

Bonnes pratiques

PHP



Plan de la présentation

0

Comment aborder cette présentation ?

Développement

Exploitation

1

S'organiser et choisir ses outils

2

Gagner performances et fiabilité

3

Améliorer sa compétitivité

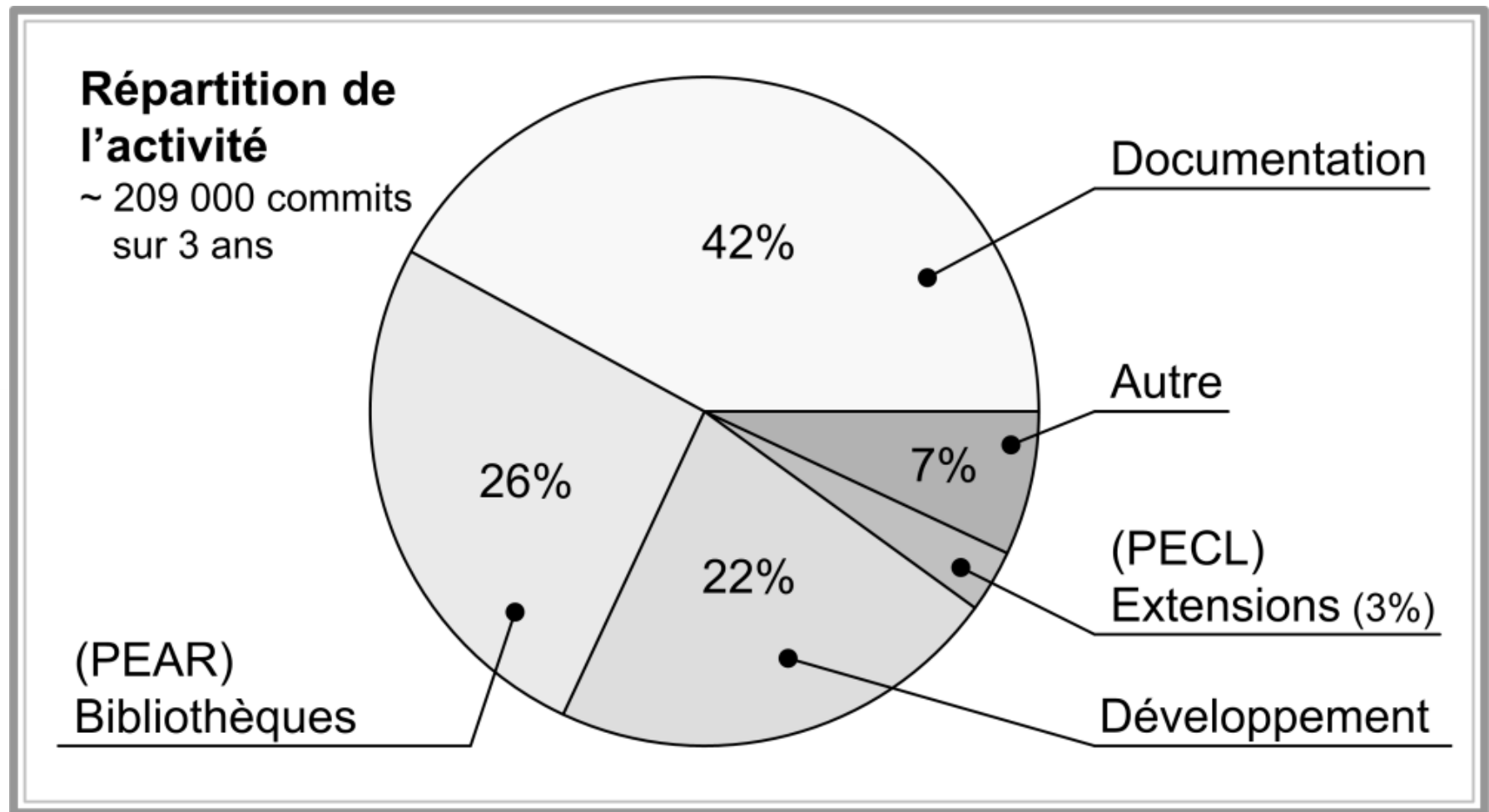
4

Ressources / Questions



Comment aborder cette présentation ?

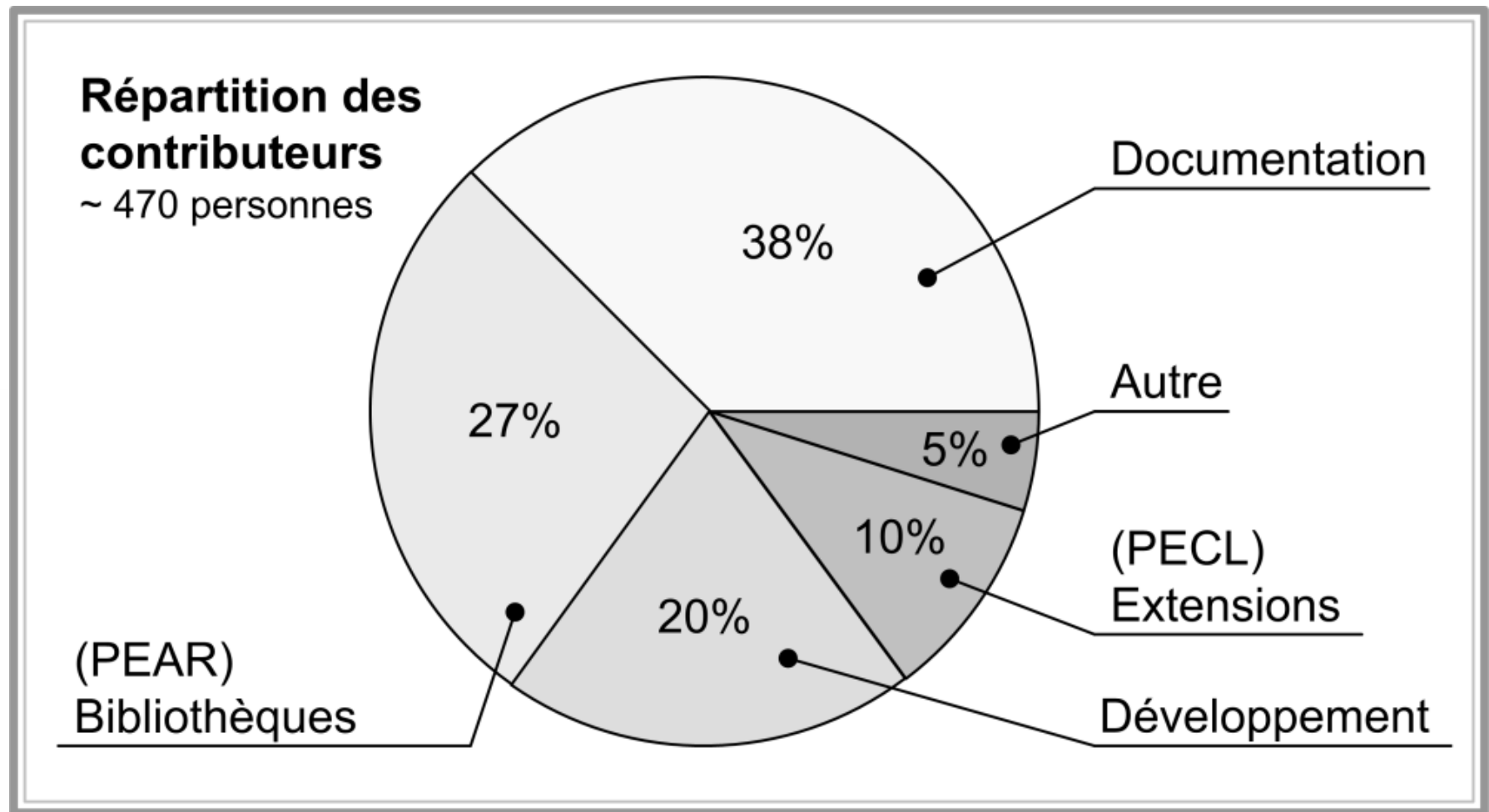
PHP : champion de la documentation et des ressources !



© 2005 - Best practices PHP 5 - Editions Eyrolles

Comment aborder cette présentation ?

PHP : champion de la documentation et des ressources !

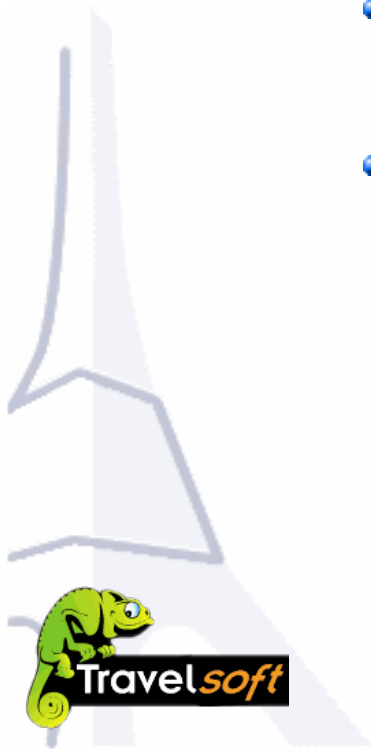


© 2005 - Best practices PHP 5 - Editions Eyrolles

S'organiser et choisir ses outils

pour développer

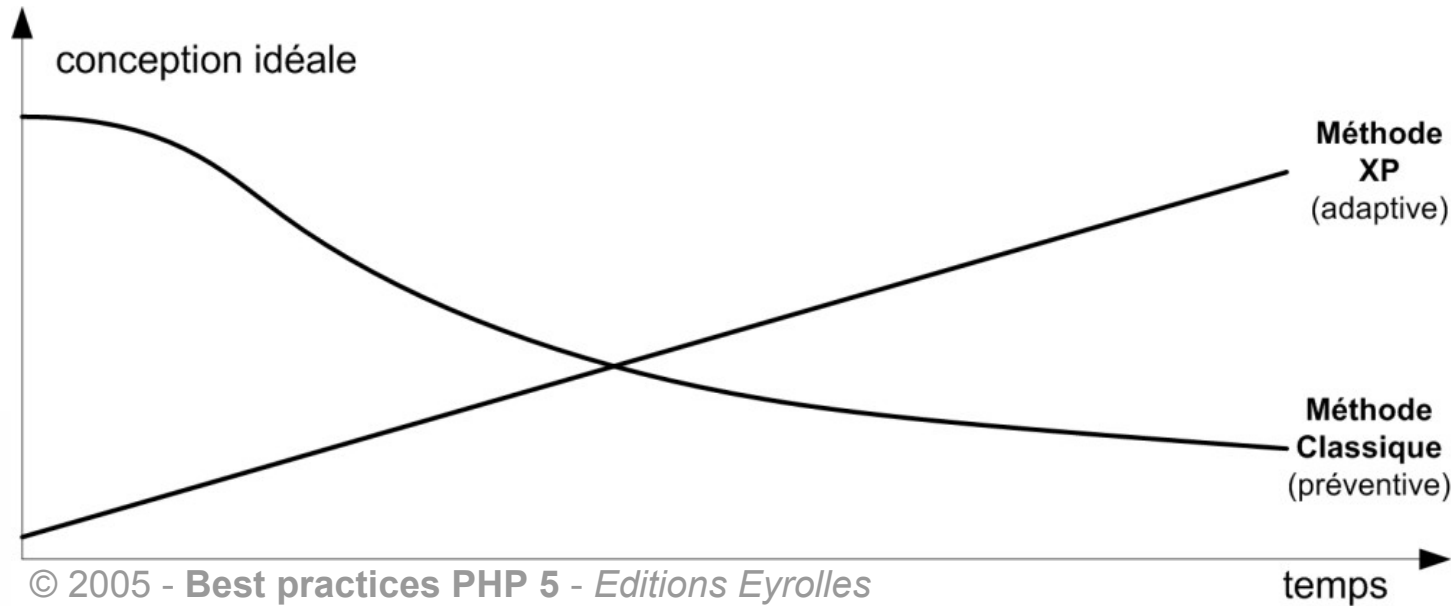
- **Votre environnement humain**
- **Votre éditeur**
- **Ressources / framework**



S'organiser et choisir ses outils

pour développer : **votre environnement humain**

- Méthodes Agiles : privilégier l'adaptation à la prévision



Extreme Programming

Crystal

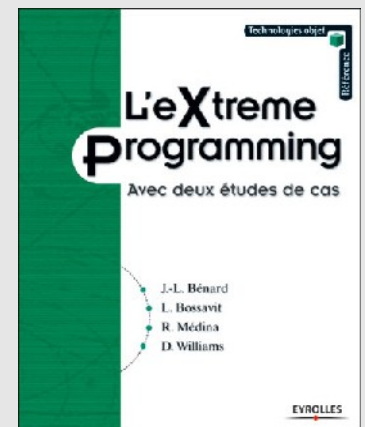
Scrum

...

