

Debreceni Egyetem
Informatikai Kar

Webalkalmazások modellezése WebML használatával

Témavezető:

dr. Adamkó Attila

egyetemi tanársegéd

Készítette:

Sándor Miklós

programtervező informatikus (MSc)

Debrecen

2009

Tartalomjegyzék

Tartalomjegyzék.....	2
Bevezetés.....	3
A Web használatának elterjedése.....	4
A Web Modeling Language (WebML).....	10
A WebRatio, mint a WebML modelljeinek megvalósítása.....	19
Egy Webkönyvtár létrehozása WebML segítségével.....	22
A WebRatio használatának alapjai.....	22
A Webkönyvtár létrehozásának főbb mozzanatai.....	25
A WebRatio által generált kód elemzése.....	31
Összefoglalás.....	39
Irodalomjegyzék.....	40

Bevezetés

1991 derekán, amikor még csak a legelső szálát szötte a világháló a CERN és a FermiLab között, alig voltak néhányan, akik hallottak volna a World Wide Webről. Ma viszont a kultúra minden területét behálózza, és bolygónkat átfogó kommunikációs közeget teremt az emberek millióinak. Bár eredetileg csak kutatási jegyzetek megosztására készült, mára már komplett alkalmazásokat is találunk az Interneten, gondoljunk csak az egyes utazási irodák honlapjain meghirdetett utazások online foglalásaira.

Persze ahhoz hogy ez a mai helyzet kialakuljon nagyon sok változást történt a Web életében. A statikus weboldalak helyett a megjelentek a dinamikusan generált weboldalak, megtörtént a böngészők szabványosítása, és persze a legfontosabb az Internet elterjedése. Ezekkel a változásokkal a Web 1. korszakából átlépett a következő korszakába: a Web 2.0¹-ba. Ez az elnevezés azonban nem túl szerencsés, mivel a Web túl általános és nem igazán körül határolható, hogy verziózni lehetne. A kifejezés azonban egy nagyon fontos újdonságra mutat rá: a webhasználat új irányaira.

A diplomamunkám célja, hogy bemutassam a WebML² modellező nyelvet, amely összetett weboldalak és webalkalmazások leírására szolgál. Azért választottam ezt a témát, mert a webalkalmazások tervezése és karbantartása a szoftver ipar egyik nagy kihívása, de a WebML segítségével könnyen megoldhatóak a felmerülő problémák. A WebML lehetővé teszi egy weboldal magas-szintű leírását jól megkülönböztethető dimenziók alapján: az adattartalma (strukturális modell), azok az oldalak, amelyek alkotják és az oldalak közötti linkek topológiája (hypertext modell), az oldalak kinézete (prezentációs modell) és az egyes tartalmak testreszabhatósága (személyre szabott modell). A WebML minden fogalma nemcsak grafikus jelöléssel bír, hanem társul hozzájuk egy szöveges XML szintaxis is, így akár a grafikus ábrázolás helyett XML leírást is készíthetünk.

Továbbá bemutatásra kerül a WebRatio nevű program is, amely a WebML modellező nyelv megvalósítása. Segítségével könnyedén megtervezhetjük a WebML-ben használt diagrammokat és a beépített kódgenerátorának köszönhetően pillanatok alatt egy működő webalkalmazást készíthetünk.

¹ Ezt az elnevezést először Tim O'Reilly használta 2004-ben

² Web Modelling Language

A Web használatának elterjedése

Tim Berners-Lee 1989 márciusában állt elő javaslatával, amelyben egy egységes információs rendszert ajánlott munkaadója figyelmébe, hogy megfelelő módon követhessék a korábban lejegyzett anyagokat.

Az "Information Management: A proposal" névre hallgató dokumentum a CERN svájci kutatóközpontjában készült, ahol 1989-ben már komoly gondot jelentett a tudástár rögzítése, tárolása és megfelelő kezelése. A katasztrófát megelőzendő, Berners-Lee kidolgozott egy rendszert, amely áthidalta az eltérő számítógépek közötti inkompatibilitást, könnyű és egyszerű módon téve elérhetővé a teljes kínálatot - ebből született meg néhány évre rá a ma is ismert World Wide Web.

Ahogy azt később a brit kutató elmondta, az elsődleges problémát az jelentette, hogy a nukleáris kutatásokkal kapcsolatos anyagok sok esetben végleg elvesztek, egyéb részleteket pedig csak hosszas nyomozással sikerült felkutatni a helyi központban. Az információs káoszt nehezítette, hogy eltérő számítógépeket használtak, és hiányzott a megfelelő rendszer, így az adatok kezelése és tárolása nem volt megoldott. Az alapjában véve egyszerű javaslattal azonban választ talált a kérdésekre, igaz néhány évre még szükségük volt, hogy a platformot élesben is kipróbálják.

Eredetileg 20 webszervert állítottak üzembe, ma pedig az elérhető oldalak száma már százmilliárd felett van. Hatalmas növekedésen ment tehát keresztül az eredeti platform, amely igazán az 1990-es évek közepén, a Netscape böngésző elterjedésével indult meg igazán. Az elképzelést egyébként Berners-Lee főnöke tágan értelmezhetőnek, de mindenképpen izgalmasnak találta, ezért is bólintott rá a munkálatok megindítására. Több mint két évükbe került, míg a kezdeti rendszert kialakították, onnantól kezdve azonban többé-kevésbé megoldottnak volt mondható az európai atomkutató szakemberek közötti gyors és kényelmes információcsere.

A CERN illetékesei 1991-ben úgy döntöttek, hogy nem tudják biztosítani a platform további fejlődését és 1993. április 30-án a bejelentették, hogy a világháló mindenki számára szabad és ingyenes.

Később, még ugyanebben az évben az NCSA³ kiadott egy alkalmazást Mosaic névvel,

³ National Center for Supercomputing Applications

amely egy webböngésző és egy Gopher kliens ötvözete volt. Eredetileg csak Unix gépeken volt elérhető forráskód szinten, de 1993 decemberében a Mosaic új verziója már Apple Macintosh és Microsoft Windows telepítőkkel érkezett. A Mosaic-kal a web népszerűsége meredeken emelkedni kezdett.

Ezt követően drasztikusan megnőtt a böngészők száma, sok egyetem és vállalat indított fejlesztési projekteket ebben az irányban. Ilyen például a Telenor (egy norvég kommunikációs vállalat), amely 1994-ben elkészítette az Opera böngésző első verzióját.

A web népszerűsége később kereskedelmi érdeklődést is generált. Marc Andreessen és Jim Clark közösen megalapították a Mosaic Communicationst, ezt később Netscape Communications Corporationnek nevezték el, majd elkezdtek dolgozni a később Netscape Navigatorként ismert böngészőn. Az első verzió 1994 decemberében jelent meg.

Az NCSA kereskedelmi ága, a Spyglass Inc. közben eladta a Mosaic technológiájának jogát a Microsoftnak, és ezzel az Internet Explorer alapjait szolgáltatta. Az első verzió 1995 augusztusában jelent meg.

A Netscape és a Microsoft közötti harc alakult ki, amely hamarosan elérte a tetőpontját, és mindketten megpróbáltak új funkciókkal előrukkolni, hogy megnyerjék maguknak a fejlesztőket. Ez később a „böngészők háborúja” néven vált ismertté. Az Operának ezalatt egy kisméretű, de stabil pozíciója volt, és innovatív fejlesztésekkel, valamint a szabványok támogatásával próbáltak fennmaradni.

A böngészőháború alatt a Microsoft és Netscape főleg az új funkciók fejlesztésére koncentrált ahelyett, hogy a már támogatott funkciókat javították volna ki. Saját, védett funkciókat építettek be a böngészőkbe, emellett olyan új funkciókat valósítottak meg, amelyek már léteztek a többi böngészőkben, de ezek nem voltak kompatibilisek egymással.

A webfejlesztők ezekben az időkben egyre inkább arra voltak kényszerítve, hogy a valódi fejlesztés helyett az ilyen zavaros helyzetek megoldásával foglalkozzanak. Néha arra volt szükség, hogy két változatot készítsenek egy oldalból a két nagy böngésző számára, amelyek gyakorlatilag teljesen megegyeztek. Egyesek egyszerűen csak az egyik böngészőt támogatták, és a másik böngésző felhasználóit letiltották az oldalról. A munka így igazi rémálommá vált, és a fejlesztők részéről már nem voltak messze az elkerülhetetlen következmények.

1994-ben Tim Berners-Lee megalapította a W3C⁴-t a Massachusetts-i Technológiai

⁴ World Wide Web Consortium

Intézetnél, a CERN, a DARPA (ez lett az ARPA új neve), valamint az Európai Bizottság támogatásával. A W3C a különböző protokollok és technológiák szabványosítását tűzte ki maga elé egy olyan web felépítésének érdekében, amelyen az információ a világ népességének lehető legnagyobb része számára elérhetővé válhat.

A következő években a W3C több specifikációt is kiadott (amelyeket „ajánlásoknak” neveztek), mint például a HTML 4.0, a PNG képek szerkezete, vagy a CSS első és második verziója.

Viszont a W3C nem követeli meg az ajánlások betartását. A gyártók csak akkor kell betartsák a W3C dokumentumokban foglaltakat, ha a terméküket meg akarják jelölni, hogy megfelel a W3C feltételeinek. A gyakorlatban ez nem értékes eladási szempont, mivel a felhasználók közül szinte senki sem tudja, sőt valószínűleg nem is érdekli őket, hogy mi fán terem a W3C. Ennek következtében a „böngészők háborúja” zavartalanul folytatódhatott.

1998-ban a böngészőpiacot az Internet Explorer 4 és a Netscape Navigator 4 uralta. Megjelent az Internet Explorer 5 beta verziója, amely már egy új és saját fejlesztésű dinamikus HTML-t támogatott. Ez azt jelentette, hogy a professzionális webfejlesztők már öt különböző módon kellett tudjanak JavaScript kódokat írni.

Ennek eredményeképpen a professzionális webfejlesztők és webdesignerek egy csoportja összeállít, és együtt megalakították a WaSP-ot (Web Standards Project). Az ötlet alapján az volt, hogy a W3C dokumentumait szabványnak nevezzék ajánlások helyett, így talán sikerülhet meggyőzni a Microsoftot és a Netscape-et arról, hogy támogassák őket.

2000-ben a Microsoft kiadta az Internet Explorer 5 Macintosh Editiont. Ez egy nagyon fontos mérföldkő volt, mivel ez lett az alapértelmezett böngésző a Mac OS alatt, és már támogatta a W3C ajánlások egy jelentős részét. Ez az Opera megfelelő CSS és HTML támogatásával együtt már megadta a lehetőséget a webfejlesztők és webdesignerek számára, hogy először tapasztalhassák meg a szabványos weboldalak fejlesztésének a kényelmét.

A WaSP-nak ráadásul sikerült meggyőznie a Netscape-et arról, hogy halassza el a Netscape Navigator 5.0 megjelenését addig, amíg nem támogatja jobban a szabványokat (ez a változat vált később a manapság nagyon népszerű böngésző, a Firefox alapjává). A WaSP ezen kívül indított egy Dreamweaver Task Force nevű csoportot is, hogy rávegye a Macromediát a népszerű fejlesztői eszközük szabványosabbá tételére.

