Debreceni Egyetem

Informatika Kar

XML alapú felhasználói felület leíró nyelvek

Témavezető:
Jeszenszky Péter
egyetemi tanársegéd

Készítette:
Hasulyó László
Programozó matematikus

Debrecen

2007

Tartalomjegyzék

1. Bevezetés

A felhasználói felület leíró nyelvek a programfejlesztésben a felhasználói felület nem kis időráfordítást igénylő elkészítését, dizájn kialakítását, és nem utolsósorban a webes alkalmazások felületének kliensoldalon történő felépítését teszik könnyebbé. A kliensoldali előállítás igényének az okai:

- a sávszélesség problémákból fakadó igények.
- a szerver oldali terheltség csökkentése.
- A RIA (Rich Internet Applicaton) igényének megjelenése.

A gazdag internetes alkalmazások (RIA-k) olyan webalkalmazások amelyek az asztali alkalmazások tulajdonságaival bírnak. A gazdag internetes alkalmazások jellemzően átküldik a megjelenítéshez szükséges adatokat, amelyeknek a feldolgozása a kliens gépen történik, de az adatok nagy többsége (pl. a program állapota) az alkalmazásszerveren marad, illetve ott van feldolgozva. Céljuk, az hogy az asztali alkalmazásokkal megegyező felhasználói élményt nyújtsanak.

A leíró nyelvek által a felület sokkal rugalmasabban alakítható ki, apróbb-nagyobb változtatások is könnyebben kivitelezhetőek. A kinézettel nem kell foglalkozniuk az üzleti logikát íróknak, a kinézettel foglalkozó szakemberek viszonylag függetlenül is tudnak dolgozni a felületen. Felületleíró eszközökre régebben is voltak megoldások, de nem voltak platformfüggetlenek, sem széles körben elterjedt megvalósítások, inkább vállalati, ill. fejlesztőeszköz-specifikus megoldások jöttek szóba. Kellett egy egységes, elterjedt eszköz, ami ezektől a hibáktól mentes. Az XML (Extensible Markup Language) megjelenésével, – amivel hordozható adat-leíró nyelveket hozhatunk létre – végre a hordozhatóság is szóba jöhetett a különböző operációs rendszerek, programozási nyelvek között. Az XML szó szerint bővíthető jelölő nyelvet jelent, mely egy olyan szintaxist ad meg, amelyet betartva különböző jelölő nyelvek (mint például az XHTML) hozhatóak létre. Az XML kiterjeszthetősége lehetőséget ad a dokumentumok (adatok) tartalmi szempontok alapján történő leírására. Egy XML dokumentum elemekből áll, amelyek neve (szókincs), egymáshoz való kapcsolata és tartalma szabályokkal rögzíthető (nyelvtan).

Az XML specifikáció megad egy szintaxist mind az XML dokumentumokra – vagyis az elemek jelölésére – mind a szabályok leírására (DTD). A megadott szintaktikai szabályok betartásával bárki saját nyelvet (dokumentum-típust) készíthet, s azt a megfelelő XML-

konform eszközzel ellenőrizheti, feldolgozhatja. A nyelvtan és a szókincs megadása nem kötelező. Az XML elsődleges célja strukturált szöveg és információ megosztása az interneten keresztül. Az ilyen alaptulajdonságokkal rendelkező felület leíró nyelv kiléphetett az őt megalkotó környezetéből, bosszantó inkompatibilitási problémáktól mentesen (pl. a karakterkódolást máshol máshogy kezelik, vagy az egész típus tárolási mérete más operációs rendszeren nem ugyanannyi bájt) terjedhet el a fejlesztők, felhasználók között. Mivel az XML nyílt szabvány, így korlátok nélkül terjedt el. Mivel egy problémára születhet több megoldás, ezért több XML alapú elképzelés is megjelent. Ezeknek egyik fő különbségük, hogy mi a céljuk egy felület készítésének segítségében. Egyesek egy mindent tudó felületleíró nyelvet akarnak megvalósítani, mások egy adott programozási nyelv, vagy webalkalmazások segítségére specializálódnak. Utóbbiak sokkal könnyebben megvalósíthatóak, mint egy univerzális nyelv, amit nehéz implementálni. Az írásom további részében bemutatok néhány megvalósított nyelvet, és azoknak az alkalmazásának lehetőségeit.

2. Az XML

2.1 Az XML története

Az XML ötlete akkor merült fel, mikor Tim Bray az IBM, az Oxford University Press és a University of Waterloo által támogatott internetes szótáron dolgozott. Mivel hatalmas mennyiségű adatot kellett tárolni és feldolgozni, kereskedelmi szoftvermérnököket vontak be a projektbe, hogy megoldást találjanak az adatok indexelésére és tárolására. Az Association for Computing Machinery (ACM) számára adott interjú során Bray azt nyilatkozta, amikor bevonták a projektbe és megmutatták neki a szótár számára kifejlesztett belső struktúrát: "kis beágyazott címke határozta meg, hogy mi a bejegyzés illetve szó, és aztán kiejtés, etimológia, rövid idézet, és aztán adat, forrás, szöveg és így tovább" (ACM Queue, 2005). Ez vált az XML elődjévé. Miután kifejlesztették a technológiát a szótár projekthez, Bray megalapította az Open Text Corporation-t, kifejlesztett egy kereső motort, valamint meghívták a W3C-be, hogy legyen az XML specifikációjuk szerkesztője.

2.2 leírása

Az így kifejlesztett XML két meglévő nyelv, a HTML (HyperText Markup Language, hipertext jelölőnyelv) és az SGML (Standard Generalized Markup Language, szabványosított általános jelölőnyelv) elveire és szabályaira épül. Alapja elsődlegesen az SGML. Az ajánlás 1998-ban jelent meg. Sikerében használhatósága mellett további jelentős szerepet játszott az is, hogy az XML bővíthető és platformfüggetlen. Az SGML 1986 óta ISO szabvány, nagyfokú bonyolultsága miatt azonban nem terjedt el széles körben. Ahogy azt Drótos László (2001) idézi, az SGML betűszót „újra alkották" a következőképpen: „Sounds Good Maybe Later" (Jól hangzik, talán később [jó is lesz].)
Az XML egyszerűbb és kihasználja az SGML adta egyedülálló technológiát, kompatibilis vele, hiszen annak részhalmaza, használata azonban jóval egyszerűbb, mint az SGML-é.

Az XML legfőbb jellemzője: a tartalom, a megjelenítés különválasztása, amelyek így külön-külön optimálisan kezelhetők.

2.2 Az XML előnyei

- mind ember, mind gép számára olvasható formátum
- támogatja az Unicode-ot, ami lehetővé teszi bármely információ bármely emberi nyelven történő közlését
- képes a legtöbb általános számítástudományi adatstruktúra ábrázolására (rekord, lista, fa...)
- öndokumentáló formátum, amely struktúra- és mezőneveket ír le speciális értékekkel együtt
- szigorú szintaktikus és elemzési követelményeket támaszt, ami biztosítja, hogy a szükséges elemzési algoritmus egyszerű, hatékony és ellentmondásmentes maradjon

Az XML-t gyakran használják dokumentumtárolási és feldolgozási formátumként, mind on-line, mind offline módon. Ez több előnnyel is jár:

- internetes szabványokon alapuló erőteljes, logikailag ellenőrizhető formátum
- a hierarchikus struktúrája megfelel a legtöbb (de nem mindegyik) dokumentum típusnak
- egyszerű szöveg formátumban valósul meg, licencektől és korlátozásoktól mentesen
- platform-független, így viszonylag ellenálló a technológiai változásokkal szemben

- az XML-t és elődjét, az SGML-t már több mint tíz éve használják, így széles tapasztalat és eszközkészlet áll rendelkezésre

2.3 Az XML hátrányai

Bizonyos alkalmazások szempontjából a következő hátrányokkal rendelkezik:

- A szintaxisa elég bőbeszédű és részben redundáns. Ez nehezítheti az emberi olvashatóságot és az alkalmazások hatékonyságát, valamint nagyobb tárolási költséggel jár. Nehézzé teszi az XML alkalmazását korlátozott sávszélesség esetén, bár bizonyos esetekben a tömörítés csökkentheti a problémát. Ez részben igaz a telefonokon és PDA-kon futó multimédiás alkalmazásokra, melyek XML-t szeretnének használni képek és videók leírására.
- A szintaxis számos homályos, felesleges tulajdonsággal bír, ami az SGML hagyatéka.
- Az első verziók, az alapvető elemzési követelmények nem támogatják az adattípusok túl széles körét, így néha a kívánt adat kinyerése a dokumentumból plusz munkával jár az elemző részéről. Például nincs lehetőség XML-ben a "3,14159" lebegőpontos számként való megjelölésére hét karakterből álló sztring helyett.
- Nem volt lehetőség a dokumentum egyes részeinek közvetlen elérésére és frissítésére.
- Egymást részben átfedő (nem hierarchikus) adatstruktúrák modellezése külön erőfeszítést igényel.
- Az XML relációs és objektum orientált paradigmához kötése néha fáradságos.

2.4 Tulajdonságai

Az XML dokumentumokra lehet jól formáltságot, és érvényességet is megállapítani, melyeknek jelentése:

- **Jól formázottság**. Egy jól formázott XML dokumentum megfelel minden XML szintaktikai szabálynak. Például ha egy nem üres elem rendelkezik nyitó tag-gel, de nem rendelkezik záró tag-gel, akkor nem jól formázott. Az a dokumentum, ami nem helyesen formázott, nem tekinthető XML-nek. Az elemzőnek meg kell tagadnia a feldolgozását.

  Egy jól formázott XML dokumentumra többek között a következő szabályok vonatkoznak rá: Egyetlen gyökér elem lehet egy dokumentumban. Azonban az XML deklaráció, feldolgozó utasítások és megjegyzések megelőzhetik a gyökér elemet.

Számos megszorításnak kell teljesülnie amit a specifikáció leír, például:

- egyetlen felső szintű elemet (gyökérelemet) kell hogy tartalmazzon. Azonban az XML deklaráció, feldolgozó utasítások és megjegyzések megelőzhetik ezt az elemet.

.Például
```
<?xml version="1.0" encoding="ISO-8859-2"?>,
 <!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0//EN"
            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- A nem üres elemeket mind nyitó, mind záró tag-eknek kell határolni.
- Az üres elemek megjelölhetők üres elem (önlezáró) tag-gel, mint például az `<Üres/>`. Ez megegyezik az `<Üres></Üres>` párossal.
- Az elemek címkéjében meg lehet adni kulcs-érték párokat ezeket attribútumnak nevezik
- Minden attribútum érték idézőjelek között van, vagy szimpla(') vagy dupla(") idézőjelek között.
- A elemek egymásba ágyazhatók, de nem lehetnek átfedők. Mindegyik nem gyökér elemet másik elemnek kell magában foglalnia.
- A dokumentum megfelel a benne leírt karakterkódolásnak. A karakterkódolás általában az XML deklarációban van meghatározva, de a szállító protokoll (pl. HTTP) is meghatározhatja. Ha nem adjuk meg az XML deklarációt, akkor alapértelmezés az UTF-8 és az UTF-16.
- Az XML megkülönbözteti a kis és nagybetűket.

  Jól formázott

  ```
  <ÉnPéldám> ... </ÉnPéldám>
  ```

  Nem jól formázott.

