SZAKDOLGOZAT

Udvari Tibor

Debrecen

2008

Debreceni Egyetem
Informatika Kar

Dinamikus webfejlesztés lehetőségei

Témavezető:
Dr. Rutkovszky Edéné
Egyetemi tanársegéd

Készítette:
Udvari Tibor
Programtervező Informatikus

Debrecen
2008

Tartalomjegyzék

Bevezetés

Mi az internet?

Az internet egy az egész világot körülölelő számítógép hálózat, amely különböző architektúrájú számítógép hálózatokat köt össze.

Az internet kialakulásának története

Az internet története az 1960-as évekre nyúlik vissza. 1969-ben az USA Hadügyminisztériuma telefonvonalon egy kísérleti jellegû, csomagkapcsolt hálózatot hozott létre (ARPAnet: Advanced Research Projects Agency Network). A hálózathoz egyre többen kapcsolódtak hozzá elsősorban oktatási és kutatási intézmények. Az ARPAnet mellett létrehozták a hasonló technológiával mûködõ MILnet (Military Network) hálózatot, és 1983-ban a két hálózatot összekapcsolták. Az ARPANET-hez ezután több hálózat is hozzákapcsolódott; pl. a MInet (a MILnet európai megfelelõje), a SATnet és WIDEBAND (mûholdas hálózatok), az NFSnet (National Science Foundation Network), a BITnet (Because It's Time Network), a USEnet, stb. Az 1990-es években már a nagy számítógépes kereskedelmi szolgáltató központok (CompuServe, America Online, stb.) is elérhetõek lettek az interneten keresztül és az üzleti alkalmazások köre azóta is rohamosan bõvül. Az internet az intézményeken belüli információ szervezésére is hatással van: kialakult az intranet, internet technológiáját használó vállalati információs rendszer.

Jelenleg Európát és Amerikát az óceánon át üvegkábelek kötik össze, és mûholdon keresztül is lehetséges az adatok átvitele. A jövõ lehetséges perspektívái közé tartozik az informatikai, hírközlõ, telekommunikációs és szórakoztató iparágak összefonódása és az Internet hálózat egységes kommunikációs közegként történõ használata. A nagy adatátviteli sebességet (un. sávszélességet) igénylõ multimédiás alkalmazások új technológiai megoldások kifejlesztését

igénylik, amelyek biztosítják a multimédia információk (pl. hangok, mozgóképek) folyamatos átvitelét.

Az internet jelene

Az internet felhasználási lehetőségei szinte korlátlanok. A felhasználók széles köréből adódóan változatos igényeknek kell megfelelnie. Az internet adatforgalmának jelentős részét a mai napig a böngészés, levelezés adja, azonban a sávszélességek növekedése miatt, a multimédiás alkalmazások is egyre nagyobb teret hódítanak, mint például: videó, hangkonferenciák. Ide tartoznak az azonnali üzenetküldő alkalmazások, rádió- és videóstreamek. Az internetforgalom másik jelentős részét az adatcserék adják, amik két vagy több felhasználók közötti adatok átvitelét jelenti. Az adatforgalom ugrásszerű növekedése azonban a sávszélesség igény rohamos növekedését, a hálózati eszközök terhelésének növekedését eredményezi. Ez az internet folyamatos fejlesztését eredményezi.

Az internet jövőképe

A fejlődés és a felhasználói igények folyamatos változása miatt, az internet dinamikusan változik. A szoftverfejlesztés is követi az internet változását, kezd abba az irányba eltolódni, hogy a fejlesztők nem asztali gépekre fejlesztik az alkalmazásaikat, hanem az internet felhasználásával távoli, úgynevezett szervergépekre fejlesztenek alkalmazást. Ennek az az előnye, hogy a felhasználóknak lehetőségük van, a világ bármely pontján, bármilyen számítógépen folytatni az otthonukban, vagy az irodájukban félbehagyott munkát, nem szükséges a szoftvereiket az általuk éppen használt számítógépre feltelepíteni. Ez az úgynevezett web2.0 technológia. Erre a technológiára nagyon jó példa többet között a Google által nyújtott szolgáltatások jelentős része, például Google Docs, ami a dokumentumaink, kimutatásaink, prezentációink online létrehozását, szerkesztését, és publikálását teszi lehetővé.

Az internet technikai oldala

Az internet tipikusan több ezer számítógép hálózat. Ezeket a hálózatokat egy kommunikációs szabvány szerint kapcsolják össze. Ez a TCP/IP protokoll. Az interneten minden hálózati eszköz egyedi azonosítóval, úgynevezett IP címmel rendelkezik. A hálózati eszköz fogalomba, nem csak a hálózatba kötött számítógépek tartoznak, hanem egyéb hálózati eszközök is, például switchek, routerek, nyomtatók. Az IP cím négy vagy hat bájt hosszúságú azonosító számsorozat, ami magát a hálózatot, és a hálózatba kötött eszközt képes azonosítani. A négy vagy hat bájt hosszúságú azonosítók közötti lényeges eltérés, hogy hat bájttal több hálózati eszköz azonosítására alkalmas. Az internet rohamos fejlődése miatt, a négybájtos azonosítók kezdenek korlátot képezni, ugyanis a világon egyre több azonosítóra van szükség. Érdekességként szeretném megjegyezni, hogy a négybájtos IP-címek két éven belül elfogynak Kínában, ezért ott az IPv6 szabványra való átállás rohamléptekben halad előre. Ez azonban globális probléma a világban. A kontinensek közötti kapcsolatot mélytengeri optikai vezetők, és műholdak biztosítják.

Az internetes tartalmat elméletben bármelyik a hálózatba kötött számítógépen tárolhatjuk. Bármelyik felhasználónak van lehetősége saját tartalmát megosztani a világhálón. Gyakorlatban azonban ez másképp néz ki. A gyors keresés, információcsere alapkövetelménynek számít napjainkban, ezért az otthoni szélessávú kapcsolatok nagy százaléka erre a feladatra alkalmatlan. Felmerül a kérdés, mit nevezünk szélessávú kapcsolatnak. Szélessávú kapcsolatnak nevezzük azt a kapcsolatot, ami legalább 384kbit/másodperc letöltési és 256kbit/másodperc feltöltési sebességet képes biztosítani.

A tartalmak gyors, folyamatos megosztásához speciális számítógépparkokat hoznak létre, úgynevezett szerverfarmokat, ahol megfelelő körülmények között üzemelnek a tartalmat megosztó gépek.

Az internet szoftver oldala

Önmagában a megfelelő hardver, még a felkarú óriástól is kevesebb. Egy az internetes tartalmat megosztó számítógépnek ugyanúgy rendelkeznie kell operációs rendszerrel, és

egyéb kiszolgáló programokkal, mint egy asztali gépnek, azonban ezek a programcsomagok különböznek asztali társaiktól. Operációsrendszerek terén mondhatjuk, hogy a bőség zavara van. Lehetőség van Windows alapú rendszerek, illetve UNIX/Linux alapú rendszerek felépítésére. Az alkalmazott operációs rendszert, illetve az egyéb programokat a felhasználási kör határozza meg. Egy weboldalak megosztására szánt rendszernek szüksége van úgynevezett web szerveralkalmazásra, levelező szerverhez levelező program, fájlmegosztó kiszolgálóhoz pedig fájlszerver programra, alkalmazás szervernek pedig alkalmazás szerver programra. Nyilvánvaló, hogy az internet méretéből adódóan, és a mindenki által elérhetősége miatt szükséges, hogy a rendszereket mind hardveresen, mind szoftveresen védjük az illetéktelen hozzáférésektől. A biztonság megteremtése a rendszergazdák feladata.

Mik azok a weboldalak?

Egy vagy több webes forrásból származó információk összessége, amelyet egyidejű feldolgozásra szánnak, és egy egyszerű URI határoz meg. Jobban mondva, egy weblap egy webes forrás, amely további webes forrásokat foglalhat magába egyetlen közös egységként való feldolgozásra, és azon egyedüli webes forrás URI hivatkozik rá, amely nem foglaltatik benne. Az URI egy rövid karaktersorozat, amely egy hálózati erőforrás azonosítására szolgál.
Weboldalt bárki készíthet, szinte minimális informatikai tudás elegendő hozzá. A weboldal publikálása azonban nem ilyen egyszerű feladat. A megfelelő hardver, szoftver és hálózati kapcsolat szükséges hozzá. Manapság elterjedt megoldás az előbb említett problémára, hogy cégek jönnek létre, akik a megfelelő infrastruktúra biztosítására szakosodva, úgynevezett hosting szolgáltatást kínálnak. A hosting szolgáltatás hardver, szoftver, hálózati kapcsolat bérlési lehetőségét takarja. Ebben az esetben a szolgáltató biztosít minden szükséges feltételt, gyakran lehetőség van a weboldal elkészítését is az ő vállára helyezni. Ezek a cégek az ügyfél weboldalainak kiszolgálására üzembeállított gépeket speciális környezetben, szerverszobákban, szervertermekben helyezi el. Ezen helyek speciális jellemzőkkel rendelkeznek, elektromos hálózat, hálózati kapcsolat. Jellemzően ezek a helységek közvetlenül az adott ország gerinchálózatához kapcsolódnak.

Statikus weboldalak

Statikus weboldalak alatt azokat a weboldalakat értjük, ahol a felhasználói interakció szinte minimális vagy egyáltalán nincs is. Ezek az oldalak nem képesek a tartalom dinamikus változtatására, frissítésére, web programozásban jártas felhasználó szakértelmére van szükség a tartalom megváltoztatására. Manapság ezek a jellegű weboldalak kihalófélben vannak, a felhasználói igények megváltozása miatt. Tipikus felhasználói köre a cégek, kisvállalatok, illetve olyan jellegű oldalak, ahol ritka és kismennyiségű adatváltozások történnek. Statikus weboldalak fejlesztéséhez a következő technológiák ismerete szükséges: HTML, CSS.

Mit nevezünk a táblázatos oldalfelépítés?

Táblázatos oldalfelépítés alatt azt értjük, hogy a weboldalunk tartalmát egy táblázat megfelelő celláiba helyezzük, ezáltal biztosítjuk azt, hogy az oldal tartalma ne csússzon el, az oldal ne essen szét. Ezt a technikát lassan leváltja a CSS által használt pozícionálás.

Táblázat nélküli oldalfelépítés

Ezt a technikát CSS felhasználásával lehet alkalmazni, előnye, hogy képernyőmérettől független, és nincs beégett fix mérete az oldalnak. Jelenleg ez az elterjedtebb oldal felépítési technológia. Fontos azonban megemlíteni, hogy a CSS-sel pozícionált oldalak tökéletes megjelenítése a régebbi böngészőkben nem maradéktalanul tökéletes, pl.: Internet Explorer 6, nem a W3C által ajánlott módon dolgozza fel a CSS állományunkat, így a megjelenítésben eltérések lehetnek. A W3C egy nemzetközi szabványügyi szervezet, ami a webes technológiák szabványosításával foglalkozik többek között.

Miért kellenek webes szabványok?

A szabványok meghatározzák, a weboldalak, böngészők működését, a weboldalt leíró stíluslapok értelmezési feltételeit. Ezáltal lehetővé válik, hogy a szabvány szerint felépített oldalak, stíluslapok a web böngésző alkalmazások nagy százalékával eltérés nélkül jelenítődnek meg. A W3C konzorcium lehetőséget nyújt a fejlesztők által elkészített oldalak validálására. A validálást végző alkalmazás segítségével bárki szabadon ellenőrizni tudja, hogy a weboldala megfelel-e a szabványnak, amennyiben nem nemcsak a hibát kapja meg, hanem javaslatokat is, milyen lehetőségek vannak a kívánt eredmény elérésére.

Dinamikus weboldalak

A dinamikus weboldalak napjaink meghatározó típusú weboldalai. Ide tartoznak mind hírportálok, közösségi oldalak, fórumok, és bármilyen jellegű oldal, ami képes a felhasználóval való interakcióra.

Hogyan épül fel egy dinamikus weboldal?

Egy weboldalt akkor tekinthetünk dinamikusnak, ha a tartalom könnyen változtatható, illetve képes az oldal felhasználóival interakciókat lefolytatni. Ezen weboldalak fejlesztése túlmutat a statikus weboldalak tervezésen, ugyanis speciális felépítést igényel. A nevéből adódik, hogy nem lehet „beleégetni” az összes olyan információt, amit meg akarunk jeleníteni. Szükség van, egyfajta vezérlőre, ami képes weboldal tartalmi frissítésére. Ennek a „motornak” a felépítésére használjuk a webes programnyelveket. Idetartozik a PHP, Enterprise Java, Perl, Flash. A PHP napjaink egyik legelterjedtebb programozási nyelve a weben. Elöljáróban annyit, hogy nagyon egyszerű, és hatásos nyelv, elemeit nagyon egyszerű elsajátítani. Az Enterprise Java alatt a SUN cég által a szerveroldalra kifejlesztett Java verziót értjük. Ez több ágat foglal magába, részletesen később tárgyaljuk.

Tehát adott egy programozási nyelv, amivel képesek vagyunk felépíteni egy „vezérlőt”, ami képes megjeleníteni az információkat. A következő lépés, hogy hol tárolódnak ezek az információk.
Az információk tárolására tengernyi megoldás lehetséges: szöveges állomány, XML állomány, adatbázis, kinek melyik a szimpatikusabb. Egy komolyabb dinamikus weboldalnál azonban a szöveges állomány lehetőségét kizárnám, hiszen, egyrészt a lemezműveletek lassítják a rendszert, másrészt szöveges állományokban keresni, rendszerezetten tárolni az információt kissé nehézkes. Az XML állományokban történő tárolás, nem más, mint egy „felcímkézett” szöveges állományban történő tárolás. Tehát az előnye, hogy könnyebben kereshető, karbantartható, ugyanakkor mégis sok lemezműveletet igényel.

Hogy épül fel egy XML állomány?

Az XML állomány egy szeriális állomány. Speciális tulajdonsága, hogy tartalma fastruktúrába van felépítve, tehát van egy gyökérelem, és ennek van gyerekelemei, a gyerekelemeknek is leszármazottjai és így tovább tetszőleges mélységben.