Lien

<http://www.xp-france.net/>

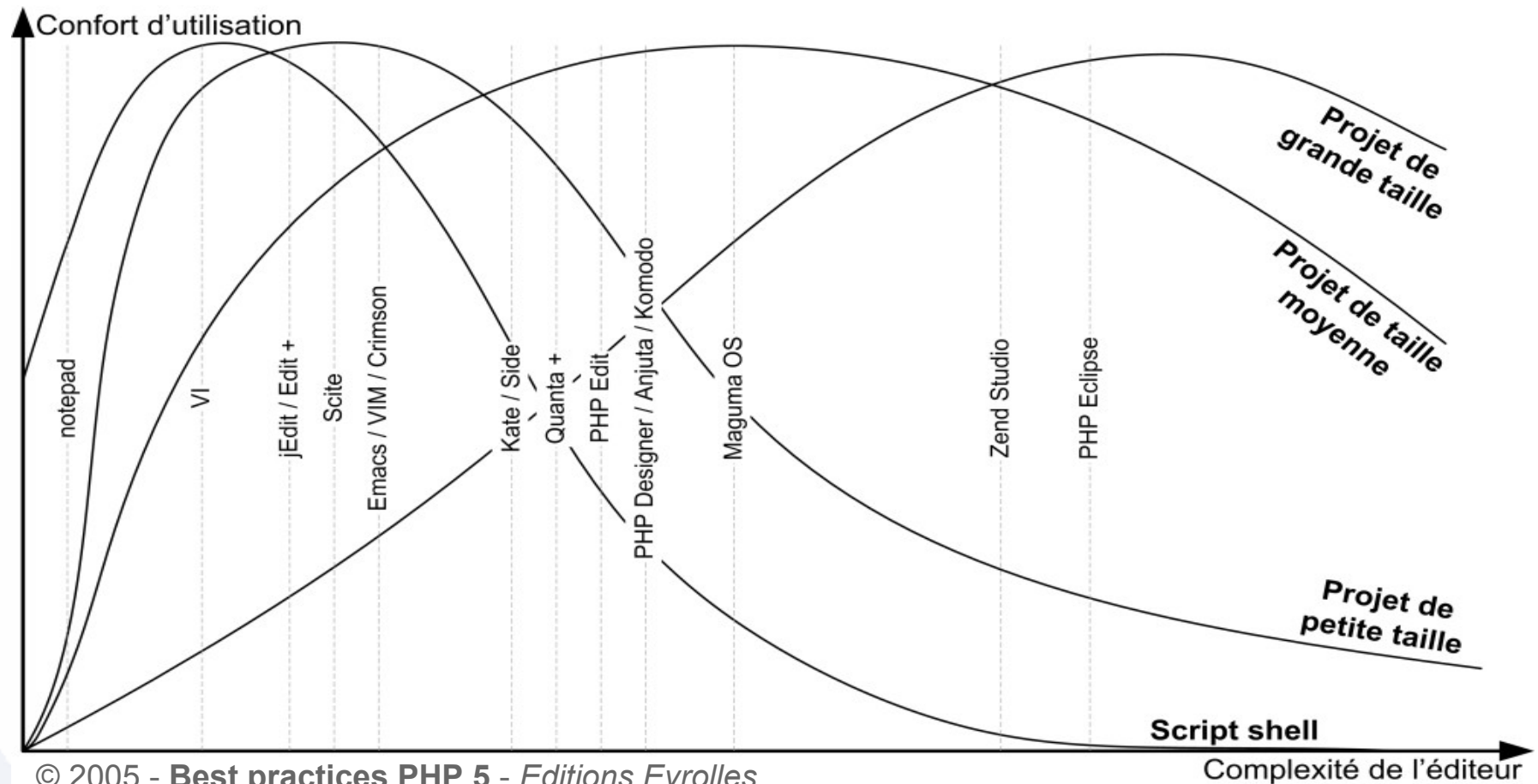
Livres



S'organiser et choisir ses outils

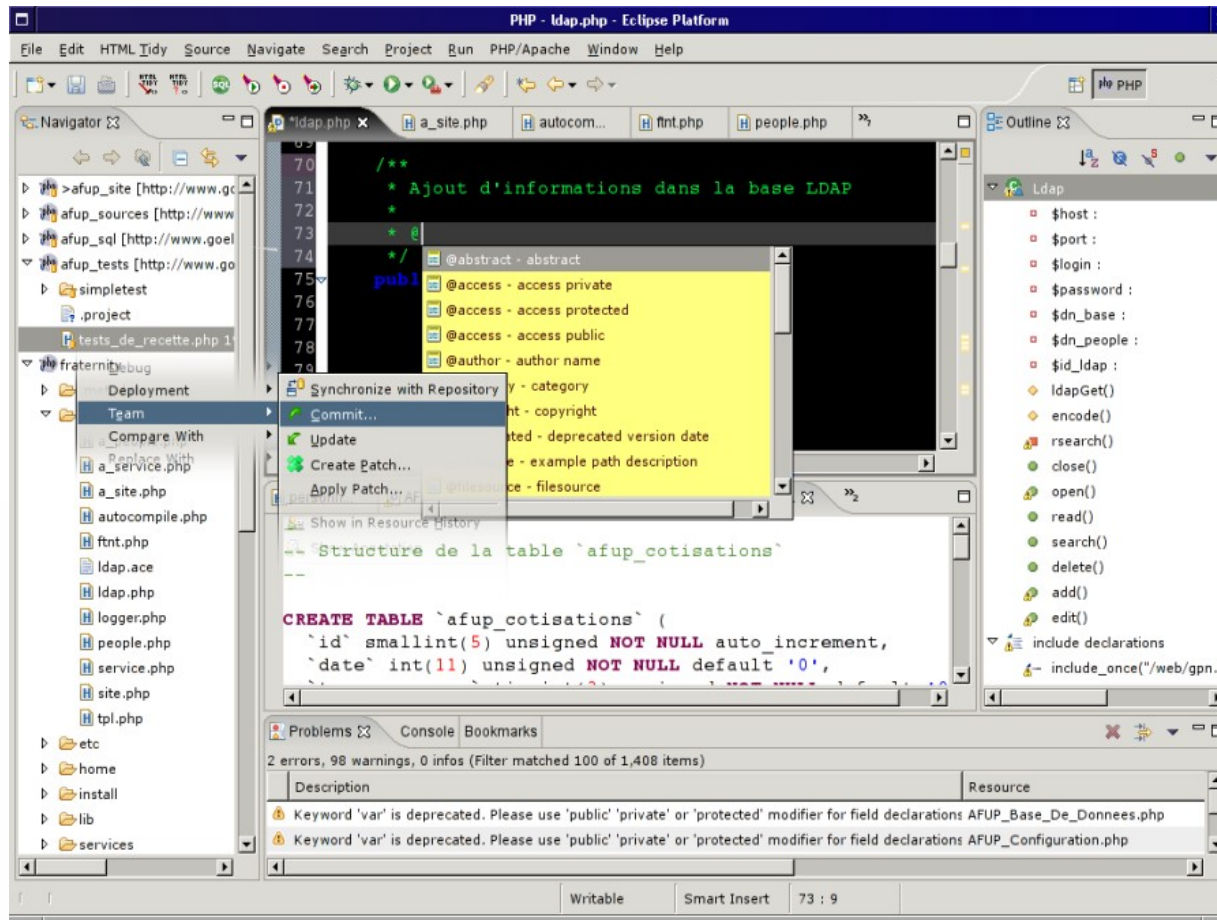
votre éditeur

- Il existe une trentaine de bons éditeurs PHP
- Critères de choix : votre besoin, vos habitudes, votre environnement, ...



S'organiser et choisir ses outils

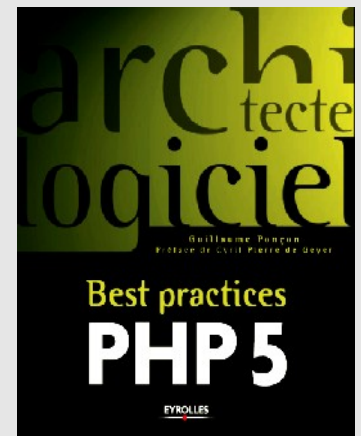
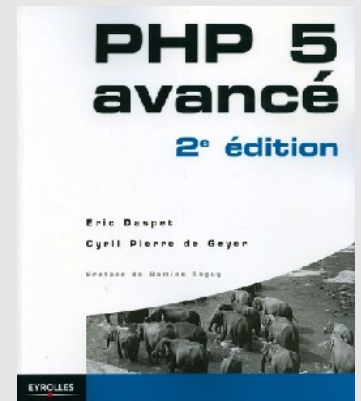
votre éditeur



Lien

<http://php.openstates.org/editeurs>

Livres



S'organiser et choisir ses outils

ressources / framework

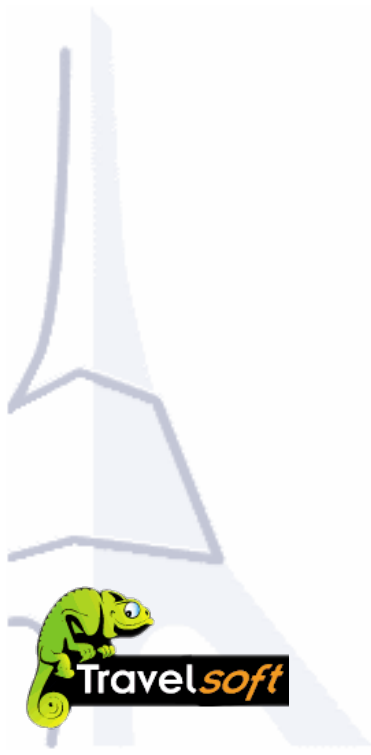
- PHP possède énormément de ressources, librairies, extensions
- Quelques critères de choix :
 - La popularité de la ressource
 - La capacité à évoluer
 - La notoriété / motivation des auteurs
- Types de ressources :
 - **Les applications** : fonctionnalités toutes faites (PhpBB, phpMyAdmin, ...)
 - **Les librairies** : fonctionnalités PHP à intégrer à un existant (PEAR, ...)
 - **Les extensions** : fonctionnalités développées en C (PECL, ...)



S'organiser et choisir ses outils

pour l'exploitation

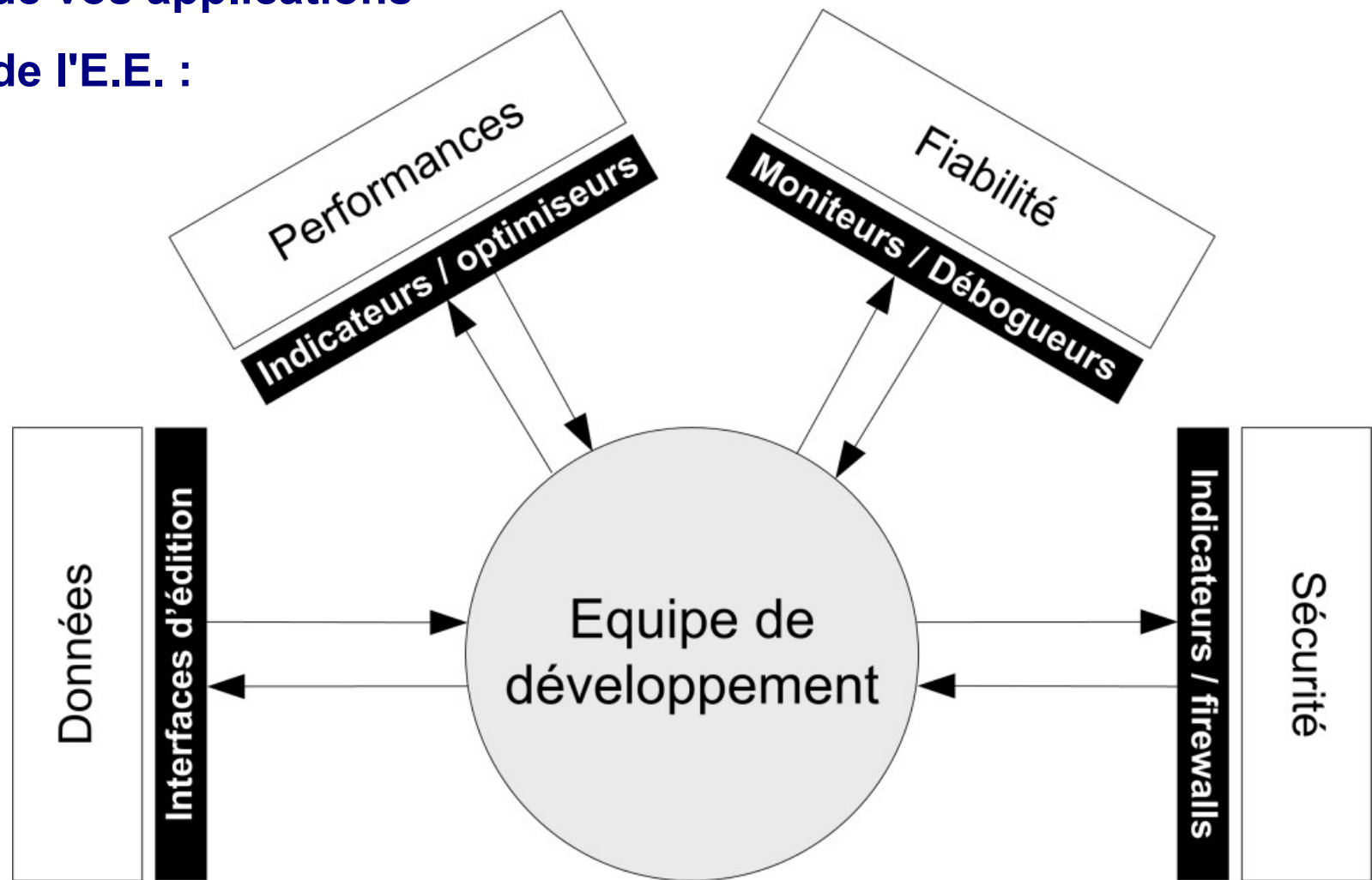
- **L'environnement d'exécution**
- **E.E. pour le développement**
- **E.E. pour la production**



S'organiser et choisir ses outils

l'environnement d'exécution

- Le support de vos applications
- Dépendent de l'E.E. :

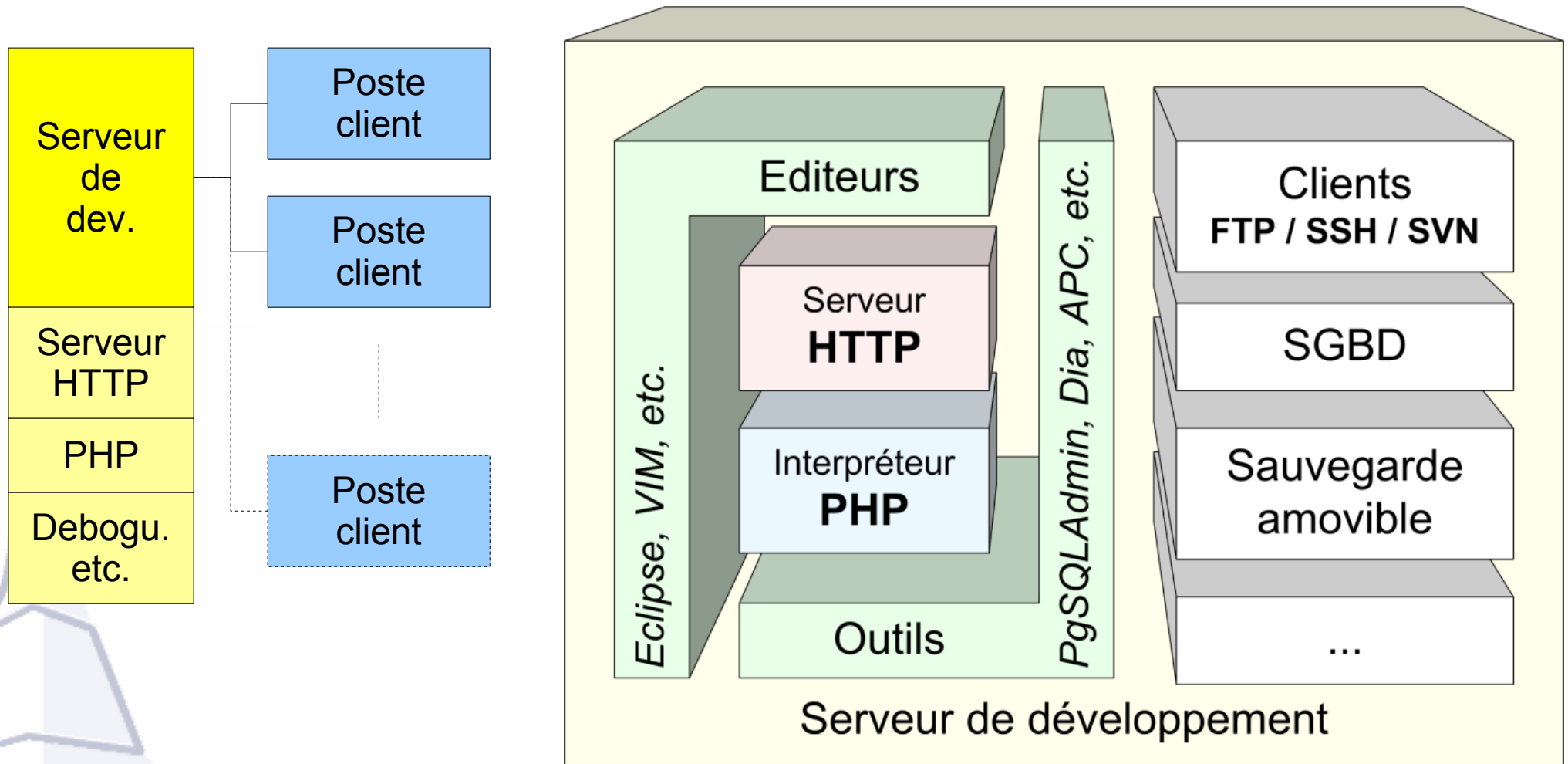


© 2005 - Best practices PHP 5 - Editions Eyrolles

S'organiser et choisir ses outils

environnement d'exécution pour le développement

- Optimisé pour la modélisation, l'édition, le débogage, le travail en équipe.

