Importance of project-based learning in software development

Gy. Schrauf

MH vitéz Szurmay Sándor Budapest Helyőrség Dandár, Budaörsi street 49-53, Budapest 1118, Hungary, mh.bhd@hm.gov.hu

Abstract

This article presents a software development project that can be used in large companies. Design, problem solving, decision making and creating a project product were essential to solving the specified requirements. Using the constructivist pedagogical method, it was possible to implement an application which makes information flow easier and serves its communication in point of employee. In this type of development clearly shows well that the knowledge which acquired in the university, how can be supplemented in accordance with the requirements of the labor market.

Keywords: *constructivist pedagogical method; project task; SharePoint*

A projekt alapú tanulás fontossága a szoftverfejlesztésben

Schrauf Gy.

MH vitéz Szurmay Sándor Budapest Helyőrség Dandár, Budaörsi út 49-53, Budapest, 1118, Magyarország, mh.bhd@hm.gov.hu

Absztrakt

A cikkben egy nagyvállalati környezetben is felhasználható szoftverfejlesztési projektmunka kerül bemutatásra. A feladat megoldása során elengedhetetlen volt a tervezés, a problémamegoldás, a döntéshozás és egy projekttermék létrehozása. A konstruktivista pedagógiai módszert alkalmazva lehetőség adódott arra, hogy egy olyan alkalmazás kerüljön megvalósításra, amely alkalmas az információ áramlás megkönnyítésére és közlésére egy dolgozó állomány tekintetében. Az ilyen típusú fejlesztés jól mutatja, hogy a munkaerőpiac elvárásainak megfelelően milyen módon egészíthető ki az iskolában szerzett tudás.

Kulcsszavak: konstruktivista pedagógia; projektfeladat; SharePoint

1. Bevezető

Napjainkban a pedagógia tudományában egyre inkább megjelenik az iskolában szerzett és a munkaerőpiacon elvárt tudás különbségének vizsgálata. A munkaadók leginkább olyan munkaerőt keresnek, akik professzionálisan dolgoznak, ismerik a vállalati kultúrát, jó kommunikáció készséggel rendelkeznek, magasszintű problémamegoldó készséggel rendelkeznek és hatékonyan tudnak csapatban dolgozni (Kővári, 2019). A projekt alapú szoftverfejlesztés megoldást jelenthet az elméleti és alkalmazott tudás különbségének áthidalására. (Márta H., 2010) (Ilter, 2014) (Boaler, 2002)

Egy felmerült igény szerint a nagyvállalti környezetben hatékonyabb működést eredményezhet, ha egy központi felületen lehetne elérni az összes osztályhoz tartozó információt. A vállalatok számára nagy probléma, hogy a havi, vagy akár a napi munkavégzésükhöz szükséges információk, adatok lassan érnek el egyik helyről a másikra. Az információáramlást szeretnék felgyorsítani, központosítani. További elvárásként megfogalmazható a levelező rendszerek tehermentesítése is, mivel körlevelek küldésekor, a levélhez csatolt dokumentumok akár több kilobájt vagy akár több megabájt méretűek is lehetnek, végsősoron az üzenet több ezer példányban haladhat át a vállalat hálózati eszközein, infrastruktúráján. A több ezer levélhez csatolt dokumentumoknak összértéke így több gigabájt méretben terheli a levelező szervereket és a hálózatot.

Korábban kialakításra került egy szolgáltatás, Hirdetmények 1.0 megnevezéssel, amit be is vezettek egy 2010-es SharePoint farmon. Ezt a szolgáltatást szeretnék tovább fejleszteni és átmigrálni egy újabb, gyorsabb, fejlettebb 2013-as SharePoint farm környezetre. A Hirdetmény 1.0 szolgáltatás azonban nem a legoptimálisabban lett kialakítva. Az információáramlást gyorsította, a levelező szolgáltatást pedig tehermentesítette ugyan, ellenben nem teljesen automatizált a folyamatok végrehajtása. A 2013-as SharePoint farmra való átállást nagymértékben hátráltatja a dokumentációk, a folyamatábrák, a telepítő scriptek hiánya.

A Hirdetmény 1.0 szolgáltatás folyamatainak megismerésében és kialakításában részt vett informatikus állománnyal átgondoltan újra terveztük az új szolgáltatást, amit a továbbiakban Hirdetmény 2.0 néven ismerhetünk meg. A Hirdetmény 2.0 keretein belül a régi szolgáltatásnak megfelelően, de már az új 2013-as SharePoint farmon, egy központi webhelyen kerül kialakításra három lista a felhasználók felé közlendő információkkal.