Példa:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

A fenti példán a <note></note> jelzi az állomány gyökérelemét. Az előbb említett elempár között pedig, a gyökérelem gyerekelemei találhatóak, ami a példánk esetében, a to, from, heading, body tagok. Mint látható minden nyitóelemhez tartozik egy-egy záró elem. Az XML

állomány feldolgozására különböző lehetőségeink vannak. Mind a PHP, mind a Java nyelv rendelkezik erre a célra elkészített csomagokkal, osztályokkal.

Az adatbázisban tárolt információ gyorsan, egyszerűen elérhető, karbantartható, mondhatni ideális dinamikus weboldalak létrehozásához. Valójában itt is fizikai állományokban tárolódnak az információk, de sokkal precízebben, illetve indexelt formában, ami a gyorsabb adatelérést eredményezi. Előnye az is, hogy az adatbázisszoftverek képesek az SQL utasítások értelmezésére, fogalmazhatunk úgy is, hogy az SQL programozási nyelv segítségével az adatbázisok programozhatók. Az adatbázis kezelő rendszerek az 1969-es CODASYL konferencia óta léteznek Ezen a konferencián fektették le a mai relációs adatbázisok alapjait. Több adatbázis kezelő rendszer létezik, van forgalomban. Talán a legelterjedtebb A SUN terméke, a MYSQL, aminek létezik ingyenes változata, ez pedig nagyban hozzájárul a MYSQL népszerűségének napról-napra történő növekedéséhez. A másik közkedvelt ingyenes megoldás a PostgreSQL, szintén ugyanarra az alapra építkezik, mint a MYSQL. Vállalati környezetben pedig az ORACLE cég megoldása preferált, ami fejlettségben előrébb jár az ingyenes megoldásoknál, lévén, hogy piacvezető adatbázis kezelő alkalmazás. Természetesen az ORACLE is kiadott, egy ingyenes, limitációkkal használható tanulói verziót, ami képes akár kisebb alkalmazások adatbázis szükségleteit kielégíteni. Az ORACLE cég termékei egymásra épülnek, így komplett rendszereket lehet kiépíteni, ami véleményem szerint jobb hatásfokkal üzemeltethető, mintha több különböző alkalmazást próbálnánk meg közös munkára hangolni.

Hogyan épül fel egy web szerver?

Az előzőekben tárgyaltuk, hogy a dinamikus weboldalak alapja egy adatbázis, amiből az információkat nyeri az oldalunk. Web szerver felépítésünkben elsődleges feladat elhatározni, hogy maga a kiszolgáló gép, vagy géppark hogy épül fel. Lehetőség van arra, hogy az adatbázist külön hardver kezelje, ennek az előnye, hogy a kiszolgáló hardver vagy szoftver csere esetén nem szükséges az adatbázis migrációja, illetve az adatbázis karbantartása, esetleg az adatbázis kezelő szoftver változtatása nem befolyásolja a kiszolgáló alkalmazását, nem igényel változtatást a kiszolgáló alkalmazásban. Persze az adatbázissal való munka

befolyásolhatja a kiszolgáló működését. Természetesen biztonsági szempontból is ezt a fajta megoldást javasolják a szakértők. Azonban nem minden felhasználó pénzügyi lehetőségei támogatják az adatbázis és a kiszolgáló alkalmazás külön-külön hardveren történő elhelyezését, ekkor lehetőség van e két alkalmazás közös hardveren való futtatására. Ez a megoldás költséghatékony, és egészen jól funkcionál kis terhelésű weboldalaknál. Az előbb említett megoldás előnye, hogy az adatbázisszerver háttérben marad, és esetleges tűzfal, proxy szerverek közbeiktatásával a biztonság tovább fokozható, az adatbázis csak ellenőrzött formában érhető el.

Egy webes alkalmazás ma már elengedhetetlen része, egy úgynevezett levelező szerver, ami elektronikus postafiókokat, azaz email fiókokat biztos a felhasználóknak. Levelező alkalmazások köre is igen széles, ingyenes megoldásoktól a fizetős alkalmazásokig széles a paletta. Természetesen a levelező alkalmazás biztonsági kérdései is fontos részét képzik a web alkalmazásunk biztonságossá tételéhez. Véleményem szerint manapság kiemelten fontos a levelezőszerverek biztonsága, hiszen ki ne talált volna már legalább egy kéretlen reklámlevelet, úgynevezett spam-et a postaládájában. Nemcsak a levelező szerverek beállítása fontos a kéretlen levelek elkerülése érdekében, hanem a web alkalmazásunk segítségével a weboldalakon elhelyezett e-mail címek „olvashatatlanná” tétele az adathalász robotok számára. Az adathalász robotok többek között alkalmasak arra, hogy az internetes keresők indexelt oldalai mentén, a weboldalak szöveges felületeit átvizsgálva egy úgynevezett címlistába összegyűjtsék az oldalakon elhelyezett e-mail címeket. Amiket aztán a feketepiacon megfelelő ellenérték cserében továbbadnak kéretlen levelek százait a nyakunkba zúdítva. Szerencsénkre a legtöbb levelező alkalmazás rendelkezik szűrő funkcióval, ami képes a kéretlen levelek kiszűrésére, azonban sajnos a spammerek (kéretlen leveleket küldő felhasználók) mindig egy lépéssel eme szűrők előtt járnak. Vannak olyan országok, ahol régóta problémát okoznak ezen felhasználók, és törvényi szabályozás van életben a levelek visszaszorítására, ilyen ország például az Egyesült Államok.

Szakdolgozatom témája

Egy viszonylag rövid bevezető után rátérnék szakdolgozatom témájára. Dinamikus web fejlesztés lehetőségei. Dolgozatom célja felvázolni, majd részletesebben kitárgyalni az

interneten uralkodó trendeket, a weboldal készítés alapköveit, bemutatni a modern technológiákat, terjedelmi okokból a teljesség igénye nélkül.

Az általam bemutatásra váró technológiák:

Java enterprise
Javascript
MYSQL relációs adatbázis kezelő rendszer
PHP
XML

Illusztrációként egy hírportál rendszert fejlesztgettem, ami rendelkezik azokkal a minimális funkciókkal, amik szükségesek ahhoz, hogy a felhasználói interakciót megvalósítottnak tekintsük.

Mit nevezek „hírportál rendszernek”?

Értelmezésem szerint, hírportál rendszernek, az olyan dinamikus weboldalakat, amik képes az információk akár percenkénti frissítésére, mindezt úgy, hogy nem igényel webes programozási nyelv ismeretét, tehát rendelkezik valamilyen felhasználói felülettel, amelyen keresztül bárki képes az információk frissítésére. Emellett rendelkezik a napjainkban igen népszerű fórum opcióval, hírlevél küldést is támogat, továbbá mindenki kifejtheti a véleményét egy-egy adott témában. Természetesen az interneten találhatunk erre a célra szánt kész eszközöket, mind szabad felhasználású, mind fizetős változatban. Kedvelt eszközök például a Drupal, Joomla.

Dolgozatom további részében eme hírportál projekt segítségével ismertetem az egyes web technológiákat.

Az adatbázis

Minden dinamikus weboldala alapja egy adatbázis. A bevezetésben már tisztáztuk, milyen alternatív lehetőségek vannak az információk eltárolására (szöveges állomány, XML állományok, stb.) Az adatbázis kezelő rendszer megfelelő kiválasztása igen fontos az oldal felépítését illetően. Itt már eldől, hogy hány bejegyzést, fórum hozzászólást vagy bármilyen egyéb információt lesz képes kezelni a rendszerünk, hiszen mint minden számítógépes szoftver és hardver kapacitásai végesek. Persze vannak technológiák, amelyek segítségével ezek a határok kitolhatók, megnagyobbíthatók, de előbb utóbb keletkezik a rendszerben egy szűk keresztmetszet, amit bővíteni, eltolni már nem lehet. Ilyen eszköz például az ORACLE adattárház megoldása, amivel képesek vagyunk egy adatbázis méretbeli korlátjait kitolni.
Jelenleg az ORACLE cég adatbázis szoftver terméke a piacvezető, de véleményem szerint a SUN által felvásárolt MySQL szoftver a közeljövőben nagymértékű fejlődésen fog keresztülmenni, így a piaci képváltozás előtt áll. Meg kell említenem még a manapság ugyancsak divatos PostgreSQL-t, ami szintén ingyenes szoftver.
Az adatbázis szoftver kiválasztása egyértelműen befolyásol számos tényezőt, mint például: biztonsági kérdések, biztonsági mentés lehetőségei, adatbázis méret határok. Ezen kérdések mindegyik nagyon fontos, és gyakran képes az ügyfél igényeit befolyásolni, hiszen ha az ügyfelünk például egy biztonsági szempontból erősen kifogásolható adatbázis szoftvert szeretne használni, akkor az ügyfél felvilágosítása után, nyilvánvaló, hogy más rendszert fogunk használni.

Biztonsági kérdések

Általános mondat az, hogy egy rendszer annyira biztonságos, mint a leggyengébb láncszeme.
Véleményem szerint a biztonsági kérdések a legfontosabb kérdés, egy adatbázis szoftverrel kapcsolatban. Mennyire biztonságos az adott rendszer? Milyen eszközökkel képes az információink védelmét biztosítani? Milyen ellenőrzött módon lehet hozzáférni az adatbázishoz? Többek között ezekre a kérdésekre kell választ keresnünk, és adnunk egy web fejlesztésnél.

A hackerek, adathalászok célja, nem az oldal grafikai elemeink az ellopása, hanem az oldalon közölt, vagy az oldalon átáramló információk megszerzése. Gondoljunk csak bele, egy online banki szolgáltatásnál, amit tipikusan egy weboldalon keresztül tudunk elérni, milyen jellegű információk áramlanak. Bankszámlaszámok, azonosítók, stb. Ezen információk illetéktelen személyek kezébe kerülése beláthatatlan következményekkel járhat, az adott ügyfél szempontjából. Nyilvánvaló, hogy több módja van az információ megszerzésének, figyelmünk most, az adatbázis ellen irányuló támadási kísérletre fordítsuk.

Azt megállapíthatjuk, hogy nem tanácsos az adatbázis szerverünket az internetre közvetlenül kihelyezni, erre nincs is szükség. Ez azt takarja, hogy az adatbázist futtató számítógép az internet felöl nem látható, bárki, aki használni szeretné az oldalt, csak egy úgynevezett web kiszolgálót tud elérni, ami egy speciális, a fejlesztő által definiált interfészen keresztül képes elérni az adatbázist, ezáltal máris ellenőrzött formában lehetséges csak az információcsere.
A web kiszolgáló és az adatbázis szerver közé lehetőség van, úgynevezett demilitarizált zóna kialakítására, ami lényegében úgy működik, hogy a zónában lévő gépek egyfajta biztonsági falat képeznek az adatbázis körül. Lehetőség van még tűzfalak beállítására is, természetesen ezt a feladatot is számítógépek, vagy intelligens hálózati eszközök látják el.

A leggyakoribb támadási forma az SQL Injection, hadd mutassam be egy kicsit részletesebben ezt a támadási formát.

Az SQL Injection lényege

Az SQL Injection lényege az, hogy a rosszindulatú felhasználók egy nem megfelelően ellenőrzött adatbeviteli mezőn keresztül hajthatnak végre a helyzettől függően akár bármilyen adatbázis lekérést.

SQL Injection típusú támadások

Az alábbiakban olvasható egy egyszerűsített példa az SQL Injection típusú támadásokra:

Tegyük fel, hogy a van egy olyan táblánk, amely a felhasználók bejelentkezési információit, felhasználónevet és jelszót tárol. Ezeket az információkat egy űrlap segítségével meg lehet adni. A mezőket azonban ki lehet tölteni „trükkösen” is. Akár úgy, hogy a támadó egy saját űrlapot készít, amely a céljainak megfelelően formázott beviteli mezőket tartalmaz, és az eredeti, saját szerverünkön található végrehajtó szkriptnek adja a vezérlést, akár pedig úgy, hogy a böngésző fejlécében paraméterként adja meg saját értékeit.
A lényeg az, hogy a paraméterekben elhelyezett idézőjel vagy aposztróf alkalmazásával „ki lehet törni” az adatok közül. Képzeljük el, hogy az alábbi SQL utasításba minden ellenőrzés nélkül behelyettesítjük a kapott felhasználónevet és jelszót:

SELECT * FROM userinfo WHERE username="<username>" AND password="<password>"

És a paraméterek alapján az alábbi utasítást is végrehajthatjuk:

SELECT * FROM userinfo WHERE username="felhaszn";DROP TABLE userinfo;" AND password="xyz";

Ekkor az első és a harmadik SQL lekérés hibás, a második viszont teljesen legális.

SQL Injection típusú támadások ellenszere

Kézenfekvő a fenti támadás ellenszere, minden olyan mezőt, ami megbízhatatlan forrásból érkezett, alaposan meg kell vizsgálni. Megbízhatatlannak kell minősítenünk minden olyan beviteli mezőt, amit a felhasználók számára küldött oldalak tartalmaznak.
A gyakorlatban ez azt jelenti, hogy a vezérlőkaraktereket, amelyek segítségével az általunk szerkesztett SQL lekérés mellé újabbakat lehet bejuttatni, el kell távolítani, vagy pedig escape karakterek elé helyezésével hatástalanítani kell.

Felmerülhet a kérdés, hogy a támadó honnan tudja megállapítani a táblaneveket; ez csak abban az esetben fordulhat elő, ha például az előző hibás SQL lekérés által visszaadott hibaüzenet ezt az információt tartalmazza.

Újabb fontos megállapítás tehát, hogy sehol nem szabad hagyni, hogy a felhasználó egy – akár csak egy egyszerű adatbázis-túlterhelésből származó – „nyers” hibaüzenettel találkozhasson. Ezáltal egyrészt elkerülhető, hogy a támadók információkhoz jussanak a táblák szerkezetével kapcsolatban, másrészt pedig a rossz szándék nélkül érkezők sem találkoznak szerver által generált hibaüzenetekkel.

Az injection típusú támadásoknak több válfaja is van, a másik igen ismert az e-mail injection, amikor levélküldő űrlapokat (kapcsolatfelvétel, hozzászólás) használnak fel spamküldésre.