  ```
  <ÉnPéldám> ... </Énpéldám>
  ```

A specifikáció még sok további megszorítást is tartalmaz.

Érvényesség. A dokumentumban a megadott elemek, a megadott módon vannak megadva. (Az adattípusok nagyon korlátozottak. Az attribútumok kapcsán használhatók csak.)

Egy XML dokumentum, ami azon kívül, hogy jól formázott, még meg is felel egy adott sémának, az érvényesnek nevezhető.

Sémák definiálásával lehetővé válik az XML dokumentumok szerkezetére vonatkozó megszorítások megadása. Sok különböző formátumot definiáltak sémák formájában. A DTD egy ilyen sémanyelv. Ez a legelső nyelv, rengeteg problémája volt. Lásd később.

Az általánosított adatleíró nyelvek (mint például az SGML és az XML) megjelenése előtt a szoftvertervezőknek speciális fájlformátumokat kellett kifejleszteniük, hogy adatokat tudjanak megosztani több alkalmazás között. Ez részletes specifikációk megírását, valamint speciális célú elemzők megalkotását követelte meg. Az XML szabályos struktúrája és szigorú elemzési szabályrendszere képessé teszi a szoftvertervezőket, hogy az elemzést szabványos eszközökre bízzák, és mivel az XML egy általános, adatmodell-orientált keretrendszert biztosít az alkalmazás-specifikus nyelvek fejlesztőinek, a szoftverfejlesztőknek csak a szabályrendszer és az adat kifejlesztésére kell koncentrálni, így viszonylag magas absztrakciós szintet lehet elérni.
Alaposan tesztelt eszközök állnak rendelkezésre, hogy az XML dokumentumot a sémával szemben validálják: az eszköz automatikusan ellenőrzi, hogy a dokumentum megfelel-e a sémában kifejtett megkötéseknek. Vannak olyan eszközök, melyek az XML elemzőbe építve érhetők el, és vannak, melyek külön használhatók.
A sémák más felhasználása is létezik: az XML szerkesztők például a szerkesztés során fel tudják használni a sémákat a hatékonyság és a kényelem növelése érdekében. Ezalatt azt kell érteni, ha meg van adva a dokumentum sémája, akkor a szerkesztő tanácsokat tud adni a következő tag nevére, ill. a beírt adat típusát ellenőrizni tudja már beíráskor is.

A legrégebbi XML séma nyelv a dokumentum típus definíció (DTD, Document Type Definition), ami az SGML-ből származik. Mivel a DTD az XML 1.0 szabvány része, így mindenhol támogatott. Azonban megvannak a maga hátrányai:

- nem támogatja az XML újabb képességeit, például a névtereket
- az SGML-ből származtatott, nem XML szintaxist használ a sémák leírására

Az XML Schema egy újabb keletű XML séma nyelv, amit a W3C-n belül fejlesztettek ki.. Gyakran XSD (XML Schema Definition) néven is szokták emlegetni. Az XSD lényegesen többre képes a DTD-nél az XML nyelvek leírása terén. Sokoldalú adattípus rendszert használ, ami részletesebb megkötéseket tesz lehetővé az XML dokumentum logikai szintjén, de ezért sokkal robusztusabb érvényesítő keretrendszert követel meg. Ráadásul az XSD XML-alapú formátumon alapul, minek következtében szokványos XML eszközöket lehet használni a

létrehozásához és feldolgozásához, bár az implementációk sokkal többet kívánnak, mint az egyszerű XML olvasási képesség.

2.5 Kiegészítései

Az XML-nek léteznek kiterjesztései is, amik nagyon hasznosak lehetnek.

- XPath: Lehetővé teszi az egyes komponensekre való hivatkozást egy XML dokumentumon belül. Ez a képesség például lehetővé teszi XSL és XSLT stíluslapokon belül, hogy dinamikusan kiválasszuk a dokumentum egyes részeit olyan sorrendben, ahogy azt a kimenet megköveteli. (Elsődleges célja az XML dokumentumok alkotórészeinek megcímzése, de ezen felül tartalmaz még néhány alapműveletet is. Az XPath nem XML szintaxist használ. XML dokumentumokban, például attribútumokban használhatjuk. „`html/head/meta[@name='author']/@content`” Az előbbi XPath kifejezés megcímzi azt a 'content' attribútumot, amelynek szülője egy 'meta' nevű elem, amelynek van egy 'name' attribútuma is, és annak tartalma az, hogy 'author'.)
- XQuery: Ez egy XML lekérdezőnyelv.(Az XQuery funkcionális nyelv, amelyben minden lekérdezés egy kifejezés. Dokumentumok halmazán működik. XML adatbázisoknál lekérdező nyelvként is használt.)
- XML névterek: Lehetővé teszik, hogy egy dokumentum több szótárból tartalmazzon XML elemeket és attribútumokat névütközések nélkül.
- XML-Signature: Digitális aláírások létrehozására szolgáló szintaxist és feldolgozási szabályt definiál.
- XML Encryption: XML dokumentumok titkosítására szolgáló szintaxist és feldolgozási szabályt definiál.

Jelenleg két XML verzió létezik. Az XML 1.0-t 1998-ban definiálták. Azóta több kiadása is megjelent. Jelenleg a 2004. február 4-én kiadott harmadik kiadás aktuális. Széles körben elterjedt. Az XML 1.1-et egy napon adták ki az XML 1.0 harmadik kiadásával. Sok olyan tulajdonsággal bír, célja az XML használatának egyszerűbbé tétele több felhasználói csoport (főleg mainframe programozók) számára. Az XML 1.1 nem túl elterjedt, és csak azoknak ajánlják, akik ki tudják használni az egyedi képességeit.

3.Leíró nyelvek

Ebben a fejezetben néhány felhasználói felület leíró nyelv rövid ismertetését szeretném adni.

3.1 HTML

A **HTML** (angolul: HyperText Markup Language*)*. a történetileg a legelső még nem is XML alapú leíró leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki. Ez az SGML egy alkalmazása, a W3C(World Wide Web Consortium) támogatásával. Az aktuális változata a 4.01.

Az SGML egy metanyelv, amivel leíró nyelveket lehet definiálni. Az SGML az 1980-as évek közepe felé született és meglehetősen stabil maradt. Rendkívüli stabilitása onnan fakad, hogy a nyelv tulajdonság-gazdag és rugalmas. Ez a rugalmasság viszont bármennyire is értékes, nem ellensúlyozza bonyolultságát. Bonyolultsága meggátolta abban, hogy a World Wide Web -hez hasonló változatos körülmények között lehessen alkalmazni.

Az eredeti elgondolás szerint a HTML tudományos és más technikai dokumentumok cseréjének a nyelve, olyan felhasználóknak, akik nem kifejezetten dokumentum-specialisták. A HTML megoldotta az SGML bonyolultság-problémáját is, mivel a strukturáló és szemantikai elemeknek csak kis készletét használta, így le lehet írni vele tudományos dokumentumokat is. Ezenkívül, a dokumentumstruktúra egyszerűsítése céljából a HTML kiterjesztette a hypertext támogatást. A multimédiás képességek a későbbi fejlesztések eredményei. A HTML figyelemreméltóan rövid idő alatt széles körben népszerűvé vált és gyorsan túlhaladta az eredeti célt. A kezdetek óta gyorsan fejlődtek az új, HTML-ben használható elemek, egyre több volt köztük a speciális célra alkalmas. Az új elemeknek ez a bősége kompatibilitási problémákat okozott a dokumentumok platformok közötti adaptációja során.

3.2 XHTML

Az (Extensible HyperText Markup Language), vagy **XHTML.** az XML alkalmazásba újrafogalmazott HTML.

Az XHTML 1.0 (ez a specifikáció) az első dokumentumtípus az XHTML családban. Az XHTML a HTML 4 XML alapú továbbfejlesztése. Ez a dokumentumtípus azon dokumentumok nyelvéül szánták, amelyek olvashatóak XML-kompatibilis, és néhány egyszerű irányelv betartásával HTML 4 kompatibilis böngészővel is. Azok a fejlesztők, akik dokumentumaikat ezentúl XHTML 1.0 formátumban teszik közzé, a következő előnyöket élvezhetik:

- Az XHTML dokumentumok megfelelnek az XML előírásainak. Könnyedén megtekinthetők, szerkeszthetők és érvényesíthetők a standard XML eszközökkel.
- Az XHTML dokumentumok ugyanolyan jól szerkeszthetőek korábbi, HTML-4-et támogató felhasználói alkalmazásokkal, mint az új, XHTML 1.0-t támogató felhasználói programokkal.
- Az XHTML dokumentumok hasznosíthatják az aktív elemeket (scripteket és appleteket).
- Ahogy az XHTML család fejlődik, az XHTML 1.0 kritériumainak megfelelő dokumentumok egyre jobban együtt tudnak működni különböző XHTML környezetekben.

Mik az előnyei a sima HTML-el szemben?

A HTML értelmezése (parsing) kezdetben meglehetősen ad-hoc jelleggel történt, így a szabványba is jó néhány olyan elem csúszott be, ami a (nagyrészt azért meglevő) struktúraegységet megtörte. Néhány problémás példa, és az XHTML rájuk adott megoldásai:

- A HTML megengedi, hogy egy elemet nem kell lezárni, ez egy „pongyolaság”. A
 (sortörés) és a <hr> (vízszintes határoló) elemek magukban (lezáró elemek nélkül) álltak. XHTML-ben ezekből teljes értékű
 és <hr /> elemek lettek.
- Bizonytalan előfordulású elemek. A (bekezdés) elemről nem lehetett tudni, hogy két bekezdés között használandó-e, vagy minden egyes bekezdést egy nyitó-záró elempárral kell határolni. XHTML-ben a szöveg megoldás, tehát az utóbbi a helyes.
- Nemzetközi karakterek. Egy <meta> elemben van megadva a karakterkódolás. Az XHTML-ben maga az XML-bevezető (prolog) jelzi a továbbiak típusát, így az XML minden előnyét automatikusan megkapja.

Szükség volt egy szigorúbb nyelvre, amit bármilyen eszközre lehet küldeni, mert a HTML bonyolult szintaxisa miatt fellépő extra gépigényt így lehet csökkenteni, ami mobil eszközökön nagyon lényeges dolog. A dokumentum értelmezését is megkönnyítik az XML jól formázottsági követelményei. Az XHTML a névterek fogalmát is átvette az XML-től. Ez rugalmasságában hasonlatos a HTML-beli „class" attribútumhoz, de ebben sokkal több lehetőség van.
Az XHTML család a következő lépés az internet fejlődésében. Az XHTML-re való áttéréshez a tartalomfejlesztők beléphetnek az XML világába, élvezhetik az ezzel járó előnyöket és biztosak lehetnek abban, hogy közkinccsé tett oldalaik visszafelé és előrefelé is kompatibilisak maradnak.

Az elfogadása elég lassan folyik. Ennek egyik fő oka a Microsoft Internet Explorerje, amely még mindig nem támogatja az XHTML dokumentumokat, habár már van benne XML értelmező 1999 óta. Az alternatív böngészőkben sincs teljesen implementálva még. Mivel a böngésző készítők nem sietnek az elfogadásával, így böngésző támogatás nélkül úgy működnek, akár a sima HTML dokumentumok. Ennek ellenére már készülnek az XHML 2.0 változatának tervei, amik elég nagyra törőek, de az igények fejlődését figyelembe véve időszerűnek tűnnek.

3.3 XHTML 2.0

Az XHTML 2.0 a web ismert leírónyelveinek HTML 4.01, XHTML 1.0 és 1.1 rokona, de nem szándékozik felülről kompatibilis maradni azokkal.
Az XHTML 2.0 -tervben szereplő új tulajdonságok:

- HTML űrlapokat fel fogják váltani XForms technológiával.
- A jelenleg használt keretek helyett XFrames-t használ.
- Bármilyen elem hyperlink-ként működhet. Például `<li href="articles.html">Articles</li>`.
- Minden elem hivatkozhat média típusokra az „src" attribútumával pl. Lánchíd Ez ugyanaz mint <object src="Lanchid.jpg" type="image/jpeg"> Lánchíd </object>.
- Új lista elem típus (<nl>).

3.4 XUL

XML **U**ser Interface **L**anguage

A XUL egy XML alapú technológia szoftveralkalmazások grafikus felhasználói felületének létrehozására. Az alkalmazások széles körében használható, pl. web-böngészők, e-mail kliensek, határidőnaplók, kalkulátorok, adatbázis-kezelők, HTML szerkesztők. A nyílt forrású Mozilla platformban benne van a XUL teljes implementációja. Részletsebben lásd a 24. oldalon.

3.5 MXML

MXML (Macromedia Flex Markup Langauge)

A Flex (ez egy gazdag internetes alkalmazásfejlesztő keretrendszer, ami Flash-be fordítja az elkészült oldalt, így bármilyen platformon lehet használni) XML alapú leírónyelve, mely segítségével meghatározhatjuk, hogy hogyan nézzenek ki az alkalmazásaink, hol helyezkedjenek el az objektumok a képernyőn. Segítségével maximálisan kihasználható az egyes részek újrahasznosítása, mert általa erősen elkülöníthető a megjelenítés és az üzleti logika.

Az alkalmazásfejlesztők az MXML-t ActionScript-ekel kombinálva használják gazdag internetes alkalmazások fejlesztéséhez.

ActionScript: egy ECMAScript (más néven JavaScript) alapú, objektum orientált szkript nyelv, mely kiválóan alkalmas a Flex alkalmazások kliens oldali logikájának megírására. A fordítás során az MXML kódból is ActionScript kód lesz, így könnyedén megírhatjuk komponenseinket az utóbbiban is.

ActionScript-en keresztül tudunk kommunikálni a külvilággal is, webszolgáltatásokat vagy távoli objektumokat elérve.

A Flex nem ingyenes, de jól használható Flash-es oldalak megírásához.

Részletesebben a 29. oldalon írok róla.

3.6 Web Forms 2.0

A Web Forms olyan szabvány amely kiegészíti a HTML 4.01 űrlapjának lehetőségeit.

A web Forms 2.0 specifikációja, amit a WHATWG (Web Hípertext Application Technology) fejleszt, egy kiegészítés az űrlap tulajdonságokhoz, amik a HTML 4.01 űrlap fejezetében vannak leírva. A Web Forms 2.0 HTML és XHTML-be is alkalmazható. Fő újdonságai:

- erősen típusos adat-mezőket lehet vele definiálni
- új attribútumokat ad arra, hogy megszorításokat tudjunk definiálni,
- ismétlést leíró eszköz vezet be az űrlap részekben lévő ismétlődés leírásához,
- XML-be menthető, és onnan inicializálható űrlapok létrehozása.

Továbbá meglévő gyakorlatokat szabványosít, amik előzetesen eddig nem voltak dokumentálva. Továbbá világosabbá tesz néhány interakciót a HTML űrlap vezérlők és a CSS között.

A Web Forms 2.0 specifikációja hamarosan végső stádiumba fog érni, ami az implementációk megjelenésével kecsegtet.

3.7 OpenLaszlo

Az OpenLaszlo egy nyílt forrású környezet, amellyel olyan „installálásmentes" webalkalmazásokat lehet készíteni, amelyeknek a felhasználói felülete az asztali alkalmazások megjelenítési képességéhez mérhető. Az OpenLaszlo platformot eredetileg Laszlo Presentation Server (LPS) néven hívták, 2001-ben kezdődött el a fejlesztése a LaszloSystems cégnél. 2004-ben egy fontos lépés következett be: az addig kereskedelmi forgalomban elérhető platform nyílt forráskódúvá vált (GPL licenc alatt) és az OpenLaszlo nevet kapta. Ez a platform két részből épül fel: Az LZX-ből, ez egy XML és JavaScript-en alapuló leíró nyelv, hasonlatos a XUL, MXML, XAML nyelvekhez, és az OpenLaszlo szerverből. A szerver egy Java szervlet, ami az LZX kódot fordítja le a kiválasztott futtatási környezethez. Az OpenLaszlo 3.x-nél csak Flash lejátszót lehetett választani. Ez azt jelenti, hogy a szerver az LZX forrást lefordítja bináris SWF mozivá, ugyanúgy, mint a Flex, így bárhol futtathatóak az OpenLaszlo-val készült gazdag internetes alkalmazások, ahol van Flash Player. Az OpenLaszlo 4.x változatától már lehet DHTML be is fordítani is.

3.8 Glade

Glade egy RAD (Rapid Application Development, gyors alkalmazás fejlesztés) eszköz, ami lehetővé teszi, hogy gyorsan és könnyen lehessen fejleszteni felhasználói felületeket GTK+ (GIMP Toolkit) és Gnome asztali környezetre. GPL licenc alatt adták ki. Egy GTK+ felhasználói felület felépítésének leghatékonyabb módja a Glade grafikus felületszerkesztő használata. A Glade az elkészült felületből képes forráskódot generálni, de igazi különlegessége a Libglade könyvtár használatában rejlik, mely képes az alkalmazás futtatásakor felépíteni annak felületét a Glade által generált XML file alapján. Ez jelentősen hozzájárul a megjelenítés és az üzleti logika elkülönítéséhez és lehetővé teszi a programok felületének megváltoztatását a forráskód újrafordítása nélkül. A Libglade felhasználásával készített felületek két ponton kapcsolódnak az alkalmazás forráskódjához. Az egyik kapcsolat az egyes vezérlőelemek elnevezése. Az alkalmazás nevük alapján hivatkozhat a felhasználó felület elemeire. A felületet ezen felül az eseménykezelő függvények kötik össze a programmal. A Libglade programkönyvtár automatikusan hozzákapcsolja a program eseménykezelő függvényeit a felület megtervezésekor beállított eseményekhez. A Glade XML formátumban tárolja a vele elkészített felhasználói interfészeket, és a libglade könyvtár használatával tudják az alkalmazások dinamikusan betölteni azt, ami nekik belőle szükséges. Libglade alkalmazásával a Glade XML fájlokat használhatjuk rengeteg programozási nyelvhez. Például: C, C++, Java, Perl, Python, C#, Pike, Ruby. Támogatás hozzáadása más nyelvekhez is könnyen elkészíthető.

3.9 JAXX

A JAXX egy nyílt forrású XML alapú felhasználói felület készítő keretrendszer Javához. A következőképpen működik. XML fájlokban leírjuk a komponenseket és az interakcióikat, amelyeket a JAXX lefordít Java osztályokká. A felhasználói felület komponenseit sokkal gyorsabban és könnyebben lehet kifejleszteni JAXX -ban, mint a szokásos Java kódolással.

Főbb tulajdonságai:

- CSS stíluslap használati lehetőség, amivel az egész program kinézetét lehet változtatni egy CSS fájlból.
- Java alapú szkript és eseménykezelés, ami azt jelenti, hogy Java kód szinte bárhol elhelyezhető a JAXX fájlban.

- A JAXX fájlok sokkal rövidebbek és könnyebben olvashatóak, mint a velük egyenértékű Java kód.

A JAXX-ot arra találták ki, hogy helyettesítse a kézzel megírt Swing komponenseket. Több lehetőséget ad a felhasználói felületek elkészítéséhez. Elemeire egyszerűen lehet hivatkozni Java osztályokból. Ez azt jelenti, hogy különösebb gond nélkül hozzáadhatjuk bármely alkalmazáshoz. Tehát nem arra való, hogy webalkalmazások részeként ezeket a fájlokat a felhasználó gépére küldve ott helyben készítsük el a felületet, hanem hogy megkönnyítse a felületek megszerkesztését Javában.

3.10 XUI

A XUI egy XML alapú Javás nyílt forrású keretrendszer, mellyel asztali, mobil, és webes alkalmazásokat lehet fejleszteni. A XUI támogatja a Swing-et, az AWT-t és más kütyü (widget) készleteket. A XUI-vel könnyen fejleszthetők, kiterjeszthetők a komponensek, hozzáadhatunk vele új vagy mások által készített komponenseket a felülethez. Létezik egy Kaleidoscope nevű kiegészítés Netbeans IDE-ben, amely XUI alapú.

4. XForms: az új generációs web űrlapok

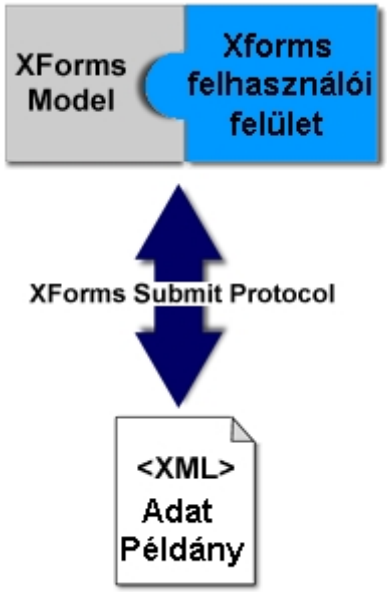
Az elektronikus kereskedelem és a webalkalmazások a magasabb fokú interakciót kínáló űrlapok iránt gerjesztik az igényt. Ezért hozták létre az XForms-ot, ami az űrlapok kényelmesebb, interaktívabb használatát ígéri.

Az XForms egy izgalmas új nyelv a W3C-től, amit sok mindenre lehet használni, egyszerű űrlapok készítésétől a komplex web 2.0 alkalmazásokig. Az XForms dinamikus, rendszerfüggetlen, szkript nyelv független és teljesen szabványos. Nem egy különálló dokumentumtípus, hanem integrálni kell más nyelvekbe, mint az XHTML vagy SVG. Sokat tanult a HTML-beli űrlapok jó és rossz tulajdonságából. Az XForms követhetőbbé és tisztábbá teszi azt, hogy milyen adatot is küldtünk el, és hová. Könnyen újra felhasználhatóvá teszi az űrlapokat, amik már nem szorosan kötődnek ahhoz az oldalhoz, amelyiken használják.

A klasszikus HTML web űrlapok nem különítik el az űrlap funkcióját a megjelenítéstől. Ezzel ellentétben az XForms elkülöníti az űrlap kezelését, a megjelenítését és az adatait. Így rugalmas megjelenítési beállításokat tesz lehetővé. Az XForms egyik fontos elve, hogy az űrlapja adatait az XForms adatait reprezentáló „instance” elemben lévő elemekben helyezi el. Ezeket az elemeket fogom adat-példányként nevezni. A többi funkció mellett az XForms modell leírja a struktúráját is az adat-példánynak.

Az XForms szerkezete:

Az XForms Submit Protocol egy adatcsatornát definiál, ami az adat-példány és az XForms feldolgozó között létesít kapcsolatot. Ez leírja, hogy hogyan küld és fogad adatot az XForms, beleértve, hogy hogyan kell a űrlapnak a felfüggesztését, folytatását és befejeződését kezelni.



1 kép: illusztráció az XForms fő elemeit és kapcsolódásait szemlélteti.

Az XForms előnyei :

- elegáns MVC (modell-nézet-vezérlő) kialakítás.
- deklaratív programozási nyelv, amit könnyű tanulni, és a nyelvvel készített programokat egyszerű karbantartani, belőni.
- kompatibilis XML szabványokkal, úgy mint CSS, XML Schema és XPath
- kiterjeszthetőség
- nemzetköziesítés: (mivel az XML-ről van szó, használható az Unikód.

Néhány hátrány:

- Beépített támogatás hiánya Microsoft Internet Explorelnél.
- Használatához le kell tölteni a böngészőbe egy kiegészítőt.
- JavaScript alapú megvalósításoknál (pl. FormFaces) az oldal megtekintéséhez egy kb. 85kb-os állományt szükséges letölteni.
- CSS szükséges ahhoz, hogy az űrlapok elfogadható kinézetűek legyenek.
- A CSS 3 számos tulajdonságát nem támogatja a Microsoft Internet Explorer.
- Kevés működő példaalkalmazás található.
- Kevés grafikus űrlapkészítést segítő alkalmazás van a piacon.

Az XForms még nem terjedt el olyan szinten, hogy a böngészőkben natívan implementálva lenne, de kipróbálásához könnyen találhatunk megfelelő eszközöket. Manapság az XForms-ot majdnem minden web-böngésző tudja futtatni valamilyen letölthető kiegészítő (plugin) használatával.

4.1 példák

Hogy megmutassam, milyen egyszerű áttérni XForms-ra, ha valaki HTML űrlapkészítésben jártas, néhány példát fogok leírni, és így bemutatni a lehetőségeit illetve azokat a dolgokat, amelyek mentén el lehet indulni bárkinek, aki meg akarja ismerni ezt a nyelvet.

A példák XHTML befogadónyelvűek.

Az adatok tárolásában, és az űrlapküldés tulajdonságainak elhelyezésében is különbség van az XForms és HTML űrlapok között. Például az előző kettő helye XForms-ban a „head” elemben található. Az űrlap megjelenítésével kapcsolatos elemek kerülnek csak a „body” elembe.

Most pedig következzen néhány példa az XForms és a HTML közötti különbségek szemléltetésre.

Egy egyszerű HTML űrlap így néz ki:

