SZAKDOLGOZAT

Szimeonov György

Debrecen
2011

Debreceni Egyetem

Informatika Kar

Vállalati Portálok Fejlesztése

Témavezető:

Bátfai Norbert

Egyetemi tanársegéd

Készítette:

Szimeonov György

Mérnök Informatikus

Debrecen

2011

Tartalomjegyzék

1. Bevezetés

1.1. Programozói ismeretek

Szakdolgozatom megértéséhez elengedhetetlen a Java SE (Standard Edition) alapos ismerete valamint némi Java EE (Enterprise Edition) ismeret. A dolgozatomban használt összes technológia nyílt forráskódú.

1.2. Témaválasztás

Jelen dolgozatom témája a Vállalati Portálok fejlesztése. Szakdolgozatomban bemutatni kívánok egy vezető Open Source, azaz nyílt forráskódú portálrendszert a Liferay-t, valamint a Liferay-hez közelálló vezető technológiákat. E dolgozat írásakor 2 éves fejlesztői tapasztalatot szereztem egy ERP (Enterprise Resource Planning) rendszer fejlesztésében és fél éves tapasztalattal rendelkezem az Országos Rendőr-főkapitányság informatikai rendszerének fejlesztésében. Munkáim során lehetőségem volt betekinteni nagyobb magyar vállalatok informatikai rendszerének működésébe, azokba a tényezőkbe melyek elősegítik az informatikai rendszerek hatékony működését.

1.3. Gyorsuló világ

Az internet napjaink legnagyobb hálózatává nőtte ki magát. Az internet beágyazódott a magán illetve a gazdasági életbe. Több millióan használják, a modern élet mára elképzelhetetlen internet nélkül. A megnövekedett felhasználószám újabb és újabb technológiák megjelenését teszik és tették lehetővé.

Napjainkban internet nélkül egy közép vagy nagyvállalat működése lehetetlenné válna. A vállalati portálok mérettől függetlenül fontos tételei a cégek informatikai kiadásainak.

1.4. Vállalati Portálok

Definíciót csak nehezen tudnánk létrehozni, annak meghatározására, hogy mit is tekinthetünk vállalati portálnak.
A vállalati portál tipikusan egy web alkalmazás. A web alkalmazás egy olyan alkalmazás ami hálózaton keresztül elérhető, legyen az internet vagy intranet. A portál alkalmazás egy olyan web alkalmazás ami együttműködhet más alkalmazásokkal és újrahasznosíthatja azok funkcionalitását a portál platform segítségével, ezzel csökkentve a fejlesztési időt és több élményt nyújtva a végfelhasználónak.

A következőkben olyan közös jellemzőket próbálok bemutatni arra vonatkozóan mit is tekinthetünk vállalati portálnak, amik általában a közép és nagyvállalati rendszerekre jellemzőek.

- Egy portál általában biztosít különböző testreszabhatósági funkciókat, jogosultsági beállításokat valamint tartalommegosztást.
- Kollaborációt a felhasználói közösségek között
  (A portál adminisztrátorok különböző közösségeket hozhatnak létre, amikhez felhasználók csatlakozhatnak a munkamegosztás vagy információcsere érdekében)
  - Csoportbeszélgetések
  - Blogok
  - Dokumentum megosztás (irat, fénykép stb.)
  - Wikipedia
  - Esemény naptár (a portál felhasználói tudomást szereznek az elkövetkezendő eseményekről)
  - Chat

.

A fent említett funkciókra többé-kevésbé minden vállalatnak szüksége van, a kérdés a megvalósításban rejlik. Számos egyedi fejlesztésben megszületnek a fenti funkcionalitások, viszont ezeknek a funkcionalitásoknak a lefejlesztése feleslegessé válhat, ha egy kész portál megvalósítást választunk, ami pénzt és legfőképp időt spórol a vállalat számára.

Még látványosabb eredményeket érhetünk el ha a választott portál alkalmazás Open Source, azaz ingyenes és forráskódja elérhető a fejlesztők számára.

1.5. Java alapú portál megvalósítások

Név	URL
Liferay	http://www.liferay.com/
Pluto	http://portals.apache.org/pluto/
Jamecs	http://jamecs.sourceforge.net/
jPortlet	http://jportlet.sourceforge.net/
JetSpeed	http://portals.apache.org/jetspeed-2/

2. Open Source

Az Open Source az informatikában a nyílt forráskódú termékeket jelenti. A nyílt forráskódú szoftverek súlya és aránya az informatikán belül folyamatosan növekszik. Mára már a legnagyobb szoftverfejlesztő cégek is ráeszméltek mekkora hatékonyságnövekedést és versenyelőnyt érhetnek el Open Source termékek piacra bocsátásával. [3]

2.1. GPL

A **GNU GPL** (GNU General Public Licence – Általános Nyilvános Licenc) feltételeit a Free Software Foundation Inc. alakította ki. Amikor egy nyitott forráskódú terméket használunk, a telepítés során el kell fogadnunk a GNU GPL licence feltételeit, ugyanúgy mintha egy kereskedelmi szoftver licence feltételeit fogadnánk el. Ennek egyik legfontosabb eleme a jogdíjmentesség, továbbfejlesztés esetén viszont a továbbfejlesztett program másolataira is ki kell terjeszteni a szabad felhasználás feltételeit. Az Open Source termékeket nem

szabadalmaztathatjuk mert ezáltal saját tulajdonúvá tennénk azokat. Jellemző tulajdonság továbbá a garanciamentesség, tehát a szerzővel szemben semmilyen garanciális jog nem érvényesíthető.
Az Open Source licence feltételei az alábbihoz hasonló módon jelennek meg. (OpenOffice)

„Ezt a terméket a GNU Lesser General Public License Version 2.1 feltételei szerint tették elérhetővé. Az LGPL licenc egy példánya megtalálható
a http://www.openoffice.org/license.html címen. A termék szabad szoftver. Ez azt jelenti, hogy tetszőleges célra, tetszőleges helyen, tetszőleges számú számítógépen licencdíj mentesen, azaz ingyenesen használható, és az LGPL licenc előírásai szerint szabadon terjeszthető, módosítható és a módosítás is terjeszthető."

2.2. LGPL

A **GNU LGPL** (Lesser General Public Licence) meglehetősen eltér a **GNU GPL**–től. Ezt a licencet azért alkalmazzák bizonyos programkönyvtárakhoz, hogy engedélyezhessük beépítésüket nem szabad programokba is. Amikor egy programot összeszerkesztenek egy programkönyvtárral, függetlenül attól, hogy statikusan vagy egy megosztott programkönyvtár használatával, a kettő jogilag közös munkának számít. A normál **GNU GPL** feltételek ezért csak akkor engedik az ilyen összeszerkesztést, ha a teljes kombináció megfelel a szabadság feltételeinek. **A GNU LGPL** viszont kevésbé ilyen szigorúan rendelkezik a programkönyvtárak összeszerkesztéséről.
Azért Lesser mert kevesebbet tesz a felhasználó szabadságának védelmében, mint a normál **GNU GPL**, és kevesebb előnyt nyújt a többi szabadszoftver-fejlesztőnek a versenytárs, nem szabad szoftverekkel szemben. Bár az **LGPL** kevésbé védi a felhasználók szabadságát, azt azonban garantálja, hogy a Programkönyvtárral összeszerkesztett program felhasználója jogosult legyen a program használatára a Programkönyvtár módosított változatával is.[1]

3. Liferay

A Liferay egy világvezető portál megvalósítás. A platform olyan beépített funkciókat tartalmaz amik rövid és hosszú távon is jelentős megtérítést hoznak egy vállalati rendszer esetében. Standard úton biztosít számos portál funkciót és integrálási lehetőséget, amely sok fejlesztési munkától szabadítja meg a vállalatot, ezzel pénzt és időt spórol, így több idő jut az üzleti funkcionalitások megvalósítására.

3.1. Liferay előnyök

A Liferay számos előnyt biztosít más keretrendszerekhez képest:

- Könnyű használat
  (számos művelet programozás nélkül elvégezhető: drag-and-drop műveletek)
- Open Source (LGPL)
  (a Liferay teljes kódja elérhető és módosítható)
- Rugalmasság
  (bármikor lehetőség van verzióváltásra, a forráskód módosítása nélkül)
- Skálázhatóság
  (nagy felhasználószám mellet is stabil)
- Nyelvisíthető
  (új nyelvi erőforrás készítésével bármikor megváltoztathatjuk a nyelvet.
  Elérhető nyelvek: Magyar, Német, Spanyol, Francia, Olasz, Japán stb.)
- Támogat számos külső keretrendszert és technológiát
  (LDAP server, népszerű adatbázis szerverek, mint Oracle, MySQL stb.)
- Standard-eken alapul
  (100%-ban java alkalmazás, ezáltal más standard-eken alapuló technológiát is tud használni, mint Hibernate, IceFaces, PHP, Ruby stb.)

3.2. Liferay verziók

A Liferay két féle verziót biztosít a fejlesztők illetve a vállalatok számára, az úgynevezett Community (közösségi) illetve Enterprise (vállalati) verziót (Community Edition CE, Enterprise Edition EE). A legnagyobb különbség a két változat között az ár. Értelemszerűen a CE az ingyenes verzió míg az EE a fizetős nagytestvére a CE verziónak. Fejlesztésben különbség lehet az eszközben, míg a CE –t Eclipse IDE-hez készített pluginnal fejleszthetjük, addig az EE-t a Liferay Developer Studio-val. Fontos megemlíteni viszont hogy mindkét verzió esetén a forráskód elérhető marad és mindkét verzió hozzáfér a Liferay központi tárolójához (repository) ahonnan letölthetőek a frissítések és az új komponensek. Az EE értelemszerűen mindent tartalmaz, amit a CE valamint extra fizetős szolgáltatásokat, mint például támogatás (support), garancia, tréning stb.

3.3. Támogatott technológiák

A Liferay támogatja az alábbi Servlet konténereket illetve alkalmazásszervereket.

- Apache Geronimo 2.x
- Glassfish 2-3
- JBoss 4.0.x
- WebLogic 8.1 SP4, 9.2, 10
- WebSphere 5.1, 6.0.x, 6.1.x
- JOnAS 4.8.x
- Resin 3.0.19
- Tomcat 5.0.x,/5.5.x//6.0.x
- Borland ES 6.5

Támogatott adatbázisok:

- DB2
- Derby
- HSQL
- Informix
- MySql
- Oracle

- Postgresql
- SQL_Server
- Sybase

Támogatott operációs rendszerek:

- Linux (Debian, RedHat, SUSE, Ubuntu stb.)
- UNIX (AIX, FreeBSD, HP-UX, OS-X, Solaris stb.)
- Windows
- MAC OS X

A Liferay mint platform együttműködik az összes Java technológiával, amit használni szeretnénk fejlesztésünkhöz. A portleteknek és a Java EE specifikációnak köszönhetően bármilyen keretrendszert használhatunk függetlenül attól, hogy a Liferay használja vagy sem.

A Liferay számos java technológiát támogat, ilyenek a GWT, Vaadin, JSF implementációk, mint például Rich Faces, PrimeFaces vagy az IceFaces.

Liferay-ben használhatók továbbá a legkedveltebb JavaScript könyvtárak, mint jQuery, Dojo, YUI, Sencha, Sproutcore stb. A Liferay 6-os verziójától már használható egy beépített JavaScript rendszer az AlloyUI, ami a YUI 3-on alapul és lehetőséget biztosít számos hasznos JavaScript komponens és HTML5 használatára, amik elengedhetetlenek portálalkalmazásunk számára. Az standard Liferay portletek az AlloyUI-t használják JavaScriptes hívásokra. Lényegében bármilyen JavaScriptes keretrendszert használhatunk portál alkalmazásunkban, amelyek nem ütköznek más keretrendszerek működésével.

A fentiek ismeretében beláthatjuk, hogy a Liferay számtalan különböző konfigurációt támogat a megfelelő működés és a fejlesztői szabadság támogatásának érdekében.

Nagyobb Liferay felhasználók:

Benetton, BMW, Lufthansa Flight Training, World Vision (bizonyított hatékonyság több különböző iparágból származó Fortune 500 listás cégnél)

Magyar Liferay felhasználók:

JavaForum, Szépművészeti Múzeum, DDN (Debrecen Developer Network – WebSynergy alapú)

Liferay működése:

A portál számos weboldalból álló alkalmazás, ezeket az oldalakat portáloldalaknak nevezzük. Minden portál oldalnak van egy fej és lábrésze valamint lehet menüje vagy valamilyen logója.
A fej és lábrész között találhatóak a különböző alkalmazások, amik valójában portletek.

A Liferay két eszközt (Portlet, OpenSocial Gadget) biztosít a fejlesztőknek, hogyan építsenek hatékonyan alkalmazást portálrendszerükbe.

Portletek:

A portletek kis web alapú alkalmazások Java-ban írva. Portletek segítségével nagyon komplex alkalmazásokat készíthetünk mindazon technológiákkal, amit a Java platform képvisel. A Portlet dinamikus tartalmat generál a felhasználó felé. Egy portáloldal egy vagy több Portlet futtatására alkalmas. A Portlet-eket a Portlet konténer vezérli.

OpenSocial Gadgets:

A gadget-ek általában kis alkalmazások, amik böngésző közeli technológiákon alapulnak mint a HTML és JavaScript. A legnagyobb előnyük hogy könnyű őket fejleszteni, és több ezer előre elkészített Gadget létezik például az iGoogle's tárolóban.

4. Liferay Adminisztráció

4.1. Telepítés

A Liferay telepítésére nagyon egyszerű út áll rendelkezésre. Csupán le kell tölteni egy előre csomagolt verziót a http://www.liferay.com/downloads/liferay-portal/available-releases címről, ami lehet Tomcat-es vagy egyéb web konténeres változat (mivel a Liferay csomag

előre konfigurált szerverekkel elérhető és telepíthető). A letöltést és kicsomagolást követően csak el kell indítani a kiválasztott web konténert vagy alkalmazásszervert. Tomcat esetében a bin mappában lévő startup.bat vagy startup.sh futtatásával.

