



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.....	2
1.1. Tesztvezérelt fejlesztés.....	2
1.2. A tesztek típusai	4
1.3. Keretrendszer és bemutató alkalmazás.....	4
1.4. A dolgozat felépítése.....	5
2. Felhasználói dokumentáció.....	6
2.1. Rendszerkövetelmények.....	6
2.2. Bemutató alkalmazás.....	6
2.3. Keretrendszer	11
3. Fejlesztői dokumentáció.....	12
3.1. Bemutató alkalmazás.....	12
3.2. Keretrendszer	12
3.2.1. Könyvtárszerkezet.....	12
3.2.2 A fő névtér (<i>\fw</i>) osztályai	13
3.2.3 Konfiguráció (<i>\fw\config</i>).....	14
3.2.4 Naplózás (<i>\fw\log</i>).....	15
3.2.?. Kiterjesztési lehetőségek.....	17
4. Összegzés.....	18
4.1.....	18
4.2. Továbbfejlesztési lehetőségek.....	18
5. Irodalomjegyzék.....	19
6. Mellékletek	20
6.1. Ábrák listája	20
6.2. Tesztek listája.....	20

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 teszt kód, melynek célja a produkciós kód automatizált ellenőrzése. A két kódbázis karbantartása pazarlásnak tűnhet, de a megfelelően létrehozott és karbantartott automatizált tesztek jelentős előnyökkel járnak:

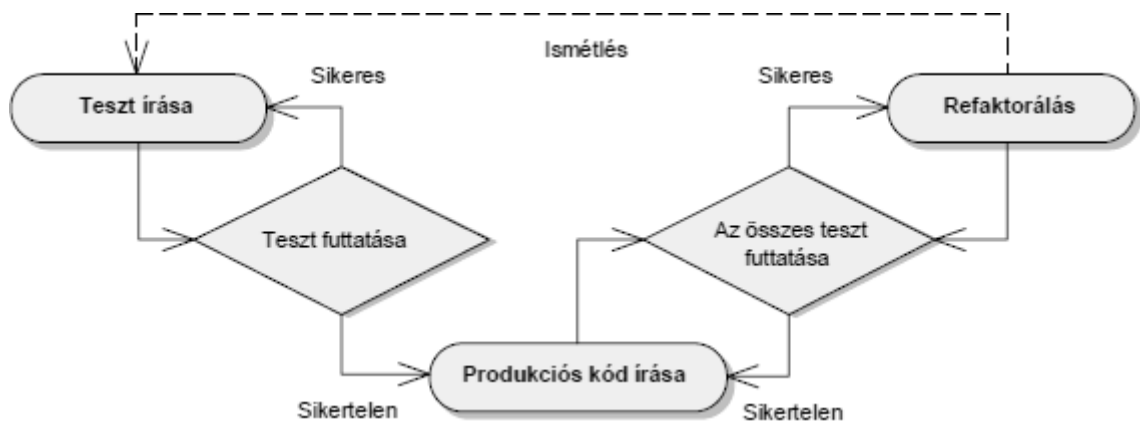
- A tesztek futtatása nem vesz el időt a fejlesztőtől (a manuális teszteléshez képest), és akárhányszor megismételhetők, önellenőrzők (azaz futásuk eredménye egyértelműen sikeres vagy sikertelen, nem szükséges az eredmény utólagos vizsgálata)
- A legtöbb esetben a hibakeresésre fordított idő jelentősen redukálódik, hiszen egy nem teljesülő teszt lokalizálja a hibás kódrészt
- Könnyebb új fejlesztőket bevonni a csapatmunkába, mert a tesztek védőhálóként funkcionálnak, így nem fognak új hibákat bevezetni régebbi komponensekben
- Nem alakul ki a kód egyes részeinek megváltoztatásától való „félelem” a fejlesztőkben (főleg hosszú távú projekteknél fontos)

A munkafolyamat ciklikus, és négy fő tevékenység köré épül: (a) tesztek írása, (b) a tesztek teljesítő produkciós kód elkészítése, (c) a tesztek futtatására, illetve (d) a kódok újraszervezésére (refaktorálása), mely során átláthatóbbá tesszük az elkészült forráskódot, illetve szerkezeti javításokat végzünk rajta, mely végső soron emeli a kód minőségét, megkönnyíti megértését, ezáltal a csapatmunkát. Fontos, hogy a tesztek a produkciós kód előtt írjuk, hiszen így fogják befolyásolni annak alakulását,

