Eötvös Loránd Tudományegyetem

Informatikai Kar

A PROJEKTMENEDZSMENT JÁTÉKOSÍTÁSA

Témavezető:

Dr. Illés Zoltán

egyetemi docens

Készítette:

Győri Tibor

programtervező informatikus

BSc, 2007, nappali tagozat

Budapest, 2012.

Tartalomjegyzék

Bevezetés

Mi is az a PlayByWork?

Egy projektmenedzsment szoftver. Az egyik cégnél egy elterjedt szoftvert használtunk a feladatok kezelésére, és nekem nagyon megtetszett. Feltelepítettem saját használatra is egyet, azonban a telepítés nagyon körülményes volt, és egy újratelepítés során el is veszett az egész. A célom az volt, hogy egy minden tekintetben egyszerű alkalmazást hozzak létre. Olyat, ami egyszerűen telepíthető, egyszerűen használható, de minden alapvető igényt kielégít.

Szerettem volna egy olyan projektmenedzsment szoftvert készíteni, ami PHP alapokra épül, és akár SQLite adatbázissal is képes dolgozni, mindemellett kellemes a szemnek, és felhasználóbarát is.

A munka során törekedtem rá, hogy használható maradjon régebbi típusú böngészőkkel is, de a

maximális élményt természetesen a modern böngészők nyújtják, melyek képesek megjeleníteni a CSS3-as stílusokat is.

Emellett a portál további célja, hogy a munkát összekösse a játékkal, ezzel hatékonyabbá, és élvezetesebbé téve a munkavégzést.

Elsősorban kis- és középvállalkozások részére ajánlott a használata, de igény szerint skálázható nagyobb mértékű felhasználáshoz is.

Felhasználói dokumentáció

Telepítési útmutatás

Előfeltételek

A telepítéshez szükség van egy webszerverre. Ez platformfüggetlen, ugyanúgy lehet Windows-os is, mint Linux-os alapú.

A webszerveren legyen elérhető a PHP minimum 5.1-es verziója, PDO támogatással a megfelelő adatbázishoz. Támogatott adatbázisok: SQLite3, MySQL, PostgreSQL.

A ***data*** és ***logs*** könyvtárakra biztosítani kell az írási jogot a webszervernek. Előbbi az adatbázis írhatósága miatt szükséges (SQLite adatbázis esetén), utóbbi pedig a naplófájlok írhatósága miatt.

Szükséges módosítások

A működéshez módosítani kell a ***config/config.xml*** és a ***.htaccess*** fájlokat.

config.xml

- **database**
  - **connectionString**: Ide kell beírni a PDO formátumú kapcsolati string-et, amivel az adatbázishoz szeretnénk kapcsolódni.
    - SQLite esetén:
      - sqlite:utvonal/fajl.nev
    - MySQL esetén:
      - mysql:dbname=<database_name>;host=<host_name>[;port=<port>]
    - PostgreSQL esetén:
      - pgsql:dbname=<database_name>;host=<host_name>[;port=<port>]
  - **username**: Az adatbázishoz való kapcsolódáshoz szükséges felhasználónév.
  - **password**: Az adatbázishoz való kapcsolódáshoz szükséges jelszó.

- **datafile**: A kezdőadatokat, táblák definícióját tartalmazó SQL fájl elérési helye.

❖ **language**: A használni kívánt nyelv. A ***translations*** könyvtárban kell hogy legyen egy *<language>.csv* fájl, ami a fordításokat tartalmazza. Az alapértelmezett érték az **en**, ami az angol kezelőfelületet hozza elő.

❖ **loggingMode**: Ennek értéke a naplózás módját befolyásolja. Ha *debug*-ra van állítva, akkor a rendszer-szintű hibaüzenetek (pl. PDOException-ök) nem csak a naplófájlba kerülnek, hanem a kezelőfelületen is megjelennek a hibaüzenetek között. Éles környezetben érdemes üresen hagyni.

.htaccess

Ebben a fájlban csak a RewriteBase-t kell átírnunk a domain-névhez képesti relatív elérési útvonalra.

Ha pl. saját domain címről fut a honlap, akkor ez **/** lesz, ha pedig egy aloldalról, akkor **/aloldal**.