Sikeres indítást követően a Liferay elérhető a localhost:8080 – as címen. A Bejelentkezés admin szerepkörrel, a test@liferay.com felhasználónévvel és test jelszóval lehetséges

4.1.1 Admin adatok megváltoztatása

Az alapértelmezett admin adatok a portal.properties állományban találhatóak a következő formában:

```
default.admin.password=test
default.admin.screen.name=test
default.admin.email.address.prefix=test
default.admin.first.name=Test
default.admin.middle.name=
default.admin.last.name=Test
```

Természetesen a beállítások felüldefiniálhatóak a portal-ext.properties állományban. [23]

4.1.2 Domain és port beállítása

Fejlesztési szempontból a localhost:8080-as cím teljesen megfelelő, viszont éles üzemben elfogadhatatlan. Windows rendszer esetén a localhost módosítására a C:/Windows/System32/drivers/etc mappában a hosts fájlt kell módosítani 127.0.0.1 localhost helyett a 127.0.0.1 domainname formában. A 8080-as portot a Tomcat server.xml konfigurációs állományában kell módosítani 80-ra, hogy ne jelenjen meg az url-ben a port.

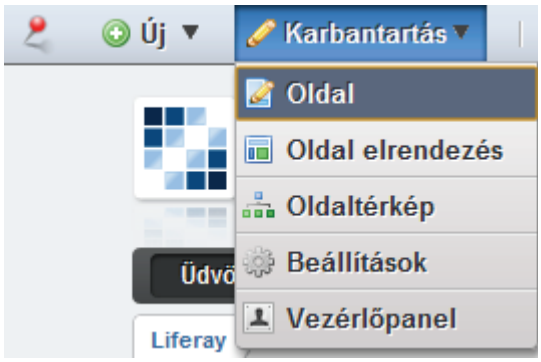
4.2. Liferay kezelőfelülete

A Liferay egy egyszerű és kényelmes kezelőfelületet biztosít a portál adminisztrátoroknak illetve a felhasználóknak. A portáloldal felső sávjában található egy kezelőfelület (dockbar) melyen megtalálhatóak a fontosabb funkciók, mint például új oldal felvétele vagy karbantartása, valamint globális portálbeállítások érhetőek el a karbantartó felületen.

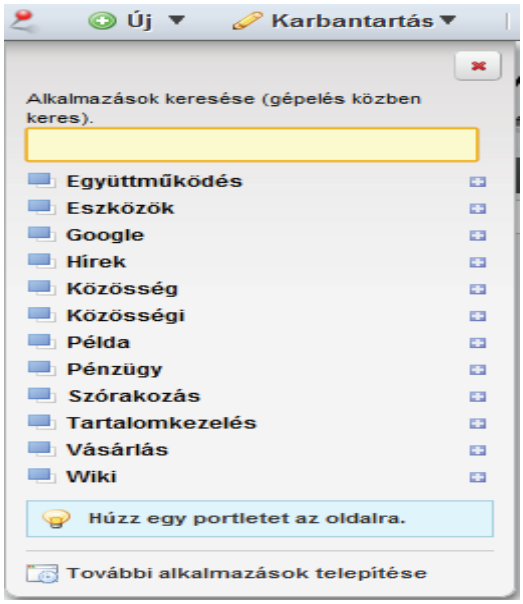
1. ábra – Liferay Kezelőpult

„Megjegyzés: a képernyőn látott magyar nyelv nem alapértelmezett beállítás lásd. 4.3”

A kezelőfelületen a legfontosabb két menüpont az *„új”* és a *„karbantartás”* menü.
Az *„új”* menüpontban van lehetőség tallózni az előre telepített portál alkalmazásokból vagy létre lehet hozni új oldalt programozás nélkül, csupán előre gyártott beállításokkal. Az *„új”* menüponton belül talán a leggyakrabban használt elem a *„több”* mivel itt jelennek meg a gyári illetve az általunk készített portálalkalmazások. A *„karbantartás”* menüpontban van lehetőség a portáloldalak konfigurációjára illetve globális konfigurációkra a vezérlőpanel elemen keresztül.



2. ábra -Karbantartás

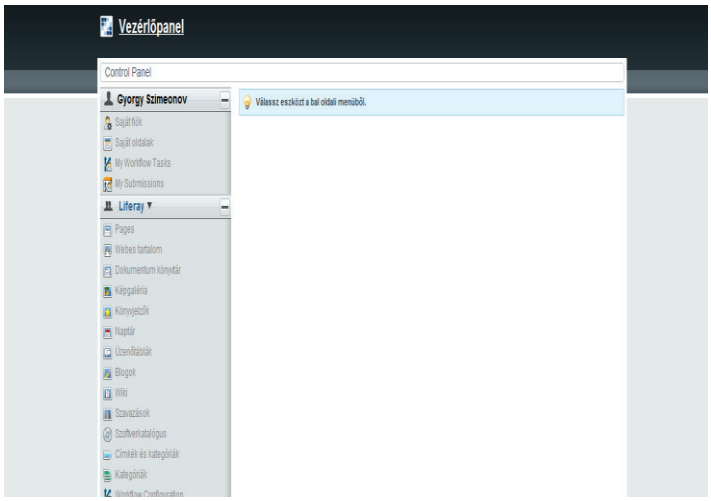


3. ábra - Alkalmazások

4.1.3 Liferay vezérlőpanel

A vezérlőpanelen van lehetőség a portál globális konfigurációjának szerkesztésére. 3 nagyobb menüpontból épül fel:

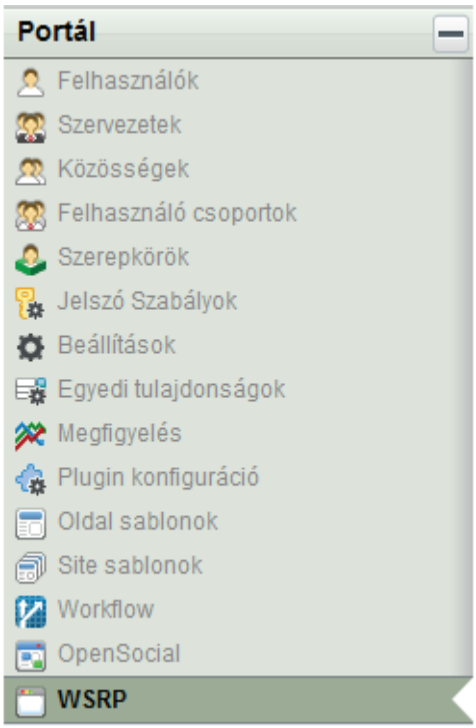
- Saját adatok szerkesztése
- Portál
- Szerver



4. ábra - Vezérlőpanel

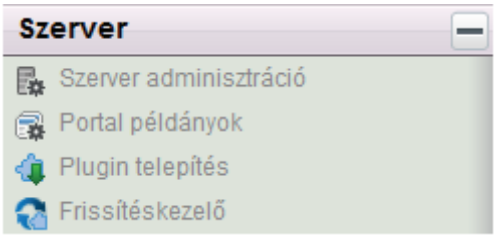
A **saját adatok szerkesztésében** van lehetőség a saját fiók és oldalak szerkesztésére.

A **portál** menüpontban van lehetőség a globális portál funkciók beállítására. Ilyen funkciók a felhasználók, szervezetek, közösségek, szerepkörök stb. beállítása. Ebben a menüpontban lehet konfigurálni a portál jelszószabályzatát. Itt lehetséges konfigurálni a különböző plugineket, legyen az Portlet vagy Gadget.



5. ábra – Portál adminisztráció

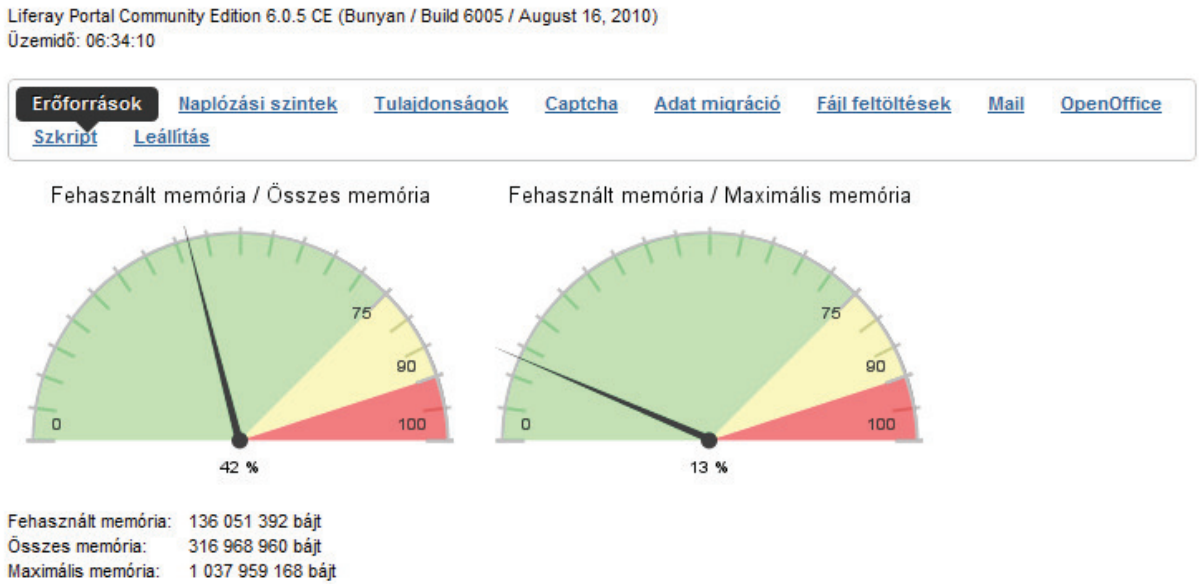
A **szerver** menüpontban elérhetőek a szerverbeállítások valamint itt végezhetőek el a plugin telepítések és frissítések



6. ábra – Liferay szerver adminisztráció

A portál adminisztrátorok számára talán a legfontosabb menüpont a szerver menün belül a szerver adminisztráció. A Liferay itt biztosít lehetőséget a szerver monitorozására, naplózására. Számos szerver művelet elérhető itt, mint például:

- Takarító algoritmusok futtatása a memória felszabadítására
- VM gyorsítótárának törlése
- A cluster gyorsítótárának törlése
- Adatbázis gyorsítótárának törlése
- Szál dump generálás
- Az összes plugin adatbázis tábláinak ellenőrzése



7. ábra – Liferay Naplozás

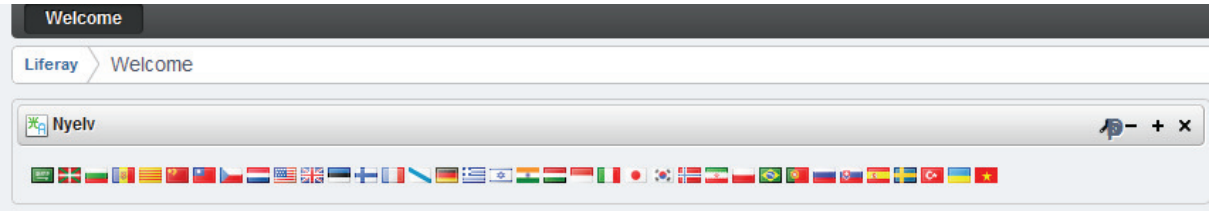
4.3. Portál beállítások

4.1.4 Nyelv beállítása

A Liferay elindítása után rendelkezésünkre áll minden beépített funkció, amit a Liferay biztosít a felhasználók és fejlesztők számára. Példa kedvéért állítsuk át a Portál nyelvét alapértelmezett angolról magyarra, ami talán a legegyszerűbb mégis nagyon fontos beállítás. A beállítási funkciókat bejelentkezés után tudjuk megtenni.

Add>More > Language

A Language Portletet egyszerűen csak be kell húzni az oldalon (drag and drop) tetszőleges pozícióba majd kiválasztani a választott nyelvet (Magyar). Egy multinacionális vállalat számára ez egy igen jelentős kényelmi funkció, mivel a különböző nemzetiségű dolgozók vagy akár partner vállalatok anyanyelvükön tekinthetik meg a portál funkcióit és tartalmát.



8. ábra – Liferay támogatott nyelvek

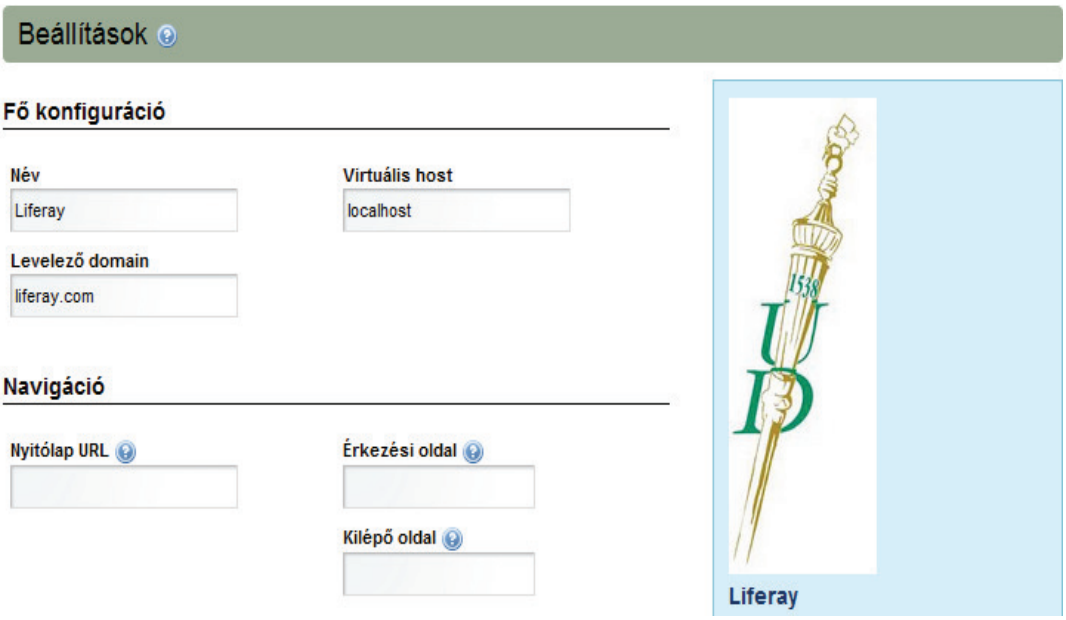
Természetesen, ha eltávolítjuk a Portlet-et a bezárás gombbal a nyelvi beállítások megmaradnak.