2001-ben az „A List Apart” nevű népszerű webfejlesztői oldalt is teljesen újratervezték, és az egyik cikkben, amely a miértekről és hogyanokról szólt, ez állt:

Hat hónapon, egy éven vagy legkésőbb két éven belül minden oldalt ezekkel a szabványokkal fognak készíteni. Figyelhetjük, ahogy a tudásunk fölöslegessé válik, vagy elkezdhetjük már most tanulni a szabványos módszereket.

Ez egy kissé optimista kijelentés volt, a szabványokat még most sem támogatja az összes oldal. De sokan hallgattak rájuk. A régi böngészők részesedése lecsökkent, és később két másik rendkívül népszerű oldalt is átalakítottak teljesen szabványosra: 2002-ben a Wired magazint, és 2003-ban az ESPN-t, akik így a webes szabványok és az új technológiák élharcosaivá váltak.

Ugyancsak 2003-ban Dave Shea elindította a „CSS Zen Garden” oldalt. Ez valószínűleg nagyobb hatást ért el a webfejlesztőknél, mint addig bármi más, ugyanis képes volt azt bemutatni, hogy egy oldal teljes designját meg lehet változtatni egyszerűen a stílus lecserélésével, miközben a tartalom pontosan ugyanaz marad.

Az utóbbi években nagyon sok változás történt, melynek jeleit már a nem szakmabeliek is észlelhetik az itt-ott megjelent újdonságok kapcsán. Rendkívül elterjedtek és népszerűek lettek a tartalomkezelő rendszerek, egy durva egyszerűsítéssel leszűkíthetjük ezt a blogok, a webnaplók területére – egész szubkultúra alakult ki körülöttük. Gyakorlatilag bárki számára lehetővé vált, hogy egyszerűen közzétegye gondolatait. A blogok a cégek körében is terjedni kezdtek, s manapság már nem csak a személyes, hanem a vállalati naplók is terjedőben vannak, nem beszélve az életünkbe hirtelen berobbant politikai blogokról.

Az újdonságok listáján a blog csak az egyik a sok közül, emellett számos eddig nem ismert közösségi szolgáltatás is napvilágot látott. Képek megosztását, közösségi/szociális kapcsolati hálók kialakítását, közösségi híroldalak létrejöttét segítő szolgáltatások alakultak, s alakulnak a mai napig gombamódra.

Fontos megértenünk, hogy a változás főként nem technikai jellegű. Nem jelentős technikai újdonság jelent meg, inkább a fejlesztők kezdtek másképp gondolkodni, érdekes szolgáltatások váltak az egyik pillanatról a másikra hihetetlenül népszerűvé. És nem szabad kihagyni a webet használókat sem, akikre az egész jelenség épít: megváltozott, megváltozik a webhez való hozzáállásunk, új kommunikációs lehetőségeket fedezünk fel, és kezdünk el aktívan használni. Közösségek alakulnak ki egy gondolat, egy alkalmazás körül, s ezekből a közösségekből áll össze végül egy tömeg, mely az egyedi felhasználókon felülemelkedve egy új minőséget képes létrehozni.

Persze ne becsüljük le a technikai újdonságokat sem, bár ezek egy része még ma is

gyerekcipőben jár, s vár az elterjedésre. A kivétel az egyre népszerűbb RSS, mely különböző hírek, tartalmak közzétételét, s akár robotok, akár emberek általi fogyasztását teszi lehetővé. Emellett megjelentek az emberi kapcsolatok jelölésére szolgáló leírónyelvek, illetve kezd elterjedni a szemantikus web fogalma is. A keresőrobotok és más intelligens weben böklászó programok így felismerhetik a kapcsolatokat az oldalak között, s ezzel egy hálót szöve intelligens módon összekapcsolhatják azokat.

A web a változások hatására egyre inkább középpontba kerül. Már ma is vannak olyan területek, ahol úgy végezhetünk munkát, hogy egy böngészőn kívül más alkalmazás nem is fut gépünkön. Például egy újságíró, szerkesztő utánanéz a neten a híreknek, a webes levelezőben kapcsolatot tart a külvilággal, a csevegőszobákban társalog, feltölt és letölt fotókat a webes albumokból, s cikket publikál a webes felületű szerkesztőségi rendszeren keresztül.

Az előbb felsorolt lehetőségeket hívjuk webalkalmazásoknak. A hagyományos alkalmazás a webre költözik, bárholnan a világon elérhetővé válik, bármely elterjedt böngészőből használható lesz. A megjelent megoldások kapcsán egyre inkább az a trend rajzolódik ki, hogy akár kompromisszumok nélküli, sőt, új lehetőségekkel felruházott alkalmazások jelenhetnek meg.

A valódi alkalmazás érzését az újratöltés nélküli kommunikáció megjelenése segítheti, de olyanok is ismertek, melyek egy operációs rendszerhez hasonlatos felületet jelenítenek meg a böngészőben: asztallal, átméretezhető ablakokkal.

Számos kiváló példát találunk web alkalmazásokra akár ma is, s több próbálkozás is létezik a szokásos irodai alkalmazások (mint például a szövegszerkesztő, a táblázatkezelő) webes implementálására is. A webfejlesztők körében ma már nem okoz meglepetést egy webes adatbáziskezelő program, a szervert adminisztrátorok pedig webes szervert karbantartó felületeket is ismerhetnek.

A webes alkalmazások előnye, hogy nem kell minden gépre külön telepíteni, elég a webszerverre, s máris bármelyik gépről használhatóak. Ez a karbantartást is jelentősen egyszerűsíti. Egy másik érdekes szempont, hogy mivel a gépekre nem töltünk le semmilyen futtatható állományt, a vírusok tekintetében is sokkal védettebbek lehetünk.

A fentiek tükrében ne lepődjünk meg tehát, ha egyszer csak kedvenc alkalmazásunkhoz egy nagyon hasonló programmal egy weblapon találkozunk. Minden bizonnyal nagy jövő áll a web előtt, számos olyan alkalmazást fogunk megismerni az

elkövetkező években, melyekről sohasem gondoltuk, hogy egyszer egy weblapon köszönnek vissza ránk, sőt, sohasem gondoltuk, hogy lehetőség van megvalósításukra.

A Web Modeling Language (WebML)

A webfejlesztés korai időszakában az volt a bevett szokás, hogy a webalkalmazások fejlesztésénél kis hangsúlyt fektettek a fejlődési folyamatra és egyszerűen csak a megoldás megvalósítására koncentráltak. Így viszont rengeteg vállalat érezhette egy idő után, hogy számos probléma előjött a megnövekedett számú és komplexitású weboldalak menedzselése folyamán. A webfejlesztőeszközök segítenek leegyszerűsíteni az adat-intenzív webalkalmazások létrehozását és telepítését olyan oldal generátorokkal, mint például a Sun Microsystems Java Server Pages terméke, amely elsődleges funkciója, hogy automatikusan nyerjen ki információkat adatforrásokból és azokat felhasználó által létrehozott sablonokba csatolja. Az ilyen rendszerek ugyan hatékony implementációs eszközök, de mégis kevés támogatást nyújtanak a fejlesztési folyamatok áthidalására. Azonban az olyan vállalatoknak, amelyek webalkalmazásokat készítenek, igencsak nagy szükségük van a metódusok megtervezésére, formalizálásra, nyelvekre és olyan eszközökre, amelyekkel hatásos módon lehet az aktuális Web technológiát kiegészíteni és így alkalmasak a tervezés minden szempontjának lefedésére.

Ezekre a szükségekre a W3I3⁵ projekt adta meg a választ. A projekt során egy újfajta webmodellező nyelv jött létre, amelyet WebML-nek neveztek el. A WebML az adat-intenzív webalkalmazások magas szintű, platform független specifikációját tűzte ki célul.

A WebML lehetővé teszi a tervezők számára, hogy magas szinten leírják egy oldal alap jellemzőit. A WebML olyan intuitív grafikus módszert használ a fogalmak meghatározására, amelyek hatékonyan ábrázolhatóak CASE eszközök segítségével, így az a fejlesztési csapatban dolgozó nem hozzáértők számára is érthető. A WebML az XML szintaxist is támogatja, így lehetőség van a Web oldal implementálását egy automatikus kódgenerátorra bízni. Egy oldal specifikációja a WebML-ben négy perspektívából történik:

- strukturális modell,
- hypertext modell,
- megjelenítési modell,
- személyre szabott modell.

⁵ W3I3: Web-Based Intelligent Information Infrastructures

Strukturális modell:

Segítségével egy oldal adattartalma jellemezhető a fontos entitások és összefüggések szempontjából. Tulajdonképpen a más informatikai ágakban (adatbázis tervezés, szoftvertechnológia) szereplő adatterv koncepcionális modelljeinek egy megfelelő átdolgozásának is tekinthető. Meglehetősen nagy hasonlóságot tapasztalhatunk az E-R modell illetve az UML modell jelölés rendszerével, de ez nem meglepő, mivel nem az volt a céljuk, hogy egy teljesen új jelölés rendszert hozzanak létre, hanem a meglévő modellek alapján egy jelölési rendszer kialakítása. A redundáns és számított adatok által támasztott követelmények miatt a strukturális modell egy egyszerűsített OQL⁶-szerű lekérdező nyelvet is a rendelkezésünkre ad.

A WebML strukturális modell alapvető elemei az entitások, amik az adatelemek hordozói, és az összefüggések, amik az entitások szemantikus kapcsolatát hozzák létre. Az entitások nevesített attribútumokkal rendelkeznek, amelyek adott típusúak. Az entitások öröklődési hierarchiába rendezhetők. Az összefüggések rendelkezhetnek számosság megszorítással vagy szerep névvel.

A strukturális modell alapjai:

- Entitás: webalkalmazás objektumának egy osztálya,
- Attribútum: egy entitás egy tulajdonsága,
- Összefüggés: entitások közötti kapcsolatok,
- IS-A hierarchia: osztályozáshoz és csoportosításhoz.

Az entitások olyan dolgok ábrázolására szolgálnak, amelyek hasonló jellemző tulajdonságokkal rendelkeznek. A való világ egyes elemei az entitások egy példányai, általában ezeket is objektumoknak szokás nevezni. Egy entitás összes példányát az entitás populációjának nevezzük. Az entitások rendelkeznek egyedi azonosítóval (OID), ami azonban csak egy absztrakt fogalom, és ezt különböző módon lehet implementálni, például kulcsként egy relációs adatbázisban vagy XML adatforrásokban XML ID-ként.

Egy attribútum egy entitás példányainak a közös tulajdonsága. Néhány objektumnak lehet nem értelmezett attribútuma is. Jellemzően felhasználó által definiált a nevük és típusuk, egyértékűek (ez akár null érték is lehet) és nem rendelkeznek belső al-struktúrával.