Ez a beállítás biztosítja a tiszta URL-ek használatát.

Adatbázisfájl

Az adatbázis létrehozására és feltöltésére alapértelmezetten a ***config/initialize.sql*** fájl szolgál. Ennek módosításával lehet kompatibilissé tenni más adatbázis típusokhoz, illetve itt lehet megadni az alapértelmezett adatokat is.

Nyelvi fájlok

Ha az angoltól eltérő nyelven kívánjuk használni a programot, szükséges létrehozni egy a nyelvhez tartozó fordítási fájlt. A fájl nevének a ***config.xml*** fájlban megadott névnek kell lennie, kiterjesztése pedig ***csv***.

Minden sorban két adat szerepel, idézőjelek közé zárva, pontosvesszővel elválasztva. Az első a fordítandó szöveg, a második pedig a lefordított. Alapul az ***en.csv*** fájlt lehet venni.

Példa egy fordítási sorra: ”Users”;”Felhasználók”

Használati útmutatás

A használathoz javaslom, hogy lehetőleg modern böngészőt használjunk (pl. Google Chrome, Mozilla Firefox).

Az útmutatót az angol nyelvű felhasználói felület alapján írom.

Bejelentkezés/Kijelentkezés

A **Login** gombra kattintva előugrik a bejelentkező űrlap. Itt a **Username** mezőjébe a felhasználói nevet, míg a **Password** mezőjébe a hozzá tartozó jelszót kell megadnunk. Ezután az űrlapon belüli **Login** gombra kattintva megtörténik a bejelentkezés. Amennyiben hibásan írtunk be valamit, felugró hibaüzenetben kapunk róla tájékoztatást.

Bejelentkezés után a **Login** gomb helyén egy **Logout** gombot találunk. Erre kattintva kijelentkezhetünk a rendszerből, és visszakerülünk a nyitólapra.

Home oldal

Bejelentkezés után itt láthatjuk a hamarosan lejáró feladatainkat, dátum szerinti növekvő sorrendben. A feladat nevére kattintva átugorhatunk a részletes információs lapjára.

A feladatok lapozható formában kerülnek listázásra. Amennyiben a beállított értéknél (alapértelmezetten 10) több elem kerül a táblázatba, azokat a táblázat felett megjelenő lapozó használatával érhetjük el.

Admin-ként itt találhatjuk a **Recreate database** gombot is, amellyel az adatbázis újratelepítését végezhetjük el. Ekkor minden újonnan létrehozott adat elveszik.

Users oldal

Itt láthatjuk az oldal felhasználóit, összesített pontszám szerinti csökkenő sorrendben felsorolva.

A nevekre kattintva átkerülhetünk az adatlapokra.

A **Create new user** gombbal megfelelő jogosultság esetén létrehozhatunk új felhasználókat.

Felhasználói adatlapok

Az adott felhasználó projektjeit sorolja fel, illetve a projektenkénti pontszámát.

A projekt nevére kattintva átkerülünk a projekt adatlapjára.

Ezen az oldalon további menüpontokat érhetünk el. Az **Edit user** az éppen megtekintett felhasználó szerkesztését teszi lehetővé, míg a **Delete user** a törlését. Fontos megjegyezni, hogy saját magunkat nem törölhetjük a rendszerből, és hogy ezeket a menüpontokat csak magas hozzáférési szintű felhasználók érhetik el (pl. **admin**).

Új felhasználó létrehozása

A **Create new user** gombbal érhetjük el ezt a funkciót. Egy űrlapot láthatunk.

A **User name** mezőbe a kívánt felhasználónevet kell megadnunk, amit a belépésnél szeretnénk használni.

A **Password** és **Password again** mezőkbe a jelszót kell beírnunk. Azért kell ezt kétszer megtennünk, hogy nehogy egy félregépelt jelszó miatt hibásan hozzuk létre a felhasználót.

A **Full name** mezőbe a felhasználó nevét kell megadnunk. Ez a név lesz látható a többi felhasználó számára.

Az **E-mail** mezőbe a felhasználó e-mail címét kell megadni.

A **Role** lenyíló szövegdobozból a kívánt jogkört kell kiválasztanunk. Ez fogja meghatározni, hogy a

felhasználó milyen szolgáltatásokat érhet el az oldalon.