```
<html>
<head><title>Search</title></head>
<body>
   <form action="http://example.com/search"
         method="get">
        Find <input type="text" name="q">
        <input type="submit" value="Go">
   </form>
</body>
</html>
```

A megfelelő XForms űrlap:

`<html xmlns="http://www.w3.org/1999/xhtml"` `xmlns:xf="http://www.w3.org/2002/xforms">`	Szükséges egy névtérdeklaráció, hogy használni tudjuk az XForms lehetőségeit

`<head>`
`<title>Search</title>`
`<f:model>`	

`<xf:instance><data xmlns=""><q/></data></xf:instance>`	Az adatok az „istance” elembe kerülnek.

`<xf:submission action="http://example.com/search" method="get" id="s"/>`
`</xf:model>`
`</head>`
`<body>`
`<p>`	

`<xf:input ref="q"><xf:label>Find</xf:label></xf:input>`	Az „input” elem definiál egy szövegbeviteli mezőt.
`<xf:submit submission="s"><xf:label>Go</xf:label></xf:submit>`	A „submit” elem olyan gombot ír le amivel be lehet vinni az adatot.

`</p>`
`</body>`
`</html>`	

HTML-ben a „form” nevű elemben kell megadni az űrlapot. Az XForms-ban nincs ilyen megkötés.

HTML -ben	XForms-ban
Keresztnév: <input type="text" name="keresztnev">	<xf:input ref="keresztnev"><xf:label>Keresztnév:</xf:label></xf:input>

Szövegmező és megfelelője.

HTML -ben	XForms-ban
Message: <textarea name="message" rows="20" cols="80"></textarea>	<xf:textarea ref="message"><xf:label>Message:</xf:label></xf:textarea>
	textarea[ref="message"] { font-family: sans-serif; height: 20em; width: 80em }

Egyszerűbb a dolgunk, ha külön stíluslapra rakjuk a szövegformázást pl:

<?xml-stylesheet href="style.css" type="text/css"?>

A nyomógombot az XForms-ban a „trigger” elemmel tudunk készíteni.

HTML	XForms
<input type="button" value="Show" onclick="show()">	<xf:trigger><xf:label>Show</xf:label> <script ev:event="DOMActivate" type="text/javascript">show()</script> </xf:trigger>
	vagy <xf:trigger ev:event="DOMActivate" ev:handler="#show"> <xf:label>Show</xf:label> </xf:trigger>

Az XForms űrlap feldolgozása eseményekkel, eseménykezelőkkel és esemény válaszokkal van meghatározva .

A példában van egy ev:event="DOMActivate" tulajdonság, ami azt jelenti, hogy erre a DOMActivate eseményre fut le a deklarált eseménykezelő. A gomb megnyomására DOMActivate esemény jön létre, erre a deklarált eseménykezelő le fog futni. Jelen esetben egy JavaScript függvényhívás.

A következő kód azt mutatja be, hogyan lehet egy kattintással egy értéket megváltoztatni.

```
<xf:trigger id="editButtons">
      <xf:label>valtoztat</xf:label>
       <xf:action  ev:event="DOMActivate">
       <xf:setvalue ref="/data/q" value="'mas szöveg'"/>
       </xf:action>
</xf:trigger >
```

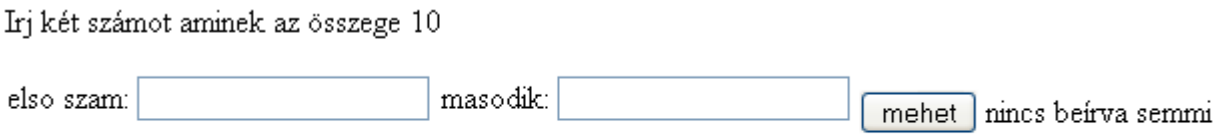
Lehet rövidebben is írni:

```
<xf:trigger id="editButtons">
      <xf:label>valtoztat</xf:label>
       <xf:setvalue ev:event="DOMActivate" ref="/data/q" value="'mas szöveg'"/>
</xf:trigger >
```

Az XForms XPath-ot használ arra, hogy:

- címezni lehessen adat-példányokat
- a adatkötő kifejezéseknél (pl. egy „output” nevü elemnél a „ref” attribútumban szereplő /data/q kifejezés, a q csomópontra hivatkozást jelenti, és ezáltal q értékét fogja kiírni),
- megszorítások kifejezéséhez
- számítások meghatározásához.

Ez nagyban megnöveli az XForms használhatóságát, amit most egy egyszerű példával illusztrálok. A példában az űrlapra két számot lehet írni, amelyekről eldönti, hogy az összegük 10-e vagy sem, és ennek az eredményét ki is írja. Ilyen műveletekhez az XForms-ban nem kell JavaScript függvényeket hívogatni.



2 kép, XForms alapú számítás.

Forrása:

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xf="http://www.w3.org/2002/xforms" xmlns:ev="http://www.w3.org/2001/xml-events" xml:lang="en"> <head> <title>összeg</title> <xf:model> <xf:instance><data xmlns=""> <elso/> <masod/> <elfogad>nincs beírva semmi</elfogad> </data> </xf:instance> </xf:model> </head> <body> Irj két számot aminek az összege 10 <xf:input ref="elso"><xf:label>elso szam:</xf:label> </xf:input> <xf:input ref="masod"><xf:label>masodik:</xf:label>	Adat-egyedek felsorolása és kezdő értékeik. Első input mező.

</xf:input> <xf:trigger id="belép"> <xf:label>mehet</xf:label> <xf:action ev:event="DOMActivate"> <xf:setvalue ref="/data/elfogad" value="if((/data/elso + /data/masod)= 10,'elfogad','nemfogad el')"/> </xf:action> </xf:trigger > <xf:output ref="/data/elfogad"/> </body> </html>	Második input mező. A „trigger” lenyomásával egy értékadás következik be az elfogad elemre nézve, aminek az értéke egy kifejezés. Ez dönti el, hogy a két beírt szám összege egyenlő-e tízzel. Kiírja az elfogad elem értékét.

A források tesztelését legegyszerűbben a FormFaces XForms implementációval lehet elvégezni, ebben JavaScript-tel implementálták az XForms eszközeit.

Hogy fusson a kód, csak annyival kell kibővíteni a HTML lapunkat a <html> elemben, hogy:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xf="http://www.w3.org/2002/xforms"
 xmlns:ev="http://www.w3.org/2001/xml-events" xml:lang="en" >
```

És a <head> elembe be kell a következő sorokat írni.

```
<link rel="stylesheet" type="text/css" href="xforms.css" />
   <script type="text/javascript" src="formfaces.js"></script>
```

Még sok más tulajdonsága is van az XForms-nak, ízelítőül csak ennyit soroltam fel. Aki a HTML, XHTML közegben dolgozik, az űrlapok kezelése szempontjából egy nagyon kényelmes kezelési módszert ismerhet meg általa. Az hogy nem különálló nyelv, hanem beépíthető más leíró nyelvekbe, nagy előnyt jelenthet az elterjedése szempontjából. Idő kell még arra, hogy az alternatív böngészők között támogatottságot nyerjen, de a vezető böngésző gyártójánál nem sok esély van a natív implementálásra. A XHTML 2.0-ban már alapból benne lesz, de sajnos az XHTML 1.0 elfogadtatása is akadozik, így még később fog bemutatkozni a nagyközönség előtt az XHTML 2.0 változatában, mint különálló kiegészítésként. Szerintem

az XForms egy egyszerű mód a HTML lapok funkcionalitásának megnövelésére, ha valaki nem akar Flash alapú oldalakkal bíbelődni, érdemes felfedeznie és használnia.

5. XUL

A XUL (ejtsd Zúúl) egy XML felhasználói felületleíró nyelv, amit egy Mozilla projekt keretében fejlesztették ki, hogy segítse a Mozilla böngésző fejlesztését. Minden XML nyelvi sajátosság elérhető XUL-ban is. A nyelvet speciálisan a portolható felhasználói interfészek készítésére készítettek. Sok időbe kerülne megírni egy alkalmazást még egy platformra is. Idő kell a fordításhoz és hosszas belövés szükséges. A XUL-lal egy interfészt gyorsan és könnyen lehet implementálni. A XUL-ban megvannak minden más XML nyelvek előnyei. Más formátumú XML kódrészeket is be lehet szúrni a kódba.