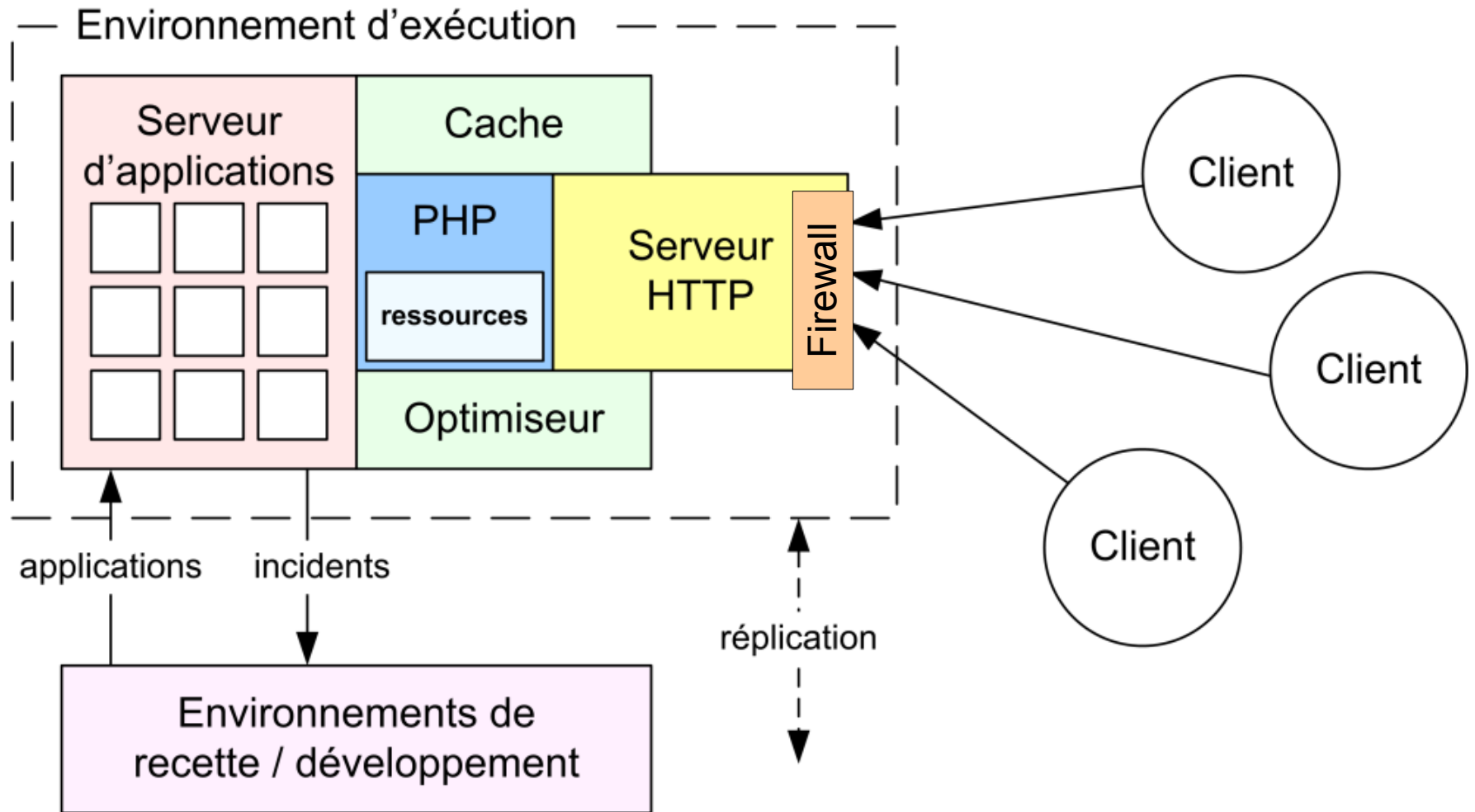


© 2005 - Best practices PHP 5 - Editions Eyrolles

S'organiser et choisir ses outils

environnement d'exécution pour la production

- Optimisé pour les performances et la sécurité.

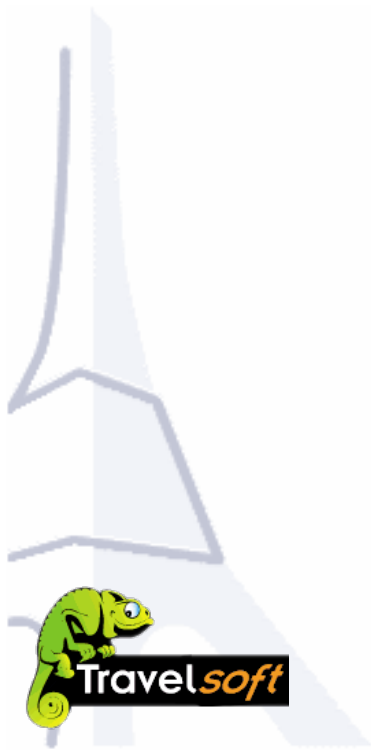


© 2005 - Best practices PHP 5 - Editions Eyrolles

Gagner performances et fiabilité

dans vos développements

- **Le cas « PHP »**
- **Réflexes de base**
- **Utilisation d'un débogueur**
- **Tests unitaires et tests de régression**



Gagner performances et fiabilité

dans le développement : **cas « PHP »**

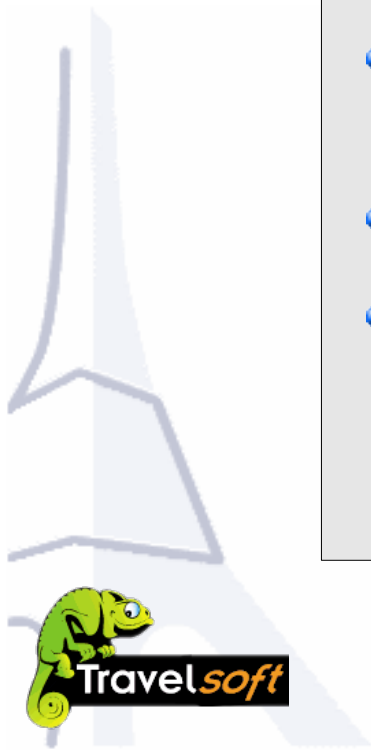
- Simplicité, souplesse, permissivité, faible typage, compilation à la volée, ...

Avantages

- Développements très rapides
- Maintenance facilitée
- Obtention immédiate d'un résultat
- ROI imbattable
- Adaptation à toutes les habitudes de développement

Inconvénients

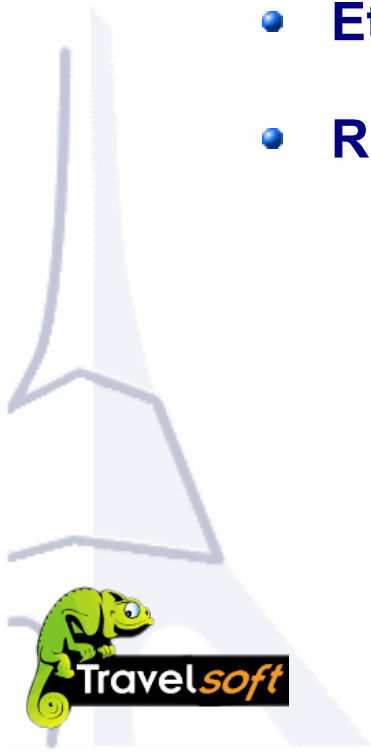
- Nécessite beaucoup de rigueur pour développer des applications professionnelles
- Les développeurs PHP ayant une véritable culture du génie logiciel sont rares
- Aucune architecture type pour développer de grosses applications



Gagner performances et fiabilité

réflexes de base

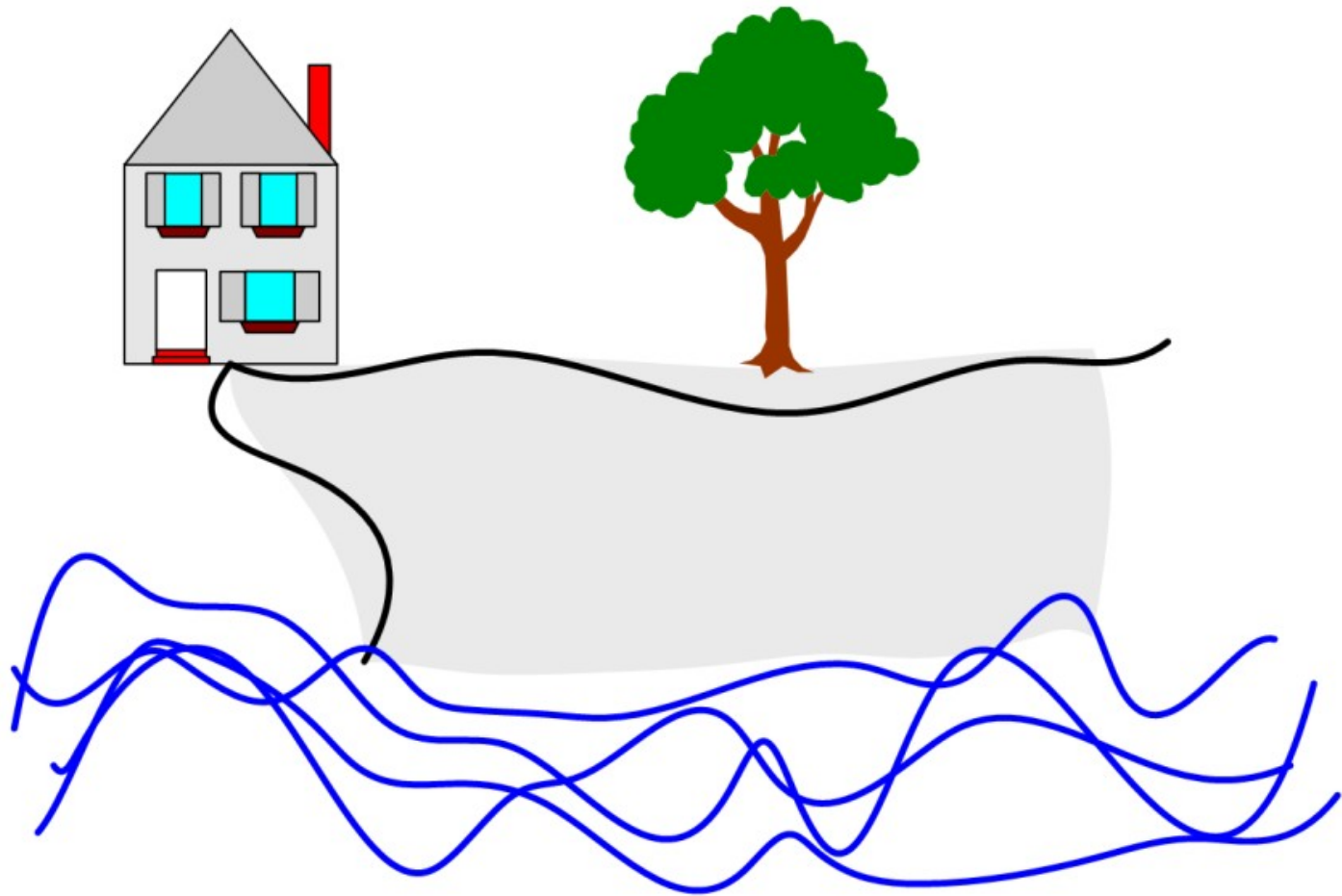
- **Respecter des conventions** (codage, documentation, etc.)
- **Ménager la mémoire et les ressources avec un débogueur**
- **Être rigoureux !** (déclarer les variables, respecter le modèle, etc.)
- **Réutiliser !** (ne pas recoder une portion de code pour la nième fois)



Gagner performances et fiabilité

utilisation d'un débogueur

- Sans débogueur...

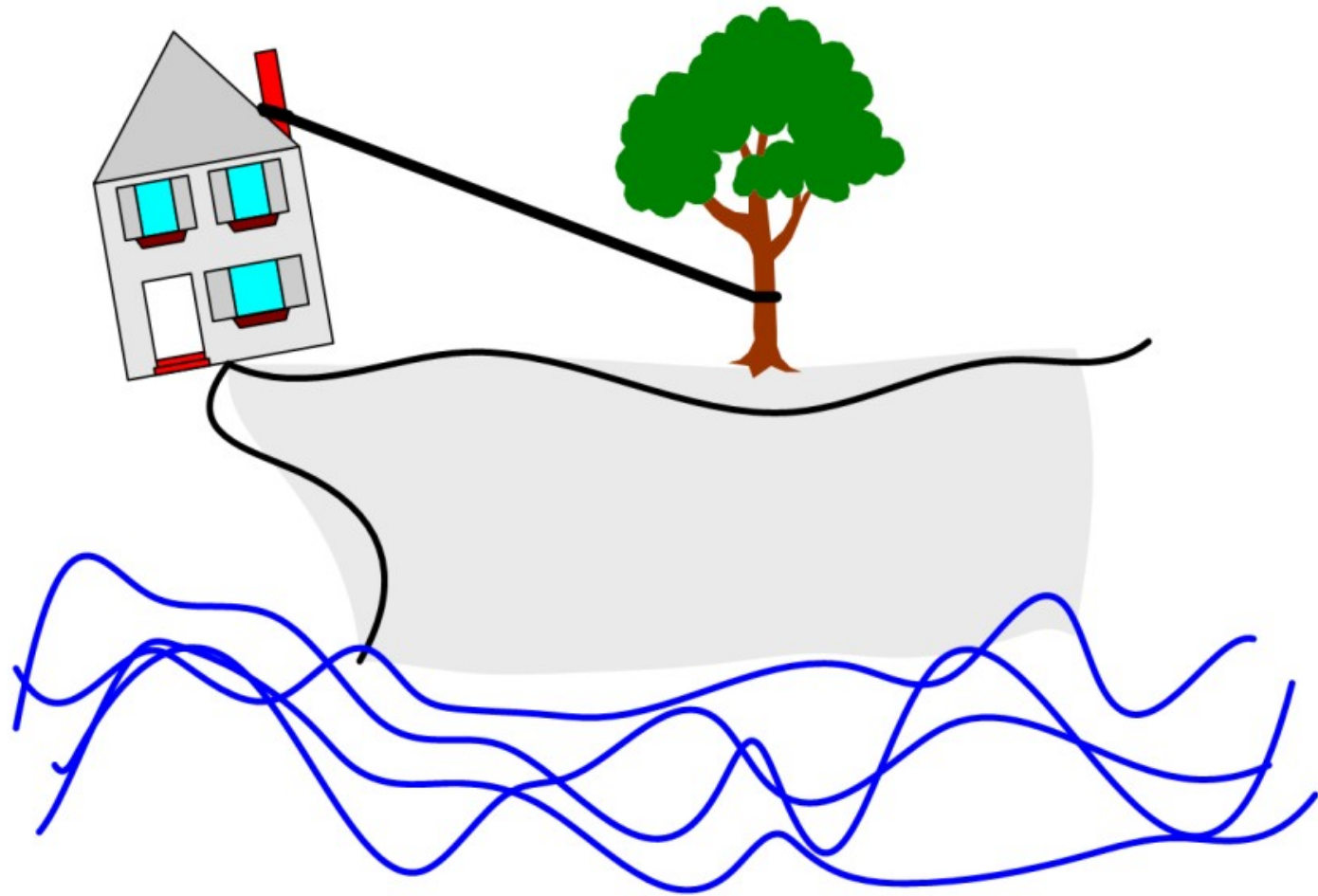


© 2005 - Best practices PHP 5 - Editions Eyrolles

Gagner performances et fiabilité

utilisation d'un débogueur

- Sans débogueur...