Végül a **Create user** gombra kattintva véglegesíthetjük a létrehozást. Ekkor még előugorhat egy hibaüzenet, ha valamit nem megfelelően töltöttünk ki. Ezt az **OK** gombbal bezárva, javítsuk ki a hibákat, majd próbálkozzunk újra.

Sikeres létrehozás esetén az új felhasználó adatlapjára kerülünk.

Felhasználó szerkesztése

Az új felhasználó létrehozásához hasonló oldalt kapunk, csak itt már ki van töltve az űrlap az aktuális értékekkel.

A jelszót csak akkor kell megadni, ha meg szeretnénk változtatni, ekkor ugyanúgy kétszer be kell írni, mint amikor új felhasználót hozunk létre. Egyéb esetben üresen kell hagyni, hogy változatlan maradjon.

A változtatások mentéséhez kattintsunk a **Save changes** gombra. Ekkor, ha helyesen töltöttük ki az űrlapot, a módosított felhasználó adatlapjára érkezünk, és egy üzenetben tájékoztatást kapunk a sikeres módosításról.

Felhasználó törlése

Ennek a menüpontnak nincs külön felülete, a sikeres törlésről rendszerüzenetben értesülünk, miután visszakerültünk az alap **Users** oldalra.

Projects oldal

Ezen az oldalon azokat a projekteket láthatjuk kilistázva, amelyekhez hozzá vagyunk rendelve. Admin-ként **Other projects** címszó alatt az egyéb projektek is listázásra kerülnek, hogy könnyebben kezelhessük azokat.

A projektek nevére kattintva azok adatlapjára kerülünk.

A **Create new project** gombbal új projektet hozhatunk létre, amennyiben van hozzá jogosultságunk.

Új projekt létrehozása

A **Project name** mezőbe a projekt elnevezését kell megadni. Ez fog szerepelni a listázásoknál.

A **Project identifier** mezőbe egy rövid elnevezést kell beírnunk. Ez csak betűket, számokat, kötőjelet és aláhúzás jelet tartalmazhat.

Az **Add users to the project** sorban a legördülő listából választhatjuk ki azokat a felhasználókat, akiket hozzá szeretnénk rendelni a projekthez. A kiválasztás után az **Add** gombra kattintva kerülnek hozzárendelésre. Amennyiben egy hozzárendelt felhasználót szeretnénk inkább eltávolítani, a neve melletti **Remove** gombra kell kattintanunk.

A **Description** szövegdobozba a projekt egy rövid leírását adhatjuk meg. Ez az adatlapján jelenik meg, és tájékoztatásként szolgál a felhasználóknak, hogy miről is szól a projekt.

A **Create project** gombra kattintva, amennyiben minden mezőt megfelelően kitöltöttünk, létrejön az új projekt, és annak az adatlapján találjuk magunkat.

Projekt adatlap

A projekt neve alatt a leírását láthatjuk. Ez alatt a hozzárendelt felhasználók kerülnek felsorolásra a projekten belüli összegzett pontszámaikkal, majd

pedig a projekthez tartozó feladatokat láthatjuk egy táblázatba szedve. A táblázat a kezdőlapihoz hasonlóan működik, nagy mennyiségű feladat esetén lapozhatóvá válik.

A felhasználók, illetve feladatok nevére kattintva azok adatlapjára kerülünk.

Az almenüben megtalálhatjuk a projekt szerkesztését lehetővé tevő **Edit project** gombot, a törlésért felelős **Delete project** gombot, és a **Create new task** gombot, amely az új feladat létrehozó képernyőre visz minket annyi kikötéssel, hogy a projekt fixen be lesz állítva az aktuálisra, és nem is módosítható.

Amennyiben másik projekthez kívánunk feladatot létrehozni, át kell fáradni az adott projekt adatlapjára, vagy pedig a **Tasks** oldalon belülről kell új feladatot létrehozni. Utóbbi lehetőség nem minden felhasználó számára elérhető.

Projekt szerkesztése

Az új projekt létrehozásához hasonló űrlapot láthatunk, amelynek a mezői a jelenlegi értékekkel vannak kitöltve.

