A PHP programozási nyelv
(webes alkalmazásokra)

Gábor Dénes Főiskola

e-jegyzet

Dr. Medzihradszky Dénes

Fontos! A jegyzet kísérleti jellegű és folyamatos fejlesztés alatt áll!
(Van néhány üres címsor is. Kiegészítésként használja a kézikönyvet!)
A forráskód melléklet nemsokára elérhető lesz.

Ez az első alkalom, hogy egy fakultatív tantárgy esetében az oktatói anyagot teljes egészében elektronikus formában adjuk közre. Ennek fő oka az új tantárgy már most előre látható dinamizmusa, biztos, hogy a jövő évre a jegyzetet teljes egészében át kell dolgozni az új programnyelvi fejleményeknek (PHP 5) megfelelően. Ezért egy, a már kiadásakor elavultnak tekinthető nyomtatott jegyzet megjelentetése nem lenne szerencsés. Ugyanakkor a tantárgy pozíciója csak mostanában körvonalazódik a tanári gárda és a hallgatók első visszajelzéseinek hatására, és szeretném, ha maximálisan a felmerülő igényekhez és a szakmai követelményekhez tudnánk szabni a tantárgyat.
Ez természetesen most plusz feladatokat ró hallgatóra, oktatóra egyaránt, de ha valóban korszerű tudást akarunk átadni, akkor ez úgy érzem vállalható és szeretném, ha partnerek lennénk a tantárgy végső és dinamikus formájának kialakításában.

Budapest, 2004. június 14.

Tartalomjegyzék

Amit a jegyzetről tudni kell

Kedves Olvasóm! A kezében tartott - vagy inkább a képernyőjén látható - jegyzet három fő részből áll. Az első részében röviden összefoglalom mindazt, amit szerver oldali programozás néven oktatunk, kiemelve az architektúrális sajátosságokat és a szerver-kliens közötti kommunikáció problémakörét. A második részben ennek a témakörnek az egyik - és talán a legdinamikusabban fejlődő eszközét, a PHP programozási nyelvet ismertetem dióhéjban. A harmadik rész a tantárgy oktatási keretének megfelelően elsősorban gyakorlati ismeretek ad, konkrét webes példákon keresztül ismertetem a nyelv alkalmazásait. Ez a példagyűjtemény is folyamatosan fejlődik, terveim szerint az oktatás során felmerülő ötletek, megoldások is fokozatosan belekerülnek, így remélem, hogy idővel egy mindenki számára jól használható, gazdag alkalmazási mintakészlet halmozódik majd fel.

Óriási anyagrész a szerver oldali programozás, mert a gyorsan fejlődő webes technikák - akár az egyszerű ismeretnyújtás, akár a csoportmunka szintjén - megkövetelik a változatos eszköztár létrehozását. Ugyanakkor szeretném bemutatni, hogy mint minden alkalmazás-fejlesztés esetében a gondolkodás elsajátítása az elsődleges, mert az egyes megvalósítási módok számos forrásból hozzáférhetők. Ezért kérem az olvasót, hogy az elvi megoldásra koncentráljon és ne a konkrét megvalósításokra. Nem kell a függvénykönyvtárakat kívülről ismerni, elég, ha ki tudjuk keresni a megfelelő függvényt, metódust.

A jegyzetben törekedtem a legkorszerűbb, de már nem kísérleti (béta) stádiumban levő fejlesztőeszközöket ismertetni, de mivel a programozásnak egy valóban szédületes tempóban fejlődő szeletét tárgyaljuk, az információ nagy valószínűséggel a leírással együtt el is avul ☹. Erre jó példa, hogy a jegyzet első nyers változatának elkészülte óta (2003. október-november) megjelent a PHP nyelv 5 verziója, amely alapjaiban változtatja majd meg a PHP nyelven végzett programozást. Ez még jelenleg csak béta változatban került ki, a stabil kiadás egyenlőre várat magára, bár a tanév végére várhatóan az is megjelenik, így a jegyzet átdolgozására máris fel kell készülni. Ennek ellenére jó kiindulópontot adhat ezekhez a technológiákhoz, de nyomatékosan felhívom a figyelmet az internet használatára - ami persze együtt jár az angol nyelvismeret szükségességével.

A fentiek miatt hivatkozásként lehetőleg azokat a webhelyeket adom meg, amelyek az egyes technikák hosszú távon is érvényes forrásai találhatók meg és kiindulási pontot biztosíthatnak az egyéb webes források felé.

Javaslatok a jegyzet használatához, előismeretek

Nem egy teljes, a PHP programozási nyelvet az alapjaitól ismertető jegyzet elkészítésére vállalkoztam. Ezt a feladatot mások már elvégezték, magyar nyelven is számtalan jó kézikönyv kapható. Célom a minták alapján történő tanítás, azaz az elméleti összefoglaló után a gyakorlati webes megoldásokra helyezem a hangsúlyt és konkrét alkalmazási példákon keresztül mutatom be a PHP nyelv használatát, annak speciális lehetőségeit kiemelve. Véleményem szerint nem egy-egy eszközből - adott esetben programnyelvből - kell mindent "kifacsarni", ezért került bele a fontosabb szerver oldali technikák ismertetése is, hogy az olvasó elhelyezhesse ezt az eszközt a webes alkalmazások fegyvertárában.

Kezdetnek javaslom az amúgy is olvasmányosnak szánt elméleti rész átnézését, elolvasását, hogy helyre tudjuk tenni a dolgokat és felfrissítsük webes, hálózati ismereteinket. Próbáljuk meg átgondolni mindazon előnyöket és nehézségeket, korlátokat, amit egy kliens-szerver architektúra hoz magával.

Ajánlatos felfrissíteni programozási ismereteinket is, mert sem a strukturált programozás alapjait, sem az objektumorientált programozás elveit nem fogom részleteiben ismertetni, csupán utalok rá, mint ismertnek feltételezett fogalmakra.

Hasonlóképpen javaslom az adatbázis-kezelési ismeretek megújítását is. A dinamikus webes alkalmazások létrehozását az adatok, dokumentumok perzisztens tárolása teszi lehetővé - jelentős mértékben az elsősorban webes eszközökhöz kifejlesztett, egyszerűen használható adatbázis rendszerek létrejötte adta a döntő lökést a dinamikus alkalmazások ugrásszerű elterjedéséhez. A gyakorlati alkalmazások során az egyszerű lekérdezések, rekordbeszúrások, adatmódosítások, néhány táblás adatbázisok létrehozása rutinszerű lépések lesznek.

Javasolom az internet alapos és kritikai tanulmányozását! Szeretném, ha a hallgatók kellő kíváncsisággal állnának hozzá ehhez az elsősorban gyakorlati ismereteket nyújtó tantárgyhoz és arra próbálnának a tantárgy keretén belül választ találni, hogy az

interneten gyakran előforduló megoldások hogyan hozhatók létre, és ezeken belül hol alkalmazhatók az itt elsajátítható ismeretek!

Ismertnek tételezem fel a HTML, JavaScript alapvető ismeretét, erre nem fogok kitérni a jegyzetben, de akik eddig csak HTML formátumba mentéseket csináltak, azok számára melegen javasolom a HTML szintaktika átnézését.

A gyakorlati problémák megoldását csak számítógép előtt érdemes végezni. Ezért szedtem össze a később ismertetésre kerülő szoftver eszközöket is, hogy bárki össze tudjon rakni magának egy fejlesztő környezetet és otthon is fejleszthessen egyszerűbb vagy bonyolultabb alkalmazásokat. Ne nézzük le ezt a környezetet, bár kétségtelen, hogy egy igazi webszerver egyes esetekben másként működik, az így fejlesztett anyagokat minden további nélkül át lehet tenni egy valódi szerverre, jól elrendezett alkalmazás esetén csupán a beállításokat, konfigurációs fájlokat kell átírni a környezetnek megfelelően.

A tanuláshoz sok sikert kívánok és minden észrevételt, javaslatot szívesen veszek a tantárggyal, jegyzettel kapcsolatban - munkámban sokat segítenek az olyan gyakran felmerülő kérdések is, mint például: Ezt a konkrét problémát hogyan lehet megoldani?

Bevezetés

A számítógépeken alapuló informatika hajnalán már felmerült, hogy az egyes drága, ám nagy teljesítőképességű számítógépek sok kisebb teljesítményű gépet ki tudnak szolgálni információval, futtatható programokkal. Ezt az elvet gyorsan belátták és hamar alkalmazásra került, így a szerver-kliens közötti munkamegosztás fogalma nagyon hamar bekerült a köztudatba. Az egyes számítógépek teljesítményének megnövekedésével később a szerverek jelentősége kisebb lett, ám ismét növekedésnek indult a számítógépek közötti kommunikáció elterjedésével és általánossá válásával párhuzamosan. Napjainkban a szerverként üzemelő számítógépek fő feladata az információ szolgáltatás, akár egyszerű szöveges és képi formában (webes alkalmazások), akár pedig strukturált adatok formájában (adatbázis szerverek). Ez a két forma természetesen keveredhet is, nagyon sokszor előfordul, hogy a dokumentum jellegű információkba kisebb-nagyobb mértékben adatbázisokból nyert információt ágyazunk bele. Szerverek intézik még az ember-ember közötti kommunikáció elosztását is, ezek az egyes szolgáltatók által üzemeltetett e-mail szerverek. A jegyzetben a dinamikus webes alkalmazások által

biztosított információ szolgáltatás eszközeit ismertetjük, a hangsúlyt a szerver oldalon megvalósítható alkalmazásokra helyezve. A technika azonossága miatt nem csak a World Wide Web keretein belül képzelhető el a kifejlesztett alkalmazás, hanem helyi hálózatokon és egyedi gépeken is üzemeltethető, bár nem ezek voltak a megcélzott területek a jegyzet írása során.

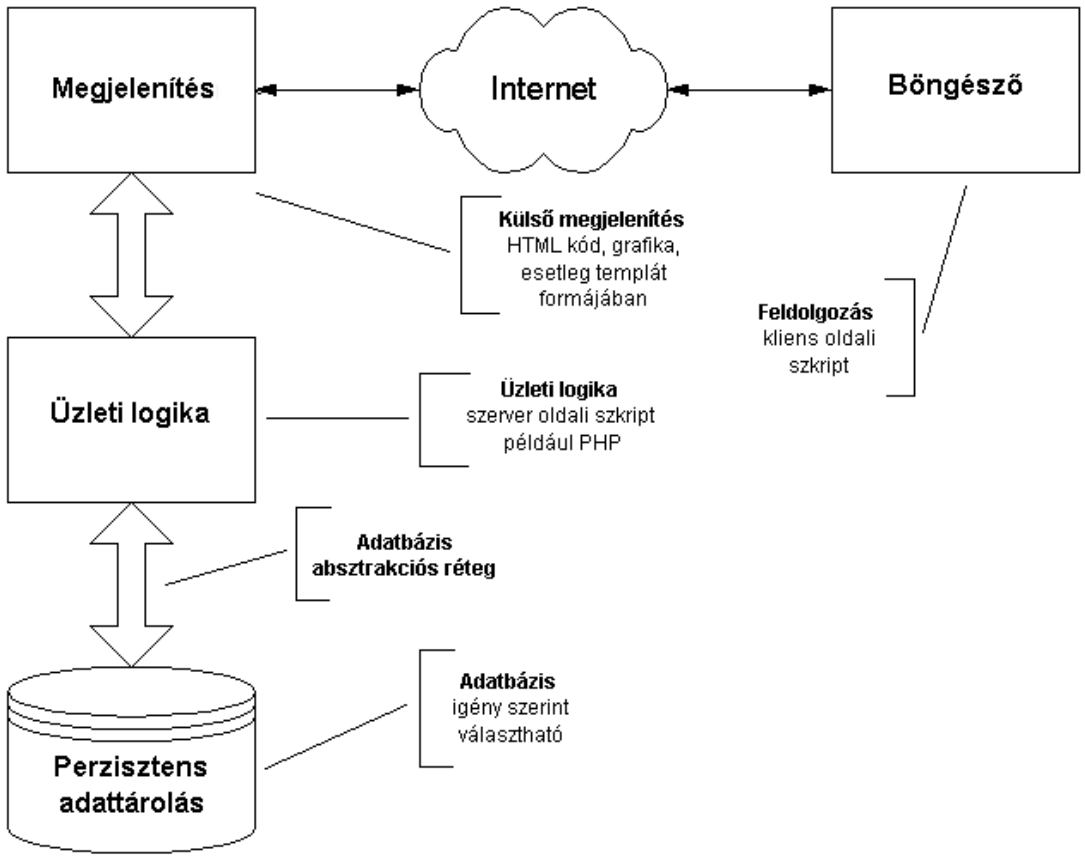
Kliens - szerver architektúra

A nagyközönség számára hozzáférhető információ döntő többségét ma az internet szolgáltatja. A fogalom mögött egy többszörösen - redundáns módon - összekapcsolt világméretű számítógép hálózat áll, ennek csomópontjait alkotják az egyes tartalomszolgáltató szerverek és az ezek azonosítását szolgáló névszerverek. A tartalomszolgáltató számítógépek illetve a rajtuk futó szerver funkciót ellátó alkalmazások a kliens gépektől beérkező kérések alapján a HTTP[1] protokoll szerint fájl lekéréseket hajtanak végre és ezeket a fájlokat az egyes kliens számítógépeken futó alkalmazások, a böngészőprogramok értelmezik. Amennyiben szükséges, a tartalomszolgáltató szerverek adatbázis szerverekhez is fordulhatnak a kért információért. Az információ hagyományos dokumentum jellege viszont nem teremti meg a lehetőséget gyorsan változó, dinamikus adatok beillesztésére és így már több éve, exponenciálisan fejlődve nő azon alkalmazások száma, amelyek a beérkező HTTP kérésekre válaszul nem statikus, hanem a kérésre összeállított dokumentumot küldenek el a böngészőprogramnak. Természetesen ebben a dinamikusan összeállított dokumentumban már helyet kaphatnak adatbázisokban tárolt vagy számított adatok is. Ennek megfelelően beszélünk ma kétrétegű (two-tier) és háromrétegű (three-tier) alkalmazásokról[2], ahol a kétrétegű alkalmazás alatt értjük a kétoldalú kliens-szerver típusú alkalmazást és háromrétegű alkalmazásnak nevezzük azt, ahol egy harmadik szolgáltató, egy adatbázis szerver is bekapcsolódik az információ előállításába.

---

[1] HTTP - hypertext transfer protocol; a hipertextes alkalmazások átviteli protokollja

[2] A jegyzetben a többé-kevésbé meghonosodott réteg elnevezést használom, bár ez nem fedi teljesen az eredeti angol szakkifejezést. Nagyon fontos megjegyezni, hogy ez a réteg elnevezés semmiféle kapcsolatban nincs az OSI rétegmodell elemeivel, nincs egymásra épülés, inkább azt kívánja kifejezni, hogy több, egymástól független, speciális feladatokat ellátó rendszer együttműködésén alapuló alkalmazásról beszélünk.



1. ábra. Az első réteget képezi a megjelenítés, az internettel mint szállító közeggel és a böngészővel, mint a renderelés eszközével. A második réteg az üzleti logika, ez tartalmazza szerver oldali szkriptjeinket és a harmadik réteg az adattárolás eszköze, a megfelelően választott adatbázis.

Természetesen ez csak elvi megközelítés, az egyes szerverfunkciót betöltő és kliens programok szélső esetben azonos fizikai egységen is futhatnak - például egy fejlesztőrendszer vagy oktatórendszer esetében - és futtathatók egymástól távol eső földrajzi helyeken is. Az egymástól távol elhelyezkedő egységek közötti kommunikációt ebben az esetben biztosíthatja az Internet fizikai hálózata vagy kisebb távolságok esetén helyi hálózatok is (intranetes alkalmazások egy-egy intézményen belül, LAN[3]). A felhasználók és a fejlesztők szempontjából ez gyakorlatilag mindegy, amennyiben a kom-

[3] LAN Local Area Network; helyi hálózat

munikáció a TCP/IP[4] protokoll és az erre épülő HTTP protokoll alapján történik, az információt szolgáltató alkalmazás működési elve minden esetben azonos lesz és az alkalmazás a hálózat geometriájától és kiterjedésétől függetlenül létrehozható.

A kliens gépen futó böngészőprogramból indított kérésekre válaszul leküldött dokumentumok feldolgozásához csupán a dokumentumok leíró nyelvét, a HTML[5] nyelvet kell a böngészőnek értelmezni és ennek megfelelően megjelentetni a dokumentumot a kliens gép képernyőjén. Nem igényel különösebb magyarázatot, hogy egy, a dokumentum szerkezetét, külső megjelenését leíró nyelv nem alkalmas arra, hogy változó tartalmat hordozzon és jelenítsen meg, valamint lehetővé tegye a felhasználó által bevitt adatok manipulálását. Erre azonban kétségtelenül szükség van és így adatbeviteli funkciót már a HTML nyelv korai változataiba is beépítettek, ezzel lehetővé téve a felhasználói adatok bevitelét a böngészőprogram segítségével. Ez az eszköz a HTML űrlap (form) objektuma és annak gyermekobjektumai, az egyes specializált beviteli mezők, amelyek állandó fejlődésben vannak a böngészőprogramokkal együtt, hiszen ezek megjelenítését a böngésző végzi.

Ha jól megfigyeljük, egy első ránézésre meglehetősen bonyolult és bizonytalan működésű rendszer áll tehát a rendelkezésünkre. Van ebben a rendszerben egy vagy több szerver, amely(ek) dokumentumokat tudnak szolgáltatni. Ezek tetszőleges rendszerbe összefűzhetőek a hipertext elemek segítségével, a kliens gépen megjelenítendő dokumentum egyes elemei több szerverről is származhatnak. A HTML által biztosított navigáció segítségével tetszés szerinti számú szervert járhatunk be, de nem tudjuk ellenőrizni, hogy a kívánt információ a lekérés időpontjában rendelkezésre áll-e. Bár lehetőségünk van szinte bármilyen információ (numerikus és szöveges adatok, kép-, szöveg-, multimédia fájlok) felvitelére az egyes szerverekre a kliens gépen futó böngészőprogramok segítségével, ez az információ nem érhető el közvetlenül, az adatok feldolgozása a dokumentum-kezelés szintjén nem valósítható meg. Bár fontos és kényelmes lenne a fogadó oldal, a kliens számítógép képességeinek ismerete, azok fel- és kihasználása, ez biztonsági okokból nem, vagy csak erősen korlátozottan valósítható meg - egy szerver ne kezelhesse a kliens számítógépet, ne kezeljen fájlokat, ne indíthasson programokat

[4] TCP/IP Transmission Control Protocol/Internet Protocol; az Internet rendszerében alkalmazott csomagküldésen és -fogadáson alapuló átviteli protokoll.

[5] HTML HyperText Markup Language; hipertext leíró nyelv.

rajta. Minden adatátadás, kommunikáció szükségszerűen két vagy több számítógép között zajlik, amelyeket ennek megfelelően kell védeni és az adatokat is ellenőrizni kell. A biztonsági elveknek megfelelően csak a kliens gép indíthasson dokumentum lekéréseket - HTTP lekérések - ezeket a szerver szolgálja ki, de regisztráljon minden kérést. Kínáljon fel szinte minden kényelmi lehetőséget a lehető legmagasabb szintű kiszolgálás érdekében, de akkor is megfelelő legyen a "tálalás", ha a kliens oldalon nem áll rendelkezésre az ideálisnak tekintett háttér. Fontos az is, hogy a gyorsabb "ügyintézés" érdekében a lehető legtöbb, gyors reagálást igénylő választ és lehetőséget a kliens oldalon kezeljünk le, számolva a gépek közötti kommunikáció szűk keresztmetszetével (például modemes kapcsolat).

Amire tekintettel kell lenni alkalmazás-fejlesztés során

Fentiek teljes körű megvalósítása egy háromrétegű webes alkalmazás keretei között nem egyszerű dolog, és biztos, hogy több különböző, specializált eszköz működését kell összehangolnunk a jó eredmény érdekében. Nézzük meg tehát, hogy mi mindenre kell figyelmet fordítani egy tipikus webes alkalmazás elkészítése során.

- A felhasználó oldalán olyan böngésző programok állnak rendelkezésre, amelyek típusa és ebből adódóan képességeik széles skálán változnak, ezért gyakorlatilag lehetetlen minden eshetőségre felkészíteni az alkalmazásunkat (lásd Microsoft Explorer, Netscape Navigator, Mozilla, Opera és mások, valamint ezek gyakrabban előforduló verziói, az IE 5.0-tól és az NN 4.73-tól kezdve a 6.0 és 7.0 verziókig kb. 6-8 különböző eszköz, nem beszélve a vakok által használható speciális böngészőprogramokról).
- Számos esetben a kiegészítő információ hordozók megfelelő megjelenítéséhez különböző lejátszó és kiegészítő (plugin) programokra van szükség (Flash, mozgókép, hang, stb.), ezek hozzáférését illik megadni.
- A felhasználó egyéni meglátásai és indítékai alapján ki- illetve bekapcsolhat kliens oldali eszközöket (JavaScript, cookies és különböző Java értelmezők). Ezek alkalmazását vagy kerülni kell, vagy az alkalmazásunk feltételeként meg kell adni - és persze a feltétel nem teljesülését is le kell kezelni.

- A felhasználó oldaláról az adatbevitelt ellenőrizni kell és ezt felhasználóbarát módon kell elvégezni, részben korlátozva a bevihető adatokat (legördülő listák) részben kliens oldalon ellenőrizve (ennek eszközei viszont kikapcsolhatók, lásd JavaScript) és a szerver oldalon feltétlenül újra ellenőrizni kell, tekintettel a kliens oldal manipulálhatóságára. Nem szabad elfelejteni, hogy a kliens oldalon egy űrlap (form) tetszés szerint megváltoztatható a kód újraírásával.
- A webes alkalmazáshoz nincs kidolgozott súgórendszer, tehát a felhasználó informálását és támogatását a web szövetébe kell ágyazni, a navigálás kialakításánál egyértelműségre kell törekedni.
- A felhasználókat azonosítani is kell és esetenként személyre/kategóriára szabott megjelenítést kell biztosítani - ehhez a belépéseket és a navigálást regisztrálni, követni kell szerver oldali eszközökkel.
- A webes alkalmazásoknak annyi belépési pontja lehet, ahány különböző URL[6] létezik az adott webhelyen belül. Ezek mindegyikét figyelni, kezelni kell. Ennek megfelelően természetesen a kilépési lehetőségek száma is nagy, minden egyes oldal lefutásakor gondoskodni kell például a fájlok, adatbázisok zárásáról.
- A böngészők, az azokat futtató operációs rendszerek alapértelmezetten rövid és hosszú távon tárolják a meglátogatott weboldalak URL címeit és navigációs eszközöket biztosítanak (előre és vissza gombok, history). Ezeket szintén figyelembe kell venni, védett oldalak esetében kezelni kell a tárolt adatok alapján megkísérelt belépéseket.
- A kliensre leküldött HTML kód és fájlok publikusnak tekintendők. Sem kép, sem kód nincs biztonságban, ezekben kényes információ nem helyezhető el (jelszó, azonosító, rejtett mezők, JavaScript kód). Az időnkét a webes fórumokon felreppenő hiedelmekkel szemben nincs lehetőség képek, fájlok védelmé-

[6] URL Uniform/Universal Resource Locator, az egységes forrás azonosító, az Internetes források (weblapok) egyedi címe. Újabban helyette előszeretettel használják az általánosabb értelmezésű URI-t, az Universal Resource Identifier elnevezést, a napjainkban leggyakrabban használt URI az URL.

re, maximum a laikus felhasználó dolgát lehet kismértékben megnehezíteni, de a jó öreg Print Screen ellen semmi sem véd meg…

- A kliens - webszerver közötti kommunikáció lassú és megbízhatatlan, és amennyiben az adatbázis szerver külön gépen, külön URL címen helyezkedik el, az adatbázis - webszerver közötti kommunikációra is ugyanez áll. Ezért mindig figyelembe kell venni a hálózat teljesítőképességét, modemes kapcsolat esetén nem tervezhetjük nagymennyiségű információ gyors leküldését. Ugyanakkor mindig számolni kell a webszerver és az adatbázis szerver időszakos elérési nehézségeivel. Lehetőleg ilyenkor ne az eszközök beépített hibaüzenetei kerüljenek elő, hanem a felhasználókat informáló hibajelentések.