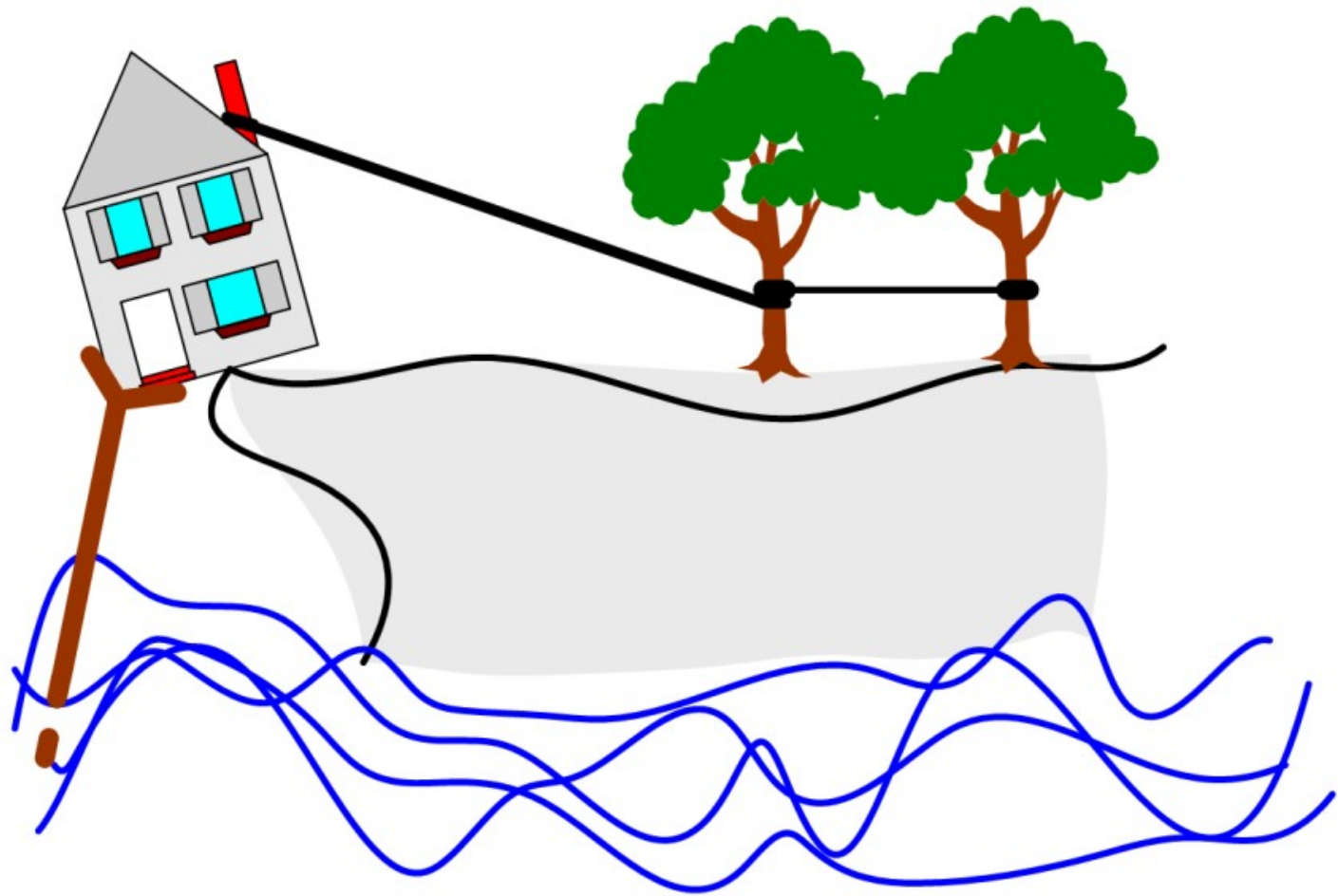


© 2005 - Best practices PHP 5 - Editions Eyrolles

Gagner performances et fiabilité

utilisation d'un débogueur

- Sans débogueur...



© 2005 - Best practices PHP 5 - Editions Eyrolles

Gagner performances et fiabilité

utilisation d'un débogueur

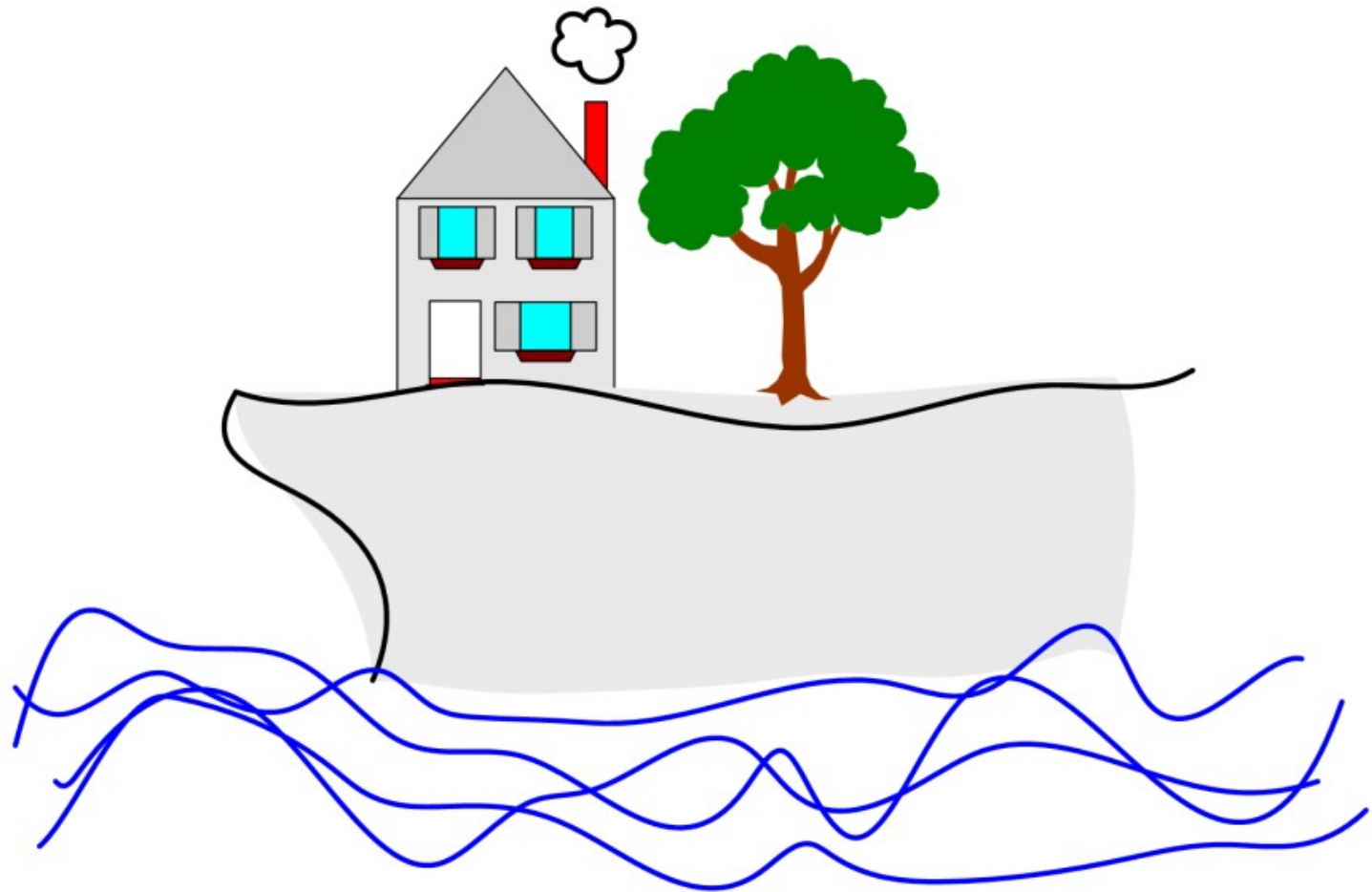
- **Apports d'un débogueur :**
 - **Maîtrise des consommations de mémoire**
 - **Visualisation de la pile des appels de fichiers / classes / fonctions**
 - **Visualisation des durées d'exécution des appels élémentaires**
 - **Amélioration du confort de gestion des erreurs**
- **Quelques débogueurs PHP : APD, Xdebug, DBG**
 - **APD (Advanced PHP Debugger)** [<http://pecl.php.net/package/apd>]
 - **Xdebug** [<http://xdebug.org/>]
 - **DBG** [<http://dd.cron.ru/dbg/>]



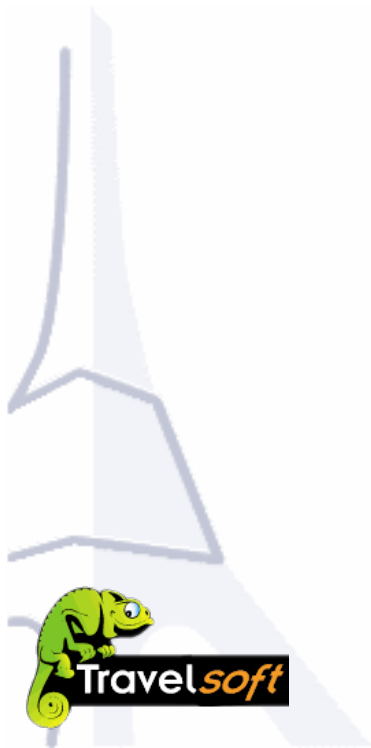
Gagner performances et fiabilité

utilisation d'un débogueur

- Avec débogueur...



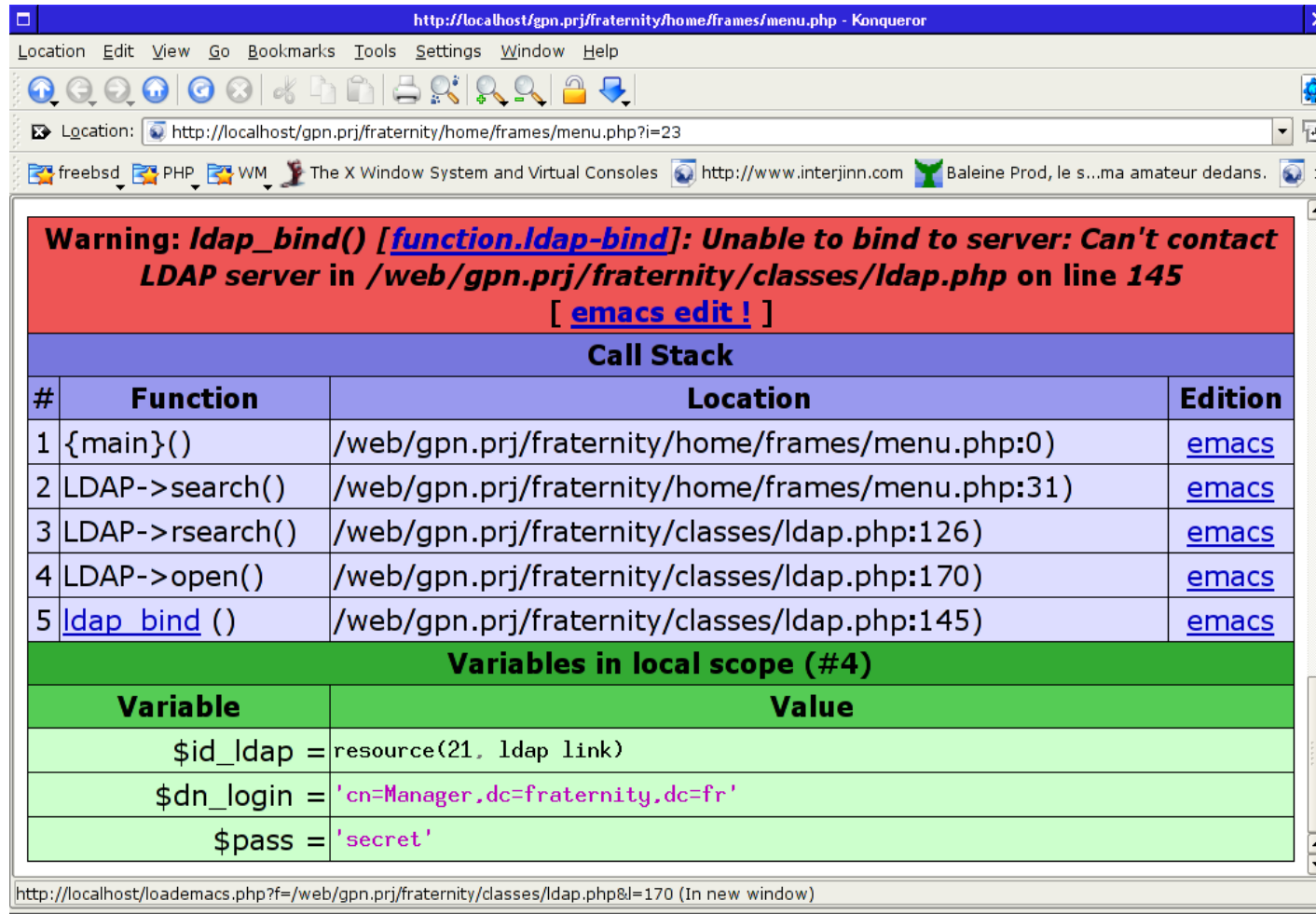
© 2005 - Best practices PHP 5 - Editions Eyrolles



Gagner performances et fiabilité

utilisation d'un débogueur

- Affichage d'une erreur avec Xdebug :



The screenshot shows a web browser window with the address bar displaying `http://localhost/gpn.prj/fraternity/home/frames/menu.php?i=23`. The browser's status bar at the bottom shows `http://localhost/loademacs.php?f=/web/gpn.prj/fraternity/classes/ldap.php&l=170 (In new window)`.

The main content area displays a warning message from Xdebug:

Warning: `ldap_bind()` [function.ldap-bind]: Unable to bind to server: Can't contact LDAP server in /web/gpn.prj/fraternity/classes/ldap.php on line 145
[[emacs edit !](#)]

Below the warning is a section titled "Call Stack" with the following table:

#	Function	Location	Edition
1	{main}()	/web/gpn.prj/fraternity/home/frames/menu.php:0)	emacs
2	LDAP->search()	/web/gpn.prj/fraternity/home/frames/menu.php:31)	emacs
3	LDAP->rsearch()	/web/gpn.prj/fraternity/classes/ldap.php:126)	emacs
4	LDAP->open()	/web/gpn.prj/fraternity/classes/ldap.php:170)	emacs
5	ldap_bind ()	/web/gpn.prj/fraternity/classes/ldap.php:145)	emacs

Below the call stack is a section titled "Variables in local scope (#4)" with the following table:

Variable	Value
<code>\$id_ldap</code>	<code>resource(21, ldap link)</code>
<code>\$dn_login</code>	<code>'cn=Manager,dc=fraternity,dc=fr'</code>
<code>\$pass</code>	<code>'secret'</code>



Gagner performances et fiabilité

utilisation d'un débogueur (démonstration)

- Appel de l'éditeur de puis Xdebug :

The screenshot shows a web browser window with a warning message: "Warning: Invalid argument supplied for foreach() in /web/gpn.prj/fraternity/etc/cfg.php on line 63". Below the warning is a link "Editer cette erreur avec emacs". An arrow points from this link to an Emacs editor window. The Emacs window shows the PHP code for the function `get_sites()`, with line 63 highlighted. The code is as follows:

```
// TODO
function get_sites()
{
    $LDAP = new LDAP();
    $sites = $LDAP->search('ou=sites,dc=fratern');
    $ret_val = Array();
    foreach ($sites AS $value) {
        if ($value['dc'][0]) {
            $ret_val[$value['dc'][0]] = utf8_de
        }
    }
    return $ret_val;
}
```

Below the code, the status bar shows: "-1:-- cfg.php 8:19PM 0.14 (PHP Abbrev)--L63--C0--".

