Common - Refactoring #3122

goodbye mdb2, welcome <new way to access database>

21/03/2011 17:51 - Anonymous

Status:	Bug resolved	**Start date:**	21/03/2011
Priority:	Normal	**Due date:**	
Assignee:	Hans De Bisschop	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	LCMS 3	**Spent time:**	0.00 hour

Description

Currently the Database works with direct calls to mdb2 wich is quite obsolete, spawn errors, is absolutely not OO, is not very performant, is not very extensible, ...

since PHP 5.1 PDO has been a standard for abstracting the access to a database (it does not abstract the SQL, it abstract the driver).

Stephane from the clarolien team is currently trying to start our an application for Chamilo and try to test any piece of functionality he's writing. Unfortunately MDB2 goes into it's way and he tried to hack the Database class for using PDO instead of MDB2. Currently he is obtaining good results but cannot know the impact for other application (even if he doesn't change the methods signature, the behaviour might change).

What would you think of changing the calls to mdb2 by calls to pdo, or change the design of Database class to allow both possibilities based on config options (maybe keep mdb2 n prod and pdo in dev)

How can we share his work ?

PS for those who speak french, here is his message :

"
Salut,

Je t'écris pour faire un peu le point sur la situation et pour te proposer un sujet de réflexion pour notre prochaine réunion.

La semaine dernière, j'ai poursuivi les tests en isolation et en intégration sur la couche modèle de l'application. L'écriture de ces tests ne posait pas de problème particulier : pris isolément, ils fonctionnaient parfaitement. En revanche, l'ajout d'un nouveau test entraînait presque
systématiquement des problèmes dans les tests déjà écrits. On a vu ensemble que cette situation était liée à l'utilisation de globales par MDB2. J'ai
donc essayé, comme nous l'avions fait, d'utiliser l'option 'backupGlobals disabled' ou de la désactiver là où elle semblait poser problème. Mais rien n'y faisait, et je perdais au moins une heure par jour à tenter toutes sortes de combinaisons pour faire fonctionner l'ensemble.

Ce week-end, j'ai exploré une voie beaucoup plus radicale : remplacer MDB2 par PDO. Je sais que ce n'était pas au programme, je sais que ce genre de remplacement concerne tous les développeurs de chamilo et ne peut se faire à la légère. Mais j'ai quand même essayé, à titre expérimental en quelque sorte. J'ai réimplémenté la classe Database de chamilo en remplaçant tous les appels à MDB2 par des appels à PDO (sans, bien sûr, modifier quoi que ce soit dans les attributs, les noms et les signatures de méthodes); j'ai également retouché quelque classes apparentées, en particulier la classe ObjectResultSet.

Ceci n'est encore qu'un brouillon mais le résultat est tout de même très significatif. Sans plus aucun appel à MDB2, mon installation de chamilo fonctionne parfaitement et mes tests tournent maintenant sans aucun problème. Les performances me semblent également meilleures (bien que sans test dédié, cette impression
reste assez subjective) ainsi que les répercussions possibles sur l'écriture du code : full-orienté objet, clair, propre et ouvert à une gestion efficace des erreurs.

Je ne vais pas te faire un éloge de PDO, dont tu connaîs sûrement mieux que moi les avantages. Je crois que la vraie question concernant chamilo n'est pas : "faut-il remplacer MDB2 par PDO ?" mais bien : "combien de temps chamilo pourra-t-il tenir sur une couche modèle notoirement périmée ?". Et concernant notre propre travail autour de ce projet, je pense que la question est encore plus critique. Bâtir une application sur une base aussi hasardeuse finira, tôt ou tard,
par devenir très problèmatique. Ne vaudrait-il pas mieux de suivre directement une voie plus sûre ?

Cette option entraînerait bien sûr pas mal de changements du côté de chamilo, et je ne pense pas que la transition puisse se faire absolument sans douleur. Mais c'est possible et souhaitable, pour peu que les développeurs de chamilo soient près à tenter le coup

d'une manière ou d'une autre.

Bon, je te laisse réfléchir à tout ça (désolé pour la longueur), et je te souhaite une bonne fin de journée.

À jeudi,

Stéphane
"

History

#1 - 22/03/2011 00:09 - Yannick Warnier

I personally recommended to the 2.0 team (about 3 years ago) to go with PDO instead of a (at the time) non-standard, non-stable MDB2 implementation.