¹ Test Driven Development (TDD)

megírásukhoz pedig a rendelkezésre álló követelményspecifikációkat és használati eseteket² kell figyelembe venni. A tesztek újrafuttatásával bármikor meggyőződhetünk róla, hogy az utolsó fázis végeztével is működőképes maradt az adott szoftverkomponens, funkcionálitása nem változott. A fejlesztési folyamat lépései az alábbiak [1]:

1. Írjunk egy tesztet
2. Tegyük lefordíthatóvá
3. Ellenőrizzük, hogy futtatása sikertelen (*piros csík*³)
4. Írjuk meg a produkciós kódot, mely átmegy a teszten
5. Futtassuk újra a tesztet, ha sikertelen, térjünk vissza az előző lépésre, ha sikeres (*zöld csík*⁴), haladjunk tovább
6. Refaktoráljuk az elkészült teszt- és produkciós kódot
7. Ellenőrizzük, hogy még mindig sikeresen lefut minden teszt (ha nem, a 4. lépéstől folytassuk)
8. Ismételjük a lépéseket előről, amíg elkészül a kívánt funkcionális



1. ábra: A tesztvezérelt fejlesztés folyamata

Az angol terminológia szerinti „piros csík” illetve „zöld csík” kifejezések a tesztek futtatásához használt keretrendszerek jellemző grafikus eredménykijelzéseire utalnak. Dolgozatomban a *PHPUnit* [2] teszt-keretrendszert használom, mely szöveges kimenettel rendelkezik. Egy teszt futási eredményét az alábbi karakterek egyikével jelzi:

² use case

³ red bar

⁴ green bar

- . (pont karakter): sikeres lefutás
- F (fault): sikertelen lefutás
- E (error): sikertelen lefutás fordítási (PHP nyelv esetében értelmezési) hiba miatt
- S (skipped): átugrott teszt (nem futtatható le valamely függőség hiányában)
- I (incomplete): befejezetlennek jelölt teszt

Egy tesztcsomag futtatásának eredménye pedig a fenti karakterek sorozata, és ha minden teszt sikeres, csak pont karakterekből (.) áll. Ezt a zöld, minden egyéb esetet a piros csík megjelenésének tekintünk.

1.2. A tesztek típusai

A tesztek alapvetően három típusba sorolhatjuk ebben a témakörben:

- A legsűrűbben és legtöbbször végrehajtott, egy-egy osztály publikus metódusainak tesztelését végző kódokat *egységteszteknek*⁵ 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*⁶ több osztály vagy alrendszer összehangolt működését ellenőrzik.
- A *funkcionális tesztek*, vagy *elfogadási tesztek*⁷ azt hivatottak biztosítani, hogy a program funkcionalitása az elvárásoknak megfelelő. (Ezek gyakran felhasználói felület tesztek, melyek automatizálása meglehetősen körülményes, de ezt is be fogom mutatni.)

1.3. Keretrendszer és bemutató alkalmazás

A dolgozathoz készült szoftver egy PHP nyelven írt webes keretrendszer. A választásom azért esett erre a nyelvre, mert rendkívül népszerű, mivel könnyen tanulható, így ideális környezet nyújt a tesztvezérelt fejlesztés bemutatására. Az elkészült keretrendszer az alapvető vezérlési funkcióján túl alábbi feladatokhoz nyújt segítséget: **IDE MÉG KÉNE VALAMI, HOGY ÁTCSÚSSZON A FELSOROLÁS**

- konfiguráció

⁵ unit test

⁶ integration test

⁷ acceptance test

- naplózás
- input-validáció
- autentikáció és munkamenet-kezelés
- távoli eljárás hívás támogatása különböző protokollokon

A keretrendszerhez készítettem egy bemutató alkalmazást is, hogy könnyebben megérthetővé tegyem működését. Ez az alkalmazás egy egyszerű feladatlista-karbantartó program, használatát és funkcionalitását a 2. fejezetben fejtettem ki.

1.4. A dolgozat felépítése

A következő fejezetben ismertetem a kifejlesztett keretrendszer és a bemutató alkalmazás rendszerkövetelményeit, telepítését, és használatát. Kitérek a *PHPUnit* segítségével írt tesztek felépítésére, illetve futtatásának módjára is.