Amennyiben az alkalmazás főleg dinamikusan változó adatokkal dolgozik, és az ember dokumentum-szerkesztő tevékenységét a minimálisan szükséges mértékre akarjuk visszaszorítani, ezeket az alapelveket fokozott mértékben kell betartani. Nyilvánvaló, hogy statikus adatok esetében több lehetőség van azok megfelelő és speciális "tálalására" míg dinamikus adatok esetében nehéz az általánosan alkalmazható keretrendszer kialakítása. Ezek egyensúlyától függ az alkalmazás minősége. Az alábbiakban egy gyors áttekintést adunk az egyes eszközökről, azok szerepéről és helyéről a komplex webes alkalmazások fejlesztésben.

JavaScript, VBScript - az "egyetlen" kliens oldali eszköz

Ha egyetlen, akkor miért kettő? merülhet fel a jogos kérdés, a válasz pedig egyszerűen annyi, hogy a két szkriptnyelv ugyanazt a feladatot látja el, ám a JavaScript népszerűsége, ismertsége jóval nagyobb - körülbelül 20:1 arányban találunk a weben JavaScript-tel foglalkozó oldalakat a VBScript oldalakkal szemben. Mindkét nyelv természetesen megfelel céljainak, nem kívánok egyik mellett sem állást foglalni, ez éppen olyan értelmetlen lenne, mint a legjobb programnyelv meghatározása. Míg a JavaScript egy C-nyelv típusú objektum-alapú nyelv, a VBScript nevének megfelelően a Basic nyelvcsalád tagja, így az egyes strukturális elemeket ennek megfelelően jeleníti meg. Előrebocsátom, hogy a JavaScript nyelvnek semmi köze a Java nyelvhez, csupán az ősük, a C-nyelv azonos és a logikus fejlődés hozza ezeket közel egymáshoz. Mivel a tantárgy keretében fő nyelvként egy másik C-típusú nyelv, a PHP oktatására kerül sor, ezért fontosabbnak tartom a JavaScript ismertetését a szükséges minimális mértékben, hogy az elkerülhetetlen esetekben alkalmazása ne okozzon problémát.

Természetesen a JavaScript nem szerver oldali eszköz, bár vannak próbálkozások a nyelv kiterjesztésére szerveren futó alkalmazásokra is. Bár ez az egyetlen interaktív eszköz, ami a kliens oldalon futtatható, nem tekinthető teljes programozási nyelvnek, biztonsági okokból teljesen hiányzik belőle a fájlkezelés megvalósítása, nem oldható meg olvasás, írás művelet sem. Ugyanakkor számtalan alkalmazás megvalósítható segítségével, az egyszerű eseménykezelésektől a komplex játékprogramokig. JavaScript segítségével valósítható meg az összes kliens oldali interaktív tartalom, a különböző menüktől kezdve az űrlapok ellenőrzéséig - ennek korlátjait már említettem - és a felhasználó választásának megfelelően módosuló tartalmakig. A böngésző dokumentum (document) és window objektumait is kezelni tudjuk JavaScript segítségével, de a webes alkalmazásokban gyakran használt ablaknyitások, egérmozgásra érzékeny tartalmak, egérkattintásra reagáló elemek megvalósítása is JavaScript segítségével történhet.

Olyan korszerű rendszerek, mint például az IBM Websphere alkalmazása is JavaScript nyelvet használ a kliens oldalon. Kihasználható a szerver és a kliens oldali programnyelvek és a HTML szinte korlátlan keverhetősége, nem ritka egy olyan alkalmazás, ahol például az adatbázisban tárolt információ alapján egy szerver oldali nyelv JavaScript kódot állít elő, ez a klienstől induló kérésre lekerül a felhasználó gépére és az ott futó böngészőprogram a JavaScript által előállított HTML kódot alapján jeleníti meg az információt. A kliens oldali interaktivitás kulcsa is az, hogy a JavaScript által előállított HTML kód értelmezhető a böngészőben, illetve a DOM[7] biztosít olyan elemeket, amelyek értéke változtatható, így a megjelenített tartalom is változik ezek felülírásával. Ezen alapszik sok egyszerűbb vagy bonyolultabb, ám rendkívül hatásos webes alkalmazás kliens oldali eleme - órák, számológépek, értékszámító űrlapok, konverterek.

Nem hallgathatjuk el azt sem, hogy mivel a JavaScript a böngészőben fut, annak típusa, változata nagymértékben befolyásolja a JavaScript alkalmazhatóságát. A JavaScript-es alkalmazások külön fejezetét alkotják a böngésző diagnosztikák, olyan

[7] DOM Document Object Model, dokumentum objektum modell. Az újabb típusú böngészők számára a HTML dokumentum nem egyszerű szöveges állomány, hanem elemei - egymással faszerkezetű kapcsolatban állva - egy erősen hierarchikus objektumot alkotnak, ezek csomópontjai hivatkozások alapján közvetlenül is elérhetők. Egy űrlap (form) például a dokumentum gyermekobjektuma, nevével hivatkozható és elemei - az egyes beviteli mezők - szintén hierarchikusan alárendelt gyermekobjektumok. Így a document.form.szovegmezo objektum tulajdonságai változtathatók, a .value értéke változtatható. Hasonlóképpen hivatkozhatunk egy névvel ellátott anchor tag-re is.

programok, amelyek a böngésző eltéréseket igyekeznek figyelembe venni és az eltérő böngészőkben közel azonosan működő kódot tudnak előállítani. Nem szabad arról sem megfeledkezni, hogy az egyes böngészők másként értelmezik a dokumentum objektum modellt is, ezzel fokozva a zűrzavart…

A jegyzet célja nem a JavaScript ismertetése, így nem tárgyalhatjuk részletesen a gyakorlati alkalmazásait. Számos oktatóprogram, könyv áll rendelkezésre és az egyes webes források kimeríthetetlen tárházai a JavaScript-es alkalmazásoknak. Mint erősen objektum alapú nyelv, ismerete értékes segítséget ad alkalmazásaink magas szintű megvalósításában de minimális ismerete nélkül egyszerű felbukkanó ablak nyitások vagy egy, az egérmozgatás hatására dinamikusan változó link sem hozható létre. Bár a legtöbb HTML fejlesztőeszköz biztosít egyszerű Javascript alapú szkripteket a rutinfeladatok megoldására, programozási tudás nélkül ezek csak a fejlesztőeszköz által korlátozott körben alkalmazhatóak.

Általában sok problémát okoz, hogy kevesen ismerik jól a JavaScript szintaktikát és a nyelvbe beépített objektumokat, függvényeket. Az alábbiakban megadok egy webes hivatkozást, ami - mivel a JavaScript kifejlesztőinek weboldalára mutat - autentikusnak tekinthető referenciákat tartalmaz:
[http://devedge.netscape.com/central/javascript/]
Az angol nyelvű JavaScript 1.5 referencia gyűjtemény és a JavaScript 1.5 nyelv leírása (kézikönyv) letölthető a következő magyarországi honlapról is:
www.medzi.hu/javascript

A szerver oldali programozás eszközei

Adatforrások

Dinamikus webes alkalmazások készítése nem oldható meg adatok tárolása nélkül. Erre elvileg két lehetőségünk van, az adatok tárolhatók fájlokban, akár egyszerű szöveges fájlokban - ebben az esetben az adatok kikeresését le kell programoznunk és a felvitelüket is meg kell oldani - vagy tárolhatók adatbázisokban - ebben az esetben az adatok kinyerése és rögzítése az adatbázis motor feladata és mivel feltételezhetően relációs adatbázisokat fogunk használni, ez SQL parancsokkal egyszerűen megoldható. Nagyon sok esetben ez a kettő keveredik is, mert vannak olyan adatok, amelyeket nem érdemes egyszerűbb alkalmazások esetében adatbázisba rakni, elég ha írható-olvasható

fájlban tároljuk. Ilyen lehet egy szimpla látogató számlálás vagy fájlokból olvashatók ki az egyes legördülő menük választható tételei is, mint például országnév listák vagy hasonló egyszerű listaadatok. Természetesen mindez megoldható adatbázisból is, de sok esetben az adatbázis hozzáférés korlátozott az egyes internetes szolgáltatóknál és ilyenkor a lehetőségek szabják meg az alkalmazásunkban felhasználható területeket. Persze egy komolyabb alkalmazás esetében nem kerülhető meg az adatbázis alkalmazása hacsak nem akarunk saját adatbázis-kezelőt írni…

Az egyes rendelkezésünkre álló programnyelvek számos adatbázis rendszerrel képesek együttműködni, így erről az oldalról gyakorlatilag nincs korlátozás. Az adatbázis rendszer kiválasztásánál azonban érdemes néhány alapvető szempontot megvizsgálni.

Általában relációs adatbázisokat használunk, de olyan alkalmazás esetében, ahol csak igen egyszerű adatokat kell kezelni, egy asszociatív kulcs-érték párokon alapuló adatbázis tökéletesen elegendő lehet. Erre alkalmas a PHP dba_*** függvénycsomagja, ezek segítségével kezelhető számos fájl alapú adatbázis. Mielőtt azonban ezt a függvénycsomagot használni kívánjuk a jelen fejlesztőkörnyezetben, a php.ini fájlban (C:/apache/php/php.ini) ki kell venni a megjegyzés (komment) „;” jelét a php_dba.dll kiterjesztés (;extension=php_dba.dll) elől, hogy a rendszer indításakor ez a modul is regisztrálva legyen.

Milyen adattömeget kell kezelnünk? Gyorsan változó és nagymennyiségű szöveges adat (például napilap és folyóirat cikkek) kezelése esetében ajánlatos olyan adatbázis-kezelőt választani, amely egyszerűen és gyorsan képes kezelni hosszabb szövegeket, biztosítja azok kereshetőségét.

Mekkora a célközönségünk, akik az adatokat használni fogják? Azaz időegység alatt mennyi lekérdezést kell kiszolgálnunk. Ennek megfelelően kell jól terhelhető adatbázis rendszert választani.

Mennyire értékesek az adatok? Természetesen minden adat értékes, de van olyan eset, amikor nem engedhető meg, hogy egy kombinált tranzakciót ne lehessen visszagörgetni annak hibája, megszakadása esetében - képzeljük el mi történne, ha egy banki tranzakció végrehajtása során a pénzösszeget az egyik számláról már leemeljük, de mielőtt a másikon jóváírnánk, megszakad a tranzakció és az összeg elvész - ilyen esetben célszerű olyan adatbázis kezelőt választani, amelyben a tranzakció kezelés beépített funkció.

Milyen bonyolult lekérdezéseket kell végrehajtani? Nem mindegy, hogy relációs adatbázis kezelőnk képes-e belső select SQL utasításokat végrehajtani vagy tudja-e kezelni az idegen kulcsokat.

Melyik adatbázis kezelőre van széleskörű támogatás különböző operációs rendszerekben? A webes alkalmazások esetében egy-egy adott esetben természetesen tudjuk, hogy milyen webszerverre fejlesztünk, de törekedni kell arra, hogy a kész komponenseinket lehetőleg többször is fel tudjuk használni. A két legelterjedtebb webszerver ma a Linux/Unix/FreeBSD rendszeren futó Apache (kb. 60% piaci részesedéssel) és a Windows 2000 szerveren futó IIS rendszer (kb. 30% piaci részesedéssel). Ezek erősen behatárolják az alkalmazható adatbázis rendszert is.

Fentieket figyelembe véve a két leginkább ajánlható relációs adatbázis kezelő ma a MySQL a hagyományos, különös biztonságot nem igénylő webes alkalmazásokra és a PostgreSQL a nagy biztonságot igénylő alkalmazásokra. Ez utóbbi objektum relációs adatbázis kezelő nyelv, így speciális megoldásokat - a táblák öröklődését - is létrehozhatunk a segítségével. Ismertebb rendszer még webes fejlesztésekre az MSSQL, az Informix vagy a DB2 adatbázis-kezelő is. Ezekről részletes információ szerezhető az Internetről, sőt az irodalomjegyzékben megadjuk a MySQL - PostgreSQL rendszerek összehasonlító elemzésének referenciáját is.

CGI programozás

Még a speciálisan webes célra kifejlesztett nyelvek megjelenése előtt felmerült az igény adatfeldolgozásra és az adatok alapján különböző kimenetek készítésére. Bármely programnyelven megoldható az, hogy a program kimenete egy HTML fájl legyen, amit viszont a böngészőprogramok képesek értelmezni, így végső soron bármely programnyelv segítségével készíthető webes kimenet. Ennek megvalósítására azonban biztosítani kell, hogy egy űrlapról beérkező adatok egységes formában, paraméterként eljussanak a programhoz, és ez a program a kimenetet a webszerveren keresztül, HTTP kérésre juttassa el a böngészőbe. Ehhez hozták létre a CGI-t (Common Gateway Interface, közös kapuinterfész) amely ezeket az igényeket képes kielégíteni és a tetszőleges nyelven megírt és lefordított programok (Pascal, C, C++, Java, Delphi, stb.) a webszerver cgi-bin könyvtárából futtatva létrehozzák a kívánt HTML kimenetet. Természetesen a program meg tud valósítani adatbázis kapcsolatot is és így megoldható az adatbázisba történő adatbevitel vagy annak lekérdezése is.

A rendszer előnye, hogy tetszőleges nyelv alkalmazható, azaz mindenki az általa jól ismert programnyelven dolgozhat, az eredmény pedig nyelv-független lesz. Hátránya, hogy minden egyes módosítás a lefordított program újraírását, -fordítását igényli és minden esetben egy teljes weblap lesz a kimenet, nem oldható meg az így nyert információ beépítése egy már kész weboldalba. További hátránya az is, hogy a nem interpretált nyelveken írt CGI programok nem hordozhatóak, különböző operációs rendszerekre különböző programokat kell írni (pontosabban külön-külön le kell fordítani az egyes programokat). Talán nem véletlen, hogy ezt az eszközt ma már inkább a számításigényesebb alkalmazások esetén használják és visszaszorul a szerver oldali nyelvek mellett. Ugyanakkor van bizonyos specifitása is. Míg nagyon ritkán találunk CGI-ként futtatott PHP programot, a Perl programok főleg itt történő futtatásra készülnek és elvétve találunk webszerver modulként futtatott Perl szkripteket. Ennek igazi racionális alapja nincs, egyszerűen így alakult ki…

Perl

A Perl (Practical Extraction and Report Language) magas szintű szkriptnyelvet Larry Wall hozta létre 1987-ben. Mint nyílt forráskódú nyelv, azóta nagymértékű fejlesztésen ment keresztül. Több őse lelhető fel, legnagyobb mértékben a C-nyelvre támaszkodik de sok más nyelvből is örökölt elemeket és ma már több platformra kifejlesztették. Mint szkriptnyelv, természetesen a Perl kód platform független (persze csak akkor, ha nem használunk speciális platform függő elemeket a kódban) és a megfelelő interpreter egyes operációs rendszerekben szinte beépített elemnek tekinthető.

Legnagyobb erőssége a sztringek kezelésében van, a beépített reguláris kifejezés[8] támogatás a szöveges adatok szinte egyedülállóan jó feldolgozó nyelvévé teszik. Ezt a reguláris kifejezés támogatást napjainkban már több más szkriptnyelv is átvette a népszerűsége miatt. A Perl legjelentősebb alkalmazási területe a CGI szkriptek írása. Primitív adattípusai rugalmasan egymásba alakíthatók, ami persze növeli a programozó felelősségét és a Perl-ben jelent meg először a sztringgel indexelt tömb - asszociatív tömb - mint aggregált adattípus. A nyelv kezeli az objektumokat, de strukturáltan is

[8] reguláris kifejezés - olyan jel és karakter kombinációk, amelyek segítségével kijelölhetők adott mintának megfelelő karaktersorozatok. Ennek legegyszerűbb megjelenési formája a Windows/DOS rendszerek keresőiben használható helyettesítő karakterek, a (*) ami tetszés szerinti számú karaktert helyettesíthet és a (?) ami egy karakter helyett állhat.

programozható. A Perl dinamikusan kezeli a memóriát (a Lisp örökség része) és a szemétgyűjtést (a már nem referált változók által elfoglalt memóriaterület felszabadítását).

Mint minden nyílt forráskódú nyelv, folytonosan fejlődik és nehéz a jövőjét megjósolni, erősen függ a nyelv adottságain túl a divattól és egyéb faktoroktól. Az biztosan állítható, hogy a CGI programozásban még hosszú ideig nem veszít a fontosságából, viszont nem lesz belőle univerzális webes szerver oldali programnyelv. A nyelv szabadon és anyagi ellenszolgáltatás nélkül felhasználható bármely alkalmazásra, de természetesen vannak megvásárolható fejlesztőrendszerei is, amelyek sok extra szolgáltatást biztosítanak a fejlesztőmunkához.

Python

A Python programnyelvet 1990-ben kezdték kifejleszteni. A nyelv maga egy interpretált, hordozható, objektum-alapú programnyelv, amelynek C-ben illetve C++-ban írt nyílt forráskódja igény szerint kiegészíthető, módosítható. Nehéz nyelvcsaládba sorolni, mert bár mutat C-szerű elemeket, nagyon sok esetben a szintaktikában egyéni utakat járnak a fejlesztők. Elsajátítása viszonylag könnyű, és interpreterjét már mindegyik jelentősebb operációs rendszerre kifejlesztettek. C-programozói tudással tetszés szerint új függvények, osztályok adhatók hozzá [http://www.python.org/].

A Perl nyelvhez hasonló módon erőssége a sztringkezelés, kezelni képes a reguláris kifejezés illesztéseket. Beépített memóriakezelése egyszerűsíti a programozó munkáját és egyszerű szintaktikája lehetővé teszi alkalmazását prototípusok kifejlesztésében illetve ideális kisebb programozói feladatok ad hoc megoldására.

Nagyobb programok is megírhatók Python nyelven, ezt segíti elő a nyelv kivételkezelése és a számos már kifejlesztett kiterjesztési modul - ezek egy része standard eszköz, mint például a matematikai könyvtár, mások alkalmazás-speciális feladatokat látnak el, mint például a képfeldolgozó vagy hangfeldolgozó modulok. Beépített módszerek vannak a nyelvben egy objektum bájtsorozattá alakítására illetve ennek fordított műveletére, így implementálhatók különböző elosztott objektum modellek a nyelvben.

Teljes mértékben szabad felhasználású nyelv, szabadon alkalmazható és terjeszthető kereskedelmi alkalmazásokban is a szerzői jogosultság feltüntetése (copyright) mellett.

Java

A Java egy általánosan alkalmazott platform független, objektum orientált nyelv, amely több szerepben is megállja a helyét a webes alkalmazásokban. Alkalmazzák mint CGI programozási nyelv, de jelentős tért hódított magának mint beépülő nyelv is. Az úgynevezett Java Server Pages (JSP) oldalakon a HTML kóddal keverve felhasználható dinamikus tartalmak megjelenítésére [http://java.sun.com/webservices/index.jsp]. Ezeket egészítik ki a Java alapú szervlet alkalmazások, amelyek nevüknek megfelelően a szerveren futó Java alkalmazások, az ezekben létrehozott osztályok, objektumok viszont elérhetők a JSP-k által is. Így ennek a kettőnek összekapcsolásával komplett alkalmazások írhatók, kiegészítve a Java Enterprise Bean technológiával, amelyek többé-kevésbé önálló, egy-egy speciális feladatot - például adatbázis kapcsolat - ellátó programok. Bár ránézésre - kódszinten - a JSP a PHP-vel összevetve éppen olyan beépülő, függetlenül értelmezett nyelvet sejtet, a JSP oldalak feldolgozása speciális módon történik.

A Java kóddal kombinált oldalak kezelésére egy külön webszervert is létrehoztak [Tomcat szerver verziók: http://jakarta.apache.org/tomcat/] amely több platformon is telepíthető teljes értékű webszerver, kezelni képes az egyszerű HTTP kéréseket, de a beépített moduljaival JSP oldalak lekérése esetén ezeket feldolgozza és először szabályos .java fájl kimenetet készít belőlük. Ebben az eredeti, csupán az adatok külső megjelenítésének leírására használt HTML kódokat már Java-specifikus kiíró utasítások (out.write()) veszik kézbe, majd a szokásos Java technikának megfelelően lesz belőlük .class fájl és végül ezt értelmezi a Java Virtuális Gép (JVM). Ennek segítségével kezelni tudja a szervlet kódokat is, így teljes körű szolgáltatásokat nyújt a Java alkalmazásokhoz. Ez a webszerver több korszerű fejlesztői rendszer beépített eleme és komoly erőfeszítéseket tesznek a folyamatos fejlesztésére. Bár önállóan is telepíthető, Apache webszerver hozzáférhetősége esetén érdemesebb annak moduljaként telepíteni, mert az egyszerű HTTP lekéréseket (sima HTML oldalak, beépülő tartalom nélkül) az Apache gyorsabban képes kiszolgálni.

A Java nyelv szabadon felhasználható, a megfelelő operációs rendszeren a fordító által előállított bájtkódot értelmező interpreter szabadon letölthető és a fejlesztőkészlet (JDK) is szabadon felhasználható, ahogy ugyanilyen hozzáférésű a Tomcat webszerver is. Természetesen komolyabb alkalmazások fejlesztése esetén szükség lehet a munkát

lényegesen meggyorsító integrált fejlesztői környezetre és ez már jelentős mértékű anyagi befektetést jelent, amely az alkalmazás-fejlesztés felgyorsításával térülhet meg.

ASP

Az Active Server Page (ASP, aktív szerver oldal) olyan szerveren tárolt weblap, amely a kódjában a szokásos HTML elemeken túl a szerveren feldolgozásra kerülő kódot is tartalmaz. Ezek kiterjesztése .htm vagy .html helyett .asp és ebből tudja a webszerver, hogy a HTTP kérésre nem csak le kell küldenie a megfelelő .asp oldalt, hanem azt fel is kell dolgozni. A feldolgozó modul a <% %> jelek közé zárt kód helyett az előállított HTML kimenetet küldi el a kliens böngészőprogramnak, amit az a szokásos módon értelmez. A Microsoft Corporation által kidolgozott technológia lehetővé teszi dinamikus tartalmak beillesztését, az egyszerű dátum és felhasználói információtól az adatbázisból nyert adatokig. Természetesen lehetőség van a felhasználótól kapott adatok (űrlap adatok) szerver oldali feldolgozására is az ASP technológiával.

Az ASP technológia lehetővé teszi a kliens oldali és a szerver oldali kódok teljes keverését, azaz mindegyiket az arra legalkalmasabb eszközzel készíthetjük el. Népszerűsége jórészt ennek köszönhető és futtatása nem okoz gondot IIS alkalmazása esetén. Nem platform független, bár az Apache webszervernek is létezik ASP interpreter modulja, így Linux rendszereken is jól futtatható. Természetesen az ASP eredeti környezete a Windows környezet és különösen a .NET alkalmazások használják előszeretettel, sőt gyakorlatilag kizárólag az ASP technológiát alkalmazzák.

Bár maga a kód - hiszen nem nevezhető külön, önmagában is alkalmazható nyelvnek, inkább csak egy hatékony technológia Basic programozási alapokon - minden korlátozás nélkül előállítható, amennyiben Windows operációs rendszeren akarjuk futtatni, az operációs rendszer szoftver licencét meg kell vásárolni, különösen, ha kereskedelmi célra kívánjuk alkalmazni. Ez persze elkerülhető Apache webszerver alkalmazásával. Így a szabadon felhasználható szoftverekkel összevetve egyszerű anyagi okokból háttérbe szorul a kereskedelmi alkalmazásoknál. Maga a technológia ismerete azonban mindenképpen előnyös, könnyen elsajátítható és komplex alkalmazások készíthetők el használatával. Kedvcsinálónak álljon itt két rövid, átfogó oktatóprogram, amely kellő betekintést biztosít a technológiába: [http://www.asptutorial.info/; http://www.w3schools.com/asp/default.asp].