Amennyiben egy hozzárendelt felhasználót kívánunk törölni a projektből, a neve melletti **Remove** gombra kattintva tehetjük ezt meg.

A változtatásokat a **Save changes** gombbal menthetjük el. Sikeres mentés után a módosított projekt adatlapjára kerülünk.

Projekt törlése

Külön felülettel nem rendelkezik, a sikeres törlésről rendszerüzenetben értesülünk, és a **Projects** főoldalra kerülünk.

Tasks oldal

A hozzánk rendelt feladatokat láthatjuk kilistázva. A táblázat a kezdőlapihoz hasonlóan lapozható nagyobb adatmennyiség esetén. A feladatok nevére kattintva azok adatlapjára kerülünk.

A **Create new task** menüpontra kattintva új feladatot hozhatunk létre.

Új feladat létrehozása

A **Title** mezőbe a feladat címét kell leírnunk. Ez lesz látható a honlap többi részén a listákban. Tömören kell utalnia a feladat tartalmára.

A **Project** lenyíló menüből azt a projektet kell kiválasztanunk, amelyikhez a feladat tartozik.

A **Type** mező a feladat típusát határozza meg. Alapértelmezetten a ***Feature***, ***Bug*** és ***Support*** típusok közül választhatunk.

A **Status** mező értéke a feladat státuszát tartalmazza. Ez a legtöbb esetben ***New*** lesz új

feladat létrehozásakor. A ***New*** és ***Open*** státuszok még be nem fejezett feladatokat jelölnek, míg a többi már befejezetteket.

Az **Assign to** listából azt a felhasználót kell kiválasztanunk, akire a feladatot szeretnénk bízni. Alapértelmezetten mi magunk vagyunk kiválasztva. Ennek a felhasználónak a listáiban meg fog jelenni a feladat, mint teljesítendő.

A **Due date** mezőibe a feladat megoldásának határidejét kell beírni év-hónap-nap formátumban. Alapértelmezetten a holnapi dátum van kiválasztva.

A **Description** szövegdobozba a feladat leírását kell megadnunk. Ez fontos, hogy elég részletes és érthető legyen, hogy mindenki munkáját megkönnyítsük, és a későbbiekben is egyértelműen lássuk, hogy mi volt a feladat.

A **Create task** gombra kattintva, amennyiben helyesen kitöltöttünk minden mezőt, létrejön az új feladat, és átkerülünk az adatlapjára.

A feladatok adatlapja

A feladat neve alatt a következőket láthatjuk felsorolva: státusz, típus, projekt, hozzárendelt felhasználó, illetve hogy ki hozta létre.

Itt a projekt, illetve a felhasználók nevére kattintva azok adatlapjára kerülünk.

Az adatok alatt a leírás szerepel.

Az almenüben a szerkesztésre szolgáló **Edit task** gomb, és a törlést végző **Delete task** gomb érhető el.

Feladat szerkesztése

Az új feladat létrehozásához hasonló űrlapot láthatunk, melynek mezői az aktuális értékekkel vannak kitöltve.

A **Save changes** gombra kattintva, amennyiben helyesen töltöttünk ki minden mezőt, a módosítások mentésre kerülnek, és a feladat adatlapjára kerülünk.

Feladat törlése

Külön felülete nincs, a törlés sikerességéről rendszerüzenet formájában értesülünk, és a **Tasks** főoldalára kerülünk.

Fejlesztői dokumentáció

Megoldási terv

A megoldáshoz egy keretrendszert hoztam létre, hogy áttekinthetőbb legyen a kód működése, és könnyen bővíthető legyen. Ezt a keretrendszert sikerült annyira általánosan megoldani, hogy könnyen átültethető más honlaprendszerek alapjává is.

URL-kezelés

A ***.htaccess*** fájl átirányít minden kérést az ***index.php***-nak. Az index fájlban megkeressük a honlap alap URL-jét. Ezt úgy tehetjük meg, hogy a $_SERVER[„PHP_SELF”] változóból megkapjuk az ***index.php*** domain-relatív címét, és töröljük belőle a fájl nevét. Így például egy domain root helyen megkapjuk, hogy az alap URL a **„/”**.

