Tartalomjegyzék

Bevezetés

Manapság Magyarországon az internetes folyószámla szolgáltatásokkal rendelkező hitelintézetek és az Internetes banki-szolgáltatásokra szerződött lakossági ügyfelek száma folyamatosan növekszik

A GKI Gazdaságkutató Rt. és a T-Mobile Távközlési Rt. felmérése alapján 2005. szeptember 30-án az internetes folyószámla-szolgáltatásokkal rendelkező hitelintézetek 664 ezer Internet-banki szolgáltatásokra szerződött lakossági ügyféllel rendelkeztek, ami 10%-kal több mint 3 hónappal, és 52%-kal magasabb, mint egy évvel korábban.

Az Internet-banki szolgáltatásokra szerződött vállalkozások száma ugyanebben az időpontban 103 ezer volt, ami 7%-os, illetve 58%-os növekedést takar a megelőző 3, illetve 12 hónap során.

Internetes szolgáltatások

A bankok várakozásai alapján az internetes bankszolgáltatások közül továbbra is az internetes átutalások aránya növekszik majd a legnagyobb mértékben mind a lakossági, mind a vállalati ügyfelek között. A második, illetve harmadik helyre a tranzakciók ellenőrzése, illetve a számlainformációk lekérdezése került. A korábbi negyedévektől eltérően, a többi online banki szolgáltatás (befektetési termékek, csoportos beszedési megbízás) tekintetében is növekedésre számítanak a válaszoló hitelintézetek.

Egy korábbi, 2001-es felmérés a bankok Interneten megvalósítható banki szolgáltatásai és a felhasználók megoszlásáról:

Internetes szolgáltatások az egyes bankoknál 1/2

Forrás: Bank&Tőzsde, Pénz a Hálón melléklet 2001/1

	IEB	OTP	Raiffeisen	Citibank	K&H
Átutalások indítása	Van	Van	Van	Van	Van
Értékpapír kereskedelem	Van	Van	Van	Nincs	Nincs
Betétműveletek	Van	Van	Nincs	Van	Van
Csoportos beszedési megbízás	Van	Nincs	Nincs	Nincs	Nincs
Elektronikus aláírás vagy belépési kód megváltoztatása	Van	Van	Van	Van	Nincs

	IEB	OTP	Raiffeisen	Citibank	K&H
Lekötött betétek lekérdezése	Van	Van	Nincs	Van	van
Sorban álló tételek követése	Van	Van	Nincs	Nincs	Nincs
Egyenleg és törzsadatok lekérdezése	Van	Van	Van	Van	Van
Számlatörténet lekérdezése	Van	Van	Van	Van	van
Szerződések kezdeményezése	Van	Van	Van	Van	Van
Szolgáltatás letiltása	Van	Van	Van	Nincs	Van

Internetes szolgáltatások az egyes bankoknál 2/2
Forrás: Bank&Tőzsde, Pénz a Hálón melléklet 2001/2

	IEB	OTP	Raiffeisen	Citibank	K&H	CIB	HWB
Felhasználó	4500	80000	700	12000	1-2 ezer	3000	200
Ebből vállalati	500	Nincs adat	Nincs	600	nincs	600	20
Számlaegyenleg lekérdezése	Van	Van	Van	Van	Van	Van	Van
Számlatörténet lekérdezése	Van	Van	Van	Van	Van	Van	Van
Átutalások indítása	Van	Van	Van	Van	Van	Van	Van
Betétműveletek	Van	Van	Nincs	Van	Van	Van	Van
Értékpapír műveletek	Van	Van	Van	Nincs	Nincs	Nincs	Nincs
Állampapírok adásvétele	Nincs	Van	Van	Nincs	Nincs	Van	Nincs
WAP-os tranzakció	Nincs	Nincs	Nincs	Nincs	Nincs	Van	Nincs
Automatikus SMS értesítés	Van	Van	Nincs	Van	Van	Van	Nincs

A témára azért esett a választásom, mert úgy vélem napjaink egyik időszerű kérdése, hogy milyen banki szolgáltatásokat intézhetünk el akár a saját szobánkban ülve akár egy éjszakai órában, és mindezt mennyi idő alatt és milyen alkalmazás keretében tehetjük meg. Valamint az sem volt utolsó szempont, hogy egy banki honlap létrehozása elég nagy feladat, hiszen ezeken a portálokon bővelkednek a különböző programozási feladatok (pl.: titkosítás, webes adatbázis kezelés, űrlapok feldolgozása, adatok átadása a weboldalak között, jelszó generálása, jelszó módosítása, kilépés után ne lehessen elérni az adatokat, bekért adatok ellenőrzése, e-mail küldés,

stb.), és ezek azok a honlapok, amelyeket a legtöbb támadás éri. A támadások viszont arra sarkallják a honlap programozóját, hogy minél jobb és hatásosabb védelmet biztosítson, mind a fogyasztók, mind a saját munkájuk és elért eredményeik érdekében.

Szakdolgozatommal egy felhasználóbarát banki szolgáltatásokra kifejlesztett, gyors és biztonságos webes alkalmazást szeretnék kifejleszteni és bemutatni.

A következő eszközöket választottam az alkalmazás fejlesztéséhez, elkészítéséhez:

HTML, PHP, és JavaScript nyelveket, valamint MySQL adatbáziskezelőt.

A PHP oldalak elkészítésénél a HTML-t gyakorlatilag csak mint formázást fogom használni, ugyanis ezen lapok teljes funkcionalitása a PHP-re épül. Amikor egy PHP-ben megírt oldalt akarunk elérni, a kiszolgáló először feldolgozza a PHP utasításokat, és már csak a kész (HTML) kimenetet küldi el a böngészőnek. Ehhez egy úgynevezett interpretert (értelmezőt) használ, amely általában egy külső modulja a webszervernek.

A **PHP** elterjedt nyílt forráskódú szerver-oldali programozási nyelv. A PHP dinamikus weboldalak készítésére használható.

A PHP nyelv lényegében nagymértékű kiegészítése a HTML-nek, ugyanis rengeteg olyan feladat végezhető el vele, amelyre az ügyféloldali szkriptek nem képesek (vagy ha igen, korlátozottan). Ilyen pl. a bejelentkezés, az adatbáziskezelés, filekezelés, kódolás, adategyeztetés, kapcsolatok létrehozása, e-mail küldése, adatfeldolgozás, dinamikus listakészítés stb. Minden olyan esetben, ahol nagyszámú ismétlődő feladatsort kell végrehajtani (Pl.: képek listázása és linkelése, listakészítés stb.), ott ez a programnyelv nagyszerű segítség.

A JavaScript egy tömör, objektum-alapú leíró nyelv, amelyet a kliens-szerver Internet alkalmazásokhoz fejlesztettek ki. A Navigator fordítja le a JavaScript utasításokat, amelyek közvetlenül bele vannak ágyazva egy HTML lapba. Egy kliens alkalmazásban a Navigátor fel tudja ismerni a HTML lapokba beágyazott JavaScript utasításokat és felelni tud felhasználói eseményekre, mint például az egérklikkelés, form bemenet és az oldal navigáció.

A JavaScriptet a HTML oldal felhasználóbarát alkalmazói felület elkészítésére fogom használni, amellyel megkönnyítem az adatok felvitelét. A JavaScript események figyelésére és ezen események lekezelésére nagyon hatásos eszköz, pl ha a felhasználó megnyitott egy lapot vagy kilépett egy lapról, vagy a bemeneti adatok hosszának figyelése, vagy az egéresemények figyelése egy gomb használata során. Ezzel az eszközzel teszem az alkalmazást színesebbé,

érdekesebbé és persze az előzőekben felsorolt események lekezelését is ezzel az eszközzel kívánom megvalósítani.

A felhasználók és számlamozgások adatait, valamint az űrlapról beérkező leveleket is a MySQL adatbázis kezelő segítségével fogom tárolni. A táblában lévő adatokat a PHP programokon keresztül fogom módosítani és végül felvinni az adatbázisba, vagy összehasonlítani az adatbázisban tárolt adatokkal, vagy állományt készítek az adatbázis adataival a PHP filekezelő utasításainak segítségével.

Tárhely szolgáltatónak a http://www.uw.hu ingyenes tárhely szolgáltatót választottam a felkínált MySQL és FTP-s szolgáltatások miatt, valamint a MySQL PHPMYADMIN szolgáltatása miatt, amellyel lényegesen kevesebb idő alatt lehet a táblák szerkezetét és tartalmát szerkeszteni, módosítani.

PHP (PHP Hypertext Preprocessor)

A PHP nyelv túlnőtt eredeti jelentőségén. Születésekor csupán egy makrókészlet volt, amely személyes honlapok karbantartására készült. Innen ered neve is: Personal Home Page Tools. Később a PHP képességei kibővültek, így egy önállóan használható programozási nyelv alakult ki, amely képes nagyméretű webes adatbázis alapú alkalmazások működtetésére is.

A PHP hivatalosan a PHP: Hypertext Preprocessor elnevezést használja. Tulajdonképpen kiszolgálóoldali parancsnyelv, amit jellemzően HTML oldalakon használnak. A hagyományos HTML lapokkal ellentétben azonban a kiszolgáló a PHP parancsokat nem küldi el az ügyfélnek, azokat a kiszolgáló oldalán a PHP értelmező motor dolgozza fel. A programjainkban lévő HTML elemek érintetlenül maradnak, de a PHP kódok lefutnak. A kódok végezhetnek adatbázis-lekérdezéseket, létrehozhatnak képeket, fájlokat olvashatnak és írhatnak, kapcsolatot létesíthetnek távoli kiszolgálókkal – a lehetőségek száma végtelen. A PHP kódok kimenete a megadott HTML elemekkel együtt kerül az ügyfélhez.

A PHP-t parancssori alkalmazásként is telepítik, így kiválóan alkalmas kiszolgáló oldali parancsfájlok készítésére. Számos rendszergazda ma már automatizálási célokra is a PHP-t használja, pedig ezt a feladatot hagyományosan Perl- vagy héjprogramokkal oldották meg.

Amióta csak létezik a Világháló, mindig is születtek különböző megoldások a parancsprogramokra. Ahogy az utóbbi években a dinamikus tartalmat szolgáltató webhelyek iránti igény növekedett, úgy vált egyre égetőbbé a szükség arra, hogy a nagy méretű környezeteket minél gyorsabban és hatékonyabban lehessen felépíteni.

Hol léphet hát be a képbe a PHP? A PHP-t kifejezetten a Világhálóra írták, így azokra a problémákra, amelyekkel a webprogramozók nap mint nap szembesülnek, magában a nyelvben megtaláljuk a megoldásokat, a PHP beépített SQL adatbáziskönyvtárat kínál, és számos más adatbázisfajtát is önműködően támogat. Röviden tehát, mivel a PHP-t webprogramozóknak készítették, szinte minden jellemző problémára megoldást nyújtó függvényeket tartalmaz, a felhasználói munkamenetek kezelésétől az XML dokumentumok feldolgozásáig.

A PHP-t úgy tervezték, hogy modulként futtatható legyen számos kiszolgálói alkalmazással, ami azt jelenti, hogy a CGI parancsfájlok lassú indulásához hasonló gondok itt egyáltalán nem jelentkeznek. Az pedig, hogy az általános feladatokat a PHP-vel végeztethetik el felszabadítja a programozókat az alól, hogy segédkönyvtárakra kelljen támaszkodniuk, ami rontja a teljesítményt.

Van néhány megcáfolhatatlan érv, amiért a PHP-t érdemes választani. Ha más programnyelveket is ismerünk, számos alkalmazás fejlesztése során észlelni fogjuk, hogy a programozási szakasz érezhetően gyorsabb, mint várnánk. A PHP, mint nyílt forráskódú termék jó támogatással rendelkezik, amit a képzett fejlesztői gárda és az elkötelezett közösség nyújt számunkra. Ráadásul a PHP a legfontosabb operációs rendszerek bármelyikén képes futni, a legtöbb kiszolgálóprogrammal együttműködve.

Mivel a PHP lehetőséget ad a HTML elemek és a programkódok elkülönítésére, az alkalmazások fejlesztésekor lehetőség van elválasztani a kódolási, tervezési, és összeállítási szakaszt. Ez jelentősen megkönnyíti a programozók életét, azzal, hogy elmozdítja az akadályokat a hatékony és rugalmas alkalmazások kialakításának útjából.

A PHP nyílt forráskódú. Számos felhasználó szemében a nyílt forráskódú egyet jelent azzal, hogy ingyenes, ami természetesen már önmagában is előnyös.

A jól szervezett nyílt forráskódú projektek azonban más előnyökkel is szolgálnak a felhasználóknak. Felvehetjük a kapcsolatot a könnyen elérhető és elkötelezett felhasználói közösséggel, hogy bármilyen problémával is kerüljünk szembe, némi kutatással gyorsan és könnyen választ találunk rá. Ha mégsem, egy levelezőlistára küldött üzenetre általában hamar érkezik intelligens és hiteles válasz.