XUL -ban a legtöbb modern grafikus felület elemet meg lehet valósítani.

Néhány ezek közül:

- Adatbeviteli elemek, mint szövegdobozok és checkboxok.
- Eszköztárak gombokkal és más elemekkel.
- Menü a menüben és felugró menük.
- Füles párbeszédpanelek.
- Hierarchikus fák, táblázatos információk megjelenítése
- Billentyűparancsok

Több mód van rá, hogy XUL alkalmazást készítsünk.

1. Firefox kiegészítés: plusz funkcionalitást ad a böngészőhöz. A kiegészítéseket más Mozilla alapú termékekhez is hozzá lehet adni, úgymint például a Thunderbird.
2. Különálló XULRunner alkalmazás. XULRunner egy csomagolt verziója a Mozilla platformnak, mellyel készíthetünk különálló XUL alkalmazást. Böngésző nem kell a futtatásához, ez egy különálló futtatható fájlt jelent.
3. XUL csomag. Ez a felső kettő között van. Úgy készült, mint egy kiegészítés, de külön ablakban úgy viselkedik, mint egy különálló program. Ennek akkor van haszna, amikor nem akarjuk használni/beletenni a programba a nagy méretű teljes

XULRunner alkalmazást , de nem bánjuk, hogy kell felinstallált Mozilla böngésző a futtatáshoz.

4. Távoli XUL alkalmazás. Csak el kell helyezni a XUL kódot a webszerverre és megnyitni a böngészőben, úgy, mint egy weboldalt. Ezzel a módszerrel korlátozva vagyunk, ugyanis ennek biztonsági vonatkozásai is vannak, ami limitálja a meghívható funkciókat, úgymint új ablak nyitása.

Az első háromnál szükség van installálásra a felhasználó gépén.

A Mozilla ugyanúgy kezeli a HTML dokumentumokat, mint a XUL dokumentumokat. Sok tulajdonság közös kettőjüknél. Például egy CSS stíluslap ugyanúgy használható egy XML dokumentumnál, mint egy HTML oldalnál. Persze van néhány specifikus sajátosság, mint ahogyan HTML specifikusak az űrlapok, XML specifikusak a menük. Mivel a XUL és a HTML egy módon van kezelve, mindkettőt be lehet tölteni helyi fájlrendszerről, weboldalról, kiegészítésből vagy különálló XULRunner alkalmazásból.

XUL fájlok hivatkozhatnak URL-ekre, ill. többfajta protokollú URL-re, éppen úgy mint a HTML fájlok. A Mozillában létezik az URL-nek egy speciális fajtája, a chrome URL. A Mozilla chrome rendszere csomagokat tárol. A XUL csomagokban fájlok és szkriptek vannak, amik a felhasználói felület működését definiálják. A csomag lehet könyvtárban vagy JAR fájlban is. A chrome rendszerbe telepített csomagot egy speciális URL-lel, a chrome URL-lel tudjuk hivatkozni. Egyes csomagok már benne vannak a Mozillában, de mi is regisztrálhatjuk a sajátunkat. Az az előnye az installált csomagoknak, hogy nem kell olyan biztonsági korlátozásoktól tartani, melyek egyébként felmerülnének sok alkalmazásnál. Másik előnye másfajta URL típusokhoz képest, hogy automatikusan tud összetett témákat és környezeteket kezelni. (Például hivatkozunk chrome URL el egy fájlra, ami egy témának egyik képe. Úgy hivatkozhatunk rá, hogy nem kell tudni, hogy a felhasználó melyik témát használja. A Mozilla tudja hol található a hivatkozott adat.) Nem kell törődni azzal, hogy hol van a csomag installálva, hogy elérjük. A chrome URL független attól, hogy hol vannak a fájlok fizikailag. Ez azt eredményezi, hogy sokkal könnyebb alkalmazásokat írni, mivel nem kell törődni a fájlok elhelyezkedésének részleteivel.

A szintaxisa a következő:

chrome://<csomagnév/<rész(part)>/<file.xul>

példa: chrome://messenger/content/messenger.xul

A példa „messenger” nevű ablakra utal. Amikor megnyitunk egy chrome URL-t, a Mozilla végignézi az telepített csomagok listáját, megkeresi a JAR fájl, vagy könyvtár helyét, ami egyezik a csomagnévvel és résszel. A hozzárendelés egy manifest fájlban van leírva, ami a chrome könyvtárban van tárolva. Ha átrakjuk a „messenger.jar” fájlt valahová máshová, de a manifest fájlban ennek megfelelően frissítjük, hogy hova raktuk, akkor a rá hivatkozó programban nem lesz probléma, az ugyanis nem egy statikus helyre hivatkozik. Előnye például, ha a felhasználó megváltoztatja a program „bőrét” (skin), a XUL és szkript fájlokat nem kell megváltoztatni, csak a chrome URL-t kell átírni az aktuális „bőrre” (skinre).

5.1 példák

Most nézzünk meg, hogy lehet a nyelvet használni! Ezt bemutatandóan egy ablak elkészítését írom be. Ezt a kódot egy olyan állományba mentem el, amelynek a neve xul-ra végződik, így a Mozilla megfelelően fogja kezelni.

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window
    id="egyszeruAblak"
    title="Ablak"
    orient="horizontal"
    xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
 <!-- ide jön a többi dolog -->
</window>
```

Egy XUL dokumentumban olyan elemeket használunk, amelyek neve a XUL névtérbe tartozik, amit az alábbi URI azonosít:
http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul

Például egy gombot így deklarálhatunk .

```
<button  id="identifier" class="dialog"  label="OK"
image="images/image.jpg" disabled="false"   accesskey="t"/>
```

Ez egy „OK” feliratú gombot ír le, amelyen egy kép található, és egy gyorsbillentyűn keresztül is aktiválható. Természetesen ez a gomb egy „window” elemben van.

Doboz modellel írja le az elemek elhelyezését a XUL. Ezzel a modellel teljesen fel lehet osztani az ablakot téglalap alakú részekre. A dobozban lévő elemeket függőlegesen és vízszintesen helyezhetjük el. Dobozok, térkitöltők, és flex attribútumú elemek megadásával lehet az elhelyezkedést szabályozni egy ablakban.

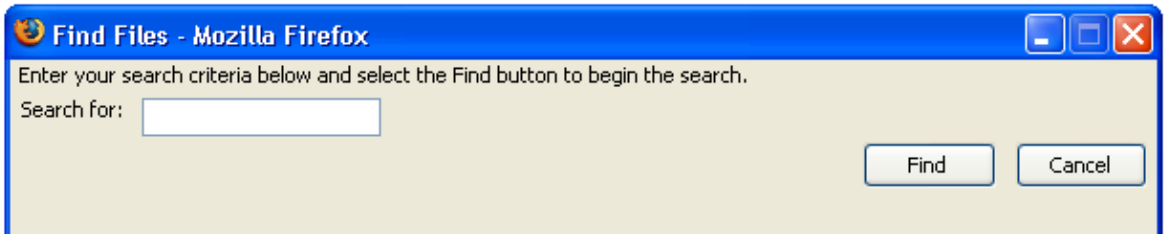
Kétfajta „box” elem létezik, a vízszintes és a függőleges.

`<hbox>`
`...`
`</hbox>`
	`<vbox>`
	`...`
	`</vbox>`

Egy példával mutatom meg a fent leírtakat:

```
<vbox flex="1">

  <description>
    Enter your search criteria below and select the Find button to begin
    the search.
  </description>

  <hbox>
    <label value="Search for:" control="find-text"/>
    <textbox id="find-text"/>
  </hbox>

  <hbox>
    <spacer flex="1"/>

    <button id="find-button" label="Find"/>
    <button id="cancel-button" label="Cancel"/>
  </hbox>
</vbox>
```

Itt a „vbox”-ba és a „spacer” nevű elembe is belekerült a „flex=1” attribútum, ami a flexibilitást jelöli. Jelen esetben hogy a függőleges téglalap „kimehet” az ablak végéig, és a „spacer” elemmel megadjuk, hogy a két gomb ne a szövegdoboz alatt legyen, hanem jobb oldalt.



3 kép XUL-ban elemek dinamikus elhelyezése.

Ennyi tudással már az eseménykezelést is megnézhetjük. Szkripthívásra nagyon egyszerű eszközök vannak, pl:

```
<vbox oncommand="alert(event.target.tagName);">
  <button label="OK"/>
  <checkbox label="Show images"/>
</vbox>
```

Itt egy gomb vagy jelölőnégyzet megnyomásakor egy „alert” üzenetet kapunk, olyan szöveggel, amilyen felhasználói felület elemet aktiváltunk (gombra kattintás esetén „Button” szöveget kapunk). A parancs esemény vagy a gombban, vagy a jelölőnégyzetben keletkezik és „felmegy” a „vbox” elem szintjére, és ott már egy művelet van rákötve. Az esemény „felmegy” a „windows” elemig is. Ha valamilyen más dolgot is kell kezelni vele, akkor az eseményt kiváltó elemet tartalmazók is lereagálhatják.

A következő forráskód programja meghatározza, hogy hol kattintott a felületen a felhasználó.

```
<script>

function updateMouseCoordinates(event)
{
  var text = "X:" + event.clientX + " Y:" + event.clientY;
  document.getElementById("xy").value = text;
}
</script>

<label id="xy"/>
<hbox width="400" height="400"
      onmousemove="updateMouseCoordinates(event);"/>
```

Az „event” objektumban megtalálhatók az egér koordinátái. Láthatjuk, hogy a JavaScript hívások a XUL-ba nagyon kényelmesen beépíthetők.

A XUL jövője elég kecsegtető, főleg úgy, hogy egy nagy cég, a Mozilla Fundation áll mögötte. A XULRunner képességével önálló alkalmazásban is szerepelhet, de ez is a hátránya, hiszen csak ilyen alapú alkalmazások használhatják. A natív implementációját Microsoft IE alatt kizártnak tartom, így nem fog a böngészők általános felület leíró nyelvévé alakulni. Ha böngészéssel, böngészővel kapcsolatos alkalmazást kell fejleszteni, és nem fontos hogy mindegyiken fusson, akkor én ezt az utat javaslom.

6. A Flex

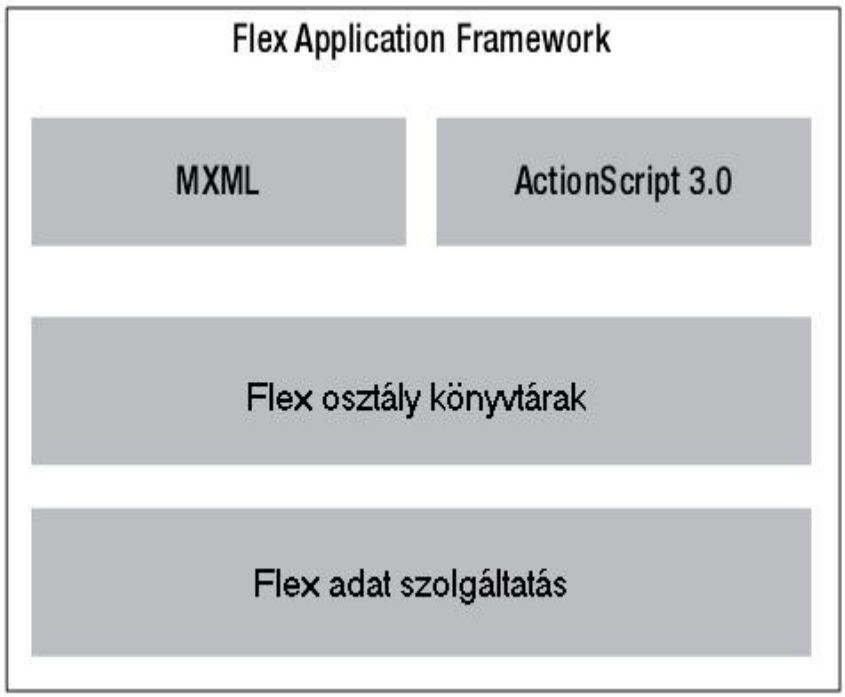
A Flex-et 2004-ben a Macromedia bocsátotta ki azzal a céllal, hogy segítségével gazdag internet alkalmazásokat lehessen fejleszteni Flash-ben. Ugyanis a klasszikus alkalmazásfejlesztő programozók nagy kihívásnak találták az időegyenesekben, jelenetekben való gondolkodást, amire eredetileg a Flash platform épült, a Flex azonban elfeledteti velünk a Flash hagyományos használatakor alkalmazott időegyenesek, jelenetek, alakzatok fogalmát, s ehelyett vezérlőkben, eseményekben gondolkodhatunk.
Az MXML (Macromedia Flex Markup Langauge) a Flex XML alapú leírónyelve, mely segítségével meghatározhatjuk, hogy hogyan nézzenek ki az alkalmazásaink, hol helyezkedjenek el az objektumok a képernyőn. Segítségével maximálisan kihasználható az egyes részek újrahasznosítása, általa erősen elkülöníthető a megjelenítés és az üzleti logika.

Weblapszerkesztés és a Flex
A legtöbb vállalati szintű web tervezéshez három rétegű (szintű) struktúra szükséges:

- A prezentációs réteg. Ebben lehet elkészíteni, amit a felhasználó látni fog.
- Üzleti logika réteg. Itt a „színfalak" mögött történnek ténylegesen a dolgok, itt van lekezelve a kapcsolat más szerverekkel, adatbázisokkal. És itt van megvalósítva az algoritmus része az alkalmazásnak.

- Gerinc „backbone" réteg. Ezen a szinten található az adatbázis szerver, ami gyűjti és kiszolgáltatja az adatokat az Üzleti logika réteg szabályai szerint.

Én most csak a prezentációs rétegre fogok koncentrálni, hiszen itt lehet felhasználói felületet létrehozni, szerkeszteni, és itt kell használni az MXML -t is. De egy rövid kitérőt teszek még az alkalmazási keretrendszer felépítésére vonatkozóan.



4 kép Flex alkalmazási keretrendszer felépítése

A tetején látható az MXML és Actionscript 3.0. Ezek meglehetősen fontosak, ugyanis ezekkel tudja a Flex a két legnépszerűbb eszközt, az XHTML-t , és az objektum orientált programozást használni. Ugyanis az MXML hasonlít az XHTML-re, az Actionscriptel pedig az objektumorientáltságot lehet megvalósítani. Ha még ez sem lenne elég weboldalunk kinézetének leírásához, mindkét előbb felsorolt eszköz tudja hasznosítani a Flash lehetőségeit, amivel szinte bármilyen kinézet kialakítható. A Flexnek osztálykönyvtárakban gazdag, előre elkészített kollekciókészlete van, amit fel tudunk használni Flash lejátszó segítségével a felhasználói interakciók kapcsán. A Flex adatszolgáltatás megkönnyíti és kezeli kapcsolatokat, az adatforrásokat stb.

Az adatszolgáltatás egy szerver technológia, segít integrálni a kiszámított adatokat az alkalmazásban. Segíteni tudja az adatokkal való felhasználói munkát, még ha a felhasználó offline módban dolgozna is.

Az XHTML-hez és XML-hez-hasonlóan, az MXML is struktúrát biztosít az alkalmazásunknak. A Flex ad még egy kis segítséget, MXML-ben behívhatunk előre elkészített komponenseket, vagy újat is összerakhatunk vele a semmiből. Az egész alkalmazás kinézetét „bőrözni” (skinezni) is lehet úgy, mintha csak CSS-t használnánk.

Akármilyen jó is az MXML, mint minden leírónyelvnek, néhány megkötése, megszorítása van. Régebben a HTML interaktívabbá tételére a JavaScript-et választották. Erre analóg

módon az ActionScript való a Flex-ben az MXML erejének fokozására. Fontos, mert a komponensek közötti bonyolultabb interakciókra szükség van az ActionScript-re.

6.1 példák

Néhány példán át próbálom megmutatni a Flex és azon belül az MXML lehetőségeit.

Egy panel létrehozása:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">

<mx:Panel x="108.5" y="105" width="250" height="200" layout="absolute" title="Testing Events">
        <mx:VBox x="79" y="55" height="100%">
        <mx:Label id="myLabel"/>
                <mx:Button label="Test" id="myButton" height="22"
                click="myLabel.text=' A gombot megnyomták'"/>
        </mx:VBox>


</mx:Panel>
</mx:Application>
```