A fejlesztői dokumentációban bemutatom a legfontosabb komponensek működését és egymáshoz való viszonyát, továbbá ismertetem keretrendszer kiterjesztési lehetőségeit.

2. Felhasználói dokumentáció

2.1. Rendszerkövetelmények

A következő szoftverek a keretrendszer és a bemutató alkalmazás használatához, és a tesztek futtatásához is szükségesek:

- PHP 5.3.0, vagy újabb verzió (ajánlott 5.3.5, vagy újabb)
- Bármilyen kompatibilis webservert és operációs rendszert (ajánlott Apache HTTP Server)
- A tesztek futtatásához: PHPUnit, illetve VfsStream PEAR-csomagok
- A kód-lefedettség jelentésekhez: Xdebug PHP-kiegészítés
- A teljesítménymérési adatok megtekintéséhez: WinCacheGrind vagy KCacheGrind (ajánlott ez utóbbi bővebb tudása miatt)

A bemutató alkalmazás működéséhez, használatához szükséges további eszközök:

- MySQL adatbázis szerver (ajánlott 5.1 verzió, vagy újabb)
- A felhasználói felület tesztek futtatásához: Selenium Server (JRE szükséges)
- Bármely korszerű, Selenium-kompatibilis webböngésző (Firefox 4, Safari 5, Internet Explorer 9)

A letöltések és telepítési útmutatók hivatkozásainak listáját az „6.1. Letöltési linkek és telepítési útmutatók” melléklet tartalmazza. Ezen felül a dolgozat mellékleteként beadott DVD-lemez tartalmaz egy előtelepített virtuális gépet, melyen megtalálható a fent említett komponensek mindegyike. A gép a *VirtualBox* virtualizációs környezettel használható, melynek telepítői Windows és Linux platformokra szintén helyet kaptak a lemezen. **MAILT NÉZNI, HOGY JÖHET-E KIEGÉSZÍTÉS**

2.2. Bemutató alkalmazás

A keretrendszer bemutatására egy egyszerű feladat- vagy tevékenység-lista karbantartó webes alkalmazást készítettem. A felhasználási esetek úgy lettek kialakítva, hogy kihasználják a keretrendszer minden lényeges, bemutatandó elemét. A fő funkciók a következők:

- Be- és kijelentkezés, regisztráció
- Saját és publikus feladatlista megtekintése (lapozással, 5 soronként)
- Új feladat hozzáadása, saját feladatok szerkesztése, törlése
- Publikus feladatok és felhasználói adataik megtekintése

Az alkalmazást a virtuális gép asztalán található „*Bemutató alkalmazás*” ikon segítségével indíthatjuk el. Ekkor elindul az alapértelmezett webböngésző, és betöltődik a „*http://szakdolgozat/*” lokális webszerver által kiszolgált oldal. Az alábbi képernyőfotók helytakarékosági okokból nem tartalmazzák a böngészőablak keretét és eszköztárait.

The screenshot shows a login form titled "BEJELENTKEZÉS". It contains two input fields: "Felhasználónév:" and "Jelszó:". Below the fields are two buttons: "Bejelentkezés" and "Regisztráció". The "Regisztráció" button is highlighted in blue.

2. ábra: Az alkalmazás indító képernyője

Amennyiben a felhasználó nem rendelkezik érvényes belépési azonosítóval, a „*Regisztráció*” linkre kattintva készíthet magának egyet:

The screenshot shows a registration form titled "REGISZTRÁCIÓ". It contains several input fields: "Felhasználónév:" (filled with "teszt"), "Teljes név:" (filled with "Teszt Felhasználó"), "E-mail cím:" (filled with "felhasznalo@teszt.hu"), "Jelszó:" (filled with 8 dots), and "Jelszó ismét:" (filled with 8 dots). Below the fields are two buttons: "Regisztráció" and "Bejelentkezés". The "Regisztráció" button is highlighted in blue.