1.1. Osztály adatok lista

A lista célja, hogy a korábbi egyedi megoldások kiváltásaként, minden osztály naprakész adatokat szolgáltat az osztályról (teljes név, osztály címe, vezető neve, beosztása) és centralizáltan szerepelteti egy központi, mindenki által elérhető helyen. Előnye a listának, hogy a korábban redundánsan szereplő adatokkal ellentétben ezen a felületen könnyen hozzáférhetők és szükség szerint frissíthetők. A közzétett adatok rögzítését, módosítását csak az osztály részéről a központi webhelyen beregisztrált tartalomszerkesztő és tartalomjóváhagyó (továbbiakban együtt tartalommegosztó) végezheti. Ezen személyek egy listában, a szerkesztő és jóváhagyók listában kerülnek regisztrálásra az üzemeltetők által.

Ezzel egyidejűleg a rendszer automatikusan e-mailben értesítheti a felhasználókat az osztály adatainak változásairól, amennyiben sürgős az információ azonnali megosztása.

1.2. Szerkesztő és Jóváhagyók lista

Tulajdonképpen ez a lista szolgál arra, hogy a központi webhelyen regisztrálja a tartalommegosztókat, akik hirdetményeket helyezhetnek el és publikálhatják azt. Illetve az osztály adatok listában az osztályukhoz tartozó listaelemet módosíthatják és publikálhatják mindenki számára. Másodlagos célja, hogy a háttérben futó folyamatok e lista alapján döntik el, hogy van e jogosultsága az aktuális felhasználónak új hirdetmény feltöltésére, illetve az osztály adatok listában az osztályhoz tartozó listaelem módosítására. Előnye a listának, ha egy felhasználó szeretne hirdetményt publikáltatni, akkor a lista segítségével könnyedén megtalálja az osztályához tartozó személyeket, akiknek kérheti a segítségét.

1.3. Hirdetmények lista

A lista elsődleges célja a központi levelező szolgáltatás tehermentesítése. Korábban a „hirdetmények” közlését körlevelek formájában továbbították a felhasználók felé, ami túlzott hálózati forgalmat generált, különösképpen a nagyobb méretű csatolmányok miatt. A Hirdetmény 2.0 biztosítja minden osztály részére az osztályszintű, osztály keretein túlmutató, esetleg a teljes állományra vonatkozó fontos információk azonnali közzétételét minden felhasználó számára hirdetmények formájában. A túlzott hálózati forgalom elkerülése érdekében, a hirdetményeket elsősorban szöveges formátumban kell elhelyezni. A hirdetmények rögzítését, módosítását csak az osztály részéről a szerkesztő és jóváhagyók listában beregisztrált tartalomszerkesztő végezheti. Az új, illetve a módosított hirdetmények közzététele pedig csak regisztrált tartalomjóváhagyó jóváhagyásával történhet. Előnye a listának, hogy egyetlen tárhelyről, erőforrás takarékos módon elérhetővé teszi azon információkat, melyeket korábban redundáns módon, minden felhasználó felé egyedileg közvetítettek. Másik előnye, hogy a csatolt dokumentumok egyetlen példányban kerülnek publikálásra, a hirdetményhez csatolva.

2. Tervezés

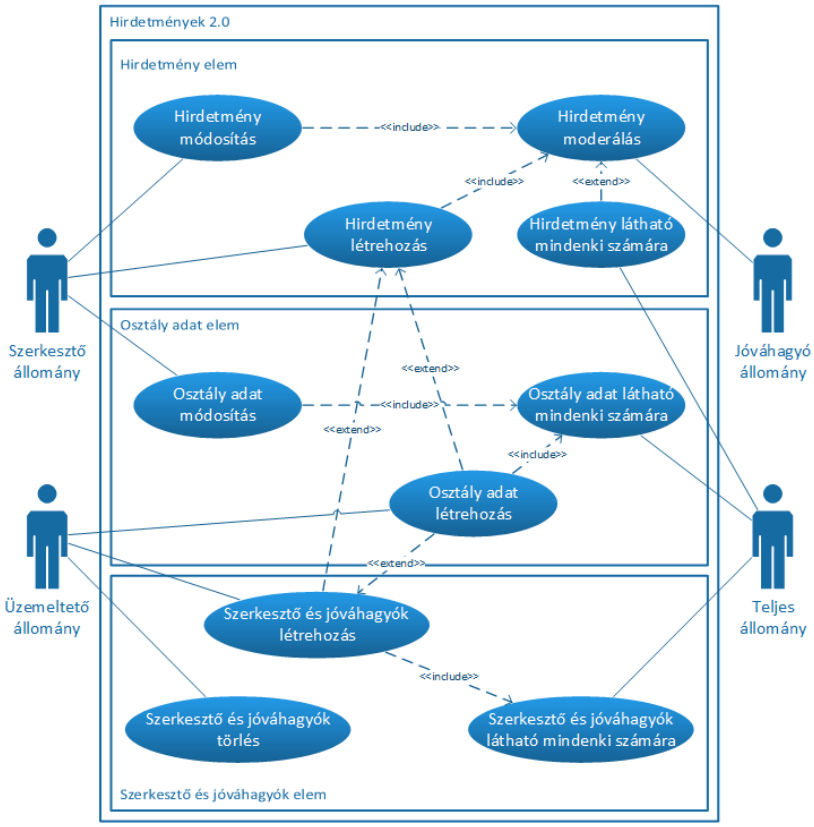
A SharePoint adta beépített funkciók segítségével nem lehet a szolgáltatást automatizáltabbá, felhasználó barátibbá tenni. Hiába a magasabb 2013-as verzió környezet, a SharePoint e verziója sem képes egyszerű beépített munkafolyamatokkal az elvárásokat teljesíteni. Így a