Ezzel a kóddal lehet kirakni egy panelt, amin egy gomb van, ha a gombot megnyomjuk, akkor kiírja „a gombot megnyomták” üzenetet.
Hogy milyen egyszerű MXML-ben az erőforrások használata, azt egy egyszerű programmal szemléltetem. A következő program megnyit egy web-blogot és egy táblázatba rakja a bejegyzések nevét és időpontját. A tartalmát is megnézhetjük, ha kijelöljük a kívánt bejegyzést. Az egész kód ennyi:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
                        creationComplete="feedRequest.send()">
<mx:HTTPService
    id="feedRequest"
    url="http://weblogs.macromedia.com/mchotin/index.xml"
    useProxy="false"/>
    <mx:Panel x="10" y="10" width="475" height="400" layout="absolute"

title="{feedRequest.lastResult.rss.channel.title}">
        <mx:DataGrid id="dgPosts" x="20" y="20" width="400"

      dataProvider="{feedRequest.lastResult.rss.channel.item}">
            <mx:columns>
                 <mx:DataGridColumn headerText="Posts" dataField="title"/>
                <mx:DataGridColumn headerText="Date" dataField="pubDate"
```

```
      width="150"/>
            </mx:columns>
        </mx:DataGrid>

        <mx:TextArea x="20" y="175" width="400"

      htmlText="{dgPosts.selectedItem.description}" />
        <mx:LinkButton x="20" y="225" label="Read Full Post"
click="navigateToURL(new
      URLRequest(dgPosts.selectedItem.link));"/>
    </mx:Panel>
</mx:Application>
```

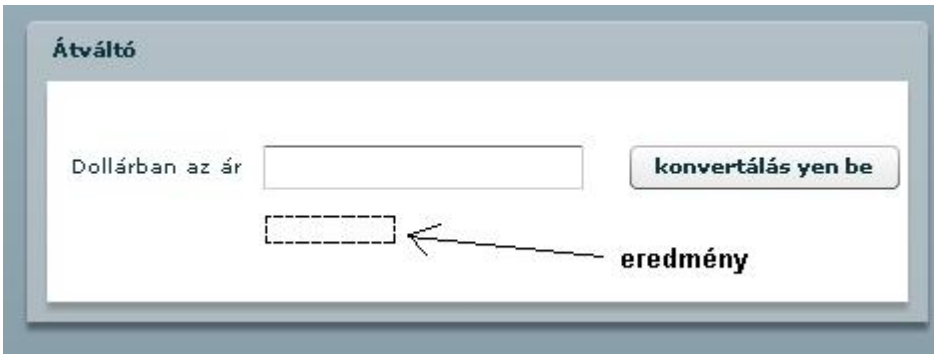
5. kép Flex példa kép.

A blog címe egy HTTPService elemben van eltárolva. A panel címe pedig a megnyitandó csatorna címe lesz. A DataGrid-ban a dataProvider-el adjuk meg a forrást, vagyis hogy mit jelenítsen meg. Most éppen a címét, és az időpontját jeleníti meg. A textArea-ban a kiválasztott cím leírása fog megjelenni. A link-gomb (Read Full Post) pedig egy új oldal betöltésével lehozza a kiválasztott elemet tartalmazó weblapot.

Az eseménykezelést az ActionScript látja el. Ezt egy egyszerű valutaátváltó alkalmazással szemléltetem. Kezdeteknek egy panelt készítettem a megfelelő kezelőkkel.

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">
      <mx:Panel x="20" y="20" width="450" height="150" layout="absolute"
title="Átváltó">
            <mx:Label x="10" y="34" text="Dollárban az ár"/>
            <mx:TextInput x="108" y="32" id="txtinput"/>
            <mx:Button x="291" y="32" label="konvertálás yen be"
id="btnConvert"/>
            <mx:Label x="108" y="73" id="lbresults"/>
      </mx:Panel>
</mx:Application>
```

Így néz ki:



6. kép Flex Dollárátváltó alkalmazás kép.

És most kezdjük el használni az ActionScript nyelvet, úgy hogy egy szkriptet rakunk a kódba, ami átváltja a a szövegdobozban megadott számot Yenre. Ehhez szkript elemet helyezek el a kódba az „application” elembe.

```
<mx:Script>
      <![CDATA[
            public function convertCurrency():void {
    var rate:Number = 120;
    var price:Number = Number(txtinput.text);
    if (isNaN(price)) {
        lblResults.text = "Kérem valós adatot adjon meg.";
    } else {
        price = price * rate;
        lblResults.text = "Yenben az összeg: " + String(price);
    }
}
      ]]>
</mx:Script>
```

Majd a gombra „rákötöm” a függvény meghívását.

```
<mx:Button x="291" y="32" label="konvertálás yen be" id="btnConvert"

      click="convertCurrency()"/>
```

„Bőröket” (kinézetet) is egyszerűen lehet hozzáadni a komponensekhez. Itt például Flash alapú „bőröket” adtam a gomboknak. Külön „bőrt” adtam meg a gombnak azokra az esetekre amikor még nem aktiválódott, amikor az egér rajta áll, és amikor már megnyomták.

```
<mx:Style>
     Button {
        upSkin: Embed(source='SubmitButtonSkins.swf', symbol='MyUpSkin');
        overSkin: Embed(source='SubmitButtonSkins.swf',
symbol='MyOverSkin');
        downSkin: Embed(source='SubmitButtonSkins.swf',
symbol='MyDownSkin');
     }
  </mx:Style>
```

A példákat az Adobe Flex builder 2 programmal készítettem. Ez egy Eclipse™ alapú fejlesztői környezet. A program egyik előnye, hogy lehet „rádobálásos” módszerrel felületeket készíteni, amiből a program MXML fájlt készít. Ezt természetesen lehet közvetlenül is szerkeszteni.

A programokon keresztül érzékelhettük, mintha nem is Flash technológiával kerültünk volna szembe, hanem a klasszikus HTML JavaScript weblapszerkesztéssel. Természetesen különbözik tőle, akárcsak a felület kinézeti lehetőségeinek száma. Ez az összehasonlítás természetesen a Flex javára billenne. Főleg, ha hozzávesszük, hogy beilleszthetünk előre lefordított Flash-es komponenseket, „bőröket”, háttereket.

Mivel nagy cég áll a fejlesztése mögött (Adobe), számíthatunk rá a jövőben is. Abban is biztosak lehetünk, hogy tovább fog fejlődni. Sem a builder, sem maga a Flex technológia nem ingyenes, de nem üzleti célra, magánszemélyként használhatjuk.

7. Felület leíró nyelvek gyakorlatban való összehasonlítása

Mindhárom felületleíró nyelven írtam egy programot, ami a Sudoku játékot valósítja meg. Ezzel a játékkal és megvalósításaival próbálok egyfajta kis tesztet készíteni a felület leíró nyelvek között, illetve néhány szempont szerint összehasonlítottam őket.

7.1 A Sudoku játék szabályai

A Sudoku rejtvény üres négyzeteibe kell beírni a hiányzó számokat úgy, hogy minden sorban, minden oszlopban és minden 3x3-as blokkban egyszer és csakis egyszer szerepeljen minden szám 1-től 9-ig. A rejtvénynek csak egyetlen egy helyes megoldása van.

Egy Sudoku rejtvény így néz ki:

		4	8				1	
			9	6		3		
3			4		2			
1	5	8					7	
	7			8				
		2			9	1	8	
	6							3
5			3		1		4	
				9		8		

7. kép Sudoku rejtvény példa

Egy Sudoku rejtvény megoldása (nem a fenti megoldása):

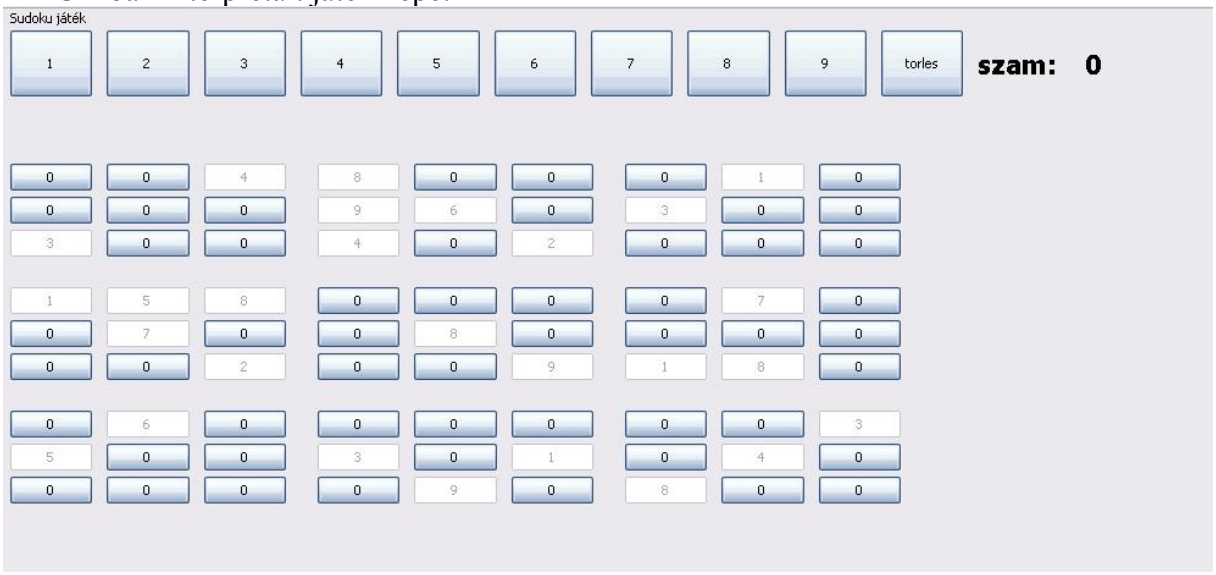
5	4	1	9	3	8	2	6	7
7	8	9	6	4	2	5	1	3
6	2	3	1	7	5	4	8	9
8	3	6	2	9	4	1	7	5
2	9	7	5	6	1	3	4	8
4	1	5	3	8	7	6	9	2
1	5	8	4	2	9	7	3	6
9	6	2	7	1	3	8	5	4
3	7	4	8	5	6	9	2	1

8. kép Sudoku rejtvény egy megoldása

A programjaim mindig az első rejtvény adatait tartalmazzák.

7.2 A XUL alapú megvalósítás

A XUL-ban interpretált játék képe:



9. kép Sudoku XUL-ban képernyőkép.

Sudoku XUL forrása a következő

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window
    id="findfile-window"
```

```
    title="Sudoku"
    orient="horizontal"
    onload="init();"
    xmlns:h="http://www.w3.org/1999/xhtml"
    xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">


                    <script>

 //A függvény megváltoztatja a hívó gomb számát,
function update(x,y,a)
{


  //var text = "X:" + event.clientX + " Y:" + event.clientY;
  var num=document.getElementById('theNUM').value;

  if(num==0) {document.getElementById(x+y).label = num;return;}

  if(negyed(x,y)=='true')/*alert('igaz')*/;
  else{/*alert('hamis'); */ return;}

  for(var j=1;j&lt;=9;j++){
        if(y==j) continue;
if(document.getElementById(x+j).label == num)
        return;
        }
        for(var i=1;i&lt;=9;i++){
        if(x==i) continue;
if(document.getElementById(i+y).label == num)
        return;
        }

  document.getElementById(x+y).label = num;
   //alert('hello2');
}
function setnum(num)
{
      document.getElementById('theNUM').value = num;


}
function init() //inicializálja az oldalt feltölti a gombok szövegrészét a
//megfelelő számokkal
{

       var tomb=
[[0,0,4,8,0,0,0,1,0],[0,0,0,9,6,0,3,0,0],[3,0,0,4,0,2,0,0,0],

      [1,5,8,0,0,0,0,7,0],[0,7,0,0,8,0,0,0,0],[0,0,2,0,0,9,1,8,0],

      [0,6,0,0,0,0,0,0,3],[5,0,0,3,0,1,0,4,0],[0,0,0,0,9,0,8,0,0]];


      for(var i=1;i&lt;=9;i++)
         for(var j=1;j&lt;=9;j++)   {
         document.getElementById(i+''+j).label = tomb[i-1][j-1];
```

```
         if(tomb[i-1][j-1]!=0)
document.getElementById(i+''+j).disabled='true'; //letiltom azokat a
gombokat amelyeknek az értékét nem szabad módosítani.
   }



}

function negyed(x,y) // boolean értéket ad vissza arról, hogy a berakandó
szám az saját 3x3 mas blokkjában elhelyezhető-e
{
      var num=document.getElementById('theNUM').value;

      if(x &lt;=3){
            if(y &lt;=3){ // 1 1 kilenced
                  for(var i=1;i&lt;=3;i++)
                        for(var j=1;j&lt;=3;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }
            if(y &lt;=6){ // 1 2 kilenced
                  for(var i=1;i&lt;=3;i++)
                        for(var j=4;j&lt;=6;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }
            if(y &lt;=9){ // 1 3 kilenced az if trivialis
                  for(var i=1;i&lt;=3;i++)
                        for(var j=7;j&lt;=9;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }

      }
      if(x &lt;=6){
            if(y &lt;=3){ // 2 1 kilenced
                  for(var i=4;i&lt;=6;i++)
                        for(var j=1;j&lt;=3;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }
            if(y &lt;=6){ // 2 2 kilenced
                  for(var i=4;i&lt;=6;i++)
                        for(var j=4;j&lt;=6;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }
            if(y &lt;=9){ // 2 3 kilenced az if trivialis
```

