¿A qué jugamos?

Estructura Nucli Java

Versió 3.0

G6

Històric de revisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versió** | **Descripció** | **Autor** |
| 20/03/2018 | 1.0 | Primer Plantejament nucli Java | Alejandro García i Sergi Portero. |
| 10/04/2018 | 2.0 | Segon Plantejament nucli Java | Alejandro García i Sergi Portero. |
| 24/04/2018 | 3.0 | Tercer plantejament nucli Java. | Alejandro García. |

Taula de continguts

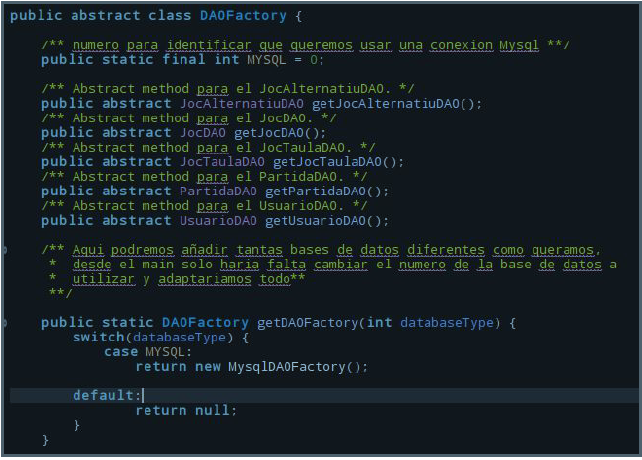
[1. Informació estructura Nucli Java 4](#_Toc514690508)

[2. Sistema de logs 7](#_Toc514690509)

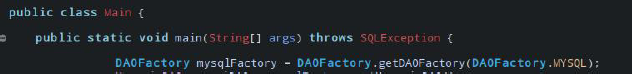
[3. Canvis 24/04/2018 8](#_Toc514690510)

1. Informació estructura Nucli Java

He utilitzat una capa DAO (Data Acces Object) per aïllar la implementació de la base de dades que utilitzem. D'aquesta forma, complim el principi de **“single responsability**”, i aconseguim un codi amb menys acoblament i alhora, mes cohesionat ja que cada classe DAO tindrà els mètodes per gestionar les seves respectives classes concretes, i no tindrem totes les funciones una sola classe de aplicacionBD.



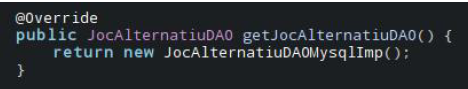
La classe DAOFactory l'única cosa que farà serà crear objectes DAO (objectes que accedeixin a la base de dades i facin operacions amb ella) amb funcions abstractes per retornar un objecte de tipus interfície (general) de cada tipus d'objecte sense tenir en compte si usen Mysql, Oracle o qualsevol altra base de dades i un mètode per especificar que tipus de base de dades volem usar, a través d'aquest mètode en el main triem que base de dades volem usar:



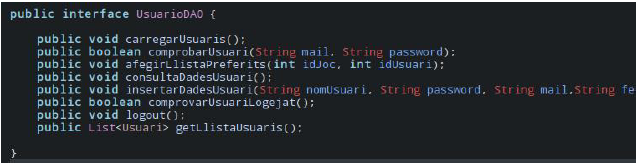
Simplement hem de passar-li per paràmetre el nombre de la base de dades a usar (en aquest cas he definit un int estàtic a la pròpia classe per usar-ho des de qualsevol costat, cridat igual que la base de dades a usar), i tot el nostre codi estarà adaptat sense tocar res més en el codi.



A la classe MysqlDAOFactory implementem el mètode estàtic que retorna una connexió de tipus Mysql.



I els mètodes que fan override als de la classe DAOFactory, per retornar una connexió de tipus Mysql. Cada objecte DAO (usuari, joc, partida, etc) té una interfície per aïllar la implementació de cadascun d'aquests, i d'aquesta manera poder tractar per igual un DAO de mysql, que un de oracle o qualsevol altre accés a dades. Això dona una gran escalabilitat i flexibilitat.