SharePoint egy magasabb szintű tudást igénylő, az esemény figyelések (event receivers) C# nyelven keresztül történő programozását választottuk. Eseményfigyelések használatával a Microsoft SharePoint lehetővé teszi, hogy egyéni kóddal, programozottan reagálhassunk, amikor egy adott esemény bekövetkezik egy SharePoint objektum változtatásakor. Visual Studio-ban úgy nevezett SharePoint 2013 projektet hozunk létre, amiben SharePoint névterek (namespace) segítségével automatizáltabb és magasabb szintű működést érhetünk el, mint az egyszerűbb beépített funkciókkal.

Elsődleges szempontunk, hogy a Hirdetmények 2.0 felhasználó barát, könnyen kezelhető és áttekinthető legyen mind a felhasználók, mind pedig az üzemeltetők számára. A különböző jogosultsági szintek és csoportok segítségével könnyedén meg tudjuk határozni, hogy a Hirdetmények 2.0 az aktuális felhasználónak (szerepkörét tekintve) milyen adatokat, információkat jelenítsen meg.

2.1. A rendszer használat felőli nézete, UML alapú használati-eset diagram tervezése

Az 1. ábra használati eset diagram segítségével szemléltetjük a rendszer szerepkörök szerinti működésének különböző eseteit, amiket a tervezés során feltártunk.



1. ábra: A rendszer használat felőli nézete

A szolgáltatás használat felőli működése szempontjából hat esetet különböztethetünk meg:

- Osztály adat létrehozás eset: amikor tartalommegosztóként szeretne egy osztály megjeleni a központi webhelyen, akkor az üzemeltető állomány egyik tagja - aki jogosult új elem felvételére - az osztály adatok listában felveszi az osztály adatait új elemként. A teljes állomány számára pedig láthatóvá válik az új osztály adat.
- Szerkesztő és jóváhagyók létrehozása eset: amikor az üzemeltető állomány egyik tagja az előzőekben már regisztrált osztályhoz felveszi az általuk kijelölt tartalomszerkesztőt és annak tartalomjóváhagyóit. A teljes állomány számára pedig láthatóvá válik az új szerkesztő és jóváhagyók.
- Osztály adat módosítás eset: amikor egy tartalommegosztóként regisztrált osztály adataiban változtatást kell alkalmazni. Első körben az osztályhoz tartozó, beregisztrált, szerkesztő és jóváhagyók feladata az osztályuk vonatkozásában a módosításokat végrehajtani. Második körben az üzemeltető állomány is végre tudja hajtani az osztály adatainak módosítását, a háttérben való támogatás végett. A teljes állomány számára pedig láthatóvá válik a módosított osztály adat.
- Szerkesztő és jóváhagyók törlése eset: amikor az osztályhoz tartozó szerkesztő, vagy jóváhagyók egyikének, vagy mindannyiuknak megszűnik a tartalommegosztói feladata osztálya vonatkozásában. Az üzemeltető állomány egyik tagja törli a szerkesztő és jóváhagyók listában a regisztrált tartalommegosztókhoz tartozó elemet.
- Hirdetmény létrehozás eset: amikor a tartalommegosztóként regisztrált osztály fontos információkkal szeretné tájékoztatni a célközönséget, vagy a teljes állományt. Az előzőekben a szerkesztő és jóváhagyók listában már regisztrált szerkesztő létrehoz egy új elemet (hirdetményt) a hirdetmények listában. A teljes állomány számára pedig láthatóvá válik az új hirdetmény.
- Hirdetmény módosítás eset: amikor a tartalommegosztóként regisztrált osztály módosítani szeretné a már előzőekben létrehozott hirdetményt. A hirdetmények listában, a regisztrált szerkesztő megkeresi az általa létrehozott módosítandó hirdetményt és módosítja az adatokat benne. A teljes állomány számára pedig láthatóvá válik a módosított hirdetmény.

