Implémentations possibles</i>	284

CHAPITRE 12

Assurer la qualité d'un développement PHP..... 285

Réflexes simples d'optimisation	286
Ménager l'utilisation de la mémoire	286
<i>include, require</i>	286
<i>Passage par valeur ou par référence ?</i>	287
<i>Exploiter les mécanismes de mémoire partagée</i>	288
<i>Maîtriser la récursivité</i>	290
Ménager l'utilisation des ressources	291
<i>Écriture sur disque</i>	291
<i>Utilisation des expressions régulières</i>	293
<i>Utilisation de la bande passante</i>	294
<i>Utilisation des boucles</i>	295
<i>Manipulation correcte des chaînes de caractères</i>	297
<i>Autres trucs et astuces en vrac</i>	298
Exploiter les exceptions	300
Déboguer et tester	301
Débogage d'applications PHP	301
<i>Déboguer avec un outil personnalisé</i>	302
<i>Un lien qui ouvre l'éditeur sur le bon fichier à la bonne ligne !</i>	302
Outils de débogage pour PHP	308
<i>APD</i>	308
<i>Xdebug</i>	311
<i>KCacheGrind, WinCacheGrind</i>	312
Élaborer des tests unitaires	314
<i>Installation de l'espace de travail</i>	314
<i>Commençons par prendre de bonnes habitudes</i>	314

CHAPITRE 13

Simplifier et pérenniser un développement PHP 321

Commenter, documenter	322
Les secrets du bon commentaire	322
<i>10 astuces pour bâcler vos commentaires à coup sûr !</i>	322
<i>10 astuces pour améliorer vos commentaires</i>	324

Utilisation d'un générateur de documentation	326
Utilisation de PHPDocumentor	327
Pratiquer le remaniement (refactoring)	328
Qu'est-ce que le remaniement ?	328
Planifier le remaniement	329
Le remaniement en action	330
Exemples	331
<i>Extraction d'une condition</i>	331
<i>Optimisation des performances</i>	332
Utiliser des templates	332
Qu'est-ce qu'un moteur de templates ?	332
Utilité d'un moteur de templates	332
Utilité d'un compilateur de templates	333
Choix d'un moteur/compilateur de templates	335
<i>Quelques critères à considérer dans le choix de votre moteur de templates</i>	335
Exemples d'utilisation avec Smarty	337
<i>Classe d'initialisation</i>	338
<i>Appel du moteur de templates dans un code source PHP</i>	339
<i>Création d'un template Smarty</i>	339
Utilisation de PHP comme moteur de templates	340
Contraintes liées aux moteurs de templates	341

CHAPITRE 14

Assurer des développements PHP performants et polyvalents... 343

Interactions avec d'autres plates-formes	344
Possibilités offertes par PHP	344
Couplage fort	344
<i>Faire interagir PHP et Java</i>	344
<i>Faire interagir PHP et C/C++</i>	346
Couplage lâche	350
Services web	353
Principe et utilité	353
Choisir une solution d'interopérabilité	354
SOAP : un standard polyvalent	355
REST : une solution simple et performante	357
XML-RPC : une autre alternative	359
Génération de code	360
À quoi sert la génération de code ?	360
<i>Que peut faire la génération de code ?</i>	360
Accélérer et optimiser des développements	361

<i>Un « générateur d'application »</i>	361
<i>La régénération automatique partielle</i>	363
<i>Table de traduction</i>	363
Interagir avec d'autres plates-formes	366
Exemples et idées de générateurs PHP	366
<i>Générer des tests unitaires</i>	366
<i>Génération de couches d'accès à la base de données</i>	367
<i>Génération d'interfaces utilisateur</i>	367
<i>Couches de services web</i>	368
<i>Générer la logique métier</i>	368
Limites et dangers de la génération de code	368
Mise en cache	369
Mise en cache sur mesure	369
Utilisation d'un outil existant	372
Mise en cache « haut niveau » via serveur proxy	373
Mise en cache à plusieurs niveaux	373
Mise en cache bas niveau du code compilé	375
Mise en cache mémoire des opcodes et des données	376
<i>Mise en pratique avec APC</i>	376

QUATRIÈME PARTIE

Définition des exigences pour l'exploitation379

CHAPITRE 15

L'environnement d'exécution 381

S'adapter aux caractéristiques de l'environnement d'exécution	382
Maîtriser les caractéristiques de l'environnement	382
Le serveur HTTP	383
<i>Les variables d'environnement</i>	383
<i>Les modules et options du serveur</i>	383
<i>Le serveur et sa version</i>	384
<i>La configuration du serveur</i>	386
La plate-forme PHP	387
<i>La version de PHP</i>	388
<i>PHP et ses modules</i>	388
<i>Configuration de PHP</i>	389
<i>Utilisation de PHP avec un optimiseur</i>	392
Installations et mises à jour	392
<i>Procédures définies avec l'administrateur système</i>	392

<i>Packaging des applications</i>	393
<i>Automatismes mis en place</i>	394
Caractéristiques d'exploitation	394
Le système d'exploitation	395
L'environnement applicatif du système d'exploitation	395
Installations avec compilation sur mesure	396
Pourquoi compiler sur mesure ?	396
Compilation du serveur HTTP	396
<i>Récupération des sources et préparation de la compilation</i>	397
<i>Compilation et installation</i>	398
<i>Configuration et lancement d'Apache</i>	399
<i>Procédure de mise à jour</i>	400
Compilation de PHP et de ses extensions	401
<i>Installation en module dynamique</i>	401
<i>Installation en module statique</i>	403
<i>Configuration d'Apache pour PHP</i>	404