3. ábra: Új belépési azonosító létrehozása — regisztrációs képernyő

Minden mező kitöltése kötelező. Az alkalmazás ellenőrzi a mezők tartalmát az űrlap elküldésekor, és tájékoztatja a felhasználót az esetleges kitöltési hibákról (ez az alkalmazásban szereplő összes űrlapra igaz). A regisztráció után az alkalmazás visszavigyeli a felhasználót egy link segítségével a bejelentkezés oldalra. Sikeres belépés után a következő képernyő jelenik meg:

Bejelentkezve: *Teszt Felhasználó*
[Kijelentkezés](#)

FELADATOK

Saját feladatok

[Új feladat](#)

Cím	Kezdés	Befejezés	Prioritás	Publikus	Műveletek
Nincs megjelenítendő elem.					

1 / 1 oldal

Más felhasználók megosztott feladatai

Cím	Kezdés	Befejezés	Prioritás	Felhasználó
Tavaszi szünet	2011. 04. 18.	2011. 04. 26.	alacsony	Karácsony Máté
Tesztelés	2011. 04. 25.	2011. 05. 12.	normál	Kovács János
Bemutató program tesztelése	2011. 05. 02.	2011. 05. 13.	normál	Gipsz Jakab
Szakdolgozat leadása	2011. 05. 16.	2011. 05. 16.	magas	Karácsony Máté
Utolsó ZH	2011. 05. 16.	2011. 05. 16.	normál	Karácsony Máté

1 / 3 oldal

[következő oldal »](#)

4. ábra: Feladatok listája

A bejelentkezés után minden további képernyőn elérhető az alkalmazás jobb felső sarkában a „*Kijelentkezés*” link, mellyel a felhasználó biztonságosan lezárhatja munkamenetét. A feladatok listája két részre bontható: felül a felhasználó saját feladatai jelennek meg (ez esetben még üres, hiszen új felhasználóról van szó), alatta pedig más felhasználók publikusként megjelölt feladatainak listáját láthatjuk. A listák mindig maximum 5 sort tartalmaznak, több elem esetén az alattuk található „*előző oldal*” és „*következő oldal*” linkekkel lapozhatók, egymástól függetlenül. Az adatok kezdési időpont szerinti növekvő sorrendben rendezve jelennek meg.

Feladatot hozzáadni a felső táblázat felett baloldalon elhelyezett „*Új feladat*” linkre kattintva lehet kezdeményezni. Az ekkor megjelenő űrlapot az 5. ábra mutatja. A „*Leírás*” mező opcionális, és többsoros szöveget is tartalmazhat. Az összes többi mezőt kötelező kitölteni. A dátum mezők elfogadott formátuma „*ÉÉÉÉ. HH. NN.*”, ahol *É*

az évszám, *H* a hónap, *N* az adott nap egy számjegye. Az űrlap elküldés után ezt a formátumot is ellenőrzi, illetve hogy a befejezés dátuma nem korábbi a kezdésnél. A *JavaScript* támogatással rendelkező böngészőkben a dátumok bevitelét egy külön komponens segíti, mely az adott dátum mezőbe lépéskor jelenik meg (6. ábra).

Bejelentkezve: *Teszt Felhasználó*
[Kijelentkezés](#)

ÚJ FELADAT

Cím:

Leírás:

Kezdés:

Befejezés:

Prioritás: ☐ alacsony
☒ normál
☐ magas

Publikus: ☒ igen

vagy [vissza a listához](#)

5. ábra: Új feladat hozzáadása űrlap

Kezdés:

Befejezés:

Prioritás: ☒ normál

Publikus: ☒ igen

2011. Április

H	K	Sze	Cs	P	Szo	V
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

6. ábra: Dátumok bevitelét segítő komponens

Ha a feladat „*Publikus*” tulajdonsággal kerül elmentésre, meg fog jelenni más felhasználók számára is, a feladatlista alsó táblázatában (de szerkeszteni vagy törölni nem tudják). Ha valamely mező hibásan lett kitöltve, azt az adott mező alatt megjelenő hibaüzenetek jelzik. A feladat sikeres elmentése után a program két lehetőséget kínál a felhasználó számára: megtekintheti a frissen elkészült bejegyzést, vagy visszatérhet a feladatlistához.

Bejelentkezve: *Teszt Felhasználó*
[Kijelentkezés](#)

FELADAT MEGTEKINTÉSE

Cím: Próba
Leírás: Tesztként használt
többsoros leírású
teendő.
Kezdés: 2011. 05. 11.
Befejezés: 2011. 05. 12.
Prioritás: normál
Publikus: igen

[Vissza a listához](#)

7. ábra: Az újonnan elkészült feladat adatlapja

Miután visszatértünk a feladatlistához látható, hogy az új feladat bekerült a felső táblázatba. Adatait módosítani a sor jobb szélén, „Műveletek” oszlopfejléc alatt található „szerkesztés”, eltávolítani pedig a „törlés” link segítségével tudjuk.