3. Implementálás és integrálás

Első körben a webhely oszlopokat és a listákat kell létrehozni, majd a listákhoz a webhely oszlopokat kell hozzácsatolni. PowerShell scriptek alkalmazásával gyorsabban létre tudjuk

hozni a kívánt webhelyet, listát, oszlopokat vagy más egyéb SharePoint objektumot. A mi esetünkben nem elég a PowerShell parancsértelmezőt megnyitni, mert nem tartalmazza a SharePoint farmhoz tartozó névtereket. Erre külön parancsértelmezőt fejlesztett ki a Microsoft, még pedig SharePoint Management Shell néven. Ez a rendszerhéj csak a SharePoint szervereken érhető el, ezért minden esetben be kell lépni egy SharePoint szerverre, ha ilyen jellegű parancssort vagy scripteket akarunk futtatni.

A kódbázis két külön részből áll. Az első része SharePoint Management Shell scriptekből áll, ami a webhelyen létrehozza és bekonfigurálja a szolgáltatás alapját. A másik rész pedig Visual Studio programmal létrehozott SharePoint 2013 projekt fájl lesz (továbbiakban: solution), ami tartalmazza az esemény figyeléseket és ezek hatására bekövetkező folyamatokat.

3.1. A kódbázis felépítése

Maga a kódbázis öt jól elkülöníthető részcsoportból áll:

- Elem esemény figyeléshez létrehozott osztályok
- Listák, csoportok és jogosultsági szintek részére létrehozott osztályok
- SharePoint Server Side Object Modell (SSOM) részére létrehozott osztályok
- SharePoint Client Side Object Modell (CSOM) részére létrehozott osztályok
- Üzenet küldésnek létrehozott osztályok

3.2. Az alkalmazásban megvalósított főbb funkciók

SharePoint listákat többféleképpen lehet deklarálni szerver oldali objektumként. Legtöbb esetben a lista megjelenítési nevével hivatkoznak rá. Ezt a lehetőséget szerettük volna elkerülni annak biztosítása érdekében, hogy ha a későbbiekben a lista megjelenítési nevét meg akarják változtatni, azt gond nélkül megtehesse az üzemeltető állomány. Ezért kerestünk egy olyan lehetőséget, amivel a lista url paraméter megadásával tudjuk deklarálni a lista objektumot. Azért döntöttünk ezen megvalósítás mellett, mert tapasztalataink szerint az elemek url elérését ritkán változtatják meg és ékezetes betűket sem alkalmaznak benne, így az bizonyult a legjobb választásnak. A létrehozott funkciónak átadandó paraméterek az SPWeb szerver oldali objektum és egy szöveg típusú adattag:

```
public static SPList ListByUrl(SPWeb web, string url)
{
      return web.GetList(SPUrlUtility.CombineUrl(web.ServerRelativeUrl, url));
}
```

A fejlesztő programban a függvény meghívása:

```
SPList list = GetSO.ListByUrl(web, ”/List/hirdetmenyek”);
```

A tervezés során arra a következtetésre jutottunk, hogy az elemnek egyedi engedéllyel kell rendelkeznie. Alapértelmezetten a SharePoint objektumok létrehozásuk során öröklik a szülőtől az engedélyeket, így első lépésben meg kell törni az öröklést, majd beállítani a hozzáféréseket. Az öröklés megtörésére `BreakRoleInheritance()` metódust lehet használni alapértelmezetten az `SPItem` objektumon is. A metódust használva be lehet állítani, hogy a szülőtől kapott engedélyek megmaradjanak az elemen vagy sem. Mivel minden esetben törölni kellene, ezért ezzel a metódussal töröljük az örökölt engedélyeket, így nem kell létrehozni a tisztításhoz szükséges eljárást. Két logikai értéket kell megadni a metódusnak. Az első érték igaz állítása esetén lemásolja a jogosultságokat a szülő elemtől és megtartja őket a megtört elemen, hamis állítás esetén nem hajtja ezt végre. A másik pedig, hogy az elemünk gyermek elemeiről szintén törölje ezeket az örökölt engedélyeket. Az esetünkben az első logikai érték mindig *false* a második pedig *true*. Viszont ehhez a beállításhoz már több jogosultsággal kell rendelkeznie az aktuális felhasználó nevében futtatott szolgáltatásnak. A regisztrált szerkesztők, jóváhagyók nem rendelkeznek ilyen engedéllyel, ezért előtte magas jogosultságkezelést kell alkalmaznunk az elemen, majd ezután tudja a szolgáltatás is végrehajtani az öröklések leállítását:

```
public static void Inheritance(SPListItem listitem)
{
      SPListItem elevatedListItem = GetSO.ElevatedLitsItem(listitem);
      elevatedListItem.BreakRoleInheritance(false, true);
}
```

