TARTU ÜLIKOOL
MATEMAATIKATEADUSKOND
Arvutiteaduse instituut
Tarkvarasüsteemide õppetool
Informaatika eriala

Aulis Sibola

VEEBIINFOSÜSTEEMID

Magistritöö

Juhendaja dots. Jaanus Pöial

Autor: . . . . . ". . ." mai 2000
Juhendaja: . . . . . ". . ." mai 2000
Õppetooli juhataja: . . . . . ". . ." mai 2000

Tartu 2000

Sisukord

Sissejuhatus

Tänapäeval ei kujuta keegi ette elu ilma veebita: me loeme ajalehti veebist, kasutame veebi kaudu e-maili teenust, maksame oma arveid internetipanga kaudu, me isegi võime vaadata veebis filme või teha kõik oma sisseostud veebi kaudu – ühesõnaga, veebist on saanud meie igapäevaelu lahutamatu osa. Eksisteerivad internetiportaalid, internetikaubamajad, iga endast lugupidav pank omab internetipanka, veeb on täis suuri andmebaase ja otsingumootoreid; pea iga väiksemgi firma omab isiklikku veebilehte, suuremad firmad on loonud oma veebist ka interaktiivse keskkonna, kus igaühel on võimalik end registreerida ja keskkonda omale meelepärasemaks muuta, firma tooteid otse veebist tellida jne. Veeb on saanud üheks osaks viimase aja põhilises ärisuunas – e-kommertsis ehk e-äris.

Kõik need veebi süsteemid on üles ehitatud kindla arhitektuuri järgi: kuskil asub mingi andmebaas, milles olevaid andmeid saab veebis näha ja vajadusel ka muuta. Mis on selle kõige taga? Milliseid vahendeid selleks kõigeks kasutatakse ja mida oleks hea kasutada? Neile küsimustele üritabki antud töö vastust anda.

Veeb oli algselt lihtne staatiliste linkide süsteem, mis võimaldas luua küllaltki ilusat staatilist keskkonda. Esimesteks infosüsteemideks veebis võib pidada otsingumotoreid, mis hakkasid koguma andmeid kogu selle linkide süsteemi kohta ja üritasid seda ka indekseerida. Seejärel ilmusid juba erinevate andmebaaside veebiliidesed, mille puhul veeb oli üheks paljudest väljunditest. Tänapäeval on aga praktiliselt suvalist infosüsteemi võimalik luua veebipõhisena.

Käesolevas töös vaadeldakse põhilisi veebiinfosüsteemi komponente: veebiservereid, programmeerimist veebis, andmebaasidega suhtlemist – seda kõike võttes aluseks Microsoft Windows keskkonna. Kuna aluseks on võetud Windows keskkond, peatutakse ka firma Microsoft toodetel kõige pikemalt.

Töö esimene peatükk tutvustab Windows keskkonnas enamkasutatavaid veebiservereid ja annab ülevaate veebi programmeerimisest.

Teine peatükk peatub pikemalt ühel populaarsel veebi programmeerimise vahendil – serveripoolsed skriptid. Tutvustatakse kolme skriptikeskkonda: *Microsoft Active Server Pages*, *Allaire ColdFusion* ja *PHP*.

Kolmas peatükk annab ülevaate andmebaasidega suhtlemise standardvahenditest Microsoft Windows keskkonnas: *OLE DB*, *ActiveX Data Objects* (ADO) ja *Open Database Connectivity* (ODBC).

Neljas peatükk vaatleb veebi kui mitmekihilist infosüsteemi ja loetleb veebiinfosüsteemi loomiseks vajalikud vahendid ning nende head ja vead.

Viiendas peatükis antakse ülevaade Eesti veebi olukorrast 2000 aasta aprilli alguses.

Tööl on kaks lisa, millest esimeses on toodud *Allaire ColdFusion*i kasutusvõimaluste maatriks ja teine annab ülevaate hetkel eksisteerivatest OLE DB toodetest.

Kõik töös viidatud nimed ja nimetused, mille kohta on teada, et nad on kas registreeritud või registreerimata kaubamärgid, kuuluvad nende omanikele.

1 Programmeerimine veebiserveris

Veeb on oma arengus läbi teinud mitmeid etappe. Algselt oli see lihtne staatiline linkide süsteem; seejärel polnud staatiline süsteem enam mõttekas, kuna infot, mis veebi paigutati, oli liiga palju ja selle haldamine staatilisena osutus liiga töömahukaks. Selleks loodi võimalus veebi dünaamilisena hoida.

Esialgu hakati lihtsalt looma programme, mida sai teatud liideste abil veebi kaudu käivitada – liidest, mille abil seda tehti nimetatakse *Common Gateway Interface* (CGI). Selle liidese abil sai luua ka interaktiivseid süsteeme, kus peale info vaatamise oli kasutajal võimalus mingit infot ka serverile saata. Kui aga andmemahud veebis suurenesid, leiti, et neid andmeid on kasulik hoida andebaasides. Loomulikult oli vaja ka programme, mis neid andmeid veebis näidata suudaksid; esialgu kõlbasid CGI programmid ka selleks, kuid CGI probleemiks osutus tema aeglus – nimelt peab iga veebi poole pöördumisega CGI programmi uuesti laadima.

Seega tekkis andmemahtude suurenedes vajadus millegi kiirema järele. Selleks kiiremaks vahendiks osutusid veebiserverid ise – see tähendab, et hakati tegema programme, mis töötasid koos veebiserveriga ehk olid veebiserveri sisse integreeritud. Selleks otstarbeks loodi mitmeid liideseid nagu näiteks *Internet Server Application Programming Interface* (ISAPI) firma Microsoft poolt ja *Netscape Server Application Programming Interface* (NSAPI) firma Netscape poolt. Neid programme nimetatakse ka veebiserveri laiendusteks – nad laiendavad veebiserverit luues serverile lisavõimalusi. Nimetatud rakendused on väga kiired – programmi alglaadimine toimub kas koos veebiserveriga või siis esimesel pöördumisel konkreetse rakenduse poolt. Seega avaneb võimalus luua väga kiireid süsteeme.

Seejärel aga leiti, et need loodud programmid on küll väga head ja kiired, aga neid on küllaltki raske hallata – näiteks ka kõige väiksema muutuse korral programmis võib tekkida vajadus veebiserveri seiskamiseks. Ka sellele probleemile leiti lahendus - loodi serveripoolsed skriptid. Kuigi ka näiteks CGI programm võib olla skript

(UNIX käsukeele skript, Perl skript jne) ei ole siin mõeldud seda – serveripoolsete skriptide korral töötab veebiserverisse integreeritud spetsiaalne programm, mis interpreteerib neid skripte. Sel korral küll veebirakendus kannatab teatava kiiruse languse all võrreldes näiteks ISAPI rakendustega, aga süsteemi haldamine on see-eest kiirem ja mugavam. Serveripoolsed skriptid võetakse vaatluse alla järgmises peatükis.

1.1 Veebiserverid

Igasuguse veebilahenduse üheks peamiseks osaks on siiski ennekõike veebiserver. Microsoft Windows keskkonnas kasutatakse peamiselt kolme veebiserverit: *Internet Information Server*, *Apache* veebiserver ja *iPlanet* veebiserver (varem tuntud kui Netscape veebiserver).

1.1.1 IIS

Internet Information Server (IIS) on Microsofti poolt välja töötatud paljude lisavõimalustega server. Lisaks veebiserverile on IIS kooseisus võimalik kasutada FTP serverit, mailiserverit ja uudisteserverit. Täieliku ülevaate saamiseks tasub vaadata firma Microsoft kodulehele (nt allikas [1] ja [2]).

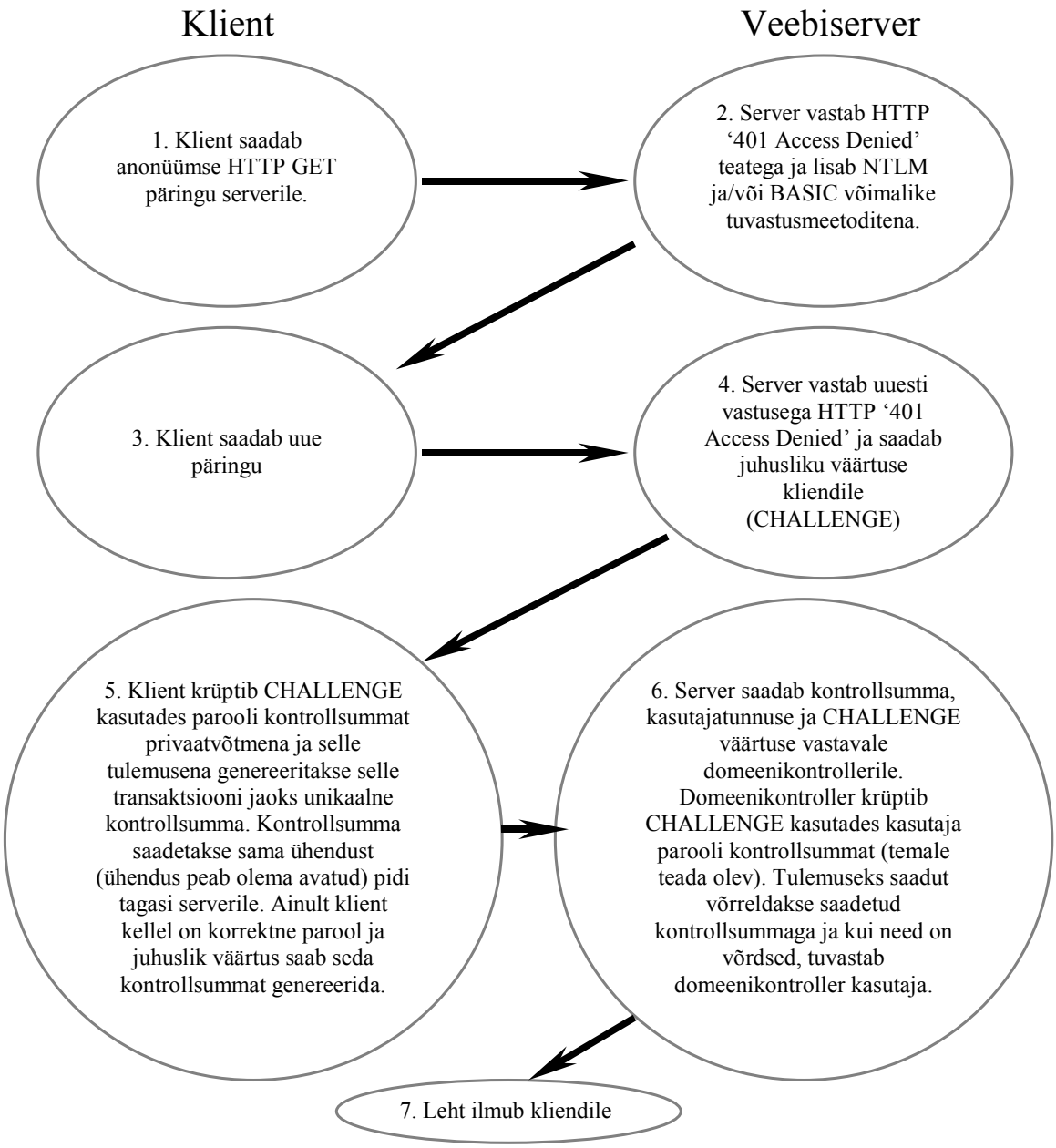
Kuna IIS on põhjalikult integreeritud Windows NT-ga, siis omab ta head ülevaadet operatsioonisüsteemist ning juurdepääsu Windowsi omadustele. Näiteks sisaldab IIS transaktsioonidel põhinevat serveripoolsed skriptivahendit *Active Server Pages*. IIS põhiomadused sisaldavad:

- HTTP/1.1 (Hypertext Transfer Protocol) standardi toetus. See tähendab, et IIS sisaldab ühenduste alalhoidmist, konveiersüsteemi ja dokumentide sisemist puhverdamist.
- Mitme veebiserveri korraga haldus.
- Veebi-, Windows- ja käsurea-põhine administreerimine.
- ISAPI laienduste ja filtrite toetus.
- *Active Server Pages* (ASP) skriptivahendit.
- Spetsiaalset teadete haldussüsteemi. ASP rakendused saavad välistele programmidele teateid saata kasutades spetsiaalset *Microsoft Message Queue*

serverit – seoses sellega saab ASP kiirelt tööd jätkata, lastes haldussüsteemil teadetega tegeleda.