4.1.5 Minta adatok törlése

A frissen telepített Liferay minta adatokkal és példa beállításokkal van beállítva. A beállítások törléséhez a vezérlőpanel\szerver\frissítés kezelőjén keresztül kell eltávolítani a sevencogs-hook plugint (Ezt megtehetjük a Liferay-t megkerülve közvetlenül a web konténer vagy alkalmazásszerver admin felületén). Természetesen a mintaadatok visszaállíthatóak a plug-in újbóli telepítésével.

4.1.6 Liferay logó és powered by felirat eltávolítása

A Liferay alapértelmezés szerint saját logóját használja a portálrendszerben. Ez a megjelenítés viszont zavaró lehet egy cég számára, aki a Liferay-t akarja használni portálrendszerének. Erre a beállításra a vezérlőpanelen belül a beállítások menüben találunk lehetőséget.



9. ábra – Liferay Logo

Ugyancsak kényelmi beállítás az alapértelmezett Powered by Liferay felirat átváltása saját cégünket reprezentáló feliratra. Mivel a Liferay forráskódja hozzáférhető ezt könnyen megtehetjük a forráskód módosításával. A feliratot a Liferay könyvtárszerkezetén belül az alábbi template fájlban találhatjuk meg:

```
html\themes\classic\templates\portal_normal.vm
```

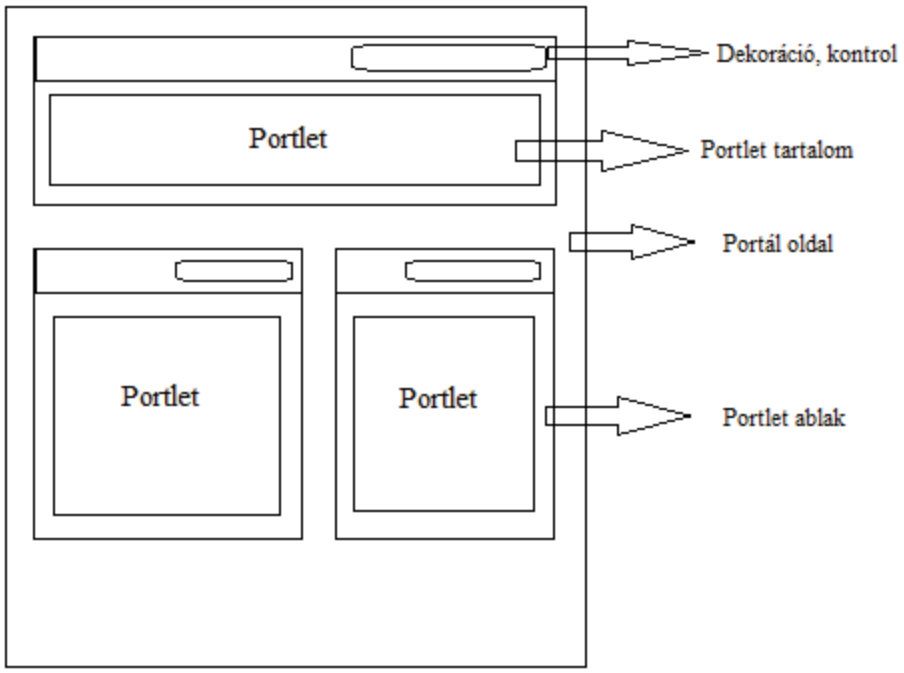
```
<footer id="footer" role="contentinfo">
          <p class="powered-by">
               #language("powered-by") <a
href="http://www.liferay.com" rel="external">Liferay</a>
          </p>
     </footer>
```

5. Portlet (JSR-168, JSR-286)

5.1. Portlet definíció

A **Portlet** fogalmat a Sun Microsystems hozta létre (mint az Applet és Servlet fogalmat), jelentése kikövetkeztethető, a portál egy kisebb (önálló) része.

A portletek web alapú komponensek, felhasználói interfésszel rendelkeznek, a felhasználó kéréseit dolgozzák fel amiből dinamikus tartalmat generálnak a felhasználó felé. A tartalom többféle lehet, HTML, XHTML, XML stb. A Portlet - eket a portletkonténer vezéreli. A portletek működését a JSR-168 és JSR-286 java specifikációk írják le pontosan. Bármilyen portál implementáció szívét a portletek adják, mivel ők képviselik a portálrendszer funkcionalitásait és az üzleti logikát. A portletek biztosítják a Liferay rugalmasságát mivel bármikor hozzácsatolhatóak, a rendszer újraindítása nélkül (hot-deploy). A Portlet - eket pluginként illeszthetjük hozzá a Liferay-hez. Egy ilyen plugin egy vagy több Portletet is tartalmazhat (.war kiterjesztésű fájlok). Portleteket bármilyen Java Web Framework segítségével készíthetünk amik támogatják a Portlet fejlesztést.



10. ábra – Portál oldal

A portletek portáloldalakon jelennek meg külön ablakban futva. Egy portáloldal egy vagy több Portlet futtatására alkalmas. Minden Portlet - nek van bizonyos dekorációs eleme, illetve vezérlő része (pl. bezárás gomb). A portletek kinézete konfigurálható (színek, szélesség, szövegstílus stb.). A különálló portleteknek különböző jogosultsági szinteket adhatunk. A portletek elhelyezkedése a futtató portál oldalon testre szabható a felhasználó által. A portáloldalhoz bármikor hozzá lehet adni egy új Portlet - et (Portlet ablakot) vagy meglévőt eltávolítani

A Portlet ablaknak különböző üzemmódjai lehetnek.

- Minimalizált (a Portlet - nek csak a fejrésze, neve látszik)
- Normál (a Portlet teljes tartalma megjelenik)
- Maximalizált (a Portlet teljes képernyős üzemmódban látszik)

Ezek mind olyan kényelmi funkciók, melyek megkönnyítik a végfelhasználó munkáját és olyan komfortérzetet sugallnak, melynek hatására a felhasználó úgy érzi, hogy a portáloldal teljes irányítása és testre szabása az ő kezében összpontosul.

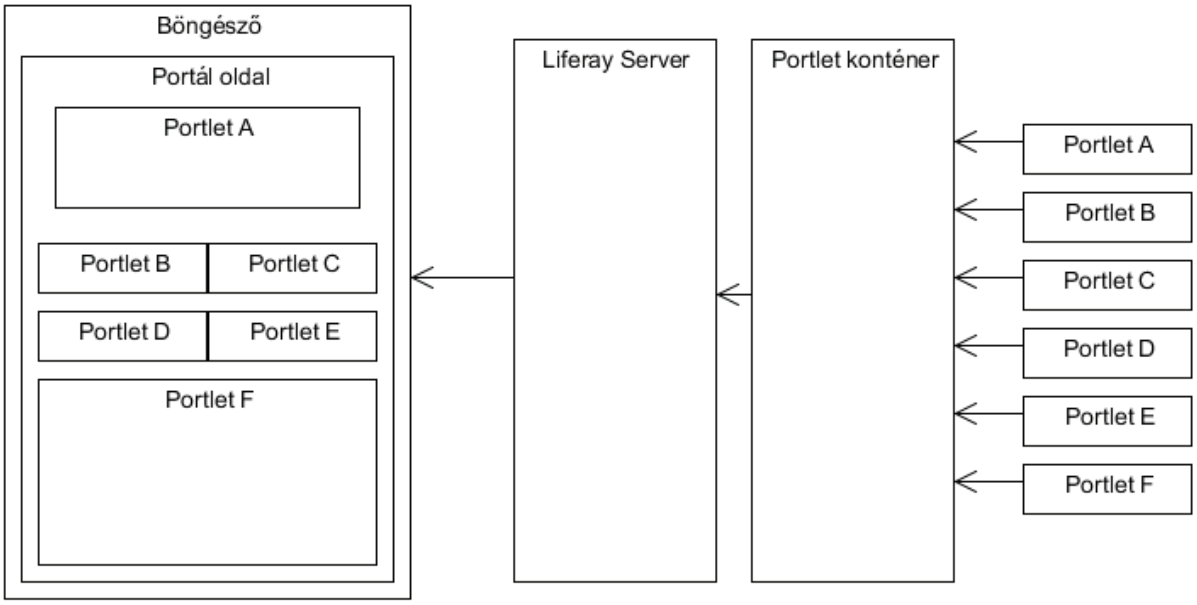
5.2. Portlet konténer

A Portlet konténer a portál része. A konténer legfőbb feladata hogy összefogja a portletek halmazát függetlenül attól azok hol találhatóak. A konténer hozza létre a Portlet példányokat, kéréseket fogad és küld a portletek felé és a konténer szabadítja fel azokat a Portlet - eket amelyekre már nincs szükség a futtatás során.

A konténer felelőssége:

- a felhasználói beállítások perzisztálása, és futtató környezet biztosítása a portletek számára
- a portletek közötti interakció és kommunikáció kezelése
- beérkező igények kezelése
- a Portlet által generált dinamikus tartalom megjelenítése

5.3. Portál oldal működése



11. ábra – Portál oldal működése

1. Mindegyik Portlet tartalmat generál a felhasználó felé, ami lehet dinamikus vagy statikus

2. Először a konténer kapja meg a generált tartalmat
3. A konténer kezeli a tartalmat majd továbbadja a portál szervernek
4. A portál szerver legenerálja a portál oldalt, ami igazából egy HTML (vagy egyéb output, amit a böngésző tud kezelni) kód amit a böngésző képes futtatni.
5. A portál szerver elküldi a legenerált tartalmat a böngésző felé
6. A böngésző megjeleníti a tartalmat és kész fogadni a felhasználó által küldött igényeket (rákattint egy gombra, esemény váltódik ki stb.)

5.4. Portlet üzemmódok

A Portlet - eket funkcionalitásuk szerint 3 féle üzemmódba csoportosíthatjuk.

- Megtekintés (View)
- Szerkesztés (Edit)
- Segítség (Help)

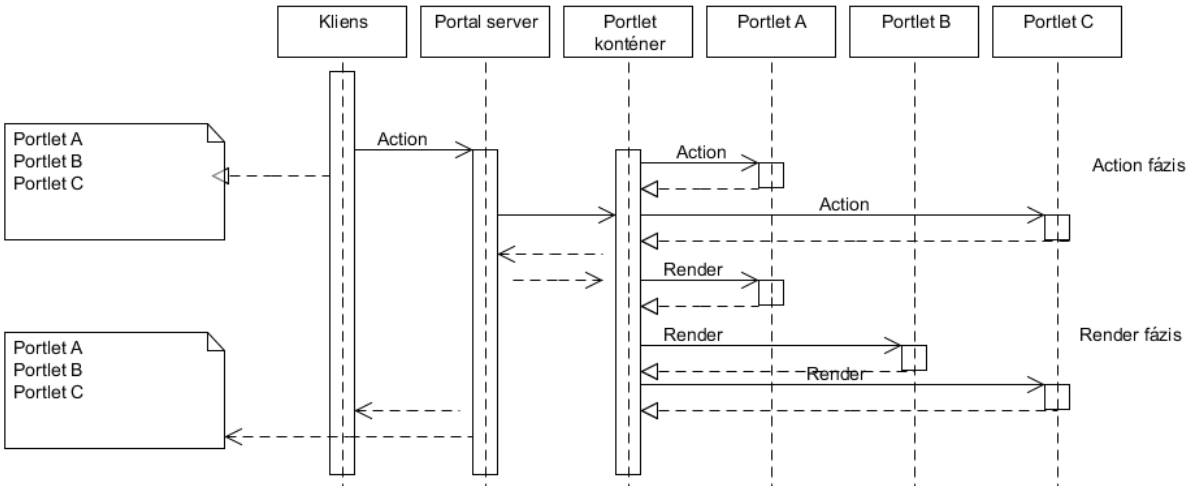
Az említett üzemmódokon kívül lehetőség van egyedi (custom) módok használatára is, illetve a portleteknek nem kötelező mindegyik módot használnia. A különböző módokat különböző jogosultsági szintekhez kapcsolhatjuk (tipikusan a felhasználó a view és help üzemmódot használja, az edit pedig a portál adminisztrátor jogosultsági körébe tartozik).

5.5. Bejövő igények kezelése (Request Handling):

A portletek két féle igény (request) kezelését teszik lehetővé.

- Action request (feldolgoz egy eseményt melynek hatására például elküldődik egy email vagy elmentődik egy rekord az adatbázisba)
- Render request (olyan esemény váltódik ki melynek hatására a Portlet megjelenítendő tartalmat generál)

Adott egy portál oldal ami tartalmaz 3 Portletet A-t, B-t és C-t. A felhasználó olyan eseményt generál, aminek hatására mindhárom Portlet tartalma megváltozik.