1. Je clique...

2. ...et l'éditeur s'ouvre automatiquement, sur le bon fichier et à la bonne ligne !

© 2005 - Best practices PHP 5 - Editions Eyrolles

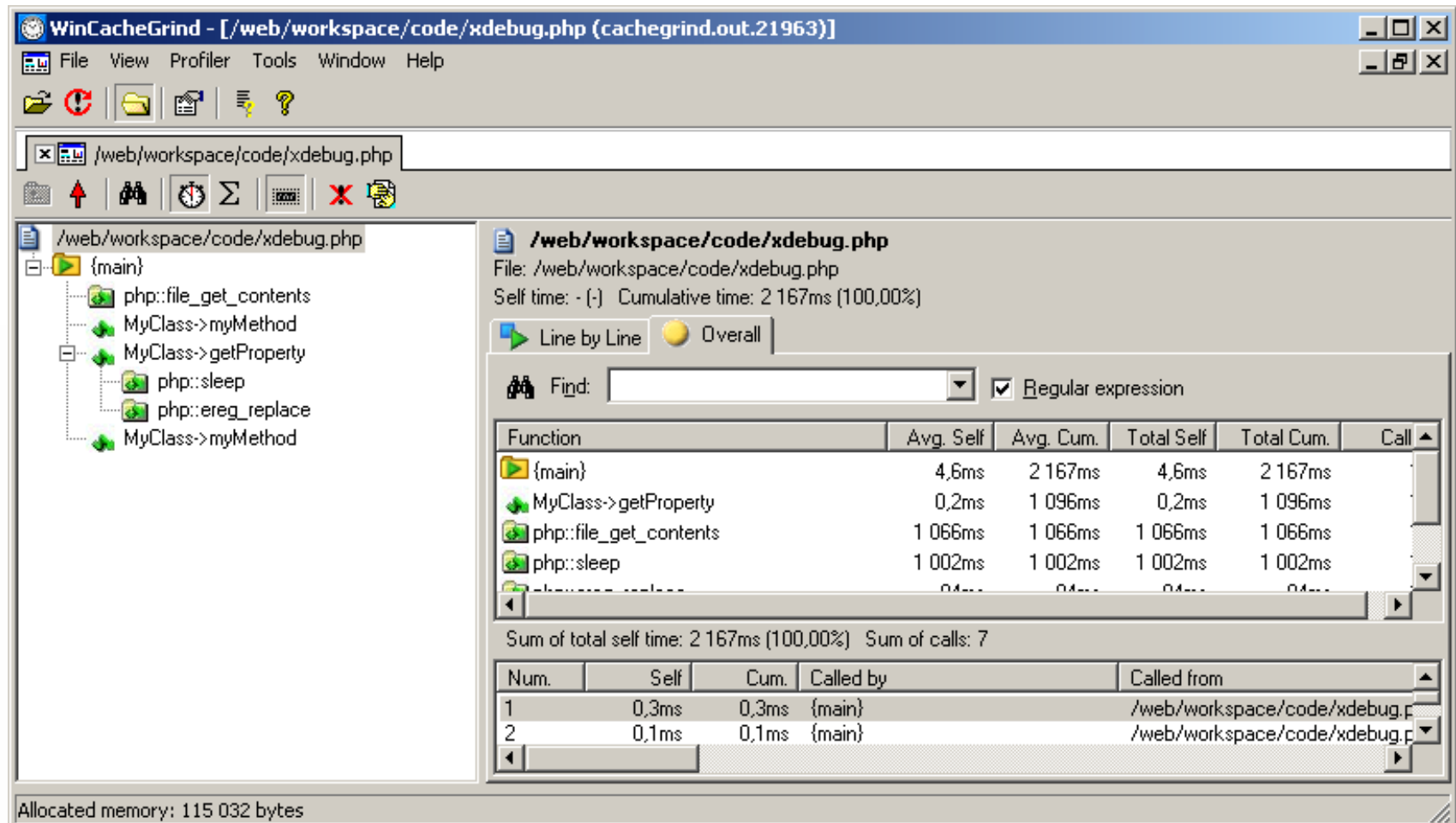
(Note : l'ouverture automatique de l'éditeur nécessite une procédure décrite dans le livre.)



Gagner performances et fiabilité

utilisation d'un débogueur

- Visualisation de la pile des appels avec wincachegrind (Xdebug2) :



WinCacheGrind - [/web/workspace/code/xdebug.php (cachegrind.out.21963)]

File View Profiler Tools Window Help

/web/workspace/code/xdebug.php

/web/workspace/code/xdebug.php

{main}

- php::file_get_contents
- MyClass->myMethod
- MyClass->getProperty
- php::sleep
- php::ereg_replace
- MyClass->myMethod

/web/workspace/code/xdebug.php

File: /web/workspace/code/xdebug.php
Self time: - (-) Cumulative time: 2 167ms (100,00%)

Line by Line Overall

Find: ☒ Regular expression

Function	Avg. Self	Avg. Cum.	Total Self	Total Cum.	Call
{main}	4,6ms	2 167ms	4,6ms	2 167ms	
MyClass->getProperty	0,2ms	1 096ms	0,2ms	1 096ms	
php::file_get_contents	1 066ms	1 066ms	1 066ms	1 066ms	
php::sleep	1 002ms	1 002ms	1 002ms	1 002ms	
php::ereg_replace	0,1ms	0,1ms	0,1ms	0,1ms	

Sum of total self time: 2 167ms (100,00%) Sum of calls: 7

Num.	Self	Cum.	Called by	Called from
1	0,3ms	0,3ms	{main}	/web/workspace/code/xdebug.p
2	0,1ms	0,1ms	{main}	/web/workspace/code/xdebug.p

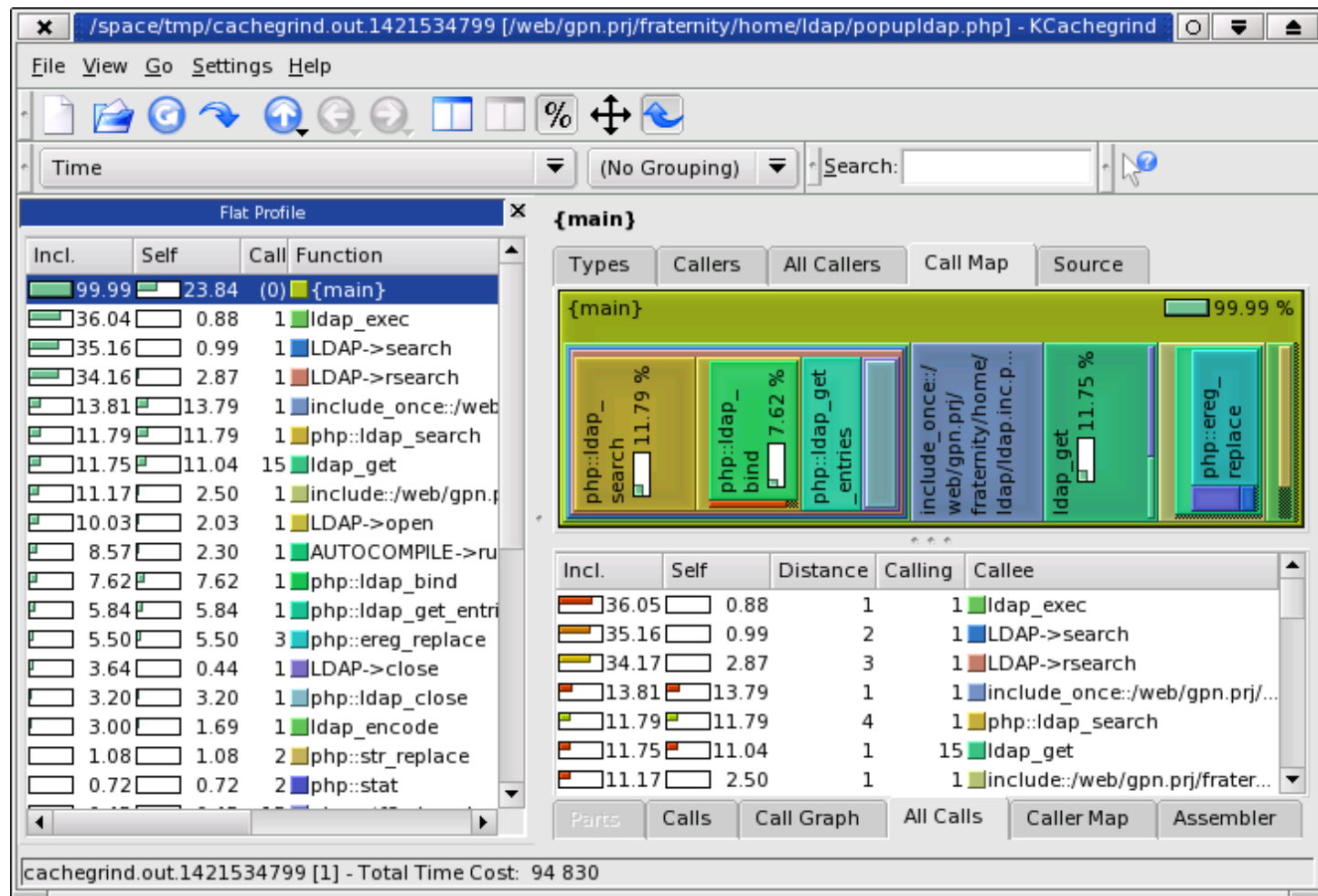
Allocated memory: 115 032 bytes



Gagner performances et fiabilité

utilisation d'un débogueur

- Visualisation du déroulement d'un script avec KCacheGrind :



Gagner performances et fiabilité

utilisation d'un débogueur

The screenshot shows the KCacheGrind application interface. On the left is a 'Flat Profile' table listing functions and their costs. The main window displays a 'TRACE->security_log' call graph, where nodes represent function calls and edges represent the flow of execution. Below the graph, the source code for the security_log function is shown, with line numbers and costs. The code includes calls to date, fopen, fwrite, fclose, and other functions.

Incl.	Self	Cost	Function
380	380	6	fwrite
325	325	2	TRACE->TRACE
296	296	5	str_replace
28	733	263	TEMP-LATE->automake
236	236	4	io_wray
232	232	2	md5
228	228	4	freac
218	218	3	time
212	212	6	strlen
5	543	207	MYSELECT->execute
180	180	2	DOMDocument->cons
179	179	4	header
164	164	2	is_numeric
145	145	4	filesize
128	128	1	strpos
115	115	1	USER->USER
107	107	1	AUTH->have_permission
1	177	99	MYSELECT->next_row
97	97	1	XML->XML
589	589	4	AUTH->get_pilot_skin
2	519	81	TRACE->security_log
78	78	1	guidde
2	266	65	TEMP-LATE->get
63	63	1	octocoolie
513	513	2	XML->check_container
59	59	1	erand
1	542	53	TRACE->info
39	176	41	AUTH->authenticate
518	518	41	AUTH->update_session
218	218	41	TEMP-LATE->TEMPLATE
31	31	1	rand
30	30	1	is_string
3	523	23	XML->xml_transform
647	647	24	MYSELECT->get_xml
37	594	21	AUTH->make_session
734	734	23	AUTH->update_session
282	282	23	MYSELECT->MYSELE
649	649	13	TRACE->debug
3	149	1	AUTH->create_session
1	139	13	AUTH->load_session
42	509	3	AUTH->require_auth

```
# Cost Source (/web/masterflow/lib/security/trace.inc)\n82 $eng = date("Y/m/d H:i:s")."."$REMOTE_ADDR."."$SCRIPT_FILENAME."";\n762 4 calls to 'date' (INTERNAL)\n83 $eng = $err."[".$function."].".$desc."\\n";\n84 if (error("[".$function."].".$desc."\\n")) {\n454 4 calls to 'ereg' (INTERNAL)\n85 if ($level == $CFG->log_level) {\n86 $file = $ENV_PILLOT_BASE."var/pilotlogs/".$CFG->err_log_file.$level."";
```

Liens

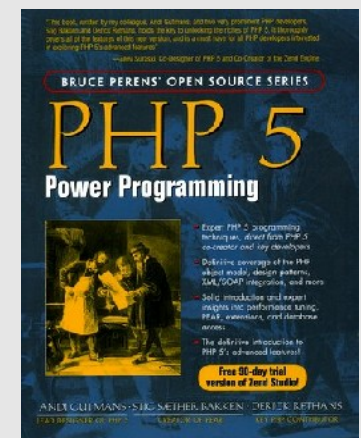
<http://dd.cron.ru/dbg/> <http://xdebug.org/>
<http://pecl.php.net/package/apd>

Livres

Advanced PHP Programming

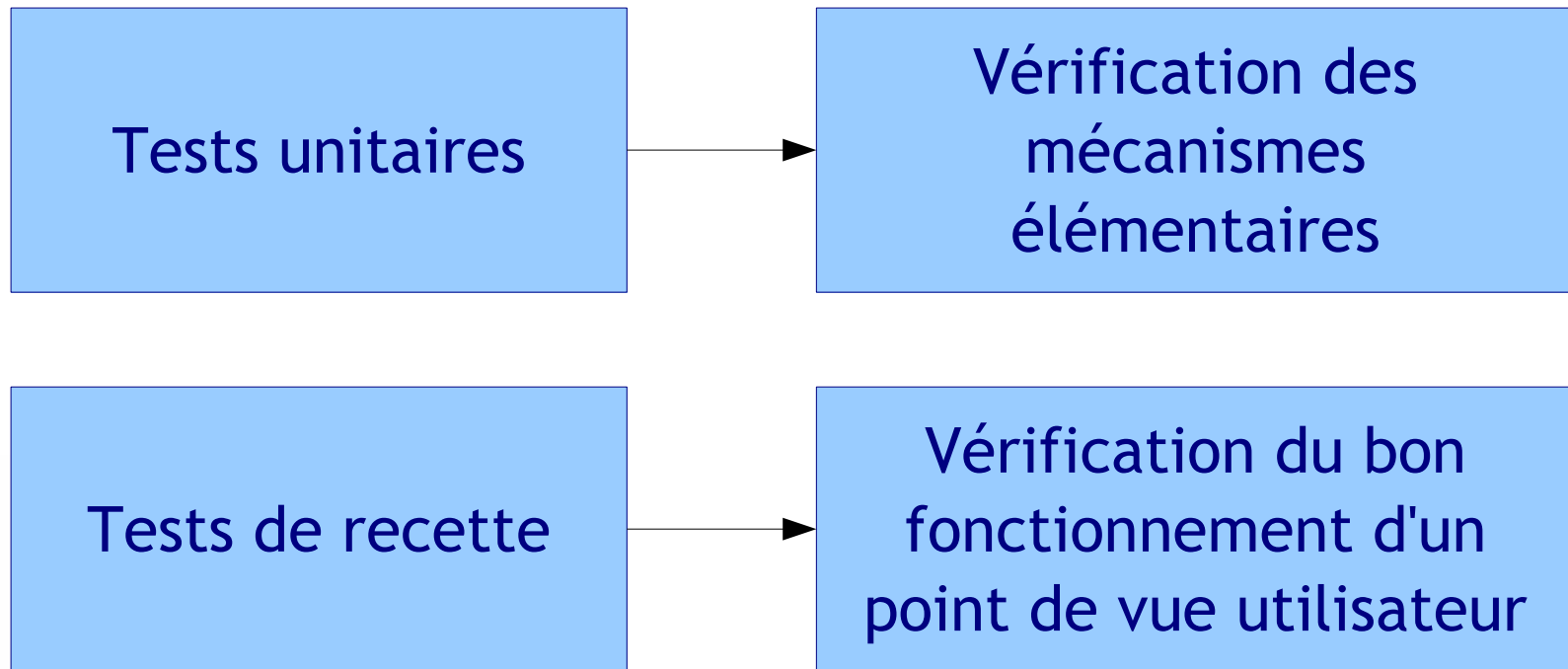
A practical guide to developing large-scale Web sites and applications with PHP 5

