Bevezetés a WebObjects alapú webalkalmazás-fejlesztésbe

Bevezető

Idén ünnepli tizedik születésnapját a WebObjects, amely a hazai webalkalmazás-fejlesztők körében viszonylag kevésbé ismert. Nem megérdemelten, mert igen erőteljes és hatékony, Java technológiára épülő alkalmazásfejlesztő környezet. Ebben a cikkben röviden bemutatjuk a WebObjects történetét, felhasználási lehetőségeit és felépítését.

A WebObjects története

A WebObjects története az 1990-es évek közepére nyúlik vissza, amikor a NeXT Computer által fejlesztett NeXTSTEP operációs rendszer a fénykorát élte. A NeXTSTEP-hez kapcsolódó fejlesztői eszközök segítségével gyorsan és egyszerűen lehetett grafikus felhasználói felülettel rendelkező alkalmazásokat készíteni. Ez hamar felkeltette a nagyvállalati szektor érdeklődését, különösen az adatbázisokhoz kapcsolódó kliens-szerver alkalmazások fejlesztése kapcsán. Ezt a piacot célozta meg a NeXT, amikor megalkotta az Enterprise Objects néven ismert technológiáját, amely az első kereskedelmileg sikeres objektum-relációs leképezést biztosító keretrendszer volt. Az Enterprise Objects segítségével az adatbázis hozzáférés különböző feladatait objektumokra lehet rábízni. 1995-ben a NeXT felismerte, hogy várhatóan a web lesz a kliens-szerver megoldások domináns közege, így mérnökei az Enterprise Objects objektum-orientált szemléletének mintáján és sikerén felbuzdulva kifejlesztettek egy keretrendszert, amely a HTTP kéréseket és a dinamikus HTML generálást kezeli. Erre a keretrendszer gyűjteményre alapozva 1996 januárjában a NeXT bemutatta a WebObjects 1.0-ás verzióját, a világ első objektumorientált webalkalmazás-szerverét.

1997-ben az Apple Computer felvásárolta a NeXT-et, elsősorban a UNIX alapú operációs rendszeréért, amely az Apple operációs rendszerének, a Mac OS X-nek az alapját biztosította. A bekebelezés során természetesen a WebObjects és annak fejlesztőgárdája is átkerült az Apple szárnyai alá. Az Apple internetes megjelenésében központi szerepet kapott a WebObjects, hiszen az Apple online boltja, valamint a .Mac névre hallgató portálja is ebben készült el. A WebObjects értékesítése azonban nem volt igazán sikeres. A közvetlen értékesítés, illetve a közel 50.000 dolláros licensz ár továbbra is csak a nagyvállalati ügyfelek körét célozta meg. A WebObjects-et például az AT&T, a BBC, az Adobe, a Fleet Bank, a Standard & Poor, a Dell, a Disney és a Deutsche Bank használta.

2000 májusában, az 5.0-ás verzió bejelentésekor az Apple stratégiát váltott azzal, hogy a WebObjects árát drasztikusan csökkentette (50.000 dollárról 699 dollárra!), valamint hagyományos módon megvásárolható, dobozos termékként kínálta tovább. Ezzel a lépéssel megnyitotta az utat a fejlesztők, a kis- és közepes vállalatok, és az oktatási intézmények felé. 2005-ben ezt még tovább erősítette azzal, hogy a WebObjects 5.3-as verzióját szabadon letölthetővé tette az Apple fejlesztői honlapjáról. Ma talán a legismertebb WebObjects alapú alkalmazás az iTunes Music Store, a világ legnagyobb online zeneáruháza.

Mire jó a WebObjects?

Történetéből következően a WebObjects legnépszerűbb felhasználási területe az adatbázis hátteret igénylő internetes/intranetes alkalmazások, ahol az alkalmazás kliens oldali felhasználói felületét dinamikusan generált HTML biztosítja. Ez persze nem jelenti azt, hogy az alkalmazás csak HTML-t szolgálhat ki. Dinamikusan generálhat például XML, PDF, SVG, WML vagy SMIL dokumentumokat is. Ha pedig a HTML-nél dinamikusabb felhasználói felületre van szükség, akkor képes böngészőn belül futó Java Applet-et vagy asztali Java kliens alkalmazást is előállítani. A WebObjects viszonylag új alkalmazási területe a SOAP alapú webszolgáltatások (web services) biztosítása. Fontos megjegyezni, hogy elvileg ugyanaz a WebObjects alkalmazás képes ezeket a típusokat párhuzamosan is kiszolgálni. A továbbiakban – az egyszerűség kedvéért – csak a HTML felülettel rendelkező alkalmazásokkal foglalkozunk.