A PHP nyelv

A tantárgy gyakorlati részének főszereplője az 1994 óta fejlesztett PHP nyelv, ennek segítségével készítjük el alkalmazásainkat, így ezt a szkriptnyelvet részletesebben ismertetjük. A betűszó először a Rasmus Lerdorf által készített Personal Home Page Tools-ra utalt, majd ennek teljes átdolgozása óta rekurzív módon a PHP Hypertext Preprocesszor (PHP hipertext előfeldolgozó) szavak kezdőbetűit rejti, de dinamikus fejlődése során már messze eltávolodott ettől a szimpla jelentéstől. Nyílt forráskódú, objektum alapú szerver oldali szkriptnyelv, szabadon felhasználható bármely web alkalmazás esetében. Lehetőséget biztosít a PHP parancsok közvetlen beágyazására (<?php *utasítások* ?> vagy a szintén használt rövid szintaxis <? *utasítások* ?>) a HTML kódba[9], ezeket a szerver oldali szkript nyelveknél már megszokott módon a webszerverbe beépülő interpreter modul értelmezi és az előállított HTML kimenetet a leküldésre kerülő weblapba illeszti. Természetesen nem csak a HTML kód módosítható, hanem például a szintén a böngésző által értelmezett Javascript kód is, így maga a kliens oldali Javascript program is dinamikusan változtatható.

A PHP szintaxison erősen látható a C-nyelv öröksége, de maga a nyelv is C-ben illetve C++-ban íródott. Átvette a Perl nagy erősségét, a reguláris kifejezések használatát, így ugyanolyan szintű szöveg manipulációk hajthatók végre a segítségével. A nyílt forráskód lehetőséget ad új speciális modulok, metódusok elkészítésére és a széles fejlesztői bázis alapozza meg a PHP nyelv gyors fejlődését. Programozóknak határozottan érdemes meglátogatni a [www.zend.com] webhelyet, ahol megtekinthető az összes fejlesztés alatt levő metódus is C forrásfájl formájában.

A PHP 3 verziót 1998-ban adták ki és gyors karriert futott be a webes alkalmazások terén. A sikerre való tekintettel átdolgozták az interpretert - Zend engine - és a PHP 4 már az új interpreterrel került a felhasználók kezébe. Mivel a Zend engine nem egy egyszerű interpreter, hanem inkább a Java JIT interpreter elveit alkalmazza, (így például az összes függvény előre lefordításra kerül, vagy az egyes kódrészletek gyakorlatilag tárolásra kerülnek és nem szükséges ezeket - például egy ciklus magját - újra és újra

[9] A PHP nyelv alkalmazói nem egységesek ennek előny/hátrány megítélésében. Bár jómagam az esetek többségében előszeretettel használom a kódok keverését és a jegyzethez is ennek alapján készültek a példák, sokan a sablonok alkalmazását tartják az egyedül elfogadható megoldásnak. Bízzuk ennek eldöntését az elkövetkező évekre...

értelmezni) ezzel jelentősen megnövelték a sebességet. Független mérések szerint a teljesítmény a teljesen újraírt interpreter hatására minimálisan ötvenszeresére nőtt.

A PHP 4 már gyakorlatilag az összes általánosan használt kereskedelmi és szabadon felhasználható adatbázist képes közvetlenül kezelni a megfelelő modulokon keresztül (Informix, Microsoft SQL Server, mSQL, MySQL, PostgreSQL, ODBC, Oracle és Sybase, hogy ízelítőt adjunk a lehetőségekből). Különböző külső modulok segítségével PDF dokumentumokat tud előállítani vagy értelmezni tudja az XML kódot de .rtf fájlt is előállíthatunk a segítségével.

A PHP elsődleges felhasználói a Linux rendszereken Apache szervert futtató fejlesztők, de hasonló jó eredményeket lehet elérni bármely Unix vagy Windows platformon is (a gyakorlat során is egy Windows rendszeren futó változatot alkalmazunk a fejlesztéseinkben). A PHP támogatja a HTTP session-öket, Java kapcsolódásokat és számos protokollt. A PHP jelenleg mintegy 9 millió webszerveren fut [http://www.netcraft.com/survey/ felmérés] és folyamatosan növekszik a népszerűsége.

A PHP nyelv a rendelkezésre álló információk szerint a jövőben is fenntartja a dinamikus növekedést. A jegyzet írása idején (2003 nov. - 2004 márc.) bocsátották ki a PHP 5 béta verzióját, amely ismét egy új interpretert kapott, a Zend Engine 2-t. Ennek segítségével jelentősen kibővítették a PHP már korábban is létező objektum-orientált lehetőségeit, bevezették az osztályok minősítését (abstract, final, protected) és több egyéb pattern (minta) mellett implementálták a Template Method design patternt (tervezési minta) a kód újrafelhasználás tökéletesítésére. Ennek egyenes következménye, hogy mire ez a jegyzet a hallgatóság kezébe kerül, neki lehet állni az új változat írásának és a példák is átszerkeszthetők a PHP 5 lehetőségeinek a teljes kihasználására :-(.

Röviden összefoglalva a PHP nyelv "felnőtt" az objektum-orientált nyelvek világába és ez egyenlőre csak nehezen sejthető távlatokba vezet. Ennek a távlatnak a felméréséhez segítséget nyújthat az a bejelentés, hogy a Zend Technologies Ltd., a Zend Engine létrehozója részt vesz a Sun Microsystems Java Specification Request (JSR) kezdeményezés kifejlesztésében, amelynek célja Java-alapú rendszerek standard elérése szkriptnyelvekből, elsősorban természetesen PHP-ből. Ez egyrészt a PHP értékét növeli, másrészt lehetővé teszi PHP webes alkalmazások (front-end) és Java üzleti logika integrációját. Az elképzelés támogatói között vannak a következő neves cégek is: Apple, Borland, Macromedia, MySQL, Sun, Oracle és persze a Zend Technologies.

A PEAR

A felnőttséget bizonyítja az is, hogy elképesztő mértékben megnőtt az interneten a PHP programok, megoldások száma. Gyakorlatilag szinte minden témára van csokornyi megoldás és az így óhatatlanul létrejövő káoszt felismerve megindult az információ rendszerezése is. Ennek talán legjelentősebb „gyümölcse" a PEAR, („PHP Extension and Application Repository", PHP kiterjesztés és alkalmazás gyűjtemény) a nyílt forrású PHP kódok rendezett könyvtára. Ugyanakkor kicsit többet is jelent, mert rendszert teremtett a kódok terjesztésében és a kódcsomagok fenntartásában, egységes kódolási stílust követ, megalapozza a PHP Foundation Classes-t, azaz a PHP osztálykönyvtárát, ahova a PHP közösség jóváhagyásával kerülhetnek be minőségi, általánosan alkalmazható és kompatibilis osztályok, amelyek készítői kicsit előre is terveznek („forward-compatible", azaz előre gondolnak a fejlesztési lehetőségekre), hogy egy általánosan használható rendszert hozzanak létre. Ennek és az itt tárolt osztályoknak a leírását is megtalálhatjuk már részben magyar nyelven a http://pear.php.net/manual/hu/ webhelyen.

Programozás PHP nyelven

A PHP szintaxisa, nyelvi elemei – néhány gyorsan elsajátítható kivételtől eltekintve - könnyen érthető mindazok számára, akik a C/C++ programnyelvben vagy a Javaban otthon vannak. Ezért ebben a jegyzetben nem akarom részletesen tárgyalni, csupán felsorolom az ismertnek feltételezhető elemeket, viszont rövid leírást adok azon elemekről és alkalmazásukról amelyek a PHP nyelv sajátosságait testesítik meg. Ezek közül is elsősorban azokat emelem ki, amelyek a tananyag elsajátítását segítő példákban előfordulnak. A tananyag elektronikus jelleg megfelelően utalok itt a http://hu.php.net/manual/hu/langref.php webhelyre - ez a magyar nyelvű kézikönyv lelőhelye -, ahol minden PHP-val kapcsolatos ismeret megtalálható. Ennek letölthető változata megtalálható továbbá a tantárgyak weboldalán is a http://www.gdf.hu/iai/Segedletek/Segedlet.htm, címen, valamint a háttérként használt saját magam által üzemeltetett oldalon is www.medzi.hu/php.

A kódolás során igyekezzünk következetesek lenni változó elnevezéseinkben, jelöléseinkben, stílusunkban. Bőségesen használjunk megjegyzéseket, hogy később is érthető maradjon számunkra az egyszer kitalált kód. A sok lelkes programozó által fejlesztett nyelvben meglehetősen nagy a redundancia, gyakran találunk közel azonos

funkciókat biztosító függvényeket, nyelvi elemeket. Érdemes kialakítani egy saját stílust és kiválasztani egy szűkebb eszköztárat, de azt alaposan kiismerni, mint mindig új és új megoldásokat alkalmazni. Persze ne féljünk ezektől, ha határozott előnnyel járnak….

Nyelvi jellegzetességek:

Változók, típusok
(kézikönyv::language.variables.html; kézikönyv::language.types.html)

A PHP-ban a változókat nem kell külön deklarálni, a változó nevének leírásával létre is jön. Minden változót megelőz a **$** jel, ez bizonyos értelemben könnyíti a kód olvasását és hamar megszokható. Típusát nem kötelező megadni, alapértelmezésben attól függ, hogy milyen érték kerül bele - ez a kódban mindenütt igaz, ha egy tömböt tartalmazó változóba integert rakunk, az attól kezdve integer típusú lesz. Hat típust különböztetünk meg, ezek a következők: **integer** (**int**), **double**, **boolean** (**bool**), **string**, **array**, **object**. A `$tortszam = 2.14;` kód például egy **double** típusú **$tortszam** nevű változót határoz meg. Típuskonverzióval - ha erre lehetőség van az érték keretein belül - tetszés szerint átalakíthatjuk a változókat. A következő kód például működik:

```
$egyik = "szöveg";
$masik = (bool)$egyik;
echo $masik;
```

és egy egyest ír ki, mert ha nem üres sztringről van szó, akkor azt **true** értéknek tekinti, ennek megfelelője az 1 érték. Ennek "megfordítottja" a következő kód:

```
$egyik = false;
$masik = (int)$egyik;
echo $masik;
```

ami viszont nullát ír ki, a **false** értéknek megfelelően.

A változók tartalmazta **értékek átalakítására** - pontosabban más típus szerinti értelmezésére - illetve a **változók vizsgálatára** több függvény áll rendelkezésre. A gyakorlatok során is előkerül majd az intval() függvény, amely megpróbálja a paraméterként kapott változót vagy karaktersorozatot egész számként értelmezni, így például az echo(intval("123asdf")) kód a "123" egész számot írja ki. Ha a változó karakterrel kezdődik, akkor a visszakapott érték "0". A különböző is_boolean(), is_array(), is_numeric(), stb. függvényekkel nevüknek megfelelően vizsgálható a változó aktuális

típusa és a szintén itt említhető isset() illetve empty() függvények segítségével eldönthető, hogy egy változó kapott-e már értéket. Például egy webes űrlap rádiógomb csoportja esetében, ha egyiket sem kattintják be, a hozzátartozó változóra az isset($radiogombcsoport) függvény hamis értéket fog adni. Egy sztring típusú változó esetében viszont az üres sztringet csak az empty() függvénnyel vizsgálhatjuk! Ezeket a függvényeket nagyon jól lehet alkalmazni űrlapok és szöveges fájlok feldolgozásánál, így ismeretük ajánlott (legalább tudjunk annyit, hogy hol kell keresni).

A PHP specifikus **dinamikus változók** sok nagyszerű lehetőséget, elegáns megoldásokat biztosítanak a programozónak, bár a fogalom elsajátítása először gondokat szokott okozni. Röviden arról van szó, hogy a változó nevét (pl. $lakcim) egy másik változóban akarjuk tárolni, valahogy így: $cim = "lakcim". Ekkor a $$cím jelöléssel hivatkozhatunk a $lakcim változó tartalmára. Ennek jelentősége a webes alkalmazásoknál nagy, ezért egy tipikus példán mutatjuk be a használatát.

Feladatunk egy sok kérdésből álló felmérés létrehozása HTML űrlap formájában, ahol minden egyes kérdés mögött egy-egy változó áll. Az egyes változók nevei utalnak a kérdés sorszámára, de nem akarunk 50-100 változót külön-külön leírni a kódban. Arról nem is beszélve, hogy ezeket majd végül adatbázisba kell vinni… Egy lehetséges megvalósítás kódja a következő:

```
<?php
//az űrlap elküldése után megszámoljuk, hány mezőt nem töltöttek ki
$hibaszam = 100;
for ($i=1; $i<=100; $i++)
{
  $kerdes = "kerdes".$i;
  if (!empty($$kerdes))
    $hibaszam--;
}

if ($hibaszam < 10)
  echo("A kitöltés sikeres, az űrlap elküldhető!");

//beviteli mezők kirakása
for ($i=1; $i<=100; $i++)
  echo("<input type=\"text\" name=\"kerdes".$i."\">\n");
echo("<input type=\"submit\" name=\"submit\" value=\"Mehet!\">");
?>
```

Látható, hogy a száz változón egyszerűen végig tudtunk futni **for** ciklus segítségével, összeraktuk a nevét egy változóban ($kerdes), majd ezt a változót dinamikus változóként kezelve ($$kerdes) megvizsgáltuk, hogy az aktuális értékének megfelelő felmérési kérdést azonosító változó üres-e.

A webes alkalmazások előre definiált változói
(kézikönyv::reserved.variables.html)

Ezek a változók szolgáltatják számunkra az átadott értékek feldolgozásakor vagy más esetekben szükséges adatokat. Bővebben lásd a phpinfo() metódus tárgyalásánál.

$PHP_SELF az éppen futó szkript fájlneve elérési úttal együtt, jól használható látogatási statisztikákhoz;

$HTTP_GET_VARS az újabb nyelvi változatokban (4.1.0 felett) $_GET, a HTTP GET metódussal elküldött kulcs-érték párok asszociatív tömbjét tartalmazza, azaz a fenti példában az űrlapon található $kerdes89 nevű változó értékét a következőképpen kapjuk meg: $HTTP_GET_VARS["kerdes89"] illetve $ _GET["kerdes89"];

$HTTP_POST_VARS az újabb nyelvi változatokban $_POST, a fentiekkel teljes mértékben analóg változó, csupán ez a POST metódussal elküldött kulcs-érték párok asszociatív tömbjét tartalmazza;

$_REQUEST csak az újabb nyelvi változatokban megtalálható szuperglobal[10] asszociatív tömb, amely tartalmazza a $_GET, $_POST, $_COOKIE változókat együtt;

$GLOBALS az összes global tartományban deklarált változó asszociatív tömbje, úgynevezett szuperglobal, azaz nem kell a metóduson belül külön hivatkozni a változó globális érvényességi tartományára.

Konstansok
(kézikönyv::language.constans.html)

Állandókat létrehozhatunk a define() metódus használatával, de nagyon sok esetben előnyösen alkalmazhatóak az előre definiált konstansok, amelyek sok értékes információt elárulnak a környezetről, vagy éppen a kezelt fájlról:

```
echo PHP_VERSION."\n";
echo __FILE__."\n";
```

[10] Minden programnyelvben fontos ismerni a változók érvényességi körét. A PHP esetében ha egy függvény, metódus belsejéből el akarunk érni egy kívül deklarált és értéket kapott változót, akkor a global kulcsszóval kell rá hivatkozni. Ekkor a változó értéke mintegy bemásolódik a metódusba és kiolvasható. A szuperglobal jelleg ezt automatikusan megteszi, nincs szükség külön global hivatkozásra.

```
echo __LINE__."\n";
```

ez a három utasítás kiírja a használt értelmező verziószámát, a futtatott fájl nevét elérési úttal együtt és annak a sornak számát, ahol a harmadik utasítás (__LINE__) megtalálható - ez nagy segítség lehet például hibakeresés közben. A pont, mint konkatenációs operátor segítségével hozzáfűzött sztring (a "\n" karaktersorozat) természetesen a soremelést biztosítja.

Vezérlő szerkezetek
(kézikönyv::control-structures.html)

Reguláris kifejezések
(kézikönyv::ref.regex.html)

Függvények, metódusok
(kézikönyv::funcref.html)

A példákban gyakrabban használt függvények, metódusok, nyelvi szerkezetek

Ezek pontos és hivatalosan elfogadott specifikációja - nagyon gyakran példákkal együtt - megtalálható a PHP magyar nyelvű kézikönyvében. A tanulás és a gyakorlati munka segítésére az alább felsorolt függvényekre a HTML forma változatlan megtartásával referenciát[11] (jegyzet.htm) készítettem a kézikönyv megfelelő lapjainak linkelésével. Ezúton is köszönet illeti a kézikönyv magyar fordítóit és gondozóit.

`include(), include_once()` - Külső fájlok beillesztését végzi a kódba. Segítségével osztályokat, komponenseket, gyakran ismétlődő kódrészleteket illeszthetünk be. Ilyen megoldásokat alkalmazunk az adatbázis kapcsolatokhoz szükséges állandó kódrészletek bevitelére. Az `include_once()` alkalmazása esetében a rendszer figyel arra, hogy a kérdéses beillesztett kód csak egyszer szerepeljen a fájlban pl. ne legyen két osztályleírás.

[11] A PHP kézikönyv sok fájlból álló csomagját (php_kezikonyv.zip a Tantárgyi segédletek oldaláról, vagy bármely más forrásból) tegyük egy könyvtárba kibontva és ugyanott helyezzük el a jegyzet.htm fájlt. Ezt böngészőben inditva rendelkezésre áll a gyors referencialista.

echo(), print() – Kiíró utasítások, gyakran használjuk a generált HTML kód fájlba írására.

strlen(), strtok(), substr(), explode(), trim() – Sztringkezelő metódusok, a strlen() a sztring hosszát adja vissza, strtok(), explode() a sztring részekre bontását végzi, adott karakternél a sztringet szétvágja és darabjait egy tömb elemeiként adja vissza, a trim() metódus eltávolítja az üres karaktereket a sztring két végéről, a substr() különbözőképpen paraméterezhető és segítségével tetszés szerinti darab vágható ki a sztringből – ez jól alkalmazható adatbázisban tárolt sztringek feldolgozására, például a MySQL időbélyegzője[12] (timestamp) esetében.

tömbök és kezelésük
(kézikönyv::language.types.array.html; kézikönyv::function.array.html)

array();

array_push();

array_merge();

array_rand();

Adatbázis kapcsolatot, kezelést biztosító függvények
(kézikönyv::function.mysql-*.html; kézikönyv::function.dbm*.html)

[12] Adatbázisokban szinte mindenütt megtalálható az időbélyegző (timestamp) funkció. Ennek segitségével a tranzakció pontos ideje rögzithető a gépidő alapján. Ugyan a számitógépek belső órái ritkán pontosak, de egy jól beállitott szerver kommunikál a világhálón elhelyezett időszinkronizáló számitógépekkel, igy a gépidőt mindig megfelelő pontossággal képes biztositani. Időbélyegzőt szinte mindenütt használunk a rekordok felvétele során - még akkor is, ha ez nem előirás a kérdéses adatbázis specifikációjában – mert felbecsülhetetlen előnnyel jár a rekordok rendezésekor és bármilyen biztonsági probléma megoldásában.

Osztályok, objektumok

(kézikönyv::language.oop.html)

A phpinfo() metódus

Ha egy weboldalba beillesztjük ezt a metódus nevet a <?php és ?> jelölések közé és lefuttatjuk, csodát látunk. Ez ugyanis kiírja az összes PHP beállítást, konfigurációt, a környezeti változókat és adatokat, a PHP beépített változóinak aktuális értékét, a telepített programcsomagokat, azok beállításait és még sok mást. Itt van például az én saját számítógépem PHP környezeti beállításainak egy részlete:

REDIRECT_STATUS	200
REDIRECT_URL	/info.php
REMOTE_ADDR	192.168.1.1
REMOTE_PORT	1400
SCRIPT_FILENAME	/apache/php/php.exe
SERVER_ADDR	192.168.1.2
SERVER_ADMIN	you@your.address
SERVER_NAME	localhost
SERVER_PORT	80
SERVER_SIGNATURE	*Apache/1.3.14 Server at localhost Port 80*
SERVER_SOFTWARE	Apache/1.3.14 (Win32)
WINDIR	C:\WINDOWS
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.0
REQUEST_METHOD	GET
QUERY_STRING	
REQUEST_URI	/info.php
SCRIPT_NAME	/php/php.exe
PATH_INFO	/info.php
PATH_TRANSLATED	C:\apache\htdocs\info.php

Itt látható ugyanakkor az installált egyéb csomagok egy részlete is, történetesen DBM adatbázisok kezeléséhez és a dinamikus képek előállításához szükséges függvénykönyvtárak csomagjai:

db

flat file support enabled

dba

DBA support	enabled
Supported handlers	db3

gd

GD Support	enabled
GD Version	1.6.2 or higher
FreeType Support	enabled
FreeType Linkage	with TTF library
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Sablonok

Különlegességek, speciális függvények

A nyelvet gyakran használjuk szövegek kezelésére, így különösen erős és változatos a sztringkezelése rendelkezésre álló eszköztára. Ennek egyik példája az EOD szerkezet használata, amit egy SQL select utasítás összeállításával szemléltetünk:

```
<?php
$nev = "Kovács Bence";
$irszam = 1194;
$select = <<<EOD
insert into szemely (nev, lakcim, telszam) values
('$nev', '$irszam Budapest, Etele út 68', '(06-1) 280-6207')
EOD;
echo $select;
?>
```

A <<<EOD - soremelés - tetszés szerinti szöveg és változó kombináció feloldó karakterek és konkatenálás nélkül - soremelés és EOD; lezárás egy sztringet határoz meg és ez a szokásos módon kiiratható egy echo utasítással.

Egy Apache + PHP + MySQL fejlesztőrendszer összeállítása, telepítése

Ahhoz, hogy PHP alkalmazásokat minimális időráfordítás mellett tudjunk fejleszteni, több, jól együttműködő szoftver eszközre van szükségünk. Kell hozzá egy HTML szerkesztő programra, hogy ne kézzel kelljen az összes HTML kódot beírni. Természetesen kell hozzá a megfelelő webszerver, amelynek beépülő modulja a PHP interpreter. Maga a PHP kód egyaránt szerkeszthető a legegyszerűbb szövegszerkesztővel és persze a HTML szerkesztőprogram kódszerkesztőjével (a modern HTML szerkesztők vagy közvetlenül, vagy beilleszthető modul formájában biztosítják a szintaxis kiemeléses PHP kódszerkesztést), de jobban áttekinthető a kód, ha erre a célra kifejlesztett PHP szerkesztőprogramot használunk hozzá. Nem utolsó sorban az egyes elkészült részprogramok, modulok kipróbálását is egyszerűen elvégezhetjük egy független PHP editor segítségével. Szükségünk van továbbá adatbázis kezelőre is, amely vagy maga tartalmazza a kezelőfelületet, amely segítségével egyszerűen létrehozhatjuk az adatbázis szerkezetét, bevihetjük az adatait, vagy szükséges még hozzá egy külön kezelőfelület is.

Ezeket a fejlesztőrendszer komponenseket alapvetően kétféle módon szerezhetjük be. Megvásárolhatunk kész fejlesztőrendszereket, amelyek biztosítják az összes komponenst és a közöttük levő kapcsolatokat, illetve kihasználva a webes fejlesztések és a PHP sok esetben alkalmazott GNU[13] licencét, szinte minden egyes komponens esetében találhatunk szabadon telepíthető változatot és így minimális költségráfordítással - de természetesen ennek megfelelően nagyobb mennyiségű munkával és az egyes konfigurációs fájlok beállítása némi szakértelmet is igényel - összeállíthatjuk fejlesztő környezetünket.

A gyakorlatban és így a jelen esetben is a helyzetünk lényegesen egyszerűbb. A fejlesztési munkákat Windows rendszeren (Win98 - WinXP) végezzük, a webszervert (Apache), annak PHP modulját (PHP 4) és az adatbázis szervert (MySQL) a PHPTriad programcsomaggal együtt telepítjük. HTML szerkesztésre a Microsoft FrontPage alkalmazható - ez a program a Microsoft Campus Licensz részeként hozzáférhető - és a

[13] GNU General Public Licence, általános közösségi licensz, a Szabad Szoftver Alapítványhoz (Free Software Foundation) tartozó programok licensze, amely biztosítja a szerzői jog megtartása és érvényesítése mellett az érvényesége alá tartozó szoftverek szabad terjesztését, felhasználását, módosítását. Ettől függetlenül az ezen szoftverekkel végzett munkáért, a terjesztésért anyagi ellenszolgáltatás kérhető.