Web alkalmazásom adatbázisa

Dolgozatom témájának demonstrálására készített web alkalmazás adatbázisának MySQL adatbázis-t választottam. Azért esett a választásom erre a szoftverre, mert széles körben elterjedt, jól dokumentált, széles támogatási háttérrel rendelkezik. A legtöbbi magyarországi ingyenes tárhely szolgáltató kedvelt adatbázis rendszere. Ugyanakkor az ingyenességnek, bárki számára elérhetőségének negatív vonzatai is vannak. A biztonsági rendszerének megkerülésére alkalmas sebezhetőségek elterjedtek, széles körben ismertek.

A MySQL rendelkezik egy webes adminisztrációs felülettel, ez a phpMyAdmin. Ezen a felületen keresztül lehetőségünk van az adatbázisunk adminisztrálására. A phpMyAdmin egy PHP nyelven fejlesztett szoftver, ami egy web-kiszolgálóra támaszkodik. Lehetőségünk van táblákkal kapcsolatos műveletek elvégzésére (létrehozás, módosítás, törlés, stb.), illetve magával az adatbázissal kapcsolatos műveletekre, például: adatbázismentés, adatbázis helyreállítás, stb.. Nagyon hasznos eszköz, ami viszonylag nagy eszközgyűjteményt ad a programozó kezébe az adatbázis adminisztrációjára.
Előző mondatomban utaltam arra, hogy az adatbázis konfigurálását a programozó végzi. Nyilvánvaló, hogy ez a programozó feladat, hiszen ő látja át, hogy milyen információkat dolgoz fel az alkalmazás, és ő tudja táblákba szervezni az információkat.

Lehetőség van az adatbázist grafikus szoftverek megtervezésére. Ilyen program például a DBDesigner. Ezzel az alkalmazással egy grafikus felületen, varázslók lefuttatásával tudunk táblákat létrehozni, a táblák közötti kapcsolatokat, külső kulcsokat is van lehetőség ábrázolni, UML diagramok segítségével. Számomra hasznos funkciója ennek a szoftvernek, hogy képes létrehozni az adatbázist a tervek alapján.

MVC (Model-View-Control)

A web fejlesztés hajnalán, még nem voltak nagymennyiségű információt tartalmazó weboldalak. Ahogy haladunk az időben, megjelentek a dinamikus weboldalak, és ezzel együtt megjelent egy probléma is. A probléma a következő: Adott egy weboldal, ahol a HTML kód és a dinamikus program részletek összekeverve voltak letárolva. Ez azt a problémát generálta, hogy a weboldal kinézetéért felelős grafikusnak szükséges volt komoly programozói tudás, hogy megértse az oldal felépítését, ezáltal el tudja végezni a munkáját. Belátható, hogy ez sokszor problémát okozott. Ezért hozták létre az MVC tervezési mintát. A minta lényeg, hogy elválasztja a programkódot, a felülettől, így nem szükséges a kód változtatása esetén, a felületet is változtatni, és fordítva. Ebből következik, hogy egy web-alkalmazás felülete könnyen cserélhetővé és módosítható vált. Manapság az MVC-nek rendkívül nagy jelentősége van.

Az MNV, vagyis modell- nézet-vezérlő (Model-View-Controller – MVC) egy szoftvermérnöki munkában használt szerkezeti minta. Összetett, sok adatot a felhasználó elé táró számítógépes alkalmazásokban gyakori fejlesztői kívánalom az adathoz (modell) és a felhasználói felülethez (nézet) tartozó dolgok szétválasztása, hogy a felhasználói felület ne befolyásolja az adatkezelést, és az adatok átszervezhetők legyenek a felhasználói felület változtatása nélkül. A modell- nézet-vezérlő ezt úgy éri el, hogy elkülöníti az adatok elérését és az üzleti logikát az adatok megjelenítésétől és a felhasználói interakciótól egy közbülső összetevő, a vezérlő bevezetésével.

Modell

Az alkalmazás által kezelt információk tartomány-specifikus ábrázolása. A tartománylogika jelentést ad a puszta adatnak (pl. kiszámolja, hogy a mai nap a felhasználó születésnapja-e, vagy az összeget, adókat és szállítási költségeket egy vásárlói kosár elemeihez).
Sok alkalmazás használ állandó tároló eljárásokat (mint mondjuk egy adatbázis) adatok tárolásához. Az MVC nem említi külön az adatelérési réteget, mert ezt beleérti a modellbe.

Nézet

Megjeleníti a modellt egy megfelelő alakban, mely alkalmas a felhasználói interakcióra, jellemzően egy felhasználói felületi elem képében. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez.

Vezérlő

Az eseményeket, jellemzően felhasználói műveleteket dolgozza fel és válaszol rájuk, illetve a modellben történő változásokat is kiválthat.

Az MVC gyakran látható web alkalmazásokban, ahol a nézet az aktuális HTML oldal, a vezérlő pedig a kód, ami összegyűjti a dinamikus adatokat és létrehozza a HTML-ben a tartalmat. Végül a modellt a tartalom képviseli, ami általában adatbázisban vagy XML állományokban van tárolva.

Habár az MVC-nek sok értelmezése létezik, a vezérlés menete általánosságban a következőképp működik:

1. A felhasználó valamilyen hatást gyakorol a felhasználói felületre (pl. megnyom egy gombot).

2. A vezérlő átveszi a bejövő eseményt a felhasználói felülettől, gyakran egy bejegyzett eseménykezelő vagy visszahívás útján.
3. A vezérlő kapcsolatot teremt a modellel, esetleg frissíti azt a felhasználó tevékenységének megfelelő módon (pl. a vezérlő frissíti a felhasználó kosarát). Az összetett vezérlőket gyakran alakítják ki az utasítás mintának megfelelően, a műveletek egységbezárásáért és a bővítés egyszerűsítéséért.
4. A nézet (közvetve) a modell alapján megfelelő felhasználói felületet hoz létre (pl. a nézet hozza létre a kosár tartalmát felsoroló képernyőt). A nézet a modellből nyeri az adatait. A modellnek nincs közvetlen tudomása a nézetről.
5. A felhasználói felület újabb eseményre vár, mely az elejéről kezdi a kört.

A modell és a nézet kettéválasztásával az MVC csökkenti a szerkezeti bonyolultságot, és megnöveli a rugalmasságot és a felhasználhatóságot.

PHP (PHP:Hypertext Preprocessor)

A PHP (PHP: Hypertext Preprocessor) egy nyílt forráskódú, számítógépes szkriptnyelv, legfőbb felhasználási területe a dinamikus weboldalak készítése. Emiatt a PHP-t jórészt szerver-oldalon használják, bár létezik parancssori interfésze is, illetve önálló, grafikus felületű alkalmazások is létrehozhatóak vele.

A nyelvet eredetileg Rasmus Lerdorf alkotta meg 1994-ben, de a ma létező egyetlen (és hivatalos specifikáció híján *de facto* szabvánnyá vált) PHP implementációt már a PHP Group tartja karban és fejleszti

A PHP a legtöbb web szerverre, operációs rendszerre és platformra ingyenesen telepíthető. Manapság több mint 20 millió weboldal és egymillió szerver futtat PHP-t, bár a nyelvet használó oldalak száma 2005 augusztusától kezdve folyamatosan csökken. A PHP emellett az Apache web szerver egyik legnépszerűbb beépülő modulja.

Története:

A PHP fejlődése kezdetén csak CGI-programok halmaza volt. Ezeket Lerdorf néhány Perl szkript lecserélésére írta, amelyeket honlapjának karbantartására (például önéletrajzának megjelenítésére és a látogatottság mérésére) használt. Később ezeket a programokat kombinálta a szintén általa írt *Form Interpreter* (űrlap-értelmező) alkalmazással - így jött létre a *PHP/FI*, ami már jóval szélesebb funkcionalitással bírt. Lerdorf 1995. június 8-án adta ki a PHP első nyilvános változatát, ebben a verzióban már megtalálhatóak voltak benne a mai PHP alapvető tulajdonságai: a Perl-éhez hasonló változók, az űrlapok kezelése és a HTML-kód beszúrásának lehetősége. A PHP szintaktikája is hasonló volt a Perl-éhez, de annál jóval korlátoltabb, egyszerűbb és kevésbé egységes volt.

Jelenleg a PHP 5ös verziója a legfrissebb. Az ötös verzió sok újítást tartalmazott: fejlettebb objektum-orientált programozási lehetőségeket, a PDO (*PHP Data Objects*) adatbázis-absztrakciós kiterjesztést, és sok teljesítményt növelő javítást is.

Mit tud a PHP?

A PHP szerveroldai szkriptek írására készült, ugyanazokat a feladatokat tudjuk vele megoldani, amit a CGI programok, például: űrlapfeldolgozás, tartalomgenerálás, stb.

Három fő területen használunk PHP-t:

- Szerveroldali programozás, a PHP program kimenetét a web böngészővel lehet megtekinteni, a szerveren keresztül elérve a szkripteket.
- Parancssori programozás, lehetőség van, szerver és böngésző nélkül is futtatatni.
- Ablakozós alkalmazások írása, nem a PHP a legalkalmasabb eszköz grafikus felületű asztali alkalmazások írásához, de lehetőség van erre is. Ehhez a PHP-GTK-t ajánlom.

PHP használatakor szabadon választható operációs rendszert, web szervert, függvény-alapú és objektum orientált programozás. Mai divatos fejlesztési módszer az OO, ezért személy szerint az OO használatát javaslom. A PHP erőssége az adatbázisok széleskörű támogatása.

Jelenleg a következő adatbázisokat támogatja, a teljesség igénye nélkül:

- Adabas D
- dBase
- Empress
- FilePro
- Hyperwave
- IBM DB2
- Informix
- Ingres
- InterBase
- FrontBase
- mSQL
- Direct MS-SQL
- MySQL
- Oracle

A PHP rendelkezik adatbázis absztrakciós kiterjesztéssel, aminek a neve PDO, haszna, hogy különböző adatbázisok azonos módon való kezelését valósítja meg. Támogatja továbbá ODBC-t, ezért bármilyen a szabványt követő adatbázishoz lehetőség van.

Támogatja a kommunikációt más szolgáltatásokkal protokollokon keresztül. Támogatja például az IMAP, POP3, HTTP, SNMP protokollokat. Továbbá érdemes megemlítenem, hogy támogatja a CORBA kiterjesztést, ami JAVA alatt is el lehet érni, ezáltal egyfajta átmenet képezhető a két programozási nyelv között.

Nem okoz gondot, bármiféle szöveg feldolgozási feladat implementálása PHP alatt, a szöveges állományoktól egészen az XML állományokig.

A PHP szintaxisa

A következő részben a PHP alapvető szintaxisát mutatom be, a teljesség igénye nélkül, a legfontosabb elemeket illusztrálom is.

Escape szekvenciák

Amikor egy PHP állományt a PHP motor feldolgoz, akkor csak a speciális nyitó és záró tagek között elhelyezett php utasításokat értelmezi, minden mást figyelmen kívül hagy. Ez főleg azon alkalmazásoknál lényeges tulajdonság, ahol az MVC nem valósult meg, és a programkód össze van keverve a felülettel, azaz a HTML oldal kódjával.

Példa:

```
<p>Ezt figyelmen kívül hagyja.</p>
<?php echo 'Ezt viszont értelmezi.'; ?>
<p> Ezt szintén figyelmen kívül hagyja.</p>
```

Rögtön ki is kell térni arra, hogy mik a PHP nyelv Escape szekvenciái. Négy féle escape szekvencia létezik

- <?php ?>
- <script language="php"></script>
- <? ?>
- <% %>

A harmadik formát csak abban az esetben tudjuk használni, ha a php.ini konfigurációs állományban a short_open_tag direktíva engedélyezve van. Fontos megjegyezni, hogy kerülni kell a harmadik, azaz a rövid formát kerülni kell abban az esetben, ha az alkalmazást több konfiguráción is futtatni akarjuk, mert a rövid tagok nem minden kiszolgálón engedélyezettek.

Utasítások elválasztása

Az utasítások elválasztására a PHP a pontosvesszőt használja. A zárótag automatikusan magába foglal egy pontosvesszőt is, ezért nem kell az utolsó utasítást pontosvesszővel lezárni. A záró tag magába foglal egy automatikus soremelést is, a záró tag a fájl végén opcionális, egyes esetekben nem szükséges.

Megjegyzések

Perl, C,C++ stílusú megjegyzéseket támogatja.

Példa:

```
<?php
   echo 'Ez egy teszt'; // egysoros c++ szerű megjegyzés
   /* többsoros komment
      még egy sor megjegyzés */
   echo 'ez egy másik teszt';
   echo 'egy végső teszt'; # egy shell-szerű komment
?>
```

Az egysoros megjegyzések az aktuális sorvégig, vagy az aktuális PHP blokk végéig tart, attól függően, hogy melyik végződik előbb. A „C” szerű megjegyzések, pedig a /* */ jelpár közötti karaktereket tekinti kommentnek.

Típusok

A PHP gyengén típusos nyelv, ez azt jelenti, hogy a C-ben, JAVA-ban megszokott szigorúan vett típusok nem léteznek.

A PHP Típusai:

- Logikai adattípus
- Egész számok
- Lebegőpontos számok
- Sztringek
- Tömbök
- Objektumok
- Erőforrások
- NULL

Logikai adattípus

Egy boolean logikai értéket reprezentál, lehet TRUE vagy FALSE. Ha bármilyen értéket logikai típussá szeretnénk konvertálni a (bool) vagy (boolean) típuskonverzió alkalmazásával lehetséges, ugyanakkor ezt nem szükséges alkalmazni, mert automatikus típuskonverzióval rendelkezik.

Egy érték boolean típusú konverziót hajtunk végre, az alábbi esetek biztosan FALSE értékkel térnek vissza:

- Boolean típusú FALSE
- Integer típusú 0
- Float típusú 0
- Üres sztring, vagy a „0” sztring
- Egy elemeket nem tartalmazó tömb
- Attribútumokat nem tartalmazó object
- Speciális NULL érték.

Minden más érték TRUE értéket ad vissza a konverzió során, érdekes, hogy a -1 is TRUE értéket ad vissza.