A WebObjects felépítése

A WebObjects mint termék három fő komponensre bontható. Az első komponenst a már említett objektumorientált keretrendszerek (frameworks) alkotják. A másodikat az ezekre épülő fejlesztői eszközök, míg a harmadikat a kifejlesztett alkalmazások futtatási (deployment) infrastruktúrája adja. Ha a fejlesztői eszközök felől tekintünk rá, akkor azt mondhatjuk, hogy a WebObjects egy alkalmazás fejlesztői környezet. Ha viszont a futtatási környezet felől, akkor egy alkalmazás szerver.

A WebObjects ma már teljes egészében a Java-ra épül. Ez azt jelenti, hogy a keretrendszerek osztályai Java-ban készültek, a futtatási környezet JVM (Java Virtual Machine) meglétét feltételezi, és természetesen az alkalmazásunk egyedi kódját is Java-ban kell megírnunk. A Java elterjedtsége és nyitottsága a biztosíték arra, hogy alkalmazásunk egyedi igényeknek is megfeleljen, hiszen segítségül hívhatjuk a különböző kereskedelmi és nyílt forráskódú Java-s kiegészítéseket. A WebObjects könnyen együttműködik más, Java alapú futtatási megoldásokkal is, így például EJB konténerekkel, szervletekkel, webszolgáltatásokkal.

A következőkben röviden áttekintjük a főbb keretrendszereket és a hozzájuk kapcsolódó fejlesztői eszközöket.

Az Enterprise Objects Framework (EOF) és az EOModeler

Ezt az objektumorientált keretrendszert a WebObjects koronaékszerének is szokták nevezni. Önmagában az EOF ismertetése is meghaladná ennek a cikknek a terjedelmi korlátait, így csak ízelítőt adunk belőle. Az EOF fő feladata az adathozzáférés menedzselése valamilyen külső

adatforráshoz kapcsolódva. Az adatforrás jellemzően egy relációs adatbázis, de lehet LDAP adattár is. Az EOF az adatbázisban tárolt adatokat úgynevezett Enterprise Objects (rövidítve EO) objektumokba képezi le. Ezt a technikát szokás objektum-relációs leképezésnek hívni. Az általunk írt WebObjects alkalmazás objektumorientált módon, közvetlenül ezeket az EOF által biztosított EO objektumokat használja, ha adattárolásra illetve adat visszakeresésre van szüksége. A WebObjects terminológiáját használva, az EOF egy réteg (layer) az alkalmazás és a relációs adatbázis (vagy egyéb adatforrás) között. Az elválasztás révén az alkalmazás nem áll közvetlen kapcsolatban az adatforrással. Relációs adatbázis esetén ez például azt jelenti, hogy az alkalmazás kódjába nem kell (és nem is ajánlott) SQL parancsokat írni.

Ahhoz, hogy az objektum-relációs leképezés megfelelően működjön, definiálnunk kell az EOF számára egy „térképet", ami megmondja, hogy az adatforrás adatai mely EO objektumokba kerüljenek majd bele. A WebObjects-ben ezt a térképet EOModel-nek hívjuk, és az EOModeler nevű fejlesztői eszközzel készítjük el. Az EOModel fájl több adatot is tartalmaz. Többek közt ebben van, hogy az adatforrás milyen URL címen érhető el. Relációs adatbázis esetén ez jellemzően egy JDBC (Java Database Connectivity) URL, hiszen a WebObjects JDBC kapcsolaton keresztül kommunikál az adatbázissal. Szinte az összes ismert adatbázishoz létezik JDBC illesztő, így ezek közül bármelyiket használhatjuk a WebObjects-es alkalmazásunk adatbázis hátteréül. Fontos megemlíteni, hogy az EOF párhuzamosan több, akár különböző gyártótól származó adatforrásból is képes dolgozni, és az objektum-relációs leképezést megvalósítani. Így elképzelhető, hogy az EO objektumunk bizonyos adatai az egyik adatforrásból, bizonyos adatai pedig másik adatforrásból származnak. Azt, hogy melyik honnan, sem az EO objektum, sem az alkalmazásunk nem tudja. Ez pusztán az EOF felelőssége.