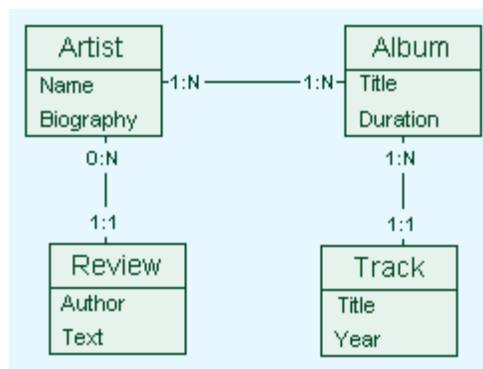
Egy bináris összefüggés, vagy egyszerűen csak összefüggés, példánya egy olyan

⁶ OQL: Object Query Language

kapcsolat két objektum között, amely a való világban egy olyan tényt jelent, ami magába foglalja ezeket az objektumokat. Egy ilyen összefüggés az objektumok közötti összefüggés példányainak a leírására szolgál. Az entitás példányai közötti sokféle kapcsolatot is képesek vagyunk reprezentálni ilyen összefüggésekkel. A bináris összefüggések esetén beszélhetünk összefüggés szerepről, ami a két irány közül az egyiket jellemzi. Ezek neveit szintén a felhasználó adja meg. Az összefüggések mindkét irányára megadhatók annak maximális és minimális számosságai. A számosságok megadása fontos, mivel olvashatóbbá és kifejezőbbé teszi a szerkezeti sémát. Így létrehozhatunk olyan attribútumokat, amelyekhez több érték is megadható. Az IS-A hierarchia két entitás közötti olyan kapcsolat amely egy entitás, mint al-entitás és egy másik entitás, mint szuper-entitás között jön létre. Az al-entitás a szuper-entitás minden jellemzőjét örökli.

Jelölések:

Az entitasokat dobozzal jelöljük és a doboz felső felében az entitasok nevét írjuk. Az attribútumok a dobozok alsó felében vannak felsorolva. Az attribútumok felsorolása opcionális. Az entitasok közötti összefüggést a dobozok között vonal jelzi. Az IS-A hierarchiát az al-entitásból a szuper-entitásba mutató nyíl jelzi.



1. ábra
Egy strukturális modell

hypertext modell:

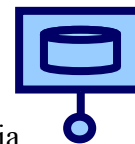
A hypertext modell az oldalon található egy vagy több hypertext leírására szolgál. Minden különböző hypertext egy úgynevezett oldal nézetet definiál. Az oldal nézetek leírása két al-modell segítségével történik: kompozíciós modell és navigációs modell.

A kompozíciós modell megadja, hogy milyen oldalakból áll egy hypertext és milyen

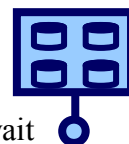
tartalmi egységekből épül fel az oldal. Hét fajta tartalmi egység áll rendelkezésünkre az oldalak létrehozásához: adat, multi-adat, mutató, többválasztásos, hierarchikus, bejegyzési és görgetős egységek. Az adat egység szolgál arra, hogy egy egyszerű objektum információt megjelenítsük, a többi egység fajta pedig különböző módjai annak, hogy egy objektum halmazzt végigjárjunk.

Egységek:

Adat egység (data unit): Az entitás egy példányáról hordoznak információkat. A megjelenítendő objektum OID-ját vagy a szelektor paramétereit kaphatja inputként. Outputként pedig szintén az objektum OID-ját és annak paramétereit adja.



Multi-adat egység (multi-data unit): Egy entitás vagy komponens több példányát mutatja be. A multi-adat egység leírása két részből áll: első része egy konténer, ami tartalmazza, hogy mely entítások, összefüggések vagy komponensek példányait akarjuk megjeleníteni, a másik része pedig a megjelenítéshez használt adat egységeket minden egyes példányhoz.



Mutató egység (index unit): Egy listaként jeleníti meg egy entitás vagy komponens több példányát. Megadása két részből tevődik össze: első része egy konténer megadása, ami tartalmazza, hogy mely entítások, összefüggések vagy komponensek példányait akarjuk megjeleníteni, a másik része az attribútumok, amelyeket index kulcsként használunk.



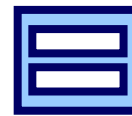
Többválasztásos egység (multichoice unit): A mutató egység egyik változata, szintén több egység megjelenítésére szolgál, de itt nem csak egy objektumot választhat a felhasználó, hanem többet is, így ennek megfelelően a kimenetben is nem egy hanem több objektum OID-ja adódik át. Szelektor segítségével előre bepipált elemeket is megjeleníthetünk.



Hierarchikus egység (hierarchical unit): A mutató egység másik változata. Az entítások olyan megjelenítésének modellezésére szolgál, amelyben az entítások és azok al-entitásai is megjeleníthetőek. A felhasználó csak egy objektumot választhat, de azt bármelyik szintről.



Bejegyzési egység (entry unit): Beviteli mezők leírására szolgáló egység. Mivel új adat bevitelére képes vele a felhasználó, ezért itt a tartalom adódik át link paraméteren keresztül. A mezőkhöz különböző tulajdonságok is megadhatóak, mint például láthatóság, típus, validáció.



Görgetős egység (scroller unit): Lehetőséget nyújt, hogy egy konténerben található objektumokon végig görgessünk. A görgetős egységet általában egy adat egységgel együtt használják, ami a konténer éppen aktuális elemét képviseli.



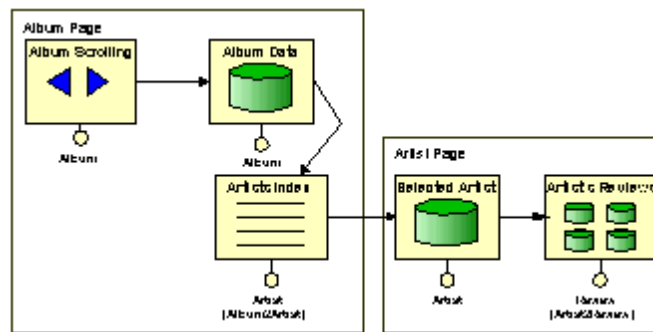
Az egységek finomsága túl apró lehet egy alkalmazás összetétel követelményeihez, amelyek általában megkövetelik, hogy a több egységben tárolt információk egyszerre adódjanak át. A WebML ezeknek az igények a kielégítéséhez egy oldal elképzelést nyújt. Az oldal egy képernyő független területének absztrakciója, amelyet független interfész blokként kezelnek.

Az oldalak belső felépítésüket tekintve egységekből és/vagy más oldalakból állnak. Az utóbbi esetben a konténeroldal al-oldalait független megjelenítési blokként kezelik.

Az egységek és oldalak nem elszigetelten léteznek, hanem valamilyen kapcsolatnak kell lenniük egymás között, hogy létrehozzák a hypertext szerkezetet. A navigációs modell célja, hogy meghatározza azt az utat, ami szerint az egységek és az oldalak összekapcsolhatók. Erre a célra nyújtja a WebML a link fogalmát, amelynek két változata van:

Kontextuális link: Koherens módon összekapcsolja az egységet. Információt ad át a forrás egységtől a cél egységnek. Ez az információ határozza meg a célegységben megjelenített objektumot vagy objektum halmazt.

Nem-kontextuális link: Teljesen független oldalakat kapcsolnak össze.



2. ábra
Egy hypertext modell

Az egyes egységek által a kontextuális linken átadott információk a következőképpen alakulnak:

- adat egység: az egységben aktuálisan jelenlévő objektum azonosítója,
- multi-adat egység: az egységben található adat egységekből jön létre az információ,
- mutató egység: a listából kiválasztott érték kulcsa,
- többválasztásos egység: a kiválasztott értékek azonosítója,
- hierarchikus egység: a hierarchiából kiválasztott érték kulcs,
- bejegyzési egység: a bevitt objektum,
- görgetős egység: a görgetős parancsok által kiválasztott objektum azonosítója.

Rejtett navigáció

A felhasználó és az alkalmazás közötti interakció kialakulás általában két irányból is történhet: nem csak a felhasználó választhatja ki, hogy milyen tartalom jelenjen meg egy hiperlinkre való kattintáskor, hanem néha az alkalmazás is önállóan meghatározza, hogy milyen oldalt mutasson, azáltal, hogy „megérzi” a felhasználó bizonyos kattintásának a hatását. Ezt a képességet úgy modellezzük a WebML-ben, hogy kifejezzük a több egységet tartalmazó oldalak „kitöltési szemantikáját”. Például a felhasználó hozzáférhet egy oldalhoz, amely két egységet tartalmaz: egy entitáson lévő mutató egységet, ami az entitáshoz tartozó adat egységre mutat. Ebben az esetben a mutatott adat egység eldöntetlen. A WebML három lehetőséget ad arra, hogy megoldjuk az eldöntetlen egységek sorsát:

1. Üresen hagyja az egységet, így a felhasználónak kell kiválasztani a listából, hogy melyik objektum jelenjen meg az egységben,
2. Egy előre definiált alapértelmezett érték alapján tölti ki az egységet,

3. Egy deklaratív lekérdezés segítségével tölti ki az egységet.

Oldal nézetek

A strukturális modell és a hypertext modell szétválasztása lehetővé teszi, hogy többféle nézetet is létrehozzunk ugyanahhoz az adathoz, amelyek hangolhatóak, így összetett hozzáférési igényeket is ki tud elégíteni. Eltérő oldal nézetek hozhatóak létre a különböző felhasználói csoportok részére. A WebML oldal nézetek olyan oldalak és linkek halmazát tartalmazzák, amelyek mind elérhetőek a kijelölt kezdőoldaltól. Egy adott Webalkalmazás minden oldalnézete ugyanazzal a szerkezeti sémával rendelkezik.

Alapértelmezett hypertextek

A WebML tartalmaz néhány rövidítést, hogy a hiányos specifikációkból is elérjünk egy futó alkalmazást, ezzel pedig gyorsítva a prototípus készítést. A WebML támogatást nyújt az alapértelmezett hypertext létrehozásában különböző szabályok alapján. Ez az alapértelmezett hypertext modell a strukturális modell minden fogalmát pontosan egy egységbe képezi le és minden definiált entitás számára alapértelmezett indexeket nyújt. Az így létrehozott hypertext segítségével alapértelmezett oldal nézetek is létrehozhatóak.

A WebML egyéb jellemzői

Származtatás

A származtatás az a folyamat, amely során redundáns adatokat adunk az oldalhoz, hogy növeljük annak kifejező erejét. A származtatás lekérdező nyelv segítségével történik, ami az OQL korlátozott verziójának is tekinthető. Akár entitásokat, attribútumokat, komponenseket és összefüggéseket is származtathatunk.