Egész számok

Az egészek megadhatók decimális, hexadecimális, oktális formában, opcionális előjellel. Ha az oktális formát választjuk, akkor a számot 0-val, ha hexadecimális formát, akkor 0x-szel kell kezdődnie a számnak.

Az egész literálok lehetséges felírási formái:

```
decimal     : [1-9][0-9]*
            | 0

hexadecimal : 0[xX][0-9a-fA-F]+

octal       : 0[0-7]+

integer     : [+-]?decimal
            | [+-]?hexadecimal
```

```
| [+-]?octal
```

Fontos megjegyezni, hogy ha érvénytelen számjegy szerepel egy oktális egészben, akkor a szám hátralevő része figyelmen kívül lesz hagyva.

Egész túlcsordulások

Az egész típus értelmezési tartományán kívül eső értékek float típusúvá alakulnak, ugyanez vonatkozik azokra a műveltekre is, ahol a művelt eredménye kívül esik az integer értelmezési tartományán. A PHP nem rendelkezik egészosztással, két integer érték osztásának eredménye float típusú lesz.

Konverzió egész típusra

Konverziót, az (int) vagy (integer) felhasználásával lehetséges.

Lebegőpontos számok

Háromféle alakban hozhatók létre lebegőpontos számok.

```
1.234;
1.2e3;
7E-10;
```

A lebegőpontos számok az alábbi szintaxis szerint hozhatók létre.:

LNUM [0-9]+
DNUM ([0-9]*[\.]{LNUM}) | ({LNUM}[\.][0-9]*)
EXPONENT_DNUM (({LNUM} | {DNUM}) [eE][+-]? {LNUM})

A számok mérete platformfüggő,a maximális méret 1.8e308.

Konverzió floatra

Sztringek konverziója kivételével, bármely érték konverziója úgy történik, hogy előbb integerre konvertálódik, aztán ez az integer érték konvertálódik floatra.

Sztringek

A sztring karakterek sorozata, egy karakter egy byte-nak felel meg, így a PHP nem rendelkezik Unicode támogatással, ezt a programozónak külön kell megvalósítania, az utf8_encode(), utf_decode() függvényekkel.

Nem létezik korlátozás a sztringek hosszát illetően, eltérően például a C nyelvtől, ahol méretbeli korlátozás van. A sztringek karaktereit nullától számozott indexekkel érjük el, a sztring neve után kapcsos zárójelek között. Sztringek konkatenálásra a „.” operátort használhatjuk. Sztringé való konverziót a (string) módosítóval, vagy a strval() függvénnyel tudjuk megvalósítani.

Sztringek létrehozása:

Háromféleképpen van lehetőség sztringet létrehozni

- Aposztróffal
- Idézőjellel
- Heredoc szintaxissal

Sztring létrehozása aposztróffal

A legegyszerűbb sztring létrehozási forma. Ha egy sztringben aposztrófot, akkor egy \ jelet kell elhelyezni az aposztróf előtt. Ennél a formánál a változók értékei és az escape szekvenciák nem helyettesítődnek be.

Példa:

```
echo 'Ez itt egy egyszerű sztring';

echo 'You deleted C:\\*.*?';
```

Sztring létrehozása idézőjellel

Ebben az esetben speciális jelek feldolgozására is lehetőségünk van, a változók értékei, illetve az escape szekvenciák is behelyettesítődnek.

Speciális jelölések idézőjeles sztringben

- \n újsor (LF vagy 0x0A (10) ASCII kódú karakter)
- \r kocsi vissza (CR vagy 0x0D (13) ASCII kódú karakter)
- \t vízszintes tabulátor (HT vagy 0x09 (9) ASCII kódú karakter)
- \\ vissza perjel
- \$ dollárjel
- \” idézőjel
- *\[0-7]{1,3}* egy karaktersorozat, ami oktális számokra illeszkedik
- *\x[0-9A-Fa-f]{1,2}* egy karaktersorozat, ami hexadecimális számokra illeszkedik

Sztring létrehozása heredoc szintaxissal

Példa heredoc használatára:

```
$str = <<<VAS
Példa egy sztringre, amely
több sorban van, és heredoc
szintaxisú
VAS;
```

Szintaktikája

A <<<jelzés után egy azonosítót kell megadni, majd a sztringet és végül az azonosítót még egyszer. A lezáró azonosítónak mindenképpen a sor legelső karakterén kell kezdődnie. Ugyancsak figyelni kell arra, hogy az azonosítónak meg kell felelnie a PHP elemek elnevezési szabályainak. Fontos, hogy a lezárót tartalmazó sor semmilyen más karaktert ne tartalmazzon, legfeljebb egy pontosvesszőt. Az azonosító nem lehet beljebb kezdve, és nem szabad semmilyen szóköz, vagy tabulátor karaktert sem elhelyezni abban a sorban. A záró

azonosító előtt azonban szerepelnie kell, egy újsor karakternek. Ha a szabályok nem teljesülnek a záró azonosítóval kapcsolatban, akkor a PHP motor tovább fogja az keresni a PHP forrásszövegben.

Helytelen példa:

```
<?php
class ize {
    public $bigyo = <<<EOT
bigyo
EOT;
}
?>
```

Fontos megemlíteni, hogy az aposztrófos sztring létrehozással ellentétben, heredoc szintaxisnál a változók értékei behelyettesítődnek.

Tömbök

A PHP tömbjei rendezett lekérdezéseket valósítanak meg. A leképezés értékeket rendel kulcsokhoz. A PHP asszociatív tömböket valósít meg.

Kétféle út járható a tömbök létrehozására:

- Array() nyelvi elemmel
- Szögletes zárójellel

Array() nyelvi elemmel való létrehozás

Szintaktikája:

array(kulcs => érték
, ...
)

Példa:

```
$arr = array("ize" => "bigyo", 12 => true);
echo $arr["ize"];
echo $arr[12];
```

A kulcs lehet integer vagy string, ha a kulcs egy integer, akkor egészként értelmezi, mint például a sztring karaktereinek kulcsa.
Ha egy tömböt úgy hozunk létre, hogy egyes elemeknek nincs kulcsa, más elemeknek van kulcsa, akkor azoknak az elemeknek, amelyek nem rendelkeznek kulcs értékkel, a PHP motor rendel automatikusan kulcsot. Mégpedig olyan formában, hogy az első „szabad” kulcsértéktől indul az elemek indexelése. Ha egy kulcsnak TRUE értéket adunk meg, akkor az 1 egész típusú értéket fogja felvenni, a FALSE pedig a 0-t.

Kapcsos zárójellel való létrehozás

A létrehozás úgy zajlik, mint egy változó deklaráció, hivatkozni az elemekre a következő formában lehet.

$tomb_nev[kulcs]

Tömbből törölni az unset() függvény segítségével lehetséges. Ha egy változót a (array) módosítóval tömbbé konvertálunk, akkor az egy egyelemű tömb lesz.

Objektumok

Az Objektumok valójában a PHP-ban megírt osztályok példányai. Objektumokkal tudjuk megvalósítani az OO-t. Egy objektum létrehozása a new operátorral történik, ami az adott osztályból származtatott objektum egy példányát hivatott létrehozni.
Ha egy objektumot konvertálunk objektummá értelemszerűen nem változik meg. Ha más típusú változót konvertálunk objektummá, akkor egy új példány keletkezik az stdClass osztályból. Ha az adott érték NULL volt, akkor a létrejött példány üres lesz.

Példa:

```
<?php
$obj = (object) 'ciao';
echo $obj->scalar;  // kimenet: 'ciao'
?>
```

Erőforrások

Az erőforrás egy változó, ami egy külső erőforrásra tartalmaz referenciát. Az erőforrásokat függvények hozzák létre és használják. Az erőforrások konverziója korlátozott, azaz nem lehet bármilyen értéket erőforrássá konvertálni. A használaton kívül lévő erőforrásokat a Zend motor automatikusan felszabadítja, hasonlóan a JAVA garbage collector-jához.

NULL

Meg kell említeni a NULL-t, mint típust, aminek egyetlen lehetséges értéke lehet,ami nem más mint a NULL érték.

Egy változó NULL-nak tekinthető, ha

- NULL érték van hozzárendelve
- Ha még semmilyen értéke nem lett hozzárendelve
- Ha az unset() függvény törölte.

Példa:

```
<?php
$valtozo = NULL;
?>
```

Változók

A változókat a változók neve előtti $ jellel jelöljük. A tömböket is ezzel a jellel jelöljük. A változónevek kis/nagybetű érzékenyek, azok a szabályok vonatkoznak rájuk, amelyek más jelzőkre is.

Szintaktika:

Egy érvényes változó név kezdődhet aláhúzással vagy betűvel, amit tetszőleges számú betű, aláhúzás, szám követhet.

Az értékadásnak két formája létezik, az elsőnél, ha egy kifejezést rendelünk a változóhoz, akkor az eredeti kifejezés egészének az értéke másolódik át a célváltozóba. Ez azt jelenti, hogy ha az egyik változó értéke megváltozik, az nem jár a másik változó értékének megváltozásával. PHP4 óta, lehetőség van a referencia szerinti értékadása, úgy, mint C-ben a mutatók. A hasonlóság annak tudható be, hogy a PHP motorját, azaz a Zend Engine-t, C-ben implementálták.
Példák:

$valtozo1 = ’tesztvaltozo’ // egyszerű értékadás
$valtozo2 = &$valtozo1 //ebben az a valtozo2 a valtozo1 értékét fogja hivatkozni, és mindig a valtozo1 aktuális értékét fogja hivatkozni

Meg kell említenem az előre definiált változókat, amik tulajdonképpen a nyelv fenntartott kulcsszavainak foghatók fel, úgy, hogy rendelkeznek értékkel. JAVA nyelvből példát merítve, a nyelv beépített konstansai. Ilyen előre definiált változó például, a GET, POST, SESSION tömb, ezek részletes tárgyalása később lesz olvasható.

Változók hatásköre

Egy változó hatásköre az a környezet, ahol a változó deklarálva van. Legtöbb esetben a változók csak egy hatáskörrel rendelkezik. Meg kell említenem kettő kulcsszót. Az első a globals. Ez akkor hasznos, ha adott egy vagy több változó amit, a php állományunkban deklaráltunk, és adott egy függvény, ami szintén rendelkezik ugyanolyan nevű lokális változókkal, amik a php állományunkban szerepelnek. Ekkor a láthatóság szempontjából magasabban lévő változók értéket a globals kulcsszó segítségével tudjuk elérni.

Példa

```
<?php
$a = 1;
$b = 2;

function Osszead()
{
    global $a, $b;

    $b = $a + $b;
}

Ossszead();
echo $b;
?>
```

A fenti szkript kiírja, hogy "3". *$a* és *$b* global-ként való deklarálásával minden utalás ezekre a változókra a globális változót fogja érinteni. Nincs megkötve, hány globális változót kezelhet egy függvény. Másik lehetőség még a $GLOBALS tömb használata.

A második kulcsszó nem más, mint a static. Lényeg, hogy ha egy függvény egy változóját ellátjuk ezzel a kulcsszóval, akkor a függvény többszöri meghívása esetén, nem foglal le mindig új memóriaterületet a változónak, hanem a változó továbbra is ugyanazt a memóriaterületet fogja hivatkozni, és ez által ugyanazt az értéket, mint a függvény első futtatásakor. JAVA nyelvre lefordítva, a static kulcsszó, ugyanazt a hatást éri el, mint JAVA-ban a static kulcsszó, azaz osztályszintű adattagokat deklarál.

Állandók

Az állandó, vagy más néven konstans egy skalár értéket tartalmaz. Ez az érték a konstans létrehozása után nem változtatható meg, az érték az állandó nevével hivatkozhatóak. Előnye, hogy ha egy kódban egy érték sokszor fordul elő, és érdemes nevesíteni, akkor az esetleges érték megváltoztatásához elég csak a konstans definícióját megváltozatni. Csak skaláris adat, azaz boolean, integer, float, string lehet konstans értéke.

Példa

```
define("IZE","valami"); // Az IZE konstans a valami sztringet kapta értékül.
```

```
Maga nyelv rendelkezik beépített konstansokkal, például:
```

```
__LINE__ a file sorainak száma
```

Operátorok

Egyszerűen megfogalmazva az operátor nem más, mint olyan művelet, ami értékekből kifejezéseket állít elő.

Hétköznapi példa az operátorra.:

3 + 5 // ebben a kifejezésben a +-jel látja el az operátor funkciót, hiszen ez reprezentálja az összeadás műveletét.

Az operátoroknak három típusát különböztetjük meg, aszerint, hogy hány értéket használ fel:

- Unáris // egy operandusa van
- Bináris // két operandusa van
- Ternális // három operandusa van

A ternális operátor speciális, valójában egy ternális operátor létezik:

(kifejezése)?(igaz a kifejezés):(hamis a kifejezés)

Szemantikája:

Ha a kifejezés igaz, akkor az igaz kifejezésen folytatódik a program futása, ha hamis a kifejezés, akkor a hamis ágon fut tovább a program.

Abban az esetben, ha egy kifejezésben több operátor szerepel, akkor az operátorok végrehajtási sorrendjét rögzíteni kell, hiszen a műveletek végrehajtási sorrendje nagyban befolyásolja a kifejezés eredményét. Erre találták ki az operátorok precedenciáját. Az operátor precedencia nem más, mint egy szabályrendszer, ami az operátorokat rangsorolja egymáshoz viszonyítva.

Vezérlési szerkezetek

A PHP vezérlési szerkezetei nagyfokú hasonlóságot mutatnak a C nyelv vezérlési szerkezeteivel. Éppen ezért nem tárgyalom részletesen az összes vezérlési szerkezetet, csak a különbségekre, illetve a PHP nyelv sajátosságaira térek ki.

Feltételes elágaztató utasítás

```
if(feltétel){
 feltétel igaz
}
else{
```

feltétel hamis

}

Szemantikája tökéletesen megegyezik a C nyelv if utasításával, az else ág opcionális, a csellengő else probléma ennek következtében a PHP-nál is fennáll.

Elseif

Ez a szerkezet valójában két egymásba ágyazott if szerkezet összeágyazása.