Az EOModel-ben megtalálható az adatforrás struktúrájának összes jellemzője is. Így például a relációs adatbázis tábláinak, oszlopainak neve, valamint az egyes oszlopok SQL adattípusa. Ha az adatbázis struktúra már létezik – azaz nem most kerül kialakításra –, akkor az EOModeler képes a struktúrát „visszafejteni", és ez alapján az EOModel-t automatikusan létrehozni.

Végezetül az EOModel tartalmazza a leképezés tényleges térképét is, vagyis azt, hogy az adatbázis oszlopai mely EO objektumok mely változóihoz vannak hozzárendelve. Az EOModeler az így felállított modell alapján legenerálja a relációs adatbázist, és az EO objektumok Java kódjának alapját.

Felvetődhet a kérdés, hogy miért jó az, ha az adatforrásunk adatai objektumokban vannak elhelyezve, majd az alkalmazásunkból ezeket az EO objektumokat használjuk, ahelyett, hogy közvetlenül az adatforrást használnánk? A WebObjects egyik fő koncepciója, hogy élesen szétválasztja az alkalmazás megjelenését, az alkalmazás funkcionalitását (logikáját), és az alkalmazás adathozzáférését. Sőt, igyekszik elkülöníteni egymástól az alkalmazás szakterületének megfelelő üzleti logikát (business logic) az alkalmazás egyéb logikájától (application logic). Ez a gondolat részben a SmallTalk Model-View-Controller paradigmájában, részben a NeXTSTEP objektumorientált fejlesztői környezetében gyökerezik. Az adathozzáférést az EOF segítségével már elkülönítettük, lényegében adatbázis-függetlenné tettük az alkalmazásunkat. Az üzleti logikát az EO objektumok kódjába helyezzük el, így az EO-k már nem pusztán adathordozók, hanem önálló viselkedéssel rendelkező objektumok lesznek. Az üzleti logika ilyesfajta elhelyezéséből több előnyünk is származik. Az elkészült EO-kat például újra felhasználhatjuk a szakterületnek megfelelő összes alkalmazásunkban. Adott feladathoz például készíthetünk egy HTML felületű alkalmazást, és egy asztali Java adminisztrációs alkalmazást. Természetesen mindkettőben ugyanazon EO-kat használva. Ha az üzleti logikában később változtatni kell, akkor a karbantartás is könnyen elvégezhető, hiszen elég az adott EO-ban végrehajtani a változást, majd az összes őt használó alkalmazás automatikusan átveszi azt.

Érdekes, hogy az objektumorientált programozásban megszokott öröklődés az EO-k esetén is használható. Különböző lehetőségeink vannak az EO öröklődés modellezésére, köztük olyanra is, amikor az EO több adatbázistábla leképezéséből áll össze.

Adatbázisok kezelésénél kiemelt fontosságú az elsődleges és az idegen kulcsok létrehozása és kezelése. A WebObjects esetén ezzel sem kell különösebben foglalkoznunk, hiszen az EOF ezeket a feladatokat automatikusan elvégzi helyettünk. Vagy magától generálja az elsődleges kulcsot, vagy valamilyen általunk megírt egyedi kódot, illetve tárolt eljárást hív meg az előállításához.

Az EOF saját tranzakció menedzsmenttel rendelkezik. Minden általunk aktuálisan használt EO a memóriában egy elkülönített területen, az úgynevezett editing contextben (szerkesztési környezetben) található. Az editing context felelőssége, hogy folyamatosan figyelje a benne lévő EO objektumok állapotváltozásait, az új EO-k létrejöttét, a meglévők módosításait, törlését. Ha végeztünk a változtatásokkal, akkor kiadhatunk az editing context-nek egy mentés parancsot. Az EOF ekkor megvizsgálja, hogy milyen változtatások történtek az editing context-ben, és dinamikusan generált SQL parancsok segítségével átvezeti azokat az adatbázisban. Az átvezetést egyetlen adatbázis tranzakció keretében hajtja végre, így ha a tranzakció sikertelen lenne, az adatbázis nem kerül inkonzisztens állapotba. Az editing context-ben visszavonás lehetőség is van. A módosítások lépésenként visszagörgethetők a szerkesztési verem kiindulási állapotáig.