```
                  for(var i=4;i&lt;=6;i++)
                        for(var j=7;j&lt;=9;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }

      }
      if(x &lt;=9){ //if nem kellene
            if(y &lt;=3){ // 1 1 kilenced
                  for(var i=7;i&lt;=9;i++)
                        for(var j=1;j&lt;=3;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }
            if(y &lt;=6){ // 1 2 kilenced
                  for(var i=7;i&lt;=9;i++)
                        for(var j=4;j&lt;=6;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }
            if(y &lt;=9){ // 1 3 kilenced az if trivialis
                  for(var i=7;i&lt;=9;i++)
                        for(var j=7;j&lt;=9;j++)
                              if(document.getElementById(i+''+j).label ==
num)
                                    return 'false';
            return 'true';
            }

      }


      return 'false';
}

</script>

<vbox>
<label value="Sudoku játék "/>
      <hbox><!--gombokkal a beírandó számot lehet kiválasztani  -->
            <button label="1" oncommand="setnum('1');" />
            <button label="2" oncommand="setnum('2');"/>
            <button label="3" oncommand="setnum('3');"/>

            <button label="4" oncommand="setnum('4');"/>
            <button label="5" oncommand="setnum('5');"/>
            <button label="6" oncommand="setnum('6');"/>

            <button label="7" oncommand="setnum('7');"/>
            <button label="8" oncommand="setnum('8');"/>
            <button label="9" oncommand="setnum('9');"/>
            <button label="torles" oncommand="setnum('0');"/>
            <h:h1>
```

```
      <label value="szam: "/>
      <label id="theNUM" value="0"/>  <!-- a berakandó szám értéke-->
      </h:h1>
</hbox>
<spacer  height="50"/>

<hbox><!--a 9x9 elemű gombtábla elkészítése -->
      <button id="11" label="OK" oncommand="update('1','1');" />
      <button id="12" label="OK" oncommand="update('1','2');" />
      <button id="13" label="OK" oncommand="update('1','3');"/>
      <label  value=" "/>
      <button id="14" label="OK" oncommand="update('1','4');"/>
      <button id="15" label="OK" oncommand="update('1','5');"/>
      <button id="16" label="OK" oncommand="update('1','6');"/>
      <label  value=" "/>
      <button id="17" label="OK" oncommand="update('1','7');"/>

      <button id="18" label="Ok" oncommand="update('1','8');" />
      <button id="19" label="OK" oncommand="update('1','9');" />

</hbox>

<hbox>
      <button id="21" label="Ok" oncommand="update('2','1');" />
      <button id="22" label="Ok" oncommand="update('2','2');" />
      <button id="23" label="Ok" oncommand="update('2','3');" />
      <label  value=" "/>
      <button id="24" label="Ok" oncommand="update('2','4');" />
      <button id="25" label="Ok" oncommand="update('2','5');" />
      <button id="26" label="Ok" oncommand="update('2','6');" />
      <label  value=" "/>
      <button id="27" label="Ok" oncommand="update('2','7');" />
      <button id="28" label="Ok" oncommand="update('2','8');" />
      <button id="29" label="Ok" oncommand="update('2','9');" />
</hbox>

<hbox>
      <button id="31" label="Ok" oncommand="update('3','1');" />
      <button id="32" label="Ok" oncommand="update('3','2');" />
      <button id="33" label="Ok" oncommand="update('3','3');" />
      <label  value=" "/>
      <button id="34" label="Ok" oncommand="update('3','4');" />
      <button id="35" label="Ok" oncommand="update('3','5');" />
      <button id="36" label="Ok" oncommand="update('3','6');" />
      <label  value=" "/>
      <button id="37" label="Ok" oncommand="update('3','7');" />
      <button id="38" label="Ok" oncommand="update('3','8');" />
      <button id="39" label="Ok" oncommand="update('3','9');" />
</hbox>

<spacer height="20"/>
<hbox>
      <button id="41" label="Ok" oncommand="update('4','1');" />
      <button id="42" label="Ok" oncommand="update('4','2');" />
      <button id="43" label="Ok" oncommand="update('4','3');" />
      <label  value=" "/>
      <button id="44" label="Ok" oncommand="update('4','4');" />
      <button id="45" label="Ok" oncommand="update('4','5');" />
      <button id="46" label="Ok" oncommand="update('4','6');" />
```

```
      <label  value=" "/>
      <button id="47" label="Ok" oncommand="update('4','7');" />
      <button id="48" label="Ok" oncommand="update('4','8');" />
      <button id="49" label="Ok" oncommand="update('4','9');" />
</hbox>

<hbox>
      <button id="51" label="Ok" oncommand="update('5','1');" />
      <button id="52" label="Ok" oncommand="update('5','2');" />
      <button id="53" label="Ok" oncommand="update('5','3');" />
      <label  value=" "/>
      <button id="54" label="Ok" oncommand="update('5','4');" />
      <button id="55" label="Ok" oncommand="update('5','5');" />
      <button id="56" label="Ok" oncommand="update('5','6');" />
      <label  value=" "/>
      <button id="57" label="Ok" oncommand="update('5','7');" />
      <button id="58" label="Ok" oncommand="update('5','8');" />
      <button id="59" label="Ok" oncommand="update('5','9');" />
</hbox>

<hbox>
      <button id="61" label="Ok" oncommand="update('6','1');" />
      <button id="62" label="Ok" oncommand="update('6','2');" />
      <button id="63" label="Ok" oncommand="update('6','3');" />
      <label  value=" "/>
      <button id="64" label="Ok" oncommand="update('6','4');" />
      <button id="65" label="Ok" oncommand="update('6','5');" />
      <button id="66" label="Ok" oncommand="update('6','6');" />
      <label  value=" "/>
      <button id="67" label="Ok" oncommand="update('6','7');" />
      <button id="68" label="Ok" oncommand="update('6','8');" />
      <button id="69" label="Ok" oncommand="update('6','9');" />
</hbox>

<spacer height="20"/>

<hbox>
      <button id="71" label="Ok" oncommand="update('7','1');" />
      <button id="72" label="Ok" oncommand="update('7','2');" />
      <button id="73" label="Ok" oncommand="update('7','3');" />
      <label  value=" "/>
      <button id="74" label="Ok" oncommand="update('7','4');" />
      <button id="75" label="Ok" oncommand="update('7','5');" />
      <button id="76" label="Ok" oncommand="update('7','6');" />
      <label  value=" "/>
      <button id="77" label="Ok" oncommand="update('7','7');" />
      <button id="78" label="Ok" oncommand="update('7','8');" />
      <button id="79" label="Ok" oncommand="update('7','9');" />
</hbox>

<hbox>
      <button id="81" label="Ok" oncommand="update('8','1');" />
      <button id="82" label="Ok" oncommand="update('8','2');" />
      <button id="83" label="Ok" oncommand="update('8','3');" />
      <label  value=" "/>
      <button id="84" label="Ok" oncommand="update('8','4');" />
      <button id="85" label="Ok" oncommand="update('8','5');" />
      <button id="86" label="Ok" oncommand="update('8','6');" />
      <label  value=" "/>
```

```
            <button id="87" label="Ok" oncommand="update('8','7');" />
            <button id="88" label="Ok" oncommand="update('8','8');" />
            <button id="89" label="Ok" oncommand="update('8','9');" />
      </hbox>

      <hbox>
            <button id="91" label="Ok" oncommand="update('9','1');" />
            <button id="92" label="Ok" oncommand="update('9','2');" />
            <button id="93" label="Ok" oncommand="update('9','3');" />
            <label  value=" "/>
            <button id="94" label="Ok" oncommand="update('9','4');" />
            <button id="95" label="Ok" oncommand="update('9','5');" />
            <button id="96" label="Ok" oncommand="update('9','6');" />
            <label  value=" "/>
            <button id="97" label="Ok" oncommand="update('9','7');" />
            <button id="98" label="Ok" oncommand="update('9','8');" />
            <button id="99" label="Ok" oncommand="update('9','9');" />
      </hbox>