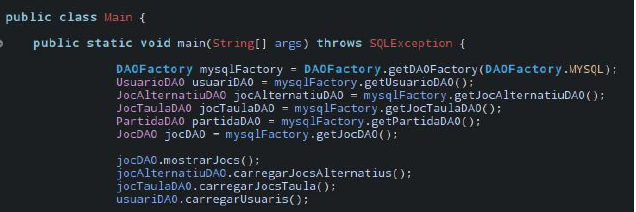
(Exemple d'interfície, aquests mètodes els tindran tots els objectes DAO sigui el que sigui

la seva implementació)

Finalment queda implementar cada DAO segons la seva base de dades (estan en el paquet com.aquejugamos.DAOImplementation, en aquestes classes podem treballar com sempre hem fet, no hem de canviar gens mes), nosaltres de moment solament necessitem la de Mysql, així que fem la implementació per Mysql (ja la teníem feta, solament he separat per classes les funcions que ja teníem).



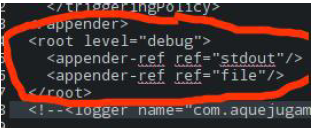
En el main solament fa falta instanciar les classes:



(UsuarioDAO tindrà els mètodes de base de dades relacionats amb usuari: login, registrar, veure favorits usuari, etc...), el de JocDAO el relacionat amb Joc, etc...) Si necessitem afegir una nova funció solament fa mancada afegir-la a la classe a la qual pertanyi i ja.

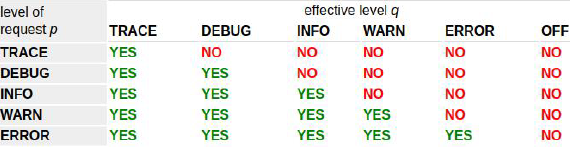
1. Sistema de logs

També he afegit les librerias necessàries per poder utilitzar logs i he configurat l'arxiu XML per personalitzar-ho. En aquest arxiu he posat que els logs es guardin en un fitxer de text i que surtin per consola.

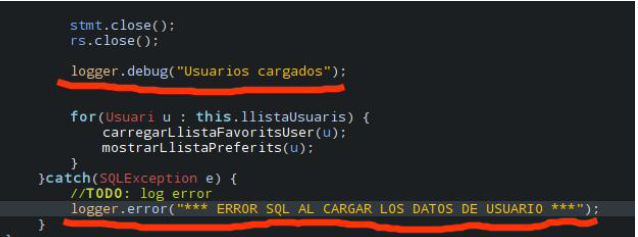


Solament hem de modificar aquests valors, si volem solament escriure'ls en fitxer podem comentar la linea que té un appender “stdout” (que és la que específica que es mostrin per consola).

Aquests són els nivells de log que ens proporciona la libreria de logback:



Es pot especificar en el XML (on surt root level=”\*debug”) que nivell de logs volem mostrar/guardar, si solament ens interessen els errors, posarem el nivell a “ERROR”. Àdhuc no he posat gairebé cap log, solament he implementat el sistema, aquí una petita mostra de com funciona:



Posem logger.nivell-de-importància-del-log(“missatge”);

En el fitxer log4j-\*application.log podem veure els logs guardats.



1. Canvis 24/04/2018

He començat a fer algun test i he hagut de fer alguns canvis. He pensat bastant sobre com testear els DAO ja que les seves funcions es basen en trucades a la base de dades, però no tenen molta mes lògica, per la qual cosa no tènia sentit fer un mockObject (el qual simula una tornada de dades i s'aplica la lògica sobre aquestes dades). He mirat diferents opcions per internet però sembla ser que és un tema complicat, hi ha gent que diu que el referent a la base de dades se sol testear en les proves d'integració i

uns altres que també s'ha de fer unit testing.

Al final he optat pel següent:

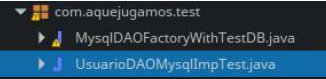
-Tenir una base de dades en local que sigui una còpia de l'original (no en dades, solament en estructura), on puguem fer les proves sabent les dades que hi ha en ella i sense que puguin haver-hi efectes col·laterals en la base de dades original.