Úgyszintén bizonyos, hogy a feldolgozóprogram hibáinak javítása nem sokkal felfedezésük után megtörténik, és a felmerült új igényeket kielégítő szolgáltatások is hamar beépülnek a nyelvbe. Nem kell várni a következő hivatalos kiadásra, hogy a fejlesztések előnyeit élvezhessük.

Nincs a PHP működtetésére egyedileg kiválasztott kiszolgáló vagy operációs rendszer. Szabadon választhatunk olyan rendszert, amely kielégíti saját vagy ügyfeleink igényeit. Biztos, hogy kódunk továbbra is futtatható lesz, bármi mellett is döntünk.

A PHP-t alapvetően úgy tervezték, hogy alkalmas legyen számos operációs rendszeren való használatra, együttműködve különböző kiszolgálókkal és adatbázis-kezelőkkel.

Fejleszthetünk UNIX rendszerre és áttérhetünk NT alapokra minden gond nélkül. A PHP alkalmazásokat kipróbálhatjuk a Személyes webkiszolgálóval (Personal Web Server) és később telepíthetjük azokat egy UNIX rendszerre, ahol a PHP-t Apache modulként használjuk.

A PHP teljesen felület- (platform-) független, ami azt jelenti, hogy fut Windows operációs rendszeren, a legtöbb UNIX rendszeren – beleértve a Linuxot -, sőt még a Macintosh OS X gépeken is. A támogatott kiszolgálók köre igen széles. A legnépszerűbbek: Apache (szintén nyílt forráskódú és rendszerfüggetlen), Microsoft Internet Information Server, WebSite Pro, iPlanet Web Server és Microsoft Personal Web Server (Személyes webkiszolgáló). Az utóbbi kettő akkor tehet nagy szolgálatot, ha internetkapcsolat nélkül szeretnénk fejleszteni, bár az Apache is alkalmas erre Windows környezetben.

A PHP a legtöbb kiszolgálóra modulként telepíthető, vagyis nem önálló alkalmazásként, hanem a kiszolgálói folyamat részeként fut, de önálló parancssori alkalmazássá is fordítható.

Az a tény, hogy a PHP parancssori alkalmazásként futtatható, azt jelenti hogy minden kiszolgáló, amely támogatja a CGI parancsfájlokat, képes együttműködni vele, igaz, a beállítások kiszolgálónként különbözőek lehetnek.

A PHP-t alapvetően úgy tervezték, hogy könnyen összhangba hozható legyen a különböző adatbázisokkal. Ez az egyik oka a PHP népszerűségének a webalkalmazások készítése terén. Szinte minden jelenleg elérhető adatbázistípus közvetlenül, vagy az ODBC-n (Open Database Connectivity) keresztül csatlakoztatható a PHP-hez.

Szakdolgozatom programjai Windows NT operációs rendszeren, Apache webkiszolgálói környezetre és MySQL adatbázis-kezelő programokra készültek.

A PHP a http://www.php.net/ címről tölthető le.

Titkosítás

A titkosítás célja az, hogy az információt úgy juttassuk el a címzetthez, hogy annak tartalmához csak ő férhessen hozzá. Vagyis az üzenetek tartalmához való hozzáférés és azok megváltoztatása nem lehetséges azok számára, akik nem jogosultak rá. Ugyanakkor fel kell tételezni, hogy a címzetten (receiver, az információ fogadóján) és a feladón (sender, az információ forrásán) kívül létezik legalább egy harmadik résztvevő is: a támadó (intruder, adviser, attacker).

A támadó alapfeltétele a titkosításnak, mert ha nem létezne, nem lenne szükség a titkosításra sem. A támadó sok fejtörést okoz, mert amíg a beszélgető felek mindent elkövetnek, hogy „beszélgetésük titkos maradjon, addig a passzív támadó, az általános szándékkal támadó, és az aktív támadó azon vannak, hogy törekvésük kudarcba fulladjon, vagyis:

- Megpróbálnak illetéktelenül hozzáférni az üzenet tartalmához.
- A beszélgető fél nevében hamis üzenetet próbálnak küldeni a címzettnek.

A címzettnek ezért egyaránt képesnek kell lennie az üzenet olvasására és a feladó személyazonosságának ellenőrzésére is.

- Azt az üzenetet, adatot, amit a feladó el akar küldeni (és nincs szükség semmi extra műveletre annak értelmezéséhez), nyílt szövegnek (plaintext, cleartext) nevezzük.
- Azt a műveletet, amely a nyílt szöveget, annak értelmét vagy más jellemző tulajdonságait eljrejti, titkosításnak nevezzük (enciphering, encryption). Eközben valamilyen kriptográf algoritmust (cipher) használunk.
- A létrejövő értelmezhetetlen adathalmazt titkosított vagy kriptoszövegnek (ciphertext) nevezzük.
- A titkosított szöveg nyílt szöveggé való jogosult visszaalakítását megfejtésnek (deciphering, decryption) nevezzük.
- A titkosított szöveg nyílt szöveggé való jogosulatlan (értsd: kulcs nélküli) megfejtését visszafejtésnek vagy feltörésnek nevezzük.

- És mindehhez kell a kulcs (key). A titkosító módszerekkel szemben alapvető elvárás, hogy egy adott információból úgy készítsen másikat, hogy ez utóbbiból csak egy kiegészítő adat ismeretében lehessen megismerni az eredetit. Ezt a kiegészítő adatot nevezzük kulcsnak, ami egy lehetőleg hosszú, véletlenszerű jelsorozat. Ajánlott a kulcsok gyakori cseréje: ha a támadó megfejt egy üzenetváltáshoz használt kulcsot, el fogja tudni olvasni az összes korábbi, e kulccsal titkosított üzenetet, de a későbbieket csak akkor, ha a kulcs továbbra is változatlan marad. Ha viszont a kulcsot gyakran – a feltételezett visszafejtési időn belül – cseréljük, a támadót passzív tevékenységre kényszerítjük, mert idejének jelentős részét az aktuálisan használt kulcs keresése teszi ki.

A titkosító rendszerek általános követelményeit 1883-ban Auguste Kerckhoffs von Nieuwenhof holland nyelvész fogalmazta meg La criptographie militaire című művében. Ma már újabb elvárások is vannak a tárgyalt rendszerekkel szemben, de Kerckhoffs gondolatai továbbra is érvényben vannak és röviden a következők:

1.) Ha egy rendszer elméletileg nem feltörhetetlen, akkor a gyakorlatban legyen az. Az elmélet gyakorlatba juttatását valamilyen módon meg kell akadályozni. A legtöbb titkosító módszer olyan algorimusokat használ, melyek feltörhetők ugyan, de a támadásnak nem kivitelezhető idő- és/vagy tárigénye van.

- Egy rendszer elméletileg biztonságos, ha a feltörésének valószínűsége független a támadó számítási kapacitásától vagy a támadásra szánt időtől.

- Gyakorlatilag biztonságos, ha a feltöréshez ismert mennyiségű lépést kell végrehajtania, de ennek lehetetlen idő- vagy társzükséglete van.

- Nem biztonságos, megfejthető, ha a feltöréshez használt módszer tárigénye kielégíthető és időszükséglete egy bizonyos reális korláton belüli.

2.) A rendszer részleteinek kompromittálódása ne okozza a rendszer egészének kompromittálódását. Ha a támadó részinformációkat szerez egy rendszerről, ne veszélyeztesse a rendszer egészét. Ez egyfelől azt jelenti, hogy az egyes biztonsági szinteken megszerzett információk a támadót ne segítsék a további szintek áttörésében. Másfelől a támadónak a biztonsági rendszer teljes ismerete sem jelenthet segítséget. Vagyis a biztonság kizárólag a kulcs ismeretének függvénye. Ez a Kerckhoffs-elv.

„A titkosítási rendszer megbízhatósága nem függhet a titkosítás algoritmusától, azt csak a kulcs titkának megőrzése garantálja.”

A ma használt algoritmusok és protokollok nagy része teljesen nyílt, illetve ismert. Az egyik legelterjedtebb civil felhasználású rendszer, a PGP forráskódja az Internetről letölthető. Hasonlóan nyilvános a DES, a Rijndael, az RC5, az RSA (és még sorolhatnánk sokáig) specifikációja is.

3.) Az alkalmazott kulcsnak – feljegyzések nélkül is – könnyen megjegyezhetőnek és könnyen megváltoztathatónak kell lennie. Egy rendszer általában kötött méretű kulcsokat használ (legalábbis a rendszeren belül), a felhasználók kulcsainak ehhez a mérethez kell igazodniuk, vagy ehhez kell azokat igazítani. Ezt az alkalmazkodási kényszert meg lehet szüntetni a hash-függvények alkalmazásával, amelyek egy tetszőleges karakterláncból rögzített hosszúságú bitsorozatot generálnak. Így a felhasználók valóban szabadon választhatnak számukra könnyen megjegyezhető jelszót vagy akár jelmondatot is, a rendszer ennek hashértékét használja kulcsként. A kulcs cseréjére két esetben lehet szükség:

- Ha egy 8 karakteres jelszót használunk, amit az összes lehetőség kipróbálásával 1 hónap alatt ki lehet találni, célszerű a jelszót 2-3 hetente vagy gyakrabban cserélni.

- Ha felmerül a gyanúja annak, hogy jelszavunkat valaki más is ismeri. Itt jegyzem meg, hogy a támadó a legtöbbször nem veri sikerét nagydobra, hiszen azzal a kulcsok azonnali lecserélését váltaná ki. Ez a követelmény igazából azt jelenti, hogy a rejtjeles szöveget a nyílt szöveggel megegyezően kell tudni továbbítani, nem igényelhet semmilyen különleges bánásmódot, kódolást vagy speciális – a nyílt szövegtől eltérő- átviteli közeget az átvitel során.

5.) A titkosító rendszer legyen hordozható és egy személy által is üzemeltethető. A szoftvereszközök ideálisan teljesítik ezt a feltételt a legtöbb elektronikus, elektromechanikus és mechanikus eszközhöz hasonlóan.

6.) A rendszer legyen egyszerű, könnyen kezelhető és ne igényelje listányi szabályok betartását. A jól elkészített eszközök biztosítják ezt a feltételt, mert az esetleges szabályok figyelését átvállalják a felhasználótól. Gyakran rejtett módon, ritkábban a felhasználó felügyelete mellett teszik ezt. Ez a követelmény azért fontos, mert ha egy rendszer biztonsága függ a betartandó szabályoktól, akkor kérdéses a rendszer biztonsága, ha valaki – rossz- vagy jóhiszeműen – elfelejti betartani a sok szabály egyikét.

A kriptográfia önmagában nem védelem

Egy teljes kriptográfiai rendszer a következő fontosabb komponensekből épül fel, melyek gyakran nem határolhatók el élesen egymástól:

- algoritmikus rendszer, az egyes szolgáltatások matematikai háttere,
- kulcselosztás, -tárolás, -továbbítás (kulcsmenedzsment)
- kiegészítő védelmi rendszer, amely a teljes rendszer (ön)védelmét látja el: a támadások és kezelői hibák károkozását igyekszik csökkenteni, lehetőleg kiküszöbölni.

Az információvédelem megvalósítási módszereire és az algorimusok felhasználására a kriptográfiai protokollok adnak útmutatást. Ezek mondják meg, hogy mit – mivel - mikor kell titkosítani, vagy megfejteni és mit – hova – mikor kell küldeni, valamint egyéb utasításokat, ellenőrzési pontokat tartalmazhatnak. Tehát az algoritmusok a protokollnak csak eszközei. Ezt azért fontos tudomásul venni, mert a kriptográfiai algoritmusok önmagukban nem nyújtanak megfelelő védelmet, tehát a kriptográfia önmagában nem védelem!

Kriptográfiai protokollnak nevezzük azt a protokollt, amely a szabályok betartását és a csalók leleplezését kriptográfiai eszközökkel (értsd algoritmusokkal) valósítja meg.

A megfelelő információvédelemhez hozzátartozik a megfelelő ügyvitel kialakítása is: a kulcsok cseréje, tárolása, a titkosított és nyílt szövegek kezelésének szabályai. Például ha egy program lehetővé teszi, hogy egy titkosítva küldött – kapott levelet, állományt titkosítás nélkül mentsünk el - esetleg a titkosított szöveg mellé - , de ennek veszélyeire még csak nem is figyelmeztet, megkérdőjelezhető a program helyes védelmi elvi működése, hiszen komoly biztonsági rést hagy rejtve a felhasználó előtt. Más kérdés, hogy gyakran előforduló hiba az alkalmazások egy részénél az is, hogy a titkosítva érkezett adatok a megtekintés vagy feldolgozás ideje alatt plaintext (kódolatlan információ).formában tárolódnak a memóriában (mégpedig lapozható memóriában, ahonnan a virtuális memória lapozásával lemezre kerülhetnek) vagy – és az a durvább hiba – eleve ideiglenes fájlban. Ezek ellen a mezei felhasználó nem sokat tehet.