A memóriában tárolt EO objektumok bizonyos mértékben gyorsítják is az alkalmazás működését, hiszen jóval gyorsabb adatelérést biztosítanak, mint a közvetlen adatbázis hozzáférés. Néhány probléma azonban felmerül, amire a WebObjects elegáns megoldásokat ad. Ha egy objektumot leképezünk az adatbázisból, akkor ahhoz kapcsolatokon keresztül további objektumok tartozhatnak. Ezeket szintén be kell olvasni a memóriába. A gond csak ott jelentkezik, hogy ezeknek a kapcsolódó objektumoknak további kapcsolatai lehetnek, így könnyen eljuthatunk egy olyan szituációba, hogy az egész adatbázis a memóriába kerül, egy EO objektum beolvasása miatt! Ezt az EOF úgy oldja meg, hogy a kapcsolódó objektumnak csak az üres vázát hozza létre, de nem tölti fel adatokkal. Csak abban a pillanatban fordul az adatbázishoz

az EOF, ha az alkalmazás adatokat kér a kapcsolódó objektumtól (egy-a-több kapcsolat esetén az objektumoktól). Abban is biztosak lehetünk, hogy bármely, a memóriába került EO objektumból csak és kizárólag egy példány létezik. Az EOF ugyanis leellenőrzi, hogy az adott objektum nem szerepel-e már a memóriában, amikor az adatbázis sorokból objektumokat hoz létre.

A WebObjects Framework (WOF) és a WebObjects Builder

A WebObjects Framework felelőssége az alkalmazás felhasználói felületének biztosítása, valamint az állapotmenedzsment. A felhasználói felületet komponensekből állíthatjuk össze. Egy komponens egy teljes oldalt takarhat, de lehet annak csak egy apró része is. A komponensek tetszőleges módon egymásba ágyazhatók. A komponensek megjelenéssel és önálló viselkedéssel is rendelkeznek. A korábban említett elkülönítés itt is megjelenik, hiszen minden komponens tulajdonképpen három fájlból áll: egy HTML sablonból (ami tisztán a megjelenést definiálja), egy Java kódból (ami tisztán a viselkedést definiálja), és egy köztes deklarációs fájlból, ami az első kettő közötti kapcsolatot térképezi le. Ezzel a megoldással azt nyerjük, hogy a megjelenés bármikor megváltoztatható a kód módosítása nélkül, és viszont. Sőt, a HTML sablon elkészítését akár egy harmadik félre is rábízhatjuk, miközben mi az alkalmazás kódját írjuk.

A WebObjects rengeteg előre gyártott komponenssel rendelkezik. Ezek között egyszerűbbek és bonyolultabbak is vannak. Az igazi csemegét azonban a saját komponensek gyártása jelenti. Itt mindig arra törekszünk, hogy ezek minél több helyen és minél több alkalmazásban újrahasznosíthatóak legyenek. E célt szolgálja az is, hogy a komponensek saját API-val rendelkezhetnek. A komponens ezen keresztül adatcserét végezhet a külvilággal (rendszerint a többi komponenssel). Az adatcserét a WOF meghatározott módon és időben, automatikusan szinkronizálja.

A komponensek elkészítésében a WebObjects Builder nevű grafikus eszközt használjuk. Ez a szoftver a komponenst alkotó mindhárom fájlt egyidőben képes kezelni. Segítségével a hagyományos webszerkesztő programokhoz hasonlóan, egy grafikus felületen állíthatjuk össze a komponens HTML sablonját. Folyamatosan látjuk és szerkeszthetjük a komponens kódjának változóit és metódusait, valamint nagyon egyszerűen elvégezhetjük a HTML sablon és a kód összekapcsolását is.