- Vigade vältimine. IIS võimaldab määrata teatud osa veebist töötama, kui eraldiseisva rakenduse – see võimaldab ülejäänud veebil tööd jätkata, kui nimetatud osa põhjustab vea.
- Automaatne vigade parandus. Kui mõni veebirakendus tekitab vea ja seiskub, käivitatakse see automaatselt järgmise pöördumise korral.
- Kohandatav logi. ISAPI rakendustel on võimalus lisada logifaili oma teateid.
- Komponentide toetus. IIS lubab kasutada suvalisi serveris asuvaid komponente; IIS haldab nende komponentide laadimist ja välja laadimist.
- *ActiveX Data Objects* toetus.
- Sisseehitatud sertifikaadiserver.
- Integreeritud indekseerimine ja otsing võimaldab indekseerida kogu veebiserverit. IIS oskab lisaks tavalistele hüperteksti- (*Hypertext Markup Language* – HTML) ja tekstifailidele indekseerida ka *Microsoft Office* ja näiteks PDF dokumente.
- Väljundi aegumine võimaldab määrata veebiserveri poolt saadetud infole aegumistähtaegu.
- Võimalus lisada kohandatud dokumendi jaluseid (*footer*) määratud HTML lehtedele.
- Võimalus luua enda koostatud HTTP veateateid.
- HTTP lisapäiste lisamine väljundile.
- Automaatsed ümbersuunamised.
- Korralik dokumentatsioon.

IIS võimaldab kasutada Windows NT turvaelemente nagu failiõiguste määramine ja doomeenikontrolleri kasutamine kasutajate tuvastamisel – seega pole vaja veebiserveri jaoks eraldi kasutajaid luua, vaid selleks saab kasutada näiteks Windows NT kasutajagruppe ning nende õigusi. IIS võimaldab kasutaja tuvastamisel kasutada Windows NT tuvastussüsteemi NTLM (nimi tuleneb *Windows NT LAN Manager* nimest, kuna seda kasutati Windowsi esimestes võrgusüsteemides). Windows NT kasutaja tuvastamise protsessi nimetatakse *Windows NT Challenge/Response*. NTLM skeemist annab ülevaate järgmine joonis:



Joonis 1.1 Windows NT Challenge/Response kasutajatuvastuse skeem

NTLM kasutajatuvastust toetab näiteks *Microsoft Internet Explorer*.

1.1.2 Apache server

Apache veebiserver on hetkel maailma populaarseim veebiserver. Seda põhiliselt tänu tema vabale kasutusele ja algkoodi avalikkusele. Igaühel on võimalus serverit täpselt selliseks kujundada nagu ta ise seda soovib. Vaatamata asjaolule, et Windows keskkonnas ei paku Apache võibolla kõiki võimalusi mida ta pakub UNIX keskkonnas, on ta seal siiski IIS järel teisel kohal.

Apache veebiserveri kohta leiab piisavalt palju informatsiooni serveri korduma kippuvate küsimuste lehelt (vt allikas [3]).

Apache serverit populaarseks tegevad omadused on:

- Multiplatvorm. Apache serverit saab edukalt kasutada suvalisel platvormil suvalises operatsioonisüsteemis.
- Toetab täielikult HTTP/1.1 protokolli.
- Võimaldab väga võimsat konfigureerimist ja laiendamist.
- Võimaldab lisada mooduleid laiendamaks serverit; selleks saab kasutada spetsiaalset liidest Apache module *Appilcation Programming Interface* (Apache API).
- Omab toetust nii Windows CGIle kui ka ISAPI liidesele (läbi vastava mooduli).
- Kogu algkood on avalik ja litsentsitingimused ei piira selles muudatuste tegemist.
- Võimaldab piisavat kasutajatuvastuse ja õiguste mehhanismi.
- Kohandatavad serveripoolsed veateated.
- Mitme veebisaidi ühes serveris hoidmine.
- Omab väga võimsat logi süsteemi, mis on vastavalt vajadustele kohandatav.
- Areneb pidevalt edasi.

1.1.3 iPlanet veebiserver

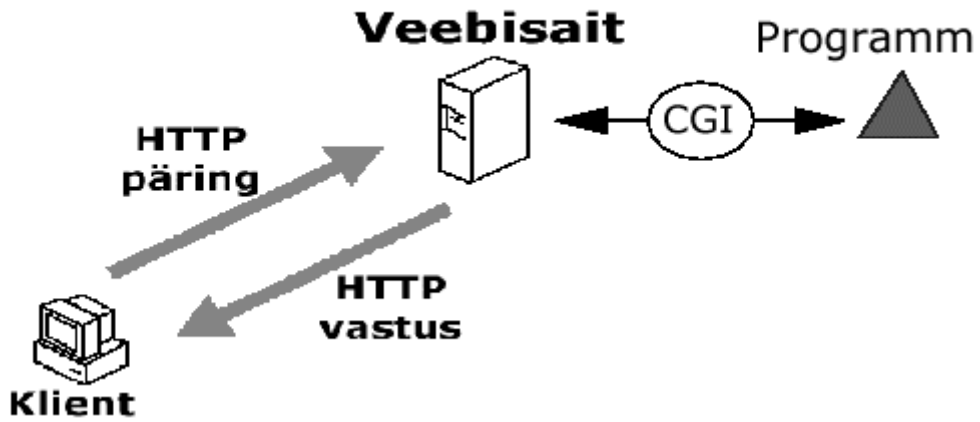
iPlanet veebiserver oli esialgselt välja töötatud firma Netscape poolt, hiljem läks serveri arendus Sun-Netscape alliansi kätte ja alates serveri neljandast versioonist kannab see nime iPlanet. Netscape ja iPlanet veebiserverid on hetkel maailmas populaarsuselt kolmandal kohal (seda nii UNIX kui ka Windows keskkonnas), seda vaatamata asjaolule, et serverid ei ole vabavara, vaid neid tuleb (küllaltki suure rahasumma eest) osta. iPlanet veebiserverist ülevaate saamiseks on soovitatav vaadata allikat [4].

iPlanet veebiserveri (versioon 4.01) põhilised omadused on:

- Töötab nii UNIX kui ka Windows platvormil.
- Väga hea server loomaks e-äri süsteeme.
- Oskus serveri koormust balansseerida kasutades mitme protsessi ja lõimesüsteemi (*mutlithreading*).
- Dünaamilised logi loomise võimalused.
- Toetab erinevaid Java rakendusi nagu *Java Servlets*, *JavaServer Pages* ja *Java Virtual Machine*.
- Toetab täielikult HTTP/1.1 protokolli.
- *Netscape Appilcation Programming Interface* (NSAPI) toetus.
- NSAPI rakenduste puhverdamise võimalused.
- Piisav kasutajate tuvastuse ja õiguste süsteem.
- Veebipõhine administreerimine.

1.2 CGI programmid

Common Gateway Interface (CGI) on lihtne liides, mis võimaldab veebiserveris käivitada väliseid programme – see tähendab, et programmid võivad väljastada HTML-i, pilte jne ja veebiserver vaatab programmide töö tulemusele nagu oleks see tavaline staatiline väljund. Seega võimaldab CGI genereerida küllaltki lihtsalt dünaamilist väljundit. Üldistatult näeb CGI töö välja järgmiselt:



Joonis 1.2 CGI programmide kasutamine veebis

CGI spetsifikatsioon on põhjalikult kirjeldatud allikas [5], CGI programmeerimisest annavad ülevaate allikad [6] ja [7].

1.2.1 Milleks CGId kasutatakse

CGId võib kasutada paljude erinevate ülesannete täitmiseks ja pikka aega oli CGI ainus vahend dünaamiliste veebilahenduste loomiseks. CGI kasutamist ülesannete kaupa võib jaotada kolmeks: lihtsad, keskmised ja keerulised ülesanded.

Lihtsad ülesanded on sellised, mille koostamiseks kulutatakse küllaltki vähe aega ja vaeva nagu näiteks veebilehtede külastuste loendur, serveri kellaaja näitamine jm. Need on sellised ülesanded, kus programmeerimist ega CGI tundmist pole praktiliselt vaja. Sellised lihtsad ülesanded osutuvad ka piisavalt kiireteks lahendatuna CGIs.

Keskmiste ülesannete lahendamiseks tuleb juba natuke vaeva näha, need ei valmi lihtsalt paaritunnise töö tulemusena, nende lahendamiseks tuleb juba tunda ka CGId. Sellised ülesanded on näiteks CGI programmid, mis moodustavad kogu HTML lehe ja animatsioonid (nt pildil klikkides see suureneb või väheneb). Neid ülesandeid on võimalik ka näiteks Javaga lahendada, kuid CGI osutub tihtipeale kiiremaks ja paljud brauserid ei toeta Javat või on see neis keelatud.

Keerulised ülesanded nõuavad juba ülesande väga täpset püstitust ja korralikku realiseerimist. Keerulisteks ülesanneteks võib pidada näiteks andmebaasisüsteeme, otsingumootoreid ja ka mitmete dünaamiliste HTML lehtede kooslust. Sellised ülesanded moodustavad tavaliselt lõpuks ka ühtse terviku – infosüsteemi.

1.2.2 Mida CGI programm peab oskama

CGI programme võib kirjutada suvalises programmeerimiskeeles, peaasi, et need programmid töötaksid veebiserveris ja vastaksid teatud nõuetele. Enamlevinud programmeerimiskeeled CGI jaoks on C, C++ ja Perl, kuid kasutatakse ka näiteks keeli nagu *Tcl*, *Visual Basic*, *Applescript* jne. Nõuded, mis seatakse CGI programmidele, võib üldistatult kokku võtta järgmiselt:

- Programm peab oskama kliendi poolt saadetud andmeid lugeda – seda muidugi juhul, kui programmi eesmärgiks on kliendi poolt andmeid saada.
- Programm peab oskama väljastada standarditele vastavat väljundit.