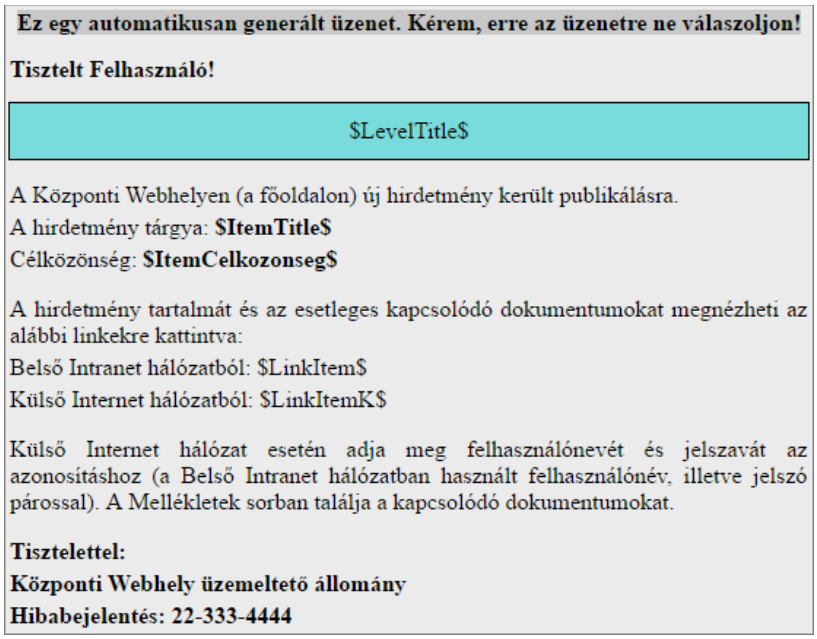
3.3. Az alkalmazás legfontosabb kódbázisai

A kódbázisnak minden egyes része fontos, de ha ki kellene emelni néhányat, akkor a szerver oldali objektum modellben alkotott magas jogosultságkezelés, mint legfontosabb függvény, és az üzenet küldésének létrehozott osztályok kerülnének kiemelésre. A magas jogosultság kezelés lehetőség a forráskód megírása közben sok problémát oldott meg, amikor a felhasználó nevében futó szolgáltatás olyan listákhoz is hozzáfért, amelyhez nincs köze, és így a szolgáltatás problémamentesen tudott tovább futni. Többször is alkalmaztuk ezt az eljárást: amíg a felhasználó rendelkezik az összes olyan engedéllyel, amivel még tud dolgozni az elemen vagy elemeken, addig a felhasználó nevében futott a szolgáltatás. Mihelyst már nem voltak elegendők a felhasználó engedélyei, akkor forgattuk át az elemet emelt jogosultsági szintre és folytatta tovább a szolgáltatás az utasítások végrehajtását. Ennek

köszönhetően nem akadt meg a szolgáltatás hozzáférési hiba miatt. Ellenben így sokszor újra kellett tervezni az algoritmust, hogy meddig legyen úgymond a felhasználó kezében a szolgáltatás működése, és mikor kapjon teljes hozzáférést ezzel az eljárási lehetőséggel.

Az üzenet küldésének létrehozott osztályok és metódusok csoportja egyike a legnagyobb, legösszetettebb, külön elszeparálható szigetrésze a kódbázisnak. Ennek köszönhetően bármikor felhasználható más szolgáltatások tervezésekor és alkalmazható más projektekben. Ebben a részben alkalmazzuk a levél sablonok listában a token cserét. Vegyük azt az esetet, amikor új hirdetményt publikáltak és a felhasználókat azonnal akarják értesíteni. A forráskódban elérkezünk ahhoz a részhez, ahol a levél sablonok listából megkeressük az új hirdetményhez tartozó levél sablon törzsét.

A szövegrész egyes részei $ jelek között helyezkednek el, mint például a $LevelTitle$, vagy a $LinkItem$. Ezek lesznek azok a tokenek, amiket dinamikusan tudunk változtatni. Most nézzük meg egy web böngészőben, hogy is néz ez ki 2. ábra:



Ez egy automatikusan generált üzenet. Kérem, erre az üzenetre ne válaszoljon!

Tisztelt Felhasználó!