Hasonló problémát jelentenek az olyan kényelmi szolgáltatások, amelyek egyes jelszavak elmentését ajánlják fel. (E szolgáltatás egyik következménye a jelszó elfelejtése is. A felhasználó csak nézi a csillagokat és nyom egy „OK” gombot. Aztán egyszer újratelepíti a gépét és a beviteli mező üres lesz, a felhasználó pedig mérges…) Bárki kipróbálhatja: végy egy főiskolai számítógéptermet. Ülj le valamelyik géphez és nézd meg, milyen FTP programok, Commanderek vannak a gépen. Ha nem találsz semmit, ülj át egy másikhoz. Gyorsan fogsz olyan gépet találni, ahol nyugodtan mászkálhatsz egyes – egyébként jelszóvédelemmel ellátott – FTP szervereken anélkül, hogy egyetlen jelszót is megtudnál. Tipikusan ilyen például a Windows Commander,

mert egyes szöveges fájlban tárolja a titkosított jelszavakat. A Windows telefonos kapcsolatok párbeszédpanelén is bepipálható a „Jelszó mentése”, így – az egyéb védelemmel el nem látott – számítógépünket bekapcsolva bárki kapcsolódhat a mi nevünkben (és a mi számlánkra) valamilyen identifikációt igénylő kiszolgálóhoz.

Másik elvi probléma, hogy egyes operációs rendszerek illetve alkalmazások a megadott jelszót a memóriában eltárolják – gyakran plaintext formában – és ha legközelebb szükség van rá nem kérik újra a felhasználótól (vagy egy autentikáló, azonosító szervertől), hanem a memóriából keresik elő azt. A baj azonban az, hogy gyakran nemcsak az operációs rendszer tudja elővenni a kért adatot, hanem egy bejuttatott szimatoló (sniffer) program is. A jelszó (egyirányúan) kódolt tárolása sem mindig megoldás, mert gyakran a jelszót kódolva kell elküldeni, így a kódolt jelszó valamilyen szempont szerint egyenértékű a kódolatlannal. (Csak a kódolt az hottentottául van.)

A jelszó egyébként is legtöbbször az emberi meggondolatlanság és kényelem miatt kerül veszélybe, hiába támogatja a rendszert egyébként erős algoritmikus háttér. Általában is igaz, hogy ha egy rendszerbe valaki be akar jutni, annak érdemes előbb a „humán” oldalról megközelíteni a rendszert. Néhány felhasználó a monitorára, vagy a billentyűzetére írva tárolja a jelszavát. Egyes titkárnők tudják a főnökül jelszavát és gyakran szó nélkül megmondják a telefonon bejelentkező szervizesnek. A módszer használható arra is, hogy egy titkos telefonszám tulajdonosának adatait a tudakozóból megszerezzük. Mindez lényegében az jelenti, hogy a támadó az emberek manipulálásával kerüli meg a védelmi rendszert. Ezt a módszer hívják nemes egyszerűséggel „social engineering” -nek (társadalmi technikának). Egy összetett védelem esetleges gyenge pontjai így nemcsak magából a rendszerből, hanem a kezelői hibákból, emberi mulasztásokból is adódhatnak. Minden rendszer leggyengébb láncszeme az ember.

Az adatlopás legfőbb okai a következő 7 emberi tulajdonság és cselekedet, melyek egyaránt lehetnek motiváló tényezők vagy módszerek:

- Hiúság	Ego
-Sikkasztás	Embezzlement
-Lehallgatás	Eavesdropping
-Ellenségeskedés	Enmity
-Kémkedés	Espionage
-Zsarolás	Extortion
-Hibás döntés	Error

Hashfüggvény

Olyan függvény, ami tetszőleges hosszúságú bemenetet egy véges hosszúságú (tipikusan pl. 128 bitnyi) kimenetre képez le. Emiatt a leképezés nyilván nem egy-egy értelmű lesz (azaz egy hash-értéket sokféle lehetséges bemenet előállíthatott), viszont az ilyen függvényeket úgy választják meg, hogy nagyon nehéz (gyakorlatilag lehetetlen) legyen: adott h(m) hash-értékhez egy megfelelő m forrás-bemenetet találni (neminvertálhatóság, *noninvertability*)

- adott m-hez olyan m'-t találni, hogy h(m) = h(m') (leképezés ütközés-ellenállósága, *image collision resistance*)
- két olyan tetszőleges m-et és m'-t találni, hogy h(m) = h(m') (ütközés-ellenállóság, *collision resistance*)
- illetve a bemenet bármely bitjének megváltozása a kimenet bitjeinek mindig átlagosan felét változtassa meg (lavinatulajdonság, *avalanche property*)

Az ilyen módon kapott értékeket sok esetben lehet a hosszú tartalom "rövid neveként" kezelni. Digitális aláírásnál például a tartalom hash-ét szokták a küldő titkos kulcsával kódolni, így letagadhatatlanul bizonyítva az egyediséget.

A fenti tulajdonságok miatt a hashfüggvények az egyszerű ellenőrzőösszegek "felnőtt" változatának tekinthetők.

- Ismertebb hashfüggvények
- MD5
- SHA1
- RIPEMD-160

MD5

Egyirányú hashfüggvény. Algoritmusa az input üzenetből 128 bites outputot készít, mely egyértelműen azonosítja az üzenetet, és adatintegritás-ellenőrzésre is használható. Az MD5-öt Ron Rivest fejlesztette ki az RSA-nál, elsősorban digitális aláírással dolgozó rendszerekhez.

Ez a módszer azért lehet MD5 esetén különösen hatékony az adatintegritás sérülésének kimutatásához, mivel az MD5 képzés algoritmusa garantálja azt, hogy a kiindulási állományban bekövetkezett csekély (néhány bites) módosultság drasztikusan megváltoztatja a generált MD5 kulcsot.

Programozás PHP nyelven

A PHP szkripteket általában HTML kódba épülve gépeljük be. Hiba esetén a hibás sor feltüntetésével hibaüzenetet kapunk. Ha olyan hibát vétettünk, hogy a böngésző nem tudja megállapítani, hogy a kódrészletünk PHP nyelven íródott, akkor a teljes kód megjelenik. Vannak tiszta PHP kódot tartalmazó állományok is a szakdolgozatban, ezeket a beágyazott programokat, csak másik HTML-be épült php oldal meghívásakor használjuk.PHP blokkok kezdése és befejezése

Amikor PHP oldalakat írunk, a PHP kódblokkok elejét és végét jelző különleges nyitó- és zárókódokkal tudatnunk kell a feldolgozóval, mely részeket hajtsa végre. Ha nem adjuk meg, hogy a fájl mely részei tartalmaznak PHP kódot, az értelmező mindent HTML-nek tekint és változtatás nélkül továbbküldi a böngésző számára.

A hagyományos nyitóelem a <?php a hagyományos záróelem pedig ?>.

Egy HTML állományba korlátlan számú –egymástól független – PHP kódrészlet szúrható be, a <?php és ?> tagok segítségével.

A böngészőnk számára a PHP kódjaink eredményei úgy jelennek meg, mintha azok HTML kódok lennének. A PHP kód átalakításáról a Web kiszolgáló (jelen esetben az Apache) gondoskodik. A kiszolgáló végzi el az összes PHP kód kiértékelését és adja át a kódot a böngészőnknek. Maga a PHP kód is tartalmazhat HTML elemeket, melyek segítségével tudjuk a szövegünket formázni, illetve mindenféle egyéb műveletet végrehajtani. Biztonságunk szempontjából is nagyon fontos dolgok, hogy a böngészőnk, már csak a végeredményt kapja meg, függetlenül attól, hogy mi volt a PHP kódunkba írva, maga a kód nem látszik és semmi sem utal arra, hogy ott korábban egy másik nyelv kódja állt. Erről könnyen meggyőződhetünk, ha megnézzük a betöltött oldal forrását.

Az utasítás a feldolgozónak adott parancs. Az utasításokat a legtöbb esetben pontosvesszővel kell lezárni. A pontosvessző tudatja a feldolgozóval, hogy az utasítás véget ért.

A Világhálón alapvetően a HTML űrlapokon keresztül áramlik az információ a felhasználó és a kiszolgáló között. A PHP-t arra tervezték, hogy a kitöltött HTML űrlapokat feldolgozza. A szuperglobális változók, olyan tömbök, amelyeket beépítettek a PHP nyelvbe, automatikusan töltődnek fel hasznos elemekkel, és minden hatókörben elérhetők. A szuperglobális tömbök egy függvényben vagy tagfüggvényben anélkül is elérhetők, hogy a global kulcsszót használnánk. Ilyenek például:

$_COOKIE Böngészősüti-ként beállított kulcsokat és értékeket tartalmaz.

$_GET A programnak HTTP GET módszerrel eljuttatott kulcsokat és értékeket tartalmazza

$_POST A programnak HTTP POST módszerrel eljuttatott kulcsokat és értékeket tartalmazza

A PHP nyelv egyik meghatározó tulajdonsága, hogy nagyon könnyen képes adatbázisokhoz csatlakozni és azokat kezelni. A PHP, az Apache és a MySQL hármasára rengeteg program támaszkodik.

MySQL függvények PHP-ben

-mysql_connect

Leírás

resource **mysql_connect** ([string $server [, string $username [, string $password]]])

Megnyit vagy újrahasznosít egy MySQL szerver kapcsolatot.

Paraméterek

server

A MySQL szerver. Tartalmazhat egy port számot is, pl. "hostname:port" vagy egy helyi socket útvonalát, pl. ":/path/to/socket".

Ha a mysql.default_host PHP direktíva nem definiált (ez az alapértelmezés), akkor az alapértelmezett értéke ' localhost:3306'. SQL safe mode-ban, ezt a paramétert kihagyja, és mindig 'localhost:3306' értéket használja a PHP.

username

A felhasználónév. Alapértelmezett értéke a mysql.default_user. SQL safe mode-ban, ezt a paramétert kihagyja, és a szerverfolyamat a tulajdonosának a nevét használja a PHP.

password

A jelszó. Alapértelmezett értéke a mysql.default_password. In SQL safe mode-ban, ezt a paramétert kihagyja, és egy üres jelszót használ a PHP.

Visszatérési érték: Sikeres végrehajtás esetén visszaad egy MySQL kapcsolat azonosítót, hiba esetén pedig FALSE-ot.

Az esetleges hibaüzeneteket a @ operátorral nyomhatod el, a függvényhívás elé való beszúrásával.

-mysql_create_db

Leírás

bool **mysql_create_db** (string $database_name [, resource $link_identifier])

A **mysql_create_db()** függvény megkísérel létrehozni egy új adatbázist a megadott kapcsolatazonosítón keresztül.

Paraméterek

database_name

A létrehozandó adatbázis neve.

link_identifier

A MySQL kapcsolat. Ha a kapcsolatazonosító nincs megadva, akkor az utólsó mysql_connect()-el megnyitott kapcsolatot használja. Ha nem talál semmilyen kapcsolatot, megpróbál létrehozni egyet úgy, mintha a mysql_connect() paraméterek nélkül lett volna meghívva. Ha esetleg semmilyen kapcsolatot nem talál és nem is sikerül létrehoznia, akkor egy E_WARNING szintű figyelmeztetés generálódik.

Visszatérési érték: Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad. Az esetleges hibaüzeneteket a @ operátorral nyomhatod el, a függvényhívás elé való beszúrásával.

-mysql_close

Leírás

bool **mysql_close** ([resource $link_identifier])

A **mysql_close()** függvény bezárja az adott azonosítójú nem-perzisztens MySQL kapcsolatot. Ha nem adsz meg *link_identifier* paramétert, akkor az utoljára megnyitott kapcsolatot zárja le.

A **mysql_close()** függvény használata általában szükségtelen, mert a nem perzisztens kapcsolatok a szkript végén bezáródnak. Lásd még: erőforrások felszabadítása

Paraméterek

link_identifier

A MySQL kapcsolat. Ha a kapcsolatazonosító nincs megadva, akkor az utólsó mysql_connect()-el megnyitott kapcsolatot használja. Ha nem talál semmilyen kapcsolatot, megpróbál létrehozni egyet úgy, mintha a mysql_connect() paraméterek nélkül lett volna meghívva. Ha esetleg semmilyen kapcsolatot nem talál és nem is sikerül létrehoznia, akkor egy E_WARNING szintű figyelmeztetés generálódik.

Visszatérési érték

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

-mysql_select_db

mysql_select_db — Kiválaszt egy MySQL adatbázist

Leírás

bool **mysql_select_db** (string $database_name [, resource $link_identifier])

Beállítja, hogy a megadott adatbázis legyen az aktuálisan kiválasztott a kapcsolatazonosítóhoz tartozó szerveren. Minden további mysql_query() hívás az aktív adatbázison lesz végrehajtva.

Paraméterek

database_name

A kiválasztandó adatbázis neve.

link_identifier

A MySQL kapcsolat. Ha a kapcsolatazonosító nincs megadva, akkor az utólsó mysql_connect()-el megnyitott kapcsolatot használja. Ha nem talál semmilyen kapcsolatot, megpróbál létrehozni egyet úgy, mintha a mysql_connect() paraméterek nélkül lett volna meghívva. Ha esetleg semmilyen kapcsolatot nem talál és nem is sikerül létrehoznia, akkor egy E_WARNING szintű figyelmeztetés generálódik.