Kliendi poolt **andmete lugemisel** tuleb arvestada asjaoluga, et klient võib andmeid veebiserverile saata (andmete saatmise meetod on meetod, millega brauser pärib veebiserverilt programmi) kolmel erineval moel:

- *GET* meetodi korral saadab klient kogu vormi info URLiga (*Uniform Resource Locator*) koos (lehe aadressi järel küsimärgiga eraldatult). CGI paigutab saadetava info keskkonnamuutujasse QUERY_STRING. Programm peab olema võimeline selles muutujas sisalduvast infost välja lugema endale vajaliku info.
- *POST* meetodi korral saadab klient kogu vormi info eraldi osana päringu lõpus – programm peab lugema seda infot standardsisendist. Selle meetodi korral tuleb meeles pidada, et server ei lõpeta saadetavat infot tavalise

lõpusümboliga, vaid selle asemel väärtustatakse CGI keskkonnamuutuja CONTENT_LENGTH, mis sisaldab saadetava info pikkust. POST meetodit on soovitatav kasutada, kui saadetav info on liiga pikk (tavaliselt on piirang seatud 1024 sümboli peale).

- *HEAD* meetod sarnaneb GET meetodiga, ainult et selle meetodi korral ootab klient tagasi ainult HTTP päiseid, andmeid saata pole vaja.

Saadetav info on teataval viisil kodeeritud. Nimelt on kõik spetsiaalsümbolid (k.a. täpitähed) kodeeritud kuueteistkümnendkoodis ja erinevate vormi muutujate (ning nende väärtuste) eraldajaks on kasutatud &-märki.

CGI programm saadab **väljundi** standardväljundisse; veebiserver võib selle saata otse kliendile või siis saadetud infot omakorda töödelda (nt kui programm suunab kliendi uuele asukohale). CGI programm peab enne tavalise väljundi saatmist seadma ka kliendile saadetavad HTTP päised nagu näiteks:

- *Content-Type* päisega annab programm teada väljundi MIME tüübi. Näiteks *text/html* tähendab tavalist HTML teksti, *image/gif* aga spetsiaalformaadis pilti.
- *Location* päisega teatab programm, et soovib klienti suunata uuele asukohale. Kui suunatav koht asub samas veebiserveris, võib server teha seda vahetult, ilma kliendi poole pöördumata, kui aga antud on täispikk URL, siis saadetakse vastus kliendile ja kliendi programm peab suunamisega ise hakkama saama.
- *Status* päisega teatab server kliendile töö staatust. Järgnev tabel annab ülevaate põhilistest staatuskoodidest ja nende tähendustest:

Staatuse kood	Staatuse tekst	Tähendus
200	OK	Päring õnnestus ilma probleemideta
202	Accepted	Päring võeti vastu, kuid on veel töötluses
301	Moved	Antud leht on üle viidud teise kohta
302	Found	Küsitud lehte ei leitud sellest kohast kust teda küsiti, kuid see leht leiti mujalt
400	Bad Request	HTTP päringu süntaks ei olnud korrektne
401	Unauthorized	Küsitud leht nõuab kasutajatuvastust

403	Forbidden	Serveril puuduvad vastavad õigused küsitud lehele
404	Not Found	Küsitud lehte ei leitud
500	Server Error	Serveris tekkis viga
502	Service Overloaded	Server on liiga koormatud ja ei suuda hetkel teenindada

Tabel 1.1 HTTP staatuskoodide tähendused

1.2.3 Keskkonnamuutujad

Keskkonnamuutujad on muutujad, mis väärtustatakse kui veebiserver käivitab mõne CGI programmi. Nende muutujatega antakse programmile infot kliendi ja serveri kohta. Tähtsamad muutujad, mis väärtustatakse, on järgmised:

- *AUTH_TYPE* määrab kasutajatuvastuse tüübi.
- *CONTENT_LENGTH* määrab kliendi poolt saadetud andmete pikkuse baitides. Muutuja on määratud ainult juhul, kui saatmine toimus POST meetodit kasutades.
- *CONTENT_TYPE* määrab kliendi poolt POST meetodiga saadetud vormis määratud MIME tüübi; tavaliselt on selleks *application/x-www-form-urlencoded*.
- *PATH_INFO* annab lisainfot URLi kohta.
- *PATH_TRANSLATED* määrab CGI programmi füüsilise asukoha serveris.
- *QUERY_STRING* sisaldab GET meetodiga saadetud infot.
- *REMOTE_ADDR* sisaldab kliendi arvuti IP aadressi.
- *REMOTE_USER* sisaldab edukalt tuvastatud kasutajanime.
- *REQUEST_METHOD* määrab päringu tüübi, millega programmi poole pöörduti (GET, POST või HEAD).
- *SCRIPT_NAME* määrab programmi virtuaalse tee (nt `/cgi-bin/Test.cgi`).
- *SERVER_NAME* sisaldab serveri nime või IP aadressi.
- *SERVER_PORT* sisaldab veebiserveri pordi numbrit.
- *SERVER_PROTOCOL* sisaldab veebiserveri poolt kasutatava protokolli.

- *SERVER_SOFTWARE* sisaldab veebiserveri tarkvara nimetust.

- *HTTP_ACCEPT* muutujaga annab klient teada, millistest MIME tüüpidest ta suudab aru saada.
- *HTTP_REFERER* muutujaga teatab klient, milliselt URLilt ta programmile tuli.
- *HTTP_USER_AGENT* muutujaga annab klient teada klientprogrammi nime.

1.3 Veebiserveri laiendused

CGI programmid teevad oma tööd hästi: koguvad veebist infot, töötlevad seda ja genereerivad väljundi. Samas on neil üks suur miinus: iga kord kui programmi poole pöördutakse, laetakse see programm uuesti serveri mällu. Kui CGI programm on skript (nt Perli skript), siis peab interpretaator iga kord skripti lisaks ka kompileerima; kui ühte CGI programmi vaatab korraga viis inimest, on serveris korraga mälus viis eksemplari antud programmist, kui kliente on tuhandeid, siis... ja seda kõike näiteks selleks, et väljastada lehele ühte numbrit (külastatavuse loendur). See ei ole just kõige parem väljavaade.

Serveri laiendused nagu *Internet Server Application Programming Interface* (ISAPI) ja *Netscape Server Application Programming Interface* (NSAPI) loovad uued võimalused veebi programmeerimiseks. Mainitud liidesed võimaldavad programmeerijatel laiendada veebiservereid ja loovad sellega neile seniolematud võimalused veebi programmeerimiseks.

1.3.1 ISAPI

Microsoft Windows keskkonnas on väga populaarne kasutada ISAPIt; seda põhiliselt tänu tõsiasjale, et Microsoft on üks ISAPI loojatest ja ISAPI töötab väga lähedaselt koos IISga. ISAPI võimaldab võimsaid kõrgtasemel arendatud dünaamilisi veebisaite – need lahendused on võimelised toime tulema paljude päringutega ilma veebiserverit oluliselt koormamata. ISAPI ei ole lihtsalt parem CGI, ta erineb põhimõtteliselt CGIst ja on loodud lahendamaks CGI probleeme.

ISAPIst antakse põhjalik ülevaade allikates [6] ja [8], programmeerimise seisukohalt on kasulik vaadata materjale allikas [9].

ISAPI põhilised eelised CGI ees on mastaapsus (*scalability*), kiirem jõudlus (*faster performance*) ja laiendatavus (*extensibility*).

Mastaapsus. ISAPI võimaldab luua veebikeskkondi, mis suudavad teenindada ühest kuni tuhandete samaaegsete ühendusteni ilma seejuures lisaressursse nõudmata. CGI korral tuleb iga uue ühenduse jaoks luua uus protsess iga CGI programmi jaoks. ISAPI korral loob veebiserver initsialiseerimisprotsessi käigus lõimetsooni ja iga uue pöördumise korral ISAPI rakenduse poole luuakse rakendusest uus lõim. Lõimetöötlus on vähem mälunõudlik ja lõime loomine toimub palju kiiremini kui protsessi loomine. CGI korral peab veebiserver programmi töö lõppedes tegema teatavat puhastustööd, ISAPI korral annab veebiserver juhtimise ISAPI rakendusele ja see teeb kogu töö (hoolitseb ka töö korrektse lõpetamise eest) ja vabastab lõime.

Kiirem jõudlus. Nagu juba mainitud, luuakse ISAPI rakendused lõimedes: see annab lisaks ressurssidele juurde ka kiiruses – rakendused töötavad kiiremini. Lisaks ei tööta ISAPI rakendused standardsisendi ja –väljundiga, vaid saadavad andmeid otse serverile, mis saadab need omakorda kliendile.

Laiendatavus. ISAPI võimaldab veebiserverit ka otseselt laiendada – muuta saab ka veebiserveri sisemist käitumist. Selliseid ISAPI rakendusi nimetatakse ISAPI filtriteks. Pea igale sammule, kui veebiserver suhtleb kliendiga, saab ISAPI filtri vahendajaks panna.

ISAPI rakendused on tavaliselt realiseeritud teekidena (*Dynamik-Link Libraries* – DLL). Teeke võib olla kahte tüüpi: ISAPI laiendused (*extensions*) on sarnased CGI programmidega, ISAPI filtrid (*filters*) võimaldavad veebiserverit laiendada väga madalal tasemel.

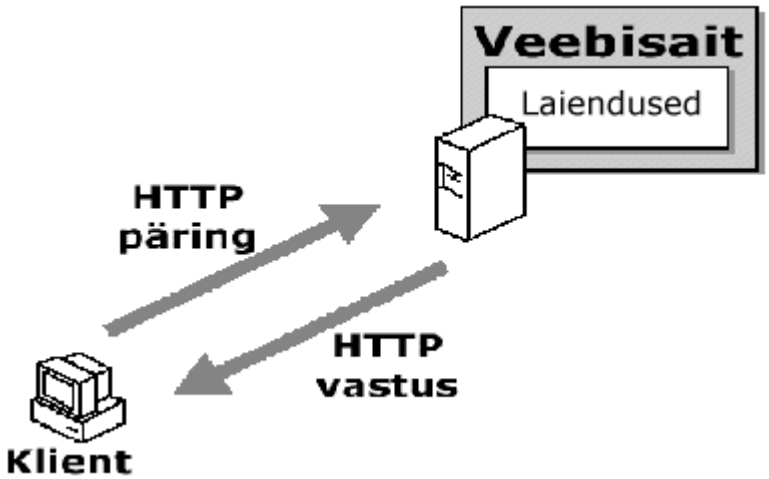
1.3.2 ISAPI laiendused

Laiendused on nagu CGI programmidki, ainult et tavaliselt on need realiseeritud DLLidena. Nad laetakse juhul, kui mõni klient pöördub esimest korda laienduse poole ja jäävad tavaliselt laetuks kuni veebiserveri seiskamiseni. Veebiserveril on alati võimalus laiendust mälust välja laadida (näiteks ressursside vabastamise eesmärgil); laiendused töötavad samas mälupiirkonnas kus veebiservergi. ISAPI rakendused töötlevad andmeid kasutades selleks spetsiaalset andmete kontrollblokki (*Extension Control Blocks* – ECB) – see kiirendab rakenduste tööd tunduvalt, kuna seda blokki haldab server ja selle korral puudub standardsisendi ja -väljundiga suhtlemine.