$LevelTitle$

A Központi Webhelyen (a főoldalon) új hirdetmény került publikálásra.
A hirdetmény tárgya: **$ItemTitle$**
Célközönség: **$ItemCelkozonseg$**

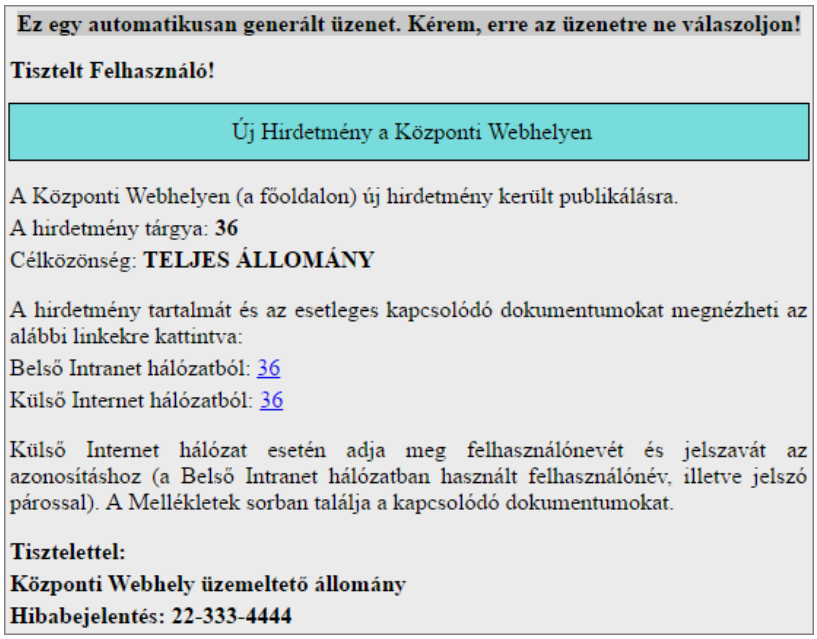
A hirdetmény tartalmát és az esetleges kapcsolódó dokumentumokat megnézheti az alábbi linkekre kattintva:
Belső Intranet hálózatból: $LinkItem$
Külső Internet hálózatból: $LinkItemK$

Külső Internet hálózat esetén adja meg felhasználónevét és jelszavát az azonosításhoz (a Belső Intranet hálózatban használt felhasználónév, illetve jelszó párossal). A Mellékletek sorban találja a kapcsolódó dokumentumokat.

Tisztelettel:
Központi Webhely üzemeltető állomány
Hibabejelentés: 22-333-4444

2. ábra: Sablon üzenet token csere előtt

Jól látható, hogy így már áttekinthetőbb és a megjelenítése is szebb, viszont még hiányoznak információk a hirdetményről. Minden létrehozott hirdetménynek eltérő lesz a tárgya (ami a hirdetmény címe), a célközönsége, és a legfontosabb, a hirdetmény elérési url linkje. A tokeneknek nevezett dinamikusan változtatható értékeket a kiküldés előtt még meg kell változtatni az aktuális hirdetmény adataival. Ezeket az információkat a levél sablon kiolvasása után egy Replace() metódussal és további függvények segítségével kicseréljük és a 3. ábra látható módon fog kinézni az üzenetet.



Ez egy automatikusan generált üzenet. Kérem, erre az üzenetre ne válaszoljon!

Tisztelt Felhasználó!

Új Hirdetmény a Központi Webhelyen

A Központi Webhelyen (a főoldalon) új hirdetmény került publikálásra.
A hirdetmény tárgya: **36**
Célközönség: **TELJES ÁLLOMÁNY**

A hirdetmény tartalmát és az esetleges kapcsolódó dokumentumokat megnézheti az alábbi linkekre kattintva:
Belső Intranet hálózatból: 36
Külső Internet hálózatból: 36

Külső Internet hálózat esetén adja meg felhasználónevét és jelszavát az azonosításhoz (a Belső Intranet hálózatban használt felhasználónév, illetve jelszó párossal). A Mellékletek sorban találja a kapcsolódó dokumentumokat.

Tisztelettel:
Központi Webhely üzemeltető állomány
Hibabejelentés: 22-333-4444

3. ábra: Sablon üzenet token csere után

4. A program tesztelése

A tesztelés fontos része a szoftverfejlesztésnek és fő célja a hibák bekövetkeztének csökkentése, valamint programhibák megtalálása. A tesztelést úgy kell megtervezni, hogy a lehető legtöbb hibát észleljük. A forráskód írása közben folyamatosan, egyes modulok megalkotása után hibakereséseket végeztünk. A Visual Studio fejlesztő program segítségével futási időben tudtunk a forráskódon soronként végig haladni és figyelemmel kísérni az egyes részek helyes vagy helytelen működését. Szerencsére nem kell mindig az elejétől kezdve soronként végig mennünk a forráskódon, a fejlesztő program segítségével elég töréspontokat elhelyezni. A fejlesztő program egyből az első töréspontra ugrik és innen már lépésenkénti végrehajtással is tovább nézhetjük a forráskódot. A lépésenkénti végrehajtásban az egyes lépések lefutása előtt lehetőségünk van elemezni a program működését. Egy hiba felismerését és kijavítását követően újra tesztelni kell, így meggyőződhetünk róla, hogy a hiba eltűnt (ellenőrző teszt).