PHP kód szerkesztését illetve az egyes modulok kipróbálását a PHPEdit szintén szabadon terjeszthető szerkesztőprogrammal végezhetjük. Ha komolyabb adatbázisokat akarunk létrehozni és ezeket kezelni is szükséges, akkor a MySQL esetében érdemes megszerezni a MySQL Front - szintén szabadon terjeszthető - adatbázis-kezelő programot is, amely segítségével közvetlenül létrehozhatjuk a táblákat és a mezőket. Ezeket a programokat még ki kell egészíteni a megfelelő dokumentációval is, ehhez szükséges az Interneten fellelhető PHP Manual, amely HTML formában tárolja az összes programozáshoz szükséges leírást, illetve az irodalomjegyzékben megtalálható ennek részben magyarra fordított változatának lelőhelye is. A fent felsorolt programok több helyről gyűjthetők össze, a kényelem kedvéért ezek mind letölthetők viszont a www.medzi.hu/php webhelyről is, az angol nyelvű kézikönyvvel együtt, sőt a webhelyen megtalálható a Javascript nyelv referencia anyaga is.

A telepítés menete Windows rendszeren:

Ahhoz, hogy eredményes legyen a telepítés, a Win2000 és a WinXP rendszereken rendelkeznünk kell adminisztrátori jogokkal. Ennek hiányában is sok minden megvalósítható, de az eredmény bizonytalan lesz. Először a PHPTriad csomagot telepítjük a telepítőkészlet .exe fájl elindításával - ajánlatos ezt a Futtatás menüpontból végezni. A folyamat automatikus és mindent beállít egy alapértelmezésnek megfelelően, azaz létrehoz egy apache könyvtárat a C: meghajtón és abban helyezi el az összes szükséges fájlt és könyvtárat, így például a htdocs könyvtárat is, amely a webszerverünk gyökérkönyvtára lesz. Ebben már van egy-két információs fájl, egy index.php fájl és a phpadmin könyvtár is, amely egy php nyelven írt MySQL adatbázis adminisztrációs program. A telepített Apache webszerver az alapértelmezett konfigurációs beállításoknak megfelelően a localhost IP címen található meg (127.0.0.1). A webszervert azonban mint szolgáltatást el kell indítani (Futtatás menüpontból c:\apache\Apache.exe; a webszerver DOS ablakban fut, amely természetesen minimalizálható) és csak ez után indítjuk a böngészőt, amelyben a lokátorsorba a http://localhost címet írjuk és erre a HTTP kérésre a webszerver a c:\apache\htdocs könyvtárban található index.php fájlt szolgáltatja ki. Amennyiben 404-es hibát (azaz a kért fájl nem található) kapunk, először mindig győződjünk meg, fut-e az Apache - a Windows világában igen könnyű elfeledkezni a manuális indításról. A modulként dolgozó PHP interpreter már lefuttatta a PHP kódokat és a megfelelő HTML kimenetet dolgozza fel a böngésző, tökéletesen

azonos rendszerben, mint egy távoli szervernél. Ezért természetesen a közvetlenül a böngészőben kinyitott .php fájlok nem értelmezhetők, a böngésző önmagában csak HTML és Javascript nyelveken ért.

Ezzel lényegében kész a webszerverünk, de még számos probléma lehet. Gyakorlati tapasztalat, hogy amennyiben IIS szervert telepítünk az Apache szerver után, az operációs rendszer (WinXP) felülbírálja a localhost alapértelmezett könyvtárát és a saját rendszere szerint a C:\ meghajtón az Inetpub\wwwroot könyvtár lesz a gyökérkönyvtár. Amennyiben a PHPTriad-ot újratelepítjük, akkor visszaáll a rendszer, de ha mindkettőt használni kívánjuk, akkor az Apache konfigurációs fájlt kell beállítani. Ezt az c:\apache\conf könyvtárban találjuk meg, httpd.conf[14] néven és a Jegyzettömb, vagy más egyszerű szövegszerkesztő segítségével két helyen át kell írni az apache/htdocs bejegyzést Inetpub/wwwroot bejegyzésre (rákeresünk és átírjuk, az egyik a DocumentRoot, a másik a Directory neve). Természetesen a konfigurációs fájl módosítását addig nem veszi figyelembe a rendszer, amíg le nem fut, azaz az Apache szolgáltatást leállítjuk (becsukjuk a DOS ablakot, becsukjuk a böngészőt) és újraindítjuk az Apache webszervert a már ismert módon.

A telepítéssel nem csak a webszerver, hanem egy MySQL adatbázis - továbbá annak kiszolgáló program csomagja - is felkerült a rendszerünkre. Mivel dinamikus webes alkalmazások szinte elképzelhetetlenek adatbázis nélkül, ez nagy segítség a számunkra, csak meg kell tanulnunk a használatát és legelőször a beállítását.

A MySQL a telepítés során szintén alapbeállításokat kap, azaz nem kell sokat bajlódnunk vele, bár a rendszer maga a sokféle feladatnak megfelelően bonyolult . Az adatok az apache\mysql\data könyvtárba kerülnek, ide a telepítéssel együtt már bekerül egy teszt adatbázis és egy mysql adatbázis, ez utóbbi a felhasználók nyilvántartását szolgálja és a rendszer saját adatainak tárolását végzi. NE NYÚLJUNK HOZZÁ!

Alapértelmezésben települ egy adminisztrátor felület is, ez a winmysqladmin, amelyet szükség esetén az azonos nevű .exe fájl segítségével indíthatunk az

[14] httpd.conf hipertext transzfer protokoll démon konfigurációs fájl, a Linux/Unix rendszerek egyszerű szöveges konfigurációs fájl szerkezete. Érdemes áttanulmányozni, mert sok minden állítható itt be, amelyre később esetleg szükségünk lehet. A # jelzi a megjegyzéseket, az Apache induláskor nem veszi figyelembe az ezzel védett sorokat.

apache\mysql\bin könyvtárból. Amennyiben feltelepült, akkor minden Windows indításnál megjelenik - az msconfig-ba bekerülő mysqld-opt program indítja - és a tálcán közlekedési lámpa ikonként látható. Ha a zöld "világít" az admin felület működik és az adatbázisok rendelkezésre állnak. Amennyiben nem működik vagy nem is látható, a fentebb leírt programmal manuálisan indíthatjuk, illetve az úszómenüjéből indítható az adatbázisszerver (jobb egér klikk az ikonra, Show me hozza be az ábrán is látható kezelőfelületet illetve a Win9x illetve WinNT menüpontokból érhető el az indítás/leállítás menüpont). Ugyanakkor ez a kezelőfelület csak az adatbázisok menedzselését képes elvégezni és a kapcsolatokat tartja nyilván, az egyes adatbázisok ugyan itt is létrehozhatóak (Admin felület - Databases fül - jobb egér klikk a LOCALHOST gyökérelemen) de táblák már nem hozhatók létre ezen a módon. SQL szakértőknek ott a mysql parancssoros kezelőfelülete, a MySQL Monitor (apache\mysql\bin\mysql.exe) amelyen keresztül minden megvalósítható az SQL ismeretében, kevésbé gyakorlottaknak pedig a http://localhost/phpMyAdmin/index.php elérési úton található azonos nevű adatbáziskezelő programot, vagy a még barátságosabb MySQL Front programot ajánlom (lásd később). Persze az adatbázis, a táblák, a mezők mind létrehozhatók PHP programból is, erre majd a gyakorlati részben találunk példákat.

MySQL Front telepítése

Beszerzését lásd fentebb, telepítése is a fentebb leírtakkal azonos módon, az .exe fájl indításával, simán megy. Mindenképpen egy MySQL adatbázis szerver felrakása után telepítsük! Első indításakor kér egy logint és jelszót az adatbázis szerverhez történő kapcsolódáshoz, ez viszont megegyezik a winmysqladmin kezelőfelületen látható login/jelszó kombinációval, ami viszont - ha csak nem állítottuk át - megegyezik a gépünk operációs rendszerének loginjével és jelszavával. Ez után létre kell hozni magát az adatbázis kapcsolatot, megadva az adatbázis szervert futtató számítógép IP címét vagy domain nevét és a felhasználó nevet. Itt viszont alapértelmezésben a fenti rendszert logikusan követve a localhost lesz a domain név és root lesz a felhasználónevünk. Ez után egy sokoldalú felületen tudjuk kezelni az adatbázisainkat, mindenkinek javasolom a rendszer kiismerését és alapos kipróbálását, de részletes ismertetése egy újabb jegyzet anyaga lehetne. Ismerete viszont bármiféle adatbázis kezelési problémánkban segíthet, hiszen kattintásokkal és SQL parancsokkal egyaránt alakíthatjuk adatbázisunkat és meg is tanulhatjuk ezzel az eszközzel egy komolyabb adatbázis kezelését.

PHPEdit telepítése

PHPEdit 0.6 verzió: Ez a szerkesztőprogram - több más hasonló található az interneten szerteszét - kényelmessé teheti a munkát, mert a PHP szintaxis kiemelésével, a kód automatikus strukturálásával és help fájlok integrációjával erőteljes eszközre teszünk szert.

A fentebb megadott helyen található install programot elindítva simán feltelepíthető és már csaknem használatra is kész. Alapértelmezetten a Program Files mappába települ PHPEdit mappát hozva létre. A telepítés során - francia nyelven - felkínálja a .php és az .inc kiterjesztések regisztrálását a programhoz - fogadjuk el, ha nincs más távolabbi elképzelésünk - és máris írhatjuk PHP programjainkat. A programok az F9 gyorsbillentyűvel vagy az eszközsoron található Run zöld nyíl gombbal futtathatók és a kimenet - némi kommentár mellett - a View → Debugger Windows → Output ablakban látható. Vigyázni kell azonban, mert ez a kimenet egy hagyományos szöveges felület és ezért itt a soremelés például \n és nem
 mint a webszerveren futtatott PHP kód esetében! Ez persze logikus, itt nem HTML kimenet készül, hanem egyszerű szöveges.

A help rendszert viszont ki kell egészíteni. A Program Files → PHPEdit → mappában találunk egy help mappát, azon belül a chm mappába telepíthetjük a .chm típusú PHP kézikönyvet és a PHP mappába a html típusú kézikönyvet helyezhetjük el. Ezeket szintén megtalálhatjuk a fentebb megadott webhelyen és az egyszerű bemásolás után már csak a PHPEdit.ini fájlt kell szerkesztenünk a webhelyen megadott minta .ini fájl szerint. Ezzel részben helyzetérzékeny súgót kapunk, ami az egyes metódusok begépelésénél már felkínálja a lehetőségeket illetve az egyes függvények nevén állva a help előhívásakor megjelenik a html formában hozzáférhető információ az adott nyelvi elemről. A rendszer viszonylag egyszerű, ám erőteljes - segítségével tesztelhetjük kísérleti kódjainkat webszerver nélkül is és minden további nélkül használhatjuk adatbázis kapcsolatokkal is - jómagam a webes programozástól függetlenül, számos szöveg feldolgozási problémát és adatbázis feltöltést oldottam meg ezen egyszerű eszköz segítségével. Nem hallgathatom el, hogy mint minden fejlesztés alatt álló eszköznek, ennek is vannak hibái, időnkét - francia nyelvű - hibaüzenetekkel bombáz, de ha a kódolás közben többkevesebb rendszerességgel mentettünk, ezeket nyugodt szívvel figyelmen kívül hagyhatjuk - legfeljebb újraindítjuk a programot, ha nagyon okvetetlenkedik.

PHPEdit 0.8 verzió: A jegyzet írása során került kezembe ennek a programnak egy magasabb szintű változata, amit szintén beraktam PHPEdit2 néven a tantárgy programcsomagjába. Ez már a 0.8 változat, természetesen többet tud, mint elődje, a legfontosabb a helprendszer megoldása. A telepítéshez vagy az internetről kell letölteni a kiegészítőket, vagy a mi esetünkben mellékeltem a .zip fájlban azokat a .gip kiterjesztésű fájlokat, amelyek mindezeket tartalmazzák. Egy közös könyvtárba kell kibontani és onnét indítani a telepítést és a rendszer mindent felismer és helyre tesz. Jöhet a kódolás!

Végre elkészült a rendszerünk, most már csak egy-két tanács az összehangoláshoz. Természetesen mindenki maga alakítja ki a munkamódszerét, én csupán az általam alkalmazott és számomra bevált rendszert ismertetem.

Bármely komolyabb HTML szerkesztőrendszerben van lehetőség a szerkesztés alatt álló webhely modellezésére. Így a Microsoft FrontPage esetében is beállítható, hogy webhelyen dolgozzunk, ez annyit jelent, hogy kijelölhetünk egy gyökérkönyvtárat a gépünkön, amelynek könyvtárszerkezete meg fog egyezni a tényleges webhely szerkezetével (fájl – létrehozás – lap vagy webhely – üres webhely – (beállítjuk a kívánt könyvtárat) - OK). Számunkra ez értelemszerűen az C:\apache\htdocs maga vagy annak valamelyik belső mappája lesz. Ez után a szerkesztőprogram felismeri és regisztrálja a lapok közötti kapcsolatokat, ha azokat itt hozzuk létre. Más szerkesztőprogramok általában alapértelmezetten tudják, de a FrontPage esetében is beállítható, hogy a .php és .inc kiterjesztéseket is megnyissák, ne a Jegyzettömböt kínálja fel ezek szerkesztéséhez (eszközök – beállítások – szerkesztőprogramok panel, itt adható meg, hogy melyik kiterjesztéshez melyik szerkesztőprogramot használja alapértelmezetten).. Ugyanezt a gyökérkönyvtárat használjuk a PHPEdit szerkesztőprogrammal is, csak azt kell észben tartani, hogy éppen mikor melyik programmal szerkesztjük a fájlt és mentsünk programváltás előtt! A látszólag bonyolult beállítás értelme, hogy gyorsan és minimális hibával dolgozhatunk a PHP részeken, de ugyanakkor nem kell hosszadalmas HTML kódokat kézzel beírni. Tekintsük úgy, hogy a HTML szerkesztőnk biztosítja a vizuális programozás eszközeit és a PHPEdit a kódszerkesztőnk.

Egy vegyes kódolású (PHP + HTML + Javascript) weboldal elkészítése

A webes alkalmazások sok szempontból különlegesek, de a legfontosabb az adatok kezelése. Olyan szoftver alkalmazást kell készíteni, amely sok önálló kisebb prog-

ramból áll, sok belépési ponttal és ugyanannyi kilépési ponttal (nem hagyhatunk például egy fájlt vagy adatbázist nyitva egy lap elhagyása esetén) és ezek még nem is ismerik egymás adatait, azok átadásáról külön gondoskodni kell. Az adatbevitel és feldolgozás külön egységen történik, a felhasználó megnyilvánulásaira csak egy másik gép tud reagálni, illetve előre le kell programozni a várható adatoktól függő lehetőségeket, ha nem akarjuk az összes válaszlépést a szerveren lekezelni - ami észlelhető időbeli késlekedéseket okoz. Nem használhatjuk a kliens gép memóriáját adatok tárolására, az összes adatforgalom szükségszerűen a kliens és a szerver között zajlik. Ezért az egyik legfontosabb dolog az adatforgalom és a navigáció megtervezése, hogy minimalizálni tudjuk a szerver felé irányuló kérések számát.

A felhasználói felületek esetében még ha egészen egyszerű is a feladat, mindig tervezzünk! Érdemes akár egy darab papírra felvázolni a webes felületeknél a már ismertetett általános elvek miatt oly fontos külső megjelenést. Tervezőprogram hiányában - kihasználva a webes alkalmazások erősen dokumentumos jellegét - akár néhány A4-es lapon is elkészíthető az egyes lapok vázlata és jelölhető azok logikai kapcsolata.

Gondoljuk végig, mit is kell csinálni az adott weblapon! Adatbevitel vagy adatmegjelenítés a döntő funkciója? Hova kerülnek az adatok? Másik weblapra vagy adatbázisba? Ezektől függ, hogy melyik lapon milyen kódot kell elkészíteni. Hol és hogyan ellenőrizzük a bevitelre kerülő adatokat? Célszerű bonyolultabb adatok esetén a kliens oldalon is elvégezni az adatok ellenőrzését, de semmiképpen sem mellőzhetjük a szerver oldali ellenőrzést!

Nem szép dolog ilyet feltételezni jövendő látogatóinkról, de gondoljunk a rosszindulatú felhasználókra! Az internet publikus, nem tudjuk meghatározni, milyen feltételek mellett böngészhet a felhasználó - egy internet kávézóban bárki válla felett más is figyelheti a képernyőn történőket és az otthagyott gépen tárolt weblapok sem védhetők. Ha van mit védenünk - belső adatok, érzékeny adatok - azokat biztonságos módon el kell zárni az illetéktelenek elől, nem kerülhetnek ki semmiképpen sem. Ezeket mind a navigáció során, mind a belépéseknél észben kell tartani, például GET metódussal nem adunk át érzékeny adatot (az URL-hez fűzi hozzá az adatokat!).

A szkriptnyelvek esetében éppúgy érvényesek az általános programozási elvek, mind a hagyományos programozásnál. Igyekezzünk ismételten felhasználható kódot írni, megéri! A webes alkalmazások standard építőelemeit elég egyszer precízen elké-

szíteni, és bármely alkalmazásunkba beilleszthető. Fórumok, dátum és időkezelések, szavazógépek, látogató számlálás, jelszógenerálás és még sok egyéb újrafelhasználható komponens PHP kódja hozható létre mint osztály és ezt már csak példányosítani kell (objektum létrehozása!) az újrafelhasználáshoz. A forráskód megjegyzésekkel való ellátása rendkívül fontos, hogy a HTML kóddal felhígított forrásban tájékozódni tudjunk. Én például minden oldal elején leírom magamnak, mit akarok megvalósítani az adott fájlban és az egyes műveleteket kódolását is megjegyzésekkel kísérem…Azt viszont sohasem felejtsük el, hogy szkriptnyelvről lévén szó, a forrás ott van a webszerveren és a PHP értelmező egy esetleges hibája miatt (nem értelmezi a kódot teljes egészében) az egész esetleg kikerülhet a felületre - határozottan kínos, ha ilyenkor a felhasználó megtudja, hogyan vélekedtünk róla kódolás közben - egyszóval csínján bánjunk a jópofa megjegyzésekkel…

Ha már mindent végiggondoltunk, megterveztünk, jöhet a kódolás. Ha grafika is van az oldalon, akkor azt kell először elkészíteni, hogy az egyes kiírások elhelyezhetők legyenek az elképzelés szerint. Ennek hiányában vagy ennek kiegészítéseként készítjük el a HTML vázat, például az űrlapok elemeit felrakjuk, elrendezzük (például táblázatban) majd beállítjuk. Ekkor már tudnunk kell a szükséges adathordozó változókat, hiszen az űrlapmezők nevei lesznek a változók nevei is.

Célszerű elkészíteni az összes szükséges weblapot, még a PHP által generáltak vázát is és érdemes létrehozni a navigációs kapcsolatokat. Felbukkanó ablakoknál megírhatók a szükséges Javascript kódok és akár ideiglenes linkek segítségével tesztelhető a működés. Sokszor ilyenkor derül ki, hogy valami túl nehézkesre sikerült, de ilyenkor minimális pluszmunkával még átalakítható a rendszer.

Ezt a szerkesztést kellő rutinnal gyorsan megvalósíthatjuk a HTML szerkesztő programok segítségével, könnyen "összekattinthatók" az oldalak. Megjegyzem, hogy ez nem pótolja a HTML tudást, mert a PHP kóddal sokszor kell HTML elemeket kiíratni, de a vizuális szerkesztőprogram segítségével gyorsabban megtanulhatók a szükséges kódok is. Arról semmiképpen ne feledkezzünk el, hogy a nyers HTML kód sokszor ütközik a PHP kódokkal így a kézi szerkesztés megint csak nem mellőzhető. A magam részéről ezért kedvelem jobban a kevesebb automatizmust használó régebbi webes szerkesztőprogram változatokat (például Dreamveawer 3, de ez nem ingyenes, meg kell vásárolni ☹).

Ne feledkezzünk meg a gyakori tesztelésről! Adatbázisból nyert adatok szépen modellezhetők definiált változókkal, így adatbázis igénybevétele nélkül kipróbálhatók a kódok. Fájl írások és olvasások, sztringfeldolgozások szintén jól modellezhetők például a PHPEdit segítségével. Új osztályok, modulok esetében is érdemes egyszerűsített tesztrendszereket létrehozni. Bár fentebb a grafika elsődlegességéről írtam, természetesen nem érdemes egy bonyolult logikai rendszert a grafikai realizálás HTML kódjai között tesztelgetni.

Természetesen még sok elméleti ismeretet lehetne felsorolni, de térjünk át a sokkal szórakoztatóbb gyakorlati megvalósításokra. Mostantól kezdve kedves olvasóm elmélettel csak az esetleges magyarázatoknál fog találkozni és programozási feladatokon át mutatom be a webes fejlesztések gyakrabban előforduló elemeit. Nem foglalkozom a külső megjelenéssel, színekkel, hogy a sok formázó kód ne fedje le a lényeget. A jobb áttekinthetőség miatt így főleg táblázatokban rendezzük el az egyes elemeket

Gyakorlati feladatok

Jelszavas belépés

specifikáció

Biztosítsunk egyszerű login/jelszavas belépést védett zónába és a jelszó megváltoztatását is tegyük lehetővé. Adatbázis most nem áll rendelkezésre, tehát a lehető legegyszerűbb megoldást kell választanunk - fájlban tároljuk az adatainkat.

a feladat elméleti háttere

A feladat két külön részből áll, ezek egymáshoz csak lazán kapcsolódnak. Célszerű a webhely nyitólapján vagy bármely más felületen, linkként elhelyezni a jelszó változtatási lehetőséget, amelyre kattintva felbukkanó ablakban végezheti el a felhasználó a jelszócserét, természetesen kötelező az új jelszó kétszeri beírása! Utána visszanavigálunk a beléptetést biztosító felületre, hogy rögtön kipróbálható legyen az új jelszavas belépés. Mind a jelszócsere, mind a belépés HTML **form** (űrlap) segítségével történik, az adatokat a szerveren ellenőrizzük és azok helyessége esetén elfogadjuk az adatot (jelszócsere esetében megkívánhatunk bizonyos bonyolultságot is, például **reguláris kife-**

jezés[15] illesztéssel) illetve átirányítjuk a felhasználót a védett zónába. A jelszavas belépés a védelem mellett elegáns módon alkalmazható a felhasználó azonosítására, például nyelvhasználat, egyéni profil szempontjából vagy bizonyos szintű hozzáférések biztosítására.

tervezés

Két űrlapot fogunk létrehozni, az egyiket magán az oldal felületén, a másikat célszerűen külön fájlban, ami csak ezt tartalmazza és ennek mérete a felbukkanást biztosító JavaScriptből szabályozható - erre ügyeljünk az űrlap objektumok elhelyezésekor. A bevitel természetesen POST metódussal történik, hogy ne legyen látható az URL-ben. Az egyszerűség kedvéért a perzisztenciát[16] biztosító fájlból a jobban kezelhető asszociatív tömbbe és onnét a fájlba visszaírást végző függvényeket külön ".inc" fájlban helyezzük el (`tombok.inc`), hiszen ezeket két helyen is használni kívánjuk.

Bár a specifikációban külön nem tértünk ki erre, bizonyos ellenőrzéseket is végzünk majd és hiba esetén a megfelelő helyen üzenetben értesítjük a felhasználót, hogy hol és mit hibázott.

megvalósítás (kód)

tombok.inc fájl kódja:

```
<?php
//fuggveny, amely beolvas egy fajlt asszociativ tombbe es visszaadja a
tombot
function tomb_feltolt($filenev){
    if(!$fp = fopen($filenev, "r")) //ha nem nyithato meg, hamis
erteket ad vissza
    return false;
    while(!feof($fp))
  {
    $sor=fgets($fp, 256);
    list($kulcs, $ertek) = split("\t", $sor);
    $asszoc_tomb[$kulcs] = $ertek;
    }
    fclose($fp);
    return $asszoc_tomb;
}

/*
```

[15] Megfelelő szabályok, helyettesítő karakterek alkalmazásával előállítható minta, amelyet függvény segítségével illesztünk a kérdéses karaktersorozatra. Reguláris expresszióval például meghatározhatjuk, mi az elfogadható telefonszám vagy email cím formátum egy beviteli mezőben.