A továbbiakban betöltjük a működéshez szükséges segédfájlokat, amelyek osztályokat és függvényeket tartalmaznak. Ide tartozik a fordítást végző ***classes/translator.php***, a naplózást végző ***classes/logger.php***, az adatbázist kezelő ***classes/database.php***, a bejelentkezést, és felhasználói adatokat számontartó ***classes/authentication.php***, illetve a vezérlők alapjául szolgáló ***controllers/baseController.php*** fájl is. Ezek működésére a későbbiekben bővebben kitérek.

A ***config/config.xml*** fájlból a konfigurációs adatokat a PHP-s SimpleXMLElement segítségével olvashatjuk ki.

A **$_SERVER[„REQUEST_URI”]** változót elemezve megkapjuk, hogy melyik vezérlő melyik metódusát kell meghívni. A szükséges vezérlő osztályát tartalmazó fájlt betöltjük, majd a paramétereket egy tömbként átadva neki meghívjuk az adott metódust a **call_user_func_array** függvénnyel.

Amennyiben hiányzik a vezérlő, vagy nincs olyan metódusa, amit meg szeretnénk hívni, esetleg a felhasználónak nincs jogosultsága megtekinteni azt, az **indexController indexMethod** metódusát hívjuk meg helyette. A jogosultság ellenőrzése az adatbázis alapján történik.

A vezérlők felépítése tehát azzal a logikával kell, hogy megtörténjen, hogy a hívások <vezérlő_neve>/<metódus_neve>[/param1[/…]] formában érkeznek. Így tehát a vezérlőt tartalmazó

fájl neve ***<vezérlő_neve>Controller.php*** lesz, a meghívandó metódusok neve pedig **<metódus_neve>Method([$param1[, …]])**.

Adatbázis-kezelés

Az adatbázis kezelését a ***classes/database.php*** fájl végzi. A konfigurációs fájlból vett értékek alapján a konstruktorban létrehozza a PDO segítségével a kapcsolatot az adatbázissal. Egy **query** metódussal biztosítja a biztonságos SQL lekérdezéseket. A kéréseket így például a következő formában lehet küldeni:

query(„SELECT * FROM vmi WHERE vmi1=?”, array($vmi1));

A lekérdezés eredményét egy változóba mentjük, és visszatérési értékként továbbítjuk. Az egészet egy try-catch blokkba téve biztosítjuk, hogy ne szálljon el a rendszer egy hibás lekérdezés esetén. Az esetleges hibaüzenet a naplófájlba kerül kiírásra.

Itt biztosítunk módot az SQL fájlok importálására is. Ennek segítségével tudjuk a kezdeti adatbázist is létrehozni.

Továbbá itt találunk egy segédfüggvényt is, amely a legutóbb beszúrt sor automatikusan generált azonosítóját adja meg. Ennek új tartalmak generálásakor vesszük hasznát.

Naplózás

A naplózást a ***classes/logger.php*** fájl végzi. Lehetőséget biztosít rá, hogy egy naplófájlba írhassunk hibaüzeneteket, illetve a konfigurációs fájltól függően ezeket a felhasználó felületen is megjelenítheti.

Erre egy újabb metódus segítségével képes. Egy Session változót hozunk létre, amely tömbként tárolja a kezelőfelületen megjelenítendő üzeneteket. A tömbben eltároljuk a megjelenítendő üzenetet, és a típusát, ami a megjelenését fogja befolyásolni. Megadható típusok: „error”, „warning”, „notice”. Ezek sorrendben piros, narancssárga, és zöld üzenetként jelennek meg a felhasználó számára.

Fordító

A fordítást a ***classes/translator.php*** fájlban található **t($str, $args)** függvény végzi.

A konfigurációs fájlból megállapítja a nyelvet, majd megkeresi az ehhez tartozó fordítási fájlt (***translations/<megadott_nyelv>.csv***).

Ezt a PHP **fgetcsv** függvénye segítségével soronként beolvassa, és ha a fordítandó szöveggel egyezést talál az első oszlopban, akkor a második oszlop tartalmát transzformálva a **vsprintf** függvénnyel, visszatér a lefordított szöveggel.

A **vsprintf** segítségével formázott szövegeket is fordíthatunk, amelyek %d, %s, stb. formában tartalmazzák a kapott paramétereket.