Visszatérési értékek

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

-mysql_query

Leírás

resource mysql_query (string $query [, resource $link_identifier])

A mysql_query() függvény egy egyedüli kérést küld (összetett kérés nem engedélyezett) a *link_identifier* paraméterben megadott szerver aktív adatbázisához.

Paraméterek

query

Egy SQL kérés

A kérésnek nem szabad pontosvesszővel végződnie.

link_identifier

A MySQL kapcsolat. Ha a kapcsolatazonosító nincs megadva, akkor az utólsó mysql_connect()-el megnyitott kapcsolatot használja. Ha nem talál semmilyen kapcsolatot, megpróbál létrehozni egyet úgy, mintha a mysql_connect() paraméterek nélkül lett volna meghívva. Ha esetleg semmilyen kapcsolatot nem talál és nem is sikerül létrehoznia, akkor egy E_WARNING szintű figyelmeztetés generálódik.

Visszatérési értékek

A SELECT utasításra alkalmazott **mysql_query**() függvény siker esetén egy erőforrással tér vissza, hiba esetén pedig FALSE-al.

Más típusú SQL utasítások esetében, mint pl. UPDATE, DELETE, DROP, stb, a mysql_query() függvény sikeres végrehajtás esetén TRUE-val tér vissza, hiba esetén pedig FALSE-al.

A visszaadott eredmény feldogozása céljából a kapott erőforrás átadható a mysql_fetch_array() függvénynek, vagy egyéb függvényeknek, amelyek eredményhalmazzal dolgoznak.

Használd a mysql_num_rows() függvényt a SELECT utasítás által visszaadott sorok számának lekérdezésére vagy a mysql_affected_rows() függvényt arra, hogy megtudd, hány sort érintett a DELETE, INSERT, REPLACE, vagy UPDATE utasítás.

A **mysql_query**() függvény akkor is meghiúsul és FALSE értékkel tér vissza, ha nincs megfelelő engedélyed a kérés által hivatkozott táblá(k)hoz.

-mysql_free_result

Leírás

bool **mysql_free_result** (resource $result)

A **mysql_free_result**() függvény az összes *result* eredményazonosító által használt memóriát felszabadítja.

A memória a php program végén automatikusan felszabadul **mysql_free_result()** függvény nélkül is.

Paraméterek

result

A feldolgozandó eredményhalmaz erőforrás. Ez az eredményhalmaz egy mysql_query() hívás eredményeként kellett létrejöjjön.

Visszatérési értékek

Siker esetén TRUE értékkel tér vissza, ellenkező esetben FALSE értéket ad.

Érdemes megjegyezni, hogy a csak mysql_query() csak SELECT, SHOW, EXPLAIN, és DESCRIBE kérések esetén ad vissza resource-t.

-mysql_fetch_array

(PHP 4, PHP 5, PECL mysql:1.0)

mysql_fetch_array — Kérés egy sorát adja vissza (tetszőleges) tömb formájában

Leírás

array **mysql_fetch_array** (resource $result [, int $result_type])

Az eredmény következő sorával tér vissza tömb formájában, és a belső adatpointert a sor elejére állítja.

Paraméterek

result

A feldolgozandó eredményhalmaz erőforrás. Ez az eredményhalmaz egy mysql_query() hívás eredményeként kellett létrejöjjön.

result_type

A visszaadott tömb típusa. Az allábbi konstansok értékét veheti fel: MYSQL_ASSOC, MYSQL_NUM, az alapértelmezett értéke pedig MYSQL_BOTH.

Visszatérési értékek

Egy tömböt ad vissza, amely a következő sornak felel meg, ha pedig nincs több sor, akkor FALSE-ot. A visszaadott tömb típusa a *result_type* paraméter értékétől függ.

A PHP md5() függvénye

Az php md5 függvénye azt tudja, hogy egy bemeneti szövegből egy 32 karakterből álló hexadecimális számot állít elő. Ennek a számnak az a tulajdonsága, hogy

- a számból nem fejthető vissza jelentős ráfordítás (értsd: 100-200 év) alatt a string
- a bemenet nagyon kis változása (egy karakter) egy teljesen más számot eredményez

Ezért ideálisan használható olyan titkosítási feladatokra, ahol csak egyirányú titkosítás kell. Pl.: ha egy jelszót az md5 algoritmussal titkosítva tárolunk, akkor lehetővé válik az, hogy ilyet írjunk:

```
if (md5($ellenorizendo_jelszo)==$titkositott_jelszo)
```

Ekkor a jelszavak nem lesznek senki számára se visszafejthetőek jelentős ráfordítás nélkül, mégis le tudjuk ellenőrizni az érvényességét.

Jelszavak sózása

Szokás a jelszót még *sózni*, azaz kódolás előtt egy fix méretű véletlen szöveggel összefűzni, majd ezt a szöveget a véletlen szöveggel (sóval) együtt tárolni.

A jelszavakat tehát "$so.md5($so.$jelszo)" formában tároljuk. Használatosak az összefűzés helyett más műveletek, pl. xor. Az így eltárolt jelszót a következőképpen ellenőrizzük:

```
if (substr($sozottjelszo, 0, $sohossz)
.md5(substr($sozottjelszo, 0, $sohossz).$jelszo) == $sozottjelszo)
    print("Beléptél!");
else
    print("Rossz jelszó");
```

Ahol a $sozottjelszo az adatbázisban a feljebb tárgyalt formátumban tárolt jelszó, a $sohossz a sózásra használt szöveg hossza, a $jelszo a felhasználó által megadott, ellenőrizendő jelszó.

Munkamenetek

Állapotok tárolására a sütiken (COOKIE) és GET típusú lekérdezéseken kívül más lehetőségeket is kínál a php. A session = munkafázis (munkamenet). Ezen beépített függvények segítségével kényelmesebben tárolhatunk állapotokat. A munkamenet egy egyedi azonosítót (session id-t) biztosít mellyel az egyik oldalról a másikra vihetünk információkat. Amikor a felhasználó betölt egy valamilyen munkamenetet támogató oldalt, kap egy azonosítót, vagy a régebbi látogatásakor kapott azonosító kerül felhasználásra. A munkamenethez kapcsolódó értékek, változók egy ideiglenes fájlban tárolódnak. Az azonosítót általában egy sütiben továbbítja, de meg lehet oldani GET típusú lekérdezéssel is (ez akkor hasznos, ha tudjuk, hogy a felhasználó letiltotta a sütiket).

A session_start() függvénnyel lehet elindítani egy munkamenetet. Generál nekünk egy azonosítót, majd egy sütiben elküldi. Ha a session elindult, a session_id() függvénnyel kérdezhetjük le és módosíthatjuk az azonosítót. Ha egy szöveget adok paraméterül a session_id() függvénynek onnantól kezdve az adott szöveg lesz a session id.

Ha még nem kaptunk vissza azonosítót, akkor létrejön egy SID nevű állandó. Ennek értéke pl.: PHPSESSID=1627bc603736b82172ee4b4f9ce67286. Ez akkor hasznos nekünk, ha a felhasználó letiltotta a sütiket, mert ha a SID értékét hozzáadjuk az URL-hez, a munkamenet azonosító automatikusan átvételre kerül amikor meghívjuk a session_start()-ot.

A session_start() meghívása után használhatjuk a $_SESSION szuperglobális tömböt. Ezt a tárolni kívánt változó nevével indexeljük. Természetesen tömb típusú változókat is tárolhatunk benne.

Munkameneteket a session_destroy(); függvénnyel törölhetünk. Ekkor a következő oldalról a munkamenet semmilyen adata nem lesz már hozzáférhető.

Fontos azonban megemlíteni, hogy függvényt meghívó oldal számára a függvényhívás után is elérhetőek maradnak a munkamenet-változók. Ezek tényleges megsemmisítéséhez a következőt kell csinálni még a session_destroy(); előtt:

```
$_SESSION=array();
session_destroy();
```

Egy üres tömböt adunk a $_SESSION értékének.

A Webbank weboldal felépítése

A megírt program a http://www.webbank.uw.hu/ oldalon tekinthető meg. Ez az alkalmazás lehetővé teszi a felhasználóknak, hogy igénybe vehessék a Webbank szolgáltatásait. Regisztrálás után lehetőségük van a számlájukról különböző információkat lekérdezni: számla részletes adatai, számlatörténet, számlakivonat; lekötni és visszavezetni a számlájukra egy bizonyos összeget; valamint átutalást tudnak kezdeményezni a Webbankon belül. Ezen felül meg tudják változtatni a belépési jelszavukat, hogy az számukra könnyebben megjegyezhető legyen. Elvárás az alkalmazással szemben, hogy az adatokat biztonságosan kezelje, külső felhasználók ne férhessenek hozzá az adatokhoz. Valamint elvárás még az alkalmazástól a felhasználóbarát felületek létrehozása, az áttekinthetőség biztosítása

A bejelentkező oldal 2 keretből tevődik össze a felső menüsorból és az alsó kiírásra szolgáló részből, ahol kezdetben a Webbank logóját láthatjuk, később pedig a választott menünek megfelelő adatokat..

Fontosnak tartom, hogy a felhasználó átlássa az oldalon lévő menüpontokat, és azokat megfelelően tudja értelmezni, ezért fontos olyan gombok és megjegyzések használata, amelyek a felhasználó tájékozódását elősegítik. Sok oldalon a rengeteg információ, mozgókép, futószöveg és reklám között a felhasználó csak a lényeget nem találja. Fontosnak tartom a színválasztást is. Én a kék és a sárga színeket választottam alapszínként, hogy a felhasználó szemét megnyugtassa.

A felső menüsorban 4 Menüpont közül választhatunk

1. Webbanki szolgáltatások: ahol is a Webbank szolgáltatásairól kapunk tájékoztatást

2. Belépés a direct bankba, ahol a regisztrált felhasználók az interneten keresztül tudják használni a Webbank, folyószámlához kapcsolódó szolgáltatásait.

3. Regisztráció a belépéshez: ahol egy regisztrációs űrlap kitöltése után a belépéshez szükséges adatokat egy email-en keresztül kapjuk meg és akár rögtön használni is tudják a Webbankba történő belépéshez.

4. Kapcsolat űrlap: egy levelező felületű űrlap kitöltésével az érdeklődők kérdéseket tehetnek fel a Webbanki szolgáltatásokkal kapcsolatban.

A felső menüsorban egy gombhoz 2 eseménykezelőt rendeltem, ha rajta állunk egy gombon akkor a gomb kap egy kis fényt, 3 dimenziós hatást keltve - ezt egy másik kép megjelenítésével lehet megtenni - , majd ha a gombot lenyomjuk a gomb közepén egy örvény effekt a gombra kattintás hatását kelti. Egy gomb 3 állapotát láthatjuk az alábbi képeken: A gombokhoz a képek az Adobe Photoshop layer és effekt funkcióinak a segítségével készültek.

Az események kezelését JavaScipt-tel valósítottam meg, az alábbi programrészlet az első gomb eseményeinek lekezelését megvalósító forráskód.

Az MM_preloadImages függvény: A rejtett képek előtöltésére szolgál, hogy pl. egy képcsere folyamatos legyen vagy az idővonal képkockáin vagy a rejtett rétegeken használt képek a megfelelő pillanatban meg tudjanak jelenni.

Az MM_swapImage függvénnyel valósítható meg a képcsere. Egy gomb kezeléséhez a teljes kódrészlet a következő:

```
<body bgcolor=rgb(60%,70%,90%)
onLoad="MM_preloadImages('image/gomb1.jpg','#931256829080');MM_preloadImages('image/
gomb2.jpg','#931256868360');MM_preloadImages('image/gomb3.jpg','#931256906470');
MM_preloadImages('image/gomb4.jpg','#931256919550')"
 link="#FF0000" vlink="#000000">

<table align="left" border="0" width=100% cellspacing="0" cellpadding="7">
  <tr><td align="center">
    <script language="JavaScript"><!--

function MM_preloadImages() { //v2.0
  if (document.images) {                // A dokumentum <IMG> elemeit tükrözi, előfordulásuk sorrendjében
```