Bejelentkezve: *Teszt Felhasználó*
[Kijelentkezés](#)

FELADATOK

Saját feladatok

[Új feladat](#)

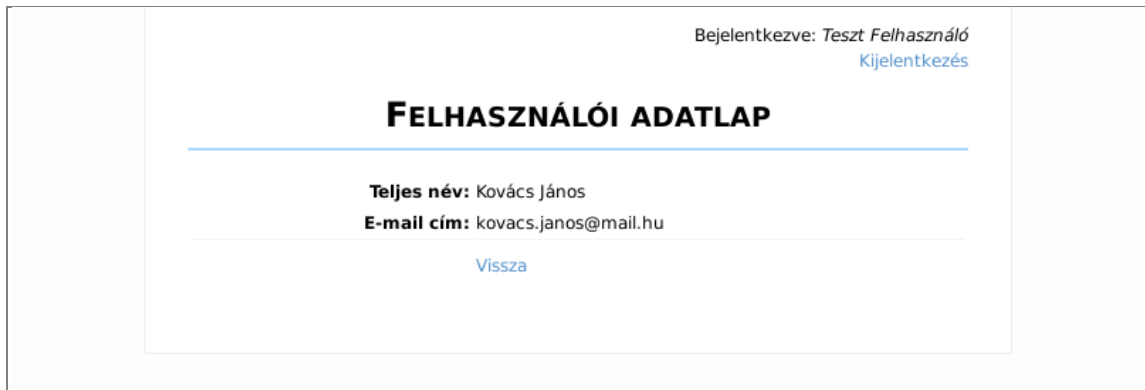
Cím	Kezdés	Befejezés	Prioritás	Publikus	Műveletek
Próba	2011. 05. 11.	2011. 05. 12.	normál	igen	szerkesztés törlés

1 / 1 oldal

8. ábra: Szerkesztés és törlés linkek a „Saját feladatok” táblázatban

A szerkesztési képernyő egy ugyanolyan űrlapot tartalmaz, mint amelyet az új feladat hozzáadásánál láthattunk, azzal a különbséggel, hogy a mezők előre ki vannak töltve a kiválasztott feladat tulajdonságaival. A törlés linkre kattintva a program megerősítést kér a felhasználótól a feladat tényleges törléséhez.

A feladatlista „Cím” oszlopában szereplő szöveges linkek mind az adott feladat adatlapjára visznek (lásd 7. ábra). Az alsó táblázatban („*Más felhasználók megosztott feladatai*”) egy felhasználó nevére kattintva az adatlapjára kerülünk, mely tartalmazza teljes nevét és regisztrációnál megadott e-mail címét.



The screenshot shows a web interface for a user profile. At the top right, it says 'Bejelentkezve: Teszt Felhasználó' with a blue link 'Kijelentkezés' below it. The main heading is 'FELHASZNÁLÓI ADATLAP' in bold, underlined. Below this, the user's full name 'Teljes név: Kovács János' and email 'E-mail cím: kovacs.janos@mail.hu' are displayed. At the bottom, there is a blue link 'Vissza'.

9. ábra: Felhasználói adatlap

Az alkalmazás használatának befejezésekor a böngészőablak bezárása előtt érdemes a „Kijelentkezés” linkre kattintani, mert hatására munkamenetünk valóban megsemmisül az alkalmazásszerveren is, így illetéktelenek semmiképpen nem vehetik azt igénybe.