[16] Programozás, szoftvertervezés során perzisztencia alatt az adatok elvi tárolási lehetőségét értjük általánosságban, ennek megvalósítása lehet egy adott formátumu fájl vagy tetszőleges adatbázis.

```
* fuggveny, amely kiir egy asszociativ tombot fajlba,
* ha sikerul, igaz erteket ad vissza
*/
function tomb_kiir($tomb, $filenev)
{
  if(!$fp = fopen($filenev, "w"))
    return false;
  while (list($kulcs, $ertek) = each($tomb))
  {
    $ertek = trim($ertek);
    $sor = $kulcs."\t".$ertek."\n";
    fputs($fp, $sor);
  }
  fclose($fp);
  return true;
}
?>
```

index.php fájl kódja:

```
<?php
//ez tartalmazza a fajl kiolvasashoz szukseges fuggvenyt
include ("tombok.inc");

//ha kitoltottek a login es jelszo mezoket
if ($login != "" && $pwd != "")
{
  //feltoltunk egy asszociativ tombot a tarolo fajlbol
  $tomb = tomb_feltolt("jelszo.txt");
  /*
  * megnezzuk, van-e ilyen login - jelszo par a tombben
  * a trim az esetleges sorveg es egyeb nem lathato
  * karakterek eltavolitasara szolgal
  */
  if (trim($tomb[$login]) == trim($pwd))
  {
    echo ("<script>location.replace('belso.php');</script>");
    exit();
  }
}
?>
<html>
<head>
<title>Jelszavas belépés</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
<style type="text/css">
<!--
td {
font-family: Arial, Helvetica, sans-serif;
font-size: 10pt
}
.table {
font-weight: bold;
color: #330000;
background-color: #FFFFCC; border: 1px #CC9933 solid
}
-->
</style>
</head>
```

```
<body bgcolor="#FFFFFF">
<table width="100%" border="0" cellspacing="0" cellpadding="6"
align="center">
<tr>
<td colspan="2">
<p><b>Nyitóoldal</b></p>
</td>
</tr>
<tr>
<td width="358" > </td>
<td align="right" width="225">
<form name="belepes" method="post" action="" >
<table width="220"  cellspacing="0" cellpadding="3"  class="table">
<tr>
<td>
<div align="center">
    <p>Jelszavas belépés</p>
    <p >login:
        <input type="text" name="login" size="15" maxlength="15">
    </p>
    <p >jelszó:
        <input type="password" name="pwd" size="15" maxlength="15">
    </p>
    <p >
        <input type="submit" name="Submit" value="  Belépés  ">
    </p>
</div>
</td>
</tr>
</table>
</form>
<p> </p>
<p align="center"><b><a href="javascript:void()"
onClick="window.open('jelszomod.php','JELSZOMOD','width=400,height=400
')">jelszócsere</a></b></p>
</td>
</tr>
</table>
</body>
    </html>
```

jelszomod.php fájl kódja:

```
<?php
//ez tartalmazza a fajl kiolvasashoz, irasahoz szukseges fuggvenyeket
include ("tombok.inc");

$hiba = "";
$loginhiba = "";
$siker = "";
//ha a login, regi jelszo kitoltve
if ($login != "" && $regipwd != "")
{
  //feltoltunk egy asszociativ tombot a tarolo fajlbol
  $tomb = tomb_feltolt("jelszo.txt");
  if (trim($tomb[$login]) == trim($regipwd))
  {
    if ($ujpwd1 == $ujpwd2 && $ujpwd1 != "")
    {
```

```
      $tomb[$login] = $ujpwd1;
      tomb_kiir($tomb, "jelszo.txt");
      $siker = "<br>A jelszó módosítás megtörtént!";
    }
    else
      $hiba = "<br>Nem egyezik a két jelszó!";
  }
  else
    $loginhiba = "<br>Nem jó a belépési jelszó/login!";
}
?>
<html>
<head>
<title>Jelszómódosítás</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
<style type="text/css">
<!--
td {
font-family: Arial, Helvetica, sans-serif;
font-size: 10pt
}
.table {
font-weight: bold;
color: #330000;
background-color: #FFFFCC; border: 1px #CC9933 solid
}
-->
</style>
</head>

<body bgcolor="#FFFFFF">
<p align="center"><b>Jelszómódosító felbukkanó oldal</b></p>
<p> </p>
<form name="modositas" method="post" action="">
<div align="center"></div>
<table width="220"  cellspacing="0" cellpadding="3"  class="table"
align="center">
<tr>
<td colspan="2" >
<p align="center">Jelszó módosítás</p>
</td>
</tr>
<tr>
<td width="53%" >login</td>
<td width="47%" >
<input type="text" name="login" size="15" maxlength="15" value=<? echo
$login; ?>>
</td>
</tr>
<tr>
<td width="53%" >régi jelszó: </td>
<td width="47%" >
<input type="password" name="regipwd" size="10" maxlength="15"
value=<? echo $regipwd; ?>>
<? if ($loginhiba != "") echo ("<font color=#FF0000 size=-
2>".$loginhiba."</font>");?>
</td>
</tr
<tr>
```

```
<td width="53%" >új jelszó: </td>
<td width="47%" >
<input type="password" name="ujpwd1" size="10" maxlength="15">
</td>
</tr>
<tr>
<td width="53%" >új jelszó ismét: </td>
<td width="47%" >
<input type="password" name="ujpwd2" size="10" maxlength="10">
<? if ($hiba != "") echo ("<font color=#FF0000 size=-
2>".$hiba."</font>");?>
</td>
</tr>
<tr>
<td colspan="2">
<div align="center">
<input type="submit" name="Submit" value="  Módosítás  ">
</div>
<?
if ($siker != "") //siker eseten kiirjuk es/vagy becsukjuk az ablakot
{
  echo ("<font color=#FF0000 size=-2>".$siker."</font>");
  echo ("<script>window.close(self);</script>");
}
?>
</td>
</tr>
</table>
<div align="center"></div>
</form>
</body>
</html>
```

továbbfejlesztési lehetőségek

A beviteli információk első szűrését (adatok meglétét, formáját, de semmiképpen sem küldünk le jelszót a kliensnek ellenőrzésre!!!) megvalósíthatjuk JavaScript segítségével a kliens oldalon, ezzel gyorsítható a használat, de nem pótolja - manipulálhatósága miatt - a szerver oldali ellenőrzést. Szükség esetén a jelszót, logint kódolhatjuk is (például MD5 kódolás) és reguláris kifejezés segítségével vagy más módon ellenőrizhető, hogy az előírásoknak (bonyolultságnak) megfelel-e a bevitt/megváltoztatott jelszó.

A gyakran használt PHP függvények kigyűjtése és linkelt lista készítése

specifikáció

Tanulásunk támogatására gyűjtsük ki magunknak a gyakrabban használt PHP függvényeket egy külön könyvtárba. A könyvtár tartalmát jelenítsük meg egy HTML lapon, linkekkel ellátva, hogy a megtekinteni kívánt függvény leírását gyorsan előszedhessük. A lista a könnyebb kereshetőség érdekében legyen ABC rendben és két hasábba

szedve. Természetesen a könyvtárat tetszés szerint bővítjük, módosítjuk, ennek megfelelően a HTML oldal és a linkek is frissüljenek.

a feladat elméleti háttere

A feladat megoldásához tudnunk kell egy könyvtárat listázni - ez nem sokban különbözik a fájl olvasástól, hiszen a könyvtár egy olyan speciális fájlnak tekinthető, amely más fájlokra mutató bejegyzéseket tartalmaz. Mivel a könyvtár csak virtuálisan szedhető ABC rendbe, ha éppen úgy kívánjuk megtekinteni, ezért egy tömbbe gyűjtjük először a fájneveket, annak rendezésére ugyanis vannak eszközeink (például a `sort()` függvény). Meg kell oldani azt is, hogy amikor táblázat sorokat írunk ki, mindig két függvénynevet kell egymás mellett megjeleníteni, de azok nem egymást követő elemek a rendezett tömbben. Végül az egyes függvénynevek linkek lesznek, ahol viszont a linkszöveg ne a függvényt tartalmazó fájl neve legyen (például `function.array-rand.html` a függvénynév, de mi `array-rand` nevet akarunk megjeleníteni a link szövegében).

tervezés

Egy `lista.php` fájlt és egy `konyvtar` nevű mappát hozunk létre és a mappába bemásoljuk a számunkra fontos fájlokat. Relatív elérési utat alkalmazunk, a `lista.php` és a mappa ugyanazon könyvtárban legyen a webszerver gyökérkönyvtárán belül valahol. A PHP kóddal végigolvassuk a mappát, kiszűrjük a számunkra felesleges fájlokat és a többit `array_push()` függvény segítségével tömbbe töltjük, amit már egyszerűen tudunk rendezni. Ezt egy függvény valósítja meg, paraméterül kapva a mappa nevét és visszatérési értéke a feltöltött rendezett tömb lesz. A rendezett tömböt a HTML kódban **for** ciklus segítségével táblázat soraiban jelenítjük meg, a kéthasábos megjelenítés miatt figyelni kell a páros/páratlan fájlszámra és ennek megfelelően kell egy plusz sort is beiktatni. A linkszöveghez a kiolvasott fájlnevet szét kell darabolni, míg a link megvalósításához a mappa nevét hozzá kell illeszteni a fájlnévhez.

megvalósítás (kód)

lista.php fálj kódja

```
<?php
$dirnev = "fuggvenyek";
//ez a fuggveny egy ABC szerint rendezett tombben adja vissza a mappa
tartalmat
function dir_lista($dirnev)
{
  $lista = array();
```

```
  if(!$dp = opendir($dirnev)) //ha nem talalhato, hamis a visszateresi
ertek
    return false;
  while($fajl = readdir($dp))
    if($fajl != "." && $fajl != "..") //kiszurjuk a "." es ".." nevu
fajlokat
      array_push($lista, $fajl);
  sort($lista); //rendezes a tombben
  closedir($dp);
  return $lista;
}

?>
<html>
<head>
  <title>Kedvenc PHP függvényeim</title>
</head>
<body>
<?php
$lista = dir_lista($dirnev); //megadjuk a dir nevet
$fileszam = count($lista);
if ($fileszam != 0)  //ha nem ures a konyvtar
{
  echo ("<table width =\"70%\" align=\"center\">");
  echo ("<tr><td colspan=\"2\">Kedvenc PHP függvényeim</td></tr>");

  for ($i=0; $i<$fileszam/2; $i++) //feltoltjuk a tablat
  {
    list($func, $nev, $kit) = explode(".", $lista[$i]);
    echo ("<tr><td><a
href=\"".$dirnev."/".$lista[$i]."\">".$nev."</a></td>");
    list($func, $nev, $kit) = explode(".", $lista[$i+$fileszam/2]);
    echo ("<td><a
href=\"".$dirnev."/".$lista[$i+$fileszam/2]."\">".$nev."</a></td></tr>
");

  }
  if ($fileszam % 2 != 0) //ha paratlan a fileszam, egy plusz sor kell
  {
    list($func, $nev, $kit) = explode(".", $lista[$i]);
    echo ("<tr><td><a
href=\"".$dirnev."/".$lista[$i]."\">".$nev."</a></td><td> </td></
tr>");
  }
  echo ("</table>");
}
?>
</body>
</html>
```

továbbfejlesztési lehetőségek

Ha egészen precíz megvalósítást akarunk, akkor ki kell cserélni a "-" karaktereket a függvénynevekben "_" karakterekre is (`ereg_replace()`), illetve meg kell oldani az egyéb kategóriák kezelését is, esetleg külön táblázatokat alkalmazva.

Dátum és idő megjelenítése

specifikáció

Az adatbázistól átvett dátum-idő adat (időbélyegző) alapján készítsük el a következő dátum és időpont kiíratást egy weblapon:
Az Ön utolsó látogatásának időpontja ezen az oldalon: 2002. február 24., 15:10

feladat

Gyakran kerül elő az a probléma, hogy akár egy adatbázisból vett adat esetében, akár a honlap futtatása, az egyes lapok lekérése esetén meg kell jelenítenünk a dátumot és az időt. Az adatbázis-kezelők többféle dátum és időformátummal dolgozhatnak és persze a nem kellő gondossággal megtervezett adatbevitelek esetén is létrejöhetnek furcsa formátumok, de összességében ajánlatos ragaszkodni az adatbázis-kezelő beépített formátumához és arra építeni fel a megjelenítő rutinokat - ezzel lehetőséget adunk a tárolt dátum-idő adat későbbi tetszés szerinti feldolgozására. Nem utolsó sorban így biztosíthatjuk, hogy az egyszer létrehozott dátum-idő megjelenítő modulunk kódja többször is felhasználható lesz. A végső cél egy fejlesztő részéről mindig a kód többszöri felhasználására törekvés, így kifejlesztendő dátum-idő modulunkat célszerű osztályként létrehozni és ennek példányosításával megoldani a megjelenítési igényeket. A gyakran használt Linux/Unix webszerverek dátumformátumát és az általánosan alkalmazott időbélyegzők formátumát figyelembe véve készítjük el kódunkat.

tervezés

Az adatbázistól átvett dátum-idő adatot most egy `$datum` változóba beolvasott sztringgel modellezzük és ez kerül feldolgozásra. A kiíratás az oldal tetején történik, grafikai vagy táblázatos struktúra nélkül. A hónapok magyar neveit tömbben tároljuk és onnét az indexük alapján vesszük ki.

megvalósítás (kód)

```
<?php

$honapok = array("", "január", "február", "március", "április", "május", "jú-
nius", "július", "augusztus", "szeptember", "október", "november", "decem-
ber");

$datum = "2002-02-24 15:10:32+01";

//szétszedjük dátumra, időre
list($datum_resz, $ido_resz) = explode(" ", $datum);

//eldobjuk az időzóna részt, a "+" jeltől a rész-sztringet kivágjuk
```

```
$ido_resz = strtok($ido_resz, "+");

// az idõ elemeit kivesszük változókba
list($ora, $perc, $mperc) = explode(":", $ido_resz);

//szétszedjük év, hó, nap értékekre, másként
$datum_resz = explode("-", $datum_resz);

// a hónap számot int értékké alakítjuk a tömb indexhez
$ho = intval($datum_resz[1]);

//kiírjuk az elõállított adatokat
echo ("Az Ön utolsó látogatásának időpontja a honlapon: ");
echo ($datum_resz[0].". ".$honapok[$ho]." ".$datum_resz[2].".,
".$ora.":".$perc);
```

továbbfejlesztési lehetőségek

A MySQL adatbázis-kezelő timestamp (időbélyegző) típusa nem a fenti, hanem 14 karakteren folytonosan írja ki az évszámot, hónapot, napot, órát, másodpercet, azaz egy tetszőleges időbélyegző így néz ki: "20040315084530". Írjunk egy függvényt, amely ezt a formátumot átalakítja a `$datum` változó formátumába.

Regisztráció, adatfelvitel, adatfelvitel ellenőrzés

specifikáció

Vegyünk fel különböző személyi és felmérési adatokat és azok realitását, meglétét ellenőrizzük le, mielőtt az adatbázisba bevittük volna. Az ellenőrzés alá vont beviteli mezők legyenek egyszerűen felvehetők, módosíthatók.

a feladat elméleti háttere

Az adatbevitel űrlap segítségével történik és már ennek megszerkesztésekor célszerű a bevitelre kerülő adatok korlátozása, azaz ahol csak lehet, adatválasztást kínáljunk fel. Nem szabad elfelejteni, hogy a HTML form (űrlap) objektumok készlete szegényes más programnyelvek komponenseivel összehasonlítva, és bár JavaScript segítségével gyakorlatilag bármi megvalósítható, annak értelmezése erősen böngésző-függő és nem várható egységesítése a közeli jövőben ☹. Törekedni kell tehát az egyszerű és ötletes megoldásokra (a primitív most kifejezetten előnyös) és mindig ki kell egészíteni a bevitelt szerver oldali adatellenőrzéssel. Célszerű amit lehet legördülő listából felkínálni, a listák feltöltése könnyen megvalósítható egyszerű szövegfájlokból és/vagy adatbázisokból (itt elsősorban nem a komoly adatbázis-kezelőkre gondolok, hanem a fájlból dolgozó adatbázisokra).

Tipikus beviteli feladat személyes adatoknál a dátumok felvitele, ez jól és biztonságosan megoldható például három legördülő listával az évek, hónapok és napok felvitelére. Igény szerint a gyakran használt dátum (például az aznapi dátum) könnyen be is állítható alapértelmezettként egy bizonylat kitöltése esetén, ahol általában, de nem feltétlenül az aznapi dátumot kell megadni. Országnevek a hivatalos megnevezés szerint, kategóriák mind legördülő listából szedhetők.

Ha egy beviteli mezőhöz több egyedi adat tartozik - például többféle érdeklődés - ezek egyszerűen megoldhatók, ha a legördülő listát többszörös kiválasztásúvá tesszük és a mögötte elhelyezkedő változó tömb lesz. Ehhez csupán a `select` mező name attribútumában a változó név mögé szögletes zárójelet kell írni:

```
<select name="ertekek[]" size="3" multiple>
      <option value="érték1">címke1</option>
      <option value="érték2">címke2</option>
      <option value="érték3">címke3</option>
    </select>
```

Legördülő lista kódja többszörös kiválasztás biztosítása esetén

Megjegyzem, hogy szinte bármelyik, erre egyébként alkalmas beviteli mező esetén alkalmazható ez a technika! Össze lehet így kapcsolni jelölődobozokat (checkbox), szöveges beviteli mezőket (text), amikor több elem neve (az értéket tartalmazó változó) azonos és a változó tömb jellegét a név mögé írt szögletes zárójel mutatja. Ezt a lehetőséget a HTML szerkesztők nem támogatják, azaz nekünk kell figyelni arra, hogy ilyen beállításokat tegyünk. Tömbök adatbázisba vitele azonban nem megy közvetlenül, vagy mi írjuk meg a tömböt sztringgé alakító rutint, vagy a PHP `serialize()` függvényét használjuk, ennek kimenete egy, az adatbázisba egyszerűen bevihető speciális sztring és a tárolt információ alapján az `unserialize()` segítségével az eredeti tömb helyreállítható belőle.

Talán a legfontosabb feladat az adatellenőrzés korrekt megoldása. Alapelv, hogy maximális szabadságot biztosítsunk a felhasználónak, de a fontos adatok esetén ez a szabadság elkerülhetetlenül csorbul. Ugyanakkor a leggondosabb tervezés sem zárja ki azt, hogy a - tréfás kedvű, vagy csak egyszerűen rosszindulatú - felhasználó valódinak látszó, de értelmetlen adatokat küldjön be. Ebből következik a szabadon hozzáférhető webes felületen át küldött adatok korlátozott megbízhatósága, ami növelhető bizonyos trükkökkel, de teljesen nem valósítható meg. Célunk a felhasználó érdekeltté tétele a precíz adatszolgáltatásban.

Ellenőrizhető viszont minden további nélkül az adatok megadása. Gyakorlati tapasztalat, hogy egy hosszabb űrlap esetében gyakran marad ki egy-egy lényeges adatmező és erre feltétlenül figyelmeztessük a felhasználót! Sokszor az is egyszerűen ellenőrizhető, hogy egy-egy adat elfogadható-e, vagy téves begépelés eredménye. Jelöljük meg valami módon a kötelező adatokat és azokat ellenőrizzük le, a többi esetében az üres mező viszont elfogadható. Ezek ellenőrzésének a legegyszerűbb módja a feldolgozás logikai döntés mögé rejtése, ha a feltétel nem teljesül, visszakapja a felhasználó az űrlapot adathelyesbítésre. Néhány adat esetében ez tökéletesen működik, de ha már 8-10 vizsgálandó adat van, senki sem akar ennyi logikai részfeltételt megadni a feldolgozáshoz vezető szelekciós ágban.

Egyszerű megoldás az úgynevezett kötelező mezők neveinek (változónevek!) tömbbe gyűjtése és tárolása. Felveszünk még egy hibákat tartalmazó asszociatív tömböt is, amely az észlelt hibák szöveges leírását fogja tartalmazni Egy ciklus - célszerűen egy **for** ciklus, vagy **foreach** szerkezet - végigmegy a kötelező mezők tömbön és a PHP dinamikus változó kezelését felhasználva ellenőrzi, hogy a felsorolt mezők értéke megfelel-e a kívántaknak, ha nem, az asszociatív hiba tömbbe "belenyom" (`array_push()`) egy aktuális hibaüzenetet, ahol az asszociatív tömb elemének kulcsa lesz a beviteli mező neve, mint azonosító és a hibaüzenet a mezőhöz tartozó érték. Ezzel lehet elegánsan megoldani, hogy az egyes beviteli mezőkhöz tartozóan jelenítsük meg a hibaüzenetet és ne egy olyan, a JavaScript **alert** metódusával közvetített általános üzenet jelenjen meg, hogy "Kérjük az összes mező kitöltését!". Most már csak a hiba tömböt kell ellenőrizni, ha nincs benne érték, mehet a feldolgozás, ha van, akkor a megfelelő helyre egy kiírást készítünk a hibáról.

Az űrlapok elküldését általában POST metódussal végezzük, ha adatbázisba bevitelről van szó, akkor a legegyszerűbb esetben a **form** önmagát hívja meg (**action** mező üres vagy a fájl saját neve), azaz újra lefut az egész űrlapot tartalmazó .php fájl. Számolnunk kell azzal tehát, hogy egy-egy hiba esetén a felhasználó nem akarja újra kitölteni az összes adatot, így ne feledkezzünk meg arról, hogy a már elfogadottakat írjuk vissza a megfelelő űrlap mezőkbe `echo()` vagy `print()` utasítások segítségével.

tervezés

Az első feladat az űrlap ergonómikus megtervezése. Legyen logikus, egyszerűen átlátható és egyértelmű, kerüljük a felesleges magyarázatokat. Az egyszerűség kedvéért

ezt általában HTML szerkesztőben tervezzük meg, ott lehetőség van a forma pontos beállítására. Esetünkben nevet, születési évet, az illető nemét kérdezzük meg, rákérdezünk jelölődobozok segítségével szakértelmére és gördíthető lista segítségével szoftverismeretére. Ezeket mind egy táblázat soraiban, celláiban helyezzük el, hogy egyszerűen áttekinthető, kezelhető legyen. Az ismétlődő felviteli mezők kirakását is PHP kód segítségével végezzük.