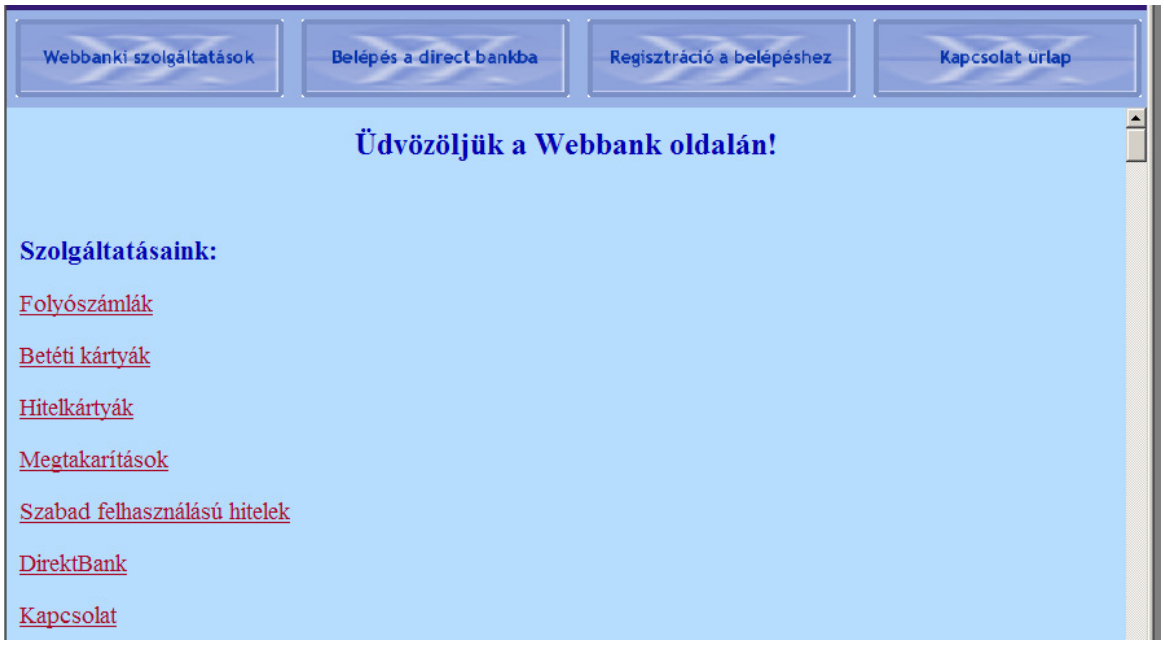
```
    var imgFiles = MM_preloadImages.arguments; //az argumentumok tömbbe rendezve
    if (document.preloadArray==null) document.preloadArray = new Array();
    var i = document.preloadArray.length;
    with (document) for (var j=0; j<imgFiles.length; j++) if (imgFiles[j].charAt(0)!="#"){
//bejárjuk az argumentumokból álló tömböt
      preloadArray[i] = new Image;  //létrehozunk egy kép objektumot
      preloadArray[i++].src = imgFiles[j];  //megadjuk a kép elérési útját
  } }
}
function MM_swapImage() { //v2.0
  var i,j=0,objStr,obj,swapArray=new Array,oldArray=document.MM_swapImgData; //elmentjük
a jelenlegi kép adatait;
  for (i=0; i < (MM_swapImage.arguments.length-2); i+=3) {
    objStr = MM_swapImage.arguments[(navigator.appName == 'Netscape')?i:i+1];
//leellenőrizzük a böngésző nevét
    if ((objStr.indexOf('document.layers[')==0 && document.layers==null) ||
        (objStr.indexOf('document.all[') ==0 && document.all ==null)) //megnézzük hogy
vannak-e a dokumentumban layer-ek vagy más objektumok, ezek ismeretében szerkeszti meg a
kép objektum elérését.
      objStr = 'document'+objStr.substring(objStr.lastIndexOf('.'),objStr.length);
    obj = eval(objStr);
    if (obj != null) {
      swapArray[j++] = obj;
      swapArray[j++] = (oldArray==null || oldArray[j-1]!=obj)?obj.src:oldArray[j]; //A kép
betöltésének útvonala
      obj.src = MM_swapImage.arguments[i+2];
  } }
  document.MM_swapImgData = swapArray; // a kép visszaállításra vonatkozó információkat
eltároljuk
}
// --></script>
```

```
<a
onMouseOut="MM_swapImage('document.Image1','document.Image1','image/gomb1.jpg','#931
256829080');window.status='SZOLGALTATASOK'; return true;"
onMouseOver="MM_swapImage('document.Image1','document.Image1','image/gombon1.jpg','#
931256829080');window.status='SZOLGALTATASOK'; return true;"
   target="main" href="szolgaltatasok.htm"
onMouseDown="MM_swapImage('document.Image1','document.Image1','image/gomble1.jpg','#
931256829080');window.status='SZOLGALTATASOK'; return true;"
   target="main" href="szolgaltatasok.htm">
<img name="Image1" border="0" src="image/gomb1.jpg"  height=60 ></a>
```

Az első menüpont:

A Webbanki szolgáltatások a szolgaltatasok.htm oldalt nyitja meg, amelynek a felső részében lévő hiperlinkek segítségével tudunk navigálni az oldalon lévő információk között. Minden szolgáltatás neve mellett szerepel egy vissza mutató hiperlink, amely visszanavigál a választható szolgáltatások elejére. A DirectBank szolgáltatás kiválasztásával a Webbank direct szolgáltatás leírásához ugrunk, ahol lehetőség van a felső menüből is elérhető Regisztráció a belépéshez űrlap-ot tartalmazó oldalra navigálni. A Kapcsolat szolgáltatás leírásán 2 hiperlinket is kiválaszthatunk: az ugyancsak a felső menüsorból elérhető Kapcsolat űrlapot és a Webbank bank fiókjainak az elérhetőségét, amely az elerhetoseg.htm oldalt nyitja meg, amely a bankfiók adatairól szolgáltat információt.:

elerhetoseg.htm



szolgaltatasok.htm

Vissza navigálás

A második menüpont:

Belépés a direct bankba kiválasztásával a belepes.php oldal töltődik be. A belépés az alkalmazás szempontjából az egyik legkritikusabb pont. A belépéshez szükséges felhasználói azonosító és jelszó ugyanis legtöbbször a nyílt Interneten utazik, amit viszonylag egyszerű módszerekkel le lehet hallgatni. Ennek következtében a webes alkalmazások esetén bevett szokás, hogy a felhasználói adatokat külön táblában tároljuk, elkülönítve az adatbázis hozzáféréshez szükséges felhasználókat tároló táblától. Ezt ebben az alkalmazásban a felhasznalo_adat és a felhasznalo táblákkal értem el, köztük a kapcsolatot a felhasznalo_adat

elsődleges kulcsa biztosítja. A jelszó tárolása az md5() titkosító algoritmus segítségével történik, így a táblában a tényleges jelszó helyett egy hosszú karaktersorozatot láthatunk csak. Az md5() – függvényt még egy 8 karakteres véletlen karaktersorozattal is összefűztem, amellyel a jelszó biztonságosabbá vált. Az alkalmazás biztonságos üzemeltetése szempontjából fontos, hogy valamennyi oldal esetén azonosítható legyen a felhasználó, illetve az, hogy érvényes legyen a bejelentkezés, ezért beléptetés során a felhasználót el kell látnunk valamilyen azonosítóval, mely segítségével a későbbiekben is ellenőrizhetjük van-e jogosultsága a hivatkozott lap eléréséhez. Az adatbázis eléréséhez szükséges azonosító adatokat a db.inc állományban tároltam:

```
$hostName = "kiszolgalo.hu";
$databaseName = "adatbazisnev";
$username = "felhasznalonev";
$password = "jelszo";
```

,amely állományt minden adatbázist használó állomány elejére include-al meghívok. Ez az adatok védelme érdekében, már egyfajta biztonságot is nyújt. A db.inc állományban az azonosító adatokon túl még tartalmaz néhány olyan eljárást, amelyeket minden oldalon használok, ahol a felhasználótól adat érkezik, köztük a már megemlített clean() függvényt.

A clean() függvény segítségével védhetjük meg adatbázisainkat az illetéktelen behatolóktól. A függvény nem csinál mást, mint levágja az input paraméter értékét, valamint a sztringből eltávolítja az összes olyan karaktert, amelyek segítségével olyan parancsokat, utasításokat lehet lefuttatni, amellyel behatolhat valaki a számítógépünkre, illetve az adatbázisunkba.

```
function clean($input, $maxlength)
{
  $input = substr($input, 0, $maxlength);
  $input = EscapeShellCmd($input);
  return ($input);
}
```

Ha például a felhasználó a beviteli mezők valamelyikére beírja, hogy ildiko; GRANT ALL PRIVILEGES ON *.* TO ’crazy’ IDENTIFIED BY ’winner’ WITH GRANT OPTION;, és a parancs sikeresen lefut, akkor attól a pillanattól ő is teljes jogúan hozzáférhet az adatbázisunkhoz. A GRANT parancs a felhasználói jogosultságok kezelésére szolgál. ALL

PRIVILEGES- megadja a felhasználónak az összes jogot. ON kulcsszó után az adatbázisnév következik, amely ebben az esetben *.*. A TO kulcsszó után a felhasználónév, az IDENTIFIED BY után pedig egy jelszó következik. Mint láthatjuk ennek a függvénynek a használata valamennyi olyan paraméter esetén ajánlott, ahol a felhasználó bevihet szöveget. Az E**scapeShellCmd()** függvény szóközre cseréli minden olyan speciális karaktert, ami nem kívánt parancsok futtatását okozná. Ez a függvény arra használható, hogy biztosítsuk a felhasználótól érkező adatok tisztaságát

A logincheck() függvény segítségével valamennyi oldalon ellenőrizhető, hogy jogos-e az oldal használata és amennyiben nem az, akkor a program átirányít minket a belepes.php oldalra.

```
function logincheck()
{
  session_start();

  if (!session_is_registered("user"))
    //visszaléptetés az beléptető képernyőre
    header("Location: belepes.php");
}
```

Egy egyszerű módszerrel elrejthető, hogy PHP-t használok, így lassítva le a támadót, aki fel akarja deríteni a rendszer gyenge pontjait. A php.ini-ben az explose_php=off beállítással csökkenteni lehet ezeket az információkat.

A showerror() függvény segítségével írathatjuk ki az esetleges adatbázis elérési és lekérdezési hibáinkat.

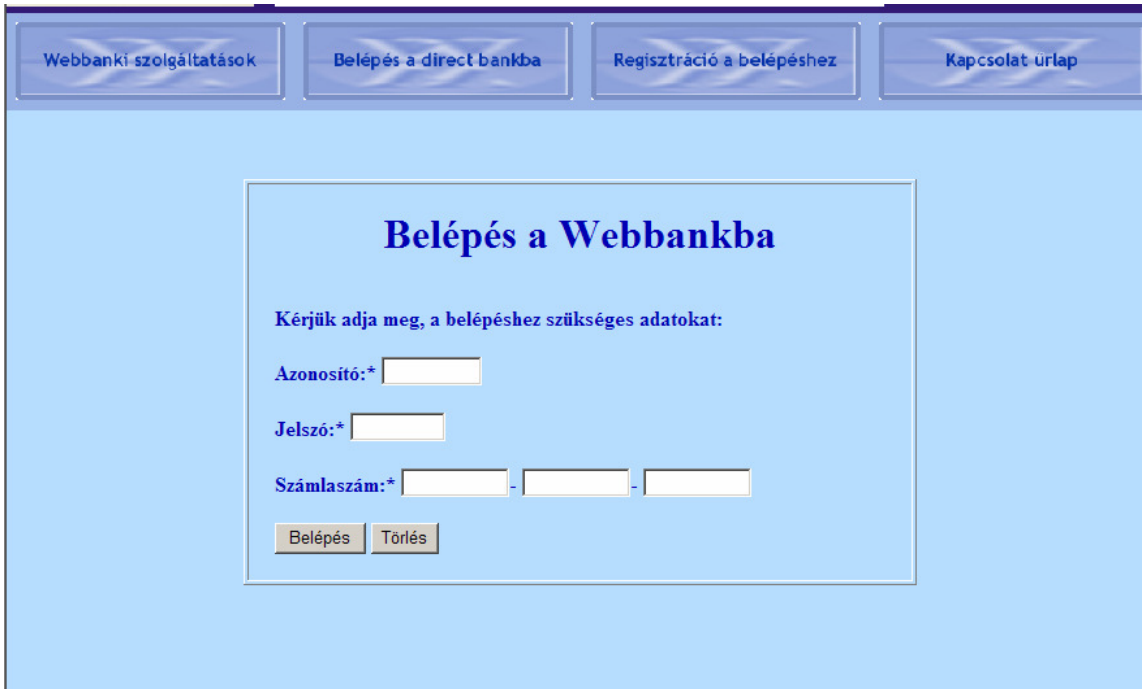
```
function showerror()
{
  if (mysql_error())
    die("Hiba " . mysql_errno() . " : " . mysql_error());

}
```

A belepes.php oldal behívásakor elindítunk egy munkamenetet. Sikeres belépéskor a munkamenet számára létrehozunk a $_SESSION[] globális változóban egy user nevű változót, amely a felhasználó azonosító értékét kapja. Az egyes oldalakra való belépéskor ennek a változónak a meglétét ellenőrizzük a session_registered() függvény segítségével. Ha nincs meg, akkor visszalépünk a belepes.php oldalra, ha megvan akkor megjelenik a hívott oldal.

