EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

MÉDIA- ÉS OKTATÁSINFORMATIKAI TANSZÉK

TESZTVEZÉRELT FEJLESZTÉS BEMUTATÁSA EGY ÁLTALÁNOS CÉLÚ WEBES KERETRENDSZEREN

Abonyi-Tóth Andor

Egyetemi tanársegéd

Karácsony Máté

Programtervező informatikus Bsc,

Nappali tagozat

Budapest, 2011

<Témabejelentő helye>

Tartalom

1. Bevezetés

1.1. Tesztvezérelt fejlesztés

Az elmúlt évtizedekben számos fejlesztési módszertant hoztak létre minőségi szoftvertermékek előállítására. Ezek közül napjainkban egyre népszerűbb a *tesztvezérelt fejlesztés*[1], melynek fő célja, hogy hibamentes, változástűrő forráskódot állítsunk elő.

Alkalmazása során két fő kódbázissal bír egy projekt: a produkciós kód, mely a valódi terméket alkotja, illetve a hozzá készülő tesztkód, melynek célja a produkciós kód automatizált ellenőrzése. Munkafolyamata három fő tevékenység köré épül: tesztek írása, a teszteket teljesítő produkciós kód elkészítése, illetve e kódok újraszervezésére (refaktorálása), mely során átláthatóbbá tesszük az elkészült forráskódot. A tesztek újrafuttatásával meggyőződhetünk róla, hogy az utolsó fázis végeztével is működőképes maradt az adott szoftverkomponens, és funkcionalitása sem változott. A folyamat lépéseit az alábbi ábra foglalja össze:

<Nincs megrajzolva!>

1. ábra: a tesztvezérelt fejlesztés folyamata

[1] Test Driven Development (TDD)

1.2. A tesztek típusai

A teszteket alapvetően három típusba sorolhatjuk ebben a témakörben:

- A legsűrűbben és legtöbbször végrehajtott, egy-egy osztály publikus metódusainak tesztelését végző kódokat *egységteszteknek*[2] nevezzük. Egy osztályhoz általában egy teszt-osztály tartozik, egy metódushoz pedig egy vagy több teszt-metódus. Dolgozatom főként ezzel a típussal foglalkozik.
- Az *integrációs tesztek*[3] több osztály vagy alrendszer összehangolt működését ellenőrzik.
- A *funkcionális tesztek*, vagy *elfogadási tesztek*[4] azt hivatottak biztosítani, hogy a program funkcionalitása az elvárásoknak megfelelő.

1.3. A dolgozat felépítése

[2] unit test
[3] integration test
[4] acceptance test

2. Felhasználói dokumentáció

2.1. Rendszerkövetelmények

A keretrendszer és a bemutató alkalmazás azonos rendszerkövetelményekkel rendelkeznek, melyek az alábbiak:

- PHP 5.3.0, vagy újabb verzió (ajánlott 5.3.5)
- Bármilyen kompatibilis web szerver illetve operációs rendszer
- A tesztek futtatásához: PHPUnit[5]
- A kód-lefedettség jelentésekhez: Xdebug[6] PHP-kiegészítés

[5] http://www.phpunit.de/
[6] http://www.xdebug.org/

3. Fejlesztői dokumentáció

4. Összegzés

5. Irodalomjegyzék

6. Mellékletek

I. számú melléklet: Tesztek listája

Osztály: *fw\ClassLoader*

- ✓ Register
- ✓ Unregister
- ✓ Auto register
- ✓ Load existing class in global namespace
- ✓ Load existing class in global namespace as slash
- ✓ Load existing class in default namespace
- ✓ Class outside default namespace will not be loaded
- ✓ Exception thrown on nonexistent class

Osztály: *fw\KeyValueStorage*

- ✓ Does not have nonexistent key
- ✓ Get returns null for nonexistent key
- ✓ Get with default value
- ✓ Get with magic method
- ✓ Set with magic method
- ✓ Set transforms array
- ✓ Set returns storage
- ✓ Has works on array wrapper
- ✓ Get works on array wrapper
- ✓ Set works on array wrapper
- ✓ Can be converted to array

Osztály: *fw\config\Configuration*

- ✓ Active section can be changed and trimmed
- ✓ Active section cant be changed to invalid name
- ✓ Different section data are independent
- ✓ Merge

Osztály: *fw\config\FileBasedConfiguration*

- ✓ Construction with missing file throws exception

Osztály: *fw\config\IniConfiguration*

- ✓ Constructor throws exception on incorrect ini file
- ✓ Constructor throws exception on invalid section name
- ✓ Constructor throws exception on missing parent section
- ✓ Can parse valid ini file
- ✓ Section inheritance

Osztály: *fw\config\XmlConfiguration*

- ✓ Constructor throws exception on incorrect xml file
- ✓ Constructor throws exception on invalid xml file
- ✓ Constructor throws exception on missing parent section
- ✓ Can parse valid xml file
- ✓ Section inheritance

Osztály: *fw\log\Log*

- ✓ New log has no log target
- ✓ Log has target after adding one
- ✓ Adding same log target twice results in added once
- ✓ Log have zero targets after all removed
- ✓ Writing error invokes write on targets
- ✓ Writing warning invokes write on targets
- ✓ Writing info invokes write on targets
- ✓ Debug disabled by default
- ✓ Writing debug does not invoke write on targets when disabled
- ✓ Writing debug invokes write on targets when enabled
- ✓ Adding target with not matching level will not log
- ✓ Adding target with invalid level throws exception

Osztály: *fw\log\LogTarget*

- ✓ Formatting works with default format string
- ✓ Formatting works with custom format strings

Osztály: *fw\log\FileTarget*

- ✓ Constructor throws exception when permission denied
- ✓ Writes formatted line with line feed to file

Osztály: fw\log\OutputTarget

- ✓ Writes formatted line with line feed to output

Osztály: *fw\control\RouteInfo*

- ✓ Controller name can be set
- ✓ Action name can be set
- ✓ Constructor works with array parameters
- ✓ Constructor works with key value storage parameters
- ✓ Constructor creates key value storage on incorrect parameters