Az adatfelvitel adatbázisba történik, így a kapcsolódás utasításait - hiszen ezt adott rendszerben sokszor használjuk - külön fájlban helyezzük el, amit egy `include()` utasítás visz be a kódba (`connect_local.php`).

megvalósítás (kód)

connect_local.php fájl kódja

```
<?php
  //konfigurációs rész
  // host neve
  $host = "localhost";
  // adatbazis felhasznaloi neve - a Windows felhasznalo neve
  $username = "24-t";
  // adatbazis felhasznaloi jelszava - a Windows belepesi jelszo
  $userpass = "";
  //adatbazis neve - mindig az aktualis adatbazis neve kerul ide
  $database = "adatok";

  //kapcsolati rész
    //kapcsolat letrehozasa a mysql adatbazis kezelovel, illetve
hibauzenet
    $conn = mysql_connect ("$host", "$username", "$userpass")
    or die ("<br>nem lehet csatlakozni a kiszolgalohoz");

  //ellenorizzuk a kerdeses adatbazis letet
  if (!mysql_query("use $database"))
    echo ("<script> alert('Az adatbazis nem áll készen!');
</script>");
?>
```

urlap.php fájl kódja

```
<?php
/**Denes Medzihradszky
 * @version 1.0
 * @copyright 2004
 **/

//hibakat tartalmazo tombot deklaralunk
$hibak = array();
//kulcsmezot jelolunk ki
if ($nev != "")
{
  //atalakitasok es ellenorzesek
  //tabulatorral tagolt sztringbe fuzzuk a jelolodobozokra adott
valaszokat
```

```
  if (isset($szakertelem))
  {
      $szakert = "";
    for ($i=0; $i<count($szakertelem); $i++)
      if ($i == count($szakertelem)-1)
        $szakert .= $szakertelem[$i];
      else
        $szakert .= $szakertelem[$i]."\t";
  }
  else
    $hibak["szakertelem"] = "Nem adott meg szakértelmet!";

  //sztringbe fuzzuk a gorditheto listara adott valaszokat is
  if (isset($szoftverek))
  {
      $szoftver = "";
    for ($i=0; $i<count($szoftverek); $i++)
    if ($i == count($szoftverek)-1)
      $szoftver .= $szoftverek[$i];
    else
      $szoftver .= $szoftverek[$i]."\t";
  }
  else
    $hibak["szoftverek"] = "Nem adott meg szoftvert!";

  //a nevet korrekt formara alakitjuk, a szavak elso betuje legyen
nagybetu
  $nev = ucwords(strtolower($nev));

  //regexp segitsegevel ellenorizzuk a szuletesi evet (19xx lehet)
  if (!ereg("^19([0-9]{2})", $szulev))
    $hibak["szulev"] = "Nem jó a születési év!";

  //ha minden rendben, feltoltheto az adatbazis
  if (count($hibak) == 0)
  {
     include("connect_local.php");
    $sql = "insert into szemelyek (nev, szulev, nem, szakertelem,
szoftverek) ";
    $sql .= "values ('$nev', '$szulev', '$nem', '$szakert',
'$szoftver') ";
    mysql_query($sql);
    mysql_close($conn);
    //ertesites es atiranyitas mas lapra
    echo ("<script language=JavaScript1.2>");
    echo ("alert('Az adatokat felvittük');");
    echo ("location.reload('tovabb.php');");
    echo ("</script>");
  }
  else
   //ez itt csak tesztelesi celokra van, kiiratjuk a hibakat!
    while (list($kulcs, $ertek) = each($hibak))
      echo ($ertek."<br>");
}
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
<title>Regisztráció</title>
```

```
</head>
<body bgcolor="#FFFFFF">
<p align="center">Adatbevitel</p>
<form method="POST" action="">
    <div align="center">
     <table width="700" cellspacing="0" cellpadding="6" border="0">
    <tr>
      <td  width="267" align="right">név</td>
      <td  width="409"><input type="text" name="nev" size="30"
maxlength="30"></td>
    </tr>
    <tr>
      <td width="267" align="right">születési év</td>
      <td  width="409"><input type="text" name="szulev" size="4"
maxlength="4"></td>
    </tr>
    <tr>
      <td width="267" align="right">nem</td>
      <td  width="409">
      <input type="radio" value="férfi" checked name="nem">férfi
      <input type="radio" name="nem" value="nõ">nõ</td>
    </tr>
    <tr>
      <td width="267" align="right">szakértelem</td>
      <td width="409">
      <div align="center">
    <?
    //checkbox csoport kirakasa tomb ertekek alapjan
    $szakert_tomb = array("video szerkesztés", "audio szerkesztés",
      "grafika", "programozás", "animáció", "webszerkesztés");
    echo ("<table border=\"0\" cellpadding=\"0\" cellspacing=\"0\">");
    //hogy ne kelljen olyan hosszu sort irni...
    $input = "<input type=\"checkbox\" name=\"szakertelem[]\"";
    for ($i=0; $i<count($szakert_tomb); $i+=2)
    {
      echo ("<tr><td>".$input."
value=\"".$szakert_tomb[$i]."\">".$szakert_tomb[$i]."</td>");
    echo ("<td>".$input."
value=\"".$szakert_tomb[$i+1]."\">".$szakert_tomb[$i+1]."</td></tr>");
    }
    //paratlan elemszam eseten ujabb sort kell kirakni itt a ciklus
utan!
    echo ("</table> ");
    ?>
      </div>
      </td>
    </tr>
    <tr>
      <td height="19" width="267" align="right">ismert szoftverek</td>
      <td height="19" width="409">
    <?
    //tobbszoros kivalasztasu lista kirakasa
    $szoftvertomb = array("MS Office", "Corel", "Adobe Photoshop",
"Dreamweaver", "After Effects", "Flash");
    echo ("<select size=\"4\" name=\"szoftverek[]\" multiple>");
    foreach ($szoftvertomb as $szoftver)
      echo ("<option value=\"".$szoftver."\">".$szoftver."</option>");
    echo ("</select>");
    ?>
    </td>
```

```
    </tr>
    <tr>
      <td  width="267"> </td>
      <td  width="409">
      <p align="center"><input type="submit" value="Elküldés"
name="submit"></td>
    </tr>
  </table>
    </div>
</form>
</body>
</html>
```

továbbfejlesztési lehetőségek

Kérdezzük meg az illető szakértelmi szintjét rádiógomb csoport segítségével (úgynevezett Likert-skálát valósítottunk meg ezzel) és mind a jelölődobozokból - ezekhez kapcsoljunk hozzá egy szövegmezőt, amelybe az egyéb szakértelmét írhatja be a felhasználó - az egyéb mezővel együtt készítsünk osztályt, hogy legközelebb ne kelljen ennyit gépelnünk, csak az osztályt fogjuk példányosítani, paraméterként átadva az értékeket (célszerűen egy tömb) és a változó nevét, esetleg egyéb, az elrendezésre utaló adatokat.

Többnyelvű oldalak - egy regisztrációs űrlap 4 nyelven

specifikáció

Egy nemzetközi oldalt kell készítenünk, ahol a bejelentkezéskor nyelv választható (magyar, angol, német, francia) és a regisztrációs kérdőívet ennek alapján kínáljuk fel. Vegyünk fel személyi adatokat és vigyük azokat adatbázisba, de regisztráljuk az illető nyelvi preferenciáját is. Később majd a bejelentkezéskor automatikusan a számára "érthető" nyelven kínáljuk a tartalmakat. Adatellenőrzést most nem kell végeznünk.

a feladat elméleti háttere

A web indulása óta ismert a probléma, a dinamikus lehetőségek megteremtése előtt ezt a lapok párhuzamos nyelvi változatainak elkészítésével lehetett csak megoldani, illetve a GET metódussal átadott nyelvre utaló változóval JavaScript segítségével is lehetett alternatív kiírásokat is tenni - ezt utóbbi bonyolult kódot és a JavaScript rendelkezésre állást igényelte (ezt viszont a felhasználó szabályozza) míg az első változatot, különösen annak módosítását, senkinek sem kívánom. Dinamikus megoldás többféle is van, a helyzettől függő lehetőségekkel. Valami olyan megoldás kellene, ami szótárszerű, egyrészt ez van legközelebb az emberi gondolkodáshoz, másrészt mindig látjuk,

hogy mi helyett mi áll - nem kell nekünk ismerni az összes nyelvet, bárki elkészítheti helyettünk a nyelvi variációkat. Továbbá így bővíthető is lehet, csak egy szótár fájlt kell kicserélni és máris rendelkezésre áll az új nyelvi változat. Az oldal grafikai-szerkezeti váza viszont állandó lehet. Az adott oldal nyelv specifitását egy átadott változó vagy adatbázisból a minden oldal elején kiolvasott változó biztosíthatja. Előre látható, hogy switch vezérlő szerkezetet érdemes használni, mert sok párhuzamos választás van és az is elképzelhető, hogy egyes képként megvalósított feliratokat is nyelvfüggővé kell tenni. Használjuk ki a PHP asszociatív tömbjeit, mert az pontosan a kívánt rendszer szerint dolgozik! Egy sztring - a magyar elnevezés - az index és értéke a kívánt nyelvi változat. Ez egyúttal azt is jelenti, hogy a kódban az index keretében mindig látni fogjuk a magyar jelentést, nem kell külön keresgélni!

tervezés

Elkészítjük a regisztrációs oldal vázát, magyar feliratokkal. Minden nyelvnek készítünk egy-egy specifikus fájl (**magyar.inc**, **angol.inc**, **nemet.inc**, **francia.inc**), amely egyetlen **$szotar** nevű változóban asszociatív tömböt tartalmaz, amelynek indexei a már megfogalmazott magyar feliratok, értékei az adott nyelv megfelelő feliratai. Ne felejtsük el, hogy kell egy magyarról-magyarra fájl is, mert a magyar változatot ugyanezen technikával kell megvalósítani az egység érdekében! Ezeket a fájlokat **include** utasítással visszük majd be, egy **switch** vezérlőszerkezet keretében. Így a **$szotar** változónak mindig az adott nyelvnek megfelelő tömbértékei lesznek. A nyelv specifikus változót - **$lang** - most egyszerűen a bejelentkező oldalról (**index.html**) kapjuk a link kiegészítésével, az URL-hez kapcsoltan (ebben semmi titkolandó nincs). Az regisztrációt egy külön, felbukkanó ablakban végezzük, amit viszont JavaScript segítségével hívunk meg. Az adatok felvétele után biztosítani kell a visszalépést, továbbá el akarjuk tüntetni az oldal ismételt lefutásakor a beviteli mezőket - ez megoldható többféleképpen, például a HTML rész megjegyzésbe zárásával, kommentezésével, amelynek kiíratása viszont egy logikai változó értéke szerint fog megtörténni, vagy egy `exit()` utasítás segítségével, amikor a kód végrehajtása megszakad és a HTML rész már nem jelenik meg az oldalon.

megvalósítás (kód)

regform.php

```
<?php
/* ez a kod a nyelvi valtozatokat hivatott lekezelni, az atvett $lang
valtozo
```

```
*  alapjan egy include beleviszi a megfelelo $szotar valtozot */

$rendben = false;
switch($lang)
{
  case "ger" :
        include("nemet.inc") ;
        break ;
  case "fra" :
        include("francia.inc") ;
        break ;
  case "eng" :
        include("angol.inc") ;
        break ;
  default :
        include("magyar.inc") ;
}
// eddig a nyelv kivalasztasa
?>
<html>
<head>
<title>Regisztráció</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
</head>
<?
if(isset($vnev) && $vnev != "")
{
// Gyartunk egy random jelszot, amit majd kikuldunk
srand((float) microtime() * 10000000);
$pswd = strval(rand(10000, 100000));

// Benyomjuk a torzsadatokat es atalakitjuk a kisbetusen irt neveket
$knev = ucwords(strtolower($knev));
$vnev = ucwords(strtolower($vnev));

//osszeallitjuk az email cimet
$fullemail = "$email_azon@$email_path";

//osszeallitjuk az adatbazisba bevitelre kerulo ertekekkel az sql
//utasitast

/*elkuldjuk emailben a jelszot es a logint,
* ez csak akkor mukodik, ha a szerveren illetve a php.ini-ben be van
* allitva a mail program
* mail($fullemail, "Regisztráció", "Loginja: $login, Ideiglenes jel
* szó: $pswd");
* helyette: */
echo("E-mail címe: ".$fullemail."<br>Regisztráció <br> Loginja:
$login, Ideiglenes jelszó: $pswd");

//ertesitjuk a felhasznalot
echo ("<body bgcolor=\"#FFFFFF\"><p>Az adatokat rendben felvet-
tük</p>");
echo ("<a href=\"javascript:void()\"
onClick=\"window.close();\">&gt;&gt; vissza az index lapra
&gt;&gt;</a>");
echo ("</body>");
//ezzel tuntetjuk el a HTML kodot sikeres adatfelvetel eseten
$rendben = true;
```

```
}
if ($rendben) echo ("<!--");
?>
<body  >
<p align="center"><b><? echo $szotar["Regisztrációs adatok (a *-gal
jelölt kötelezõ)"];?></b><br>
</p>
<form method="POST" action="">
  <table width="500" border="0" cellspacing="0" cellpadding="6"
align="center">
    <tr>
      <td width="192"><? echo $szotar["vezetéknév*"]; ?></td>
      <td width="284">
        <input type="text" name="vnev" value="<? echo $vnev; ?>">
      </td>
    </tr>
    <tr>
      <td width="192"><? echo $szotar["keresztnév*"];?></td>
      <td width="284">
        <input type="text" name="knev" value="<? echo $knev;?>">
      </td>
    </tr>
    <tr>
      <td width="192"><? echo $szotar["megjelenõ név (login)*"];?>
      </td>
      <td width="284">
        <input type="text" name="login" maxlength="12" value="<? echo
$login;?>">
      </td>
    </tr>
    <tr>
      <td width="192"><? echo
$szotar["foglalkozás/szakterület*"];?></td>
      <td width="284">
        <input type="text" name="szakma" value="<? echo $szakma;?>">
      </td>
    </tr>
    <tr>
      <td width="192">e-mail*</td>
      <td width="284" valign="middle">
        <input type="text" name="email_azon" size="8" value="<? echo
$email_azon;?>">
        @
        <input type="text" name="email_path" size="20" value="<? echo
$email_path; ?>">
      </td>
    </tr>
    <tr>
      <td width="192"><? echo $szotar["telefon"];?></td>
      <td width="284">
        <input type="text" name="tel" value="<? echo $tel;?>">
      </td>
    </tr>
    <tr>
      <td width="192"><? echo $szotar["város"];?>*</td>
      <td width="284">
        <input type="text" name="varos" value="<? echo $varos;?>">
      </td>
    </tr>
    <tr>
```

```
      <td width="192"><? echo $szotar["irányítószám, házszám, ut-
ca"];?></td>
      <td width="284">
        <input type="text" name="cim" value="<? echo $cim;?>">
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <div align="right">
          <input type="hidden" name="lang" value="<? echo $lang;?>">
          <br>
          <input type="submit" name="submit" value="  <? echo
$szotar["regisztrálás"];?>  ">
        </div>
      </td>
    </tr>
  </table>
</form>
<? if ($rendben) echo ("-->"); ?>
</body>
</html>
```

továbbfejlesztési lehetőségek

Egy bonyolultabb rendszerben egyszerűbb, ha **session** segítségével kezeljük a **$lang** változót és minden oldal elején a felhasználót azonosítva töltjük be számára a megfelelő nyelvi oldalt. Előfordul az is, hogy nyelv specifikus grafikát használunk. Ebben az esetben a képek fájlneveihez adjunk hozzá egy nyelvre jellemző elemet és ez a **switch** szerkezetben kapjon értéket, majd a képek a megjelenítés helyén ezzel a változóval kiegészített URL-t kapjanak.

Névnapozás, az aktuális névnap kiírása[17]

specifikáció

Készítsünk weboldalunk nyitólapjára az adott naphoz tartozó névnapokat feltüntető modult. Törekedjünk az objektum-orientált megközelítésre!

a feladat elméleti háttere

Adatbázisban fogjuk tárolni a névnapok dátumhoz rendelését biztosító rekordokat. Ezek egy azonosító mellett - bár a hónap sorszáma és a nap sorszáma is egyedi kulcsot ad - tárolni fogják a hónap sorszámát, a nap sorszámát és karaktersorozat formájában, közvetlenül megjeleníthető módon az adott naphoz tartozó névnapo(ka)t, illetve bizto-

sítsunk lehetőséget állami ünnep esetén az ünnepnap nevének megjelenítésére is. A rendszertől lekérjük a dátumot és azt feldolgozva előállítjuk a hónap és nap értékeket, amelyek segítségével azonosíthatjuk a megfelelő rekordot.

tervezés

Az adatbázis tábla az azonosítón (id) kívül négy mezőből áll, ezek: hónap (honap), nap (nap), névnap (nevnap), ünnep (unnep). Létrehozható egy már meglevő adatbázis névnap táblájaként .sql fájlból, vagy a táblát külön létrehozva, egy Excel táblázatból tabulátorral elválasztott formában lementett szöveges fájlból az SQL copy parancsával. Először elkészítünk egy strukturált megvalósítást, majd ezt kiegészítve, átdolgozva elegáns megoldásra törekszünk, - csak nem fogunk külön kódot írni mindegyik alkalommal ha használni kívánjuk a rendszert! (a gyakorlatban a webhely több oldalán is meg kell jelentetni a névnap információkat, nem csak a főlapon), - tehát osztályt készítünk és annak példányosításával állítjuk elő a keresztnevek illetve az esetleges ünnep kimenetét. A használandó adatbázis referenciáját, a dátum értéket paraméterként adjuk át, a kimenet sztring.

megvalósítás (kód)

továbbfejlesztési lehetőségek

(a) Vegyük figyelembe a szökőévek eltérő névnap hozzárendelését (hivatalosan február 24. a szökőnap, így szökőévben a februári névnapok 23. után a 24.-t kihagyva, egy nappal csúsznak), és írjuk meg a szökőéveket megadó algoritmust[18]. Ebben az esetben vizsgálnunk kell az év értékét is a rendszerdátumból.

---

[17] Ne felejtsük el, hogy a névnapok ünneplése nem általános szokás még az úgynevezett nyugati kultúrájú világban sem! Ne próbáljunk egy angolszász közösségnek névnapozást csinálni, mert nem fogják tudni, miről is van szó… Ez a kérdés felmerült egy webfejlesztői fórumon, ezért hívom fel rá itt is a figyelmet!

[18] Törekedjünk a precizitásra és gondoljunk a jövőre ☺! Nem csak egyszerűen a néggyel osztható éveket kell figyelembe venni, mert nem szökőévek a százzal osztható évek, ugyanakkor azonban szökőévek a négyszázzal oszthatóak. Ugyan a következőre sokat kell várni, de 1900 történetesen nem volt szökőév, míg 2000 az volt, így van némi ráció ennek figyelembe vételében egy általánosan használható algoritmus kidolgozása során.

(b) Nem mindig áll rendelkezésünkre szerveren telepített adatbázis! Dolgozzunk ki olyan adattárolási formá(ka)t, - természetesen a kiolvasó rutinokkal együtt - amelyek segítségével adatbázis nélkül is megoldható a névnapok kikeresése.

Dátum és időkezelés felsőfokon

specifikáció

Készítsünk különböző adatforrásokból, különböző magyar nyelvű dátum és idő kiírási formákat megvalósító osztályt. Az osztály legyen továbbfejleszthető, azaz szükség esetén újabb és újabb kiírásokat megvalósítható metódusokat adunk majd hozzá és újabb adatforrásokat is be kell tudnunk építeni a rendszerbe.

a feladat elméleti háttere

A magyar nyelvű dátum kiírásokhoz az objektum adatai között kell, hogy szerepeljenek a hónapok nevei és a hét napjainak nevei (célszerűen egy-egy tömb elemeiként). A számításba jövő adatforrások: Linux/Unix rendszeridő, Windows rendszeridő, PostgreSQL adatbázis és MySQL adatbázis **timestamp** adatformátuma. Az objektum létrehozása a konstruktor metódussal történik, ennek lefutása feltölti a hét napjai és a hónapok tömböt, majd az objektumot a kívánt dátum-idő értékre állítjuk a megfelelő `init()` metódus meghívásával. Célszerű a különböző adatformátumokat egységes szerkezetbe hozni és egy osztályszintű változóban tárolni. Ennek alapján következhetnek a kiírások, először csak néhány formátumot készítünk el, de ez később igény szerint bővíthető. Az osztály kódja egy külön .inc fájlban kap helyet, alkalmazáskor ezt az `include_once()` szerkezettel visszük be a kódba. Fontos az `include_once()` változatot használni, hogy az osztály kódja biztosan csak egyszer szerepeljen a teljes kódban.

tervezés

Az osztály neve legyen Mydatum. Az objektum adatai: `$honapok`, `$napok` tömbök, `$datum` változó, amely a közös formátumra hozott dátumértéket fogja tárolni (ez ebben az esetben a UNIX dátum/idő kezelésének megfelelően az 1970. január 1 óta eltelt másodpercek számát fogja tárolni). Az objektumot létrehozó konstruktor az osztály nevével karakterre megegyező nevű metódus - ebben az esetben a Mydatum() metódus - ez tölti fel az adat tömböket és az utána meghívott most_init() illetve mysql_init() metódus a kívánt dátum/idő értékre állítja be az objektumot (rendszeridőre illetve a MySQL

adatbázisból visszakapott időbélyeget feldolgozva az adatbázisban tárolt időre. A kiíró metódusok a következő dátum-idő formátumokat adják vissza:

`ev_ho_nap_napneve()`	2004. április 22. csütörtök
`ev_honap_nap()`	2004. április 22.
ev_ho_nap()	2004-04-22
`honap_nap()`	április 22.
`ora_perc()`	15:14

megvalósítás (kód)

mysql_datum.inc fájl - a Mydatum osztály kódja

```
<?php
/* copyright 2004; Denes Medzihradszky, Peter Szabo  */
/* Osztaly, ami mysql datumformatumbol is tud magyar datum-ido
* kiirasokat csinalni. */
class Mydatum
{
  var $datum, $honapok, $napok;

  function Mydatum() //konstruktor, ami erteket ad a honap, napok
tomboknek
  {
    $this->honapok = array("", "január", "február", "március", "ápri-
lis", "május", "június", "július", "augusztus", "szeptember", "októ-
ber", "november", "december");
    $this->napok = array("vasárnap", "hétfõ", "kedd", "szerda", "csü-
törtök", "péntek", "szombat");
  }

  function most_init()  //a jelenlegi datumra allitja az objektumot
  {
    $this->datum = time();
  }

  function mysql_init($datum)  //mysql timestamp adatformatumot vesz
at parameterkent
  {
    $ev = intval(substr($datum, 0,4));
    $ho = intval(substr($datum, 4,2));
    $nap = intval(substr($datum, 6,2));
    $ora = intval(substr($datum, 8,2));
    $perc = intval(substr($datum, 10,2));
    $mperc = intval(substr($datum, 12,2));
    $this->datum = mktime($ora, $perc, $mperc, $ho, $nap, $ev);
  }

  function ev_ho_nap_napneve()  //ev. honap nap., het napja
formatumban irja ki
  {
```

```
    return date("Y", $this->datum).". ".$this->honapok[ intval( date(
"m", $this->datum ))]." ".intval( date( "d", $this->datum )). "., ".
$this->napok[ date( "w", $this->datum )];
  }

  function ev_honap_nap() //ev. honap nap. formatumban irja ki
  {
    return date("Y", $this->datum).". ".$this->honapok[ intval( date(
"m", $this->datum ))]." ".intval( date( "d", $this->datum )). ".";
  }

  function ev_ho_nap() //ev-ho-nap formatumban irja ki
  {
    return date("Y", $this->datum)."-".date( "m", $this->datum )."-
".date( "d", $this->datum );
  }

  function honap_nap() //honap nap. formatumban irja ki
  {
    return $this->honapok[ intval( date( "m", $this->datum ))]."
".intval( date( "d", $this->datum )).".";
  }

  function ora_perc() //ora:perc formatumban irja ki
  {
    return date("H", $this->datum ).":".date("i", $this->datum );
  }

}
?>
```

mysql_datum.php fájl - a megjelenítést végzi

```
<?php
include ("mysql_datum.inc");
$dt = new Mydatum();
?>
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
</head>
<body bgcolor="#FFFFFF">
<?
//adatbazis idobelyegzot modellezunk
$db_stmp = "20040505203452";
//"adatbazisbol" vett datumra allunk ra
$dt->mysql_init($db_stmp);
echo $dt->ev_ho_nap_napneve();
echo "<br>";
echo $dt->ev_honap_nap();
echo "<br>";
echo $dt->ev_ho_nap();
echo "<br>";
echo $dt->honap_nap();
echo "<br>";
echo $dt->ora_perc();
echo "<br>";

$dt->most_init(); //a jelenlegi datumra allunk ra
```

```
echo $dt->ev_ho_nap_napneve();
?>
</body>
</html>
```

továbbfejlesztési lehetőségek

Írjunk olyan metódust, amely átvesz két dátum-időt reprezentáló objektumot és a különbségüket írja ki mettől-meddig formátumban, de csak azokat az elemeket, amelyekben eltérés van, azaz például 2004-03-15 és 2004-03-17 esetében a kiírás március 15-17. lesz!

Öröklődéssel készítsünk új osztályt, amely más nyelveken is képes dátumokat megjeleníteni. Ehhez természetesen felül kell írni a konstruktorként működő metódust és egyik-másik kiíró metódust is.

Reguláris kifejezés alkalmazása beviteli mezők ellenőrzésére

specifikáció

Oldjuk meg az email cím beviteli ellenőrzését[19], illetve biztosítsuk hogy egy beviteli mezőbe csak telefonszámot (annak jellemző karaktereit és elrendezését) lehessen bevinni. Próbáljuk megvalósítani a lehető legjobb megközelítést, ha tökéleteset nem is tudunk alkotni!

a feladat elméleti háttere

Mindenki használt már reguláris kifejezést, ha nem is tudta róla, hogy így nevezik. Az egyszerűbbek közül való az, amikor egy fájlnév keresésben "??nap*.doc" mintát adunk meg, mert rákeresünk minden olyan fájlnévre, amely két tetszés szerinti karakter után a "nap" részláncot tartalmazza, majd tetszés szerinti hosszúságú karaktersorozat jöhet és a kiterjesztése "doc" legyen. Ezt továbbfejlesztve a szövegfeldolgozásra szakosodott nyelvekben (Perl, PHP) standardizált jelöléseket vezettek be és így gyakorlatilag bármilyen mintát illeszthetünk a megfelelő függvény segítségével (`ereg()` és a kisbe-

[19] Ennek a reguláris kifejezésnek az eredetijét az interneten találtam, majd programozó ismerősöm alkalmazta - tőlem függetlenül szintén az internetről begyűjtve - egy űrlap ellenőrzés elkészítése során. Az űrlap tesztelésénél azonban kiderült, hogy nem működik jól, sőt az eredeti is hibás, így az itt közreadott már a javított változat. Ez mellesleg igen jól alátámasztja azt a gyakorlati tapasztalatot, hogy soha ne fogadjuk el a "talált" kódokat kritika és alapos tesztelés nélkül!

tű/nagybetű különbségeket figyelembe nem vevő `eregi()` függvények), ha a minta illeszkedik, a függvény visszatérési értéke igaz, ha nem, értelemszerűen hamis.