Az alkalmazás biztonságát tovább növeli, hogy a db.inc állományra nem engedélyeztem a kiszolgálónak az ilyen kiterjesztésű állományok lekérdezését. Ezt a httpd.conf fájlban a

```
<Files ~ ".inc$">
  Order allow,deny
  Deny from All
  Satisfy All
</Files>
```

bejegyzéssel, majd az Apache kiszolgáló újraindításával tehető meg.



belepes.php

Ebben az ablakban a belépéshez szükséges eljárást valósítottam meg. A belépéshez egy űrlap került megjelenítésre, amelyben bekérem az azonosítót, a jelszót, a számlaszámot. Az alkalmazásba történő belépés csak akkor kezdődik meg, ha mindhárom beviteli mezőt jól töltöttük ki. A mezők kitöltésének vizsgálatára az empty() függvényt használom. Ha nincs kitöltve valamelyik mező, vagy hibás adatokat adott meg, újra megjelenik az űrlap valamint a

hibalista, mivel ugyanazon a belepes.php oldalon van az adatok bekérése és a feldolgozása is. Ha nem talál ilyen adatokkal felhasználót, újra kéri az adatokat. A hibalista kiírása az első alkalommal nem fog megtörténni, az $elsoe változó értékét egy rejtett beviteli mezőn keresztül adjuk át. Az isset($_POST[elsoe]) függvény a változó meglétét vizsgálja, de ez a változó csak az első lefutás után, a Belépés gomb lenyomásakor kap értéket. A form -ban található $_POST[’nev’] globális változók a beviteli mezőbe gépelt karakterek megőrzésére szolgálnak az elküldés után, ahol a nev mindig az aktuális beviteli mező nevét jelenti, melynek a hatására az űrlap újbóli betöltése során beíródnának az eredeti tartalmukkal, éppen ezért, ha valamelyik adat nem stimmel az összes adatot kitörlöm (üres karaktersorozatot adok neki értékként), hogy az adatok megfejtése ne legyen könnyebb az illetéktelenül bejutni vágyóknak.

A FORM utasítás részben a jelszó bekérésére a password típust használtuk, amely elrejti a képernyőn a bevitt karaktereket az illetéktelen felhasználók szeme elől.

A jelszó mező és a számlaszám mezők között egy speciális ugras(hány_karakter_után,melyik_mezőről,hova_ugorjon) JavaScript függvényt használok, mivel a jelszó és a számlaszám részmezőinek a hossza rögzített így a 8. karakter begépelése után átugrik a következő beviteli mezőre, ezzel megkönnyítve, hogy a felhasználónak ne kelljen minden mező kitöltése után a tabulátor vagy enter billentyűt használnia.

```
function ugras(hossz,mezo1,mezo2){
if (document.getElementById(mezo1).value.length>=8)
  {document.getElementById(mezo2).focus();
}//10
```

Sikeres belépés után egy 4 részre osztott weboldal töltődik be, ahol a bal felső sarokban a logó és az aktuális nap dátuma látható. A bal oldalon választható ki a folyószámlával végzendő feladat. Jobb oldalon fent két menüpontból álló menüsort találunk: Jelszó módosítása és Kilépés. Jobb oldalt lent pedig a számlánk aktuális állapotáról kaphatunk aktuális információkat, és itt jelenik meg a bal oldali menüsorból választott feladat eredménye is.

Belépéskor a szamlaadatok.php oldal hívódik meg a jobb alsó sarokban. Ez az oldal az első sorban kiírja az utolsó belépés dátumát. Majd sorban a számla adatait: Számlaszám, Devizanem, Számla elnevezése, Egyenleg.

szamlaadatok.php

Ezek az adatok látszanak, de ennek az oldalnak van egy rejtett funkciója is, amellyel az atutalas táblából kigyűjtjük, hogy történt-e erre a számlánkra máshonnan átutalás, ha igen akkor a szamla tábla egyenlegét módosítjuk, felviszünk egy új sort a tranzakcio táblába és töröljük a feldolgozott sort az atutalas táblából.

Az adatok kinyerésére SELECT utasítások sorozatát használom mindig ellenőrizve, hogy a megfelelő felhasználó férjen csak hozzá az adatokhoz.

A felső menüsorban a jelszó módosítása menüpontban, lehetősége van a felhasználónak egy könnyebben megjegyezhető jelszót választania, és érdemes is legalább 3-4 hetente lecserélni a jelszót a biztonság érdekében.

Ez a pass_change.php oldal szintén egy űrlap, 3 beviteli mezővel: Először is bekéri a jelenlegi jelszót, majd kétszer egymás után az új jelszót. Ezen az oldalon szintén használom az ugras() JavaScript függvényt, mivel a jelszó csak 8 karakter hosszú lehet. Az első lefutás után, megvizsgáljuk hogy minden adatbeviteli mező kitöltésre került-e, megfelelő-e az eredeti jelszó, valamint a két új jelszó mező értékének egyezőségét. Ha nem akkor kiírja a $hibalista tömb értékét. Az új jelszót eltárolás előtt egy újabb 8 karakteres generált jelsorozat (só) és az új jelszó összefűzése után az md5()-el kódolom, és az új generált jelsorozatot (sót) újra összefűzzük az md5() függvénnyel előállított jelsorozathoz. Ezáltal a jelszó kódoltan szerepel az adatbázis táblában, és az adatok áramlása során is kódoltan közlekedik a Világhálón.

Sikeres jelszó módosítás után a szamlaadatok.php oldal kerül meghívásra.

A Kilépés menüpont választásával egy sor bezáró utasítás és a munkamenet függvény megsemmisítése történik, valamint meghívásra kerül a kilepes.htm oldal, amely egy ablakbezáró utasítást hajt végre betöltődéskor.

felsomenu.htm oldalon a Kilépés menüpont:

```
<a href="kilepes.htm" target="_top" onClick="<?php
include "db.inc";
$sql_csatlakozas = @mysql_connect($hostName, $username,$password)
mysql_close($sql_csatlakozas);
$_SESSION=array();
session_destroy();?>" > Kilépés</a>
```

Kilepes.htm-ben:

<body onload="window.close()">

Folyószámlával végezhető feladatok Direct bankon keresztül

A baloldali menüsorban a következő menüpontok közül választhatunk:

- Számlainformáció
- Lekötött betét
- Átutalás

Mindegyik menüben almenük találhatók, ezek csak akkor jelennek meg, ha a főmenüsorból kiválasztunk egyet és rákattintunk. A két kép beszúrása között eltelt egy nap ezért látszik más-más dátum a két képen.

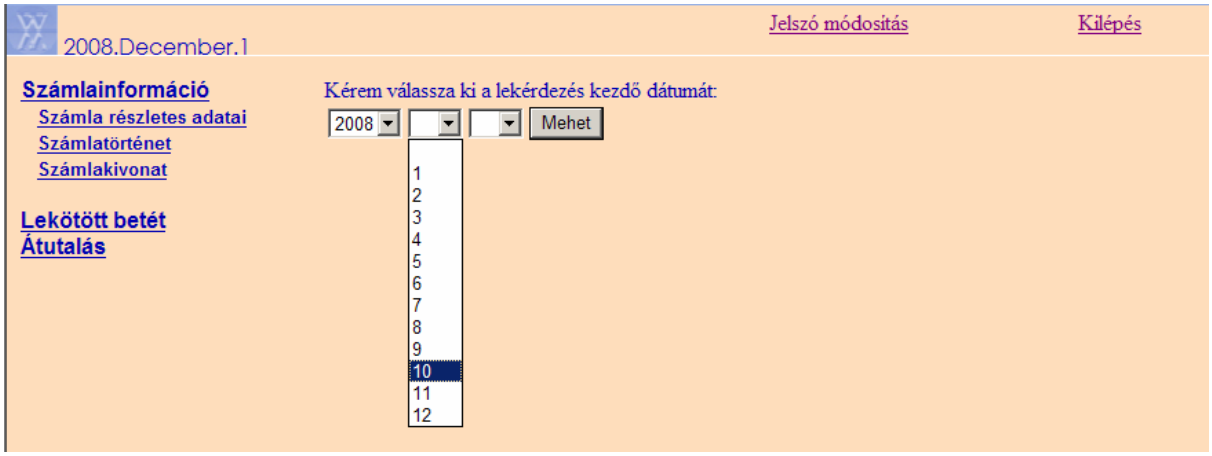
Az almenük egy újabb kattintásra a főmenün visszatérnek eredeti állapotukba. Ezt szintén JavaScript -el tudtam megvalósítani a menusor.htm oldalon.

A Számlainformáció főmenüben a következő almenük közül választhatunk:

- Számla részletes adatai
- Számlatörténet
- Számlakivonat

A Számla részletes adatai menüpont kiválasztásával a szamlareszl.php oldal hívódik meg, amellyel a szamlaadatok.php egy kibővített változatát kapjuk, ahol már információt kaphatunk a zárolásokról, előjegyzett kártyaforgalmakról, előjegyzett bruttó betéti kamatról, előjegyzett költségről, a könyvelt egyenlegről és a lekötött betéteinkről. Ez egy sor SELECT lekérdezést megvalósító MySQL utasítással tehető meg. Ugyanis a könyvelt egyenleg a tranzakcio tábla, tranzakcio_azon mező által sorba rendezett adatok közül a legutolsó rekord egyenlegével egyezik meg, ahol a számlaszám megegyezik a megadott számlaszámmal. A lekötött betétek számlainformáció pedig a lekotes tábla összegeinek összesítése, ahol a számlaszám szintén megegyezik a megadott számlaszámmal. A többi adat a szamla tábla azon rekordjának az értékével egyezik meg, ahol a felhasználó belépési azonosítója megegyezik a szamla tábla azonosito nevű mezőjének értékével.

A Számlatörténet menüpont kiválasztásával a szamlatort.php oldalon 3 lenyíló mező-vel kiválaszthatjuk a számlán történt tranzakciók (tranzakcio tábla) lekérdezésének kezdő időpontját. Kiválasztás után pedig ugyanabban az ablakban megkapjuk a lekérdezés eredményét is. A hónapok és a napok egy ciklusban kerülnek be a legördülő mező választható értékei közé. A hónapok ciklusának kiíratásával nincs semmi probléma, hiszen 1-től 12-ig mehet csak.



De a dátum hónapjainak kiválasztása után a napok számának megjelenítése számomra egy kis problémát jelentett, hogy minden hónapban csak annyi nap jelenjen meg, ahány abban a hónapban lehet. De a hónap kiválasztása után egy onChange JavaScript eseménnyel meghívom a document.urlap.submit() utasítást, amellyel elküldésre kerül az éppen kiválasztott $_POST['ho'] legördülő mező értéke. Ennek a leellenőrzése után már nem nehéz a napok számának meghatározása sem. A következő problémám a dátum értékek összehasonlítása volt ugyanis a táblázatban szereplő dátum értéket, stringként kezeli. Ezért a táblázat soraiban szereplő dátum string-et felbontottam substr() függvénnyel $sorho és $sornap változókra és (integer)-é alakítottam, ezután az összehasonlítást egy trükkel végeztem el, ugyanis a $sorho és $sornap által meghatározott dátumnak nagyobbnak kell lennie a legördülő mezőből kiválasztott $kivho és $kivnap által meghatározott dátumnál. Nem elég csak a hónapok vagy csak a napok

összehasonlítása, és ezek együttes összehasonlítása sem jár helyes eredménnyel, mert tegyük fel hogy $sorho>$kivho de nem feltétlenül fontos, hogy $sornap>$kivnap is teljesüljön, ezért a következő cselt alkalmaztam: $sorho*100+$sornap>$kivho*100+$kivnap. Ez így már helyes eredményt hozott. Ezután amelyik sor a tranzakcio táblából megfelelt ennek a feltételnek kiírásra került.

A következő almenü a Számlakivonat. Ezzel a menüponttal a kivonat.php oldal töltődik be. Itt szintén 2 legördülő menü fogad bennünket, amelyekkel a kivonaton szereplő időszak, és az arra az időszakra vonatkozó számlakivonat adatokat tartalmazó file elkészítését indítjuk el. A file megírása után egy gombbal tudjuk megjeleníteni a kivonatot, amely egy új ablakban jelenik meg. A kivonaton szerepelnek azok a fontos információk, amelyek lehetővé teszik hogy hiteles legyen könyvelés szempontjából. Szerepel rajta a bankfiók neve címe, a számlatulajdonos neve, címe, számlaszám, időszak, kivonatszám, kiállítás dátuma, a tranzakciók listája, amelyek a kiválasztott hónapban bekövetkeztek, valamint szerepel még rajta a nyitó-, és záróegyenleg és a jóváírások és terhelések összesen. Ezek után a kivonat elmenthető, kinyomtatható, stb. az ablak menüpontjainak megfelelően.

http://webbank.uw.hu/belepes/szamlainf/kivonat1.txt - Windows Internet Explorer

Fájl Szerkesztés Nézet Kedvencek Eszközök Súgó

Webbank Bankfiók
4700, Mátészalka
Délibáb út 6.

Kovács Ildikó
4700, Mátészalka Kossuth Lajos út 33/A II/6.