George Siobudzie

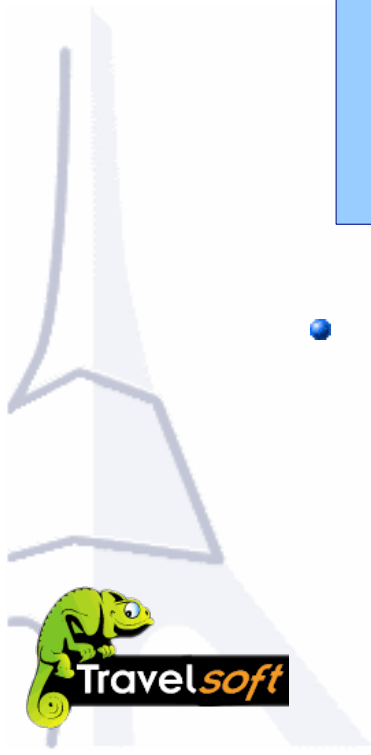


Gagner performances et fiabilité

tests unitaires et tests de recette



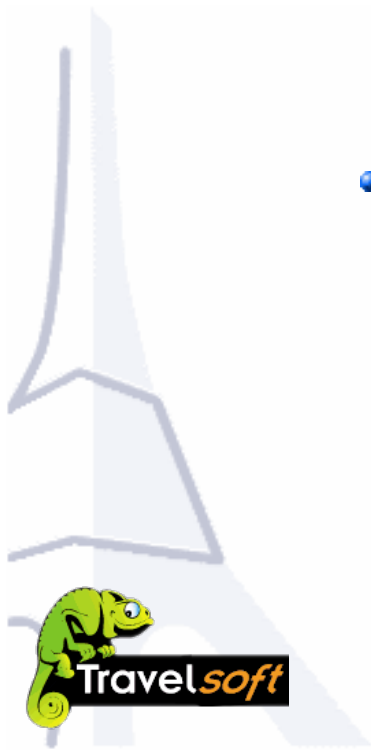
- Pour développer des tests :
 - **SimpleTEST** : http://www.lastcraft.com/simple_test.php
 - **PHPUnit** : <http://pear.php.net/package/PHPUnit>



Gagner performances et fiabilité

dans l'exploitation

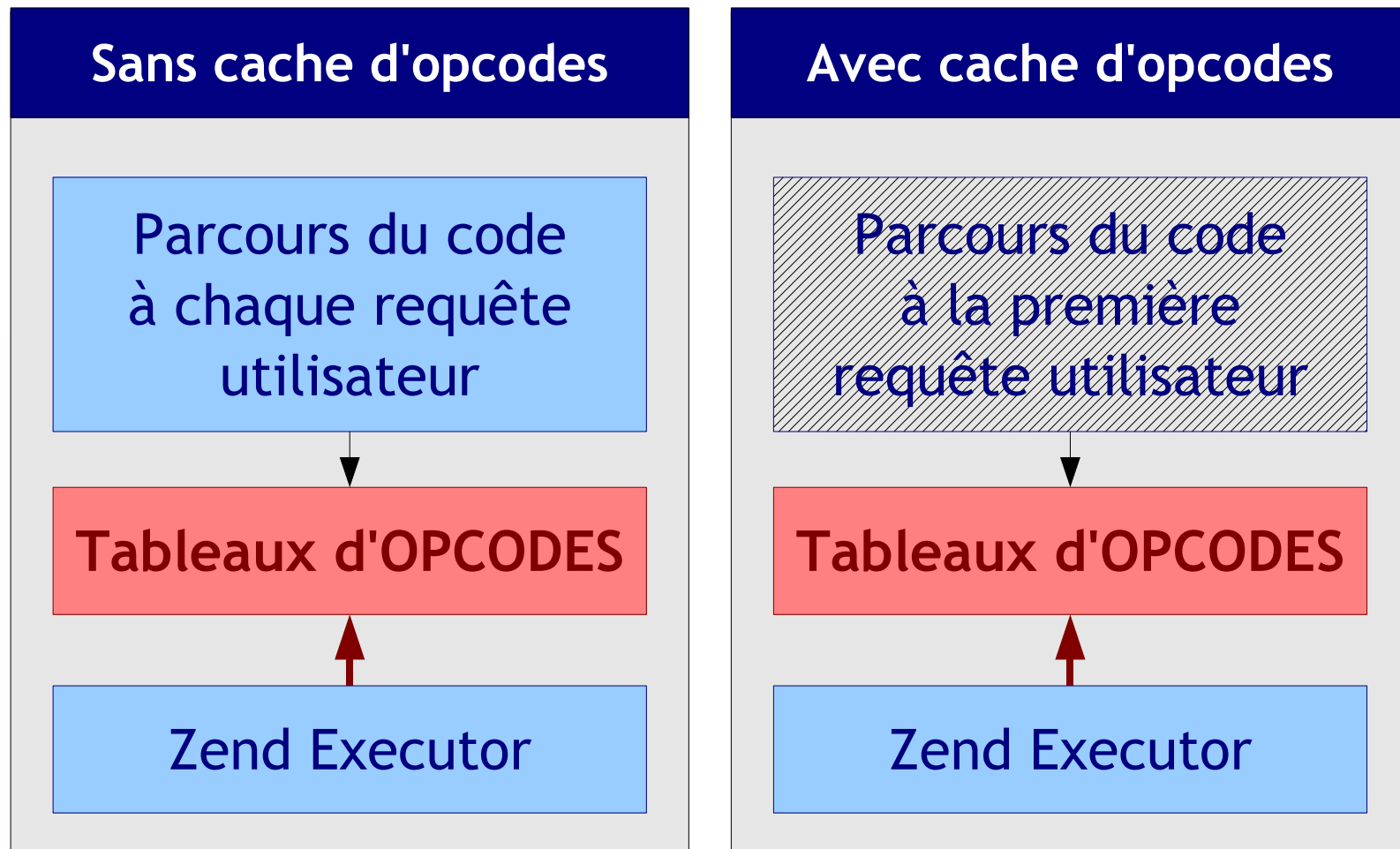
- **Caches d'opcodes**
- **Exploiter votre dépôt de données**
- **Le trio « développement, recette, production »**



Gagner performances et fiabilité

cache d'opcodes

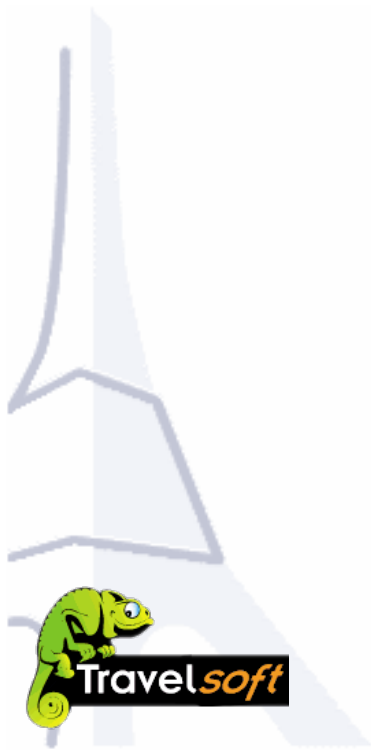
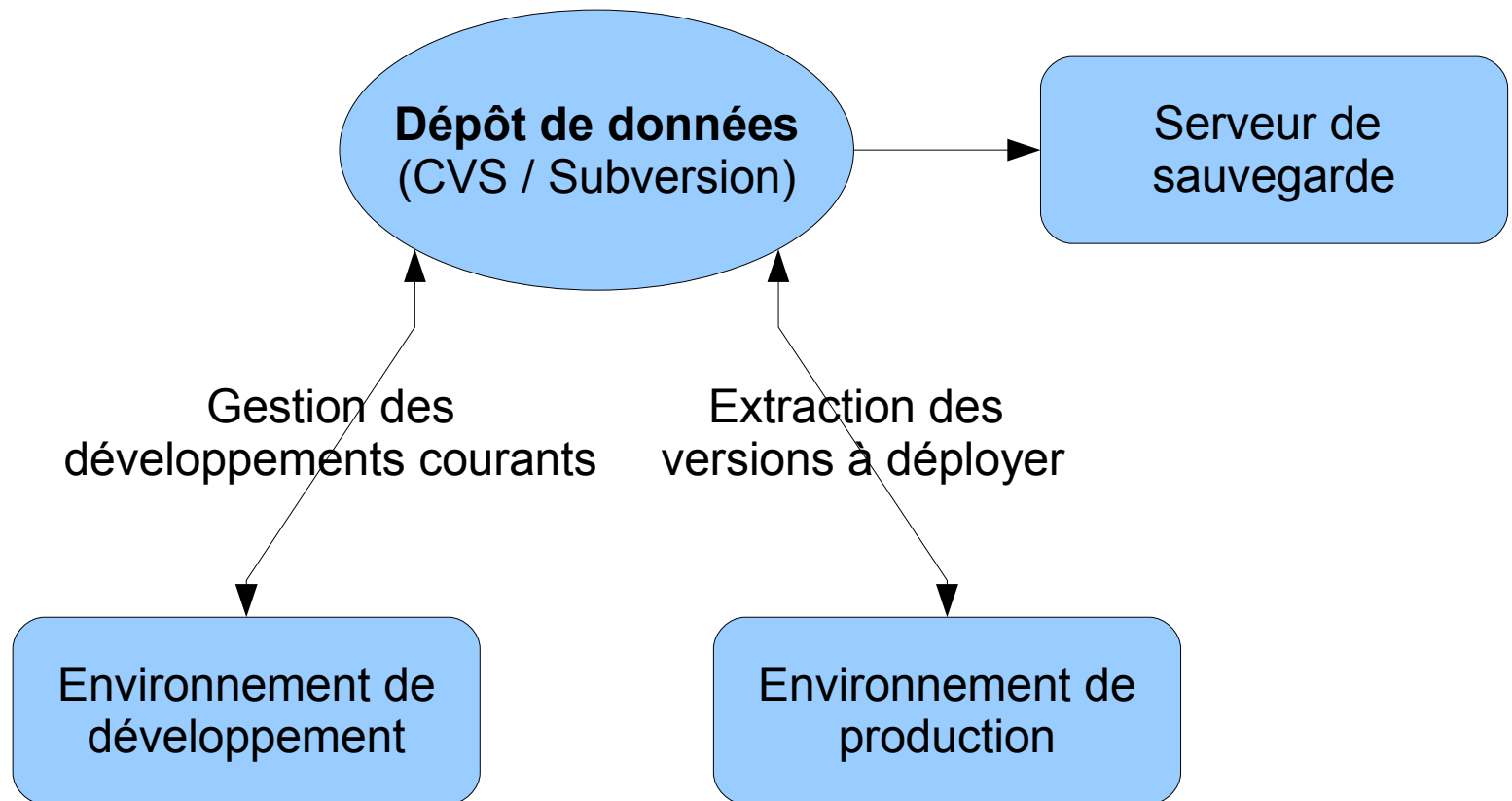
- Principe du cache d'opcodes (APC, eAccelerator, ...)



Gagner performances et fiabilité

exploiter votre dépôt de données

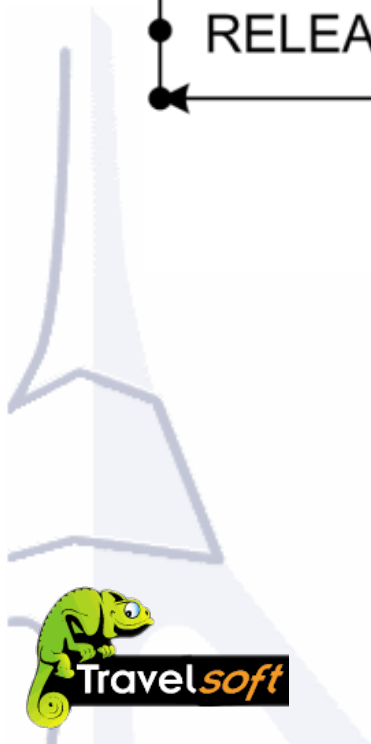
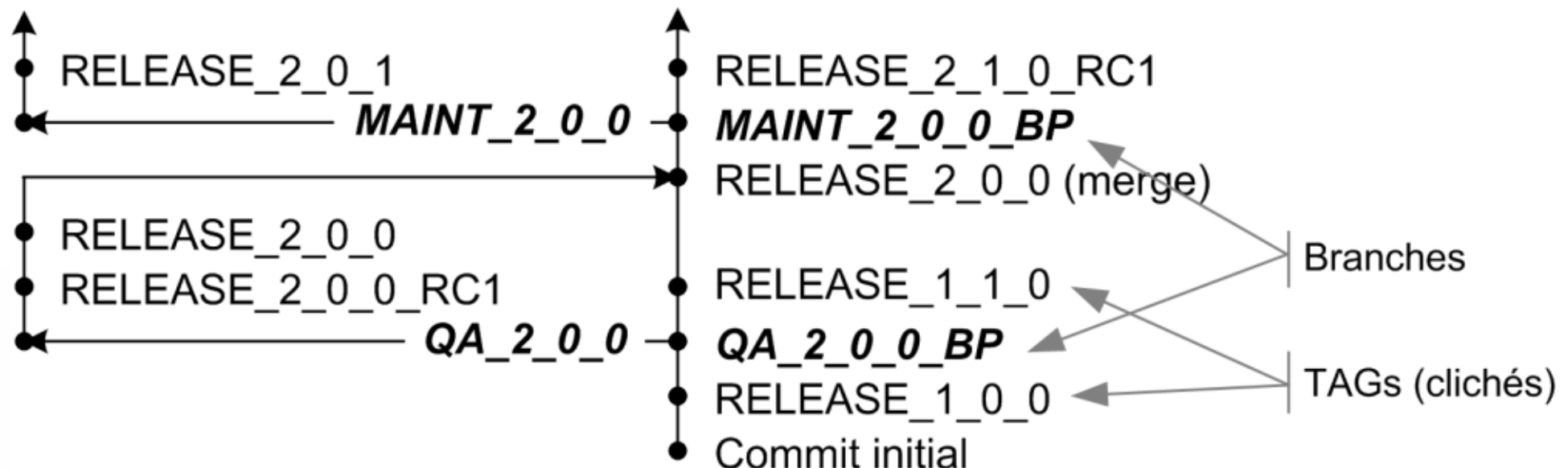
- Le dépôt de données au centre de vos projet