Ezekkel a jelölésekkel megadhatjuk, hogy milyen karakterek, azokból egymás után mennyi fogadható el, hányszor ismétlődhet egy-egy elem, és még sok mást is előírhatunk. Előírhatjuk azt is, hogy hova illeszkedjen az elem, a karaktersorozat elejére, avagy a végére. Igazán jó reguláris kifejezést írni művészet, mert át kell látni az illeszkedés törvényszerűségeit és előre tudni kell, hogy melyek lehetnek a megengedhető esetek.

Természetesen egy reguláris kifejezéses illeszkedés nem jelenti azt, hogy az email például valós cím, ez a módszer csak a formai ellenőrzésre alkalmas. Ha fontos számunkra, hogy a felhasználó valóban a saját email címét adja meg, tegyük ebben érdekeltté, például a belépési jelszót vagy a regisztrációs felület linkjét a megadott email címre küldjük el, esetleg még időkorlátot is felállíthatunk a bejelentkezéséhez.

tervezés

Először is, azt kell belátnunk, hogy milyen egy email cím lehetséges formátuma. Az első részben a "@" jelig betűk és számok lehetnek, betűvel kell kezdődjön és pont vagy aláhúzás karakter lehet benne. Egy és csak egy "@" karakter legyen benne, majd a domén név részben tetszés szerinti számú aldomén lehet ponttal elválasztva, amelyek tartalmazhatnak még kötőjelet, de aláhúzás karaktert nem. Végül a cím lezárása olyan karaktersorozattal történik, amely a pont után minimum két karaktert tartalmaz. Nincs helye különleges karaktereknek sem, általában nem engednek meg speciális karaktereket a szolgáltatók. Erre kell tehát mintát írni.

A telefonszám esetében a szám kezdődhet plusz jellel és/vagy zárójellel és természetesen számmal (országkód és körzetkód) amit lezárhatunk zárójellel, és amit követhet üres karakter majd ismét szám, de ezeket megszakíthatjuk kötőjellel vagy üres karakterekkel. Betű viszont egészen biztosan nem szerepelhet benne.[20]

[20] Egyenlőre! A telefonszámok könnyebb megjegyezhetősége érdekében az Egyesült Államokban már elég gyakori a betűkkel kifejezett telefonszám, amikor a cég rövid nevének vagy szlogenjének megfelelő számokat választják telefonszámként. Ezzel most egyenlőre nem foglalkozunk…

megvalósítás (kód)

ellenorzes.inc fájl kódja

```
<?php
//ellenorzo funkciok regexp alapon
function check_email($email)
{
  if(ereg("^([0-9,a-z,A-Z]+)([.,_,-][0-9,a-z,A-Z]+))*[@]([0-9,a-z,A-
Z]+)([.,-][0-9,a-z,A-Z,-]+))*[.][0-9,a-z,A-Z]){2}([0-9,a-z,A-Z])?$",
$email))
  return true;
}

function check_telszam($telszam)
{
  if (ereg("^([0-9,\(,+]+)([-,\(,\)][0-9]+))*[/,0-9][0-9]+)?$",
$telszam))
  return true;
}
?>
```

továbbfejlesztési lehetőségek

Tervezzünk meg egy ellenőrző osztályt, amelynek különböző metódusai lesznek az egyes gyakran használt beviteli értékek reguláris kifejezéseken alapuló ellenőrző függvényei. A jól megfogalmazott és kitesztelt függvényekkel szinte mindent le lehet ellenőrizni, és nem csak a szerver oldalon, mert a JavaScript is ismeri és kezelni tudja a reguláris kifejezéseket!

Bannercserélők

specifikáció

(a) Véletlenszerűen akarunk megjeleníteni **Flash** segítségével készített bannereket, azaz olyan hirdetési célokat szolgáló képeket, ahol a képhez tartozó linket nem az **img tag** köré írt HTML kód biztosítja, hanem az a képben magában van kódolva. Az egyes hirdetéseket bannercsere egyezség keretében azonos gyakorisággal akarjuk megjeleníteni.

(b) A véletlenszerűen megjelenített képek egyszerű **.jpg** vagy **.png** képek[21] és ennek következtében a hirdetett link nem a képfájlban van kódolva. A megjelenítési követelmények azonosak.

[21] Ismert, hogy gyakran használnak animált .gif gépeket bannerként. Fontos megjegyezni, hogy a gif technika alkalmazása licenszet igényel, e nélkül felhasználása szerzői és szomszédos jogokat sért.

a feladat elméleti háttere

Az (a) esetben a képfájl hordozza az összes információt, azaz csak ezek véletlenszerű HTML oldalba illesztéséről kell gondoskodni, a többi megy magától. A webhely üzemeltetése során célszerű lenne, ha csak fel kellene tölteni a képeket egy bannerek elnevezésű mappába, ahonnan szkriptünk véletlenszerűen kivesz egy-egy képet és azt megjeleníti a HTML oldalba ágyazva. A problémát tehát leszűkítettük adott mappa tartalmának kiolvasására, azokból véletlenszerű kiválasztásra. A PHP mint szerver oldali szkriptnyelv képes fájlkezelésre (a JavaScript a kliens oldalon nem!) és a könyvtárak kezelésének mechanizmusa megegyezik a fájlkezeléssel (ne felejtsük el a Linux/Unix felfogást, a könyvtár is fájl, csak különleges tulajdonságai vannak!), azaz először megnyitjuk, a tartalmát kiolvassuk, véletlenszerűen kiválasztunk egy képfájlt, majd megjelenítjük.

A (b) esetben két, egymáshoz kapcsolódó információra van szükségünk, a képfájl nevére és elérési útjára, valamint a hirdetendő linkre. Ezeket tárolhatjuk adatbázisban vagy fájlban, sőt magában a kódban is elhelyezhetünk egy asszociatív tömböt, ahol a kulcs lehet a fájlnév, az érték a link URL-je. Ezekből kell az összetartozó információpárt véletlenszerűen kiválasztani és megjeleníteni. Az egyértelmű, hogy új képek és a hozzájuk tartozó link távoli feltöltése csak adatbázison át történhet, a másik két esetben akár az információt hordozó fájlt, akár a kódot manuálisan szerkeszteni kell.

tervezés

A PHP nyelvben egy könyvtár megfelelő módon történő és sikeres megnyitása esetében (`opendir()`) egy azonosítót kapunk vissza, amelynek segítségével kiolvashatjuk soronként a mappát (fájlonként) majd bezárjuk. Minden mappa tartalmaz egy "." és egy ".." fájlt is, ezekre persze vizsgálni kell a kiolvasást és ki kell szűrni. A kiolvasott fájlokat egymás után egy erre a célra létrehozott tömbbe (`$bannerek`) nyomjuk bele (`array_push()` függvény) majd az `array_rand()` segítségével véletlenszerűen indexet választunk ki és kiolvassuk/megjelenítjük az index alapján a képfájlokat tartalmazó tömb megfelelő elemét.

A (b) esetben kicsit bonyolultabb a helyzet. Adatbázis alapú megoldás esetén ki kell olvasni az összes aktuális rekordot, meg kell számolni (az SQL `count()` függvénye) majd az így megismert határok között véletlenszámot generálunk és annak alapján a recordset-ből kiemeljük az így kiválasztott sort. Ezt feldolgozva kapjuk meg a megjele-

nítendő képfájl URL-jét és a linket. Ha nem áll rendelkezésünkre adatbázis, akkor vagy fájlban tároljuk a kép URL - hozzátartozó link párokat egy-egy sorban, vagy magában a kódban asszociatív tömb formájában. Fájlból kiindulva a fájlt megnyitjuk, a kapott fájlazonosító alapján soronként kiolvassuk és az elválasztó (célszerűen tabulátor) mentén szétbontjuk, majd az információkat két párhuzamos tömbbe belenyomjuk és visszavezetjük a megoldást a fentebb leírtra. Kódban történő tárolás esetében a legegyszerűbb, ha a kép és linkinformáció egy asszociatív tömbben található és ebből történik a véletlenszerű kiválasztás.

A háromféle megoldást (elvileg csak kettő!) fogjuk össze egy osztály keretében. Utána már csak ezt kell példányosítani és a megfelelő metódus meghívásával tetszés szerinti fájlban megvalósítható a bannercserélő működtetése.

megvalósítás (kód)

index.html fájl kódja:

```
<?php
include_once("banner.inc");

?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
<title></title>
</head>

<body>
<div align="center">
  <center>
  <table border="0" cellpadding="6" cellspacing="0" width="700">
    <tr>
      <td>Asszociativ tömb tartalmazza a képfájlok neveit és elérési
útját (kulcs)
     valamint a hozzájuk tartozó linket (érték)</td>
      <td>
    <?
      $tomb = array("images/kek.gif" => "http://www.medzi.hu/php",
"images/okker.gif" => "http://www.gdf.hu",
     "images/piros.gif" => "http://www.index.hu", "images/zold.gif" =>
"http://www.greenpeace.org");
    $bn = new Banner();
    $hivatkozas = $bn->standard($tomb);
    echo $hivatkozas;
    ?>
    </td>
    </tr>
    <tr>
      <td>Egyszerû szöveges fájl tartalmazza a képneveket elérési út-
tal együtt
```

```
    valamint a hozzájuk tartozó linket</td>
      <td>
    <?
      $bn = new Banner();
    $hivatkozas = $bn->standard_file("bannerek.txt");
    echo $hivatkozas;
    ?>
    </td>
    </tr>
    <tr>
      <td>A kép hordozza a link információt, csak kiolvassuk a
megfelelõ könyvtár tartalmát</td>
      <td>
    <?
      $bn = new Banner();
    $kepurl = $bn->flash("images");
    echo ("<img border=\"0\" src=".$kepurl." >");
    ?>
    </td>
    </tr>
  </table>
  </center>
</div>
</body>
</html>
```

banner.inc fájl kódja:

```
<?php

class Banner
{
  //visszaad a konyvtarban talalhato fajlok kozul egyet, random
kivalasztva
  function flash($konyvtar)
  {
    $bannerek = array();
    $dp = opendir($konyvtar) or die("nem sikerült a könyvtár megnyitá-
sa!");
    while ($file = readdir($dp))
    {
         if ($file != "." && $file != "..")
          array_push($bannerek, $file);
    }
    closedir($dp);
    srand ((float) microtime() * 10000000);
    $index = array_rand ($bannerek);
    return $konyvtar."/".$bannerek[$index];
  }

  //visszaad a tomb alapjan egy random kivalasztott kepet es a
hozzatartozo linket
  function standard($tomb)
  {
    srand ((float) microtime() * 10000000);
    $index = array_rand($tomb);
    return "<a href=".$tomb[$index]."><img src=\"".$index."\"
border=\"0\"></a>";
  }
```

```
  //visszaad egy textfajl alapjan egy random kivalasztott kepet es a
hozzatartozo linket
  function standard_file($filenev)
  {
    $kepek = array();
    $linkek = array();
    $fp = fopen($filenev, "r") or die("nem sikerült a fájl megnyitá-
sa!");
    while (!feof($fp))
    {
      $puffer = fgets($fp, 128);
      $sor = explode("\t", $puffer);
      array_push($kepek, $sor[0]);
      array_push($linkek, $sor[1]);
    }
    fclose($fp);

    srand ((float) microtime() * 10000000);
    $index = array_rand($kepek);
    return "<a href=".$linkek[$index]."><img
src=\"".$kepek[$index]."\" border=\"0\"></a>";
  }
};

?>
```

bannerek.txt fájl tartalma

```
images/piros.gif  http://www.index.hu
images/kek.gif    http://www.medzi.hu/php
images/zold.gif   http://www.greenpeace.org
images/okker.gif  http://www.gdf.hu
```

továbbfejlesztési lehetőségek

Csaljunk egy kicsit! Bár a bannercserélő megvalósítása során alapvetően a véletlenszerű kiválasztásra törekedtünk – egyenlő esélyt adva mindegyik megjelenésnek – de előfordulhat olyan igény, biztosítsunk egy idő – pontosabban letöltés – arányos megjelenítést. azaz valósítsuk meg, hogy adott bannerre beállítható legyen a kódban a százalékos előfordulás értéke!

DBM adatbázis kezelés - utolsó látogatás időpontja

specifikáció

Webszerverünkön nem áll rendelkezésre SQL adatbázis kezelő rendszer. Már a PHP futtatási lehetőséget is csak nagy nehezen tudtuk engedélyeztetni, de adatbázisról szó sem lehet! Mi mégis szeretnénk nyilvántartani, hogy az egyes látogatók esetében mikor történt a legutóbbi belépés - és ezt közölni is akarjuk velük az üdvözlésben.

Valahogy így:

Szia xy! Ez az első látogatásod nálunk!

Illetve visszatérő látogatóknál:

Szia xy! Legutolsó látogatásod időpontja 2004-04-15 22:10:12 volt.

a feladat elméleti háttere

Tipikus adatbázisos feladat, de ha nincs adatbázis? Van viszont a PHP nyelvnek olyan függvénycsomagja, amely alkalmas egyszerű kulcs-érték párokon alapuló adatbázisok kezelésére (ilyen lehet a Berkeley DB, a Gdbm, egyes rendszerkönyvtárak). Az adatbázis ezekben az esetekben egyetlen fájl amit a PHP megfelelően kezelni képes, de az egyszerű fájlkezelésen túl nyitáskor egy .lck (lock) fájl létrehozása segítségével saját zárolást is csinál, a felülírások elkerülésére.

A login alapján tároljuk a belépéseket, mindenki esetében az utolsót (most nem ellenőrizzük a login egyediségét, majd a továbbfejlesztésben). Ha nincs ilyen azonosítóval rendelkező felhasználó, akkor új rekordot készítünk és arról is gondoskodni kell, hogy az első használat esetén létrehozzuk az adatokat tartalmazó fájlt is. Ehhez a dbmopen() függvényben megnyitási típusként a "c" jelölést használjuk (olvasás, írás, létrehozás).

Ne felejtsük el ellenőrizni a php.ini fájl beállításait! A legtöbb esetben ez a függvénycsomag alapértelmezetten nem áll rendelkezésre, a PHPTriad esetében például a php.ini fájlban a kiterjesztések (Windows extensions) részben az extension=php_db.dll bejegyzés elől ki kell venni a kommentezésre használt ";" jelölést.

tervezés

A login (egyedi azonosító) bekérése egy űrlap segítségével történik, ha ez az azonosító létezik adatbázisunkban, akkor az űrlap elküldése után kiírjuk az utolsó belépés időpontját, ha nem, csak tájékoztató szöveg jelenik meg.

A logint kulcsként alkalmazzuk és értékként a time() függvénnyel lekérdezzük a rendszeridőt módosításkor illetve új rekord felvételekor, kiírás esetében pedig ez szolgál időbélyegzőnek a date() függvényhez, amit a specifikációnak megfelelő formázó sztringgel hívunk meg. Az alapeseteknek megfelelő kiírást is készítünk a szelekció egyes ágaihoz.

megvalósítás (kód)

adatbazis.php fájl

```
<html>
<head>
<title>DBM adatbázis kezelése</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
</head>
<body bgcolor="#FFFFFF">
<?

if ($userid != "")
{
  $dbm = dbmopen ("legutobb.db", "c") or die("nem sikerült adatbázist
nyitni!");
  if (dbmexists ($dbm, $userid))
  {
    $latogatas = dbmfetch($dbm, $userid);
    echo ("Kedves ".$userid."! <br>Az utolsó látogatásod ideje:
".date("Y-m-d H:i:s", $latogatas));
    dbmclose ($dbm);
    $dbm = dbmopen ("legutobb.db", "w");
    $time = strval(time());
    dbmreplace ($dbm, $userid, $time);
    dbmclose ($dbm);
  }
  else
  {
      echo ("Kedves ".$userid."! Üdvözöljük honlapunkon!");
    dbminsert ($dbm, $userid, time());
    dbmclose ($dbm);
  }
}
?>

<form method="POST" action="">
  <table width="50%" cellpadding="6" border="0" cellspacing="0">
    <tr>
  <td>
       Login: <input type="text" name="userid">
       <input type="submit" name="Submit" value="Elküldés">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

továbbfejlesztési lehetőségek

Hozzá lehetne szerkeszteni egy látogatási számot is, amely személyekre megmondja, ki hányszor járt már az adott lapon - ez átvezet minket a statisztikákhoz, amikor figyelni akarjuk az egyes látogatókat, kinek mi tetszik, merre barangol a lapon.

DBM adatbázis kezelés - jelszavas beléptetés

specifikáció

Webszerverünkön nem áll rendelkezésre SQL adatbázis kezelő rendszer (még mindig). Már a PHP futtatási lehetőséget is csak nagy nehezen tudtuk engedélyeztetni, de adatbázisról szó sem lehet! Mi mégis létre akarunk hozni belső használatra egy fórumot, ahol jelszavas belépés van és csak a társaságunk tagjai üzenhetnek egymásnak a fórum hasábjain. Tárolnunk kell a tagok loginját (azonosítóját) - ez a rendszeren feltétlenül egyedi legyen -, jelszavát, email címét. Legyen lehetőség a jelszó közvetlen megváltoztatására a tagok részéről. A fórumot most nem kell megvalósítani és védett navigációt sem kell létrehozni.

a feladat elméleti háttere

Meg kell tehát oldani, hogy csak login/jelszó ismeretében léphessenek be a tagok, de azért ahhoz sokan vagyunk, hogy mindezt az információt fájlban tároljuk és minden változtatáshoz kézzel szerkesszük át az adatokat tartalmazó fájlt. Tipikus adatbázisos feladat, de ha nincs adatbázis? Van viszont a PHP nyelvnek olyan függvénycsomagja, amely alkalmas egyszerű kulcs-érték párokon alapuló adatbázisok (ilyen lehet a Berkeley DB, a Gdbm, egyes rendszerkönyvtárak) kezelésére. Kulcsként adódik a login, hiszen előírás, hogy egyedi legyen a rendszerünkben, az értékre ugyan több elem is van, de nem olyan bonyolult két vagy több elem összefűzésével egy tárolható sztringet előállítani. Itt most tömbökbe szervezzük a kulcshoz tartozó egyéb adatokat, ezeket kell majd az adatbázisba felvinni.

Kihasználom a lehetőséget az egymásra épülő stíluslapok (Cascading Style Sheets, CSS) rövid gyakorlati bemutatására is. Az egyes fájlokon belül a weblapba épített stíluslapok adják meg a beviteli mező külalakját, a színeket, a kereteket, úgy hogy átdefiniáljuk a táblacellákat (td) és új osztályt hozunk létre (.table), amit majd a táblázat fejlécében helyezünk el, hogy a táblázat egészére vonatkozzon.

Ne felejtsük el ellenőrizni a php.ini fájl beállításait! A legtöbb esetben ez a függvénycsomag alapértelmezetten nem áll rendelkezésre, a PHPTriad esetében például a php.ini fájlban a kiterjesztések (Windows extensions) részben az extension=php_db.dll bejegyzés elől ki kell venni a kommentezésre használt ";" jelölést.

tervezés

Három fájlt hozunk létre, az index.php feladata a jelszavas belépéshez az űrlap megjelenítése és a bevitt adatok ellenőrzése, majd azok elfogadása esetén az átirányítás a belso.php weblapra. Ezen a lapon egy újabb űrlap segítségével a felhasználó megváltoztathatja a jelszavát. Az azonosítót nem is kérjük be újból, hiszen csak a megfelelő login/jelszó ismeretében kerülhet ide[22]. A harmadik fájl a regform.php, nevének megfelelően ezen a lapon egy egyszerű regisztráció hajtható végre, ahol új egyedi azonosítót (login), jelszót, e-mail címet vihetünk fel. Ezeket a login egyediségén kívül külön nem ellenőrizzük. A regisztráció után a felhasználót átirányítjuk az index.php lapra, ahol be tud lépni a rendszerve, immár regisztrált felhasználóként.

Az index.php belépési űrlapot akkor dolgozzuk fel, ha mindkét mezőt kitöltötték. Először megvizsgáljuk a login meglétét az adatbázisban, ha van ilyen, lekérjük ennek alapján a hozzátartozó értéket (ne feledjük, egyszerű kulcs-érték párról van szó!) kiszedjük belőle a jelszó sztringet és összehasonlítjuk a bevittel. Igaz eredmény esetén JavaScript segítségével megoldjuk az átirányítást, az URL-ben átadva a logint is.

Ha nincs ilyen login, figyelmeztetést írunk a $loginhiba változóba, amit majd meg is jelenítünk újratöltéskor, ha nem azonos a bevitt és a tárolt jelszó, ugyanezt csináljuk a $jelszohiba változóval.

A regform.php regisztrációs űrlap esetében hasonlóképpen járunk el, mindhárom mező kitöltése után kezdünk el foglalkozni az adatokkal és először ellenőrizzük, van-e már ilyen azonosító az adatbázisban. Ha igen, ezt a melléírásos technikával jelezzük is a beviteli mező közelében, míg a többi mező értékét visszaírjuk, ne kelljen azokat újra kitölteni a felhasználónak. Siker esetén egy JavaScript üzenet után átirányítjuk a felhasználót a regisztrált belépésekhez.

A jelszómódosítást hordozó belso.php fájl hasonlóképpen működik, mivel a logint már tudjuk az URL-ből - feltételezésünk szerint ide csak regisztráltan léphet be a fel-

[22] Feltételezésünk szerint! Egy éles rendszer esetében ezt mindig le kell ellenőrizni a szerveroldalon és időkorlátot kell felállítani a belépéshez, hiszen az otthagyott, a kliens gépen lementett lap alapján különben más is be tudna lépni, illetve egyszerűbb lenne a hacker élete, ha az azonosítót már nem kellene megkeresni.

használó - csak az új jelszót gépeltetjük be vele, a szokásos ismétléssel és átírjuk az adatbázisban annak értékét a kulcshoz tartozó szerializált sztringben.

A kódban csak a regform.php-t mutatjuk be teljes egészében, a másik kettőből csak a PHP kód részletet másoljuk be ide.

megvalósítás (kód)

regform.php fájl kódja

```
<?php
//csak akkor foglalkozunk vele, ha ki vannak toltve a mezok
$hiba = "";
if ($login != "" && $pwd != "" && $email != "")
{
  //megvizsgaljuk, egyedi-e a login es ha nincs adatbazis, letrehozzuk
  $dbm = dbmopen("tagok.db", "c") or die("nem sikerült adatbázist
nyitni!");
  if (!dbmexists($dbm, $login)) // ha nem foglalt
  {
    $tomb = array($pwd, $email);
    $ertek = serialize($tomb);
    dbminsert($dbm, $login, $ertek);
    dbmclose($dbm);
    //regisztralt belepeshez atiranyitas JavaScript segitsegevel
    echo ("<script>");
    echo ("alert('A regisztráció sikeres');");
    echo ("location.replace('index.php');");
    echo("</script>");
    exit();
  }
  else
    $hiba = "ilyen login már van!";
}
?>
<html>
<head>
<title>DBM adatbázis - jelszavas belépés</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
<style type="text/css">
<!--
td {
font-family: Arial, Helvetica, sans-serif;
font-size: 10pt
}
.table {
font-weight: bold;
color: #330000;
background-color: #FFFFCC; border: 1px #CC9933 solid
}
-->
</style>
</head>

<body bgcolor="#FFFFFF">
<table width="100%" border="0" cellspacing="0" cellpadding="6"
align="center">
```