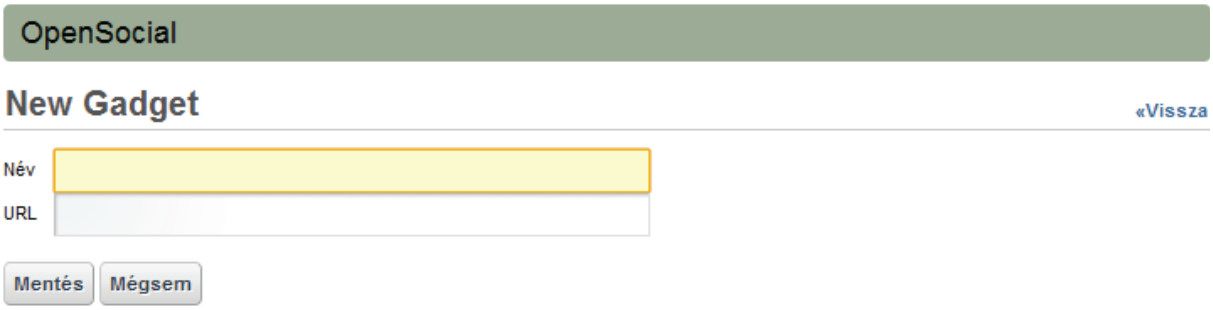


12. ábra – Portlet Request

1. Felhasználó kattint az oldalon, hogy frissüljön valamilyen információ
2. A felhasználói igény Action eseményt generál a portál szerverhez
3. A portál szerver eseményt generál a Portlet konténernek
4. A Portlet konténer eldönti hogy „A” és „C” Portletet kell frissíteni
5. A konténer Action requestet küld „A” Portletnek
6. „A” Portlet feldolgozza a kérést és válaszol a konténernek
7. A konténer Action requestet küld „C” Portletnek
8. „C” Portlet feldolgozza a kérést és válaszol a konténernek
9. Az Action fázis befejeződik, konténer minden adatot visszaküld a felhasználó felé amit „A” és „C” Portlet generált
10. Elkezdődik a renderelési fázis
11. A konténer renderelés eseményt küld mindhárom Portletnek
12. A konténer begyűjti a portletek válaszait
13. A konténer továbbküldi a választ a portál szervernek
14. A szerver megjeleníti a tartalmat a felhasználó böngészőjében
15. A request-response folyamat befejeződött
16. A szerver várakozik a felhasználó következő kérésére [23]

6. OpenSocial Gadgets

A Portlet fejlesztéshez hasonlóan a Gadget - ek fejlesztése is standard utat biztosít Liferay kiegészítők létrehozásához. Technológiai megközelítésből egy különbség van hogy a Gadget - ek nem tartalmaznak szerveroldali logikát. Egy másik különbség hogy Gadget - ekben úgymond közösségi alkalmazásokat készíthetünk, míg Portletek - ben bármilyen alkalmazás elkészíthető. Ebből adódóan a Gadget-ek nem biztosítanak olyan sok technológiát, mint a Portletek, de nagyon sok API létezik a funkcionalitások megvalósítására.



13. ábra – Liferay Gadget

Kétféle módon lehetséges OpenSocial Gadget-eket telepíteni (deployolni) Liferay alkalmazásunkba.

6.1. Távoli Gadget (Remote Gadget)

A távoli Gadget - ek egy távoli serveren futnak, viszont a felhasználó számára úgy tűnnek, mintha azok is az alkalmazás részei lennének. Ez egy nagyon egyszerű módszer más alkalmazások gyors integrációjához, viszont el kell fogadnunk a távoli szerver által okozott függőségeket. Például ha a távoli szerver üzemképtelen lesz, mi sem érjük el a távoli Gadget - eket. Intranetes alkalmazások fejlesztésekor ilyen Gadget - ekre nem támaszkodhatunk, mivel a hálózati környezet nem mindig teszi lehetővé a teljes Internet hozzáférést.

6.2. Helyi Gadget (Local Gadget)

A lokális Gadget - ek a Liferay serveren futnak ugyanúgy mint maga a Liferay és a portletek. A Gadget - ek egy XML fájlban vannak definiálva, így az egyetlen teendő az XML fájl feltöltése a Liferay serverre.

7. Újrahasznosíthatóság

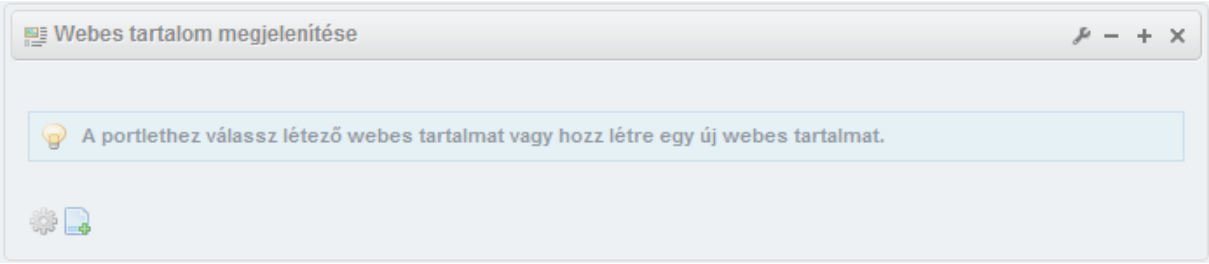
Előfordulhat az-az eset amikor a funkcionalitások amikre szükség van Liferay alkalmazásunkban már léteznek egy általunk írt web alkalmazásban, viszont az nem támogatja a Portlet - eket vagy az OpenSocial Gadget-eket. Egy különálló alkalmazás integrálása nem egy könnyű feladat. Nincs olyan megoldás, ami minden körülmények között működik, és ne kellene valamilyen kompromisszumot kötnünk.

Hogyan is importálhatnánk be az alkalmazást Liferay rendszerünkbe?

- újraírjuk az alkalmazást Portlet - ekben
- készítünk egy egyszerű Portlet - et ami kommunikál az alkalmazásunkkal (például web szolgáltatásokkal) és ugyanazt kínálja a felhasználónak mint a régi alkalmazásunk
- készítünk egy OpenSocial Gadget-et ami magába csomagolja régi alkalmazásunkat. A Gadget egy IFrame lesz, ami része lesz egy portál oldalnak.

7.1. Web Application Integrator

Ha az alkalmazásunk egy Java Web alkalmazás volt, akkor a Liferay Web Application Integrator (WAI) segítségével beépíthetjük rendszerünkbe.
A Liferay Web Application Integrator-nál csupán annyit kell tennünk, hogy az alkalmazásunk WAR – fájlját deployol - juk a szerverünkre mintha az egy Liferay plugin lenne. Az eredmény, hogy a Liferay automatikusan készít egy portletet ami integerálja alkalmazásunkat egy IFrame-ben.
„Megjegyzés: ha alkalmazásunk nem java web alkalmazás volt akkor ugyanez a funkcionalitás elérhető az IFrame portleten keresztül”



14. ábra – Liferay IFrame

8. Fejlesztési eszközök

A Liferay két Open Source technológiát biztosít az alkalmazásfejlesztéshez (Apache Ant-et és a Plugins SDK-t)

8.1. Plugins SDK

A Liferay biztosít egy fejlesztési környezetet, aminek a neve Plugins SDK, amivel lehetőséget biztosít bármilyen plugin előállítására előre meghatározott parancsokkal. Az SDK-t lehet használni parancssorból, bármilyen szövegszerkesztővel lehet szerkeszteni vagy bármelyik IDE-be integrálható.

8.2. Eclipse és a Liferay IDE

Az Eclipse az egyik legnépszerűbb IDE a java fejlesztők körében, ami nagyon sok funkciót biztosít a fejlesztés gyorsítására. A Liferay IDE egy plugin az Eclipse-hez, ami megkönnyíti a portálalkalmazásunk fejlesztését. A Liferay IDE a Plugins SDK-t használja, viszont elrejti a fejlesztő elől a kettőjük közti kommunikációt, ezzel is segítve a gyors fejlesztést.

„Megjegyzés: manapság már sok fejlesztő a Maven buildelési mechanizmusát preferálja jobban az Ant-al szemben, erre is van lehetőség mivel létezik Liferay Artifact amire könnyen hivatkozhatunk a pom.xml-ből bővebben :
http://www.liferay.com/web/thiago.moreira/blog/-/blogs/liferay-s-artifact-are-now-mavenized

```
<project>
  ...
   <repositories>
        <repository>
            <id>liferay-repository</id>
            <name>Liferay's
repository</name> <url>http://oss.sonatype.org/c
ontent/groups/public</url>
        </repository>
    </repositories>
    <dependencies>
        <dependency>
            <groupId>com.liferay</groupId>
            <artifactId>portal-
service</artifactId>
            <version>6.0.3</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    ...
</project>
```

Továbbá létezik alternatív út az Eclipse elkerülésére is:

http://netbeans.org/kb/articles/portalpack.html

”

9. Fejlesztés

9.1. Fejlesztési környezet felépítése

Első lépésként szükségünk lesz egy Liferay Portal-ra valamint a Plugins SDK-ra. A letöltés az alábbi linken elérhető.

http://www.liferay.com/downloads

9.1.1 Tomcat

A Tomcat lényegében egy web szerver, ami implementálja a Java Servlet illetve a Java Server Pages (JSP) specifikációkat. Ezeket a specifikációkat támogató servereket nevezzük Servlet konténernek. A Tomcat jelenleg a hetedik verziójánál tart és támogatja a Servlet 3.0, valamint a JSP 2.2 specifikációkat.

A portálfejlesztés során Tomcat-et fogok használni mivel kicsi, gyors és kevesebb erőforrást igényel mint más Servlet konténerek.
A letöltött Liferay portál tartalmaz egy Tomcat webkonténert, ennek elindítása windows alól tomcat bin mappájának **startup.bat** file-ja vagy egyéb operációs rendszerben parancssorból a **startup.sh.**
Az alapértelmezett port a 8080-as lesz, ügyelnünk kell rá hogyha más programunk használja ugyanezt a portot akkor nem fog elindulni a Tomcat - ünk és meg kell változtatnunk a Tomcat server.xml állományában.

9.1. 2 Ant konfiguráció

Mielőtt elkezdenénk a Portlet fejlesztést, szükségünk lesz egy Ant-ra. Az Ant letölthető a http://ant.apache.org/ címen. Az Apache Ant egy olyan eszköz, amely szoftverek automatizált buildelésére használható. Hasonló a C világában megszokott Make eszközhöz. Java-ban íródott és leginkább Java projektek buildeléséhez használják.
A letöltést követően tetszőleges mappába kicsomagoljuk és beállítjuk környezeti változóként, aminek a neve ANT-HOME lesz, ami az ANT könyvtárárra fog mutatni.

„Környezeti változók beállítása:
Windows:
http://www.chem.gla.ac.uk/~louis/software/faq/q1.html
Linux:
http://www.codecoffee.com/tipsforlinux/articles/030.html „

10. Portlet fejlesztés

10.1. Hello World Portlet

Az SDK gyökér-könyvtárában található a build.properties állományban állítsuk be a kívánt web-konténert, a megfelelő elérésekkel. Amennyiben több felhasználó is használja ugyanazt az SDK-t, a build.properties mellé hozzunk létre felhasználónként egy-egy build.${username}.properties fájlt, amelyben perszonalizálni tudjuk a beállításokat.

A Portlet készítés a Plugins SDK-val nagyon egyszerű. Létezik egy portlet könyvtár a Plugins SDK könyvtárában. Ebben a könyvtárban fognak elhelyezkedni portletjeink.
Első lépésben nevet kell adnunk a portletnek. Szükségünk lesz a project nevére szóközök nélkül és egy megjelenítendő névre (display name) ami tartalmazhat szóközöket.

A portlet mappában adjuk ki az alábbi parancsot, amivel elkészítjük első portletünket.

```
./create.sh hello „Üdvözöllek Portletemen”
```

A parancs hatására a portlets mappában létrejön a hello-portlet mappa
Ebből a mappából indítva egyetlen paranccsal kideployolhatjuk alkalmazásunkat Liferay szerverünkbe.

```
./ant deploy
```

10.2. A portletek anatómiája

A Portlet projektek általában 3 féle komponensből állnak.

- Java források
- Konfigurációs fájlok
- Kliens oldali komponensek (*.js, *.css, grafika)

Amikor a Plugins SDK-t- használjuk a projectünk a következő struktúrában épül fel.

```
/PORTLET-NEVE/
     build.xml
     /docroot/
            /css/
             /js/
            /WEB-INF/
                 /src/ (new jön létre automatikusan)
                 liferay-display.xml
                 liferay-plugin-package.properties
                 liferay-portlet.xml
                 portlet.xml
                 web.xml
            icon.png
            view.jsp
```

A portletek alapértelmezett módon az MVCPortlet keretrendszert használják, ami egy könnyűsúlyú keretrendszer, ami elrejti a portletek közötti komplexitást. Az MVCPortlet különálló JSP-oldalakat használ a Portlet összes oldalán (Portlet üzemmódnak megfelelően Edit.jsp, View.jsp, Help.jsp).

A java forráskódok a docroot/WEB-INF/src mappájában helyezkednek el.

A konfigurációs fájlok a docroot/WEB-INF mappájában helyezkednek el.

A standard JSR-286 portlet konfigurációs fájl a portlet.xml is itt van. A portlet.xml lényegében egy telepítési leírás a portál rendszer részére a portletről. A portál rendszer a telepítési leírás alapján hozza létre a portlet példányt.

```
<liferay-portlet-app>
    <portlet>
        <portlet-name>my-greeting</portlet-name>
        <icon>/icon.png</icon>
        <instanceable>false</instanceable>
        <header-portlet-css>/css/main.css</header-portlet-css>
        <footer-portlet-javascript>/js/main.js</footer-
portlet-javascript>
        <css-class-wrapper>my-greeting-portlet</css-class-
wrapper>
    </portlet>
    <role-mapper>
        <role-name>administrator</role-name>
```

```
        <role-link>Administrator</role-link>
    </role-mapper>
    <role-mapper>
        <role-name>guest</role-name>
        <role-link>Guest</role-link>
    </role-mapper>
    <role-mapper>
        <role-name>power-user</role-name>
        <role-link>Power User</role-link>
    </role-mapper>
    <role-mapper>
        <role-name>user</role-name>
        <role-link>User</role-link>
    </role-mapper>
</liferay-portlet-app>
```

Liferay esetében +3 darab konfigurációs fájl szükségeltetik. A Liferay konfigurációs fájlok opcionálisak, viszont nagyon fontos szerepet töltenek be, ha Portlet-ünket Liferay szerveren akarjuk használni.