</vbox>

 </window>
```

XUL-ban éreztem a legkönnyebbnek a program megírását. Egy szövegszerkesztővel dolgoztam, és sűrűn nézegettem a xultutorial-t (a XUL-hoz kiadott referencia és példaalkalmazás gyűjtemény). A XUL esetén a JavaScript-el nem voltak problémák, benne minden használható volt, ellenben egyes JavaScript alapú XForms implementációkkal. Letiltottam azokat a gombok amelyeknek a megfelelő számjegye rögzített. Mint látható, csak JavaScript-et használtam, nem mentem bele a XUL más nyelvekkel való kapcsolatának megvalósításaiba. A JavaScript a feladathoz tökéletes volt.

7.3 A XUL alapú megvalósítás értékelése

könnyen kezelhetőség: könnyen kezelhető/ átlátható kisebb projekteknél

gyorsaság: gyors

erőforrásigény: kevés

készítési idő: viszonylag kevés

tőlem mekkora időt vont el: keveset

kereszt platformitás: közepes indoklás: csak Mozillában vagy XULRunner alkalmazásával.

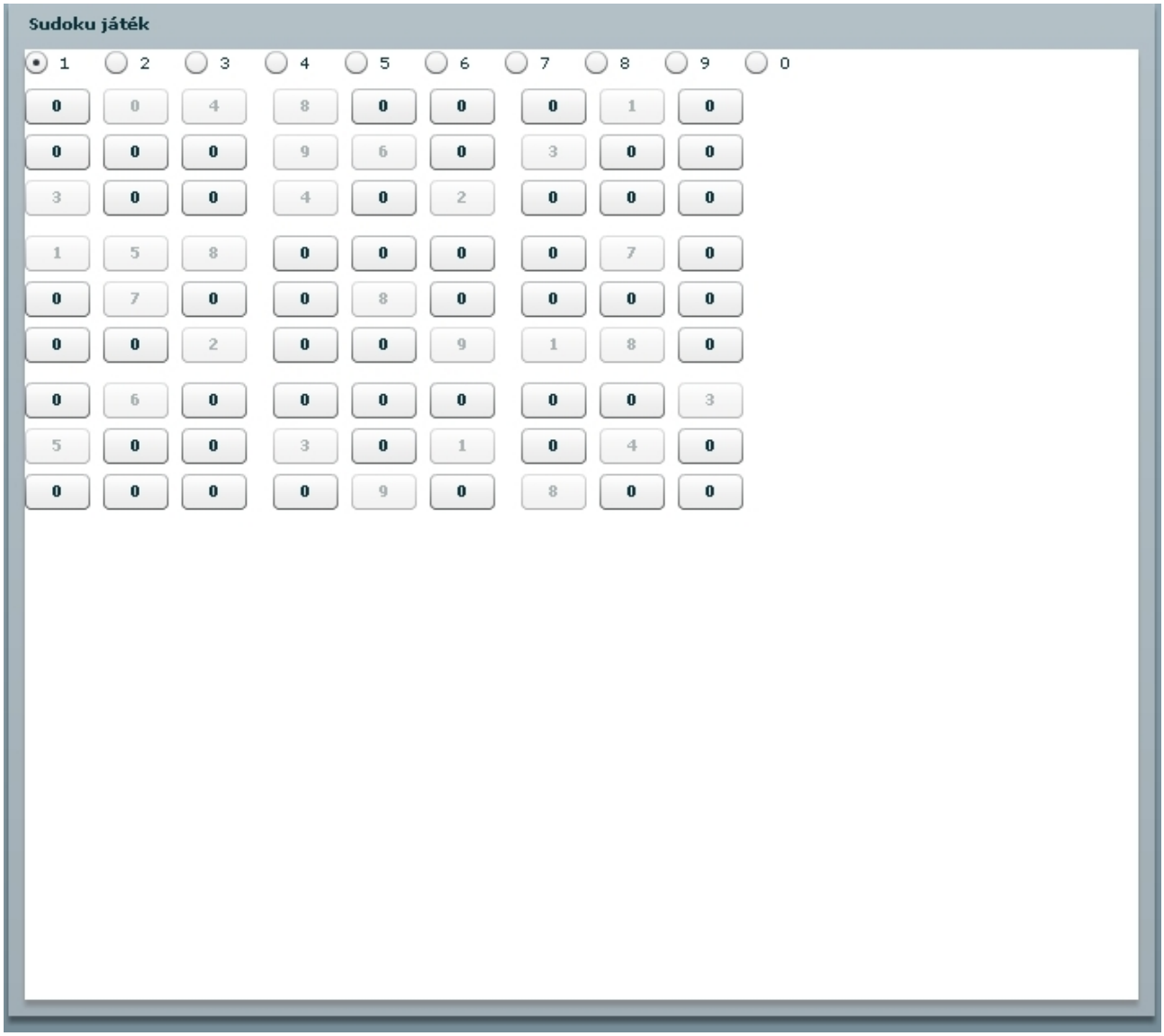
kinézet: átlagos

pénzfüggés: ingyenes

installálás: Benne van a Mozillában.

7.4 A Flex alapú megvalósítás

A Flex-ben interpretált játék képe:



10. kép Sudoku Flex-ben képernyőkép.

Flexben írt játék forrása

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
applicationComplete="pageInit()">
```

```
<mx:Script>   <![CDATA[
      import mx.controls.DataGrid;
      import mx.containers.HBox;
      import mx.controls.Alert;       //ActionScript statements
    var tomb:Array=
[[0,0,4,8,0,0,0,1,0],[0,0,0,9,6,0,3,0,0],[3,0,0,4,0,2,0,0,0],

      [1,5,8,0,0,0,0,7,0],[0,7,0,0,8,0,0,0,0],[0,0,2,0,0,9,1,8,0],

      [0,6,0,0,0,0,0,0,3],[5,0,0,3,0,1,0,4,0],[0,0,0,0,9,0,8,0,0]];

public function update(x:int,y:int,d:Event):void { //A függvény
megváltoztatja a hívó gomb számát,

// és a tömbbeli elem értékét, ha erre a szabályok szerint lehetőség van
    var rate:Number = 120;

    var value=   radiogroup1.selectedValue.toString();
   // myLabel.text="mas";

      if(value=='0'){tomb[x][y]=0; Button(  d.target).label  = '0';
return;}
      for(var i:int=0;i<9;i++){
            if(i==y) continue;
               if(tomb[x][i]==value )
                  {    /*Alert.show("sormiatt");      */return;}
      }
      for(var j:int=0;j<9;j++){
            if(j==x) continue;
               if(tomb[j][y]==value)
               {/*Alert.show("oszlop");*/ return;}
      }
      if(negyed(x,y));
       else{/*Alert.show("negyed miatt");*/ return;}


       tomb[x][y]=  value;
      Button(  d.target).label  =    value;
```

```
}

function negyed(x:int,y:int):Boolean // eldönti hogy az adott 3x3 mas
blokkba be lehet-e szúrni a kiválasztott elemet
{
      var num=radiogroup1.selectedValue.toString();

      if(x <3){
            if(y <3){ // 1 1 kilenced
                  for(var i=0;i<=2;i++)
                        for(var j=0;j<=2;j++)
                              if(tomb[i][j] == num)
                                    return false;
            return true;
            }
            if(y <6){ // 1 2 kilenced
                  for(var i=0;i<=2;i++)
                        for(var j=3;j<=5;j++)
                              if(tomb[i][j] == num)
                                    return false;
            return true;
            }
            if(y <9){ // 1 3 kilenced az if trivialis
                  for(var i=0;i<=2;i++)
                        for( j=6;j<=8;j++)
                              if(tomb[i][j] == num)
                                    return false;
            return true;
            }

      }
      if(x <6){
            if(y <3){ // 2 1 kilenced
                  for(var i=3;i<=5;i++)
                        for(var j=0;j<=2;j++)
                              if(tomb[i][j] == num)
                                    return false;
            return true;
            }
```

```
	if(y <6){ // 2 2 kilenced
		for(var i=3;i<=5;i++)
			for(var j=3;j<=5;j++)
				if(tomb[i][j] == num)
					return false;
	return true;
	}
	if(y <9){ // 2 3 kilenced az if trivialis
		for(var i=3;i<=5;i++)
			for( j=6;j<=8;j++)
				if(tomb[i][j] == num)
					return false;
	return true;
	}

}
if(x <9){ //if nem kellene
	if(y <3){ // 1 1 kilenced
		for(var i=6;i<=8;i++)
			for(var j=0;j<=2;j++)
				if(tomb[i][j] == num)
					return false;
	return true;
	}
	if(y <6){ // 1 2 kilenced
		for(var i=6;i<=8;i++)
			for(var j=3;j<=5;j++)
				if(tomb[i][j] == num)
					return false;
	return true;
	}
	if(y <9){ // 1 3 kilenced az if trivialis
		for(var i=6;i<=8;i++)
			for( j=6;j<=8;j++)
				if(tomb[i][j] == num)
					return false;
	return true;
	}
```

```
      }


      return false;
}
// amikor az oldal elkészül akkor hívódik meg
public function pageInit():void{  //ez egy kis trükközés. Így érem el a
gomb elemeket
                                                  //és letiltom azokat a
gombokat amelyeknek az értéke már meg van adva
      var ar:Array=     valami.getChildren();
  for(var i:int=0;i < ar.length;i++){
   if(   ar[i] instanceof HBox){
        var hboxx:Array =HBox(ar[i]).getChildren();

        for(var j:int=0;j <hboxx.length ;j++)
           if(   hboxx[j] instanceof Button){
            if( Button(hboxx[j]).label!='0')
                   Button(hboxx[j]).enabled=false;
           }
   }


  }

}

 ]]> </mx:Script>

      <mx:Panel title="Sudoku játék" width="702" height="622" x="5" >


            <mx:HBox >



            <mx:RadioButtonGroup  id="radiogroup1"/>
            <mx:RadioButton label="1" groupName="radiogroup1"
selected="true"/>
```

```
            <mx:RadioButton label="2" groupName="radiogroup1"/>
            <mx:RadioButton label="3" groupName="radiogroup1"/>
            <mx:RadioButton label="4" groupName="radiogroup1"/>
            <mx:RadioButton label="5" groupName="radiogroup1"/>
            <mx:RadioButton label="6" groupName="radiogroup1"/>
            <mx:RadioButton label="7" groupName="radiogroup1"/>
            <mx:RadioButton label="8" groupName="radiogroup1"/>
            <mx:RadioButton label="9" groupName="radiogroup1"/>
            <mx:RadioButton label="0" groupName="radiogroup1"/>
                        </mx:HBox>
            <mx:VBox id="valami" height="100%" width="675">
                  <mx:HBox width="100%">
                  <!-- igy nem szabad mert igy statikus az adatkotes-->
                        <mx:Button id="df" label="{tomb[0][0]}"
click="update(0,0,event )"/>
                        <mx:Button label="{tomb[0][1]}"
click="update(0,1,event)" enabled="false"/>
                        <mx:Button label="{tomb[0][2]}"
click="update(0,2,event)"/>
                        <mx:Spacer/>
                        <mx:Button label="{tomb[0][3]}"
click="update(0,3,event)"/>
                        <mx:Button label="{tomb[0][4]}"
click="update(0,4,event)"/>
                        <mx:Button label="{tomb[0][5]}"
click="update(0,5,event)"/>
            <mx:Spacer/>
                        <mx:Button label="{tomb[0][6]}"
click="update(0,6,event)"/>
                        <mx:Button label="{tomb[0][7]}"
click="update(0,7,event)"/>
                        <mx:Button label="{tomb[0][8]}"
click="update(0,8,event)"/>
                  </mx:HBox>
                  <mx:HBox width="100%">
                        <mx:Button label="{tomb[1][0]}"
click="update(1,0,event)"/>
                        <mx:Button label="{tomb[1][1]}"
click="update(1,1,event)"/>
```

```
                        <mx:Button label="{tomb[1][2]}"
click="update(1,2,event)"/>
                        <mx:Spacer/>
                        <mx:Button label="{tomb[1][3]}"
click="update(1,3,event)"/>
                        <mx:Button label="{tomb[1][4]}"
click="update(1,4,event)"/>
                        <mx:Button label="{tomb[1][5]}"
click="update(1,5,event)"/>
                  <mx:Spacer/>
                        <mx:Button label="{tomb[1][6]}"
click="update(1,6,event)"/>
                        <mx:Button label="{tomb[1][7]}"
click="update(1,7,event)"/>
                        <mx:Button label="{tomb[1][8]}"
click="update(1,8,event)"/>
                  </mx:HBox>
                  <mx:HBox width="100%">
                        <mx:Button label="{tomb[2][0]}"
click="update(2,0,event)"/>
                        <mx:Button label="{tomb[2][1]}"
click="update(2,1,event)"/>
                        <mx:Button label="{tomb[2][2]}"
click="update(2,2,event)"/>
                        <mx:Spacer/>
                        <mx:Button label="{tomb[2][3]}"
click="update(2,3,event)"/>
                        <mx:Button label="{tomb[2][4]}"
click="update(2,4,event)"/>
                        <mx:Button label="{tomb[2][5]}"
click="update(2,5,event)"/>
                  <mx:Spacer/>
                        <mx:Button label="{tomb[2][6]}"
click="update(2,6,event)"/>
                        <mx:Button label="{tomb[2][7]}"
click="update(2,7,event)"/>
                        <mx:Button label="{tomb[2][8]}"
click="update(2,8,event)"/>
                  </mx:HBox>
```

```
                <mx:Spacer/>
                <mx:HBox width="100%">
                      <mx:Button label="{tomb[3][0]}"
click="update(3,0,event)"/>
                      <mx:Button label="{tomb[3][1]}"
click="update(3,1,event)"/>
                      <mx:Button label="{tomb[3][2]}"
click="update(3,2,event)"/>
                      <mx:Spacer/>
                      <mx:Button label="{tomb[3][3]}"
click="update(3,3,event)"/>
                      <mx:Button label="{tomb[3][4]}"
click="update(3,4,event)"/>
                      <mx:Button label="{tomb[3][5]}"
click="update(3,5,event)"/>
                <mx:Spacer/>
                      <mx:Button label="{tomb[3][6]}"
click="update(3,6,event)"/>
                      <mx:Button label="{tomb[3][7]}"
click="update(3,7,event)"/>
                      <mx:Button label="{tomb[3][8]}"
click="update(3,8,event)"/>
                </mx:HBox>
                <mx:HBox width="100%">
                      <mx:Button label="{tomb[4][0]}"
click="update(4,0,event)"/>
                      <mx:Button label="{tomb[4][1]}"
click="update(4,1,event)"/>
                      <mx:Button label="{tomb[4][2]}"
click="update(4,2,event)"/>
                      <mx:Spacer/>
                      <mx:Button label="{tomb[4][3]}"
click="update(4,3,event)"/>
                      <mx:Button label="{tomb[4][4]}"
click="update(4,4,event)"/>
                      <mx:Button label="{tomb[4][5]}"
click="update(4,5,event)"/>
                <mx:Spacer/>
```

```
                        <mx:Button label="{tomb[4][6]}"
click="update(4,6,event)"/>
                        <mx:Button label="{tomb[4][7]}"
click="update(4,7,event)"/>
                        <mx:Button label="{tomb[4][8]}"
click="update(4,8,event)"/>
                  </mx:HBox>
                  <mx:HBox width="100%">
                        <mx:Button label="{tomb[5][0]}"
click="update(5,0,event)"/>
                        <mx:Button label="{tomb[5][1]}"
click="update(5,1,event)"/>
                        <mx:Button label="{tomb[5][2]}"
click="update(5,2,event)"/>
                        <mx:Spacer/>
                        <mx:Button label="{tomb[5][3]}"
click="update(5,3,event)"/>
                        <mx:Button label="{tomb[5][4]}"
click="update(5,4,event)"/>
                        <mx:Button label="{tomb[5][5]}"
click="update(5,5,event)"/>
                  <mx:Spacer/>
                        <mx:Button label="{tomb[5][6]}"
click="update(5,6,event)"/>
                        <mx:Button label="{tomb[5][7]}"
click="update(5,7,event)"/>
                        <mx:Button label="{tomb[5][8]}"
click="update(5,8,event)"/>
                  </mx:HBox>

                  <mx:Spacer/>
                  <mx:HBox width="100%">
                        <mx:Button label="{tomb[6][0]}"
click="update(6,0,event)"/>
                        <mx:Button label="{tomb[6][1]}"
click="update(6,1,event)"/>
                        <mx:Button label="{tomb[6][2]}"
click="update(6,2,event)"/>
                        <mx:Spacer/>
```

```
                        <mx:Button label="{tomb[6][3]}"
click="update(6,3,event)"/>
                        <mx:Button label="{tomb[6][4]}"
click="update(6,4,event)"/>
                        <mx:Button label="{tomb[6][5]}"
click="update(6,5,event)"/>
                  <mx:Spacer/>
                        <mx:Button label="{tomb[6][6]}"
click="update(6,6,event)"/>
                        <mx:Button label="{tomb[6][7]}"
click="update(6,7,event)"/>
                        <mx:Button label="{tomb[6][8]}"
click="update(6,8,event)"/>
                  </mx:HBox>
                  <mx:HBox width="100%">
                        <mx:Button label="{tomb[7][0]}"
click="update(7,0,event)"/>
                        <mx:Button label="{tomb[7][1]}"
click="update(7,1,event)"/>
                        <mx:Button label="{tomb[7][2]}"
click="update(7,2,event)"/>
                        <mx:Spacer/>
                        <mx:Button label="{tomb[7][3]}"
click="update(7,3,event)"/>
                        <mx:Button label="{tomb[7][4]}"
click="update(7,4,event)"/>
                        <mx:Button label="{tomb[7][5]}"
click="update(7,5,event)"/>
                  <mx:Spacer/>
                        <mx:Button label="{tomb[7][6]}"
click="update(7,6,event)"/>
                        <mx:Button label="{tomb[7][7]}"
click="update(7,7,event)"/>
                        <mx:Button label="{tomb[7][8]}"
click="update(7,8,event)"/>
                  </mx:HBox>
                  <mx:HBox width="100%">
                        <mx:Button label="{tomb[8][0]}"
click="update(8,0,event)"/>
```

```
                        <mx:Button label="{tomb[8][1]}"
click="update(8,1,event)"/>
                        <mx:Button label="{tomb[8][2]}"
click="update(8,2,event)"/>
                        <mx:Spacer/>
                        <mx:Button label="{tomb[8][3]}"
click="update(8,3,event)"/>
                        <mx:Button label="{tomb[8][4]}"
click="update(8,4,event)"/>
                        <mx:Button label="{tomb[8][5]}"
click="update(8,5,event)"/>
                  <mx:Spacer/>
                        <mx:Button label="{tomb[8][6]}"
click="update(8,6,event)"/>
                        <mx:Button label="{tomb[8][7]}"
click="update(8,7,event)"/>
                        <mx:Button label="{tomb[8][8]}"
click="update(8,8,event)"/>
                  </mx:HBox>




            </mx:VBox>
    </mx:Panel>

</mx:Application>
```

Flex-ben az elkészítés sokkal egyszerűbb volt mint a másik kettőben, mivel a szerkesztőjében az elemeket „rádobálással” is lehetett a felületre beépíteni. Nagyon hasznos volt az automatikus kódkiegészítés funkciója is, természetesen ez már alapértelmezett egy komoly fejlesztői környezetben. Nem volt gyorsabb az elkészítése, ugyanis a fordítás rengeteg időt vesz el a fejlesztés során egy szerényebb teljesítményű gépen. Kellemes volt az ActionScript használata, alig kellett a referenciát olvasni, mivel a metódusnevek beszédesek. Könnyen elértem minden elemet, hiszen az ActionScript-ben is meg lehet ugyanazt tenni, mint MXML-ben. A játék adatait egy tömbben tároltam az egyszerű kezelés miatt.

7.5 A Flex alapú megvalósítás értékelése

könnyen kezelhetőség: könnyen kezelhető/ átlátható kisebb projekteknél

gyorsaság: viszonylag lassú

erőforrásigény: sok

készítési idő: közepes, a fordítási idő miatt

tőlem mekkora időt vont el: átlagosat

kereszt platformitás: jó indoklás: Flash-ben van írva.

kinézet: jó

pénzfüggés: van

installálás: Flash lejátszó felrakásával

7.6 Az XForms alapú megvalósítás

Az XForms-ban interpretált játék képe:

Sudoku

Melyik számot választja? 1 2 3 4 5 6 7 8 9 torles

A választott szám:
1

beszúrhat-e: lehet
Milyen módban van:(teszt) beszur

0	0	4	8	0	0	0	1	0
0	0	0	9	6	0	3	0	0
3	0	0	4	0	2	0	0	0
1	5	8	0	0	0	0	7	0
0	7	0	0	8	0	0	0	0
0	0	2	0	0	9	1	8	0
0	6	0	0	0	0	0	0	3
5	0	0	3	0	1	0	4	0
0	0	0	0	9	0	8	0	0

11. kép Sudoku XForms-ban képernyőkép.