Kui toimub pöördumine laienduse poole, toimuvad serveris järgmised sammud:

- server saab päringu, mis nõuab laienduse kasutamist;
- server kontrollib, kas laiendus on juba laetud ja kui seda pole veel tehtud, laiendus laetakse;
- laiendus saab andmeid andmete kontrollbloki kaudu;
- andmed töödeldakse;
- laiendus saadab tulemuse kliendile;
- server katkestab laienduse töö ja laeb selle vastavalt vajadusele ka mälust välja.

Laienduste tööd veebiserveriga iseloomustab järgmine pilt:

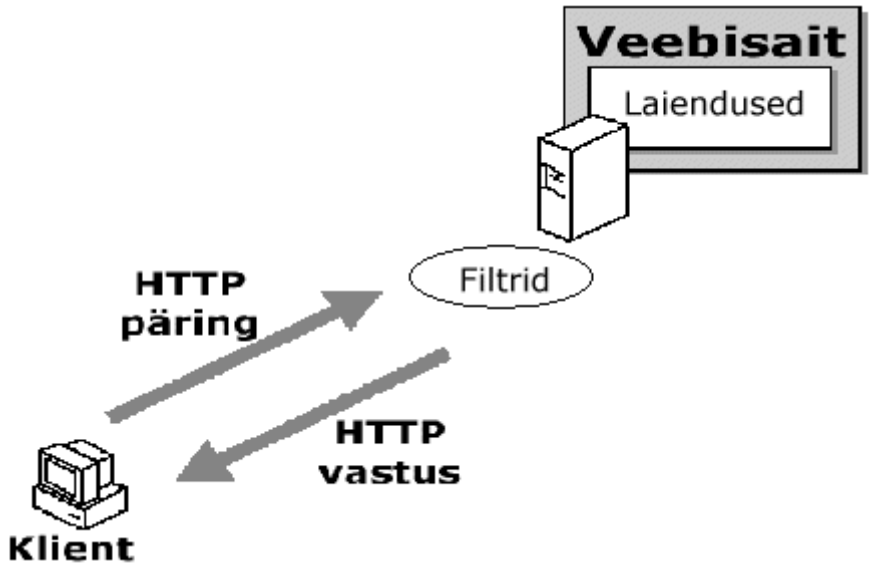


Joonis 1.3 ISAPI laienduste töö veebiserveris

ISAPI laiendustel põhinevad tavaliselt ka serveripoolsete skriptide interpretaatorid nagu näiteks *Active Server Pages*, *ColdFusion* ja *PHP*.

1.3.3 ISAPI filtrid

Filtrid pakuvad funktsionaalsust, mida CGIga pole võimalik saavutada. Filtrid sekkuvad HTTP ühendustesse nende ühenduste väga madalal tasemel – see võimaldab muuta veebiserveri standardset ühenduste haldamist. Filtrid laetakse koos veebiserveriga ja need jäävad laetuks kuni serveri seiskamiseni. Filtrid võimaldavad näiteks kontrollida päringute tegemist veebiserverile ja neile vastamist. Filtri eesmärk on filtreerida mingit infot, seega kui veebiserver saab HTTP päringu kontrollitakse see kõigepealt filtri poolt ja kui filter leiab infot, mida ta peab töötlema, see töödeldakse ja alles seejärel saab server oma tööd jätkata.
Filtrite tööd veebiserveris võib illustreerida järgmise pildiga:



Joonis 1.4 ISAPI filtrite sekkumine veebiserveri töösse

ISAPI filtreid saab kasutada järgmistel eesmärkidel:

- Kohandatud kasutajatuvastus.
- Info pakkimine.
- Info krüptimine.
- Logi pidamine.
- Päringute analüüs.

1.3.4 NSAPI

Netscape Server Application Programming Interface (NSAPI) on firma Netscape poolt välja töötatud liides Netscape veebiserverite laiendamiseks. NSAPI võimaldab lisada serverile funktsionaalsust, nagu seda teeb ka CGI, kuid samas lubab NSAPI muuta veebiserveri funktsionaalsust ja seda väga madalal tasemel.

NSAPIst saab ülevaate allikast [6], programmeerimise poole pealt tasub vaadata allikat [10].

NSAPI põhilisteks eelisteks peetakse:

- **Jõudlus**. NSAPI funktsioonid on palju kiiremad kui CGI programmid. Seda just seetõttu, et NSAPI on väga tihedalt integreeritud veebiserveriga.
- **Protsessi jagamine**. NSAPI funktsioonid töötavad samas protsessiruumise veebiserveri protsessiga ja jagavad ühtseid ressursse.
- **Andmetele juurdepääs**. NSAPI funktsioonidel on juurdepääs palju rohkematele andmetele kui näiteks CGI programmidel. Seda just seetõttu, et nad töötavad veebiserveri sees ja teevad ka osaliselt veebiserveri tööd.

NSAPI rakendusi saab kasutada näiteks järgmistel eesmärkidel:

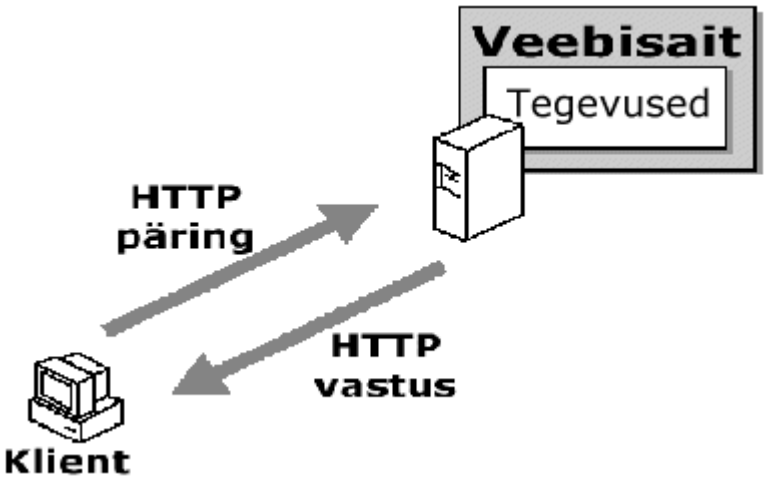
- Kohandatud kasutajatuvastus.
- Logi pidamine.
- Turvalisuse suurendamine.
- Aruandlus.
- Automaatne dokumentide teisendamine.
- Küpsiselaadse info kaudu kasutaja seisu säilitamine.

Võib julgelt öelda, et NSAPI võimalused on piiratud ainult veebi arendaja kujutlusvõimega.

2 Serveripoolsed skriptid

Veeb oli algselt mõeldud staatilise linkide süsteemina: hüperteksti (HTML) lehti tuli alati käsitsi muuta, et neis kasvõi kõige väiksemaid muutusi teha. Kuigi selline mudel võimaldab juurdepääsu hästi kujundatud lehtedele, ei võimalda see kiiresti muutuvas infoajastus kiiret info muutmist. Tänapäeval, kus e-äri on väga populaarne ei ole sellise mudeliga midagi peale hakata.

Staatilise mudeli asemel võib kasutada niinimetatud dünaamilist mudelit, kus serveris ei hoita mitte tavalisi lehti, vaid seal töötavad teatavad programmid, mis moodustavad HTML lehekülgi. Klientprogrammi (brauseri) jaoks pole mingit vahet, kas serverist tagastatakse tavaline (füüsiline) fail või on tagastatav vastus genereeritud mõne programmi poolt. Seega näeb brauseri ja veebiserveri vaheline suhtlus (üldistatud kujul) alati välja, nagu seda on kujutatud järgmisel joonisel:



Joonis 2.1 Veebiserveri ja kliendi vaheline suhtlus

Esialgu hakati serveris tavalisi programme kasutama (loomulikult vastasid need programmid teatavatele nõuetele) ning selleks loodi spetsiaalne liides, mis seda teha lubab – *Common Gateway Interface* (CGI). Kuna aga seegi meetod osutus aeglaseks, loodi veel mitmeid liideseid (näiteks *Internet Server Application Programming Interface* – ISAPI, *Netscape Server Application Programming Interface* – NSAPI), mis teevad võimalikuks veebiserverite laiendamise (programmid töötavad veebiserveri sees selle osana). Kuna aga kõik need lahendused raskendavad veebi

programmeerimist (tuleb tunda mingit programmeerimiskeelt ja lisaks sellele ka CGI või mõne muu liidese iseärasusi), siis loodi uus lähenemisviis: serveripoolsed skriptid. See tähendab, et veebiserveris on failid, milles on HTML segatud teatud skriptikeelse programmiga, mida täidetakse serveri pool – tulemuseks on tavaline HTML. Tegelikult eksisteerib serveris sel juhul spetsiaalne programm (nt ISAPI rakendus), mis skripti täita oskab. See lahendus on küll aeglasem kui spetsiaal-liideste rakendused (kuna skriptid tuleb eelnevalt kompileerida), kuid kindlasti on see lahendus lihtsaim ja kiiremini teostatav. Serveripoolsetest skriptidest on enim tuntud *Perl* (algselt CGI skriptidena, seejärel ka serveripoolsete skriptidena), *Active Server Pages*, *ColdFusion*, *Java Server Pages* (JSP) ja PHP. Järgnevas antakse ülevaade kolmest skriptikeskkonnast veebiserveris: pikem ülevaade antakse *Active Server Pages* (ASP)st, lühem ülevaade antakse *ColdFusion*ist ja *PHP*st.

2.1 Active Server Pages

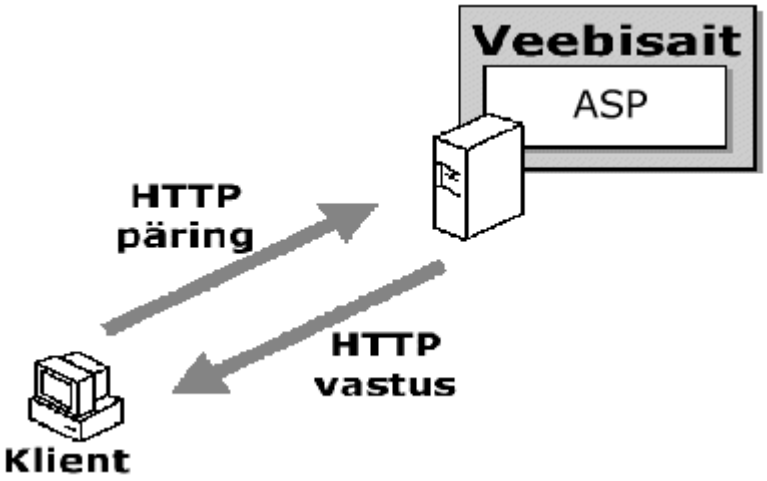
Active Server Pages (ASP) on avatud kompileerimisvaba rakenduskeskkond, mis võimaldab HTMLi, skripte ja serveripoolseid *ActiveX* komponente kasutades luua dünaamilisi ja võimsaid veebipõhiseid ärilahendusi. ASP võimaldab kasutada serveripoolseid skripte *Microsoft Internet Information Server*il ja ka teistel veebiserveritel, standardkomplektis lubatakse kasutada skriptikeeli *VBScript* ja *JavaScript*, samas on võimalik vastavate lisateekide abil kasutada ka teisi keeli nagu näiteks *Perl*.