Autentikáció

Az azonosítást a ***classes/authentication.php*** fájl végzi. Ez egy egyke mintájú osztály, ami a Session változók segítségével jegyzi meg a szükséges információkat.

Itt végezhetjük el a be- és kijelentkezést is, melyek során az adatbázissal történő kommunikáció után beállítjuk a szükséges Session változókat.

A jogosultságok ellenőrzéséhez is biztosít metódust. Egy **hasAccessTo($controller, $method)** formában érkező lekérés alapján ellenőrzi az adatbázisban, hogy az aktuálisan bejelentkezett felhasználónak van-e jogosultsága megtekinteni a kérdezett párosítást.

Továbbá segédmetódusokat biztosít az osztályváltozók elérésére. Például a felhasználó egyedi azonosítóját kérhetjük el, vagy a bejelentkezettség állapotát.

Template-kezelés

A template-kezelést a ***classes/pageBuilder.php*** fájl végzi. Biztosítja a szkriptek, és stílusfájlok kiírásának módját, és a template-fájl kiírását.

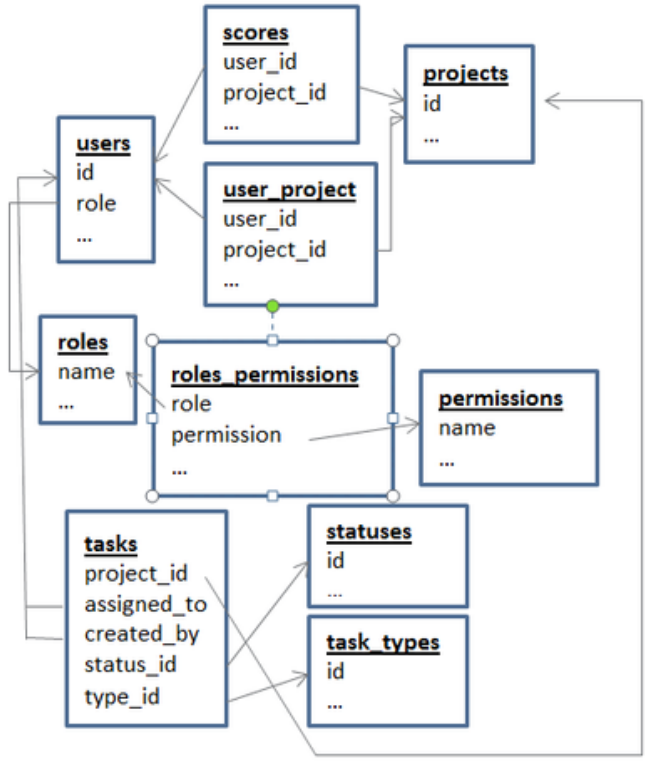
A template fájl nagyrészt HTML kódból áll, de azért előfordul benne PHP kód is, ami a megjelenítéshez szükséges.

A **pageBuilder** osztály **buildPage** metódusa úgy van kialakítva, hogy többféle template fájlt is kezelni tudjon, paraméterként megadhatjuk, hogy épp melyiket akarjuk kiíratni. A projekt működéséhez azonban elég volt egy darab template fájl is, aminek csak a **content** *div*-beli tartalma változik elsősorban a megjelenítendő oldaltól függően.

A ***classes/widgets.php*** segédosztály a menügombok megjelenítését teszi egyszerűbbé, de további elemekkel is bővíthető.

Megvalósítás

Adatbázis kapcsolatok



Vezérlők

baseController

A vezérlők ősosztálya. Egy osztályváltozót biztosít a template-kezelőnek, a konstruktorban ezt létre is hozza. Ezen kívül egy alap index metódust definiál, amelyben átirányítja a felhasználót a kezdőlapra, amennyiben a leszármazottnak nem lenne ilyen metódusa.

indexController

indexMethod

Létrehoz egy gombot, amivel az adatbázist építhetjük újra, majd a createDashboard() függvény visszatérési értékével meghívja a template-kezelő osztályt.