Megjelenítési modell

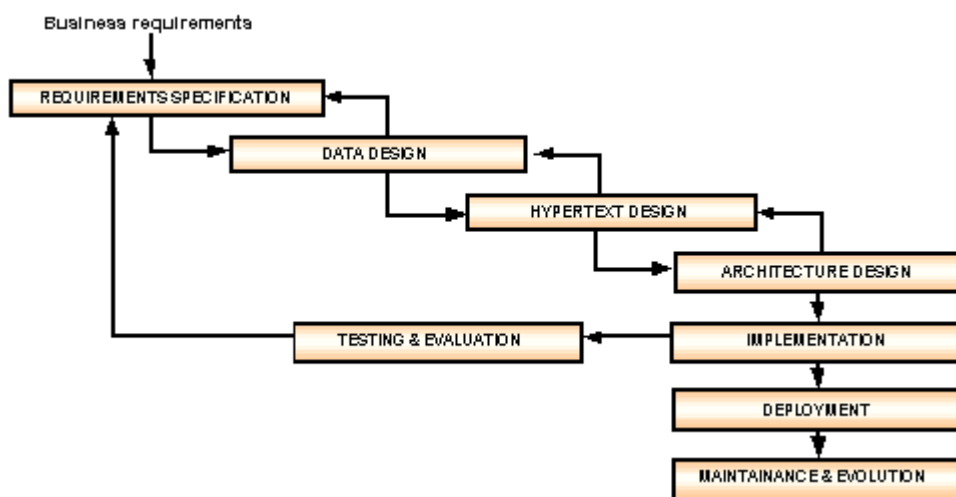
A megjelenítési modellel fejezhető ki az oldalak szerkezete és grafikus megjelenése, a kimeneti eszköztől és a fordítási nyelvtől függetlenül egy absztrakt XML szintaxis segítségével. A megjelenítési specifikációk lehetnek oldal központúak vagy általánosak. Előbbi esetben előírja egy kitüntetett oldal megjelenését és explicit módon utal annak tartalmára, utóbbi esetben pedig előre definiált modelleken alapulnak függetlenül az oldal

különleges tartalmától és csak utalnak az általános tartalmi elemekre.

Személyre szabott modell

Felhasználókat és felhasználó csoportokat modelleznek a szerkezeti sémában az User és Group nevű előredefiniált entitások révén. Ezeknek az entitásoknak a jellemzői tárolják a csoport specifikus és egyéni tartalmakat. Minden felhasználó tagja legalább egy csoportnak. OQL-szerű deklaratív kifejezéseket adhatunk hozzá a szerkezeti sémához, amely az User és Group entitásokban tárolt profil adatok alapján meghatározza a tartalmat. Ezt a megszemélyesített tartalmat használhatjuk mind az egységek megadásakor mind pedig a megjelenési specifikációk meghatározásakor. Egy egyszerű XML szintaxis segítségével olyan oldal által előidézett eseményekre való reagálást is meg tudunk határozni, mint például kattintás, tartalom frissítés. Ezeket üzleti szabályoknak nevezzük. Az üzleti szabályok jellemzően új felhasználóval kapcsolatos információt hoznak létre vagy frissítik az oldal tartalmát. A lekérdezések és az üzleti szabályok két alternatív paradigmát (egy deklaratív és egy procedurális) adnak a személyes követelmények hatékony kifejezésére és kezelésére.

A WebML életciklus



3. ábra
A WebML életciklus

Mint az ábrából is látszik, a webalkalmazások fejlesztésének a WebML megközelítése olyan különböző fázisokból áll, amiket ismétlődő és inkrementális módon kell alkalmazni. Az eljárás több cikluson megy keresztül, mindegyik ciklusban létrejön egy prototípus vagy az

alkalmazás egy részleges verziója, ami lehetővé teszi a tesztelést és értékelést a korai fejlesztési fázisoktól.

Az egész eljárásból, az adat és a hipertext tervezései azok a tevékenységek, amelyek a legbefolyásoltabbak a WebML által.

Az adat tervel nem az a cél, hogy konceptuális adatmodellezéssel váltsa fel az általános célú irányelvet, hanem egyszerűen kiterjessze őket néhány speciális ököl szabállyal, amelyek segíthetnek a Webre történő adatok tervezésekor. Az adat megjelenítésnek és a tartalom kezelésének vannak olyan szabályosságai és sajátosságai, amelyek kihasználhatóak az adatok tervezésekor. Ezeknek a felismerése segítheti az adattervezőt, hogy a munkáját szisztematikus módon szervezze meg, ezáltal következetesebb adatsémákat hozzon létre. Tehát ez a módszer azt hangsúlyozza, hogy a különböző szerepeket különböző objektumok valósítják meg, és ezt a megkülönböztetést használja, hogy egy webalkalmazás adatsémájának összeállításához egy lépés listát adjon.

Az adat tervezés után a hipertext tervezéskor a következők szerint kell eljárni:

- Az első nyers tervezési folyamatnak az a célja, hogy létrehozzon egy magas szintű leírását az oldalnézeteknek, ami egy informális szöveges jelölést használ, hogy kifejtse az egyes területek láthatóságát és az entitások és az adatséma kapcsolatai tekintetében meghatározza az egyes területek tartalmait.
- A részletes terv a nyers terv egy hierarchikus kifinomultsága, amiben az oldal nézetek vázlatát progresszív módon átdolgozzák, amíg azok a felhasználó követelményeinek megfelelően nem lesznek WebML oldalak és egységek gyűjteményei.

A WebRatio, mint a WebML modelljeinek megvalósítása

A WebRatio egy Web Models által készített termék, ami egy Eclipse alapú CASE⁷ eszköz kifejezetten webalkalmazások fejlesztésének segítésére. A cég elsődlegesen a WebRatio fejlesztésére és reklámozására fókuszál. Célpiaca webügynökségek, szoftverintegrátorok, IT tanácsadó vállalatok, közepes és nagy vállalatok IT osztályai és közigazgatások.

WebRatio az első Model Driven fejlesztési környezet, ami lehetővé teszi, hogy webalkalmazások interfészeit modellezzük és automatikusan generáljuk. A generált alkalmazások szabványos Java alkalmazások, így lehetőségünk van rá, hogy bármilyen Java alkalmazás szerverre feltelepíthessük, mint például Tomcat, JBoss, Resin, IBM WebSphere, BEA WebLogic, stb. Ezenkívül az alkalmazások bármilyen adatbázissal képesek kommunikálni.

Az 1996-97-es években a Politecnico di Milano olasz egyetem kutatói egy forradalmi munkába kezdtek az adat-intenzív webalkalmazások fejlesztése terén és létrehozták az első verzióját egy Model Driven kódgenerátor alapjának. 1998-ban az Autoweb System-et bemutatták az Extending Database Technology nevű európai konferencián, mint az első nemzetközi példáját a három rétegű webalkalmazások Model-Driven kódgenerátoraként. 2001. október 31-én megalakult a Web Models s.r.l.⁸ mint a Politecnico di Milano egyetem leányvállalata. A WebRatio első nagy erőpróbájára is ebben az évben került sor az Acer keretein belül. Az új honlap 14 országban lett elérhető, mindössze 7 heti fejlesztés után. A 2002 és 2004 közötti időszakban a WebRatio-t kibővítették, hogy a webszolgáltatások interakcióját is támogassa. A következő két évben két fő kiadásban is arra törekedtek, hogy fejlesszék az egyes modellek kifejező erejét és a használat könnyebbségét. 2007-ben kiadják az 5-ös verziójú WebRatio-t, amely immár az Eclipse fejlesztői platformba integrált.

WebRatio 5

A WebRatio 5 nem csak támogatja a WebML nyelvet és módszertant, de az előző verzió implementációs tapasztalatait kihasználva egy felhasználóbarát alkalmazást nyújt

⁷ CASE (Computer-Aided Software Engineering): Számítógéppel támogatott szoftver-fejlesztés

⁸ A magyar Kft-nek megfelelő cégforma

paradigmák tervezésére és kódgenerálásra. Ez az eszköz egy Eclipse plugin készletként lett kifejlesztve, ezáltal kihasználva ennek az IDE keretrendszernek az előnyeit. Azonfelül új képességeket nyújt a modellek kibővítésében, projekt dokumentációban és a fejlesztés folyamat új fázisainak lefedésében. A teljes szemlélet a webalkalmazások specifikációjának, tervezésének, ellenőrzésének és implementációjának teljes lefedése felé halad.

Az Eclipse egy nyílt forráskódú platformfüggetlen IDE keretrendszer, amely Linux, Windows vagy akár Mac OS X operációs rendszereken is futtatható. A WebRatio 5-ös verziója tulajdonképpen egy különleges Eclipse perspektíva.

A WebRatio hosszú távú célja, hogy teljes mértékben lefedje a fejlesztési folyamatot. Ennek megvalósítása érdekében néhány béta eszköz áll fejlesztés alatt, amelyek a dokumentáció generálásban és új tervezési lépések lefedésében segítenek.

Használata során először meghatározzuk az alkalmazási modellt: ez a modell tartalmazza a Web alkalmazás három alapvető szintjét: adat, üzleti-logika, megjelenítés. A három szint egymástól különálló és független, tehát lehetséges az, hogy úgy hozzunk létre egy új alkalmazást, hogy például csak a megjelenését változtatjuk meg, az adat és üzleti-logikát nem.

Az adatmodell meghatározása az E-R modell segítségével valósul meg.

Az üzleti logikát a WebML segítségével határozza meg, mivel az képes minden funkcionalitást leírni, amely egy webalkalmazásnál előfordulhat.

A megjelenítési modellben szabály halmazok (sablonok) segítségével meghatározza a szerkezetet, amelyet aztán bármely nyelvre átvihetünk, például html, css, javascript, flash, pdf, stb.

A modellek meghatározása után a WebRatio képes egy gombnyomással legenerálni minden webalkalmazást.

Az adatmodell alapján legenerálja és biztosítja a fizikai adatbázis szerkezetét. Ugyanez fordítva is működik, tehát lehetőségünk van egy létező adatbázishoz csatlakozni, hogy a szerkezete alapján automatikusan létrehozza a program az adatmodellt. Egy alkalmazáshoz több adatbázist is definiálhatunk.

A logikai modell segítségével létrehozza azokat a Java osztályokat és konfigurációs fájlokat, amelyek az alkalmazás funkcionalitásához szükségesek: adat írás és olvasás, tranzakciók, eljárások, számítások, stb.

A megjelenítési modellből pedig a dinamikus JSP oldalak hozhatók létre.

A létrehozott alkalmazásnak nincsenek saját futásidejű komponensei és teljesen független a WebRatiotól.

A WebRatio által generált kód teljes mértékben testreszabható. A testreszabás a generálási szabályokon valósul meg, nem a generált kódon. A WebRatio azt is lehetővé teszi a felhasználók számára, hogy saját komponenseket és generálási szabályokat hozzanak létre.

A WebRatio képes arra, hogy egy alkalmazást végigvezessen az életciklusának minden állapotában: az első prototípustól az utolsó kiadásig valamint a karbantartási folyamatok alatt is.

A diplomamunkám írásakor az aktuális WebRatio verzió az 5.1.0, amely 2008. november 27-én jelent meg. Ez a verzió a következő újításokat vezette be:

- Újrafelhasználható modulok (Reusable Modules),
- Munkák (Jobs),
- URL kulcsszavak (URL Keywords),
- Adat exportálása Excel formátumba,
- A WebRatio az Eclipse 3.4.1-es kiadásán működik.