2.3. Keretrendszer

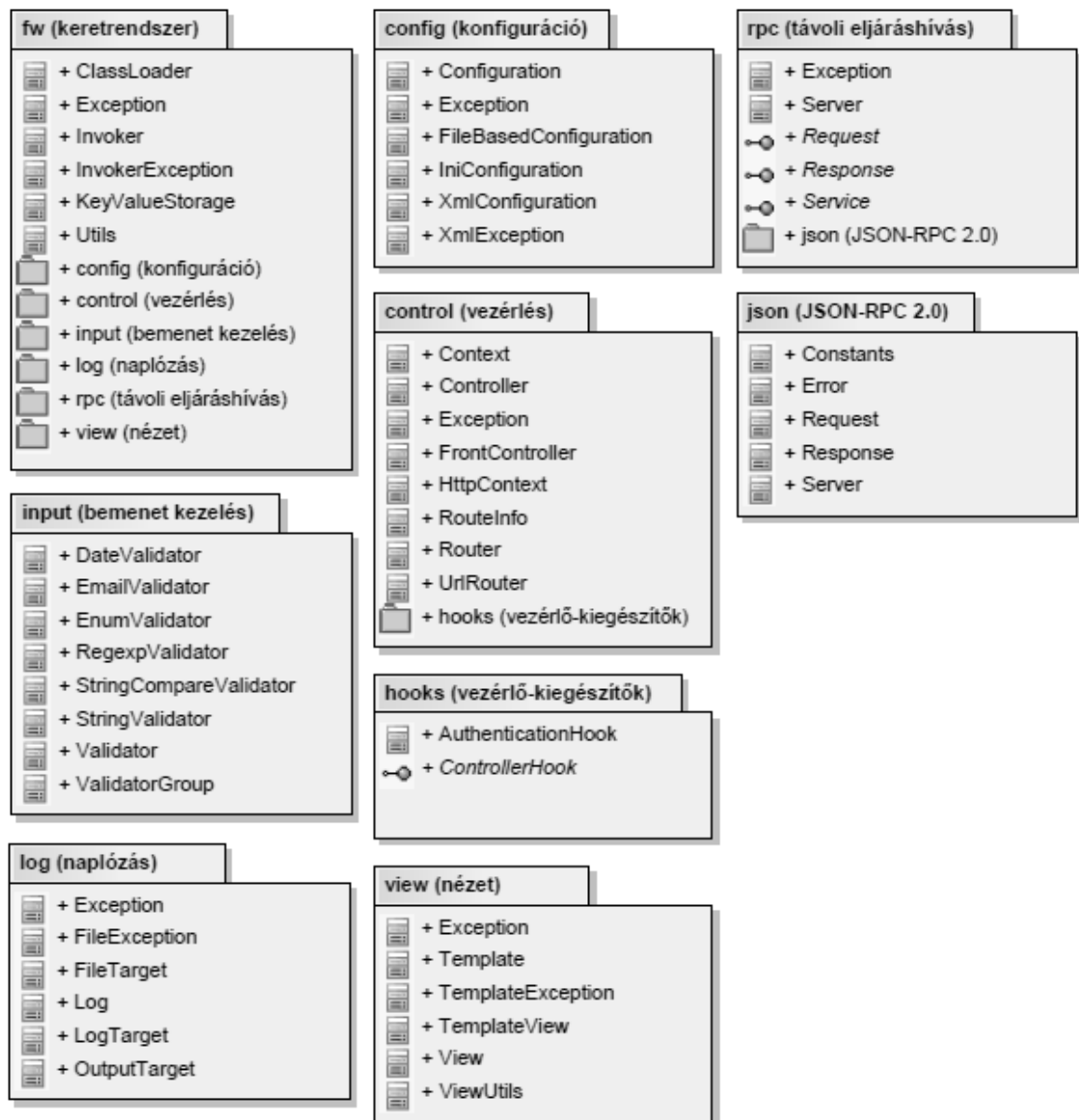
3. Fejlesztői dokumentáció

3.1. Bemutató alkalmazás

3.2. Keretrendszer

3.2.1. KÖNYVTÁRSZERKEZET

A keretrendszer főkönyvtára (*framework*) három alkönyvtárt tartalmaz. A *classes* könyvtárban találhatók meg a keretrendszert alkotó osztályok és interfészek. Egyben ez a könyvtár tartalmazza a fő névteret (*\fw*). Minden további névtér ennek egy alkönyvtárában foglal helyet. Struktúráját az alábbi ábra szemlélteti:



A *coverage* könyvtár a *PHPUnit* által generált kódlefedettségi jelentést tartalmazza. Ez egy speciális, HTML formátumú dokumentum, mely minden keretrendszer-osztály minden metódusához tartalmazza, hogy mely tesztek mely végrehajtási ágait járták be. A *tests* könyvtár és alkönyvtárai a fejlesztés során elkészült egység- és integrációs teszteket tartalmazzák. Ennek a könyvtárnak a szerkezete lényegében megegyezik a *classes* könyvtár szerkezetével. Az egyetlen eltérés az *assets* alkönyvtár jelenléte, melyben a tesztekhez szükséges segédosztályok, konfigurációk, adatok kaptak helyet.