Mivel a HTTP egy állapotmentes protokoll, ezért minden többfelhasználós webalkalmazásnál nagy kihívást jelent az állapot megőrzése. Ezt a WebObjects úgy oldja meg, hogy minden felhasználó számára létrehoz egy önálló Session objektumot, aminek egyedi, véletlenszerűen generált azonosítót ad. Az egyes felhasználók ez után másmás webcímet látnak, amiben a session azonosító és az éppen lekért oldal komponenseinek azonosítói vannak belekódolva. A beérkező HTTP kérésről a WebObjects így pontosan meg tudja állapítani, hogy az melyik felhasználótól és melyik komponensről érkezett. A Session objektum szabadon bővíthető. Az itt elhelyezett változó például a felhasználó kosarának tartalmát hordozhatja, ha egy e-bolt alkalmazásról van szó. Természetesen más állapotmenedzsment megoldásokat is támogat a WebObjects, köztük a session azonosítójának cookie-ban történő elhelyezését.

A Direct To Web Framework (D2W) és a Rule Editor

Ha az EOF-re a koronaékszer minősítést használtuk, akkor a D2W-re a „titkos fegyvert" lehetne. Ez a keretrendszer az úgynevezett Szabály Alapú Gyors Alkalmazásfejlesztés (Rule-Based Rapid Application Development – RBRAD) technológián alapszik. Ennek lényege, hogy a kliens oldali HTML felület, az EOModel és egy szabálygyűjtemény alapján önállóan, futásidőben jön létre. Megfigyelhető ugyanis az a tendencia, hogy míg az adatmodellezés és az üzleti logika lefektetése viszonylag rövid idő alatt elvégezhető, addig a hozzá tartozó HTML felület és alkalmazás logika – a modellben szereplő entitások számának arányában – sokkal lassabban. Azt mondhatjuk, hogy minden entitáshoz kell minimum egy kereső-, egy listázó-, egy megtekintő- és egy szerkesztőoldal, ráadásul két entitáshoz már ennek duplája, tízhez ennek tízszerese szükséges. A D2W lényege, hogy az entitásokkal végzendő feladatot (keresés, listázás, megtekintés, szerkesztés) tekinti állandónak, és a konkrét entitás ismeretében ehhez gyártja le ő maga az oldalakat. Ezt a gyártási folyamatot általunk megadott szabályokkal (rules) tovább finomíthatjuk. Ezek a szabályok egyszerű logikai állítások. Például: „Ha az aktuális objektum egy Személy, és a vezetéknevéről van szó, és jelenleg a megtekintés fázisában vagyunk, akkor a vezetéknév megjelenítéséhez a WOTextField nevű komponenst használd." A szabályokat a Rule Editor nevű eszközzel adhatjuk meg. A D2Wben egy tucatnyi szabály alapkiépítésben is megtalálható, ami azt eredményezi, hogy egészen rövid (akár órákban mérhető) idő alatt is egy működő webalkalmazást tudunk létrehozni. Fontos, hogy a D2W nem egy „alkalmazás-varázsló", hiszen kódot nem generál. Ugyanakkor a D2W nem is egy merev technológia, hiszen teljes egészében testreszabható.

Összefoglalás

Szinte minden webalkalmazásnak szüksége van egy adatforrásra, illetve valamilyen webes integrációra, ami legtöbbször a HTTP kérések kezelését és a dinamikus HTML oldalak előállítását jelenti. A WebObjects objektumorientált keretrendszerei kiforrott infrastruktúrát kínálnak e két lényeges összetevő magabiztos kezelésére. Olyan szilárd alapot adnak, hogy segítségükkel akár további kód írása nélkül is létrehozhatunk egy többfelhasználós, adatbázis alapú webalkalmazást.

Kapcsolódó linkek:

- `http://www.apple.com/webobjects`
- `http://developer.apple.com/tools/webobjects`
- `http://wodev.spearway.com`
- `http://wocode.com`
- `http://www.omnigroup.com/developer/mailinglists/webobjects-dev/`

Szántai Károly

Informatikus mérnök, szaktanácsadó. A Tactus Multimédia Stúdió valamint a Martin & Charles Webstandards Consulting társtulajdonosa és munkatársa. Hobbiszinten húsz, hivatásszerűen tizenkét éve foglalkozik informatikával. Több neves hazai és nemzetközi cégnek készült multimédiás szoftver, oktatóprogram, interaktív érintőképernyős kioszkrendszer és webalkalmazás vezető fejlesztőmérnöke. Webes fejlesztései kapcsán néhány éve érdeklődésének középpontjába a webszabványok és a hozzáférhetőség került. A Martin & Charles szaktanácsadójaként az internet webszabványokra épülő minőségellenőrzési és minőségbiztosítási kérdéseivel foglalkozik.