Egy Webkönyvtár létrehozása WebML segítségével

A következőkben bemutatom, hogy hogyan lehet az előbb felvázolt technológiák segítségével egy egyszerű webalkalmazást létrehozni. Sorra veszek minden fontosabb szempontot, amire egy webalkalmazás tervezésekor és megvalósításakor figyelni kell.

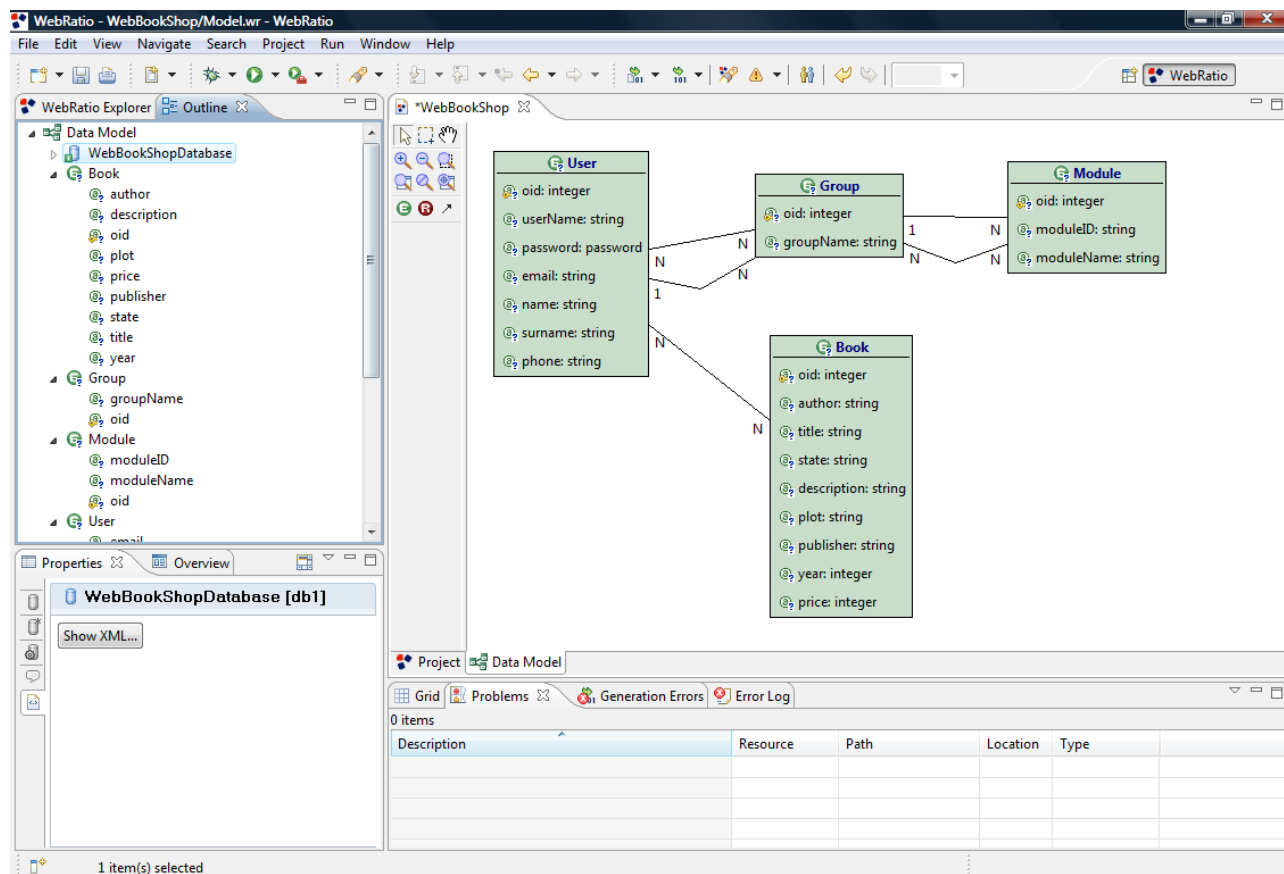
A WebRatio használatának alapjai

Mielőtt neki állnánk WebRatio-ban dolgozni, előbb végezzünk el egy apróbb beállítást. Erre azért van szükségünk, mivel a WebRatio kódgenerálás közben előfordulhat, hogy OutOfMemory hibával elszáll. Hogy ezt megelőzzük a WebRatio home könyvtárában lévő WebRatio.ini fájlban kell az egyes értékeket megváltoztatnunk. Kezdsnek célszerű, ha a MaxPermSize és az Xmx változók értékét megnöveljük. Ha ezek után mégis előfordul ilyen hiba, akkor figyeljük meg, hogy mire hivatkozik. Ha PermGen space error áll a hibaüzenetben, akkor a MaxPermSize méretét növeljük tovább, egyébként pedig az Xmx változónak adjunk nagyobb értéket.

A WebRatio elindítása után az Eclipse-ben programozókat ismerős kép fogadja. Aki még nem használt, az is nagyon hamar megszokja és meg kedveli ezt a környezetet. Első dolgunk egy új projekt létrehozása. Ezt a File → New → Web projekt menüpont segítségével tehetjük meg. A projekt név megadása után célszerű az adatbázis kapcsolatot is megadni, és leellenőrizni, hogy projektünk tud-e hozzá kapcsolódni. A WebRatio számos SQL alapú adatbázissal képes kapcsolatot létrehozni, így többek között MySQL, Oracle, PostgreSQL, Microsoft SQL Server közül választhatunk. Én a bemutatáshoz az Apache Derby adatbázis-kezelőt használtam. Azért ezt választottam, mivel ez a WebRatio-val feltelepülő Apache Tomcat része, így nincs szükség külön adatbázis-kezelő rendszer telepítésére és konfigurálására. Persze, ha már rendelkezünk jól beállított adatbázis-kezelő rendszerrel, vagy esetleg külön adatbázis szerver áll a rendelkezésünkre, akkor nyugodtan állítsuk be annak megfelelően a kapcsolatot. Miután ezzel készen vagyunk megnyílik előttünk a projekt.

A strukturális modell

A projektünk egy Data Model nézetet tartalmaz. Duplán kattintva erre a WebML-ből megismert strukturális modellel találjuk magunkat szembe. Itt három entitás és a köztük lévő összefüggés már fel van véve alapértelmezetten. Ezek az oldal felhasználóit és azok csoportjait reprezentálják, valamint hogy milyen modulokhoz férhetnek hozzá. Új entitás a baloldali menüből az E betűt tartalmazó zöld kört kiválasztva majd az ábrán egy üres helyre kattintva adható a modellhez. Adjunk a modellhez egy új entitást Book néven. A név megadása után, az entitáson történő jobb gombos kattintással előhívható annak gyorsmenüje, ahol az entitáshoz tartozó funkciókat érjük el. Itt van lehetőségünk attribútumok valamint összefüggések hozzáadására is. Attribútum felvételéhez annak nevét és típusát kell megadnunk. Amíg nem ismerjük a WebRatio által használt típusokat, addig célszerű a név megadása után az ablak bal alsó részében található Properties fülön beállítani a megfelelő adattípust. Vegyünk fel néhány attribútumot az entitáshoz, mint például cím, szerző, kiadás éve, stb. Két entitás közötti összefüggés felvételéhez a baloldali menüből az R betűt tartalmazó piros kört kell választani, majd először a forrás entitásra végül a cél entitásra kell kattintanunk. Az összefüggést szimbolizáló vonalra kattintva a Properties fülön beállítható annak neve, számossága és egyéb tulajdonságai. Ha elkészültünk a strukturális modellel, akkor a következő lépés az adatbázissal való kapcsolat élesztése és az adattáblák létrehozása. A képernyő bal felső részében található Outline fülön megtaláljuk a projekt elején létrehozott adatbázis kapcsolatunkat. Az adatbázishoz való kapcsolódáshoz kattintsunk a jobb gombbal erre, majd a helyi menüből válasszuk ki a Refresh menüpontot. A WebRatio ezután kapcsolódik az adatbázishoz, mely ha sikeres, akkor a kapcsolatot mutató henger bal alsó sarkában egy zöld jelzés lesz látható. Amennyiben valamilyen ok miatt nem volt sikeres a kapcsolódás, akkor egy piros jelzés lesz a bal alsó sarkon. Ebben az esetben több információt kaphatunk a hiba okáról, ha a sor elején található háromszögre kattintva lenyitjuk az adatbázis kapcsolatot és az Error feliratra kattintunk.



4. ábra
A strukturális modell kinézete a WebRatióban

Miután sikeresen kapcsolódtunk az adatbázishoz következő lépésként az előbb létrehozott modell alapján el kell készíteni az adattáblákat. Ezt a már előbb említett helyi menü Synchronize paranccsal tehetjük meg. Ha még nem választottuk ki, hogy melyik adatbázis schema-t akarjuk használni, akkor egy ablakban most ki kell választanunk azt. Ezután a Database Synchronizer ablak nyílik meg előttünk. Itt láthatjuk, hogy melyek azok a változások amelyeket a strukturális modellben végeztünk, de az adatbázisba még nem vittük fel. Válasszuk ki mind az Entities, mind a Relationship szöveget és a helyi menüjükből válasszuk ki az Export to Database menüpontot. Így minden változást érvényesíteni fogunk az adatbázisban. Miután ezzel megvagyunk és tovább lépünk, akkor a következő képernyőn látjuk, hogy az adatbázisban milyen SQL utasítások fognak végrehajtódni. Ezek itt nem módosíthatóak, ezért ha úgy érezzük, hogy valamit elrontottunk, akkor azt csak a strukturális modellen javíthatjuk ki. Persze dönthetünk úgyis, hogy nem hajtjuk végre most az utasításokat, hanem lehetőségünk van arra is, hogy csak elmentjük egy SQL script fájlba és

később mi magunk futtatjuk le. Így olyan esetekre is van megoldásunk, ha még nem vagyunk közvetlen kapcsolatban az éles rendszerrel, hanem csak a helyi gépünkön tervezünk, és az általunk végzett módosítások nagyon gyorsan és egyszerűen érvényesíthetőek a központi rendszeren is. Ezzel már el is készítettük a webalkalmazás strukturális modelljét. A Data Model nézet elhagyásához válasszuk az oldal alsó részén található Project fület.

A hypertext modell

A hypertext modell kialakításához létre kell hoznunk egy új Site View-t. Ezt a helyi menü Add Site View parancsával tehetjük meg. A név megadása után meg is nyílik a Site View. A Data Model nézethez hasonlóan itt is a bal oldalt található az eszköztár, amiben megtalálható minden olyan egység, amire egy oldal összeállításához szükségünk lesz. Egy oldal létrehozásához először válasszuk ki a Page ikont az eszköztárból, majd kattintsunk a szerkesztő területen valahova. Adjuk meg az oldal nevét, majd a fenti menüsorban válasszuk ki a Generate Full Web Project parancsot, hogy webalkalmazásunkat létrehozzuk. Ha valamit változtatunk az alkalmazásban, akkor ezt az utóbbi műveletet mindig el kell végezni, hogy az elvégzett változtatások életbe lépjenek. Ha a WebRatiohoz tartozó Apache Tomcat webszervert használjuk, akkor elég ha csak arról gondoskodunk, hogy elindítsuk a szervert, a többi a WebRatio maga elintézi. Ha nem lokálisan használjuk a Tomcat-et, vagy más webszervert használunk, akkor a telepítést magunk kell megoldani. A WebRatio az elkészített forrásokat a telepítési könyvtár \Tomcat\webapps könyvtárába helyezi el. Az oldal megtekintéséhez használhatunk bármilyen böngészőt, legegyszerűbb, ha csak annyit teszünk, hogy a helyi menüben kiválasztjuk az Open in Browser parancsot és a WebRatio-n belül nézzük az elkészült oldalt. Ennyit a WebRatio alapjairól, a következőkben egy konkrét alkalmazás néhány fontosabb elemének elkészítését mutatom be.