A **CreateDashboard** függvény a főoldal tartalmát állítja elő. Bejelentkezett felhasználó esetén a dashboard-ot építi fel, míg vendég esetén egy üdvözlő szöveget ír ki.

createdbMethod

Az adatbázis újralétrehozását végzi a config fájl database/datafile tulajdonsága alapján, amennyiben a felhasználónak admin jogosultsága van. Egyéb esetben hibaüzenettel a kezdőoldalra lép.

authController

Itt a login- és logoutMethod meghívja az Authentication osztály megfelelő függvényét a be- és kiléptetéshez.

projectsController

indexMethod

Kirajzolja az új projekt létrehozása gombot, majd felépíti az oldalt a listProjects függvény segítségével.

idMethod

A paraméterül kapott azonosító alapján megmutatja a keresett projekt adatait a listTasks függvény segítségével. A getName, getTasks és

getProjectUsers függvények értelemszerűen a projekthez tartozó adatokat kérik le az adatbázisból, és egy tömb formájában térnek vissza, a getName pedig egy string-gel.

newMethod

Létrehozza az új projekt létrehozásához szükséges űrlapot.

saveMethod

A **$_POST** változóban kapott űrlapelemek értékei alapján létrehoz egy új bejegyzést az adatbázisban az új projektről.

Amennyiben az id paraméter is meg van adva, egy már meglévő bejegyzés adatait frissíti a kapott értékekkel.

editMethod

A szerkesztéshez szükséges űrlapot hozza létre és tölti ki a kapott azonosítójú projekt adatai alapján.

deleteMethod

A kapott azonosítójú projektet törli az adatbázisból.

tasksController

indexMethod

Létrehozza az új feladat gombot, és kilistázza a már meglévő feladatokat, amelyek a bejelentkezett felhasználóhoz vannak rendelve, a listTasks függvény segítségével.

listMethod

Egy alias-t biztosít az indexMethod elérésére.

idMethod

A paraméterként kapott azonosítójú feladat adatait jeleníti meg a showTask függvény segítségével. A getTitle függvény az adatbázisból lekérdezi a feladat címét.

newMethod

Új feladat létrehozásához biztosít egy űrlapot. Amennyiben paraméterben kap egy azonosítót, a projektek lenyíló menüjét lekorlátozza a megadott projektre. A határidőt alapértelmezetten az aktuális napot követő napra állítja, a hozzárendelt felhasználót pedig az aktuális felhasználóra.

saveMethod

A létrehozni kívánt feladat adatait menti el az adatbázisba. Amennyiben paraméterként kap egy azonosítót, úgy egy meglévő feladat adatait módosítja a **$_POST** változó tartalma alapján.

editMethod

A paraméterben kapott azonosítójú feladat szerkesztését teszi lehetővé egy kitöltött űrlapon keresztül.

deleteMethod

A paraméterül kapott azonosítójú feladatot törli az adatbázisból.

usersController

indexMethod

Kirajzolja az új felhasználó létrehozását lehetővé tevő gombot, majd a listUsers függvény segítségével listázza a regisztrált felhasználókat az összesített pontszámukkal. A getUsers függvény a felhasználók listáját kérdezi le az adatbázisból.

idMethod

A paraméterül kapott azonosítóval rendelkező felhasználó adatait jeleníti meg a showUser függvény segítségével. A getName függvény a felhasználó nevét kérdezi le az adatbázisból.

newMethod

Az új felhasználó létrehozásához szükséges űrlapot hozza létre.

saveMethod

A **$_POST** tömbben kapott adatok alapján hoz létre új felhasználót az adatbázisban, vagy frissít egy már

meglévőt, az azonosító paraméter kitöltöttségétől függően.

editMethod

A paraméterül kapott azonosítójú felhasználó adatainak módosításához szükséges űrlapot hozza létre. Amennyiben nincs megadva azonosító, az aktuális felhasználó adatainak módosítását teszi lehetővé.

deleteMethod

A paraméterül kapott azonosítóval rendelkező felhasználó törlését végzi el az adatbázisból.

Javascript (main.js)

A Javascript (JS) használata során nagy hasznát vettem a jQuery keretrendszernek. Ezáltal sokkal könnyebben manipulálhatóak az oldal elemei.