XForms-ban írt játék forrása

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!DOCTYPE
  html PUBLIC "-//W3C//DTD XHTML 1.0//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">


<html                                    xmlns="http://www.w3.org/1999/xhtml"
xmlns:xf="http://www.w3.org/2002/xforms"
xmlns:ev="http://www.w3.org/2001/xml-events" xml:lang="en" >

<head>


    <title>Sudoku</title>



    <xf:model id="amodel">

       <xf:instance><data xmlns="">
       <nem>0</nem> <!-- a kiválasztott elemet tárolja-->
        <loaded>false</loaded>
        <tabla title="Mr" merete="9">

            <sor id="s0">
                 <elem name="0" ertek="0" />
                 </sor>

                 <sor id="s1">
                 <elem name="1"  ertek="0"></elem>
                 <elem name="2"  ertek="0"></elem>
                 <elem name="3"  ertek="4"></elem>
                 <elem name="4"  ertek="8"></elem>
                 <elem name="5"  ertek="0"></elem>
                 <elem name="6"  ertek="0"></elem>
                 <elem name="7"  ertek="0"></elem>
                 <elem name="8"  ertek="1"></elem>
```

```
<elem name="9"  ertek="0"></elem>
</sor>

<sor id="s2">
<elem name="1"  ertek="0"></elem>
<elem name="2"  ertek="0"></elem>
<elem name="3"  ertek="0"></elem>
<elem name="4"  ertek="9"></elem>
<elem name="5"  ertek="6"></elem>
<elem name="6"  ertek="0"></elem>
<elem name="7"  ertek="3"></elem>
<elem name="8"  ertek="0"></elem>
<elem name="9"  ertek="0"></elem>
</sor>

<sor id="s3">
<elem name="1"  ertek="3"></elem>
<elem name="2"  ertek="0"></elem>
<elem name="3"  ertek="0"></elem>
<elem name="4"  ertek="4"></elem>
<elem name="5"  ertek="0"></elem>
<elem name="6"  ertek="2"></elem>
<elem name="7"  ertek="0"></elem>
<elem name="8"  ertek="0"></elem>
<elem name="9"  ertek="0"></elem>
</sor>
<sor id="s4">
<elem name="1"  ertek="1"></elem>
<elem name="2"  ertek="5"></elem>
<elem name="3"  ertek="8"></elem>
<elem name="4"  ertek="0"></elem>
<elem name="5"  ertek="0"></elem>
<elem name="6"  ertek="0"></elem>
<elem name="7"  ertek="0"></elem>
<elem name="8"  ertek="7"></elem>
<elem name="9"  ertek="0"></elem>
</sor>
<sor id="s5">
<elem name="1"  ertek="0"></elem>
```

```
<elem name="2"  ertek="7"></elem>
<elem name="3"  ertek="0"></elem>
<elem name="4"  ertek="0"></elem>
<elem name="5"  ertek="8"></elem>
<elem name="6"  ertek="0"></elem>
<elem name="7"  ertek="0"></elem>
<elem name="8"  ertek="0"></elem>
<elem name="9"  ertek="0"></elem>
</sor>
<sor id="s6">
<elem name="1"  ertek="0"></elem>
<elem name="2"  ertek="0"></elem>
<elem name="3"  ertek="2"></elem>
<elem name="4"  ertek="0"></elem>
<elem name="5"  ertek="0"></elem>
<elem name="6"  ertek="9"></elem>
<elem name="7"  ertek="1"></elem>
<elem name="8"  ertek="8"></elem>
<elem name="9"  ertek="0"></elem>
</sor>
<sor id="s7">
<elem name="1"  ertek="0"></elem>
<elem name="2"  ertek="6"></elem>
<elem name="3"  ertek="0"></elem>
<elem name="4"  ertek="0"></elem>
<elem name="5"  ertek="0"></elem>
<elem name="6"  ertek="0"></elem>
<elem name="7"  ertek="0"></elem>
<elem name="8"  ertek="0"></elem>
<elem name="9"  ertek="3"></elem>
</sor>
<sor id="s8">
<elem name="1"  ertek="5"></elem>
<elem name="2"  ertek="0"></elem>
<elem name="3"  ertek="0"></elem>
<elem name="4"  ertek="3"></elem>
<elem name="5"  ertek="0"></elem>
<elem name="6"  ertek="1"></elem>
<elem name="7"  ertek="0"></elem>
```

```
                <elem name="8"  ertek="4"></elem>
                <elem name="9"  ertek="0"></elem>
                </sor>
                <sor id="s9">
            <elem name="1"  ertek="0"></elem>
                <elem name="2"  ertek="0"></elem>
                <elem name="3"  ertek="0"></elem>
                <elem name="4"  ertek="0"></elem>
                <elem name="5"  ertek="9"></elem>
                <elem name="6"  ertek="0"></elem>
                <elem name="7"  ertek="8"></elem>
                <elem name="8"  ertek="0"></elem>
                <elem name="9"  ertek="0"></elem>
                </sor>

             <GivenName id="uzenet" uz="rendb">Rene</GivenName>
             <SurName>Smith</SurName>
          </tabla>
          <beszurandoert>2</beszurandoert>
          <kordx>s1</kordx>
          <kordy>2</kordy>
              <testbeszur>teszt</testbeszur> <!-- ez jelzi hogy teszt vagy
beszúr zemmdba vagyunk -->
            <atestbeszur>beszur</atestbeszur> <!-- ez jelzi hogy teszt vagy
beszúr zemmdba vagyunk -->
          <status>meg nem jelolt ki semit</status>



      </data></xf:instance>

      <xf:bind nodeset="/data/status" calculate="if(
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='1']/@ertek  =
/data/beszurandoert)  or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='2']/@ertek  =
/data/beszurandoert) or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='3']/@ertek  =
/data/beszurandoert) or
```

```
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='4']/@ertek  =
/data/beszurandoert) or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='5']/@ertek  =
/data/beszurandoert) or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='6']/@ertek  =
/data/beszurandoert) or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='7']/@ertek  =
/data/beszurandoert) or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='8']/@ertek  =
/data/beszurandoert) or
                (/data/tabla/sor[@id=/data/kordx]/elem[@name='9']/@ertek  =
/data/beszurandoert) or

              (/data/tabla/sor[@id='s1']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s2']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s3']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s4']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s5']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s6']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s7']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s8']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert) or
              (/data/tabla/sor[@id='s9']/elem[@name=  /data/kordy]/@ertek  =
/data/beszurandoert)



          ,'nem lehet beszúrni','lehet')"/>

    </xf:model>
```

```
</head>
<body>
     <p>
     Sudoku
     </p>
     <p>



    </p>
    <p>Melyik számot választja?


                    <xf:trigger><xf:label>1</xf:label><xf:toggle   case="1"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'1'"/>
       </xf:trigger>

                    <xf:trigger><xf:label>2</xf:label><xf:toggle   case="2"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'2'"/>
       </xf:trigger>

                    <xf:trigger><xf:label>3</xf:label><xf:toggle   case="3"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'3'"/>
        </xf:trigger>

                    <xf:trigger><xf:label>4</xf:label><xf:toggle   case="4"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'4'"/>
       </xf:trigger>
                    <xf:trigger><xf:label>5</xf:label><xf:toggle   case="5"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'5'"/>
       </xf:trigger>
                    <xf:trigger><xf:label>6</xf:label><xf:toggle   case="6"
ev:event="DOMActivate"/>
```

```
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'6'"/>
       </xf:trigger>
                    <xf:trigger><xf:label>7</xf:label><xf:toggle   case="7"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'7'"/>
       </xf:trigger>
                    <xf:trigger><xf:label>8</xf:label><xf:toggle   case="8"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'8'"/>
       </xf:trigger>
                    <xf:trigger><xf:label>9</xf:label><xf:toggle   case="9"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'9'"/>
       </xf:trigger>
              <xf:trigger><xf:label>torles</xf:label><xf:toggle  case="tor"
ev:event="DOMActivate"/>
        <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="' '"/>
       </xf:trigger>
    </p>
     <p>A választott szám:

       <strong><xf:switch>
         <xf:case id="1">1
          <xf:setvalue ev:event="DOMActivate" ref="/data/nem" value="'1'"/>
          </xf:case>
         <xf:case id="2">2
         <xf:setvalue   ref="/data/nem" value="'2'"/>
         </xf:case>

         <xf:case id="3">3</xf:case>
         <xf:case id="4">4</xf:case>
         <xf:case id="5">5</xf:case>
         <xf:case id="6">6</xf:case>
         <xf:case id="7">7</xf:case>
         <xf:case id="8">8</xf:case>
         <xf:case id="9">9</xf:case>
         <xf:case id="tor">erase</xf:case>
       </xf:switch></strong>
    </p>
```

```
     <xf:output ref="case"/>

    <h4> beszúrhat-e: <xf:output ref="/data/status"/><br/>
     Milyen módban van:(<xf:output ref="/data/testbeszur"/>)
      <xf:trigger>


        <xf:label ref="/data/atestbeszur">
          </xf:label>
          <xf:action  ev:event="DOMActivate">
                                    <xf:setvalue      ref="/data/testbeszur"
value="if(/data/testbeszur='teszt','beszur','teszt')"/>
                                   <xf:setvalue      ref="/data/atestbeszur"
value="if(/data/atestbeszur='teszt','beszur','teszt')"/>
          </xf:action>
      </xf:trigger>
      </h4>
```

Mivel túl sok lenne a teljes forráskód ezért csak egy gomb kialakítását rakom be. A többi ugyanúgy működik, csak az index más.

```
 <table width="50%" border="2">
    <tr><!-- a táblázat egy bombja -->

      <td>

      <xf:trigger>

        <xf:label ref="/data/tabla/sor[@id='s1']/elem[@name='1']/@ertek ">

          </xf:label>
          <xf:action  ev:event="DOMActivate">
                <xf:setvalue   ref="/data/kordx" value="'s1'"/>
                          <xf:setvalue     ref="/data/kordy"  value="'1'"/>

                               <xf:setvalue      ref="/data/beszurandoert"
value="/data/nem"/>
                                                                  <xf:setvalue
ref="/data/tabla/sor[@id='s1']/elem[@name='1']/@ertek"
```

```
value="if(/data/testbeszur='beszur',/data/nem,/data/tabla/sor[@id='s1']/ele
m[@name='1']/@ertek)"/>
          <xf:recalculate model="amodel"/>
          </xf:action>
      </xf:trigger>
```

Az XForms nagy csalódás volt a másik két megoldásokhoz képest. Ezzel a nyelvel kezdtem az ismerkedést. Sok idő kellet arra, hogy az XPath lehetőségeit hogyan lehet használni az XForms esetén. Én egy JavaScript-editorral dolgoztam, ami a színkiemeléssel segített átlátni a kódot. Röviden fogalmazva egy rémálom volt valami egyszerű dolognál komolyabbat elkészíteni XForms-ban. A kipróbált implementációk egyike sem fedte le teljesen az XForms specifikációját, de már használhatóak voltak. A teljes implementációkra még várni kell. A szkriptek használata elég nehézkes. Gyorsaságban nincs probléma, az űrlapok készítése is egyszerű, de nehéz összetettebb dolgokat készíteni benne. A kevés példaprogram miatt még több időbe telik egy megoldás megtalálása.

7.7 Az XForms alapú megvalósítás értékelése

könnyen kezelhetőség: könnyen kezelhető/ átlátható kisebb projekteknél
gyorsaság: gyors (Komolyabb XPath kiértékeléseknél sokat lassul.)
erőforrásigény: kevés
készítési idő: közepes
tőlem mekkora időt vont el: nagyon sokat
kereszt platformitás: közepes indoklás: kiegészítők révén bárhol indoklás:
kinézet: átlagos
pénzfüggés: ingyenes
installálás: kiegészítők installálásával

7.8 A három nyelv összehasonlítása

A három nyelv és a többi közül a Flash-re építőket látom kijönni győztesen a webes felületek készítésénél. Ezt a kinézet manipulálásának erőssége miatt gondolom. De nem kétséges, hogy az asztali alkalmazások terén már nincs ekkora előnye. Nekem a XUL okozott kellemes meglepetést. Egyszerűsége és kezelhetősége nagyon tetszett. Az XForms megszorításai szokatlanok voltak nekem, és zavaró volt az is, hogy sokszor nem lehetett eldönteni az hogy az implementáció hibás, vagy valóban nincs meg az a tulajdonság amit használni akartam volna. Nem szeretem a Flash alapú oldalakat, de a Flash használata eléggé meggyőző volt. Szerintem Komplexebb webes megoldásokhoz a Flash alapú megoldásokat érdemes használni (Flex, OpenLaszlo).

8. Befejezés

A jövő eléggé a felületépítő nyelvek felé megy el. Windowson az alábbi lehetőségek vannak: A Vistában jelent meg a VPF(Windows Presentation Fundation) a .NET keretrendszer új prezentációs eszköze, ennek része a XAML, amely pedig egy új XML alapú grafikus elemeket reprezentáló formátum. Erre épülve tetszőleges nyelven lehet (C#, VB, C++) lehet VPF alkalmazásokat, webalkalmazásokat fejleszteni. Linuxon az alábbi lehetőségek állnak rendelkezésre: A Gnome alatt Glade alapú felületek jönnek szóba, de bármelyik program használhatja a saját maga által preferált felhasználói felület leíró nyelvet. Ugyanis Linux rendszerek kinézete nincs egységesítve (csak irányok vannak pl: Gnome, KDE), ez elképzelhetetlen lenne Windows rendszereken, ugyanis ott a cég diktál. Ezek a nyelvek a jövőbeli operációs rendszerek megjelenésének megvalósításához nagy segítséget fognak nyújtani. Webalkalmazásoknál pedig elengedtethetetlen lesz az alkalmazásuk, illetve ilyen irányba halad a szakma. Hogy itt (webes rendszereknél) melyik fog nyerni, azt a jövő dönti el.

9. Irodalom

Könyvek:

Charles E. Brown. The essential guide to flex 2 with actionscript 3.0. Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, 2007.

Magyarországi Web Konferencia 2006 Füzet. Pdf dokumentum web.conf.hu/2006

Ivánfi Zoltán Szoftverfejlesztés Linux alapú kézi számítógépekre 2004. Pdf dokumentum: ivanfi.uw.hu/download/**linux**pda.pdf

INTERNETES adatgyűjtés:

http://www.stud.u-szeged.hu/Erdei.Grete.Katalin/foxml.html

http://htmlinfo.polyhistor.hu/xhtmlref/xhtml1.htm

http://wiki.hup.hu/index.php/XHTML

http://en.wikipedia.org/wiki/Xhtml

http://www.whatwg.org/specs/web-forms/current-work/

http://glade.gnome.org/manual/index.html

http://www.jaxxframework.org/wiki/Documentation

http://xui.sourceforge.net/

http://www.openlaszlo.org/lps4/docs/reference/

http://www.jaxfront.org/pages/overview_whitepapers.html

http://www.xoetrope.com/zone/manual/index.php?zone=XUI

http://skimstone.x-port.net/book/introduction-to-xforms

http://www.w3.org/TR/2007/WD-xforms11-20070222

http://www.w3.org/MarkUp/Forms/2003/xforms-for-html-authors.html

http://www.w3.org/MarkUp/Forms/

http://livedocs.adobe.com/flex/201/langref/index.html

http://www.adobe.com/support/documentation/en/flex/

http://czinkos.wikidot.com/xmlinfomiazxpath

http://developer.mozilla.org/en/docs/The_Joy_of_XUL

http://www.xulplanet.com/tutorials/xultu/