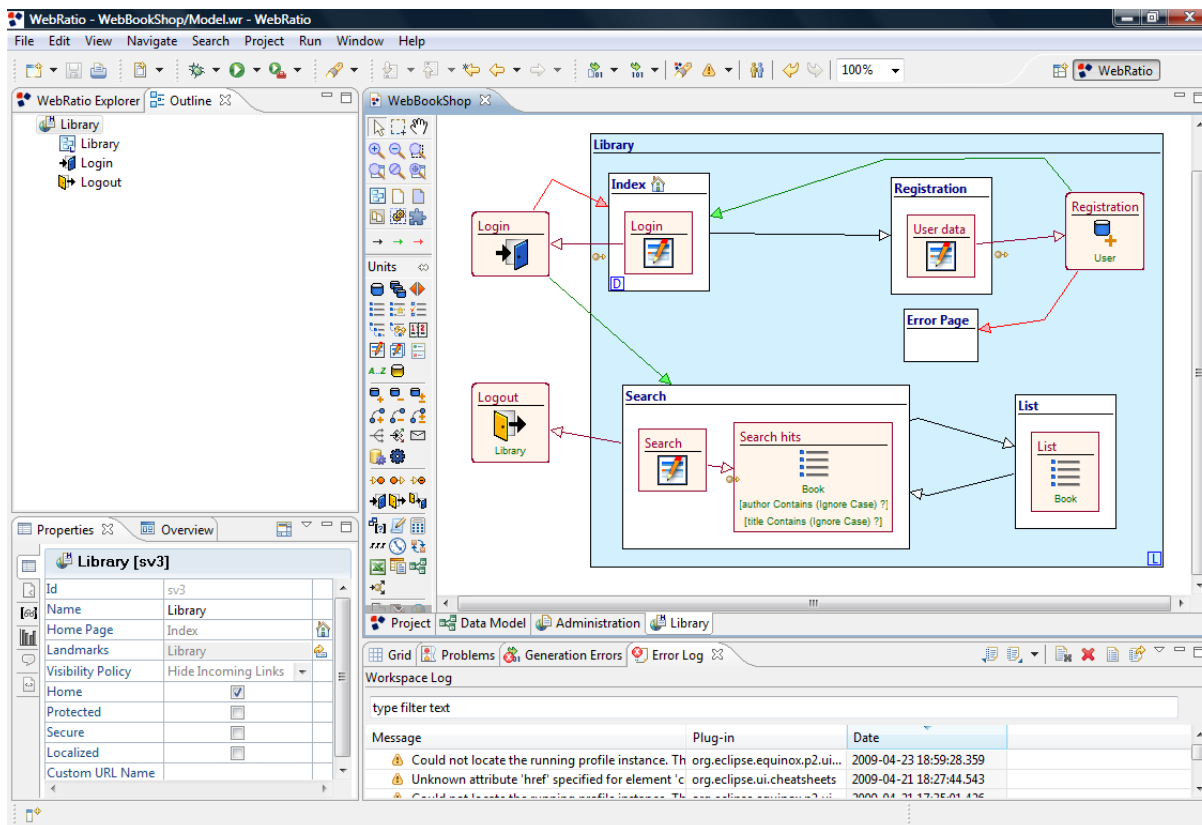
A Webkönyvtár létrehozásának főbb mozzanatai

Az alkalmazásunk esetében két Site View-ot kell felvenni, hogy a felhasználók könyv rendeléseitől biztosítsuk a könyvek, felhasználók adminisztrációjának szeparáltságát. Az elsőnek adjuk az Administration nevet, a másodiknak pedig a Library nevet. A könyv foglalással kapcsolatos oldalakat, funkciókat a Library nevű Site View-hoz vesszük fel, az

adminisztrációval kapcsolatos oldalakat pedig az Administration nevű Site View-hoz. Ügyeljünk arra, hogy a Library Site View Home tulajdonsága be legyen pipálva, így az lesz webalkalmazásunk kezdő lapja is. Vegyünk fel mind a két Site View-ra egy területet (area), ebbe fogjuk rakni az egyes oldalakat és egységeket. Alapértelmezetten a területek Landmark tulajdonsága bepipált állapotban van, ami azt jelenti, hogy minden oldalon van egy link ezen területek alapértelmezett oldalához. Ez nagyon hasznos lehet, ugyanis nem kell minden oldalból egy linket húznunk a kezdő oldalra, amelynek az eredménye nagyon sok oldal esetén nagymértékben csökkenti az ábra átláthatóságát. Mivel mind a két területen olyan műveletek lesznek, amelyeket csak a regisztrált felhasználók számára engedélyezünk, ezért szükség van arra, hogy az egyszerű látogatók számára elérhetatlenné tegyük az oldalakat, számukra az egyetlen elérhető funkció a regisztráció lesz. Ahhoz, hogy elérjük, a WebRatio a Login egységet nyújtja segítségül, mellyel felhasználói jogosultságokat ellenőrizhetünk. A Login egységet az Entry egységgel együtt kell használnunk, mivel ez fogja biztosítani, hogy a felhasználótól bekérjük annak felhasználói nevét és jelszavát. Helyezzünk egy Entry egységet a kezdőoldalra, majd vegyünk fel hozzá két mezőt, egyet a felhasználói névnek, egyet pedig a jelszónak. Gondoskodjunk róla, hogy a jelszó mező típusa password legyen, így az ilyenkor szokásos pöttyök jelennek meg a beírt karakterek helyett. Ezt a password mező Properties ablakában tehetjük meg, a Type tulajdonság password értékére állítva. Miután ezzel megvagyunk vegyünk fel egy Login egységet, amit a területen kívülre helyezünk el és húzzunk egy linket az Entry egységtől a Login egységhez. A linknek adjuk a Login nevet. Célszerű minden egységnek, oldálnak, linknek beszédes nevet adni, ugyanis ezekkel a nevekkal fog szerepelni a webalkalmazásunkban is. Így az előbb Login-nak elnevezett link az alkalmazásban úgy fog szerepelni, hogy a link felirata Login lesz. Miután ezzel megvagyunk a hozzá tartozó Properties fülön kattintsunk a Coupling tulajdonság sor végén lévő ikonra. A megjelenő Parameters Coupling ablakban tudjuk beállítani, hogy a kiindulási egységből átadott paraméterek a cél egység mely paramétereinek feleljenek meg. Amennyiben egymásnak egyértelműen megfeleltethető neveket adtunk meg, akkor használhatjuk az Automatic Coupling funkciót is. Vegyük ki a pipát az Enable Default Coupling előtti négyzetből, majd párosítsuk a paramétereket és kattintsunk az OK gombra. A Login egységhez még egy kötelező elem tartozik, ez pedig a KO Link, ezzel tudjuk megmondani, hogy sikertelen belépést követően melyik oldal jelenjen meg. Erre vehetünk fel egy külön hiba oldalt is, de mivel egy egyszerű bejelentkezésről van szó, amelynek hibája nincs

semmilyen hatással az adatbázis integritásra, ezért elég ha csak a kezdő oldalra irányítjuk vissza felhasználót, ahol újra próbálkozhat a belépéssel. A felhasználók regisztrációjához vegyünk fel egy oldalt, adjuk neki a Registration nevet. Vegyünk fel rá egy Entry egységet, majd vegyünk fel annyi mezőt hozzá, amennyi attribútuma van a User entitásnak a strukturális modellben. A könnyebbség kedvéért használjuk a Field Wizard opciót az Entry egység helyi menüjéből, így csak ki kell pipálnunk azokat az attribútumokat, amelyekhez szeretnénk mezőket felvenni, a többit elintézi a varázsló. Mivel egy felhasználói regisztrációhoz nélkülözhetetlen bizonyos adatok megadása, ezért ezekhez a mezőkhöz Validation Rule-okat veszünk fel, hogy biztosítsuk ezek kötelező jellegét. Ezt a megfelelő mező kiválasztásával, majd a helyi menüből az Add Validation Rule menüponttal tehetjük meg, majd ezt követően a Validation Rule Predicate tulajdonságát állítsuk Not Null-ra. Ezután vegyünk fel egy Create egységet a területre, majd pedig húzzunk egy linket az Entry egység és a Create egység között. Állítsuk be a paraméterek párosítását, majd húzzunk egy OK linket a főoldalra. Vegyünk még fel egy oldalt Error Page néven, és húzzunk a Create egységtől egy KO linket hozzá. Ezzel el is készítettük a felhasználók számára a regisztráció funkciót. Miután elkészítettük a bejelentkezés és regisztráció funkciókat, a könyvkeresés funkciót kell megvalósítani. Vegyünk fel egy Paget a területre Search néven. Erre az oldalra is felkerül egy Entry egység, amelyben a keresési paramétereket fogjuk bekérni. Vegyünk fel két mezőt hozzá, az egyiket szerző alapú keresésre, a másikat cím szerinti keresésre fogjuk használni. Helyezzünk el ezen az oldalon egy Index egységet, entitásához a Book entitást társítsuk. Ezt a Properties fülön lévő Entity tulajdonságának sor végén lévő Select gomb segítségével tehetjük meg. A megnyíló ablakban válasszuk ki a Book entitást. Vegyünk fel két Attributes Condition-t is az Index egységünkhöz, amelyet az egység helyi menüjében tehetünk meg, az Add Attributes Condition parancs kiválasztásával. Az egyik Attributes Condition-hoz az author attribútumot rendeljük, a másikhoz pedig a title attribútumot, majd állítsuk be mindkettőnél a Contains (Ignore Case) értéket a Predicate tulajdonsághoz, továbbá győződjünk meg, hogy mind a két esetben a Boolean Operator értéke Or legyen. Ezekkel a tulajdonságokkal úgy állítottuk be a kereső működését, hogy szerzőre és címre lehet keresni és elég megadni az egyik keresési kritériumot. Az Index egységnek még állítsuk be, hogy mely attribútumokat szeretnénk ha megjelenítene, amit a Display Attributes tulajdonságnál tehetünk meg. Még egy linket kell húznunk az Entry és az Index között, ezen beállítani a paraméter párosítást és el is készültünk a keresés funkcióval. Helyezzünk még egy Logout egységet a területen kívülre és

húzzunk a Search oldalról egy linket hozzá, és már a kijelentkezést is megoldottuk.