ASP-rakendused on

- täielikult integreeruvad HTML-lehtedega;
- lihtsalt programmeritavad, käsitsi kompileerimine (ja linkimine) puudub;
- objekti-põhised ja laiendatavad ActiveX serveri komponentidega.

ASP alustab tööd, kui klientprogramm soovib näha mõnda .asp-laiendiga faili (asp-fail) veebiserverist. Veebiserver kutsub välja ASPi; ASP loeb küsitud asp-failist HTMLi ja skripti; täidab selles olevad skripti käsud, koostab lõpliku HTML-lehe ja saadab selle brauserile.

Seega näeb ASPi mudel välja nagu on kujutatud järgmisel joonisel:



Joonis 2.2 Active Server Pages töö veebiserveris

ASP on loodud Microsofti poolt ja seega töötas ASP esialgu ainult Microsofti veebiserveritel (IIS ja selle analoogid), kuid tänapäeval on loodud mitmeid lahendusi, mis teevad võimalikuks ASPi kasutamise ka muudel veebiserveritel ja platvormidel (nt Chili!Soft ASP (vt [11]) võimaldab ASPi kasutada ka Apache ja Netscape veebiserverites).

Täieliku ülevaate ASPist annavad allikas [12] ja [13], ASPiga esimest tutvust tehes on soovitatav vaadata mõnda mitte-autori poolt kirjutatud õpetust (nt allikas [14]).

2.1.1 ASP skriptid

ASP-fail on tekstifail, mis võib sisaldada suvalist kombinatsiooni järgnevatest osadest:

- tekst;
- HTML elemendid;
- skripti käsud.

Seega on lihtsaim viis ASP-faili tekitamiseks HTML-failil ‘htm(l)’-laiendi ‘asp’-laiendiga (.asp) asendamine. Klientprogramm näeb tulemusena täpselt sama, mis htm(l)-laiendi korral. Tegelikult alustab ASP tööd, kui lisada HTML-i ka skripti käske. Skript on rida käske, mis võivad näiteks:

- omistada väärtusi muutujatele;

- käskida veebiserveril saata midagi (näiteks mingi muutuja väärtuse) klientprogrammile;
- koondada käske protseduurideks.
- Kasutada serveris olemasolevaid COM (*Component Object Model*) ja DCOM (*Distributed Component Object Model*) objekte.

Objektide kasutamist peetakse üheks suurimaks ASP plussiks.

ASP ei ole programmeerimiskeel, ta pigem loob keskkonna, mis võimaldab tavalist HTMLi skriptiosadega siduda. HTML elemendid on tavalisest tekstist eraldatud eraldajatega (sümbolid '<' ja '>'), ASPi puhul peavad aga eraldajad skripti osa eraldama nii tekstist kui HTML elementidest. Selleks kasutatakse eraldajat '<%' skripti alguses ja '%>' skripti lõpus – nende eraldajate järgi saab ASP aru, et tegemist on skriptiga ja selles osas olev tuleb serveri pool täita. Spetsiaalsed eraldajad on väljundi saatmiseks (näiteks mingi muutuja väärtuse saatmiseks klientprogrammile): '<%=' alguses ja '%>' lõpus – selline väljastusviis on mõeldud programmeerija töö hõlbustamiseks: sellisel juhul ei pea näiteks ühe muutuja väljastamiseks spetsiaalset väljastuskäsku kasutama. Skript võib sisalduda suvalises kohas HTMLis. Seega on õige:

```
<h1>Hetke serveri kuupäev ja kellaaeg on <i><%=Now()%></i>.</h1>
```

Toodud näites näeb klient brauseris ainult lauset, mis sisaldab funktsiooni Now poolt tagastatud väärtust.

Selleks, et täpsustada skriptikeelt, milles skript on kirjutatud, tuleb kasutada tavalisi HTML elemente <SCRIPT> ja </SCRIPT> koos atribuudiga LANGUAGE (sellega teatatakse ASPile skriptikeel). Et eristada brauseris käivitatavaid skripte serveris käivitatavatest, tuleb määrata ka atribuut RUNAT=SERVER (sellega teatatakse ASPile, et skript tuleb käivitada serveris). Tuleb aga meeles pidada, et selle meetodiga saab defineerida ainult oma protseduure. Kui soovitakse muuta kogu ASP-faili skriptikeelt (keelt, mida kasutatakse eraldajate '<%' ja '%>' vahel), tuleb faili alguses määrata ASP direktiiv <%@LANGUAGE %>, mille süntaks on järgmine:

```
<%@ LANGUAGE=Skriptikeel %>
```

Skriptikeel tähistab keelt, mida selles ASP-failis kasutatakse.

ASP-failidesse on võimalik sisestada teisi faile – selleks kasutab ASP *Server Side Includes* (SSI) metoodikat. ASPis on realiseeritud selle metoodika üks käsk/element: #INCLUDE, mille süntaks on:

```
<!--#INCLUDE VIRTUAL|FILE="failinimi"-->
```

Selle meetodiga sisestatakse fail (mille nimi on antud atribuudiga failinimi) ASP-faili enne kui ASP töötleb jooksvat faili (loomulikult töödeldakse ka sisestatud faili kui põhifaili osa). Võtmesõna *VIRTUAL* võimaldab määrata faili virtuaalse tee abil (tee, mis on määratud veebiserveris füüsilisele teele vastavaks). Võtmesõna *FILE* lubab määrata faili füüsilise tee jooksvast kataloogist alates.

2.1.2 ASP rakendused

ASP-rakendus koosneb (virtuaalsest) kataloogist ja kõikidest selles olevatest failidest ning alamkataloogidest. ASP-rakendus võib olla üks ASP-lehekülg, võib sisaldada mitmeid dünaamilisi lehti (eri kasutajate jaoks erinev leht, eri brauserite jaoks erinevad lehed jm) või koosneb mitmetest (omavahel loogiliselt seotud) lehtedest. Kasutades ASP-rakendusi saab säilitada kasutaja seisu (informatsiooni hoidmine kasutaja kohta); kasutaja seise on kahte tüüpi:

- Rakenduse seis, mille korral säilitatakse kogu rakenduse kohta käivat infot ja see info on kättesaadav kõigile rakenduse kasutajatele.
- Sessiooni seis, mille korral säilitatakse konkreetse kasutaja seis rakenduse kasutamise ajal. Säilitatav info on kättesaadav ainult ühele kasutajale. Sessiooni seisu on võimalik säilitada kasutajate kohta, kes kasutavad küpsiseid toetavat klientprogrammi, seega ei ole sessioonis andmete säilitamine eriti mõttekas, kui on teada, et süsteemi kasutavad kliendid ei soovi oma brauseritel küpsiseid lubada.

Seise võimaldavad hallata ASP objektid **Application** (rakenduse info hoidmiseks ja kõikide kasutajate vahel jagamiseks) ja **Session** (konkreetse kasutaja info hoidmiseks kui, see kasutaja rakendust kasutab).

Iga ASP-rakendus võib sisaldada ühte fail nimega Global.asa (laiend asa tuleneb väljendist *Active Server Application*), mis peab asuma rakenduse peakataloogis. ASP loeb faili Global.asa, kui:

- Peale veebiserveri laadimist toimub esimene pöördumine mõne rakenduse ASP-faili poole.
- Kasutaja, kellel ei ole aktiivset rakenduse sessiooni, pöördub mõne rakenduse ASP-faili poole.
- Kasutaja sessioon lõpeb.
- Kogu rakendus lõpetab töö (enne veebiserveri seiskumist).

Fail Global.asa võib sisaldada protseduure juhtimaks rakenduse ja sessiooni tööd:

- Protseduurid, mis käivitatakse rakenduse ja/või sessiooni alguses (*Application_OnStart* ja *Session_OnStart*).
- Protseduurid, mis käivitatakse rakenduse ja/või sessiooni lõppedes (*Application_OnEnd* ja *Session_OnEnd*).
- Globaalsete objektide esindajate loomist – objektid, mis on kasutatavad kõikide rakenduse failide poolt kogu rakenduse/sessiooni töö jooksul.

Sessioon luuakse järgnevatel viisidel:

- Kasutaja pöördub ASP-faili poole rakenduses, mille spetsiaalfailis Global.asa on defineeritud protseduur *Session_OnStart*.
- ASP-failis salvestatakse mõni väärtus sessioonimuutujasse (*Session*-objekti muutujasse).
- kasutaja pöördub ASP-faili poole rakenduses, mille spetsiaalfailis Global.asa on loodud mõni sessiooni-objekt (elemendiga OBJECT).

Sessiooni loomisel genereeritakse igale kasutajale vastav numbriline identifikaator (*SessionID*), mille kaudu ASP suudab konkreetse kasutaja üheselt identifitseerida. Klientprogrammile saadetakse spetsiaalne küpsis, mille alusel genereeritakse identifikaator. Siit ka põhjus, miks pole võimalik *Session*-objekti luua kasutajatele, kelle brauser ei toeta küpsiseid. Sessiooni identifikaatori poole saab pöörduda (ainult lugemiseks) atribuudi *Session.SessionID* kaudu.

Sessioon hävitatakse, kui sessioon kaotab kehtivuse (kasutaja ei pöördu teatud aja jooksul ühegi rakenduse ASP-faili poole) või kui sessioon lõpetatakse. Vaikimisi kehtib sessioon 20 minutit, aga seda saab ka ASP-failides muuta, kasutades *Session*-objekti atribuuti *Session.Timeout* (määratakse minutites). Sessiooni lõpetamiseks võib kasutada ka *Session*-objekti meetod *Abandon*. Tegelikult hävitatakse Session-objekt alles pärast meetodit välja kutsunud ASP-faili käskude täitmist. Seega on võimalik samal lehel kasutada kõiki *Session*-objekti omadusi.

Rakenduse objekt luuakse, kui peale veebiserveri alglaadimist pöördub esimene kasutaja mõne ASP-faili poole rakenduses. Järgmiste pöördumiste korral rakendust ei looda, vaid see säilub, kuni veebiserveri seiskamiseni.

Rakenduse tasemel saab näiteks veebiserveri administraator saata kõikidele kasutajatele päevakohast infot. Samuti saab rakenduse tasemel kasutada loendurit külastajate üle arve pidamiseks (rakenduse alguses võib loenduri nullida, külastajate andmebaasis algseise taastada jms). Rakenduse objekti omadusi saab kasutada suvaline rakenduse ASP-fail suvalises sessioonis. Selleks, et rakenduse omadusi (näiteks rakenduse objekti salvestatud muutujaid) turvaliselt (kartmata, et keegi teine samal ajal omadusi muudab) kasutada, võib rakenduse objekti lukustada (meetodiga *Application.Lock*). Sel ajal kui rakendus on lukustatud, on rakenduse objekt muudetav ainult lukustava ASP-faili poolt. Rakenduse lukust vabastamisel (meetodiga *Application.Unlock*) võivad kõik rakenduse ASP-failid rakenduse objekti muuta.