- **liferay-display.xml:** Ez a fájl írja le hol fog elhelyezkedni a dockbar-ban a portletünk.

- **liferay-portlet.xml:** Ez a fájl opcionális konfigurációs Liferay specifikus beállításokat tartalmaz.. Például itt adhatjuk meg hogy a portletünk futhat-e ugyanazon az oldalon több példányban, és dolgozhatnak e különböző adatokkal.

- **liferay-plugin-package.properties**: Ebben a fájlban lehet megadni a portletünk különböző függőségeit.

A kliensoldali fájlok .HTML, .CSS, és JavaScript amik a userinterface-t alkotják ugyancsak ebben a mappában találhatóak. (Természetesen ezen belül olyan könyvtár szerkezetben ami logikus a fejlesztés szempontjából)

10.3. Portlet interfészek

10.3.1 GenericPortlet

Portlet készítéskor a Portlet osztálynak meg kell valósítania a javax.portlet.Portlet interfészt. Általános esetben elég, ha a Portlet osztály őse közvetlenül a GenericPortlet absztrakt osztály. Liferay esetében ez lehet a com.liferay.util.bridges.mvc.MVCPortlet ősosztály, ami szintén kiterjeszti a GenericPortlet absztrakt osztályt.


com.liferay.util.bridges.mvc

Class MVCPortlet

java.lang.Object
└ javax.portlet.GenericPortlet
└ com.liferay.portal.kernel.portlet.LiferayPortlet
└ com.liferay.util.bridges.mvc.MVCPortlet

All Implemented Interfaces:
EventPortlet, Portlet, PortletConfig, ResourceServingPortlet

Direct Known Subclasses:
JSPPortlet, TranslatorPortlet

15. ábra

10.3.2 Portlet interfész

Az interfész által definiált metódusok a következőek:

- **destroy ()** : void
- **init** (PortletConfig) : void
- **processAction**(ActionRequest, ActionResponse) : void
- **render**(RenderRequest, RenderResponse) : void

Az **init** metódus akkor hívódik meg amikor a portlet megkezdi életciklusát (új portletpéldány keletkezik).

A **processAction** metódust a konténer ActionRequest esetén hívja meg melynek paraméterei egy ActionRequest és ActionResponse objektum. Az ActionRequest tartalmazza az elvégzendő műveletekhez szükséges információkat, míg az ActionResponse tartalmazza a generált választ a konténer felé.

A **render** metódus a portleten megjelenítendő tartalomért felelős. Két paramétere a RenderRequest objektum, ami tartalmazza a tartalomgeneráláshoz szükséges információkat és a RenderResponse objektum melyben választ kap a Portlet konténer a tartalomgenerálásról.

A **destroy** metódust a konténer akkor hívja meg, amikor a Portlet életciklusa véget ért és meg kell szüntetni.

10.3.3 PortletConfig interfész

A PortletConfig interfészen keresztül lehet hozzáférni a Portlet számtalan beállításaihoz mint például:

- **getPortletName** (Portlet neve)
- **getResourceBundle** (nyelvi erőforrás)
- **getInitParameter** (Portlet beállításait adja vissza név alapján)
- **getInitParameterNames** (beállítások neveit adja vissza)

10.3.4 PortletRequest interfész

A PortletRequest interfész biztosítja a bejövő igény információiról a Portlet - et. A RenderRequest illetve az ActionRequest osztály implementálja ezt az interfészt.
Legfontosabb PortletRequest interfész által nyújtott szolgáltatások:

- **getPortletMode** (visszaadja a Portlet üzemmódját reprezentáló objektumot lásd. Portlet üzemmódok)
- **getWindowState** (visszaadja az ablak üzemmódját reprezentáló objektumot lásd. Portlet ablak üzemmódok)
- **getPortletSession** (eljárás segítségével hozzáférhetünk a Session információkhoz)
- **isUserInRole** (segítségével megállapíthatjuk, hogy van e megfelelő hozzáférése a felhasználónak a művelet elvégzésére)
- **getParameter** (segítségével kérhetjük le a különböző request paramétereket, például egy form tartalmát)
- **getPortalContext** (portál rendszerről szolgáltat információkat)
- **getPortletContext** (futási környezet adatai kérdezhetőek le a konténertől)

10.3.5 PortletResponse interfész

A PortletResponse interfészen keresztül tudjuk visszaadni a felhasználó számára generált információt és tartalmat. A RenderResponse és ActionResponse osztály implementálja ezt az interfészt.

11. Liferay IDE

A Liferay IDE az Eclipse IDE kiegészítése, ami lehetővé teszi a Portleteink (plugin) hatékony fejlesztését. 5 féle Liferay plugin fejleszthető vele

- Portlet
- Hook
- Layout template
- Theme
- Ext plugin

A Liferay IDE-hez Java EE fejlesztői csomag szükséges az Eclipse Galileo vagy Helios verziókból.

Eclipse plugin:

- Eclipse Helios verzióhoz:
  http://releases.liferay.com/tools/ide/eclipse/helios/stable/
- Eclipse Galileo verzóhoz:
  http://releases.liferay.com/tools/ide/eclipse/galileo/stable/

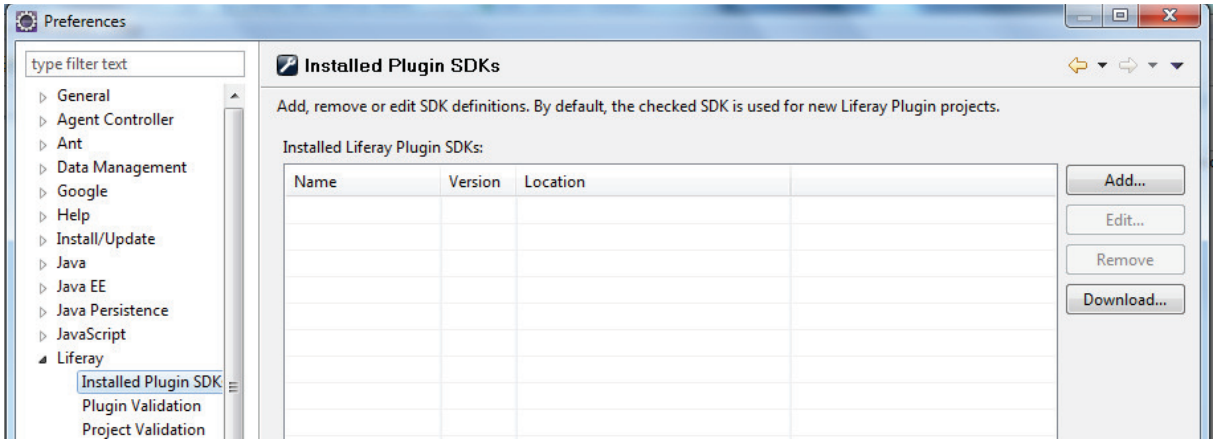
Az Eclipse plugin telepítés az Eclipse Help menüjéből érhető el az „Install New Software” menüpontból. Egyszerűen hozzá kell adni a letöltendő plugin linkjét a plugin manager-hez, majd az letölti és telepíti azt.

11.1. Eclipse konfiguráció

Ahhoz hogy eclipse-ben fejleszthessünk Liferay Portált szükséges még pár alapvető beállítás.

A plugin telepítése után be kell állítani a használni kívánt SDK-t (Software Development Kit). Ezt az alábbi menüpontban tehetjük meg:

Window > Preferences > Liferay > Installed SDKs



16.ábra- SDK beállítás

Az SDK beállítására két lehetőségünk van:

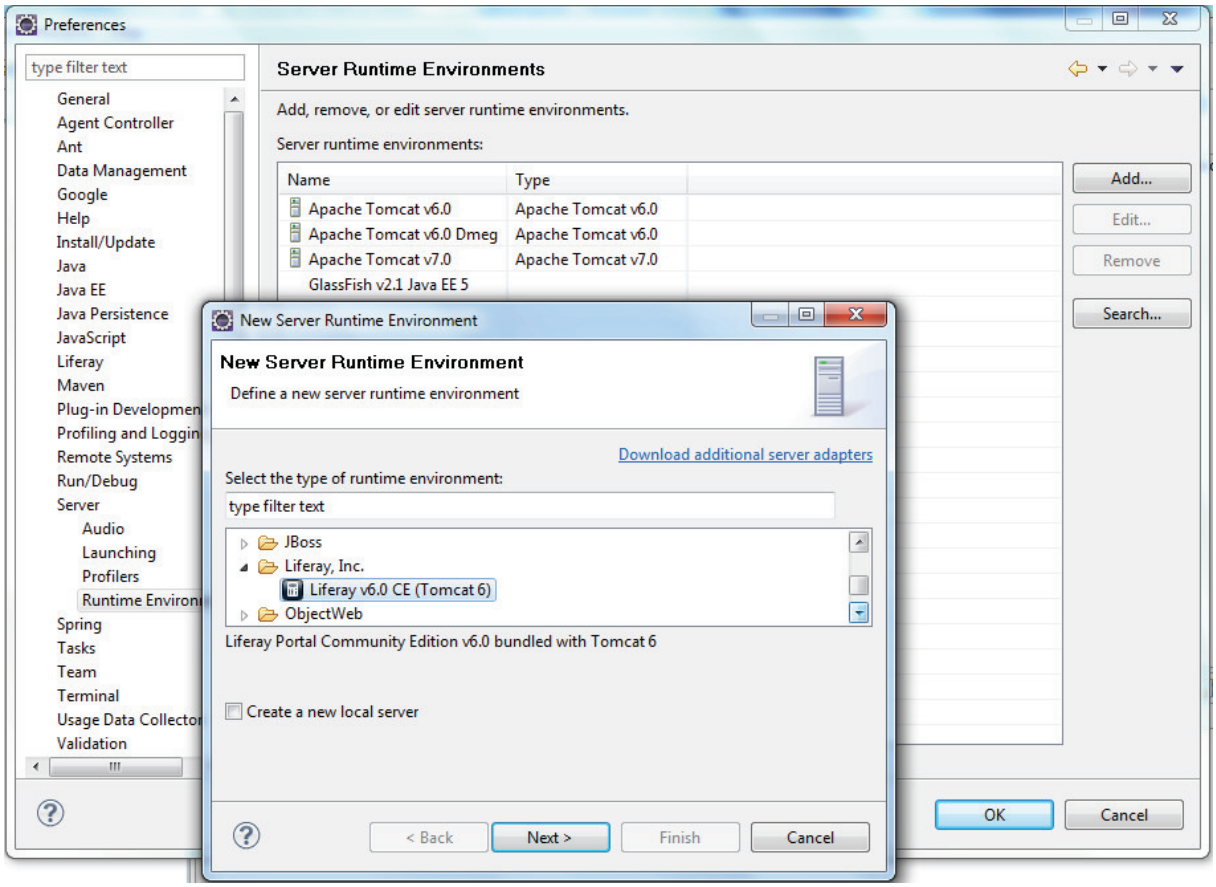
- Korábban letöltött SDK hozzáadása (Add)
- SDK letöltése Eclipse - en keresztül

Az SDK telepítése után be kell állítanunk a használni kívánt futtató környezetet.

Window > Preferences > Server > Runtime environments

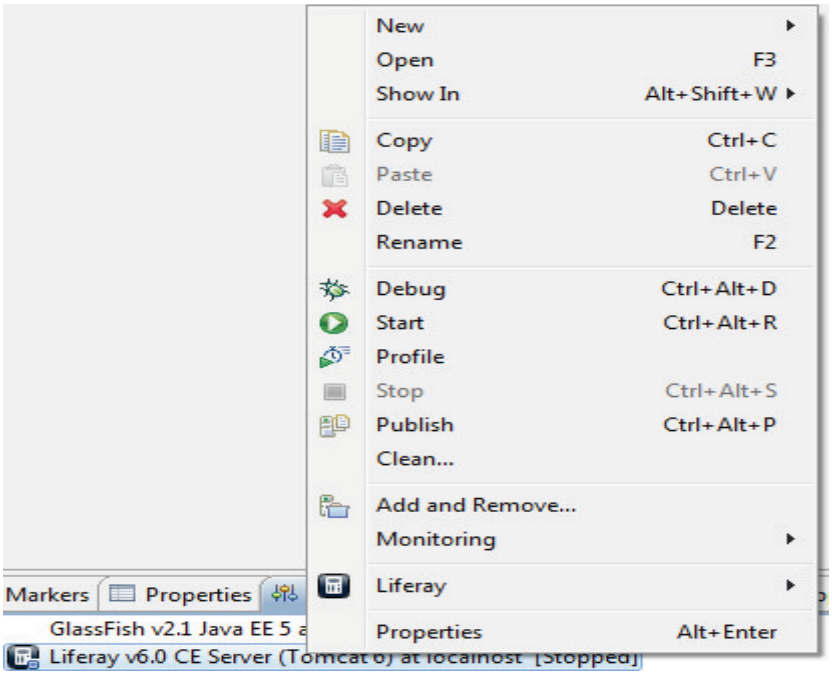
Futtatókörnyezetként a 6-os verziójú Tomcat-et fogom használni.

A beállítás során ugyancsak lehet választani előre telepített Tomcat szerverből, valamint használhatjuk az IDE által nyújtott „Szerver letöltés” funkciót.