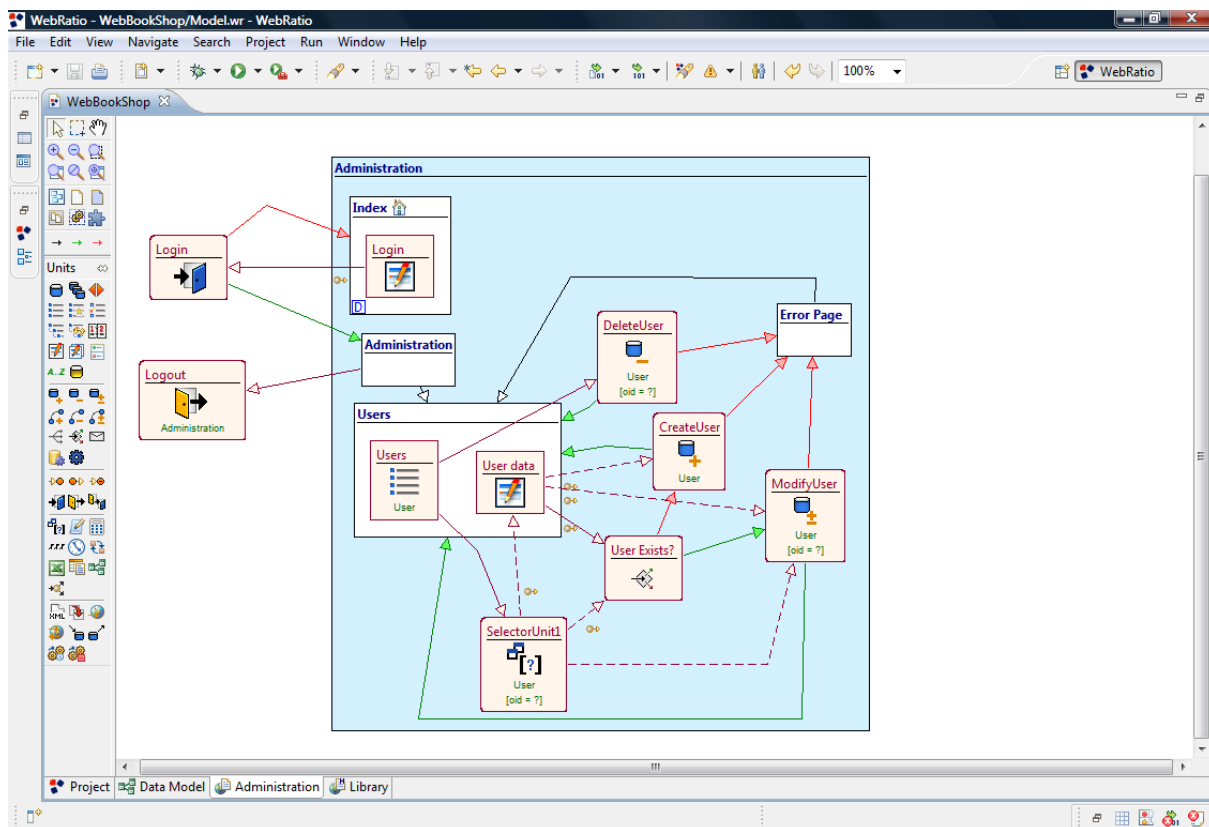


5. ábra

A Webkönyvtár publikus oldalának hypertext modellje a WebRatióban

Ezután következhet az adminisztrációs feladatok megvalósítása. Itt is kezdjük a dolgot egy terület és egy index oldal felvételével. Ezt követően a már előzőleg ismertetett bejelentkezést itt is biztosítjuk. Miután ezzel megvagyunk, két csoportra kell osztani a működést. Az első csoportba kerülnek a felhasználók adminisztrációjával kapcsolatos műveletek, a másikba pedig a könyvekkel kapcsolatos műveletek. Ezek a műveletek a felvétel, módosítás, törlés funkciók, melyeket mind a két csoportra megvalósítjuk. Mivel a két csoport egyes funkciói ugyanazok, ezért csak az egyik megvalósítását fogom leírni. Vegyünk fel egy oldalt a területre és húzzunk hozzá egy linket a kezdőoldaltól. Minden felhasználóval kapcsolatos funkció ezen az oldalon fog történni. Vegyünk fel egy Entry és egy Index egységet az oldalra. Állítsuk be az Entry egységet annak megfelelően, hogy a User entitás minden attribútumát meg lehessen adni benne, az Index egységet pedig úgy, hogy minden attribútumot megjelenítsen, persze ez alól kivétel a jelszó megjelenítése. A felhasználó felvétel funkció megvalósítása hasonló módon történik, mint a regisztrációnál, azonban itt azt

is figyelembe kell vennünk, hogy esetleg nem új felhasználót akarunk felvenni, hanem egy meglévő adatait módosítani. Ehhez ugyanúgy vegyük fel az egységeket, mint ahogy a regisztrációhoz felvettük. Vegyünk fel egy Selector egységet és hozzunk létre linket az Index egység és a Selector között. Ezzel a linkkel tudunk majd módosítani. Vegyünk fel egy Is Not Null egységet, ami arra fog szolgálni, hogy eldöntsük új felhasználóról van-e szó, vagy pedig egy meglévőt módosítunk. Ennek az egységnek a logikája az, hogyha a felvett feltétel igaz, akkor az OK linken folytatódik a működés, ellenkező esetben pedig a KO linken. A mi feltételünk a User entitáshoz rendelt oid-re vonatkozik. Ha már szerepel a táblában, akkor meglévő felhasználó módosításáról van szó, különben új hozzáadásáról. Mind az Entry egységből, mind a Selector egységből húzzunk egy linket az Is Not Null egységhez. Az Entry egység és a Create egység közötti link típusát állítsuk át transport-ra. Húzzunk két linket a Modify egységhez is, egyet az Entry egységből, egyet pedig a Selector egységből. Gondosan állítsuk be a paraméter párosítást a linkeknél. A Modify egységből húzzunk egy OK linket az oldalra, egy KO linket pedig az Error oldalra. Ahhoz hogy teljesen befejezzük a működési láncot, még el kell helyeznünk egy OK és egy KO linket az Is Not Null egységből. Az OK link a mi esetünkben a Modify egységre fog mutatni, a KO link pedig a Create egységre. Most már csak a törlést kell megvalósítani. Vegyünk fel egy Delete egységet és húzzunk hozzá egy linket az index egységből. Vegyünk fel egy OK linket az oldalhoz, egy KO linket az Error oldalhoz. Ezzel készen is vagyunk, a felhasználókkal kapcsolatos funkciókat sikeresen megvalósítottuk.



6. ábra
Az adminisztrációs felület hypertext modellje WebRatióban

Az egyes egységek és a közöttük lévő linkek elhelyezésére figyeljünk oda, mivel már az ilyen sok elemet tartalmazó modelleknél könnyen átláthatatlanná válik az ábránk. Ha a linkek keresztezését nem is tudjuk teljesen megszüntetni, akkor törekedjünk az egyértelműsége. Szerencsére a WebRatio a különböző link fajtákat más színnel jeleníti meg, ami szintén segíti a modell értelmezhetőségét.

A könyvek adminisztrációja ugyanezekből a funkciókból épül fel, így annak a megvalósítására nem térek ki. Lényegében csak az entitásokat kell kicserélni, hogy a User entitás helyett a könyv entitással dolgozzon, valamint az egyes egységeknek kell más nevet adni.

Az előbb leírtakból is látható, hogy milyen egyszerűen létrehozható egy webalkalmazás, könnyedén megoldhatóak a fejlesztés elején lévő olyan robot munkák, mint például az oldalak közötti navigáció megtervezése és megvalósítása. Ezek mellett még egy nagyon jól kidolgozott tervrajzot is kapunk, ami a későbbi munkánkat vagy az esetleges módosításokat is nagyban megkönnyíti.

A WebRatio által generált kód elemzése

Mint említettem a WebRatio az elkészített modellek alapján képes kódgenerálásra is. Ezek a generált kódok java nyelvűek, így tehát platform függetlenek és bármilyen java alapú alkalmazás szerveren működnek.

A generált kód két részre oszlik: a strukturális modellből java osztályok jönnek létre, hypertext modellből pedig jsp fájlok.

A példa kedvéért tekintsünk meg mind a két fajta generált kódból egy-egy részletet.

Java osztály:

```
package com.webratio.webapp;

public class User implements java.io.Serializable {
    /** Serial version identifier. */
    private static final long serialVersionUID = 1L;
    private java.lang.Integer _oid;
    private java.lang.String _userName;
    private java.lang.String _password;
    private java.lang.String _email;
    private java.lang.String _name;
    private java.lang.String _surname;
    private java.lang.String _phone;
    private java.util.Set _groups = new java.util.HashSet();
    private Group _defaultGroup;
    private java.util.Set _userToBook = new java.util.HashSet();
    private float _searchScore;
    public java.lang.Integer getOid() {
        return _oid;
    }
    public void setOid(java.lang.Integer _oid) {
        this._oid = _oid;
    }
    public java.lang.String getUserName() {
        return _userName;
    }
}
```

```

public void setUsername(java.lang.String _userName) {
    this._userName = _userName;
}
...
public String toString() {
    java.lang.StringBuffer sb = new java.lang.StringBuffer();
    sb.append(super.toString());
    int n = sb.length() - 1;
    if (sb.charAt(n) == ']') {
        sb.setCharAt(n, ',');
    } else {
        sb.append('[');
    }
    if (_oid != null)
        sb.append("oid=" + _oid + ',');
    if (_userName != null)
        sb.append("userName=" + _userName + ',');
    if (_password != null)
        sb.append("password=" + _password + ',');
    if (_email != null)
        sb.append("email=" + _email + ',');
    if (_name != null)
        sb.append("name=" + _name + ',');
    if (_surname != null)
        sb.append("surname=" + _surname + ',');
    if (_phone != null)
        sb.append("phone=" + _phone + ',');
    n = sb.length() - 1;
    if (sb.charAt(n) == ',') {
        sb.setCharAt(n, ']');
    } else if (sb.charAt(n) == '[') {
        sb.deleteCharAt(n);
    }
    return sb.toString();
}
public boolean equals(java.lang.Object obj) {
    if (!(obj instanceof User)) {
        return false;
    }
}

```

```

        User otherUser = (User) obj;
        java.lang.Object key = null;
        java.lang.Object otherKey = null;
        key = this.getOid();
        otherKey = otherUser.getOid();
        if (key == null) {
            if (otherKey != null) {
                return false;
            }
        } else {
            if (otherKey == null) {
                return false;
            } else if (!key.equals(otherKey)) {
                return false;
            }
        }
        return true;
    }

    public int hashCode() {
        int hashCode = 0;
        java.lang.Object key = null;
        key = this.getOid();
        if (key != null) {
            hashCode |= key.hashCode();
        }
        return hashCode;
    }
}

```

Látható, hogy minden egyes attribútumnak, amit felvettünk az entitáshoz, létre jött egy-egy adattag. Ezekben az adattagokban tárolódnak majd az egyedek tulajdonsága és így tárolódnak az adatbázisban. Az egyedek tárolása a Hibernate keretrendszer segítségével valósul meg, ezért az entitások alapján generált szabvány java osztályokon kívül még a szükséges hbm.xml fájlokat is elkészültek. Az attribútumok mellett az összefüggéseket is megtaláljuk adattagok formájában. Ezeket vagy a megfelelő típusú java osztályok, amelyek szintén generált java osztályok, vagy pedig halmazok reprezentálják. Hogy egy összefüggés milyen formában van ábrázolva, az az adott összefüggés számosságától függ. Ha a számosság

maximum egy lehet, akkor egyszerű java osztály az adattag típusa, ellenkező esetben pedig halmaz. Minden adattaghoz elkészül egy getter és setter metódus pár is, ezek teszik a Hibernate számára, hogy szerializálni tudja az egyedet és eltárolhassa az adatbázisban, majd később az adatbázisból való kiolvasáskor képes legyen azt deszerializálni és újra létre hozza belőle a java objektumot. A szokásos toString metódus mellé létrehoz egy equals metódust is az egyedek összehasonlíthatósága miatt és egy hashCode metódus, amivel az egyedet igazoló hasht hozhatunk létre.