Lehetséges tesztelési módszerek (Ammann,, Offutt, 2016) (Myers, Sandler, Badgett, 2011):

- Egység- és rendszer teszt
- Fekete és fehér doboz tesztelés
- Verifikáció és validáció

Mivel egyik módszerrel sem lehetséges a kimerítő tesztelés, el kell fogadnunk, hogy nem tudjuk egyetlen program hibamentességét sem szavatolni. A tesztelés során feltárt hibák

helyének megállapítása a hiba keresése. Általában a program növekedésével együtt növekszik a hibák száma is.

4.1. Egység- és rendszer teszt

Egységteszt során a rendszer elemeit, moduljait teszteljük külön-külön, egymástól függetlenül. Az általunk deklarált kisebb, nagyobb egységek például:

- új hirdetmény esetén létrehozó felhasználó jogosultságának ellenőrzése
- a szerkesztő és a hozzá tartozó jóváhagyók adatainak megállapítása
- tartalommegosztókhoz tartozó osztály elem keresése
- extranet eléréséhez szükséges adatok megállapítása
- hirdetmény másolása extranet webhelyre
- jogosultság öröklésének megtörése
- jogosultságok beállítása
- azonnali üzenet küldésének megállapítása
- célközönség email címének megállapítása
- értesítő üzenet összeállítása
- értesítő üzenet küldése

4.2. Fekete és fehér doboz tesztelés

A dinamikus tesztelési módszerek alapelve, hogy a programot működés közben vizsgáljuk. Két esetét ismerjük a fekete és a fehér doboz tesztelést.

Fekete doboz tesztelés (adatvezérelt tesztelés) esetén csak a működési rendellenességek felderítésére van lehetőségünk, a hibák helyének lokalizálására nincs. Ebben az esetben a tesztelő nem rendelkezik információval a szoftver belső struktúrájáról, tehát a forráskód ismeretlen számára. Csak a bemenő és kimenő adatok kapcsolatát figyeljük. Az ilyen teszteket jellemzően nem a fejlesztő csapat végzi.

- Az értesítő beállítása listán a teljes állomány számára nincs felvéve terjesztési lista email cím. Ilyenkor, ha az osztály adatok listában új elemet vesznek fel és a felhasználókat is értesíteni akarják, akkor nem kap értesítő üzenetet a teljes állomány. Ellenben a szolgáltatásnak hiba nélkül kell tovább működnie.
- Az értesítő beállítása listán az osztályhoz nincs felvéve a hozzá tartozó terjesztési lista email címe. Ilyenkor, ha a hirdetmények listán új hirdetményt publikálnak és a

célközönséget is értesíteni akarják, akkor a célközönségnek beállított osztály nem kap értesítő üzenetet. Ellenben a szolgáltatásnak hiba nélkül kell tovább működnie.

- A felhasználó valamilyen oknál fogva bekerült vagy beragadt a szerkesztő csoportba, és a szerkesztő és jóváhagyók listában nincs regisztrálva, továbbá a felhasználó új hirdetményt vesz fel a hirdetmények listán. Ilyenkor a szolgáltatásnak törölnie kell a hirdetményt, értesítenie kell a felhasználót, hogy nincs tartalommegosztói jogosultsága a hirdetmény publikálásához.

Fehér doboz tesztelés (logika vezérelt tesztelés) esetén figyelembe vesszük, hogy ismerjük a forráskódot és tudjuk, hogy mikor minek és hogyan kell történnie. Tehát a tesztelőnek pontosan ismernie kell a szoftver belső szerkezetét. A teszteseteket a program struktúrája alapján választjuk, annak tudatában, hogy nem lehet minden végrehajtási sorrendre kipróbálni. Az utasítás lefedése, hogy minden utasítás egyszer hajtódik végre. Döntés lefedése, hogy minden elágazás minden ágát legalább egyszer körbejártuk.

4.3. Verifikáció és validáció

- Nulladik tesztelésként határoztuk meg a PowerShell scriptes telepítés utáni ellenőrzést, hogy minden oszlop, lista, csoport és jogosultsági szint a megadott paraméterekkel, elnevezésekkel szerepel-e a központi webhelyen. Valamint mindezek megfelelően vannak-e felkonfigurálva. Ugyanis, ha a strukturált osztályok adataiban bármelyik elem neve vagy tulajdonsága akár egy karakterrel is másként szerepel, mint a szolgáltatás kódjában, akkor a működési folyamatok hibával megállnak és a szolgáltatás helytelen működését okozhatja.
- Az üzemeltető csoportba felvett felhasználó mindegyik listát meg tudja-e nyitni, új elemet felvenni, meglévő elemet módosítani, törölni tud-e. Ugyanis a hirdetmények lista kivételével mindegyiken képes ezekre. Új hirdetmény felvételekor egy értesítő üzenetet kell kapnia, hogy nem rendelkezik tartalommegosztói jogosultsággal, valamint az elemet törölnie kell a szolgáltatásnak. Az üzemeltető felhasználó az olvasó és szerkesztő csoportba képes-e új felhasználót felvenni, és meglévőt kivenni a csoportból.
- Az osztály adatok listán új elem felvételekor vagy létező elem módosításakor és a felhasználók értesítését aktiválva a teljes állomány meg kapja-e az értesítő üzenetet és az információs adatok megfelelnek-e benne. Elem törlésekor nem kap-e egyetlen felhasználó sem értesítést.