Gagner performances et fiabilité

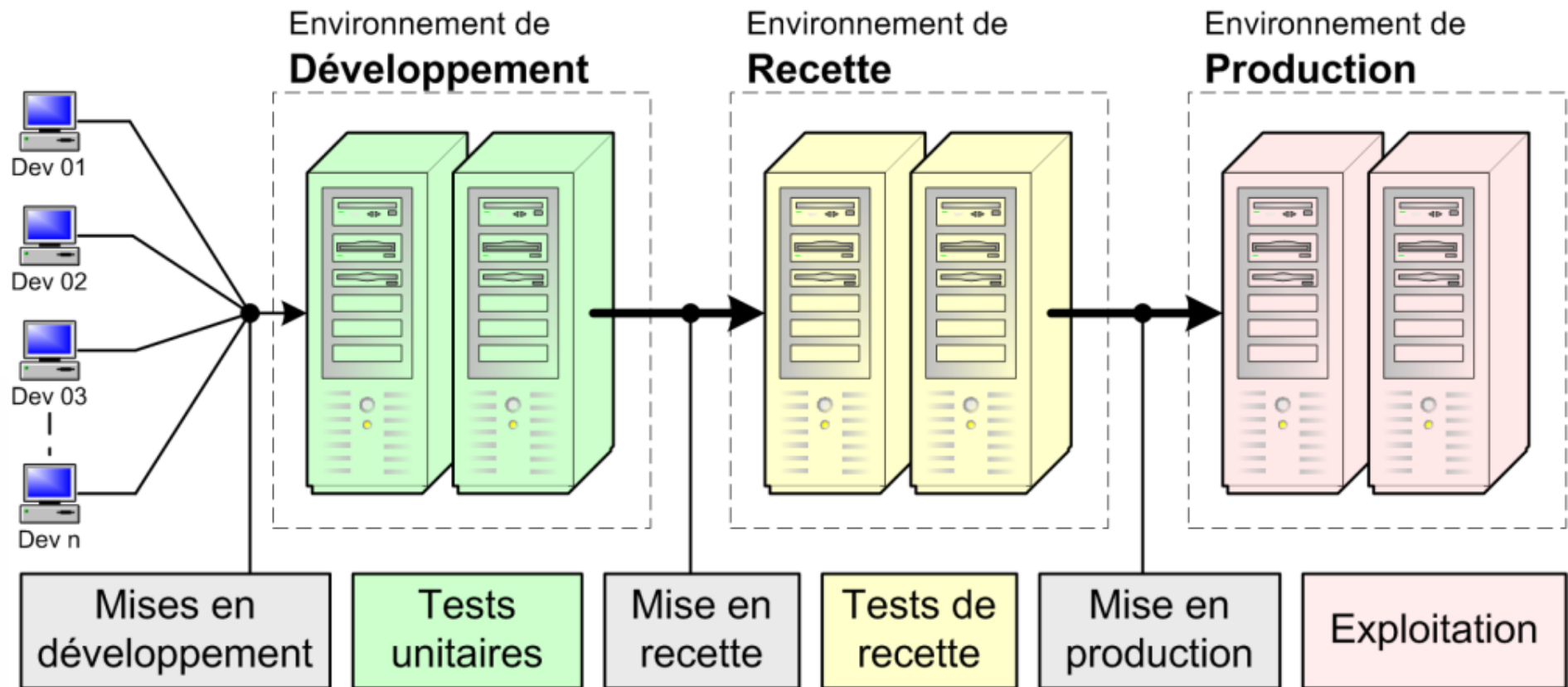
exploiter votre dépôt de données

- Se fixer des règles de nommage pour les numéros de versions



Gagner performances et fiabilité

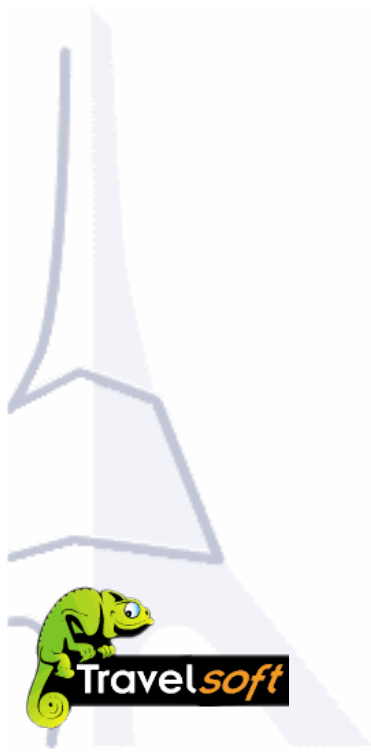
le trio « développement, recette, production »



Augmenter sa compétitivité

dans le développement

- **L'évolutivité technique**
- **Modéliser !**
- **Les motifs de conception**
- **L'interopérabilité**



Augmenter sa compétitivité

l'évolutivité technique

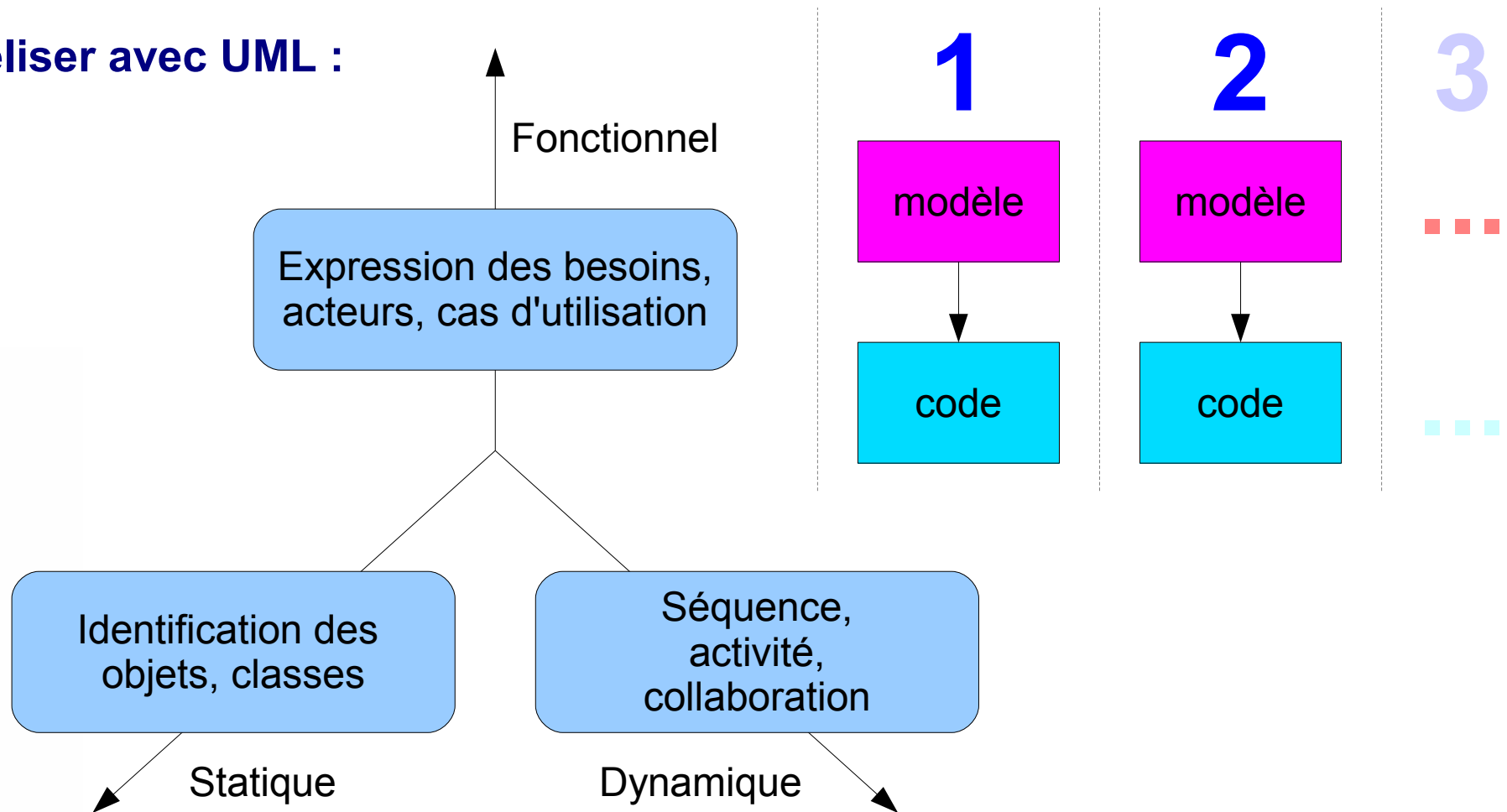
- **Evoluer avec les nouvelles technologies**
 - **Changements de versions** (PHP, serveur HTTP, dépendances)
 - **Adaptation aux standards** (SOAP, RSS, XHTML, etc.)
- **Adopter une architecture stratégique**
 - **Définir des couches d'abstraction**
(extensions, logique métier, présentation, briques logicielles)
 - **Favoriser la réutilisabilité, la modularité** (POO)
- **Prévoir les évolutions futures**
 - **Model Driven Architecture**
 - **PHP 6**



Augmenter sa compétitivité

modéliser !

- Toute application professionnelle sérieuse fait l'objet d'une modélisation
- Modéliser avec UML :



Augmenter sa compétitivité

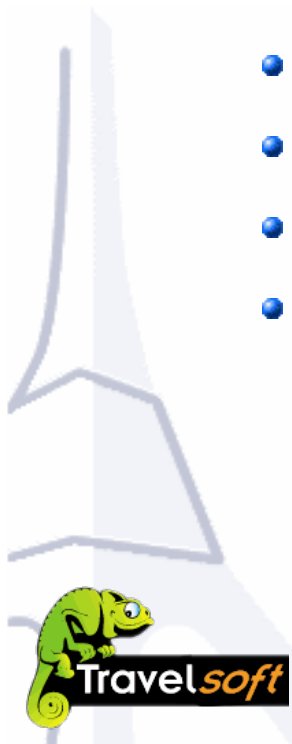
les motifs de conception (design patterns)

- Solutions standards pour répondre à des problèmes récurrents d'architecture et de design de logiciels.
- Quelques motifs utilisés en PHP :
 - **Singleton** : la classe qui s'instancie une seule fois
 - **Fabrique** : simplifier la création d'objets
 - **Prototype** : objet destiné à être cloné
 - **Façade** : manipulation d'un sous-système complexe
 - **MVC** : modèle d'architecture adaptée au web
 - **Itérateur** : pour itérer sur des collections