```
<tr>
<td colspan="3">
<p> </p>
<p> </p>
</td>
</tr>
<tr>
<td  width="25%"> </td>
<td width="50%" >
<form name="regisztracio" method="post" action="">
<div align="center">
<table width="350"  cellspacing="0" cellpadding="3"  class="table">
<tr>
    <td colspan="2"  >
        <div align="center">
            <p>Regisztrálás</p>
        </div>
    </td>
</tr>
<tr>
    <td width="100">login: </td>
    <td>
        <input type="text" name="login" size="15" maxlength="15"
value="<? echo $login; ?>">
<? if ($hiba != "") echo ("<font color=#FF0000 size=-
2>".$hiba."</font>");?>
    </td>
</tr>
<tr>
    <td width="100">jelszó:</td>
    <td>
        <input type="password" name="pwd" size="15" maxlength="15"
value="<? echo $pwd; ?>">
    </td>
</tr>
<tr>
    <td width="100">email: </td>
    <td>
        <input type="text" name="email" size="30" maxlength="30"
value="<? echo $email; ?>">
    </td>
</tr>
<tr>
    <td colspan="2"  >
        <div align="center">
            <input type="submit" name="Submit" value="  Belépés  ">
        </div>
    </td>
</tr>
</table>
</div>
</form>
</td>
<td  width="25%">  </td>
</tr>
</table>
</body>
</html>
```

index.php fájl kódrészlete

```
<?php
//ha kitoltottek a login es jelszo mezoket
$loginhiba = "";
$jelszohiba = "";
if ($login != "" && $pwd != "")
{
  //eloszor megnezzuk, van-e ilyen login
  $dbm = dbmopen ("tagok.db", "w") or die("nem sikerült adatbázist
nyitni!");
  if (dbmexists($dbm, $login))
  {
    //kiszedjuk a tarolt jelszot es osszevetjuk
    $tomb = unserialize(dbmfetch($dbm, $login));
    dbmclose($dbm);
    if ($tomb[0] == $pwd)
    {
      echo
("<script>location.replace('belso.php?login=".$login."');</script>");
      exit();
    }
    else
    {
      $jelszohiba = "nem jó a jelszó!";
    }
  }
  else
  {
    $loginhiba = "nincs ilyen login!";
    dbmclose($dbm);
  }
}
?>
```

belso.php fájl kódrészlete

```
<?php
//ha kaptunk egy jelszomodositasi kerelmet: (mar tudjuk a logint is)
//ha a jelszavak rendben vannak, kitoltve es azonosak
if ($ujpwd1 != "" && $ujpwd1 == $ujpwd2)
{
  //fel kell tolteni a szerverre egy tagok.db fajlt legelso alkalommal
  $dbm = dbmopen ("tagok.db", "w") or die("nem sikerült adatbázist
nyitni!");
  $tomb = unserialize(dbmfetch($dbm, $login));
  $tomb[0] = $ujpwd;
  $ertek = serialize($tomb);
  dbmreplace($dbm, $login, $ertek);
  dbmclose($dbm);
}
?>
```

továbbfejlesztési lehetőségek

Szerkesszük össze a két DBM adatbázis példát egyetlen alkalmazásba és oldjuk meg azt is, hogy legördülő menüből kiválasztva a tagok loginját, üzenetet küldhessünk adott személynek, aki ezt belépésekor rögtön láthatja is a belso.php fájl erre a célra fenntartott paneljában.

Egyszerű fórum, üzenőfal

specifikáció

Hozzunk létre egyszerű fórumot, ahol a hozzászólásokat időrendben tüntetjük fel és lapozgathatunk a régebbi-újabb közlemények között, egyszerre adott számú tételt jelenítve meg. Mindenki hozzászólhat korlátozás nélkül, a hozzászóló nevét, becenevét (nick) és e-mail címét, valamint hozzászólását és annak rövid összefoglalását tüntetjük fel. Gondoskodjunk arról, hogy a szöveggel ne lehessen "elrontani" a fórumot!

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Látogató számlálás - egyszerű

specifikáció

Számolni akarjuk látogatóinkat és ezt az adatot tárolni is akarjuk. Ki akarjuk védeni viszont a manipulálási lehetőségeket, azaz ne lehessen "pörgetéssel" (frissítés gombbal újratöltve az oldalt) látogatószámot növelni. Valamilyen szintű látogató azonosítást (például IP cím) is el akarunk végezni.

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Látogató számlálás - statisztika

specifikáció

Az előző feladat továbbfejlesztése, ezúttal a számolás mellett az egyes oldalak látogatottságát is fel akarjuk mérni. Külön kérésre kapjunk táblázatos összefoglalást az oldalak látogatottságáról, a megfelelő szűrések mellett. Ki akarjuk védeni viszont a manipulálási lehetőségeket, azaz ne lehessen "pörgetéssel" (frissítés gombbal újratöltve az oldalt) látogatószámot növelni. Valamilyen szintű látogató azonosítást (például IP cím) is el akarunk végezni.

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Képek kirakása, képgaléria

specifikáció

Egy könyvtárba töltjük fel képeinket és ezeket akarjuk megmutatni, egyszerre mindig csak adott számú darabot. Bemutatásukhoz először körömnyi méretű (thumbnail), adott és egységes szélességű képeket rendezünk el a felületen, ezekre kattintva viszont megjelenítjük a teljes méretű képet egy felbukkanó ablakban.

a feladat elméleti háttere

Ne felejtsük el ellenőrizni a php.ini fájl beállításait! A grafikai függvénycsomag alapértelmezetten nem áll rendelkezésre, a PHPTriad esetében például a php.ini fájlban

a kiterjesztések (Windows extensions) részben az extension=php_gd.dll bejegyzés elől ki kell venni a kommentezésre használt ";" jelölést.

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Dinamikus képek - felirat készítés

specifikáció

Fotógyűjteményünket szeretnénk megjeleníteni a weben, de nem akarjuk, hogy bárki letölthesse és felhasználhassa. Ennek egyik lehetséges megoldása, hogy mindegyiken elhelyezünk egy szerzői jogunkat deklaráló feliratot, de ezt nem akarjuk képszerkesztéssel megoldani, nem akarjuk "elrontani" a képeinket.

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Dinamikus képek - egyszerű idomok kirajzolása

specifikáció

Hozzuk létre egy piros háttérszínű képet, amelyben két kék hasábot helyezünk el. A hasábok szélessége adott - mindig 20 pixel méretűek - de magasságuk változó, az egyik a mindenkori képmagasság 50, a másik a 75 százaléka. A kép mérete legyen vál-

toztatható űrlapon keresztül történő adatbevitellel (szélesség, magasság), de a hasábok mindig az kép alapjához igazodjanak. Készítsünk egy feliratot is a képre, a "Hasábok" szót írjuk ki a kép felső részére. Nem kell külön ellenőrzést végezni az egyes elemek elhelyezéséhez, tételezzük fel, hogy a képméret változtatásánál nem visznek be irreális értékeket.

a feladat elméleti háttere

A feladat lényege a PHP képkezelési elvének megértése. Az ezt szolgáló függvényeket a GD nevű csomag tartalmazza, amelyet be kell állítani a függvények elérése érdekében. A kép létrehozása a következő lépésekből áll:

Alapelv, hogy HTML oldalba illesztett kép esetén létre kell hozni az oldalon a kép HTML hivatkozását, azaz bele kell írni/szerkeszteni a kódba az `<img src=kep.php width="ddd" height="ddd">` képet meghatározó tag-et, amelyben a `kep.php` fájlnév a létrehozandó kép leírását tartalmazó PHP fájl neve és a szélesség, magasság attribútumok opcionálisak, illetve általában a képet leíró fájlban adjuk meg (ez különösen akkor igaz, ha dinamikusan változik a képméret adott határok között).

A képet leíró fájl valóban egy fájl hoz létre, tehát először egy header utasítást küldünk ki, meghatározva a kép típusát (jpg, png, stb.). Ezt követi a kép alapjának[23] létrehozása az `imagecreate()` függvénnyel, amelynek paraméterei a kép szélesség, magasság adatai és visszatérési értéke a szokásos logikai értéken túl (sikeres/sikertelen végrehajtás) a kép azonosítója (például `$img`). Utána színeket határozhatunk meg, alapértelmezésben az első meghatározott szín a háttér színe. Ezeket követik a kiíró utasítások, amelyek a GD csomag verziójától és a környezettől függően eltérhetnek, azaz ezt ajánlatos kipróbálni a megcélzott futtató rendszeren is. Ha szöveget is ki akarunk írni, akkor először meg kell határozni a használandó fontkészletet (meg kell adni annak elérési útját, ha az alapértelmezettől eltérő fontokat akarunk használni) majd pozicionált kiírást készítünk. Az utasítás sorozatot a képnek megfelelő `imagepng($img)` vagy `imagejpg($img)` függvény zárja le, amely létrehozza a fájlban az előbbi utasításokkal lert képet. Természetesen a kép akkor is létrejön, ha nem HTML-be ágyazva készítjük, akkor egy közönséges képfájl kapunk a leírásnak megfelelően (ezt a viselkedést ki lehet például használni dinamikusan megjelenő feliratok készítésére).

[23] grafikai rendszerekben ez a "canvas", azaz vászon, az olajfestmények ugyanis vászonra készültek ☺

Fontos! Ne felejtsük el ellenőrizni a php.ini fájl beállításait! A grafikai függvénycsomag alapértelmezetten nem áll rendelkezésre, a PHPTriad esetében például a php.ini fájlban a kiterjesztések (Windows extensions) részben az extension=php_gd.dll bejegyzés elől ki kell venni a kommentezésre használt ";" jelölést.

tervezés

Programtervezésről itt nem nagyon beszélhetünk, inkább a grafikai tervezés a fontos. Létre kell hozni a képet leíró `kep.php` es az ezt megjelenítő `kephtml.php` fájlokat. Ez utóbbi tartalmaz egy beviteli űrlapot is, a méret változtatásához. Ha nincs érték megadva, alapértelmezett méreteket határozunk meg. Felhívom a figyelmet egy JavaScript trükkre - a form objektum nevet kapott (bevitel) és így hivatkozni tudunk annak `submit()` metódusára (`javascript:document.bevitel.submit()`) így egy linkre kattintva, automatikusan az onClick eseménykezelőhöz kapcsoltan is megvalósítható a form elküldése.

megvalósítás (kód)

kephtml.php fájl kódja

```
<html>
<head>
<title>Kepek megjelenitese</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
2">
</head>
<body bgcolor="#FFFFFF">
<?
if ($sz != "" && $m != "")
    echo ("<p><img src=\"kep.php?sz=".$sz."&m=".$m."\"></p>");
else
    echo ("<p><img src=\"kep.php\"></p>");
?>
<p> </p>
<form name="bevitel" method="post" action="">
  <table width="50%" border="0" cellspacing="0" cellpadding="6"
align="center">
    <tr>
      <td>képszélesség</td>
      <td>
        <input type="text" name="sz" size="3" maxlength="3" value= <?
echo $sz; ?> >
      </td>
    </tr>
    <tr>
      <td>képmagasság</td>
      <td>
        <input type="text" name="m" size="3" maxlength="3" value= <?
echo $m; ?> >
      </td>
    </tr>
```

```
    <tr>
      <td><a href="javascript:document.bevitel.submit()"><font
size="2" face="Arial" color="#FF0000" ><b>Adatbevitel</b></font></a>
      </td>
      <td>
        <input type="submit" name="Submit" value="  Adatbevitel  ">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

kep.php fájl kódja

```
<?php
//keptipus meghatarozasa
header ("Content-type: image/png");
//alapertelmezett ertekek kiosztasa
if (!isset($sz) && !isset($m))
{
      $sz = 200;
      $m = 200;
}
//kepalap letrehozasa
$img = @ImageCreate ($sz, $m)
      or die ("Nem inicializalodik az uj GD image stream");
//szinek meghatarozasa
$hatter_szin = ImageColorAllocate ($img, 255, 0, 0);
$szoveg_szin = ImageColorAllocate ($img, 0, 0, 255);
//sikidomok kirajzolasa
ImageRectangle($img, $sz/2, $m/2, $sz/2+20, $m, $szoveg_szin);
ImageFill($img, $sz/2+5, $m/2+5, $szoveg_szin);
ImageRectangle ($img, $sz/4*3, $m/4*3, $sz/4*3+20, $m, $szoveg_szin);
ImageFill($img, $sz/4*3+5, $m/4*3+5, $szoveg_szin);
//fontkeszlet meghatarozasa
$font = "c:\windows\fonts\arial.ttf";
//felirat elkeszitese
ImageTTFtext($img, 12, 0, 10, 20, $szoveg_szin, $font, "Hasábok");
//a kep osszeallitasa az utasitasoknak megfeleloen
ImagePng ($img);
?>
```

továbbfejlesztési lehetőségek

(a) Készítsünk különböző kitöltött és körvonalas síkidom kirajzolásokat.

(b) Hozzunk létre 5-10 véletlenszerűen megjelenő téglalapot a képen, amelyek színei is változnak minden egyes kirajzoláskor.

Dinamikus képek - oszlopdiagram kirajzolása

specifikáció

Két feladatot oldunk meg ebben a példában, az első esetében a grafikon oszlopainak adatait egy tömb biztosítja és elrendezésüket a kép leírásában meghatározzuk, a

második esetben viszont majdnem mindent a kép méreteihez viszonyítva rajzolunk ki és az adatokat fájlból vesszük. Ebben az esetben az az összehasonlítandó elemek száma is változhat bizonyos logikus határok között, de a kép kirakásakor egyenletesen elosztjuk az oszlopokat a rendelkezésre álló képfelületen.

a feladat elméleti háttere

Az elmélet lényegében megegyezik az ezt megelőző feladatban leírtakkal, itt csak a legfontosabb részét ismételjük meg, hogy a feladat önmagában is érthető legyen.

A képet leíró fájlban először egy header utasítást küldünk ki, meghatározva a kép típusát (jpg, png, stb.). Ezt követi a kép alapjának létrehozása az `imagecreate()` függvénnyel, amelynek paraméterei a kép szélesség, magasság adatai és visszatérési értéke a szokásos logikai értéken túl (sikeres/sikertelen végrehajtás) a kép azonosítója (például `$img`). Utána színeket határozhatunk meg, alapértelmezésben az első meghatározott szín a háttér színe. Ha szöveget is ki akarunk írni, akkor először meg kell határozni a használandó fontkészletet majd pozicionált kiírást készítünk. Az utasítás sorozatot a képnek megfelelő `imagepng($img)` vagy `imagejpg($img)` függvény zárja le, amely létrehozza a fájlban az előbbi utasításokkal leírt képet. Dinamikusan változó elemek esetében - mint például ezek az oszlopdiagrammok - gondoskodni kell az elemek algoritmus segítségével történő elrendezéséről is, illetve fájlból történő adatátvétel esetén a fájl kiolvasásáról. Látható, hogy a képet leíró fájl egy teljes PHP program, amely minden szükséges adatot és adatforrást önmagában kezelni képes, azonban paraméterként is átvehet értékeket (például egy adatokat tartalmazó tömböt).

Ne felejtsük el ellenőrizni a php.ini fájl beállításait! A grafikai függvénycsomag alapértelmezetten nem áll rendelkezésre, a PHPTriad esetében például a php.ini fájlban a kiterjesztések (Windows extensions) részben az extension=php_gd.dll bejegyzés elől ki kell venni a kommentezésre használt ";" jelölést.

tervezés

Egy grafikonok.php fájl biztosítja a HTML környezetet a megjelenítéshez és az adatok.txt a kirajzolásra váró adatokat. A grafikon_1.php és a grafikon_2.php biztosítja a kétféle grafikon leírását.

megvalósítás (kód)

grafikon_1.php fájl kódja

```php
<?php
header ("Content-type: image/png");

$tomb = array(100, 120, 190, 250, 40);

$img = @ImageCreate (400, 300) or die ("Nem inicializalodik az uj GD
image stream");
$hatter_szin = ImageColorAllocate ($img, 255, 0, 0);
$szoveg_szin = ImageColorAllocate ($img, 0, 0, 255);
$x = 50;
for ($i=0; $i<count($tomb); $i++)
{
      ImageRectangle ($img, $x, (300-$tomb[$i]), ($x+20), 300,
$szoveg_szin);
      ImageFill($img, ($x+2), 290, $szoveg_szin);
      $x += 70;
}
$font = "c:\windows\fonts\arial.ttf";
ImageTTFtext($img, 12, 0, 10, 20, $szoveg_szin, $font, "Grafikon PHP-
vel megjelenitve");
//ImageColorTransparent($img, $hatter_szin);
ImagePng ($img);
?> <?php
header ("Content-type: image/png");

$tomb = array(100, 120, 190, 250, 40);

$img = @ImageCreate (400, 300) or die ("Nem inicializalodik az uj GD
image stream");
$hatter_szin = ImageColorAllocate ($img, 255, 0, 0);
$szoveg_szin = ImageColorAllocate ($img, 0, 0, 255);
$x = 50;
for ($i=0; $i<count($tomb); $i++)
{
      ImageRectangle ($img, $x, (300-$tomb[$i]), ($x+20), 300,
$szoveg_szin);
      ImageFill($img, ($x+2), 290, $szoveg_szin);
      $x += 70;
}
$font = "c:\windows\fonts\arial.ttf";
ImageTTFtext($img, 12, 0, 10, 20, $szoveg_szin, $font, "Grafikon PHP-
vel megjelenitve");
//ImageColorTransparent($img, $hatter_szin);
ImagePng ($img);
?>
```

grafikon_2.php fájl kódja

```php
<?php
header ("Content-type: image/png");
//header ("Content-type: image/jpeg");

$kep_szeles = 400;
$kep_magas = 300;
$szeles = 15;
$tav = 50;
$font = "c:\windows\fonts\arial.ttf";
```

```
$fp = fopen("adatok.txt", "r");
$sor = fgets($fp, 1024);
$tomb = explode("\t", $sor);
fclose($fp);

$tav = ($kep_szeles - count($tomb)* $szeles)/(count($tomb)+1);

$img = @ImageCreate($kep_szeles, $kep_magas) or die ("Nem
inicializalodik az uj GD image stream");
$hatter_szin = ImageColorAllocate ($img, 254, 207, 137);
$szoveg_szin = ImageColorAllocate ($img, 166, 64, 0);
$x = $tav;
for ($i=0; $i < count($tomb); $i++)
{
      ImageRectangle ($img, $x, ($kep_magas - intval($tomb[$i])),
($x+$szeles), $kep_magas, $szoveg_szin);
      ImageTTFtext($img, 11, 0, $x, ($kep_magas -10 -
intval($tomb[$i])), $szoveg_szin, $font, "$tomb[$i]");
      ImageFill($img, ($x+2), ($kep_magas-2), $szoveg_szin);
      $x += $tav + $szeles;
}
ImageTTFtext($img, 12, 0, 10, 20, $szoveg_szin, $font, "Grafikon PHP-
vel megjelenitve, \n\radatok fajlbol es auto rendezes");
ImagePng($img);
//ImageJpeg ($img, '', 100);
?>
```

továbbfejlesztési lehetőségek

Dinamikus képek - vonaldiagram kirajzolása

specifikáció

Ábrázoljunk egy időben változó tulajdonságot - például a napi középhőmérsékleteket egy héten keresztül - vonalas diagrammon. Az egyszerűség kedvéért az adatokat egy tömbből vesszük, de más adatforrások is szóba jöhetnek a továbbfejlesztés során.

a feladat elméleti háttere

Ne felejtsük el ellenőrizni a php.ini fájl beállításait! A grafikai függvénycsomag alapértelmezetten nem áll rendelkezésre, a PHPTriad esetében például a php.ini fájlban a kiterjesztések (Windows extensions) részben az extension=php_gd.dll bejegyzés elől ki kell venni a kommentezésre használt ";" jelölést.

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Szavazógép kódolása

specifikáció

Készítsünk szavazógépet, amely tetszés szerint elhelyezhető az oldalon, függőleges elrendezésben a kérdések számának megfelelő számú rádiógombból és feliratból áll és a szavazógép beillesztéséhez csupán a kérdéseket kelljen megadni, akár fájlban, akár tömb formájában, onnét minden automatikus legyen. Egy **Szavazok** és egy a **Szavazás állása** feliratú gomb legyen, mindegyik a feliratának megfelelően működjön. A szavazatoknál nem kell a felhasználókat azonosítani és a legegyszerűbb változatnál a "pörgetést" sem védjük ki (az oldal frissítéssel történő újratöltése).

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Felmérés készítése

specifikáció

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Tesztprogram kódolása

specifikáció

a feladat elméleti háttere

tervezés

megvalósítás (kód)

továbbfejlesztési lehetőségek

Ellenőrző kérdések

Irodalomjegyzék

Perl

- Programming Perl, Larry Wall és Randal L. Schwartz, O'Reilly & Associates, Inc. Sebastopol, Kalifornia ISBN 0-93715-64-1
- Learning Perl, Randal L. Schwartz, O'Reilly & Associates, Inc., Sebastopol, Kalifornia
- Perl 1-2. Michael McMillan, Panem Könyvkiadó, Budapest, 1998

- www.perl.com

PHP

-
-
-
- http://www.developerfusion.com/php/ - PHP oktatóanyagok, cikkek, gyakorlati példák
- http://pear.php.net/ - a PEAR osztálykönyvtár és -csomagok lelőhelye
- http://www.php.net/ a PHP "hivatalos" webhelye
- http://www.zend.com/ a PHP interpreter kifejlesztőinek weboldala, kereskedelmi megoldások is!
- http://pointernet.mirrors.phpclasses.org/browse.html/browse.html - PHP osztályok lelőhelye, leírások

PHP fejlesztőeszközök

- Abriasoft Merlin Server www.abriasoft.com
- FoxServer www.foxserv.net
- FoxSev http://foxserv.linuxmax.net/portal.php
  apache+php+mysql+perl+python+myadmin 37mb xp-n is fut
- merlin server http://www.abriasoft.com/ 28mb
- phptriad http://sourceforge.net/projects/phptriad/ 11mb xp-n is megy
- EasyPHP

Hasznos linkek

(mind a szerver oldali, mind a kliens oldali programozáshoz)

Az egyes weboldalak nagyon gyakran referencia jellegűek, tehát a kínált példák, oktatóanyagok, forráskódok több különböző helyen fordulnak elő és persze kereszthivatkozások is előfordulnak. A hivatkozások általában kellő információt közölnek az eligazodáshoz (mit találhatunk a hivatkozott webhelyen), amennyiben erre nem történik utalás a link szövegében, ott többnyire gyűjteményes weboldalakra hivatkoznak. Szándékosan nincs megadva fontossági sorrend, ezt mindenki tegye meg saját ízlése és egyéni igényei alapján. A linkek mind ellenőrzött és élő webhelyekre mutattak a kézirat létrejötte-

kor (2003. október - 2004. március) de természetesen ez nem garancia a későbbi feltalálhatóságra. Az összeállítás messze nem teljes és természetesen tükrözi a szerző személyes tapasztalatait, irányultságát. Ugyanakkor minden itt nem szereplő, nem hivatkozott, de az olvasó által jól használhatónak tartott oldalt szívesen beillesztek a "gyűjteménybe", kérem, hogy ezek linkjeit juttassák el hozzám a jegyzet folyamatos bővítése és naprakész jellege megőrzése érdekében.

Magyar nyelvű oldalak

- http://www.ats-group.net/webmaster többféle téma a webfejlesztés témakörében
- www.prog.hu sokféle programozási téma, csoportosítva
- www.weblabor.hu cikkek, ötletek, sok PHP
- http://hu.php.net/manual/hu/ a PHP magyar nyelvű dokumentációja
- www.javascript.lap.hu - minden amit a JavaScript-ről tudni lehet…

Angol nyelvű oldalak

- http://www.evilwalrus.com/codearchive.php
- www.hotscripts.com
- http://php.resourceindex.com/Complete_Scripts/
- px.sklar.com/ elsősorban PHP
- http://www.phporacleadmin.org/
- http://www.phpbuilder.com
- http://www.zend.com/zend/tut/ PHP oktató cikkek (tutorial)
- http://www.onlamp.com/php/ vegyes anyagok, PHP cikkek
- http://phplens.com/lens/ néhány érdekes alkalmazás
- http://phpnuke.org/ automatikus portálrendszer PHP-ben
- http://www.mysql.com/ minden, amit a MySQL adatbázisról tudni kell/lehet
- http://www.weberdev.com/ JavaScript, PHP, MySQL cikkek, forráskódok