TESTDOX KÖNYVTÁR KI NE MARADJON

KELL IDE VALAMI SABLON, HOGY HOGYAN ÍROM LE A DOLGOKAT ITT ALATTA

KÉNE ÍRNI ÚGY ÁLTALÁBAN ARRÓL HOGY (M)VC TÍPUSÚ AZ FW

3.2.2 A FŐ NÉVTÉR (\FW) OSZTÁLYAI

`\fw\Exception`

A nyelvbe beépített kivétel osztály közvetlen leszármazottja. A keretrendszer minden további kivételosztályának ősosztálya. Célja, hogy a keretrendszer kivételei megkülönböztethetők legyenek a ráépített egyéb alkalmazásosztályokétól. Sem konstanssal, adattaggal vagy művelettel nem egészíti ki ősosztályát.

`\fw\ClassLoader`

`\fw\KeyValueStorage`

`\fw\Invoker`

`\fw\InvokerException`

`\fw\SessionManager`

`\fw\Utils`

3.2.3 KONFIGURÁCIÓ (\FW\CONFIG)

A web-alkalmazások számára különösen fontos, hogy jól konfigurálhatóak legyenek. Ez többnyire abból fakad, hogy jelentős eltérések lehetnek a különböző szerverek kialakításában, illetve kimondottan fejlesztési és tesztelési célokra fenntartott környezetek között is gyorsan kell tudni áthelyezni őket. Ennek legegyszerűbb módja, ha a komponensek konfigurációját fájlokba helyezzük. A keretrendszerben implementált konfiguráció-kezelés háromféle formátumot támogat: PHP tömböket és objektumokat, illetve INI és XML fájlokat képes kezelni. Ezen kívül a futtatási környezetek közti áthelyezést az is segít, hogy a konfigurációk szekciókra osztottak, melyek örökölhetik, illetve felülírhatják egymás beállításait. Így egyszerűen az aktív szekció megadásával választhatunk, hogy melyik környezetbe alkalmas konfigurációt használja az alkalmazás. **KÉNE IDE PÉLDA A SEKCIÓK MEGMAGYARÁZÁSÁRA?**

`\fw\config\Exception`

`\fw\config\Configuration`

`\fw\config\FileBasedConfiguration`

`\fw\config\IniConfiguration`

`\fw\config\XmlConfiguration`

`\fw\config\XmlException`

IDE (VAGY EGYEL FELJEBB) LE KELL ÍRNI HOL A SÉMA AMIT VALIDÁLOK MEG KELL A
KÉP A SÉMA SZERKEZETRŐL!

3.2.4 NAPLÓZÁS (\FW\LOG)

`\fw\log\Exception`

`\fw\log\Log`

`\fw\log\LogTarget`

`\fw\log\OutputTarget`

`\fw\log\FileTarget`

`\fw\log\FileException`

3.2.?. KITERJESZTÉSI LEHETŐSÉGEK

3.2.?. TELJESÍTMÉNY, OPTIMALIZÁCIÓ

4. Összegzés

4.1.

4.2. Továbbfejlesztési lehetőségek

ITT MOST CSAK KERETRENDSZER VAGY MEGINT SZEDJEM SZÉT? AZTHISZEM UTÓBBI

5. Irodalomjegyzék

- [1] Becoming Agile: Test a little, code a little - a practical introduction to TDD,
<http://danbunea.blogspot.com/2005/12/test-little-code-little-practical.htm>
(2011. április)
- [2] The PHP Unit Testing framework
<https://github.com/sebastianbergmann/phpunit/>
(2011. április)

6. Mellékletek

6.1. Letöltési linkek és telepítési útmutatók

Az alábbi hivatkozások a 2011. május hónap szerinti állapotot tükrözik.

➤ **Apache HTTP Server:** <http://httpd.apache.org/>

➤ **PHP:** <http://php.net/>

➤

➤

➤

➤

➤

➤

➤

➤

➤

➤

➤

➤

6.2. Ábrák listája

6.3. 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