FOLYÓSZÁMLA KIVONAT

Számlaszám:10400020-52529120-03900025 Időszak:2008.11 Kivonatszám:011/2008
Számlakibocsátás kelte: 2008-12-01

Dátum	Könyvelés	Megjegyzés	Forgalom	Érték:
2008-11-24	2008-11-24	Számlanyitás	Bevetel	50000
2008-11-25	2008-11-25	Lekötés	Kiadas	6000
2008-11-25	2008-11-25	Lekötés	Kiadas	7000
2008-11-25	2008-11-25	Lekötés	Kiadas	5000
2008-11-25	2008-11-25	Lekötés	Kiadas	8000
2008-11-26	2008-11-26	Feltörés	Bevetel	6000
2008-11-26	2008-11-26	Feltörés	Bevetel	7000
2008-11-26	2008-11-26	Feltörés	Bevetel	5000
2008-11-26	2008-11-26	Feltörés	Bevetel	8000
2008-11-26	2008-11-26	Lekötés	Kiadas	5000
2008-11-26	2008-11-26	Lekötés	Kiadas	6000
2008-11-26	2008-11-26	Lekötés	Kiadas	7000
2008-11-26	2008-11-26	Feltörés	Bevetel	5000
2008-11-26	2008-11-26	Feltörés	Bevetel	6000
2008-11-26	2008-11-26	Feltörés	Bevetel	7000
2008-11-26	2008-11-26	Lekötés	Kiadas	8000
2008-11-26	2008-11-26	Lekötés	Kiadas	5000
2008-11-26	2008-11-26	Lekötés	Kiadas	7000
2008-11-26	2008-11-26	Feltörés	Bevetel	8000
2008-11-26	2008-11-26	Feltörés	Bevetel	5000
2008-11-26	2008-11-26	Lekötés	Kiadas	9000
2008-11-26	2008-11-26	Feltörés	Bevetel	6000
2008-11-26	2008-11-26	Feltörés	Bevetel	8000
2008-11-26	2008-11-26	Feltörés	Bevetel	9000
2008-11-26	2008-11-26	Befizetés	Bevetel	200000
2008-11-26	2008-11-26	Lekötés	Kiadas	10000
2008-11-26	2008-11-26	Átut 10400091363384704650025-ra	Kiadas	10000
2008-11-26	2008-11-26	Átutalás költsége	Kiadas	100
2008-11-26	2008-11-26	Átut 10400064890492857000025-ra	Kiadas	20000
2008-11-26	2008-11-26	Átutalás költsége	Kiadas	100
2008-11-26	2008-11-26	Lekötés	Kiadas	5000
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	30000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Lekötés	Kiadas	10000
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	5000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	5000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	5000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	5000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	5000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Átut 10400064890492857000025-ra	Kiadas	1000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100
2008-11-28	2008-11-28	Átut 12154851545154151515 1515-ra	Kiadas	1000
2008-11-28	2008-11-28	Átutalás költsége	Kiadas	100

Nyitó egyenleg:0
Jóváírások összesen:337000
Terhelések összesen:200000
Záró egyenleg:137000

A file írásához 3 segédfile-t használok, Egyben tárolom a bank adatait, egy másikban a fejlécet, és a harmadikban egy új sort, ugyanis nekem csak négyzeteket írt a file-ba a PHP fputs() függvénye \n karakterpár hatására. Ezért készítettem egy ujsor.txt file-t amelyben szerepel egy sor szóköz, majd egy új sor, ezért úgy látszik üres, de nagyon lényeges a kivonatírás szempontjából.

És mert úgy gondoltam, hogy elég sok minden egyes szót, szóközt, és új sort külön fputs-olni, ezért írtam egy fileiro($filenev,$sor,$szam,$kell,$ujsor) függvény amely sorban kéri be az irandó file nevét, a kiírandó sort, hány szóköz szerepeljen utána, kell-e újsor, és az újsor változót, amelyet egy $new változóba tettem a ujsor.txt beolvasása után. Természetesen a $filenev által átadott file-nak írásra nyitva kell állnia, mielőtt a függvényt használnánk, különben hibát eredményez.

A kivonat.php-ben lévő legördülő menük nem teljesen egyeznek meg a szamlatort.php – legördülő menüivel. Mert itt a napok nem szerepelnek. A hónapok közül pedig csak az aktuális hónap és a megelőző három hónap jelenik meg, ezzel korlátozva a választható kivonatok számát. És nem a hónapok nem számmal, hanem szövegesen jelennek meg. Itt problémát jelentett hogyha például Január (1. hó) az aktuális hónap az előtte lévő három hónap a 10,11,12. hó pedig ezek értéke nagyobb mint 1. Ezt úgy valósítottam meg, hogy az aktuális hónap értékéhez hozzáadtam 12-őt és ebből vontam le 3-at így 13-3=10 –től kezdődött a sorszámozás, és hogy a hónapok jól szerepeljenek ezután vettem a ciklus értékének 12-es maradékát. Ekkor már csak a december hónap okozott gondot, de egy feltétel vizsgálattal ezt is megoldottam. Figyelni kellett az évekre is, hiszen januárra gondolva, már két évet is meg kell jeleníteni.

A hónapok neveit egy tömbben tárolom és mindig a megfelelő indexű elemeit íratom ki a tömbnek, a tömb 0. indexű eleme, így egy üres string.

A következő főmenü a Lekötött betét, amelynek két almenüje van:

- Új lekötés
- Lekérdezés/feltörés/visszavezetés

Az Új lekötés almenüben lehetőségünk van a folyószámlánkról egy bizonyos összeget meghatározott ideig elkülöníteni, amelyre nem lesz szükségünk egy pár hónapig, így erre az elkülönített összegre nagyobb kamat jár, mintha csak a folyószámlán tartanánk a pénzünket.

Az uj.php oldalon a Számlaadatok megjelenítése után egy űrlap jelenik meg, amely bekéri az elkülöníteni kíván összeget. Kiválaszthatjuk a kamatozás módját, hogy hány hónapra tesszük be ezt az összeget, és hogy ha lejár a kiválasztott határidő akkor mi történjék a lekötött pénzünkkel, újra lekössék, vagyis folyamatos legyen, vagy visszavezessék a számlára, egyszeri lekötés esetén.

Az összegbeviteli input mezőbe nem fogad el más karaktert csak számot, valamint azt is vizsgálja, hogy van-e ennyi pénz a számlán, és hogy ez az összeg 5000 Ft-nál nem kisebb-e. Ha valamelyik feltételnek nem felel meg megjelenik a hibaüzenet(ek) listája.

Ezt az oldalt a lekotes_feld.php oldal dolgozza fel, amely sikeres lekötés után egy „A lekötést elkönyveltük” jelentés után a számla aktuális adatairól és a lekötés tranzakció időpontjáról kapunk információkat.

A Lekérdezés/feltörés/visszavezetés almenüben lehetőségünk van megnézni milyen lekötéseink vannak, valamint ha úgy kívánjuk a kiválasztott összegeket visszavezethetjük a folyószámlára. Ezt egy-egy checkbox beviteli mezővel oldottam meg, így a felhasználó látja, hogy milyen lekötései vannak, és így tudja kiválasztani azt az összeget, amelyet fel akar törni. A lekötések kiírásában szerepel, hogy mikor járna le ez a betett összeg, valamint hogy mennyi kamatot kockáztat a feltöréssel. Ugyanis, ha nem járt le a határidő természetesen elveszti a kamatot. Ha a határidő lejárt, a kamat a feltört összeggel együtt visszavezetésre kerül a folyószámlára.

A modosit.php oldal valósítja meg a számla adatok kiírását és a lekotes tábla azon sorainak kiíratását checkboxokkal, amelyeknél szamlaszam mező értéke a megadott számlaszámmal megegyezik. A mod_feld.php pedig feldolgozza a kitöltött űrlap adatait. Ez az

oldal kiírja, ha minden rendben lezajlott, hogy a „A kijelölt lekötések feltörése sikeres volt az összegek visszavezetése megtörtént!”, valamint az új számlaadatokról kapunk információt.

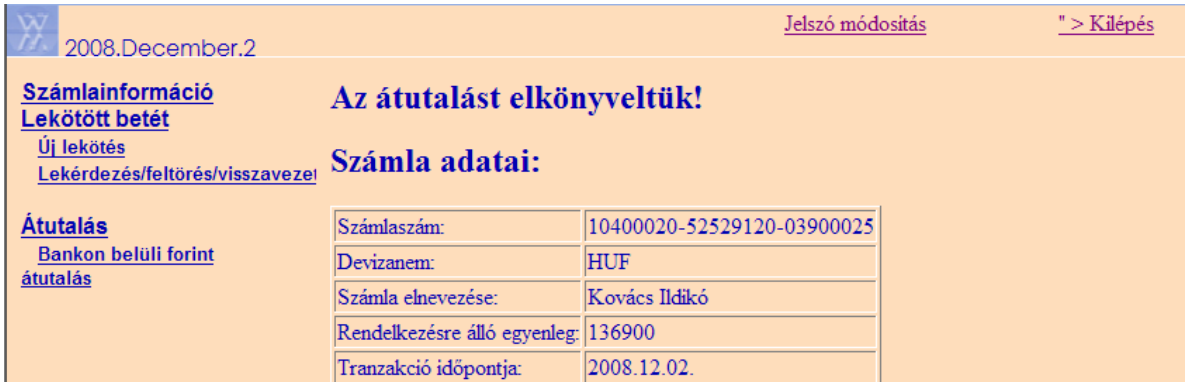
Jelen esetben egy 10000-es lekötést választottam ki feltörésre.

Több okból is másképp készítettem el ezt a két oldalt, egyrészt mert a Script futási ideje elég hosszúra sikerült volna egy oldalon, valamint az adatok frissítésére van szükség az űrlap feldolgozása után mindkét esetben, hogy a felhasználó egyből láthassa milyen változások történtek a számláján, és a form megjelenésére már nincs szükség.

Az utolsó főmenü az Átutalás, amelyben egy almenü található, a Bankon belüli forint átutalás. A bankon kívüli átutalásokat nem áll módomban megvalósítani, de egy valódi helyzetben persze más átutalásokat is képesnek kell ellátnia egy Webes banki felületnek.

Ebben a menüpontban, szintén a számlaadatok jelennek meg valamint egy űrlap az átutalás adatainak bekérésére. Meg kell adnunk az átutalás összegét, a számlaszámot, amelyre utalni szeretnénk, a számla tulajdonosának a nevét és végül az átutalás közleményét. A számlaszámot 3 beviteli mezővel adhatjuk meg, mivel kényelmesebb egyszerre csak 8 karaktert figyelemmel kísérni, mint 24-et. És a számlaszámokat általában ugyanilyen tagolással kérik mindenhol. A számlaszám mezői között a már többször alkalmazott ugras() JavaScriptet használom.

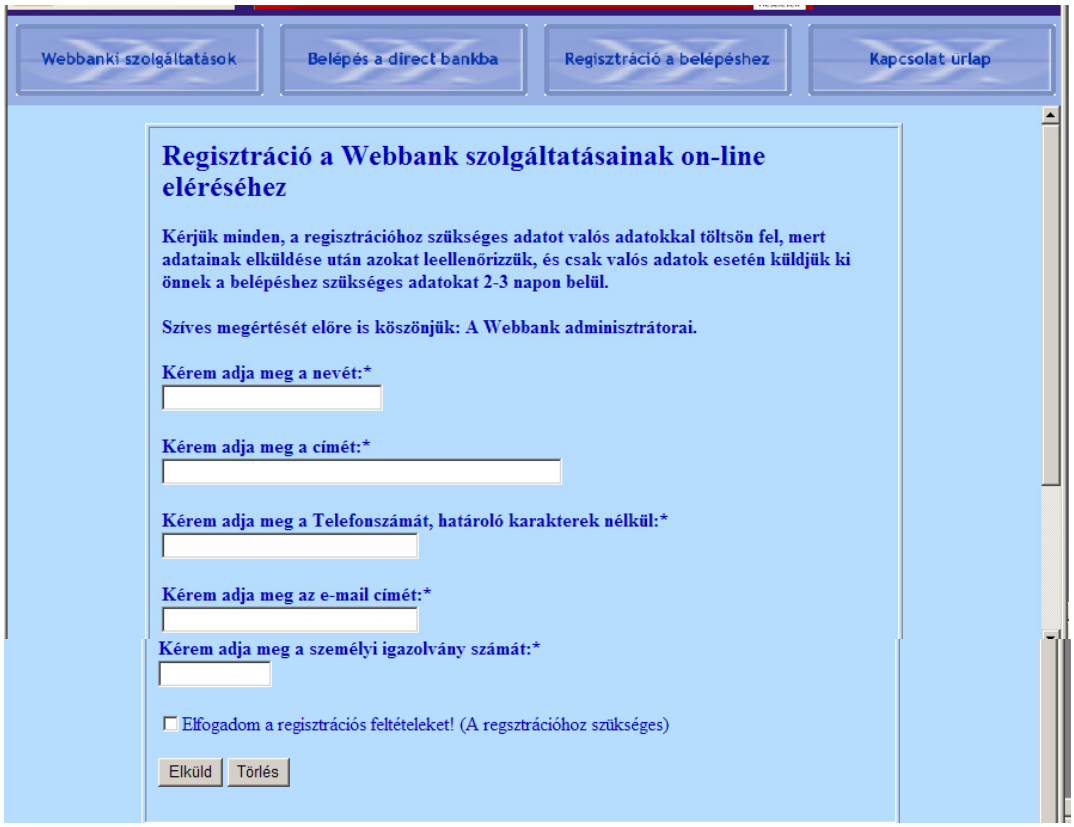
Az átutalás beviteli mezőit és a kezdeti számlaadatok kiírását az egyszeri.php valósítja meg, lekezelését pedig az egyszeri_feld.php oldal, MySQL-ben egy atutalas táblával történik az átutalás lekezelése, ahová felvisszük az adatokat leellenőrzés után. Az egyszeri_feld.php oldal megnézi, hogy az összeg szám karakterekből áll-e, és hogy van-e ennyi összeg a folyószámlán. Leellenőrzi a számlaszámot is, egyrészt hogy számokból áll-e, valamint hogy az adatbázisban létezik-e ilyen számlaszám. Ha bármelyik feltételnek nem felel meg, akkor nem végzi el az átutalást. A közleménybe bevitt karaktereket, pedig a clean() függvénnyel ellenőrizzük. Ha minden rendben, akkor beszúrunk egy új sort az atutalas táblába, megadva a számlaszámot, dátumot, összeget, és a megnevezést. A megnevezésben szerepel, hogy melyik számlaszámról történt az átutalás, valamint a közlemény rovatba felvitt karaktersorozat is itt jelenik meg. Az átutalás gombra kattintás után a számlánk aktuális állapotáról kapunk adatokat, valamint hogy az átutalás megtörtént. Természetesen mint minden banknál átutaláskor itt is egy bizonyos összeg (100 Ft) az átutalás költségeként lesz elkönyvelve.