Rakenduse objekt kustutatakse vahetult enne seda, kui veebiserver seiskub.

2.1.3 ASP objektid

Nagu juba mainitud, on ASP objekti-põhine – see tähendab, et teatud vajaminevad meetodid(protseduurid)/omadused(andmed) on koondatud ühtseks tervikuks - objektiks. Et aga enda poolt loodud objekte ASPis kasutada, tuleb need kõigepealt luua. Samas on ASPi sisse ehitatud mõned objektid, mis ei vaja eelnevat spetsiaalset loomist, vaid on kohe kasutatavad:

Objekt	Milleks kasutatakse
Application	Kogu rakenduse piires ühtseks info hoidmiseks.
Session	Kasutaja konkreetse sessiooni haldamiseks.
Request	Kasutaja (brauseri) poolt HTTP päringuga saadetud infole juurdepääsu tagamiseks.
Response	Kasutajale (brauserile) info saatmiseks.
Server	Teatavate serveri meetodite ja omaduste kasutamiseks (näiteks enda loodud objektide loomiseks ASPis).
ObjectContext	IIS 5-e poolt sisse toodud objekt spetsiaalsete komponent-teenuste transaktsioonide haldamiseks.
ASPError	IIS 5-e poolt sisse toodud objekt ASP vigade halduseks (võimaldab ASPi vigade kohta detailsema info saamist).

Tabel 2.1 Active Server Pages objektid ja nende kasutusvõimalused

Objektide süntaks sõltub konkreetsest skriptikeelest, aga kuna *VBScript* on vaikimisi kasutatav keel, siis võetakse allpool aluseks selle keele süntaks.

Objektid võivad sisaldada kogumeid: hulk omavahel mingil moel ühendatud elemente objektis, mille poole pöördutakse samal viisil. Objektide poole saab pöörduda kasutades objekti meetodeid ja omadusi. *Meetod* on protseduur, mis tegutseb objektil. Meetodi üldine süntaks on järgmine:

Objekt.Meetod parameetrid

Omadus on nimeline atribuut objektil, mis tavaliselt iseloomustab objekti. Omaduse üldine süntaks on järgmine:

Objekt.Omadus parameetrid

Rakenduse (*Application*) ja **sessiooni** (*Session*) objektidest oli juttu eelmises alajaotuses, seega neil siin pikemalt ei peatu. ***ObjectContext*** ja ***ASPError*** on ASPi toodud alles IIS 5-e poolt ja selle tõttu ei vaadelda neid siin pikemalt.

Brauseri poolt saadetud info lugemiseks ASPis kasutatakse **Request** objekti. Selle objektiga on võimalik lugeda igasugu infot, mida brauser saadab serverile ja/või server seab:

- HTTP protokolli ja serveri standardinfo ehk keskkonnamuutujad (kogum *ServerVariables*).
- HTML-vormilt POST-meetodiga saadetud info (kogum *Form*).
- Päringuparameetrid (GET-meetod, kogum *QueryString*).
- Niinimetatud küpsised (*cookies*), millega on võimalik kasutaja kohta infot säilitada, et siis edaspidi näiteks kasutajat ära tunda vms (kogum *Cookies*).
- Kliendi sertifikaadid (kogum *ClientCertificate*). Sertifikaate kasutatakse kasutaja ja veebiserveri vahelise turvalise ühenduse loomiseks ja hoidmiseks.

Kui Request objekti poole pöördutakse kasutaja info saamiseks ilma kogumit määratlemata, otsitakse vastavat infot kõikidest kogumitest, kuni otsitav info leitakse (või tagastatakse spetsiaalväärtus, mis tähendab tühja väärtust). Kogumeid vaadatakse läbi järgmises järjekorras: *QueryString*, *Form*, *Cookies*, *ServerVariables* ja *ClientCertificate*. Alates IIS 5-st on *ServerVariables* ja *ClientCertificate* järjekord vahetatud. Kui klient saadab ASPile binaarseid andmeid (näiteks saadetakse binaar-fail), siis neid andmeid võimaldab lugeda Request objekti ainus meetod *ReadBinary*.

Kliendile (brauserile) info saatmise haldamiseks on loodud objekt **Response**. Sellega saab kliendile infot saata ja saadetava info parameetreid muuta (nt millal info oma kehtivuse kaotab).

Response objekt sisaldab ainult ühte kogumit *Cookies*, mida kasutatakse brauserile küpsiste saatmise haldamiseks (küpsise parameetrite määramiseks). Samas on objektil defineeritud mitmed atribuudid:

- *Buffer* määrab kas HTML-väljundit puhverdatakse (säilitatakse töö käigus serveril ja väljastatakse peale töö lõppu kõik korraga). See atribuut võimaldab ASPi tööd paremini kontrollida; näiteks saab vea tekkimise korral keelata

seni koostatud väljundi saatmise kliendile ja saata selle asemel hoopis veateate.

- *CacheControl* võimaldab kontrollida lehe säilitamist puhverserverites (*proxy*), st lehega saab saata infot, kas puhverserverid säilitavad lehte või mitte.
- *Charset* võimaldab määrata väljundi märgistikku.
- *ContentType* määrab väljundi tüübi.
- *Expires* määrab aja, mille jooksul HTML-leht brauseris oma kehtivuse kaotab.
- *ExpiresAbsolute* määrab täpse aja (kuupäev ja kellaaeg), millal leht kehtivuse kaotab.
- *IsClientConnected* võimaldab kontrollida, kas klient on endiselt serveriga ühenduses.
- *PICS* võimaldab määrata HTTP *pics-label* päise väärtust.
- *Status* määrab HTML-lehe päises staatusrea (tavaliselt pannakse see veebiserveri poolt).

Tasub tähele panna, et siin toodud atribuutidest on mõned (nt *CacheControl*, *IsClientConnected*) kasutatavad alates IIS viiendast versioonist.

Response objektil on defineeritud ka mitmed meetodid:

- *AddHeader* võimaldab lisada HTTP päisesse ridu (sellega saab skript lisada isiklikke päiseridu).
- *AppendToLog* lisab veebiserveri logifaili parameetrina määratud teksti.
- *BinaryWrite* väljastab kõik sümbolid neid teisendamata. See meetod on kasulik väljastamaks mitte-tekstilisi andmeid (binaarsel kujul olevaid andmeid).
- *Clear* kustutab puhverdatud info (kui seda pole veel väljastatud). Funktsiooni kasutamisel on mõte ainult siis kui eelnevalt on defineeritud, et väljundit tuleb puhverdada (*Buffer* atribuudiga).
- *End* lõpetab ASP-faili töötlemise ja tagastab selleks hetkeks moodustatud väljundi.
- *Flush* väljastab puhverdatud info. Funktsiooni kasutamisel on mõte ainult siis kui eelnevalt on defineeritud, et väljundit tuleb puhverdada (*Buffer* atribuudiga).

- *Redirect* võimaldab klienti uuele asukohale (määratakse meetodi parameetriga) saata.
- *Write* väljastab parameetrina määratud teksti brauserile.

Server objekt võimaldab kasutada teatavaid serveripoolseid omadusi ja meetodeid – enamus neist on tihti vajaminevad utiliidid.

Server objektil on defineeritud ainult üks atribuut: *ScriptTimeout*, millega saab määrata aega (sekundites), mille jooksul antud skript peab täitmise lõpetama. Kui skripti töö ei lõpe määratud perioodi jooksul, siis väljastab ASP vastava veateate kliendile. Vaikimisi on see aeg 90 sekundit.
Samas on objektil defineeritud mitmed kasulikud meetodid:

- *CreateObject* loob serveripoolse objekti ASPis kasutamiseks.
- *Execute* meetod loeb parameetrina määratud ASP faili ja täidab selle nagu oleks see meetodi välja kutsunud faili osa.
- *GetLastError* meetod võimaldab veatöötlust ASPis (tagastab *ASPError* objekti).
- *HTMLEncode* meetod rakendab parameetrina määratud stringile HTML kodeerimist.
- *MapPath* teisendab parameetrina määratud virtuaalse või relatiivse tee vastavaks füüsiliseks failiteeks.
- *Transfer* annab juhtimise üle teisele ASP-failile (parameetrina määratud) andes seejuures kogu kliendilt tuleva info teisele ASP failile.
- *URLEncode* meetod rakendab parameetrina määratud stringile URLi kodeerimist.

Tasub tähele panna, et siin toodud meetoditest on mõned (nt *Execute*, *Transfer*) kasutatavad alates IIS viiendast versioonist.

Nagu juba eelpool mainitud, saab ASPis lisaks eeldefineeritud objektidele kasutada ka muid serveris registreeritud objekte (realiseeritud COM/DCOM komponentidena). Selleks, et objekti kasutada, tuleb ta kõigepealt luua; selleks võib kasutada server objekti meetodit *CreateObject* või spetsiaalset elementi <*OBJECT*>. Kasutades *OBJECT*-elementi, tuleb tingimata määrata lisaatribuut RUNAT=SERVER. Lisaks

enda pool loodud objektidele võimaldab IIS installeerida serverisse ka mõned eelvalmistatud objektid, mis võimaldavad näiteks reklaamibannerite näitamist lehel, brauserite omaduste tundmist ASPis, andmebaasidega suhtlemist, lehekülje külastuste lugemist, kasutaja õiguste kontrolli jne.

2.2 ColdFusion

ColdFusion on firma Allaire poolt toodetud rakendusserver, mis on täielik veebirakenduste server loomaks väga võimsaid e-äri lahendusi. ColdFusion on loodud võimaldamaks e-äri ja veebirakenduste põhilisi omadusi:

- **Kiire arendus**. Visuaalsed programmid ja element-põhine programmeerimiskeskkond teevad ColdFusionist väga hea vahendi lahenduste loomiseks.
- **Laiahaardelisus**. Kõrge jõudlus, lõimtöötlusega (*multithreading*) arhitektuur ja erinevad omadused nagu jooksev kompileerimine, koormuse jaotamine ja veatöötlus garanteerivad, et loodud rakendused hõlmavad kõikvõimalikke situatsioone.
- **Avatud suhtlus**. Avatud suhtlus andmebaasidega, e-maili süsteemidega, Java, XML ja muu sellisega tähendab, et keerukaid veebilahendusi on võimalik luua kiirelt ja lihtsalt.
- **Täielik turvalisus**. Viimased Interneti turvatehnoloogiad ja turvaline suhtlus veebiserveri ja võrgu vahel loovad võimalused turvaliste süsteemide arendamiseks.

Tänu ColdFusioni nendele omadustele ja asjaolule, et ColdFusioni server on piiramatute funktsioonidega, ei ole ColdFusion ka vabavara. See asjaolu on ColdFusioni levikut küll tõenäoliselt kuigipalju piiranud, kuid vaatamata sellele on see maailmas väga populaarne.