At the time, I was answered that MDB2 was really an ORM tool, while PDO was not, and that it had a lot of advantages. 3 years later, I still fail to see the advantages (and MDB2 does not seem to have a bright future either), so although my voice doesn't account for much because I will probably not intervene in the refactoring, I am fully supporting it, as long as you can actually make it so that it doesn't break the software for another year or so.

Of course, unit tests should be developed to avoid regressions, so this can only come if one or two person ensure a continuity to the work of Philippe so far.

Yannick

#2 - 22/03/2011 09:13 - Anonymous

Great idea. It would be a ton of work though, since there are various places in the code where the Database abstraction leaks out MDB2 objects. This should all be refactored before this could be done. It would be very hard to do this now without tests, and break even more stuff than my postgres patch ;)

Also, while you're in there modifying all the db stuff I'd argue for getting rid or simplifying the way objects, assoc-arrays and collections of such are handled.
Specifically, there's no reason to have an ObjectResultSet, ArrayResultSet and RecordResultSet. The code is almost identical, so it isn't DRY. Just one non-abstract ResultSet would be enough, with specific methods to fetch either objects, records or arrays. Especially ArrayResultSet is ludicrous; it just adds overhead and doesn't abstract anything away. Instead, it complicates the code that uses it rather than simplifying it by making it impossible to use standard language constructions to loop through and access specific elements.

It would make more sense to let a new ResultSet implement the SPL Iterator interface so that you can just use language constructions like foreach() to loop through stuff. This would make it a lot more convenient and natural to use.

#3 - 24/03/2011 13:10 - Goulwen Reboux

The doctrine project has produced a DBAL on top of PDO that can be used without the ORM.

http://www.doctrine-project.org/projects/dbal

I think we can plug the DatabaseManager on top of Doctrine2 DBAL (that's what Propel2 will do for example)

#4 - 24/03/2011 13:21 - Anonymous

I would advise against using Doctrine's DBAL because it locks in another dependency.

PDO is at least semi-standardized, while doctrine is yet another project you'd be dependent on. This DBAL will follow the Doctrine release schedule, and will probably not do things in Chamilo's best interests when Doctrine wants one thing and Chamilo another.

#5 - 25/03/2011 17:39 - Anonymous

Doctrine 2 has been chosen by symfony 2.0 and Zend Framework 2.0 as the default db access library (and orm also). I think we can consider it much more standard than mdb2 ^^

#6 - 25/03/2011 18:38 - Hans De Bisschop

As discussed during the code sprint it's perfectly possible to implement an alternative "storage engine", so I'm really interested in exploring true alternatives to MDB2. I had a look at Doctrine 2 and it looks quite interesting. (very interesting even)

Using PDO directly means providing extra implementations for all dbms-systems we want to support. Using Doctrine 2 DBAL means adding a dependency. Both have advantages, both have disadvantages. I personally consider true database abstraction important, so I'm leaning towards an option like Doctrine 2, but that's just me ...

At any rate I'd be very interested in rolling out a gradual replacement for MDB2.

#7 - 29/03/2011 11:12 - Anonymous

- Subject changed from goodbye mdb2, welcome PDO to goodbye mdb2, welcome <new way to access database>

Changed the title, since we do not know yet what tool will replace mdb2

#8 - 01/04/2011 14:56 - Stefaan Vanbillemont

- Project changed from Chamilo LCMS Connect to Repository

#9 - 04/04/2011 09:22 - Anonymous

- Project changed from Repository to Common

#10 - 14/04/2011 15:35 - Stefaan Vanbillemont

- Target version set to 2.1.0

#11 - 16/06/2011 14:30 - Stefaan Vanbillemont

- Target version changed from 2.1.0 to 31

#12 - 17/06/2011 13:32 - Stefaan Vanbillemont

- Target version changed from 31 to Backlog (default)

#13 - 02/07/2012 14:16 - Stefaan Vanbillemont

- Status changed from New to Bug resolved

- Assignee set to Hans De Bisschop

- Target version changed from Backlog (default) to LCMS 3

First implemtation for Doctrine is present. Considering this issue as resolved. Further roll-out should be planned with new core change iterations