Ciklusok

A PHP a következő ciklusfajtákkal rendelkezik

- Elől tesztelő ciklus (while)
- Hátul tesztelő ciklus (do-while)
- Előírt lépészámú ciklus (for, foreach)

A while, do-while ciklus szemantikailag megegyezik a C nyelv megfelelő ciklusaival. A for ciklusnál lehetőség van abban, hogy a ciklusfejben deklaráljuk a ciklusváltozót, amit az ANSI C szabvány nem engedélyez.

Foreach ciklus

```
foreach (tömb_kifejezés as $ertek)
   utasítás
foreach (tömb_kifejezés as $kulcs => $ertek)
   utasítás
```

A foreach ciklus tökéleten alkalmas a PHP által támogatott tömbök bejárásra. Lévén, hogy a PHP tömbjei asszociatív, igy ha a tömb kulcs értékeit hivatkozzuk, akkor megkapjuk az értékeket.

Break, Continue

Ezen két utasítás a ciklusmag ismétlését szabályozza. A break utasítás hatására bármilyen ciklusmag végrehajtása azonnal véget ér és a vezérlés a ciklus utáni utasításon folytatódik. A continue utasítás pedig a ciklusmag végrehajtását szakítja meg, a ciklusfeltételt kiértékeljük és amennyiben a feltétel igaz, a ciklusmag végrehajtásra kerül.

Többirányú elágaztatás

Ennek megvalósítására a switch szerkezetet használjuk. Szemantikailag megegyezik a C-beli rokonával, ezért részletes tárgyalásától eltekintenék.

Declare

Egy kódblokk számára adott futtatási direktívák beállítását teszi lehetővé. A declare szintaxisa a következő.:

```
declare (direktíva)
    utasítás
```

A direktíva rész a declare blokk működését szabályozza. Jelenleg csak egy direktíva használható, a ticks. A blokk utasítás része mindig egyszer fut le, a konstrukció globális hatáskörben is használható, ekkor minden utána következő kódra hatással van.

Tick

A tick egy olyan esemény, amely minden N darab alacsonyszintű utasítás végrehajtásakor bekövetkezik a declare blokkban. Az N értéket a ticks = N szintaxissal kell megadni, a direktíva blokk részében.

Return

A return() utasítás kiadása után, a program befejezi az aktuálisan futó függvény végrehajtását, és visszatér a függvényhívást követő utasítás végrehajtására.

Include() és require()

Mindkét utasítás segítségével lehetőségünk van külső php állományok beillesztésére és feldolgozására. Mindkét utasítás működése megegyezik, a kivételkezelés kivételével.

Függvények

A függvény feladata, hogy egyetlen értéket határozzon meg. Tipikusan azokat a programrészleteket szokás függvénybe átszerezni, amiket a program futása során többször szeretnénk végrehajtani.

Példa:

```
<?php
function foo($arg_1, $arg_2, /* ..., */, $arg_n)
{
   echo "Példa függvény.\n";
   return $retval;
}
?>
```

A függvényspecifikáció a function kulcsszóval kezdődik. Ezután egy formális paraméterlista következik, ami null vagy több elemet tartalmazhat. Ezután következik a függvény törzse, ami a végrehajtandó utasításokból áll, és amennyiben a függvény egy vagy több értéket számít ki, abban az esetben a függvénytörzs végén kell egy return utasítás, az értékek visszaadása a hívó kódrészlet számára.

Lehetőség van bizonyos feltételhez kötni egy függvény definiálását, továbbá a függvények egymásba ágyazását, azonban ebben az esetben a csak a legkülsőbb függvény hivatkozható, a hívó kódból.

Osztályok és objektumok

A PHP osztály fogalma, lefedi az OO-ban tárgyalt osztályfogalmat. A PHP a programozó választására bízza, hogy eljárás orientált vagy objektumorientált módon fejleszt. Véleményem szerint, mindig az adott feladat határozza meg, hogy melyik módszer alapján fejlesztünk. Kisebb feladatokra nem feltétlenül van szükség OO-ra. A PHP fejlesztést megkönnyítik a különböző keretrendszerek, amik tipikusan OO szemléletben gondolkodnak, és így valósítják meg az MVC-t. Ilyen keretrendszer például a symfony, Zend.

Példa:

```
class A
{
   function foo()
   {
      if (isset($this)) {
         echo '$this definiálva van (';
         echo get_class($this);
         echo ")\n";
      } else {
         echo "\$this nincs definiálva.\n";
      }
   }
}
```

Automatikusan betöltődő objektumok

Sok fejlesztő létrehoz egy különálló php fájlt, amiben osztálydefiníciókat tárolnak. Ez nagyban megnehezíti a munkát, mert az osztályok eléréséhez, be kell tölteni az állományokat, amiket kézzel kellett megoldani. Erre megoldás az automatikusan betöltődő objektumok. Ezt úgy lehet megvalósítani, hogy az osztályban létrehozunk egy __autoload nevű függvényt, ami mindig meghívódik, ha olyan osztályt próbál használni a kód, ami még nincs definiálva.

Példa:

```
<?php
function __autoload($class_name) {
   require_once $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
?>
```

Konstruktorok, destruktorok

Lehetőség van arra, hogy konstruktor metódust adjunk az osztályoknak. Azok az osztályok, amelyek rendelkeznek konstruktorral, meghívják a konstruktort az osztály minden egyes példányosításánál, ezért a konstruktor alkalmas arra, hogy az objektum számára fontos változók kezdőértékét beállítsuk.

Szintaktikája:

void __construct([mixed $args[,$...]])

Destruktor nem más, mint egy függvény, ami akkor hajtódik végre, ha az összes referencia egy objektumhoz, vagy az objektum nyíltan megsemmisül.

Szintaktikája:

void __destruct()

A konstruktorhoz hasonlóan, a szülő destruktort nem hívja meg a motor közvetlenül. A szülő destruktor futtatásához nyíltan meg kell hívni a parent::__destruct() függvényt.

Láthatóság

A láthatóság az OO egységbezárás fogalmát valósítja meg. A PHP láthatósági fogalmai, szintjei fedik a standard OO szinteket, private, public, protected. Láthatóságot alkalmazhatunk változókra, osztályoknál attribútumnak nevezzük őket, illetve metódusokra, amik gyakorlati szemmel nézve egy interfészt alkotnak az objektumok attribútumainak elérésére.

Static kulcsszó

A statikus osztály adattagok, vagy metódusok elérhetőek bármelyik osztálypéldány segítsége nélkül is. Egy statikusként deklarált adattag nem érhető el egy példányosított objektumból. A statikus deklarációnak a láthatóság deklarációja után kell lenni. Mivel a statikus metódusok meghívhatóak osztálypéldány nélkül, a $this ál-változóval nem érhető el a statikus metóduson belül.

Osztály konstansok

Lehetséges konstans értékek definiálása, amelyeknek értéke állandó és megváltoztathatatlan. A konstans deklarálásnál nem kell $ jelet használni, illetve az elérésükhöz sem szükséges. A statikus adattagokhoz hasonlóan, a konstans értékekhez sem lehet közvetlenül hozzáférni az objektumpéldányon keresztül.

A PHP szintaktikai ismertetését befejezettnek tekinteném a teljesség igénye nélkül. Terjedelmi okokból nem tárgyaltam minden részletet maximális részletességgel, nem is ez a dolgozatom célja. A továbbiakban a PHP adatbázishoz való kapcsolódásának lehetőségeit ismertetem, majd egy keretrendszer működését vázolnám fel, azután pedig a web oldalak grafikai elemeinek elkészítését, azaz az oldal dizájnjához tartozó programozói eszközöket tárgyalom

PHP és az adatbázisok

A PHP többféleképpen képes adatbázisokhoz kapcsolódni, azonban ez adatbázisfüggő. A MySQL és a PostgresSQL adatbázisok támogatása a PHP motor részét képezik, tehát maga a nyelv rendelkezik azokkal az eszközökkel, amelyekkel a munka elvégezhető. Más a helyzet az ORACLE adatbázisokkal. A nyelv nem tartalmaz erre eszközt, ehhez a feladathoz egy speciális kiegészítést kell alkalmazni, aminek a neve ORACLE Client Interface, röviden OCI. Ezt az „eljárásgyűjteményt” az ORACLE cég fejleszti, és teszi elérhetővé. Felhasználása hasonlóan egyszerű a fent említett két adatbázis típus eszközeinek felhasználásával. Természetesen ezen eszköz dokumentációja is nyilvános. A következőkben pár példán keresztül szeretném bemutatni, hogyan is kell kapcsolódni egy MySQL és egy ORACLE adatbázishoz, azért erre a kettőre esett a választásom, mert az adatbázisok nagy százaléka, e két rendszert használja.

Kapcsolódás MySQL adatbázishoz:

Példa

$host = 'localhost' ; // A szerver IP címe, vagy domain neve

$user = 'select' ; // Adatbázis felhasználó neve

$password = '123456' ; // Adatbázis felhasználó jelszava

$kapcsolat = mysql_connect($host,$user,$password) ; // itt épül fel egy kapcsolat az adatbázishoz, megadott névvel, jelszóval.

mysql_select_db('portal') ; // Séma kiválasztása

Kapcsolódás ORACLE adatbázishoz

$felhasznalo="javadev"; // adatbázis felhasználó neve

$jelszo="123456"; // adatbázis felhasznló jelszava

$adatbazis="localhost/orcl"; //adatbázis host neve, illetve az adatbázis példány neve

$kapcsolodas = oci_connect($felhasznalo, $jelszo, $adatbazis); // kapcsolat felepitese

Látható, hogy szinte minimális a különbség a MySQL és az ORACLE adatbázishoz való kapcsolódásának folyamatában. A keretrendszerek felhasználásával ezek a lépések elfedhetők, tehát egy interfészen keresztül zajlik a kapcsolódás

Lekérdezések MySQL alatt

$query = "SELECT user_nev FROM portal.users where user_id = 2

// a $query változóba sztringként megadja a lekérdezésünket

 $result2 = mysql_query($query) ;

//végrehajtjuk a lekérdezést, az eredmény egy változóba kerül

$row2 = mysql_fetch_array($result2) ;

// az eredményt tömbbé alakítjuk, ezután már csak a tömb bejárása szükséges az eredmények kezelésére. Fontos, hogy ha egy lekérdezés több sorral tér vissza, akkor többdimenziós tömböt kapunk eredményül

Lekérdezés ORACLE alatt

$query = "SELECT beosztas from javadev.felhasznalo where felhasznalo_nev = '$usernev' ";

// a guery változóba sztringként megadjuk az SQL utasítást

$stmt = oci_parse($kapcsolodas,$query) ;

// a lekérdezés szintaktikai elemzése az adatbázisban

```
oci_execute($stmt) ;

//amennyiben a parsolás sikeres volt, a lekérdezés végrehajtásra kerül

$row = oci_fetch_array($stmt) ;

//a lekérdezés eredményének feldolgozása
```

Látható, hogy a lekérdezések végrehajtásában is nagyfokú hasonlóság van az ORACLE és a MySQL között.

Zend Keretrendszer

Jelen fejezetben egy keretrendszert mutatok be, a teljesség igénye nélkül. Keretrendszerek használatával a programozói munka megkönnyebbül, gyorsabbá válik, a csapatmunka gördülékenyebb, és nem utolsó sorban az MVC tervezési módszertan is megvalósítható.

A Zend keretrendszer egy nyíltforrású PHP5-re épülő web alkalmazás fejlesztő rendszer, ami objektum orientált szemléletben gondolkodik. Természetesen az MVC-t betartja, a különböző részek elkülönülnek egymástól. Egy igazán robosztus és szerteágazó rendszer, amiben elkülönül az adatbázis kezelés, a HTML rendelelés, validáció.

Ízelítő a Zend képességeiből

- Széleskörű adatbázis támogatás
- Felhasználó authentikáció
- XML kezelés
- Web szolgáltatások pl.: Yahoo, Google Gdata
- Form feldolgozás
- Rugalmas FrontController
- Azonosítás és hozzáférés vezérlő listán alapuló jogosultságkezelés
- Munkamenet kezelés
- Naplózás
- AJAX, JSON

A Zend keretrendszer használatához szükséges követelmények:

- PHP 5.1.4 vagy magasabb támogatás
- Web szerveralkalmazás mod_rewrite funkcióval. Például: apache

Ezen követelmények teljesítése nem jelent számottevő problémát, hiszen a PHP 5-ös verziója manapság a legelterjedtebb. A web szerver pedig bármilyen HTML oldal megjelenítéséhez szükséges. Ezen követelményeken felül még szükségünk van magára a keretrendszerre, ami ingyenesen letölthető.

Zend projekt könyvtárszerkezete:

/application
    /controllers
    /models
    /views
/library
/public
    /images
    /scripts
    /styles

Az application mappa tartalmazza a web alkalmazásunkat. A controllers mappa tartalmazza a vezérlő elemeket, a models az adatbázismodelleket, míg a views a megjelenítés elemeit tartalmazza.
A library könytárba magát a Zend keretrendszert helyezzük el.
A public mappa tartalma pedig a view-hez szükséges képeket, javascripteket, és CSS állományokat tartalmazza.

A Vezérlők

A vezérlők működésének megértéséhez vizsgáljuk meg a zend frameworkben használható linkeket.

Például:

http://enalkalmazasom.hu/listaz/tablalistaz

Ahol az enalkalmazsom.hu a domain címem, a listaz a vezérlő vagy angolul controller nevem, és a tablalistaz a művelet.

A vezérlő értelmezésem szerint más, mint az valamilyen tulajdonság alapján összefogható műveletek gyűjteménye. Azért mondom ezt, mert célszerű azokat a műveleteket egy vezérlőben kezelni, amik ugyanarra az objektumra, elemre vonatoznak.

Például: Azokat a műveleteket venném egy vezérlő műveletei közé, amelyek egy adatbázis azonos tábláját használják vagy módosítják.