JSP fájl:

A JSP fájlok viszonylag nagy mérete miatt csak a néhány részlet bemutatására térek ki. Első példaként tekintsük a keresési oldal forrásából a bejegyzési egység kódját:

```
<div class="plain ">
  <div class="plain EntryUnit">
    <table>
      <tr>
        <td colspan="3" class="error">
          <html:errors property="enu2" />
        </td>
      </tr>
      <tr class="row">
        <th valign="middle" class="header">author</th>
        <td valign="middle" nowrap="nowrap" class="value">
          <html:text onkeypress="if(event.keyCode==13)
            {clickButton('ln9'); return false;}" size="25" styleId="fld3"
            styleClass="field" property="fld3" readonly="false"/>
        </td>
        <td class="error">
          <span class="error"><html:errors property="fld3"/></span>
        </td>
      </tr>
      <tr class="row">
        <th valign="middle" class="header">title</th>
        <td valign="middle" nowrap="nowrap" class="value">
          <html:text onkeypress="if(event.keyCode==13)
            {clickButton('ln9'); return false;}" size="25" styleId="fld4"
            styleClass="field" property="fld4" readonly="false"/>
        </td>
      </tr>
    </table>
  </div>
</div>
```

```

        </td>
        <td class="error">
        <span class=" error"><html:errors property="fld4"/></span>
        </td>
    </tr>
    <tr>
        <td colspan="3">
        <table>
        <tr>
        <td>
        <input title="Search" onclick="$
        ('page3FormBean').target='_self'" class=" button"
        id="button#ln9" name="button#ln9" type="submit" value="Search">
        </td>
        </tr>
        </table>
        </td>
        </tr>
    </table>
</div>
</div>

```

A kódból látható, hogyan épül fel az általunk igen egyszerűen elhelyezett bejegyzési egység minden mezője, tulajdonsága. A mezők elrendezése táblázattal van megoldva, amelyet egy div egység foglal magába, így egy egységként kezelhető a bejegyzési egység. Ennek az egyszerű egységnek az elkészült kódja, néhány kivételtől eltekintve, szinte tisztán statikus html elemeket tartalmaz.

Sajnos látható az is, hogy hiába adtunk mindennek értelmes elnevezést, bizonyos nevek és hivatkozások a program által megadottak maradtak. Ilyen például a bejegyzési egység neve is, amelynek mi hiába adtuk a Search nevet, a generált kódban mégis a WebRatio által adott enu2 azonosítja ezt az egységet. Bár jól lehet, ezzel elkerüli, hogy ugyanazt az elnevezést használja azonosítóként két egységre, de ha mi már kód szinten akarunk módosítani valamit, akkor ezzel nagyon megnehezíti a dolgunkat.

Második példának pedig tekintsük meg a keresési találat megjelenítésére szolgáló kódrészletet:

```

<c:if test="${not(empty inul) and (inul.dataSize gt 0)}">
<table width="100%">
    <tr>
    <td>
    <h2>Search hits</h2>
    </td>
    </tr>
    <tr>
    <td>
    <div class="plain ">
        <div class="plain IndexUnit">
            <table border="0" cellspacing="1" cellpadding="2"
            id="inul_1_sortable">
                <tr>
                <th class="header"></th>
                <th nowrap="nowrap" class="header">author</th>
                <th nowrap="nowrap" class="header">title</th>
                <th nowrap="nowrap" class="header">state</th>
                <th nowrap="nowrap" class="header">description</th>
                <th nowrap="nowrap" class="header">plot</th>
                <th nowrap="nowrap" class="header">publisher</th>
                <th nowrap="nowrap" class="header">year</th>
                </tr>
                <c:forEach var="current" varStatus="status" items="$
                {inul.data}">
                <c:set var="index" value="${status.index}"/>
                <tr class="row<c:if test="${inul.currentIndex eq
                index}">Current</c:if>">
                <td class=" value<c:if test="${inul.currentIndex eq
                index}">Current</c:if>">
                Current</c:if>"
                border="0" alt=""/>
                </td>
                <!-- not linked attribute -->
                <td class=" value<c:if test="${inul.currentIndex eq
                index}">Current</c:if> string">
                <c:out value="${current.author}"/>
                </td>

```

```

<!-- not linked attribute -->
<td class=" value<c:if test="\${inul.currentIndex eq
index}">Current</c:if> string">
<c:out value="\${current.title}"/>
</td>
<!-- not linked attribute -->
<td class=" value<c:if test="\${inul.currentIndex eq
index}">Current</c:if> string">
<c:out value="\${current.state}"/>
</td>
<!-- not linked attribute -->
<td class=" value<c:if test="\${inul.currentIndex eq
index}">Current</c:if> string">
<c:out value="\${current.description}"/>
</td>
<!-- not linked attribute -->
<td class=" value<c:if test="\${inul.currentIndex eq
index}">Current</c:if> string">
<c:out value="\${current.plot}"/>
</td>
<!-- not linked attribute -->
<td class=" value<c:if test="\${inul.currentIndex eq
index}">Current</c:if> string">
<c:out value="\${current.publisher}"/>
</td>
<!-- not linked attribute -->
<td class=" value<c:if test="\${inul.currentIndex eq
index}">Current</c:if> integer">
<fmt:formatNumber value="\${current.year}" pattern="\$
{integerPattern}"/>
</td>
</tr>
</c:forEach>
</table>
</div>
</div>
</td>
</tr>
</table>

```

</c:if>

A kódrészlet első sorából látszik, hogy a kódgenerátor a JET 2⁹-es vezérlő tageket használja. Ez az egyik elterjedt kódgenerátor az EMF¹⁰ keretein belül. A JET 2 olyan template technológiát használ, amelynek szintaxisa nagyon hasonlít a JSP szintaxisához. Minden JET 2-es vezérlő tagekhez tartozik egy a vezérlő taget lezáró pár is. A tagek a c: kódrészlettel kezdődnek, ezzel jelezve, hogy JET 2-es vezérlő tagról van szó, majd pedig a vezérlő parancsok találhatók.

Az egész találati lista megjelenítése feltételek segítségével történik. Maga a kódrészlet is egy vizsgálattal kezdődik. Ha a keresésünk üres találati listát ad vissza, abban az esetben még maga a táblázat se jön létre. Bentebb látható, hogy a kód generálása során olyan vizsgálatok is létrejöttek, amelyek szükségességére talán nem is gondolnánk. Az egy egyedhez tartozó összes attribútumra történik egy vizsgálat, amely eldönti, hogy az egyed attribútumának van-e valós értéke. Ezzel is növelve a létre jövő táblázat átláthatóságát, amennyiben az hiányos adatokat tartalmaz. A találati listát egy halmazban tárolja az alkalmazás, és a bejárás megvalósítása a foreach segítségével valósul meg.

Látható, hogy a kódgenerátor nem csak legenerálja a kódot számunkra, hanem azt jól átlátható formába is rakja, megkönnyítve az emberi feldolgozhatóságot. Bár az egységek azonosítására szolgáló nevek zavaróan hatnak, de egy idő után azzal is meg tud barátkozni az ember, mivel ezek az elnevezések eléggé logikusan következnek az egységek típusából.

⁹ Java Emitter Templates

¹⁰ Eclipse Modelling Framework

Összefoglalás

Mint ez előbb leírtakból is nagyon jól látszik, a WebML segítségével képesek vagyunk a teljes életciklust végigvezetni a tervezéstől egészen a megvalósításig. A WebML jól kidolgozott modelleket nyújt a tervezés megkönnyítéséhez, melyekkel az olyan apró, de fontos dolgok, mint például a navigáció megtervezése is nagyon egyszerűen megvalósítható mellőzve az unalmas, ismétlődő robotszerű munkákat. A strukturális modellben nagyon könnyen olyan ábrát hozhatunk létre, amely nem csak a táblák áttekinthetőségében segít, de a WebRatio segítségével az egész adatstruktúra pillanatok alatt felépíthető az adatbázisban is. A hypertext modell nem csak hogy az alkalmazás során felhasznált elemeket mutatja be, hanem az egységek kapcsolatát szimbolizáló linkek segítségével a közöttük folyó adatáramlás is nagyon egyszerűen jellemezhető vele.

Eleinte gondot jelenthet meghatározni az egyes egységek fontosságát és használatának helyét, de könnyen megtanulható és kezelhető elemekről van szó, így néhány bevezető alkalmazás elkészítése után már bátran neki állhatunk élesben használni ezt a modellező nyelvet. Mivel a strukturális modell E-R és UML alapokra épít, ezért annak a megismerése nem okoz nagy gondot azoknak, akik ismerik ezeknek az elemeit.

A WebRatio, mint a WebML modellező nyelvet és a hozzá tartozó tervező módszert megvalósító eszköz, nem csak tervezésre használható, hanem a beépített kódgenerátorának köszönhetően az elkészült modellek legenerálására is, így meggyorsítva nem csak a tervezést, hanem az implementálást is.

A generált kódokkal szintén nagyon sok idő meg tudunk spórolni, hiszen nem kell minden egyes entitáshoz elkészíteni a java osztályt, ráadásul biztos, hogy nem vétünk benne semmi olyan hibát, ami hátráltatná a fejlesztést. Az elkészített kódok alapján könnyen tovább is fejleszthető az alkalmazásunk ezzel is gyorsítva a fejlesztést.

Mindazon által szem előtt kell tartanunk azt a fontos tényt, hogy a WebML webalkalmazások tervezésére szolgáló eszköz, ezért ne használjuk személyes honlapok, vagy kis alkalmazások tervezésére, ugyanis nem kapunk annyival jobb eredményt, mint amennyivel több energiát fektetünk a tervezésbe és az elemzésbe.

Irodalomjegyzék

- [1] <http://wish.hu/cikkek/web20.html>
- [2] <http://www.w3.org/History/1989/proposal.html>
- [3] <http://webml.org/webml/page1.do>
- [4] <http://www.webratio.com/portal/homePage.do>