$(document).ready

Az oldal betöltődésekor Javascript-ből elrejtem a bejelentkező ablakot. Erre azért van szükség, hogy ha egy felhasználó JS nélküli böngészőből érkezik az oldalra, akkor is elérhető maradjon számára a bejelentkezés.

Ezután meghívom a createPagers függvényt, ami a lapozást teszi lehetővé a listákon.

createPagers

Elrejti az összes sort, majd megjeleníti az első **limit** darabot. A **limit** változó a szkriptfájl elején került beállításra. Ezután az oldalon található listákon egyenként megnézi, hogy mennyi sort tartalmaznak. A sorok számát osztjuk a megjelenítendő sorok számával. Ezt felfelé

kerekítve megkapjuk az oldalak számát. Ha ez több, mint 1, akkor létrehozzuk a lapozót.

showTasks(offset, obj)

A kapott **obj** linkhez tartozó táblázat elemeit módosítja, hogy az **offset** sorszámútól jelenítsen meg **limit** darabot.

loginPopup

A bejelentkező ablakot jeleníti meg, és a kurzort a felhasználóvén mezőjére helyezi.

ajaxLogin

Elküldi a bejelentkezési adatokat az authController-nek, majd a sikeres bejelentkezést követő üzenetet megkapva újratölti az aktuális oldalt. Hiba esetén megjeleníti a kapott hibaüzenetet.

ajaxLogout

Meghívja az authController kijelentkeztető metódusát, majd a sikeres kijelentkezést követő üzenetet megkapva újratölti az aktuális oldalt. Hiba esetén megjeleníti a kapott hibaüzenetet.

setError(msg)

Az **msg** üzenetet megjeleníti a hibaüzenet-ablakban.

checkNew[User/Project/Task]

Elvégzik a szükséges ellenőrzéseket az űrlapokon, és ha nem találtak hibát, továbbítják azokat. Hiba esetén üzenettel jeleznek a felhasználó számára.

addUserToProject

Az új projekt létrehozásánál lévő felhasználó-hozzárendelést teszi lehetővé.

removeParent(obj)

Az előző függvényben hozzáadott sorok eltávolítását teszi lehetővé. Ezen kívül máshol is felhasználható, annyit tesz, hogy az **obj** objektum szülőelemét eltávolítja a DOM-ból.

Tesztelés

A tesztelés során érdemes odafigyelni a szándékos felhasználói rontásokra. Amennyiben egy azonosítót váró metódust azonosító nélkül hívunk meg, annak erre fel kell készülnie. Továbbá az oldalak gombok nélkül is elérhetőek az URL ismerete alapján, ezek elérését is kezelnünk kell a bejelentkezett felhasználó jogosultsága alapján.

Az SQL injekciót a PDO használata kiküszöböli, így ennek a veszélye nem áll fent.

A kitöltetlen/hibásan kitöltött adatokat szkript ellenőrzi.

A létrehozható tartalmak nevét és leírását a PHP **htmlentities** függvényének segítségével jelenítem meg, hogy ilyen módon se lehessen belepiszkálni a tartalomba.

Összefoglalás

Sikerült tehát egy olyan projektmenedzser szoftvert létrehoznom, amely az alapvető felhasználási céloknak megfelel, és igény szerint könnyedén tovább bővíthető.

Továbbá a létrehozott keretrendszer átültethető egyéb honlaprendszerek alapjává is. Ezzel lehetővé tehetünk egy egyszerű vezérlő-kezelést, adatbázis-elérést, hozzáférés-szabályozást és template-kezelést.

A rendszer fejlesztéssel skálázható. Az alapértelmezett SQLite adatbázis-motor lecserélésével gyorsítható, a string-ek lefordításával többnyelvűsíthető, illetve további funkciókkal bővíthető.

Fejlesztési lehetőségek

Egy plugin menedzselő beépítésével személyre szabottan lehetne funkciókkal bővíteni a szoftvert.

További játékosítási elemekkel még hatékonyabbá tehető a munka. Többek közt egy jelvény-rendszer, vagy egy személyes avatárfigura nevelgetése munka által megfelelő elemek lehetnek.

Irodalomjegyzék

http://www.php.net/manual/en [2011. november]

http://api.jquery.com [2011. november]

http://gamification.org [2011. június]