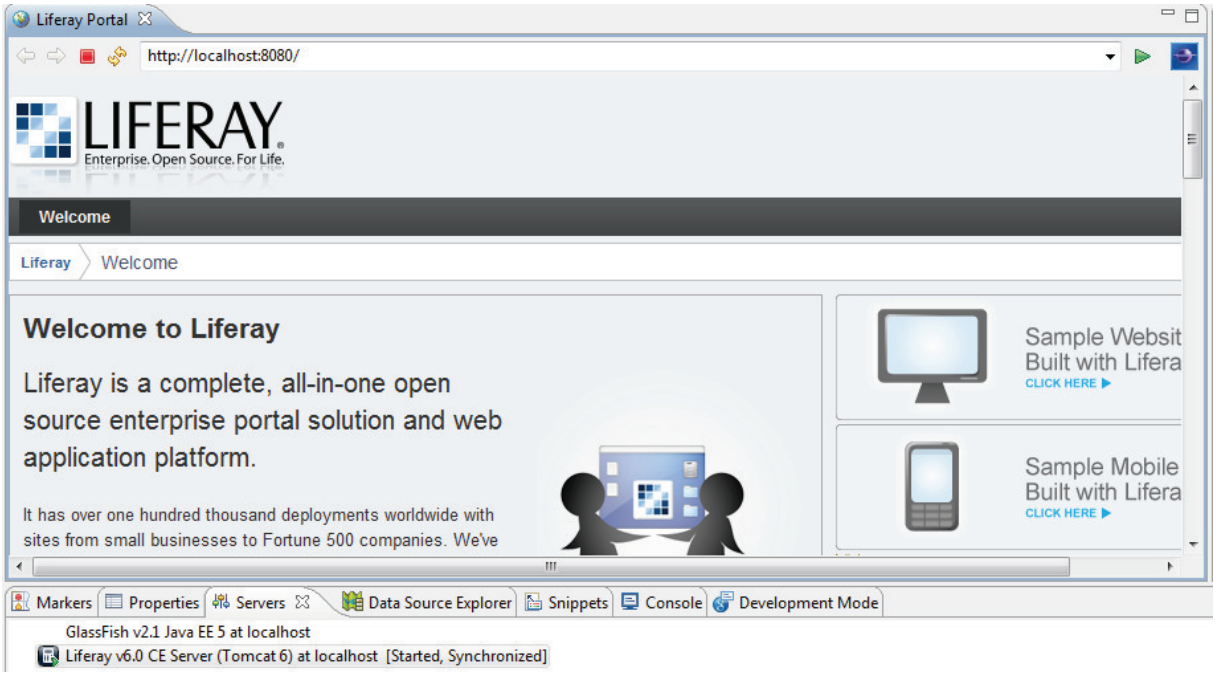
17. ábra – Tomcat beállítása

Az SDK és a szervertelepítés után tesztelhetjük a telepített környezetünket. (Start vagy Debug)



18. ábra – Liferay tesztelés

A szerverindítás után a localhost:8080-as címen a Liferay Portál elérhetővé válik. Az Eclipse tartalmaz egy beépített böngészőt a gyorsabb fejlesztés érdekében, a Liferay itt fog megjelenni.



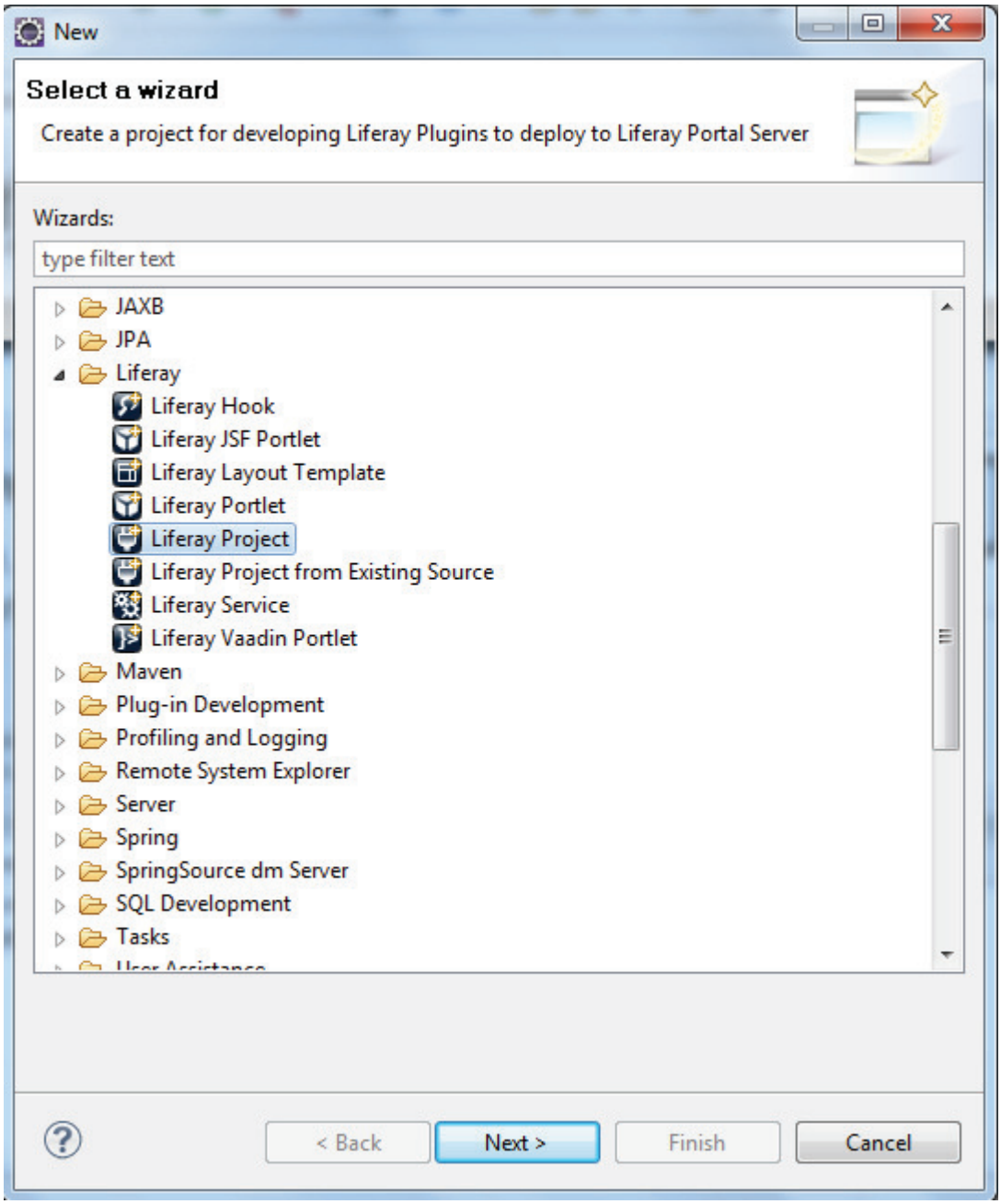
19. ábra - Liferay

11.2. Eclipse Plug-in Project

A fent említett konfigurációs beállítások után létrehozhatjuk első Liferay Project-ünket eclipse-ben.

`New Project... > Liferay > Liferay Plug-in Project`

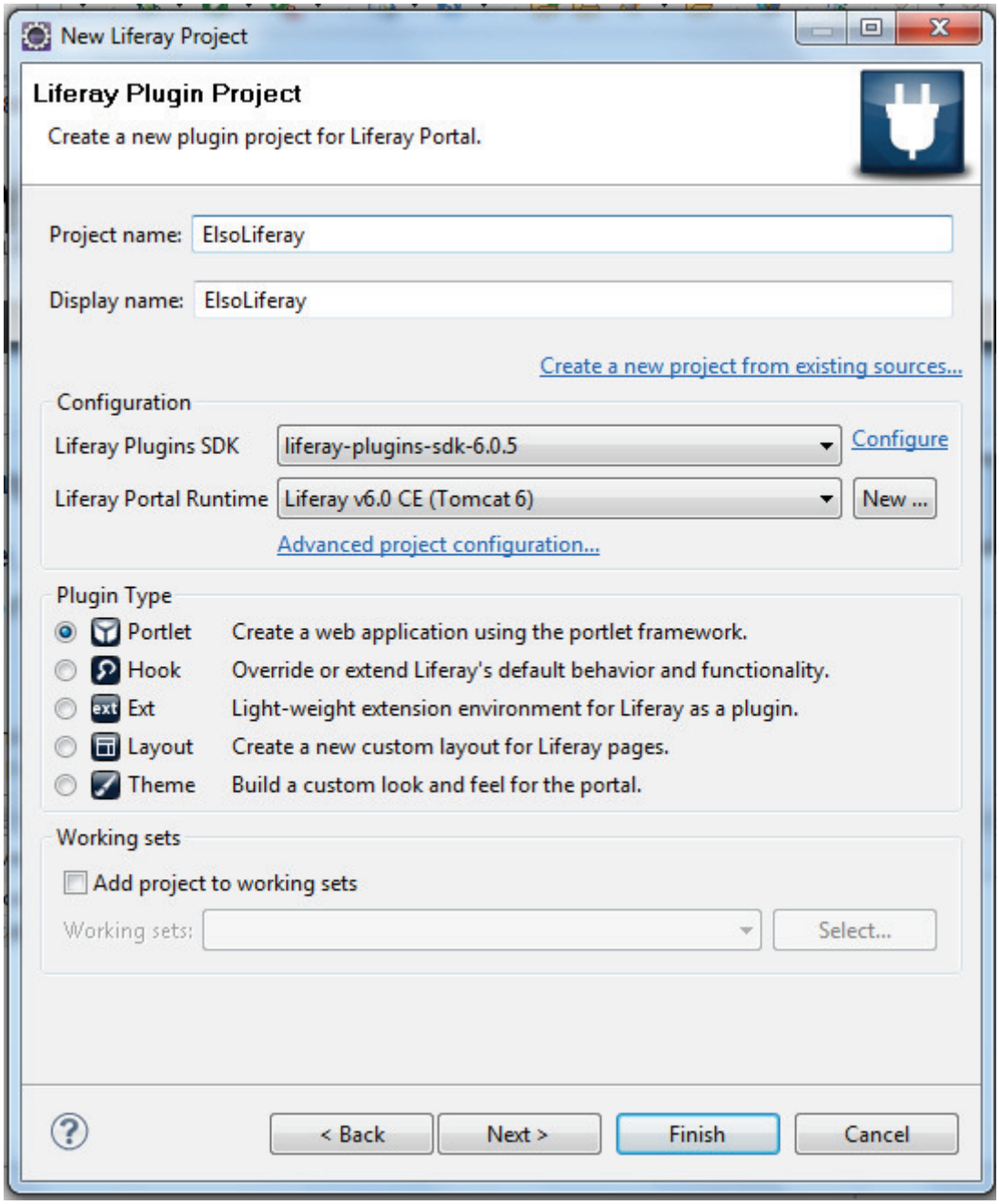
A projectvarázsló segítségével egyszerűen elkészíthetjük projectünket.



20. ábra – Liferay Project

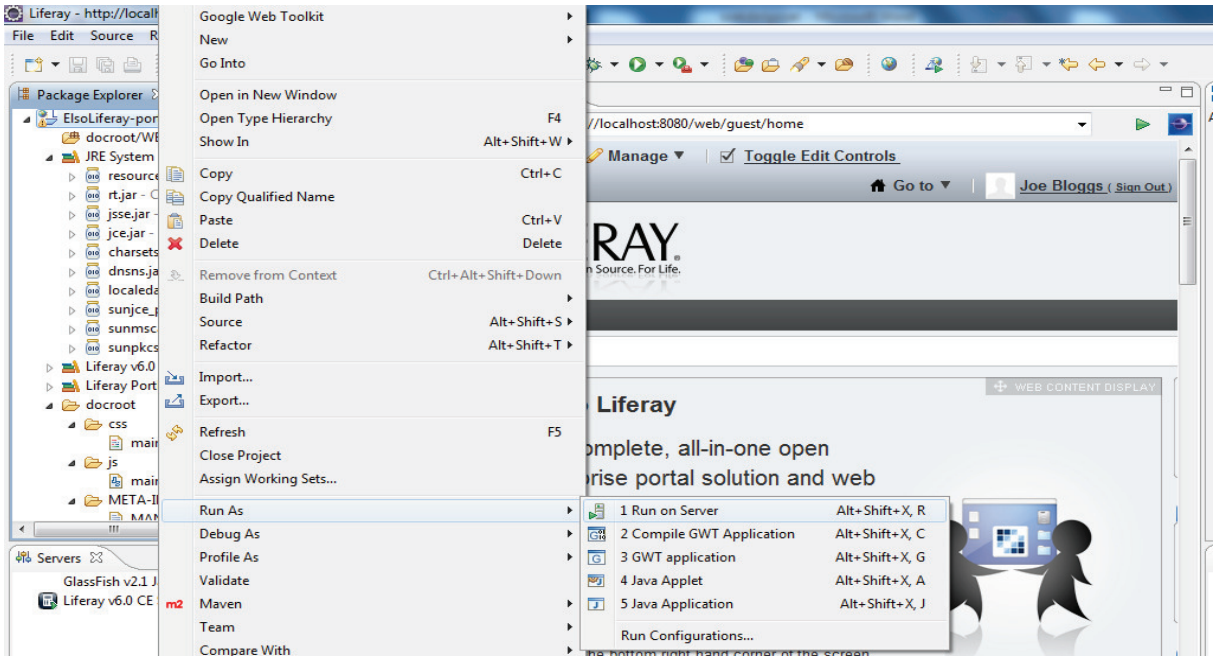
A Project létrehozása során meg kell adnunk a készített Plug-in

- Nevét (Project Name)
- A használt SDK-t (Liferay Plugins SDK)
- A futtatókörnyezetet (Liferay Portal Runtime)
- A Plugin típusát (Plugin type)