ColdFusion sisaldab komplekti visuaalseid programme, võimsat serverit ja avatud keelekeskkonda:

- **ColdFusion Studio** on tihedalt integreeritud *ColdFusion Server*-iga võimaldades seejuures visuaalset programmeerimist, sisaldab kõike

andmebaasidega suhtlemiseks ja silumisvahendeid ning on seega võimas vahend veebirakenduste loomiseks.

- **ColdFusion Server** annab veebi loojale kõik vahendid e-äri lahenduste loomiseks.
- ColdFusion kasutab serveripoolset **keelekeskkonda** *ColdFusion Markup Language* (CFML), mis on tihedalt seotud HTMLi ja XMLiga (*Extensible Markup Language*). CFML hõlmab endas 70 serveripoolset elementi, 200 funktsiooni ja 800 erinevat juurdelisatavat komponenti.

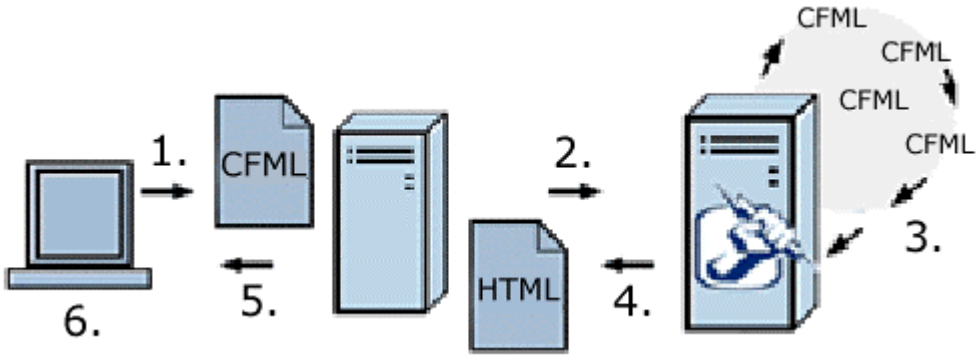
Täieliku ülevaate ColdFusionist võib leida allikast [15]**,** CFMList annab ülevaate allikas [16], ColdFusioniga esimest tutvust tehes on soovitatav vaadata mõnda mitte-autori poolt kirjutatud õpetust (nt allikas [17]).

2.2.1 ColdFusioni töö

ColdFusion töötab samal põhimõttel, kui näiteks eelpool kirjeldatud ASP:

1. Klient pöördub veebiserveri poole ja küsib lehte, mis sisaldab CFML elemente (CFML lehte).
2. Veebiserver annab juhtimise ColdFusionile.
3. ColdFusion töötleb lehte ja täidab kõik CFML elementidega antud direktiivid. Seejuures võib ColdFusion pöörduda andmebaasi poole andmete saamiseks, neid andmeid andmebaasi lisada, neid muuta.
4. ColdFusion saadab tulemuseks saadud väljundi veebiserverile.
5. Veebiserver saadab vastuse kliendile.

Järgmine joonis kirjeldab toodud algoritmi:



Joonis 2.3 ColdFusion serveri töö veebis

Lisas 1 on toodud tabel, milles on kirjeldatud ColdFusioni koostöö erinevate veebiserverite ja andmeallikatega.

2.2.2 Andmebaasidega suhtlemine CFMLis

Andmebaasidega suhtlemiseks kasutab ColdFusion põhiliselt ODBCd (*Open Database Connectivity*), kuid samas tunnistab ta ka mõningaid OLE DB andmetarnijaid ja mõningate (nt Oracle) andmebaasisüsteemide draivereid. Andmebaasidega suhtlemiseks on CFMLis defineeritud järgmised elemendid:

- *CFQUERY* võimaldab saata erinevaid SQL lauseid andmeallikale.
- *CFSTOREDPROC* võimaldab andmeallikas defineeritud salvestatud protseduure käivitada.
- *CFPROCPARAM* võimaldab määrata salvestatud protseduuridele parameetreid.
- *CFPROCRESULT* võimaldab salvestatud protseduuride tulemusi CFMLis käsitleda.
- *CFINSERT* võimaldab sisestada uusi kirjeid andmeallikasse.
- *CFUPDATE* võimaldab andmeallikas kirjeid uuendada.
- *CFTRANSACTION* võimaldab luua transaktsioone (nt täita mitu päringut korraga).

2.2.3 Andmete väljastus CFMLis

Kui andmed on andmeallikast saadud (näiteks CFQUERY elemendiga), siis peab neid mingil moel ka lehel näitama. Tuleb tähele panna, et väljastada saab ainult nende päringute tulemusi, mis on eelnevalt defineeritud, seega on soovitatav lehekülje alguses defineerida päringud ja hiljem lehel neid siis kasutada. Järgmised elemendid võimaldavad päringu tulemust väljastada:

- *CFOUTPUT* võimaldab väljastada päringu (või näiteks salvestatud protseduuri) tulemust. CFOUTPUT elemente ei tohi üksteise sisse paigutada.
- *CFTABLE* võimaldab koostada tulemusest tabelit.
- *CFCOL* kasutatakse koos CFTABLE elemendiga ja see defineerib tabeli veerud.

Kui väljastust tehakse CFOUTPUT elemendiga, siis selle sees määratakse väljastatavad väljanimed #-sümbolite vahel. Näiteks:

```
<CFOUTPUT QUERY="query1">
#item#<br>
</CFOUTPUT>
```

Selle tulemusena väljastatakse päringu *query1* kõik välja *item* väärtused üksteise alla.

Lisaks päringu tulemuse väljastamisele saab CFMLis näiteks lisada oma päiseridu HTTP päisesse, see toimub elemendiga *CFHEADER*.

2.2.4 Juhtimisstruktuurid CFMLis

Lisaks tavalisele käskude järjestikusele täitmisele võimaldab CFML kasutada ka mitmesuguseid juhtimisstruktuure (tingimusi, tsükleid). Juhtimisstruktuurid CFMLis on järgmised:

- *CFABORT* katkestab töö lehel kohas, kus element esineb. ColdFusion väljastab kogu selleks hetkeks moodustatud väljundi ja katkestab töö sellel lehel. Tavaliselt kasutatakse seda elementi tingimustes.
- *CFEXIT* võimaldab (lisaparameeter *METHOD*):
    - katkestada hetkel töödeldavat programmeerija poolt defineeritud elementi (METHOD=ExitTag);
    - väljuda hetkel aktiivsest mallist (METHOD=ExitTemplate);
    - täita teatud koodilõike uuesti (METHOD=Loop).
- *CFIF CFELSE* ja *CFELSEIF* võimaldavad koostada tavalist tingimuslikku käskude täitmist.
- *CFSWITCH CFCASE* ja *CFDEFAULTCASE* võimaldavad koostada tavalist valikulist käskude täitmist.
- *CFLOOP* võimaldab luua väga võimsaid tsükleid nagu näiteks:
    - tavaline loendustsükkel (tsükkel lõpeb, kui teatud muutuja saab teatud väärtuse);
    - tingimuslik tsükkel (tsükkel lõpeb, kui teatud tingimus saab tõeseks);
    - tingimus üle päringu (tsükkel lõpeb, kui määratud päringus lõpevad kirjed);
- *CFBREAK* võimaldab tsüklist väljuda.

- *CFLOCATION* võimaldab sisestada lehele mõnda teist lehte.
- *CFTRY* ja *CFCATCH* võimaldavad CFMLis veatöötlust.
- *CFTHROW* võimaldab CFMLis viga esile kutsuda.

2.3 PHP

PHP on serveripoolne multiplatvorm HTMLiga integreeritud skriptikeel, mis võimaldab luua dünaamilisi veebilehti. Nimi 'PHP' pärineb kõige esimesest versioonist PHPst, mida nimetati *Personal Home Page Tools* ja hiljem *Personal Home Page Construction Kit*; praegu nimetatakse PHPd ka kui *PHP: Hypertext Preprocessor*.

Paljud ülesanded, mis teistes keeltes nõuavad teadmisi ning oskusi, saavad PHPs lahendatud väga kiirelt. PHP on kombinatsioon Perli, Java ja C kontseptsioonidest. Süntaksi struktuur on suurel määral laenatud C-st, seega on C programmeerimisega kokku puutunud inimesel väga lihtne PHPd õppida. PHP on võimeline tegema keerulisi matemaatilisi kalkulatsioone, esitama võrguinfot, pakub e-maili võimalusi, regulaaravaldiste kasutamist ja veel paljutki. Suurimaks PHP tugevuseks peetakse siiski andmebaasidega suhtlemise omadusi – andmebaasi Internetti ühendamine on tehtud väga lihtsaks. Veelgi enam, PHP toetab enamust populaarsetest andmebaasiserveritest nagu näiteks MySQL, Oracle ja Sybase; loomulikult omab PHP ka ODBC toetust. PHP teeb niivõrd populaarseks asjaolu, et see on eesmärgile orienteeritud keel – see on kirjutatud sellisel moel, et võimaldab saavutada eesmärke kiirelt ja lihtsalt.

PHP juhendi võib leida allikast [18], PHPst ülevaate leiab allikast [19].

2.3.1 PHP süntaks

PHP skriptide eraldamiseks tavalisest HTMList on neli võimalust:

- Eraldajad '*<?*' PHP skripti alguses ja '*?>*' lõpus.
- Eraldaja '*<?php*' PHP skripti alguses ja '*?>*' lõpus.
- Märkides PHP skripti algust elemendiga *<script language="php">* ja skripti lõppu elemendiga *</script>*.
- Kasutades ASP-stiilis eraldajaid ('*<%*' skripti alguses ja '*%>*' lõpus ja '*<%=*' ning '*%>*' väärtuste väljastamiseks).

Seega järgmised viis rida on kõik korrektsed:

```
<? echo ("Tere maailm\n") ?>
<?php echo("Tere maailm\n"); ?>
<script language="php"> echo ("Tere maailm\n"); </script>
<% echo ("Tere maailm\n"); %>
<%= $muutuja; %>
```

Käsueraldajad on sarnased programmeerimiskeelte C ning Perl süntaksile: iga käsk peab lõppema semikooloniga. Erandina on semikoolon lubatud käsu lõpust ära jätta, kui käsule järgneb skripti lõpetav eraldaja (*?>*).

Kommentaaride eraldamiseks on lubatud kasutada C, C++ ja UNIX käsukeele (*shell*) stiilis kommentaaride eraldajaid: //, /* ja */ ning #. Kommenteerimisel tuleb tähele panna, et C stiilis kommentaare ei tohi üksteise sisse paigutada.

Muutujaid märgitakse PHPs dollarimärgiga muutujanime ees ja muutujanimi selle järel; muutuajanimed on tõusutundlikud (*$muut* ei ole sama mis *$Muut*). PHP omab toetust järgmisi muutujatüüpe: massiiv (*array*), ujukomaarv (*floating-point numbers*), täisarv (*integer*), objekt (*object*) ja string (*string*). Programmeerija ei pea tavaliselt eraldi tüüpi määrama, kuna PHP määrab selle automaatselt töö käigus sõltuvalt muutuja kasutamise kontekstist. Lisaks enda poolt defineeritud muutujatele võimaldab PHP kasutada ka mitmeid eeldefineeritud muutujaid, mis tähistavad näiteks CGI muutujaid ja PHP keskkonnamuutujaid.