Fontos megjegyezni, hogy az alkalmazás nyitóoldalát is el kell helyezni egy vezérlőben, ez az IndexController, ami rendelkezik egy index művelettel.
A következő hivatkozások ugyanúgy a nyitóoldalt hivatkozzák:

- http://enalkalmazasom.hu
- http://enalkalmazsom.hu/index
- http://enalkalmazsom.hu/index/index

Példa az IndexController felépítésére:

```
<?php

class IndexController extends Zend_Controller_Action
{
```

```
    public function indexAction()
    {
      // az index művelet programtörzse
    }
}
?>
```

A fenti példából látható hogy az IndexController felépítése a Zend_Controller_Action osztályból való származtatással indul ki. Ezen minta alapján további vezérlők és műveletek létrehozása nem jelenthet problémát.
A nézet

A nézet valójában nem más, mint egy php oldal, azonban művelet implementációkat szigorúan a vezérlőkre kell hagyni, itt csak annyi programkód lehet, amennyi a tartalom felépítéséhez szükséges. Tipikusan adatbázis lekérdezések eredményét reprezentáló adatszerkezetek feldolgozása. A nézet állományok kiterjesztése Zend specifikus és phtml. Mondhatjuk, hogy a nézet állományokban a HTML kódok az uralkodóak, hiszen a HTML utasítások az alkalmazásunk kinézetét befolyásolják. Amennyiben a műveletek implementációt is itt helyeznénk el, úgy felborulna az MVC architektúra, aminek elkerülése a célunk. Akkor mondhatjuk ideálisnak a projekt struktúránkat, illetve az MVC-t betartottnak, ha itt minél kevesebb művelet hajtódik végre.

A modell

A modell valójában az üzleti logikát takarja. Hogy mit takar az üzleti logika, az egy komplex kérdés. A Zend frameworkben tipikusan az adatbázis tábláit, és azok sorait reprezentáló osztályokat találjuk. A sorokat reprezentáló osztályokat nem mindig implementálják, mert nem mindig indokolt ezek létrehozása. Az egyes táblát reprezentáló osztályban azokat a műveleteket kell implementálni, amit az adott táblán szeretnénk végrehajtani.

Példa egy adattáblát reprezentáló osztályra:

```
class News extends Zend_Db_Table
{
        protected $_name = 'news';

        public function insert($data)
        {
                if (empty($data['date_created'])) {
                $data['date_created'] = date('Y-m-d H:i:s');
                }
                return parent::insert($data); #1
        }
        public function update($data)
        {
          if (empty($data['date_updated'])) {
                $data['date_updated'] = date('Y-m-d H:i:s'); |
          }
           return parent::update($data);
         }
}
```

CSS

A CSS angolul Cascading Style Sheets, nem más, mint egy stílusleíró nyelv, amellyel a weboldalak megjelenését írja le. Ezen kívül használható még az XML alapú dokumentumok stílusának leírására is. Például: SVG, XUL

A CSS-t a weboldal fejlesztői használják a weboldal megjelenésének beállítására például: betűszín, betűtípus, betűméret, háttérszín, stb. CSS létrehozásának a célja, hogy elkülönítsék a dokumentum struktúráját, a megjelenítést vezérlő elemektől. Ennek több haszna van, növeli a használhatóságot, csökkenti a komplexitást, könnyebbé teszi a szerkesztést.

Története

Az 1990es évek óta van jelen a CSS. Eredetileg a stíluslapokat a felhasználók használták, mert a HTML korai verziói még csak kevés prezentációs attribútumot tartalmaztak. Mivel nőtt a web fejlesztők és a felhasználók esztétikai igénye a weboldalakkal szemben, a HTML nyelv lassú bővülési folyamaton ment keresztül, aminek hatására egyre több olyan elem került a nyelvbe, ami a megjelenítés szerkesztésére szolgál. A CSS eredetileg Håkon Wium Lie ötlete volt 1994-ben. Bert Bos időközben egy Argo nevű böngészőn dolgozott, mely saját stíluslapokat használt; végül ők ketten döntöttek a CSS kifejlesztése mellett. Ekkor már több stílusleíró nyelv létezett, de a CSS rendelkezett azzal a tudással, hogy egy oldal kinézete több stíluslapból állhatott, ez volt a kapcsolás ötlete. A CSS szintaktikáját jelenleg a W3C konzorcium gondozza, e társaság fejleszti a nyelvet.

CSS lehetőségei

A CSS információk megadásának formái:

- Külső CSS állományban
- Beágyazva a dokumentumba

- Azonnali, felülírva az előbb definiált stílust

Az elemek stílusát szelektorokkal lehet kiválasztani:

- Minden elemre a * szelektorral
- Az elem neve alapján. például: minden ’p’ elem tulajdonságait tudjuk megadni.
- Class vagy id azonosítók alapján

CSS használatának előnyei:

- Több lap vagy akár egy teljes webhely stílusait egy állományban lehet tárolni, így könnyebb módosítani.
- A felhasználók rendelkezhetnek egyedi stílusdefiníciókkal.
- Csökken a komplexitás, a megjelenítés elválik a tartalomtól.

Egyszerű szintaxissal rendelkezik, a CSS stílusleíró szabályok sorozata, a szabályokhoz egy szelektor és egy deklarációs szakasz. A szelektor segítségével lehet megjelölni azon elemeket a weboldalon, amit az adott stílussal akarunk használni. A deklarációs részben a stílus szabályokat találjuk.

Példa:

```
p { // a szelektor a p html tag
  font-family: "Garamond", serif; // betűtípus állítás
}
h2 { // a szelektor
  font-size: 110%; //betűméret
  color: red; // betűszín
  background-color: white; // háttérszín
}
```

Javascript

Eddig a szerver oldal lehetőségeit vizsgáltuk, térjünk át egy kicsit a kliensoldali lehetőségekre. Természetesen a kliensoldali paletta is igen széles, a legelterjedtebb technológiát szeretném bemutatni. Ez nem más, mint a Javascript. A Javascript egy objektumalapú szkript nyelv, amelyet a Netscape mérnökei fejlesztettek ki. Szintaxisa fejlődése során hasonlóságot mutat a JAVA nyelvvel, azonban fontos megemlítenem, hogy a Javascript nem JAVA.

Javascript kód elhelyezhető a HTML állományokban, illetve külön állományban is, ebben az esetben a javascriptet tartalmazó állományt js kiterjesztéssel kell ellátni. Célszerű a javascript kódokat külön állományban elhelyezni, ugyanis ez a módszer nagyban segíti az újratervezést illetve a kód újrafelhasználását.

Példa: egy javascriptet tartalmazó állomány

```
var myDate = new Date() ;
var myHour = myDate.getHours() ;
if(myHour < 10){
 document.write('Jó reggelt') ;
}
else{
 if(myHour >= 10 && myHour < 18){
  document.write('Jó napot') ;
 }
 else{
  document.write('Jó estét');
 }
 document.write(' kívánok!') ;
}
```

A fenti javascript kód nem csinál mást, mit lekérdezi az aktuális időt, és ennek alapján választja ki az üdvözlési formát, amit kiír a HTML oldalon.
Javascript lehetőséget biztosít események figyelésére, ilyen események például:

- Kattintásra: onClick
- Egérmozgatásra: mouseOver
- Fókusz műveletek: blur

Java és a web

Nem csak a PHP az egyetlen nyelv, ami képes webes feladatok elvégzésére. Az Enterprise Java kifejezetten a szerveroldali alkalmazások készítésére lett kifejlesztve, ehhez természetesen hozzátartozik a webes alkalmazások fejlesztése is. A Java több különböző lehetőséget biztosít a webes alkalmazások létrehozása: Szervletek, JSP.

Szervletek

A szervletek Java nyelven íródnak, azonban az általuk nyújtott szolgáltatás, bármely operációs rendszeren, bármely nyelven írt programmal el lehet érni. Fontos, hogy ezek az alkalmazások képesek legyenek a hálózatok keresztül történő kommunikációra. A szervletek nem csak szerver szerepkört tölthetnek be, hanem például middleware feladatokat lássanak el. Például: háromrétegű kliens/szerver struktúrában a szervletek a középső, alkalmazásfüggő szerver komponensek szerepét is betölthetik úgy, hogy közben maguk is kliensek, háttéradatbázis elérést biztosító szerveren.

A szervletek sokféle célra használhatók. A tipikus célok: dinamikus tartalom előállítás, HTML űrlapfeldolgozás, adatbázis műveletek elvégzése JDBC felülettel. Továbbá jól felhasználható több kliens kommunikációjának szinkronizálására. Képes továbbítani a kliens oldali alkalmazások kéréseit más szerveroldali komponensekhez, ezáltal lehetőség válik az erőforrás elosztásra.
Szervletekkel képesek vagyunk a korosodó CGI alapú alkalmazásaink leváltásra, ami a teljesítmény növekedését eredményezi.

JSP (JavaServer Page)

A JavaServer Pages a Java szervletekhez hasonlóan általában szöveges,HTML vagy XML formátumú dokumentum kérés alapján,dinamikus szerveroldali szolgáltatások előállítására szolgáló technológia, ami az Enterprise Java része.

A JSP oldalak lényegét úgy fogalmazhatjuk meg, hogy visszájukra fordított szervleteknek tekinthetjük őket. JSP oldalaknál rögzített szövegek közé kerülnek a dinamikus tartalommódosító utasítások, „inline” kódot tartalmazó HTML/XML oldalaknak tekinthetőek.

A JSP kifejlesztésének célja, hogy elrejtse a szervletek írásának nehézségeit a programozásban kevésbé jártasak elől azáltal, hogy szétválasztja a programozási részét a tartalomtól és a megjelenítéstől.