Lien

http://php.openstates.org/generateur_de_motifs.php

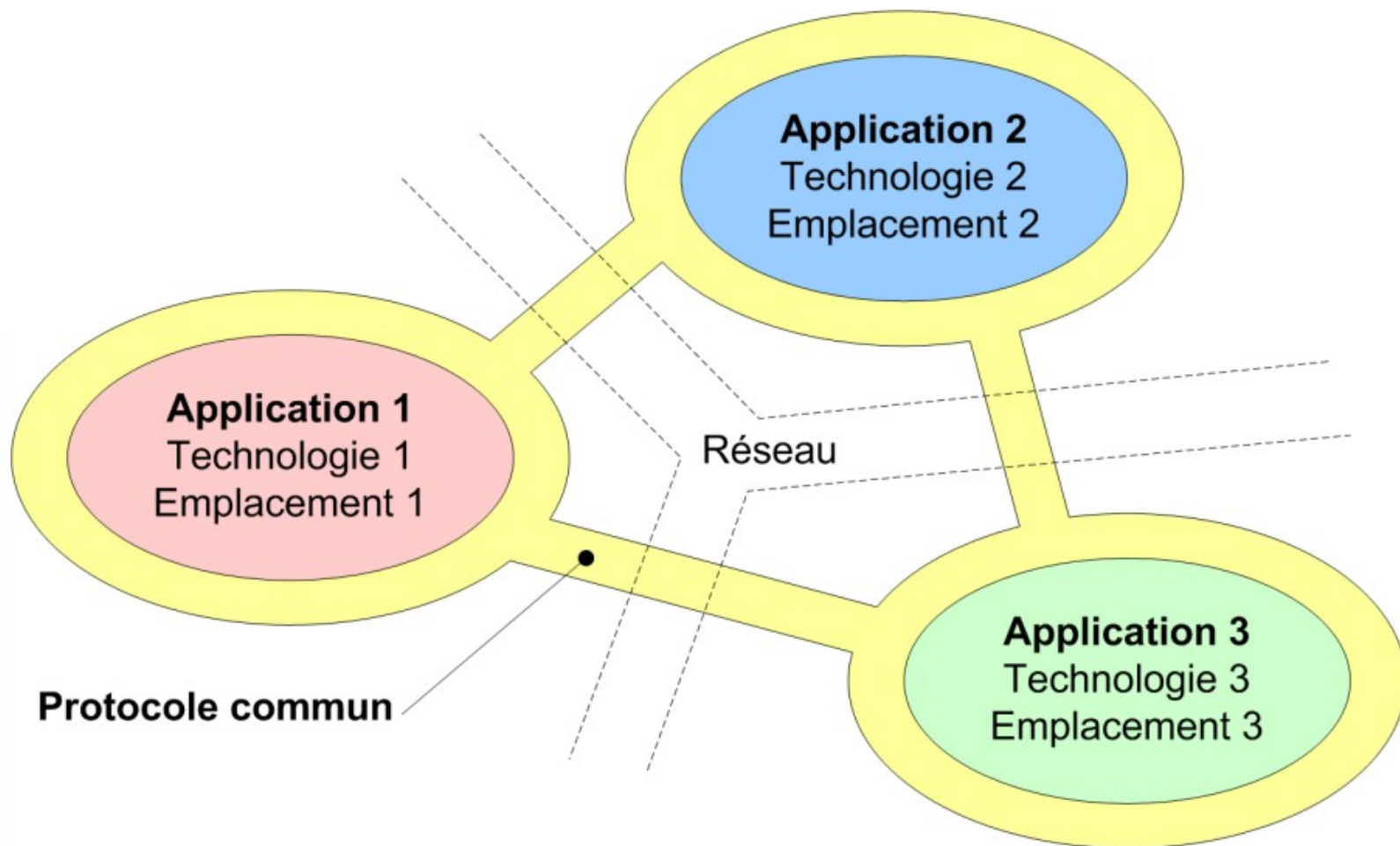
Livres



Augmenter sa compétitivité

l'interopérabilité

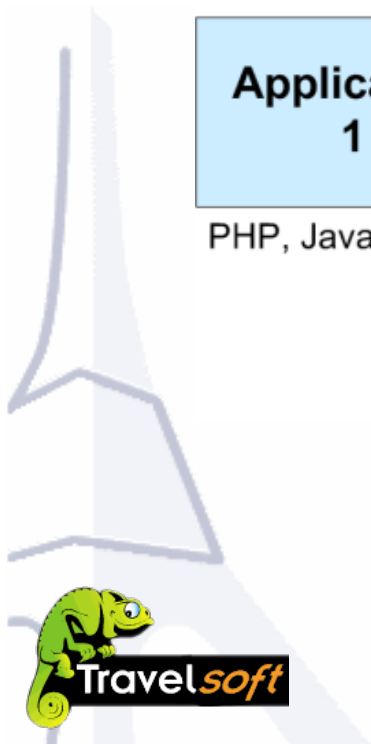
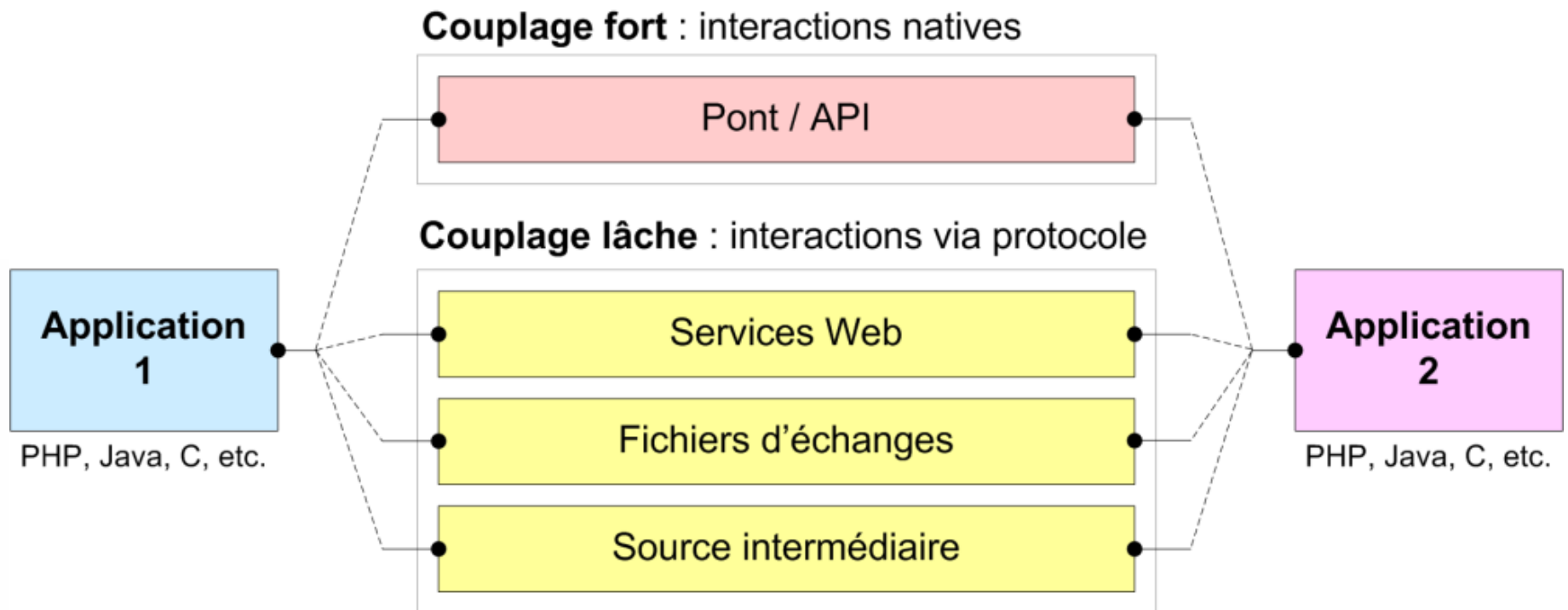
- Comment rendre vos applications communicantes ?



Augmenter sa compétitivité

l'interopérabilité

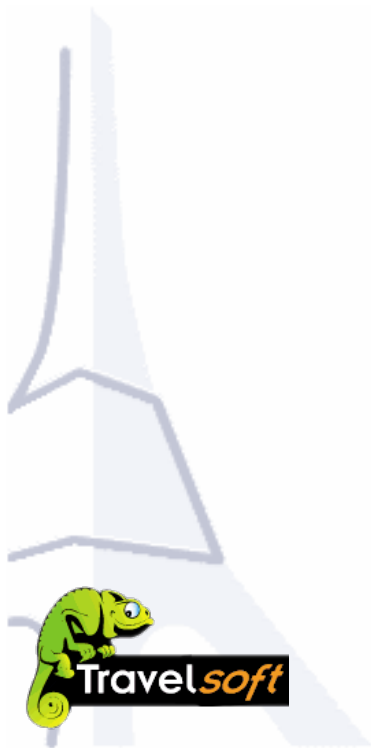
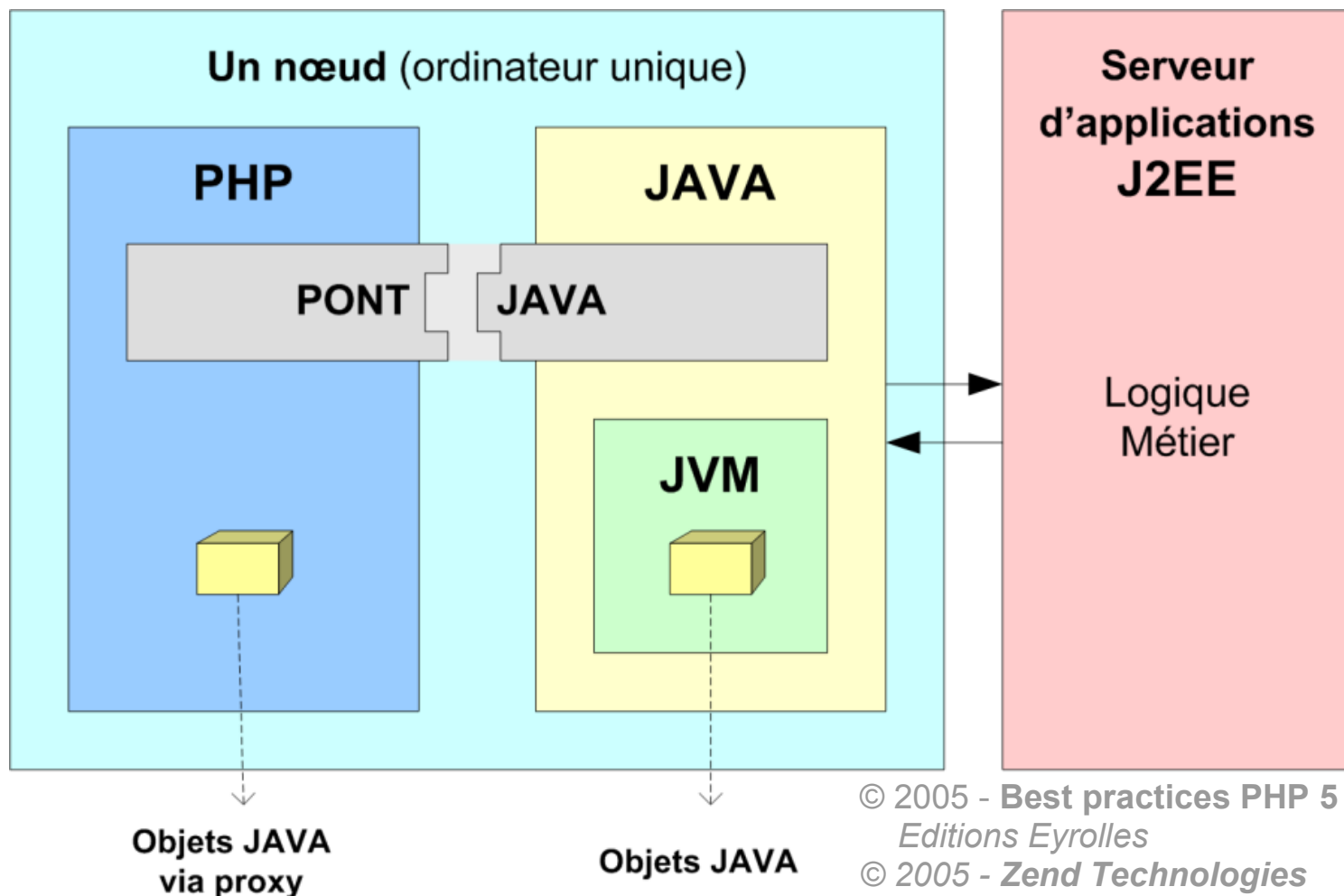
- Types de couplages



Augmenter sa compétitivité

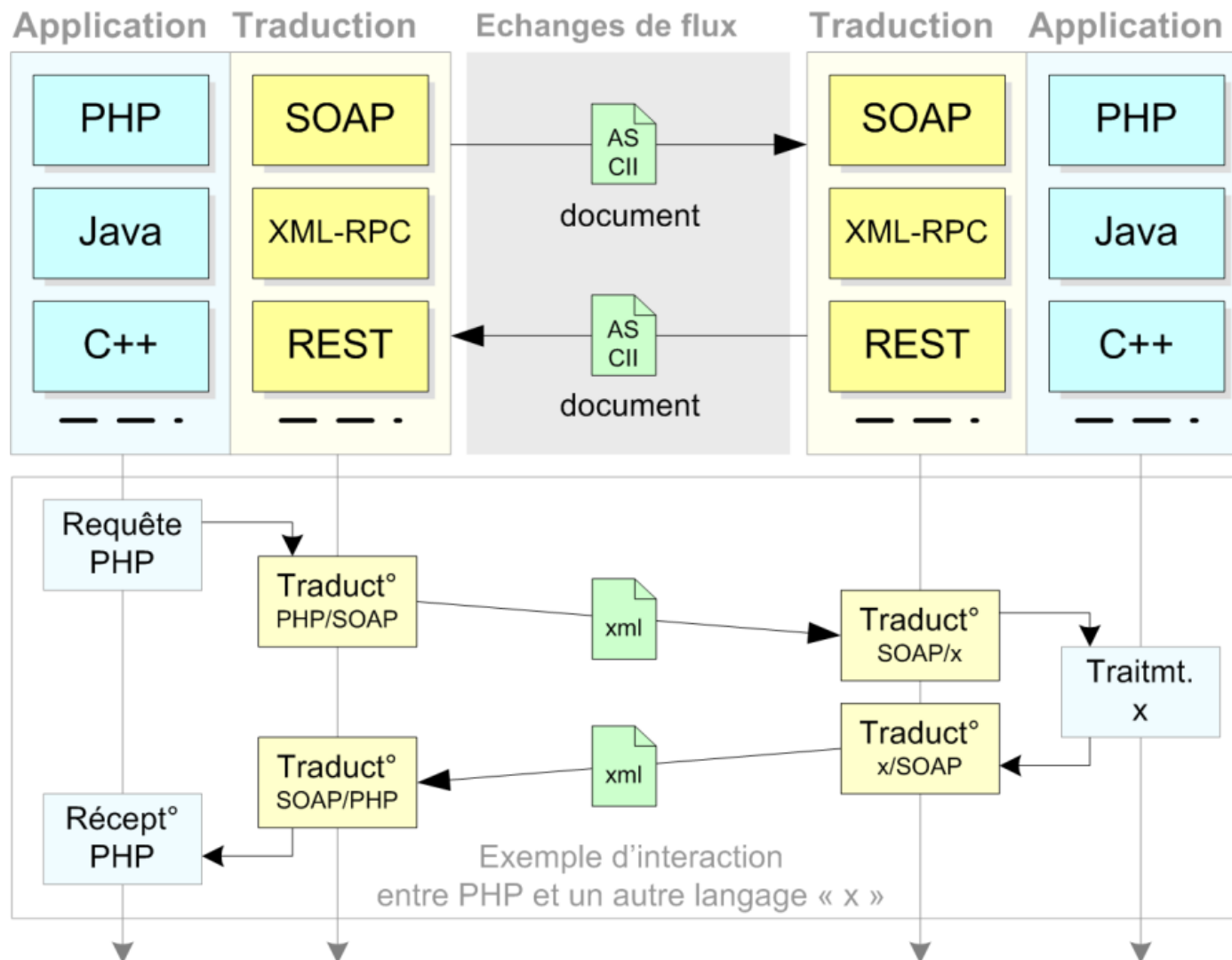
l'interopérabilité

- Couplage fort



Augmenter sa compétitivité

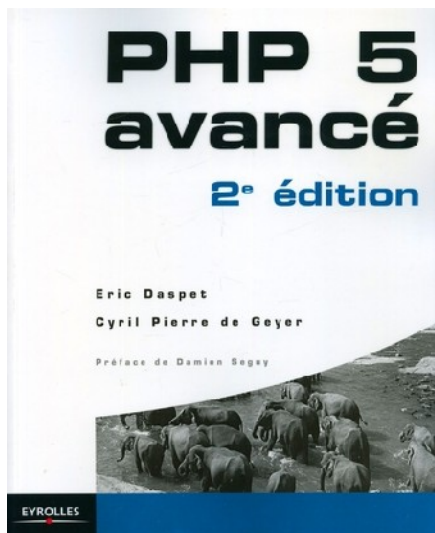
l'interopérabilité



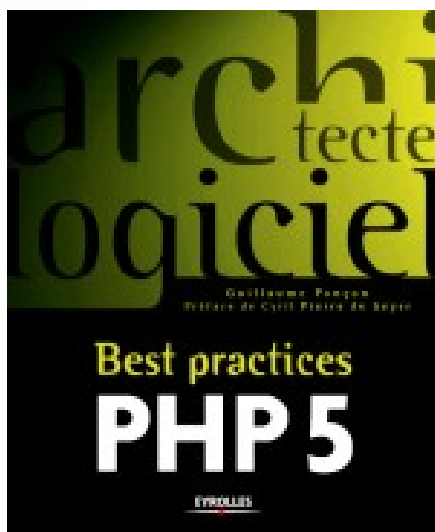
© 2005 - Best practices PHP 5 - Editions Eyrolles

Ressources

livres



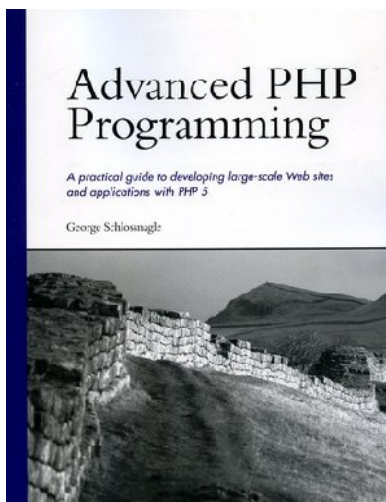
- **Un livre de référence destiné aux débutants comme au confirmés.**
 - Nombreuses explications pédagogiques
 - Exemples pratiques
 - A jour sur les dernières nouveautés PHP 5



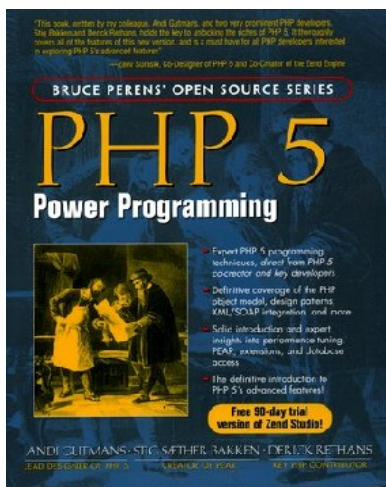
- **Les bonnes pratiques pour une utilisation fiable et professionnelle de PHP.**
 - Destiné aux décideurs, chefs de projets et développeurs qui souhaitent aller plus loin avec PHP 5.
 - Livre récent (sortie officielle le 17 novembre)

Ressources

livres



- **Pratiques de programmation et concepts bas niveau expliqué par un développeur de PHP.**
 - Utilisation de PHP en entreprise
 - Gestion des performances
 - Création d'extensions



- **La référence PHP 5 de la Core Team de PHP.**
 - Conseils d'utilisation de PHP 5
 - Débogage d'applications
 - Intéropérabilité

Conclusion

avez-vous des questions ?