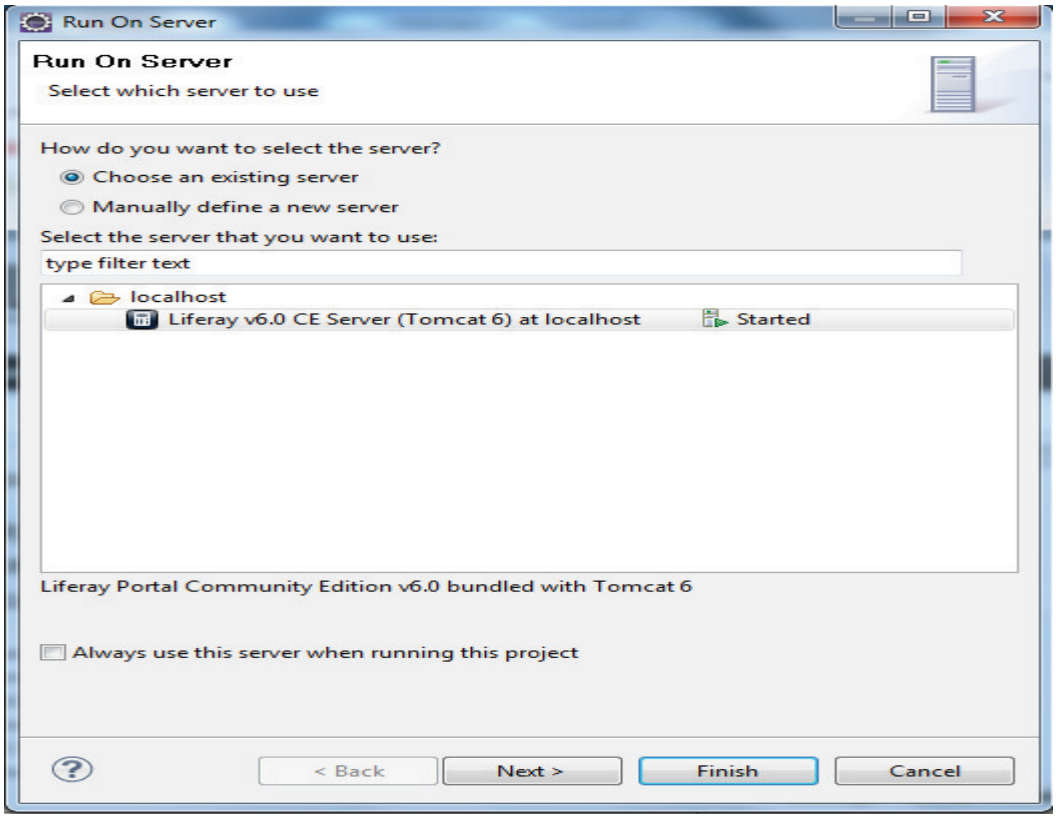


21. Liferay Project 2.

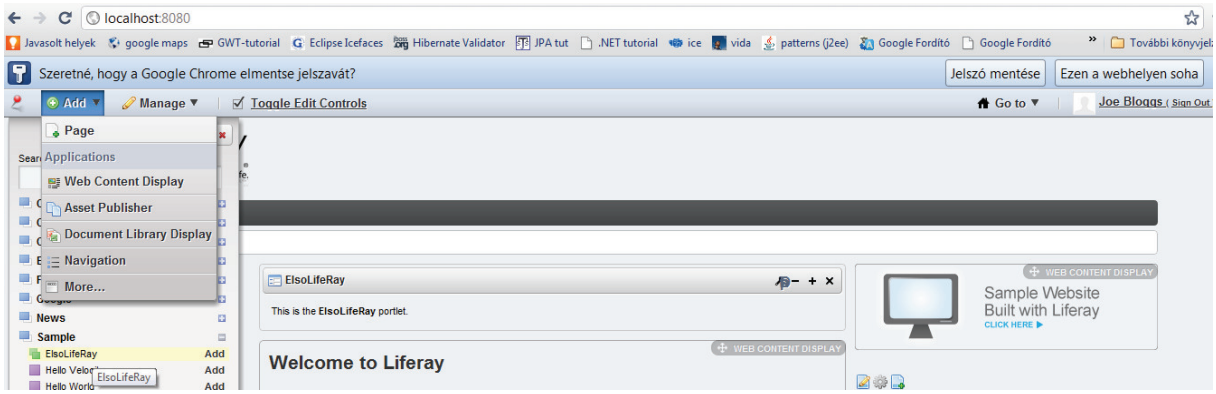
Az elkészült project megfelelő beállításokkal alkalmaz a futtatásra



22. ábra – Project futtatása 1.



23. ábra – Project futtatása 2.



24. Project futtatása 3.

12. Vaadin Plug-in

12.1. Vaadin

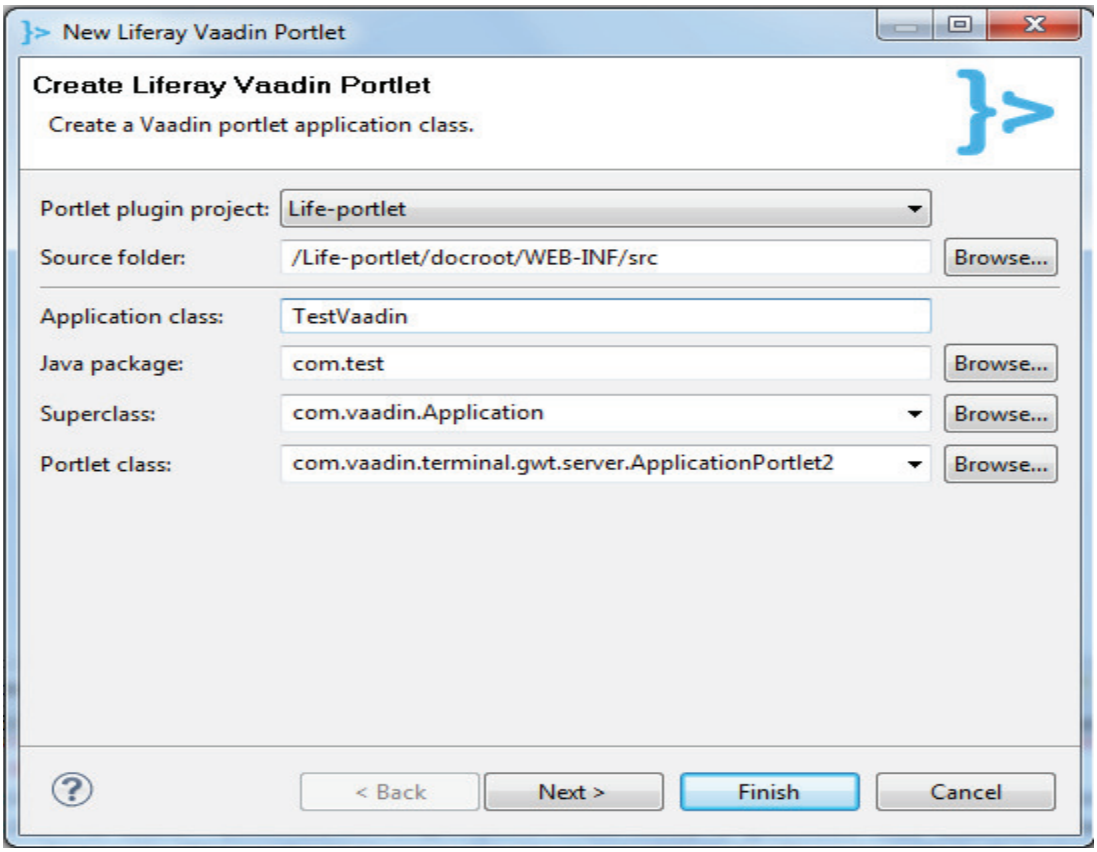
A Vaadin egy Java RIA (Rich Internet Application) Framework. A fő célja magas minőségű web alapú user interface készítése. A fő koncepciója, hogy úgy készíthetünk web user interface-t mintha az egy standard asztali java alkalmazás (AWT, Swing, SWT) lenne csak jobb minőségben. Elfelejthetjük a web kliens oldali architektúráját (HTML, JavaScript) és pusztán szerver oldalon fejleszthetünk, ami sokkal hatékonyabb fejlesztést tesz lehetővé, mivel több idő jut az üzleti logika megvalósítására. A szerver oldali programozási modellben a Vaadin gondoskodik a felhasználói felületről, nincs szükség különböző pluginokra, valamint a böngésző és szerver közötti AJAX-os kommunikációról. Továbbá a Vaadin keretrendszer böngészőfüggetlenséget biztosít, így nem kell a különböző böngészőkre optimalizálnunk alkalmazásunkat.

Támogatott böngészők:

- Internet Explore 6+
- Mozilla Firefox 3+
- Safari 3+
- Opera 10
- Google Chrome 4

12.2. Vaadin Portlet

Vaadin Portletet Eclipse IDE-n keresztül könnyedén hozzáadhatunk meglévő alkalmazásunkhoz néhány lépésen keresztül.

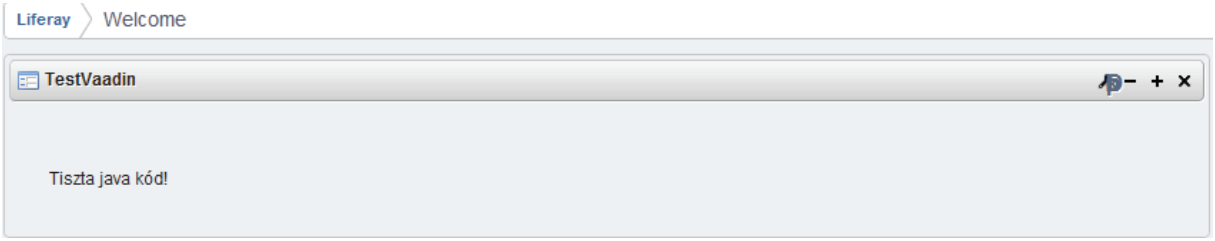


25. ábra – Vaadin Portlet 1.

A Vaadin alapú fejlesztés legnagyobb előnye, hogy elfelejthetjük a HTML-t és JavaScriptet és gondolkozhatunk tisztán Java kódban.

```
package com.test;

import com.vaadin.Application;

public class TestVaadin extends Application {

    public TestVaadin() {
    }

    public void init() {
        Window window = new Window("Ez egy vaadin portlet");
        setMainWindow(window);
        window.addComponent(new Label("Tiszta java kód!"));
    }

}
```

26. ábra – Vaadin Portlet 2.



27. ábra Vaadin Portlet 3

13. JSF Plug-in

13.1. JavaServer Faces

A JavaServer Faces (JSF) egy Java alapú keretrendszer mellyel webes felhasználói felületű alkalmazásokat lehet építeni. A JSF egy MVC-n (Model View Controler) alapú technológia. Nagyon jól elkülönül benne a belső logika és a megjelenítés. Számos UI komponens áll rendelkezésre hozzá melyeket különböző külső gyártók tesznek közzé (PrimeFaces, RichFaces, IceFaces stb.). Állapot és eseménykezelésre kiválóan alkalmazható technológia.

13.2. JSF - Liferay

Az alap probléma a Portletek-kel a felhasználható UI komponensek hiánya. Ezt a problémát csak úgy tudjuk kiküszöbölni, ha valamilyen JavaScriptes vagy ahhoz közelálló

keretrendszert alkalmazunk. A különböző Open Source JSF keretrendszerek nagyon sok ingyenesen használható komponenst kínálnak. Többek között ilyen keretrendszer az IceFaces. IceFaces segítségével ugyanolyan egyszerűen készíthetünk Portlet alkalmazásokat mintha csak JSF alkalmazást készítenénk. Az IceFaces egy hidat biztosít a JSF komponensek a portletek és a Portlet konténer között. Továbbá a Portlet 1.0 specifikáció nem tartalmaz portletek közötti kommunikációt viszont az IceFaces Ajax-os portletek egyszerű Ajax Push-t alkalmazva kommunikálhatnak egymással.

14. Liferay tesztelés

14.1. Miért is fontos a tesztelés?

Alkalmazásfejlesztés közben mindig adódnak hibák. Nincs tökéletes program, még a legjobb fejlesztők is követnek el hibákat. A hibák nem csak a fejlesztők munkájából (alkalmazás) adódhatnak.