Példa egy JSP állományra:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page contentType="text/html;charset=windows-1250"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<sql:setDataSource var="db1" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/portal" user="insert" password="123456" />
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=windows-1250"/>
  <title>Menü elem törlése</title>
 </head>
 <body>
 <sql:query var="query1" dataSource="${db1}" sql="SELECT MENU_STRING,MENU_ID
FROM portal.menus"/>
 <table>
 <c:forEach var="row" items="${query1.rows}">
  <c:set var="id" value="${row.menu_id}"/>
  <tr>
   <td><c:out value="${row.menu_string}"/></td>
   <td>
    <%
     int iid = new Integer((String)pageContext.getAttribute("id").toString()).intValue() ;
     out.println("<a href='delete_menu.jsp?id="+iid+"'>Törlés</a>") ;
    %>
   </td>
  </tr>
 </c:forEach>
 </table>
 </body>
</html>
```

Demonstrációs rendszer

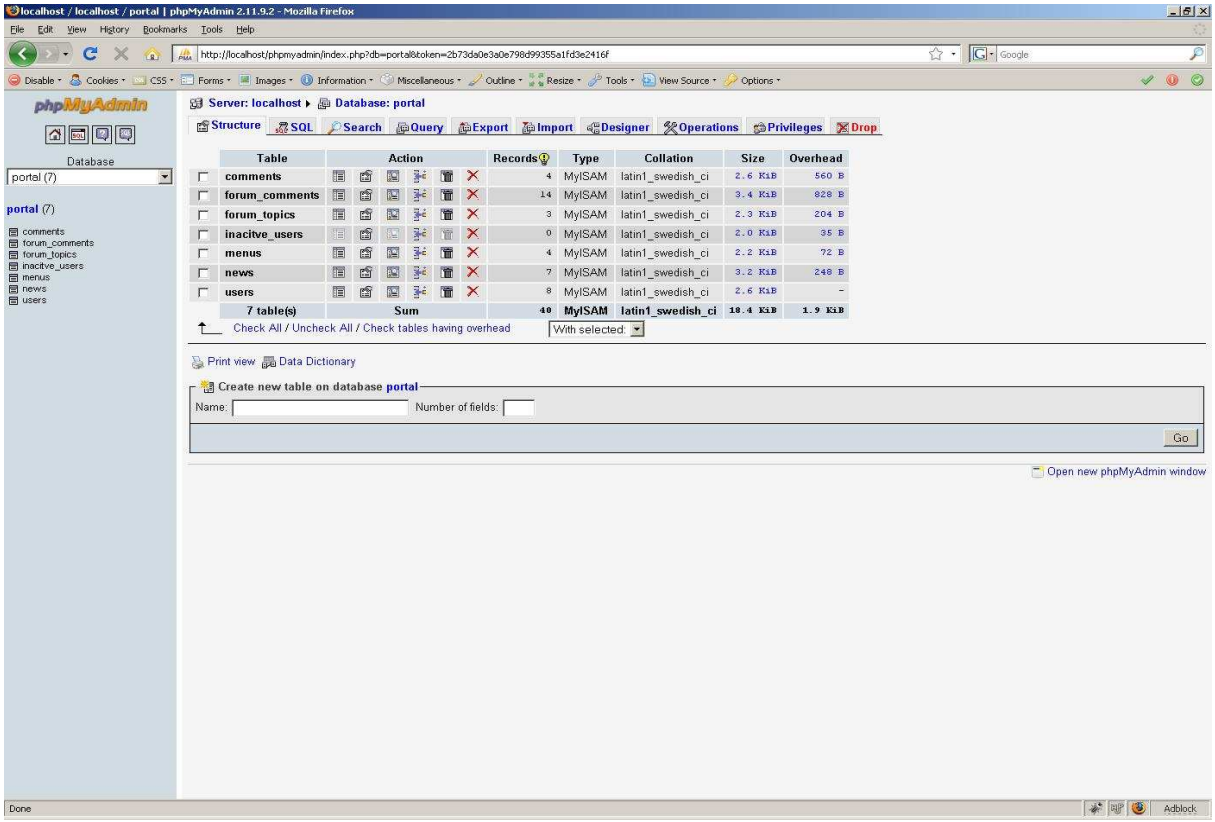
Dolgozatom előző szakaszaiban, az olvasó már megismerhette a manapság trendinek számító web fejlesztési technológiákat. A következő fejezetben a dolgozatomhoz demonstrációs céllal készített web alkalmazást ismertetem, forráskód részletekkel, és képernyőképekkel illusztrálva.

Az adatbázis

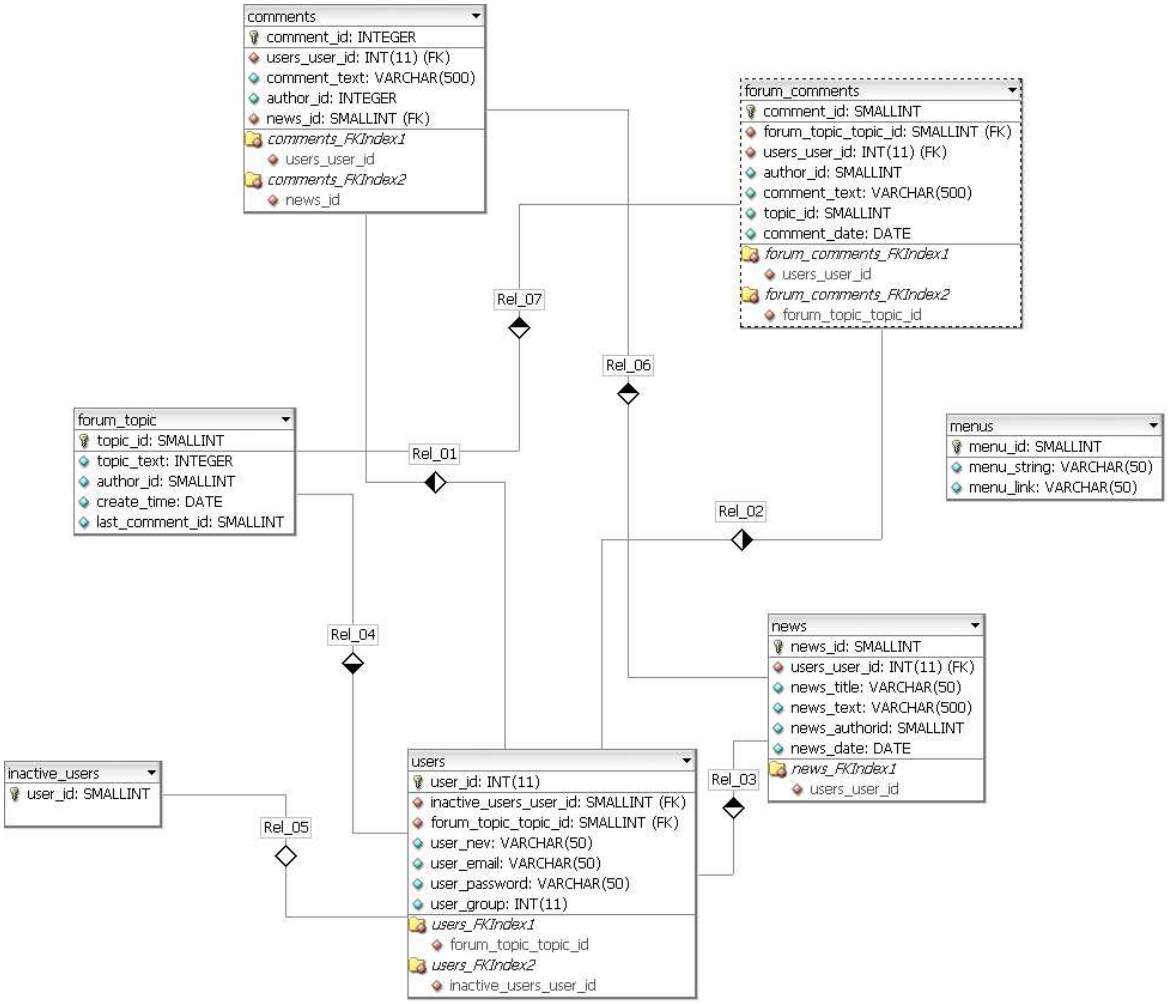
Korábban már tárgyaltuk, hogy minden dinamikus web alkalmazás alapja az adatbázis. Adatbázisok nélkül ma nem tartana ott az informatika, ahol jelenleg tart. Gondoljunk csak bele, banki szektor, légiközlekedés, vagy bármilyen hivatal, bármilyen cég működése mennyivel lenne nehezebb, ha nem elektronikusan tárolnánk az információkat.

Adatbázisnak MySQL-t választottam. Ingyenes, gyors, jól dokumentált, széles felhasználói bázissal rendelkezik.

Adatbázis és adattáblák létrehozása két úton lehetséges, vagy SQL utasítások kézi kiadásával, vagy grafikus felületen, ami mondjuk a phpMyAdmin segítségével történhet.



A phpMyAdmin felülete



Az adattáblák közötti kapcsolat.

A fenti ábrán az alkalmazás adattáblái, a táblák szerkezete és a táblák közötti kapcsolatot láthatjuk. Látható, hogy egy viszonylag egyszerű alkalmazás is sok adattáblát igényel, és a táblák közötti kapcsolat gyakran bonyolult. Fontos a táblák precíz megtervezése, ez nagyban befolyásolja az alkalmazás sebességét, terhelhetőségét, és legfőképp az adatbázisban tárolt információk mennyiségét. Dolgozatomban igyekeztem, a lehető legkevesebb adatot letárolni, hiszen amit lehetőség van algoritmus által előállítani, azt felesleges letárolni, ezek a származtatott adatok. A másik nagyon fontos feladat a redundáns információ tárolás minimalizálása. Ez nem jelent mást, minthogy egy adatot többször eltárolunk, ami nem jó eljárás, hiszen, ha az információ módosul, akkor sok helyen kell módosítani. Ez pedig magában hordozza azt a veszélyt, hogy átsiklik az adatbázis felhasználója egy ilyen adat elem felett, és máris inkonzisztens állapotba kerül az adatbázisunk.

A feldolgozó motor

Ebben az alfejezetben, az alkalmazás azon részét fogom tárgyalni, ami a felhasználói interfésznek tekintek. Azért nevezem interfésznek, mert meghatározott formában, rögzített „szabályok” mellet biztosít lehetőség az adatbázisban tárolt információk elérésére, és bizonyos feltételek teljesítése esetén (regisztráció) módosítására. Itt jegyezném meg, hogy a dolgozatom témájának demonstrálására írt alkalmazást lehetőség szerint tekintsük anti mintának. Azért mondom ezt, mert az alkalmazás fejlődése során, az alkalmazás túlnőtte azt a határt, amikor még nincs szükség keretrendszer és MVC alkalmazására. A funkciók száma miatt, indokolt lett volna egy keretrendszer felhasználása, azonban úgy gondoltam, hogy ettől eltekintek és egy anti mintát mutatok be.
A feldolgozó motor feladat, az adatbázis elérése, és a felhasználói interakció megvalósítása. A „motor” nem más, mint egy program, amit PHP nyelven írtam meg. A motor lekérdez az adatbázisból, módosít adatokat, illetve bővítéseket végez el.
Lekérdezésre példaként emelném ki, hogy a felhasználók az oldalra kihelyezett híreket egy adatbázis lekérdezés segítségével érik el.

Forráskód:

```
$query = 'SELECT news_date,news_title,news_text,news_authorid,news_id FROM portal.news ORDER BY news_date DESC' ;
 $result = mysql_query($query) ;
```

Adatbázis módosításra példaként a következő funkciót nevezném meg. Az oldal regisztrált felhasználóinak lehetősége van a regisztrált email címük módosításra, ez programkód szempontjából az adatbázis egy mezőjének módosítását takarja.

Forráskód:

```
$email = $_POST['email'] ;
$usernev = $_SESSION['usernev'] ;
$query = "UPDATE portal.users set user_email = '$email' where user_nev = '$usernev' ";
mysql_query($query) ;
```

A fenti példakódban kettő érdekességet lehet megfigyelni. Az első a $_POST, ez nem más mint egy a php motor egy speciális változója, ami valójában egy asszociatív tömb, ami a HTTP fejléc POST elemét címzi meg. Ennek az elemnek a felhasználásával van lehetőségünk a HTML űrlapok adatainak továbbítására a php kódunk számára. Másik adattovábbítási elem a $_GET, ami szintén a HTTP fejléc GET elemét reprezentálja, a GET elemeket a hivatkozások fel paraméterezésével használható.

Például: http://hivatkozasom.php?id=1

Ahol az oldal neve hivatkozasom.php, a GET tömb pedig egy id nevű elemmel fog rendelkezni, aminek az értéke 1.
A $_SESSION szintén a php nyelv speciális tulajdonságú eleme, ami szintén felfogható egy asszociatív tömbként, valójában minden egyes felhasználó, aki az oldalt böngészi, egyedi SESSION tömbbel rendelkezik, ez tökéletes alkalmas a felhasználók autentikációjára.

Végezetül nézzük az utolsó adatbázis műveletet, a törlést. Az alkalmazásban lehetőséget biztosítok bizonyos moderátori szerepkörök gyakorlására. A moderátornak lehetősége van a hírekhez, vagy a fórumtémához érkezett hozzászólások törlésére. Ez tehát nem más, mint egy adatbázis elem törlés.

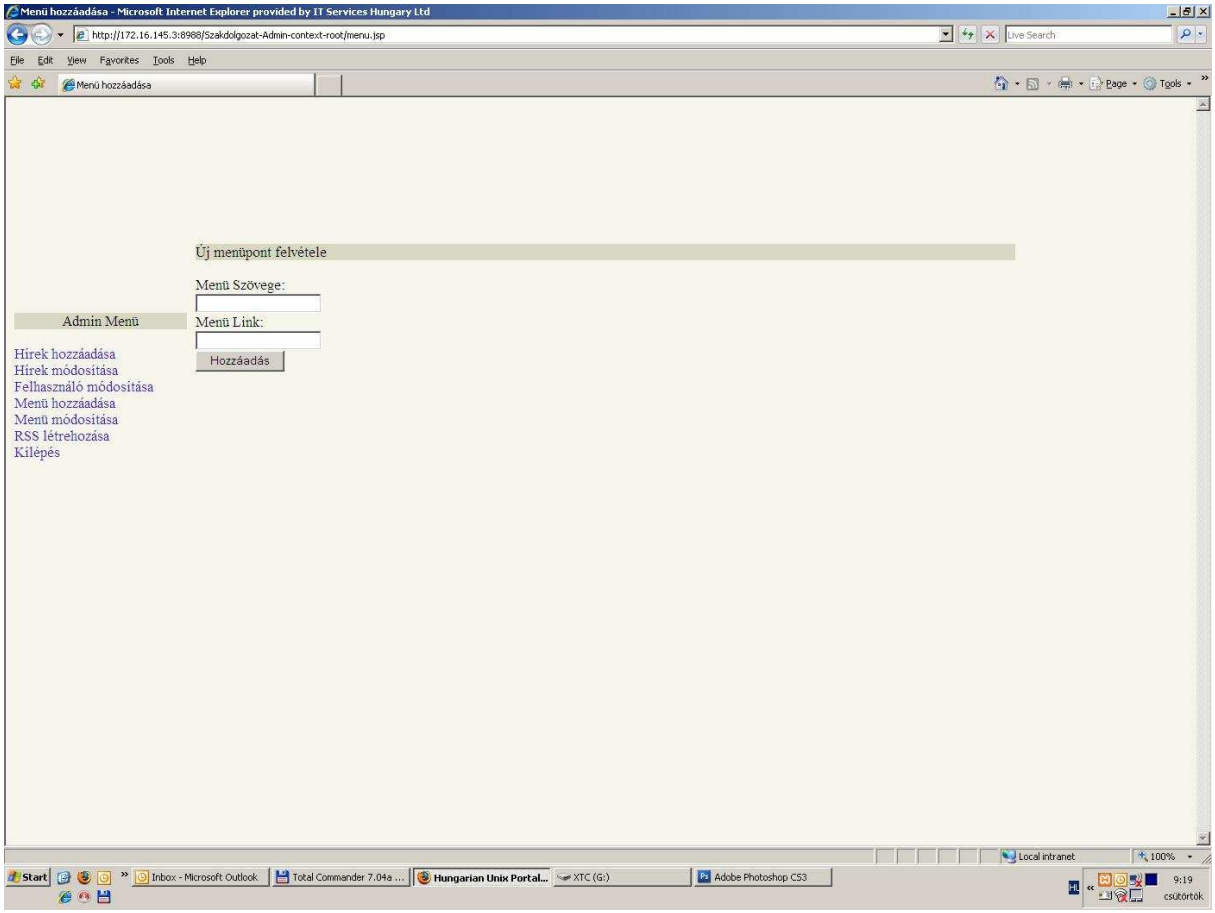
Forráskód:

```
$news_id = $_GET['news_id'] ;
 $comment_id = $_GET['comment_id'] ;
   $query = "DELETE FROM portal.comments where news_id = $news_id AND comment_id
= $comment_id" ;
   mysql_query($query) ;
```

A fenti kódrészletben látható a $_GET alkalmazása, és megfigyelhető, hogy az egyes adatbázis műveletek között szinte minimális különbség van. Lényegében az SQL utasításokban különböznek, valamint ezen utasítások milyen eredményt adnak visszatérési értékként.

Az alkalmazás tervezésekor egy keretrendszer kialakítását tartottam szemem előtt. Alapelvem az volt, hogy szeretnék olyan a későbbiekben bővíthető alkalmazást készíteni, ami modul

szerkezetű, azaz bárki szabadon hozzátehet az alapokhoz, és nem igényel módosítást a már kész alkalmazás. Ezt úgy valósítottam meg, hogy közös adatbázis konfigurációs állományt hoztam létre, amiben minden olyan adat szerepel, ami az adatbázis eléréséhez szükséges. Ezen kívül az oldalon található menüpontokat, amik valójában hivatkozások más HTML vagy PHP állományokra szintén egy adattáblában tárolom. Ezzel megoldható az, hogy új menü pontokat vegyünk fel, esetlegesen töröljünk. Ehhez pusztán csak egy HTML űrlap kitöltésére van szükség.



Menü hozzáadó felület.

Természetesen nem csak a Menü elemek hozzáadására van szükség, hanem a menü pontok törlésére, illetve módosítására. Ezek a már fentebb adatbázis rekord módosítása, illetve törlése. Ezért ezekre nem térnék ki részletesen.

Az alkalmazásban szerepköröket definiáltam, amik segítségével, az egyes szerepkörhöz tartozó felhasználók lehetőségeit szabályozom. Az oldal átlag felhasználóinak lehetősége van

fórumtopicokat indítani, illetve a hírekhez hozzászólni. A moderátorok rendelkeznek a felhasználók lehetőségeivel, és módjukban áll a felhasználók hozzászólásainak moderálására. Az oldalon természetesen a legtöbb jogkörrel az oldal adminisztrátorai rendelkeznek. Nekik lehetőségük van a felhasználók törlésére is. A jogköröket egész számokkal prezentálom,

- 0 átlagfelhasználó
- 1 moderátor
- 3 adminisztrátor

A következő pár gondolatban pedig szeretnék más vizekre evezni. Az előzőekben a frontendet mutattam be. Nézzük meg a backendet, hiszen az adatbázist fel kell tölteni.

Erre a célra egy külön adminisztrációs felületet alakítottam ki, JAVA technológia felhasználásával. JSP alapú weboldalakat használok JSTL könyvtár felhasználásával. A JSTL nem más mint egy tag gyűjtemény,ami előre megírt „függvényeket” tartalmaz.

Példa: JSTL felhasználásával adatbázishoz kapcsolódni.

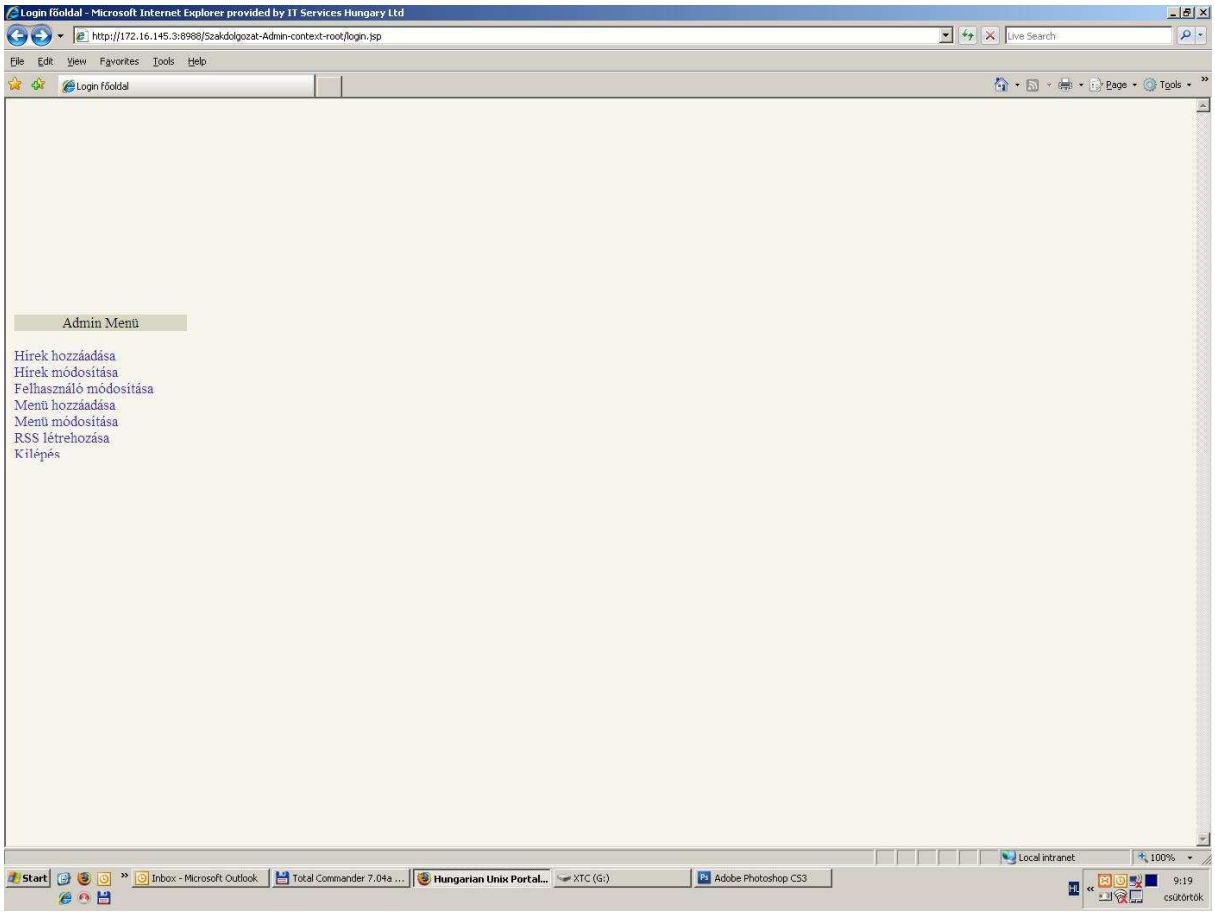
<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost/portal" user="select" password="123456"/>

A példában a var paraméter az adatbázis kapcsolat nevét határozza meg, a driver a JDBC vezérlő program pontos nevét, az url az adatbázis kapcsolat elérhetőségét, user a felhasználónév, password pedig az adatbázis felhasználó jelszava

Az adminisztrációs felületen a következő feladatokat lehet elvégezni:

Menü elemek hozzáadás/módosítás/törlés
Hír hozzáadás/módosítás/törlés
Felhasználó módosítás/törlés
RSS létrehozás

Az első három menüpont azt gondolom, nem igényel további magyarázatot, alaposan körüljártam dolgozatom korábbi szakaszaiban. Az RSS létrehozása funkció érdekesebb. Az RSS nem más, mint egy kivonat a weboldalak tartalmáról, ami automatikusan frissül, ezáltal az oldalak folyamatos böngészése feleslegessé válik, amivel időt takaríthatunk meg. Az RSS lényegében nem más, mint egy XML dokumentum, amiben minden hírhez tartozik egy cím bejegyzés, egy rövid ismertető a hírről, illetve egy hivatkozás az adott oldalra. Az XML állomány generálását JAVA nyelven végzem. Maga a JAVA nyelv több eszközzel rendelkezik az XML állomány létrehozására, például: DOM API, JAX, SAX. Az RSS olvasására a legtöbb mai böngésző képes, természetesen lehetőség van erre a célra kifejlesztett alkalmazások használtára, illetve a Google regisztrált felhasználói számára netes alapú RSS olvasó alkalmazást tesz elérhetővé.



Az adminisztrációs felület nyitóoldala.

Az alkalmazás kinézete

Ez az a rész, ami szerintem önmagáért beszél, hiszen egy weboldal kinézete nagyban befolyásolja, hogyan viszonyulnak a felhasználók az oldalhoz. A demonstrációs alkalmazás szerintem leggyengébb oldal szerintem a kinézet, sajnos nem rendelkezek művészi vénával, ahhoz, hogy kimondottan tetszetős arculatot hozzak létre. Ezért a technológiai oldalát ismertetném. Az oldalt CSS felhasználásával alakítottam ki, táblázat nélküli oldalfelépítést használok A pozícionálást abszolút módon végzem, az egyes területek méretét a margók, és width, height paramétérek beállításával érem el. A CSS-t általánosan ismertető részben már felvázoltam a CSS lehetőségeit. Kiemelném, hogy lehetőség van animációk alkalmazására is. Több út is létezik, animált gifek, amik nem mások, mint képek, vagy a másik út a FLASH elemek felhasználása. Az oldalon található kettő darab FLASH animáció, amik reklámfelületet szimbolizálnak. A kinézet kialakításához érdemes megemlítenem, hogy felhasználtam még a Yahoo cég által fejlesztett, szerkesztett Yahoo UI javascript gyűjteményt. Ez nem más mint javascriptek katalógusa, amit bárki felhasználhat. Ezek a javascriptek paraméterezett formában lettek megírva, a paraméterek megfelelő módosításával, a szkriptek egyedi tartalommal ruházhatóak fel. Véleményem szerint ez igazán hasznos gyűjtemény, ami segítségével egyedi és tetszetős elemek helyezhetők el az alkalmazásunkban, komoly javascript fejlesztői ismeretek nélkül.

Telepítési lépések

A telepítés viszonylag egyszerűen zajlik. A letölthető csomagolt állományban, minden benne van, ami szükséges, ahhoz, hogy működjön az alkalmazás.

Szoftverfeltételek:

- Apache
- MySQL legalább 5.0
- PHP 5
- Egy SMTP alkalmazás például: Mercury Mail
- Valamilyen Java alkalmazás szerver, például: Apache Tomcat, Glassfish, JBoss

A feltételek teljesítése után, nincs más hátra, az adatbázist kell létrehozni, erre a tömörített állományban van egy SQL szkript, amely lefuttatva létrehozza a szükséges szerkezetet. Ezután már csak a web alkalmazás szerver web könyvtárába kell bemásolni a felhasználói oldalt. Az adminisztrációs oldal elemeit, pedig a Java alkalmazás szerver web könyvtárában kell elhelyezni.

Befejezés

Zárszóként foglaljuk össze egy web alkalmazás fejlesztésének lépéseit:

Feladatspecifikáció: A megrendelővel közösen specifikáljuk az elvárásokat az alkalmazással szemben. Erre azért kell nagy hangsúlyt fektetnünk, mert a későbbi esetleges félreértéseket lehet vele elkerülni. Kiemelném a dokumentálás fontosságát, amivel egyrészt a specifikációt rögzítjük, másrészt az írásos dokumentumok védik és kötelezik a fejlesztőket a későbbiekben.

Tervezés: A feladatspecifikáció alapján az alkalmazást meg kell tervezni, át kell gondolni. Több okból is kell tervezni, egyrészt egy jó architektúrával rendelkező szoftver implementálása is egyszerűbb, lehetőség van modulokra bontani a projektet, ami által nem csak egy architekt képes átlátni a struktúrát, és könnyíti a csoportos munkát. Továbbá jól megtervezett szoftvereknél egyes hibák már a tervezésnél felszínre kerülnek. A tervezésre számtalan eszköz létezik, a legelterjedtebb az UML modellező nyelv. UML felhasználásával lehetőség van diagramokat létrehozni, amik a modell vizuális megjelenését végzi. A tervezés folyamatába sorolnám az alkalmazandó technológiák kiválasztását, nyílván a követelményeket legjobban teljesítő technológiákat kell az implementálás során alkalmazni. Továbbá ide venném a tesztfázisok tervezését is, milyen körülményeket kell szimulálni, milyen teszteket kell végrehajtani, a tesztesetek végeredményét is definiálni kell, hogy mikor mondjuk azt, hogy az alkalmazásunk kvázi helyesen működik.

Implementálás: A tervek alapján a kiválasztott technológiák felhasználásával az alkalmazás elkészítése. Az implementálás során a fejlesztők tesztelik az általuk elkészített modulokat, egységteszteket, modulteszteket hajtanak végre Az implementálás folyamata során áll elő kész alkalmazás. Az optimalizálás lépése is ide tartozik, ami során a kész forráskód újraszervezésével egy gyorsabb, precízebb alkalmazás állítható elő, ezt nevezzük refactoringnak.

Validáció: Miután az implementációs fázis véget ért, az alkalmazásunk tesztelése a feladatspecifikáció alapján. Ezt a lépést a fejlesztők végzik. Nyilvánvalóan ebben a lépésben jönnek elő a tervezésből adódó hibák, hiányosságok, amiket lehetőség van javítani.

Verifikáció: A validálás után a megrendelők bevonásával történő tesztelés. Ebben a lépésben a fejlesztők bemutatják a megrendelőnek a szoftvert, a felhasználó pedig eldönti, hogy megfelel-e az elvárásainak az, avagy sem. Természetesen a felhasználó által támasztott olyan igények teljesítése, amit nem dokumentáltak, nem számít követelménynek.

A fenti folyamatok után mondhatjuk azt, hogy elkészült az alkalmazás. Természetesen egy szoftver fejlesztés életútja nem ér véget, akkor, amikor elkészül, a folyamatos fejlesztés, karbantartás, időközben felszínre került biztonsági és egyéb hiányosságok javítása, legalább annyira fontos, mint a fejlesztés. Ide tartozik az adatbázis szoftver hibáitól kezdve, a web szerver alkalmazás biztonsági réseinek befoltozásáig minden.
Nyilván a tartalom frissítése, naprakészen tartása nem a fejlesztők feladata, ez az alkalmazás üzemeltetőit terheli.

Bízom benne, hogy jelen dolgozat segítségével sikerült ablakot nyitnom a mai web fejlesztési trendekre, irányokra, támpontot szolgáltat, hogy milyen technológiai lehetőségek állnak rendelkezésünkre, ha egy dinamikus web alkalmazás fejlesztésébe szeretnénk kezdeni. Sajnos a teljesség, és a teljes mértékben részletes, alapos tárgyalás lehetetlen, a témához kapcsolódó szakirodalom több tízezer oldalt is elérheti, illetve több alternatív út is bejárható egy alkalmazás fejlesztésénél, ami szabadkezet biztosít a fejlesztők számára. Dolgozatomban olyan technológiákat mutattam be, amelyek alkalmasak arra, hogy dinamikus web alkalmazásokat hozzunk létre. Igyekeztem bemutatni adattárolásra, feldolgozásra és megjelenítésre alkalmas technológiákat bemutatni, amelyek ma a legterjedtebbnek számítanak.
Remélem, hogy bárki, aki elolvassa dolgozatomat, el tud indulni a modern web alkalmazások létrehozásának rögös útján.

Felhasznált irodalom

Peter Moulding: PHP Haladóknak
Perfact-Pro Kft., 2002
Virginia DeBolt: HTML és CSS Webfejlesztés stílusosan
Kiskapu Kft 2005

Christopher Shcmitt : CSS Cookbook
O'Reilly 2004

Síkos László: XHTML - A HTML megújulása XML alapokon
BBS-INFO 2004

David Sklar: Learning PHP 5
O'Reilly 2004

http://hu.wikipedia.org/wiki/Internet
http://www.inf.unideb.hu/~bodai/internet/internet_tortenete.html
http://hu.wikipedia.org/wiki/Weblap
http://hu.wikipedia.org/wiki/Uniform_Resource_Identifier
http://hu.wikipedia.org/wiki/HTML
http://www.w3c.org
http://www.oracle.com/solutions/business_intelligence/dw_home.html
http://www.hatekonyhonlap.hu/Az-SQL-Injection.html
http://wiki.cihar.com/pma/phpMyAdmin
http://hu.wikipedia.org/wiki/PHP
http://framework.zend.com/
http://php.net