Per a això, he creat una classe que s'encarregarà de crear connexions Mysql a la

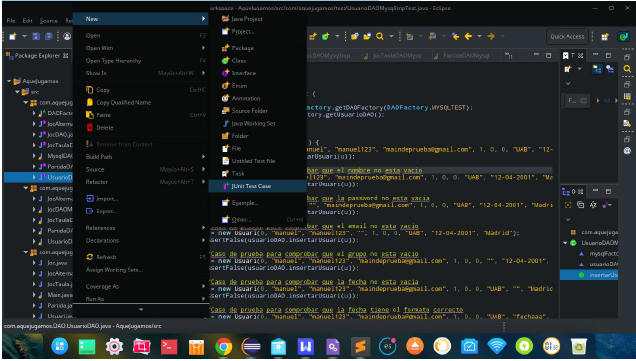
base de dades de testeig:



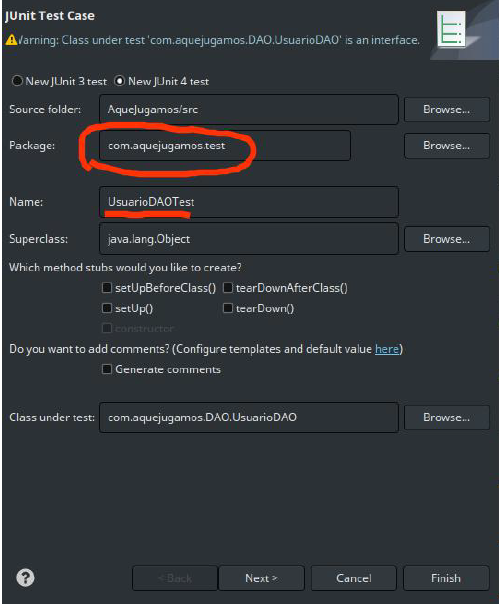
I per a cada classe DAO d'implementació (les que contenen Imp) farem un fitxer de testing amb Junit (de moment solament he començat el de Usuari)



Per crear els fitxers de testing hem de seleccionar amb clic dret sobre la classe a testear i seleccionar Junit Test Casi:

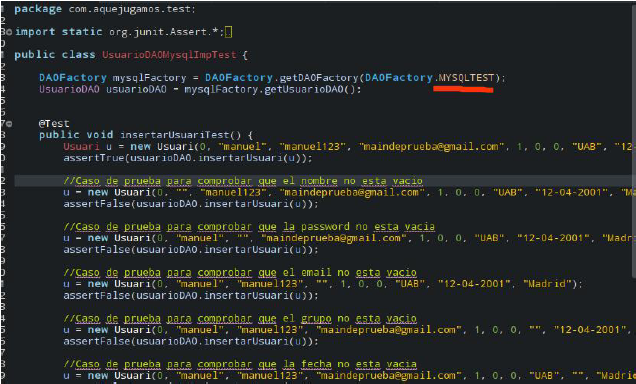


I ens sortirà una finestra com aquesta:



Hem de seleccionar que ho posarem en el paquet de test, i com a convenció podem cridar als fitxers de test igual que el fitxer a testear afegint un “Test” al final.

Finalment ja podem implementar els test casi:

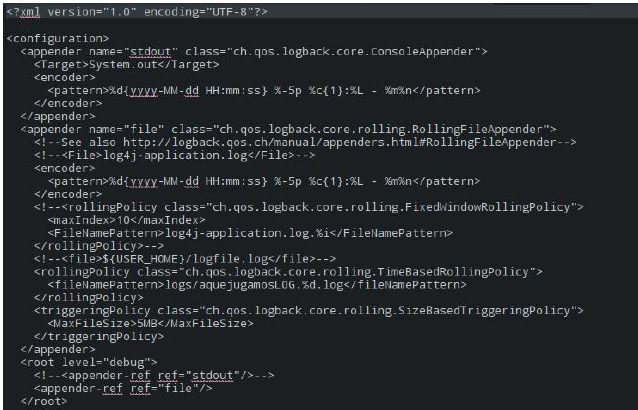


Hem d'indicar que usarem la base de dades de test, i per a cada funció de la classe que testegem, creem una funció de test, on posarem tots els casos a cobrir, i compararem si les dades que retorna concorden amb el que hauria de retornar. Si fem clic dret sobre el fitxer de test, i li donem a Run as < Junit Test, s'executarà el test.



En aquest cas surt en vermell, indicant que no ha passat el test. Això és a causa que en el codi àdhuc no tenim implementades les validacions de les dades que introdueix l'usuari, per la qual cosa de moment el test troba els errors.

Finalment he fet uns petits canvis en el sistema de logs:



He modificat el XML per guardar un arxiu log en una carpeta anomenada logs, on es crearà un arxiu de log diferent per cada dia, i cada fitxer tindrà com a nom aquejugamosLOG”\*fechaDeCuandoSeCreo”.log, per la qual cosa podrem buscar els logs per dia mes fàcilment.