CHAPITRE 16

Assurer la disponibilité : sécurité et maintenance 405

Assurer la sécurité de l'environnement d'exécution	406
Installation, configuration et maintenance du serveur de production	406
<i>Sécuriser un serveur</i>	407
<i>Prévoir tous les cas de catastrophes possibles</i>	409
Mise à jour des routines de sauvegarde	414
<i>Exemple de routine de sauvegarde</i>	414
<i>Fréquence de sauvegarde</i>	415
<i>Archivage des sauvegardes</i>	415
<i>Quelques outils utiles</i>	416
Générer des rapports d'incidents	416
<i>Prévoir et surveiller les incidents possibles</i>	416
<i>Outils de monitoring</i>	417
<i>Centraliser et gérer les incidents</i>	417
Mettre en place le mécanisme de surveillance	420
Surveillance du système, des serveurs et du réseau	420
<i>Les outils disponibles sur Internet</i>	420
<i>Ressources utiles à surveiller</i>	420
<i>Créer soi-même un réseau de tests pour le monitoring</i>	422
Surveillance applicative	422

CHAPITRE 17

Exploiter un environnement d'exécution clé en main..... 425

La Zend Platform comme environnement pour la production	426
À qui s'adresse la Zend Platform ?	427
Avantages et inconvénients de la Zend Platform	428
<i>Performances et qualité</i>	428
<i>Une interaction native avec Java</i>	430
<i>Répartition de charge et clustering</i>	432
<i>Une solution Zend Exclusive</i>	433
Installation/paramétrage de la Zend Platform	433
Mise en place d'une application sur la Zend Platform	435
Avenir de la Zend Platform et de ses dérivés	435

CINQUIÈME PARTIE

Témoignages437

CHAPITRE 18

Témoignages d'utilisateurs 439

Zeev Suraski, directeur technique de Zend Technologies	440
<i>Pouvez-vous vous présenter ?</i>	440
<i>Pouvez-vous nous expliquer en deux mots votre parcours avec PHP ?</i> ...	440
<i>Quels sont selon vous les trois avantages qu'ont les professionnels</i> <i>à utiliser PHP ?</i>	440
<i>Quels sont au contraire les trois points faibles que PHP devrait améliorer ?</i> ...	441
<i>Quelles sont selon vous les qualités requises pour être un bon</i> <i>développeur PHP ?</i>	441
<i>Quelles sont les principales erreurs que les développeurs PHP font ?</i> ...	441
<i>Pouvez-vous nous présenter Zend Technologies et son rôle vis-à-vis</i> <i>de l'entrée de PHP dans le monde professionnel ?</i>	442
Gérald Croës, consultant chez Aston	443
<i>Pouvez-vous vous présenter ?</i>	443
<i>Quel a été votre parcours avec PHP ?</i>	443
<i>Pouvez-vous nous décrire le projet de framework Copix</i> <i>dont vous êtes l'auteur ?</i>	443
<i>Quels sont selon vous les trois avantages de PHP pour les professionnels ?</i> .	444
<i>Quels sont à l'inverse les trois points faibles que PHP devrait améliorer ?</i> .	444
<i>Quelles sont selon vous les qualités requises pour faire du bon travail en PHP ?</i> .	444
<i>Et quelles sont les principales erreurs que font les développeurs PHP ?</i> ...	445
<i>Quelle ont été votre meilleure et votre plus mauvaise expérience avec PHP ?</i> ...	445

Perrick Penet, responsable de la société No Parking	446
<i>Pouvez-vous nous parler de la méthode eXtreme Programming que vous pratiquez ?</i>	447
<i>Que pensez-vous de l'utilisation d'un framework avec XP ?</i>	448
<i>Que conseillez-vous aux développeurs pour apprendre le PHP ?</i>	448
Romain Bourdon, gérant de la société Kaptive	449
<i>En quoi consiste votre projet ?</i>	449
<i>Quelle valeur ajoutée apportent les choix technologiques de votre solution ?</i>	449
<i>Quelles ont été les trois difficultés majeures rencontrées dans ce projet ?</i>	
<i>Comment avez-vous réagi ?</i>	449
<i>Avez-vous adopté une méthode de gestion de projet ?</i>	450
<i>Quels outils avez-vous utilisé pour ce projet ?</i>	450
<i>Que retenir-vous de cette expérience ?</i>	450
Matthieu Mary, ingénieur de développement chez Travelsoft	451
<i>Pouvez-vous vous présenter ?</i>	451
<i>En quoi consistait votre dernier projet professionnel développé en PHP ?</i> ..	451
<i>Quelles difficultés avez-vous rencontrées lors de ce développement ?</i>	451
<i>Qu'est-ce que ce développement vous a apporté de positif ?</i>	451
<i>Que vous a apporté PHP par rapport à d'autres technologies ?</i>	451
<i>Quelles sont selon vous les qualités requises pour être un bon développeur PHP ?</i> ..	452
Bibliographie	453
Index	455