A harmadik menüpont:

Regisztráció a belépéshez egy regisztrációs űrlapot nyit meg, ahol bekéri a felhasználó adatait. Az adatok bekérése és leellenőrzése után, beszúrunk egy új felhasználót a felhasznalo_adat nevű táblába, A tábla által előállított azonosító lesz a felhasználó adat_azon azonosítója. A felhasznalo táblába beszúrt sor után kapjuk vissza a tényleges belépési azonosítót, amely a felhasználó tábla azonosito (auto_increment) mezője. A 24 karakteres számokból álló számlaszámot, generator1.php-vel generálom, amely úgy határozza meg a számlaszámot, hogy a számlaszám eleje kötött: 104000, és a vége is kötött 00025. Ezzel a számlaszámokat egységessé, és a napjainkban használt számlaszámokhoz hasonlóvá alakítottam. Hiszen minden banknak egyedi számlaszámai vannak, amelyekről akár már első ránézésre eldönthető, hogy melyik bankhoz tartozik. A számlaszámok generálása során természetesen azt is figyelemmel kisérem, hogy van-e már ilyen számlaszám a rendszerben, a generálást addig végezzük, amíg végül új számlaszámot állítunk elő. Ezután generálunk egy 8 karakteres jelszót, a generator.php-vel. Majd szintén generálunk egy 8 karakteres (só) karaktersorozatot, amelyet a jelszó titkosításánál használunk fel.

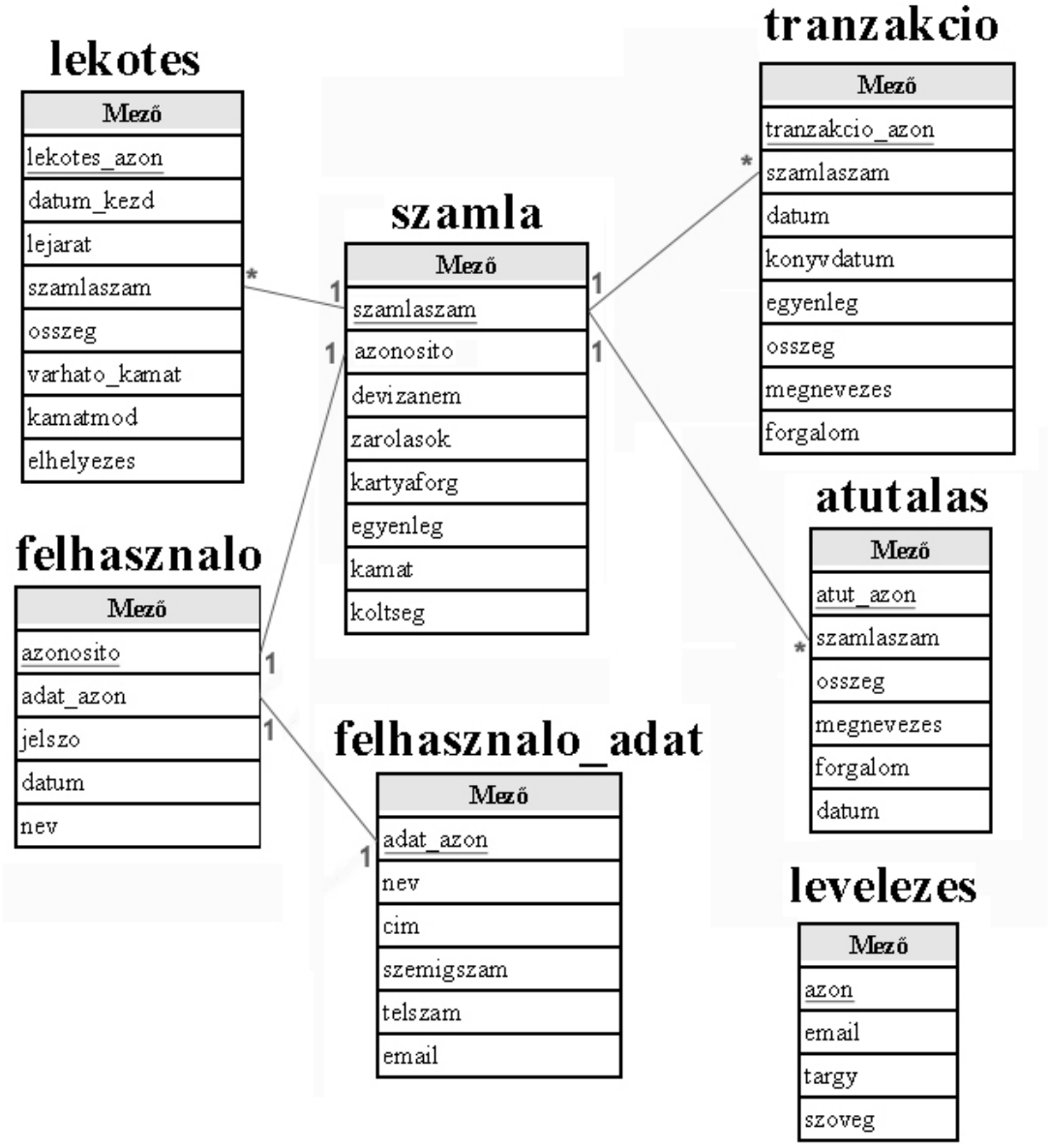
A clean() függvénnyel minden beérkező adatot megtisztítunk a támadásra felhasználható karakterektől. Ha üres valamelyik beviteli mező, vagy az adatok nem felelnek meg az előírásnak, (Pl. a telefonszám csak szám lehet, a személyi igazolvány szám pedig 2 betű karakter után 6 szám karakter következhet), addig nem történik meg a felhasználó felvitele, amíg minden hibát ki nem javítunk. A regisztrációs mező értékét is figyeljük. Ha minden rendben van beszúrunk egy sort a szamla táblába is, megadva a felhasználó azonosítóját, és a számlaszámot. A szamla tábla ezen rekordjának többi értéke, felveszi a mezőhöz hozzárendelt alapértékeket. Végül a belépéshez szükséges adatokat egy e-mail-ben elküldjük a megadott e-mail címre.


Webbanki szolgáltatások
Belépés a direct bankba
Regisztráció a belépéshez
Kapcsolat urlap
Regisztráció a Webbank szolgáltatásainak on-line eléréséhez
Kérjük minden, a regisztrációhoz szükséges adatot valós adatokkal töltsön fel, mert adatainak elküldése után azokat leellenőrizzük, és csak valós adatok esetén küldjük ki önnek a belépéshez szükséges adatokat 2-3 napon belül.
Szíves megértését előre is köszönjük: A Webbank adminisztrátorai.
Kérem adja meg a nevét:*
Kérem adja meg a címét:*
Kérem adja meg a Telefonszámát, határoló karakterek nélkül:*
Kérem adja meg az e-mail címét:*
Kérem adja meg a személyi igazolvány számát:*
Elfogadom a regisztrációs feltételeket! (A regsztrációhoz szükséges)
Elküld
Törlés

A negyedik menüpont:

Kapcsolat űrlap egy levelező űrlapot nyit meg, ahol a begépelt e-mail cím, tárgy és levél szövege egy levelezes táblában tárolódik az adatbázisunkban. Az űrlap megjelenítését, és az űrlapba bevitt értékek lekezelését ugyanaz a kapcsolat_feld.php file valósítja meg. A levelezes tábla értelemszerűen nem kapcsolódik a felhasználókhoz és a számlákhoz sem hiszen bárki küldhet levelet a Webbanknak. Azért ezt a megoldást választottam, mert nincs minden gépre alapból beállítva egy levelező rendszer. Ha lett volna lehetőségem egy másik adatbázist hoztam volna létre ennek a táblának, de ez a lehetőség nem volt elérhető a választott ingyenes tárhely szolgáltatónál.

A MySQL adatbázisban lévő adattáblák szerkezete, és kapcsolataik egymással:



Összefoglalás

Az információs forradalom, az adatok áramlásának és mennyiségének rohamos növekedése a gazdasági élet minden szereplőjét kihívás elé állítja. Dolgozatomban azt igyekeztem elérni, mely szerint az Internet a felhasználók segédeszközévé, a rohamosan fejlődő technológia pedig a felmerülő igények kielégítőjévé kell, hogy váljon. Az Internet banki szolgáltatások elérésének a lehetősége megmutatja ennek a technológiának a sokrétű felhasználását, és annak a lehetőségét, hogy mindezt hogyan állíthatjuk a fogyasztói igények minél szélesebb körű kielégítésének szolgálatába.

Az Internet magyarországi használata, a banki szolgáltatások területén, még gyerekcipőben jár, melynek több oka is van. Hazánkban a számítógéphez és az Internethez jutó emberek száma egyelőre alacsony, ami történeti és anyagi körülményekkel egyaránt indokolható, de ez növekvő mértékű, az emberek tájékozottságának növekedése, tudásigénye, és az internetes szolgáltatások egyre jobb áron való elérésének köszönhetően. Mindezek mellett, vagy ennek ellenére akinek van Internet hozzáférése, bizalmatlan a webes banki szolgáltatásokat illetően, nem meri az adatait kiszolgáltatni, ami nagyrészt köszönhető a rengeteg adathalász, és támadó szoftverek megjelenésének. Ezért alkalmazásommal igyekeztem megvalósítani mindazt az elképzelést, amellyel egy alkalmazás biztonságossá, egyszerűen kezelhetővé és felhasználóbarát szolgáltatássá tehető.

Az Internet alkalmazása a banki területen hosszabb távú stratégiai döntésként is értelmezhető, hiszen évek múlva ez a lehetőség már a versenyképesség alapvető feltétele lehet.

A webes banki szolgáltatások üzemeltetését végző bankok egyre újabb és újabb szolgáltatásokat nyújtanak a felhasználók számára ilyenek például a feltölthető mobilegyenleg, a szolgáltatói számlák (áram, víz, telefon stb.) beszedésére adható megbízások internetes kezelése, a MávStart-nál elektronikusan megvehető vonatjegyek, - amely az átutalás során szintén egy bank weboldalára irányít minket - és a növekvő igényekkel párhuzamosan egyre több és több szolgáltatást kínálnak nekünk az Online bankok. Éppen ezért az ilyen szolgáltatásokra specializálódott weboldalak állandó fejlesztése, és szolgáltatásainak bővítése alapvető követelmény.

Úgy gondolom, hogy a Webes banki szolgáltatások című weboldal olyan alapot biztosít, amelyre lehet építeni, bővíteni a lehetőségek és a felmerülő igények növekedésével párhuzamosan. Lehetőségeimhez mérten kiépítettem Webbank szolgáltatásait.

Irodalomjegyzék

Stolnicki Gyula: SQL programozóknak
ComputerBooks Budapest, 2005

Sági Gábor: Webes adatbázis-kezelés MYSQL és PHP használatával
BBS-INFO, 2005

Virrasztó Tamás: Titkosítás és adatrejtés (Biztonságos kommunikáció és algoritmikus adatvédelem) NetAcademia Kft., 2004

Nagy Péter: JavaScript 1.2 kézikönyv
Kiskapu Kft., 1998

Matt Zandstra: Tanuljuk meg a PHP 5 használatát 24 óra alatt
Kiskapu Kft., 2005

Michael Moncur: Tanuljuk meg a JavaScript használatát 24 óra alatt
Kiskapu Kft., 2002

Julie C. Meloni: Tanuljuk mega a MySQL használatát 24 óra alatt
Kiskapu Kft., 2003

A segítségként felhasznált weboldalak:

Html referencia: http://ik.inf.unideb.hu/html/referencia.html
PHP Kézikönyv: http://www.php-welt.net/handbuecher/hungarian/index.html
JavaScript példák: http://www.thomas98.hu/webmuhely.php
PHP utasítások. http://htmlinfo.polyhistor.hu/bginfos/phpshort.htm
Kliens-oldali JavaScript Referencia v1.3: http://htmlinfo.polyhistor.hu/js13ref/toc.htm
Felmerült kérdéseimre a választ megtaláltam: a http://weblabor.hu/ fórum topicjaiban