A hibákat a következő csoportokba sorolhatjuk:

- **Error, mistake** (a programot az elvárttól különböző módon használják és ezért hiba alakul ki)
- **Defect, bug, fault** (fejlesztésből adódó hibák)
- **Failure (**konkrét hiba következik be, lehet Error vagy bug egyaránt)

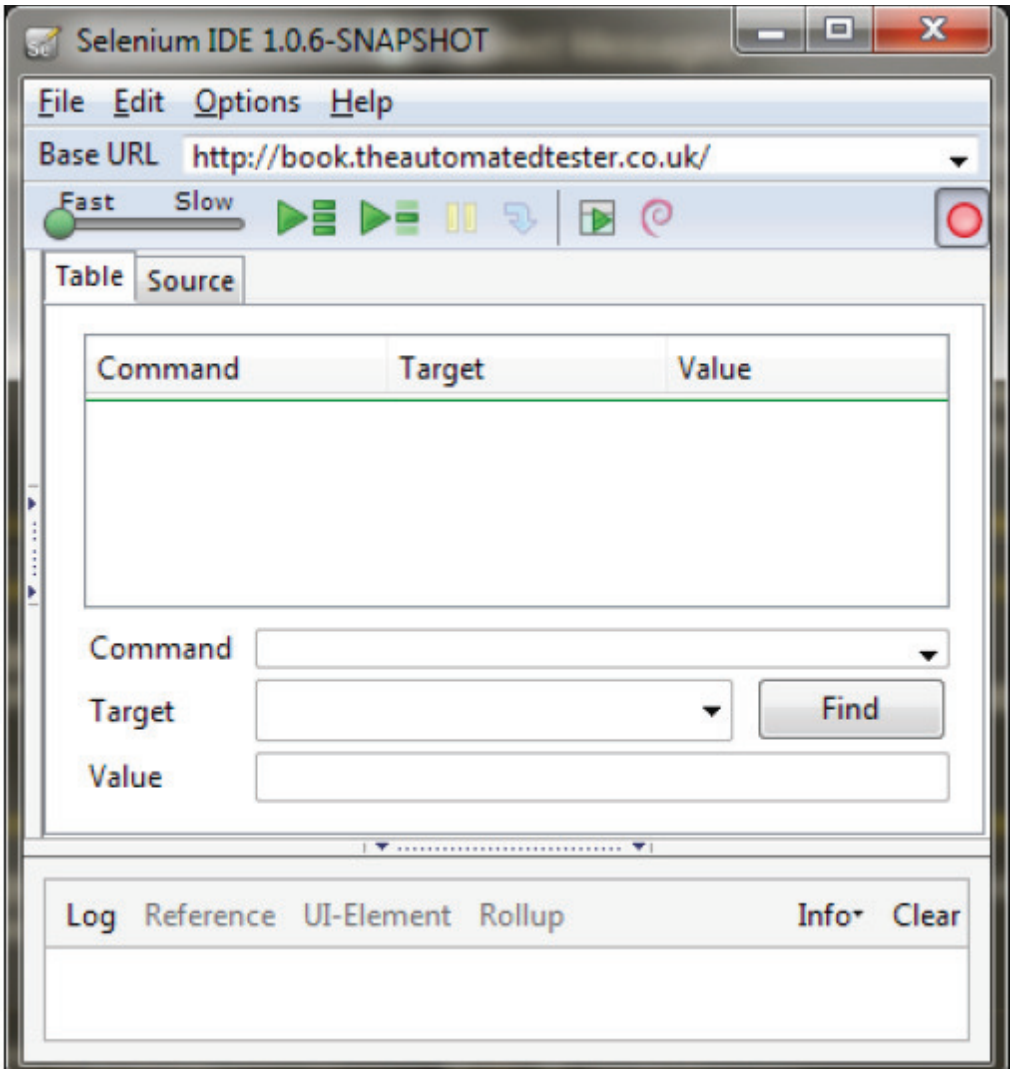
A hibák végső eredménye mindig valamilyen konzekvenciavesztés. A hibák egyaránt származhatnak a specifikációból vagy tervezésből, az alkalmazás használatából, környezeti hatásokból és emberi mulasztásból.

14.2. Selenium IDE – Tesztautomatizálás

Manapság a teszt automatizálás igen nagy népszerűségnek örvend a fejlesztők és a vállalatok körében. Ennek egyértelmű magyarázata, hogy a mai világban a fejlesztések felgyorsultak viszont a tesztelésre nem jut elegendő emberi és anyagi erőforrás. A fejlesztők viszont tudni akarják, hogy az alkalmazásuk két eltérő időpontban még mindig ugyanúgy funkcionálnak, mint elkészültük időpontjában. A fejlesztők nagyon sokféle tesztelést alkalmazhatnak, különböző metodikák és rendszerkövetelmények alapján.

A Selenium az egyik legnagyobb tesztelő keretrendszer, ami nagy népszerűségnek örvend a fejlesztők és tesztelők körében.A Selenium egy Open Source tesztelő keretrendszer, ami funkcionális tesztelést tesz lehetővé tetszőleges böngészőn keresztül. A Selenium támogatja a leggyakrabban használt böngészőket, mint Internet Explorer, Mozilla Firefox, Google Chrome, Opera és a magját egy JavaSript motor hajtja.

A Selenium a http://seleniumhq.org/download/ címen letölthető, ami egy Firefox bővítmény ezért szükség van hozzá Firefox böngészőre. Telepítést követően beépül a böngészőbe.



28. ábra - Selenium

A Selenium bővítmény úgy működik mint egy egyszerű video felvevő/lejátszó. A funkcionális tesztelés során felvett lépéseinket rögzíthetjük, majd visszajátszhatjuk segítségével. A JavaScript motorjának köszönhetően beazonosíthatjuk a weboldal bármely

komponensét és eseményeket generálhatunk a komponensen (Gomb kattintás, űrlap kitöltés). A felvétel során a Selenium egy scriptet készít lépéseinkről, amit tetszőleges programozási nyelvbe exportálhatunk, legyen az Java, C#, HTML. Az elmentett teszteket bármikor visszajátszhatjuk a fejlesztés bármelyik, szakaszában, így megbizonyosodva arról, hogy az általunk készített alkalmazás még mindig funkciójának megfelelően működik.

15. Összefoglalás

A Liferay a világ egyik legnagyobb portál alkalmazása. A Liferay a legnagyobb olyan portálszoftver gyártó cég, amely csupán egyetlen termékre összpontosít, így a legnagyobb figyelmet felhasználóira összpontosíthatja, és nem azt veszi alapul mivel érhet el bevételt többi szoftverénél (lásd Microsoft technológiák). A tipikus felhasználói tábora a közép illetve nagyvállalatok Intranetes rendszere, de alkalmas internetes tartalommegosztói funkciók betöltésére egyaránt. A Liferay-t használva a vállalat időt és pénzt takaríthat meg. A Java technológiák standard - jeire épülve szinte univerzális konfigurálási lehetőséget biztosít a fejlesztők és portál adminisztrátorok számára. Open Source mivoltjából adódóan nincs olyan probléma, amit ne lehetne orvosolni rövid idő alatt. Egyedülálló módon integrációt biztosít nem Java alapú fejlesztések integrációjához legyen az Ruby, PHP, Python, Grails stb. így már meglévő webes szoftverkomponensek is integrálhatóak a portálrendszerbe. Beépített funkcióinak köszönhetően a fejlesztők válláról hatalmas munkát vesz le. A Liferay követi a kor modern technológiáit és a nyílt szabványokat követve ezzel is csökkenti a továbbfejlesztések költségeit.

15.1. CMS Matrix

A CMS Matrix egy web oldal ahol a világ különböző portál, illetve CMS rendszereit lehet nagyon sokféle szempontból elemezni és összehasonlítani. A Liferay ezen oldal adatai alapján is igen kiemelkedő helyet foglal el még úgy is hogy nem a legújabb Liferay verzió adatai szerepelnek a rendszerben. Szinte minden funkcionalitást tartalmaz, ami más rendszerekben megvalósult. [20]

	Drupal 6.10	Joomla! 1.6.0	Liferay Portal 5.2
Last Updated	2/26/2009	2/15/2011	8/10/2009
System Requirements	Drupal 6.10	Joomla! 1.6.0	Liferay Portal 5.2
Application Server	Apache	CGI	J2EE
Approximate Cost	Free	Free	Also available in an Enterprise Edition with professional support.
Database	MySQL	MySQL	DB2
License	Open Source	Open Source	Closed Source
Operating System	Platform Independent	Platform Independent	Platform Independent
Programming Language	PHP	PHP	Java
Root Access	No	No	Yes
Shell Access	No	No	Yes
Web Server	Apache	Any	Other
Security	Drupal 6.10	Joomla! 1.6.0	Liferay Portal 5.2
Audit Trail	Yes	No	Yes
Captcha	Free Add On	Free Add On	Yes
Content Approval	Yes	Yes	Yes
Email Verification	Yes	Yes	Free Add On
Granular Privileges	Yes	Yes	Yes
Kerberos Authentication	No	No	Yes
LDAP Authentication	Free Add On	Yes	Yes

29. ábra – CMS Mátrix

15.2. Személyes vélemény

Immár 3 éve foglalkozom java alapú webes szoftverfejlesztéssel. A Java technológiák és keretrendszerek hatalmas szabadságot biztosítanak a szoftverfejlesztők számára. Munkáim során nem találkoztam olyan problémával, amit ne lehetett volna megoldani valamilyen nyílt forráskódú alkalmazás/keretrendszer segítségével. 3 év alatt megismerhettem több nagyobb magyar vállalat informatikai rendszerét. Részt vettem olyan szoftverek fejlesztésében, amelyek kimondottan vállalati igények kielégítésére épültek. Saját bőrömön tapasztalhattam meg egy-egy üzleti funkció fejlesztése milyen körülményes lehet emberi, technikai és pénzügyi szempontból. A Liferay architektúrálisan egy biztos keretet nyújt egy vállalati portálalkalmazás építésében. Számos olyan beépített funkcióval rendelkezik, amiket a legtöbb vállalat használ, így ezeknek a funkcióknak a lefejlesztése továbbá feleslegessé válik. A Liferay egy robosztus portál megvalósítás, ami könnyen skálázható, így elkerülve a teljesítményből adódó problémákat. Nem szükséges olyan mély szakmai ismeret a fejlesztéséhez mint egy egyedi fejlesztéshez, támaszkodhatunk a Liferay fejlesztőinek és magára az Open Source világ szakértelmére. Liferay-re támaszkodva egy a világban elismert standard-re építhetjük alkalmazásunkat. Nehezebben történhet meg az az eset amikor egy fejlesztő kezében összpontosul az architektúrális és üzleti logika túlnyomó része.

Dolgozatom és eddigi tapasztalataimat figyelembe véve bátran ajánlom a Liferay-t más fejlesztők és vállalatok számára, valamint következő munkáimban továbbra is mérlegelni fogom a Liferay bevethetőségének lehetőségét.

16. Irodalomjegyzék

[1] GNU-LGPL:
http://gnu.hu/lgpl.html

[2] EPAM - Nádudvari Mihály(Liferay):
https://docs.google.com/leaf?id=0B-LJCg4j57VNMTE5NzRjNTYtMDc0ZS00ZjQyLTlhYTgtNTljYmFiMDAxNzU2&hl=en&authkey=CP-864QK&pli=1

[3] Open Source Initiative
http://www.opensource.org/osd.html

[4] Dr Cséfalvay Miklós PhD, Budapesti Műszaki Főiskola – Open source éljünk szabad hozzáférésű szoftverekkel
http://regi.kvk.uni-obuda.hu/konf2008/doc/cikkek/csefalvay_miklos.pdf

[5] Informatikai Vállalkozások Szövetsége
www.ivsz.hu

[6] Liferay Portal 6.0 – Development Guide
http://www.liferay.com/documentation/liferay-portal/6.0/development

[7] Google Gadget Library
http://www.google.com/ig/directory?synd=open

[8] Icefaces
http://www.icefaces.org/main/home/

[9] Richfaces
http://www.jboss.org/richfaces/

[10] Apache Ant
http://ant.apache.org/

[11] Kovács Richárd Java blogja – Barátkozás a Liferay-vel
http://jpattern.blogspot.com/2010/06/baratkozas-liferay-portlettel-telepites.html

[12] Magyar Liferay Közösség
http://www.liferay.com/community/forums/-/message_boards/category/4087838

[13] Magyar Liferay specialista
http://www.i-logic.hu/liferay

[14] László Dávid : Webes alkalmazások Fejlesztése, Java portál technológia segítségével
http://jcp.org/en/jsr/detail?id=286

[15] JSR-168/286 Specifications
http://developers.sun.com/portalserver/reference/techart/jsr168/

[16] Javaforum – Mi a Portál?
http://wiki.javaforum.hu/display/JFPORTAL/Portal

[17] Portál megvalósítások
http://java-source.net/open-source/portals

[18] Eclipse plugin telepítése
http://agile.csc.ncsu.edu/SEMaterials/tutorials/install_plugin/index_v35.html

[19] David Burns : Selenium 1.0 Testing Tools Beginner's Guide

[20] CMS Matrix
http://www.cmsmatrix.org/

[21] Antonio Goncalves : Beginning Java EE 6 Platform whith Glassfish 3 From Novice to Professional

[22] Doroty Graham, Erik Van Veenendaal, Isabel Evans, Rex Black : Foundations of Software Testing

[23] Liferay Portal 6 Enterprise Intranets

17. Köszönetnyilvánítás

Ezúton szeretném köszönetemet és tiszteletemet kifejezni mindenkinek, aki a diplomamunkám elkészítéséhez hozzájárult.

Elsősorban köszönöm családtagjaimnak és kedvesemnek, Lillának, akik egyetemi tanulmányaim alatt végig támogattak szeretetükkel és bíztatásukkal.

Köszönöm tanáraimnak, akik remek szakmai hozzáértésükkel tanították mindazt amire a mai kor Mérnök Informatikusának szüksége lehet.

Tanáraim közül elsősorban témavezetőmnek Bátfai Norbertnek, aki diplomamunkámon kívül számos területen tanított újat és hasznosat. Bátfai Norbert volt tanáraim közül az első, aki az elméleti oktatáson túlmutatóan, olyan gyakorlati ismeretekkel látott el a szakmában, amiket a mindennapi munkám során hasznosítani tudok. Bátfai Norbert oktatásának és támogatásának köszönhettem két Nyári Szakmai ösztöndíjamat valamint a három féléves demonstrátori munkámat Magas szintű Programozási Nyelvek 1-2 valamint Operációs Rendszerek tantárgyakból. Norbi nemcsak tanárként, hanem barátként is támogatott engem és hallgatótársaimat tanulmányaim során. Köszönöm!

Dolgozatom felépítésében bár nem, de szakmai és erkölcsi támogatásban segítségemre volt egyetemi tanulmányaim során mindvégig Dr. Végh János és Dr. Almási Béla tanár úr. Köszönöm Végh tanár úrnak a változatos Programozás labor tárgyakat, ahol a mindennapi életben előforduló problémákat és változatos feladatokat oldhattunk meg. Köszönöm Almási tanár úrnak azt a precizitást és fegyelmet, amivel oktatta különböző hálózati tantárgyait. Ennek a precizitásnak köszönhetően tudatosulhatott meg bennem annak az ismerete, hogy egy apró hiba is végzetes lehet a való életben.

Végül, de nem utolsó sorban szeretném kifejezni köszönetemet és megbecsülésemet legjobb barátomnak Ráthonyi Tamásnak, aki szakmailag és emberileg is mindig mellettem állt és segített, ha bajban voltam! Köszönöm!