- A szerkesztő és jóváhagyók listában az üzemeltető felhasználó által felvett új tartalommegosztók megkapják-e az értesítő üzeneteket és az információs adatok megfelelnek-e benne. A szerkesztőnek kijelölt felhasználónak a tartalomszerkesztői, a jóváhagyónak pedig a tartalomjóváhagyói üzenetet.
- A regisztrált szerkesztő képes-e az osztályához tartozó osztály elemet szerkeszteni, de törölni nem. Képes-e új hirdetményt létrehozni, csak a saját hirdetményét módosítani, de a saját, illetve mások hirdetményét törölni nem. A hirdetmény elutasításáról vagy jóváhagyásáról kap-e értesítő üzenetet a megfelelő adat tartalommal.
- A regisztrált jóváhagyó képes-e az osztályához tartozó osztály elemet szerkeszteni, de törölni nem. Képes-e csak a hozzá tartozó hirdetményt moderálni, de a saját, illetve mások hirdetményét törölni nem. Új hirdetmény felvételekor kap-e értesítő üzenetet a moderálandó hirdetményről, a jóváhagyó és az információs adatok megfelelnek-e benne.
- Új hirdetmény publikálása után a teljes állomány képes-e megnyitni azt. Ha a célközönség számára értesítést kértek az új hirdetménnyel kapcsolatban, akkor kapott-e a célközönség értesítő üzenetet és az információs adatok megfelelnek-e benne. Továbbá az extranet oldali webhelyen a publikált hirdetmény megjelenik-e összes csatolmányával együtt a teljes állomány számára.
- A törölt szerkesztő képes-e a volt osztályához tartozó osztály elemet szerkeszteni, képes-e új hirdetményt létrehozni. Törlést követően kapott-e értesítő üzenetet tartalomszerkesztői feladatainak elvételéről és az információs adatok megfelelnek-e benne.
- A törölt jóváhagyó képes-e a volt osztályához tartozó osztály elemet szerkeszteni. Képes-e a hozzá tartozó hirdetményt moderálni. Törlést követően kapott-e értesítő üzenetet tartalomjóváhagyói feladatainak elvételéről és az információs adatok megfelelnek-e benne.

5. Összegzés

A projekt résztvevői a Hirdetmények 2.0 létrehozása során nagyon sokat tanulhattak a SharePoint kulisszái mögötti területekről. A felhasználói felületen kialakítani egységesen működő szolgáltatást egyre nehezebbnek bizonyult, nagyon sokszor olyan igények merültek fel, amit már így nehezen lehet megoldani. A szolgáltatás megírásával világossá vált, hogy a SharePoint-on az igényeket létrehozni, alkalmazni, működtetni sokkal könnyebb, mint

felhasználói felületen összehozni. Természetesen itt is vannak nehezen megvalósítható igények, de a lehetőségek tárháza messzemenően meghaladja a felhasználói felület adta lehetőségeket.

A projekt alapú oktatás és tanulás több műszaki területen is jelenthet hatékonyabb tudás szerzést, jobb problémamegoldó készséget és hatékony kommunikáció fejlődést. (Katona et al, 2016).

Irodalomjegyzék

Ammann, P., & Offutt, J. (2016). *Introduction to software testing*. Cambridge University Press.

Boaler, J. (2002). Learning from teaching: Exploring the relationship between reform curriculum and equity. *Journal for Research in Mathematics Education*, 33(4), 239–258.

Ilter, İ. (2014). A study on the efficacy of project-based learning approach on Social Studies Education: Conceptual achievement and academic motivation. *Educational Research and Reviews*, *9*(15), 487.

Kővári, A. (2019). A felnőttoktatás 4.0 és az az ipar 4.0 kihívásai az életen át tartó tanulásban. PEDACTA, 9(1), 9–16.

Katona J et al (2016). A Brain–Computer Interface Project Applied in Computer Engineering. *IEEE Transactions on Education, 59*(4), 319-326.

Márta, H. (2010). Projektmódszer a 21. században II. *Új pedagógiai szemle, 1*, 2.

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.