PHPs on realiseeritud kõik enamkasutatavad juhtimisstruktuurid:

- tingimuslikku juhtimist (*if, else, elseif*);
- valikulist juhtimist (*switch, case*);
- tsükleid (*while, do..while, for*);
- lisafailide sisestamist PHP skripti (*include, require*).

2.3.2 Andmebaasidega suhtlemine

PHP võimaldab suhelda paljude andmebaasisüsteemidega. Selleks võib kasutada PHP ODBC toetust, samas omab PHP ka mitmete andmebaasiserveritega otse suhtlemise võimalust (näiteks Microsoft SQL Server, Oracle, MySQL). ODBC abil on võimalik teha kõiki tavalisi andmebaasidega seotud tehinguid:

- Luua ühendusi ODBC andmeallikaga (*odbc_connect*).
- Katkestada ühendust (*odbc_close* ja *odbc_close_all*).
- Luua transaktsioone (*odbc_autocommit* võimaldab alustada transaktsioone, *odbc_commit* transaktsiooni edukalt lõpetada ja *odb_rollback* transaktsiooni katkestada).
- Käivitada SQL lauseid päringute, muudatuste ja lisamiste tegemiseks andmebaasi ning lugeda päringute jms tulemusi.

Lisaks ODBC vahenditele on PHPs defineeritud ka paljude enam kasutatud andmebaasisüsteemidega suhtlemiseks vajalik funktsionaalsus. Näiteks MS SQL Serveri puhul võimaldab PHP:

- Luua ühendust serveriga (*mssql_conect*).
- Ühendust sulgeda (*mssql_close*).
- Valida SQL serveris olevat andmebaasi kasutamiseks (*mssql_select_db*).
- Teha serveris päringuid (*mssql_query*) ja kasutada päringu tulemusi (*mssql_result* või näiteks *mssql_fetch_row*).

2.3.3 Kliendiga suhtlemine

PHP on teinud kliendiga suhtlemise väga mugavaks. Nimelt on näiteks kõik kliendi poolt saadetud vormi muutujad PHP poolt eeldefineeritud kui programsed muutujad. Näiteks kui vormil sisaldub tekstiväli nimega *Eesnimi*

```
<input type=”text” name=”eesnimi”>
```

ja klient saadab selle PHP skriptile, siis võib skript selle vormi muutuja väärtuse poole pöörduda lihtsalt muutuja *$eesnimi* kaudu. Keskkonnamuutujate (ka CGI muutujad on keskkonnamuutujad) poole pöördumiseks võib kasutada funktsioone *phpinfo* (võimaldab muu hulgas ka keskkonnamuutujate nimistu saamist) ja *getenv* (konkreetse muutuja väärtuse saamiseks).

Kliendile info saatmine toimub tavalise väljastusena. Seega võib kogu väljastatava teksti väljastada funktsiooniga *echo* või *print*.

HTTP päiste lisamine toimub funktsiooniga *Header*. See funktsioon võimaldab PHP skriptis sundida veebiserverit kasutajat tuvastama, selleks võib näiteks kasutada järgmist koodilõiku:

```
<?php
     if(!isset($PHP_AUTH_USER)) {
          Header("WWW-Authenticate: Basic realm=\"XX\"");
          Header("HTTP/1.0 401 Unauthorized");
          echo "Kasutajatunnus kohustuslik!\n";
          exit;
     } else {
          echo "Tere $PHP_AUTH_USER.<P>";
     }
?>
```

3 Suhtlus andmebaasidega

Kõik tänapäevased infosüsteemid põhinevad mingitel andmetel. Infosüsteem hoiab andmebaasis kasutajate andmeid, kasutajad saavad teha päringuid andmebaasi, muuta seal andmeid, lisada andmeid. Andmete põhjal tehakse erinevaid aruandeid, nende põhjal analüüsitakse kasutajate soove, mis võimaldab süsteemi paremaks teha. Seega võib öelda, et andmed on üheks väga tähtsaks osaks infosüsteemis. Andmeid võib hoida tavalise tekstifailina või säilitada neid erinevate võimalustega andmebaasisüsteemides – relatsioonilistes või mitte.

Selleks, et süsteem andmebaasi kasutada saaks, on võimalik kasutada erinevaid vahendeid:

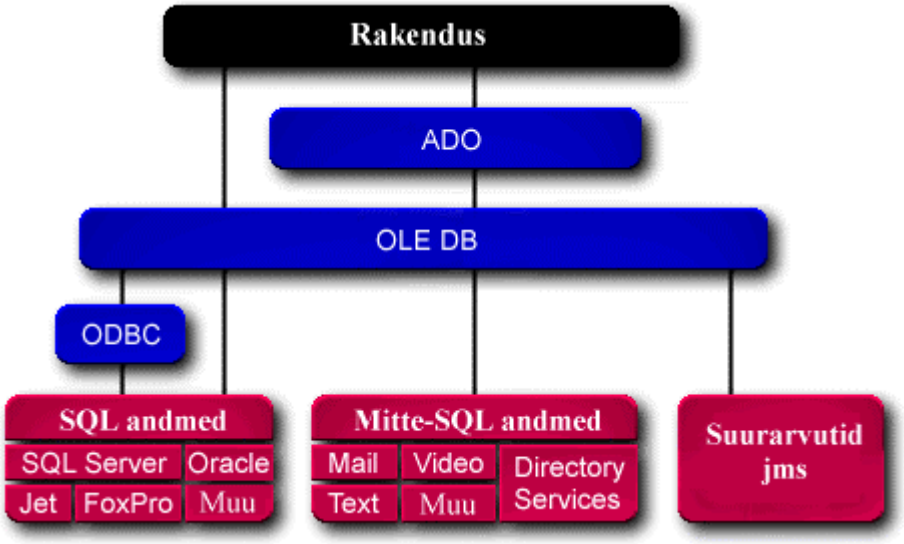
- Võib kasutada andmebaasi poolt pakutud vahendeid. Selliste vahendite miinuseks on asjaolu, et andmebaasi vahetuse korral tuleb tavaliselt teha suuri muutusi kogu infosüsteemis.
- Võib kasutada standardvahendeid, mis on mõeldud üldiseks kasutamiseks. Selliste vahendite eeliseks on asjaolu, et need võimaldavad vähese vaevaga vahetada andmebaasisüsteemi – muutusi tuleb teha ainult selles osas, mis kasutab konkreetse andmebaasisüsteemis spetsiifilisi omadusi.

Microsoft Windows keskkonnas kasutatakse andmebaasidega suhtlemiseks universaalset andmeühendust (*Universal Data Access*). *Universal Data Access* on Microsofti poolt kirjeldatud allikas [20].

Universal Data Access (UDA) on kujunenud Microsofti strateegiaks võimaldamaks juurdepääsu suvalistele andmetele suvalises kohas. UDA annab võimsad vahendid suhtlemaks erinevate andmekogudega (relatsiooniliste või mitte-relatsioonilistega), annab lihtsad programmeerimiskeelest sõltumatud tehnoloogiad suhtluse teostamiseks. Need tehnoloogiad võimaldavad luua süsteeme, mis on lihtsalt hallatavad, mida on võimalik kasutada erinevates keskkondades ja platvormidel. UDA ei nõua andmete koondamist ühte punkti ega ühe konkreetse toote kasutamist; UDA põhineb avatud süsteemidel ja toetab lai-süsteeme ning töötab enamustel

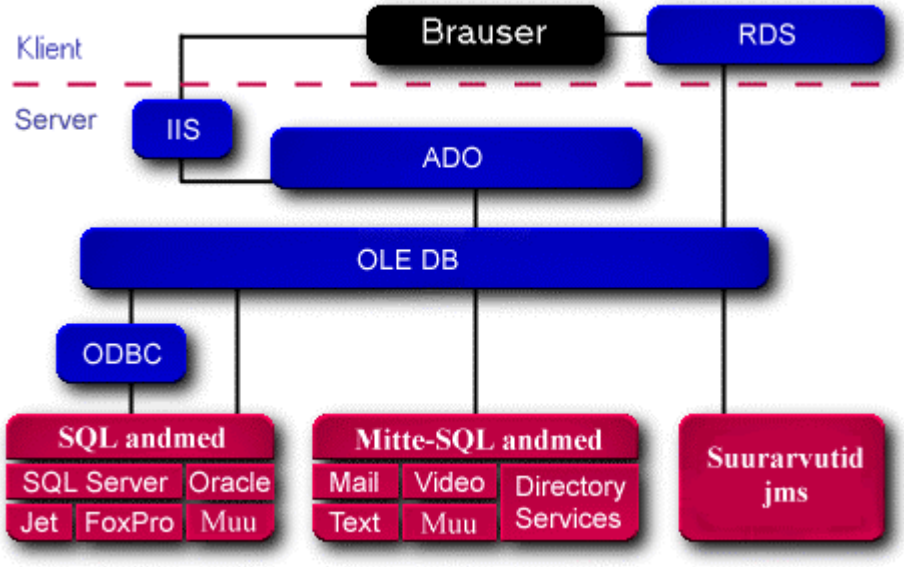
levinud andmebaasi-platvormidel. UDA on samm edasi levinud tehnoloogiatest nagu ODBC, RDO ja DAO, samas ta laiendab nende tuntud tehnoloogiate võimalusi.

Microsoft Data Access Components (MDAC) komponendid teevad võimalikuks UDA kasutamise. Nendeks komponentideks on *ActiveX Data Access* (ADO), *Remote Data Service* (RDS, varem tuntud *Advanced Database Connector* (ADC) nime all), OLE DB ja *Open Database Connectivity* (ODBC). Järgmine pilt illustreerib MDAC komponentide koostööd:



Joonis 3.1 Microsoft Data Access Components komponentide töö

ADO ja OLE DB tööd Internetis illustreerib järgmine pilt:



Joonis 3.2 ActiveX Data Objects ja OLE DB töö internetirakendustes

3.1 OLE DB

Tänapäeva ärilahendused ei koosne ainult üksikutest lauaarvutitest ega neis töötavatest rakendustest, vaid tänapäevane rakendus ühendab endas lauaarvuteid, suurarvuteid ja Interneti tehnoloogiaid. Erinevate kasutuses olevate andmehoidlate hulk ja neis sisalduvate andmete erinevad kasutusviisid nõuavad, et rakendused suudaksid kasutada nii vana lähenemisviisi (üksikarvuti), kui uut (laivõrkude kasutamine). Selle tulemusena tekkis nõue lihtsalt kasutatava ja hallatava andmeühenduse süsteemi jaoks. OLE DB ongi spetsifikatsioon, mis kirjeldab hulga andmetega suhtlemiseks vajalikke liideseid, mis lubavad kasutada suurel hulgal suvalist tüüpi või suurust omavaid andmehoidlaid, seejuures garanteerides liideste koostöö. OLE DB on Microsofti poolt kirjeldatud allikates [21] ja [22].

OLE DB liidesed moodustavad omaette andmesuhtluse harustandardi, mis tagab erinevate andmete kooskõla ja koostalitusvõime. OLE DB on arenenud tavalisest andmeühendusest kaugemale paigutades kogu relatsiooniliste andmebaasidega suhtlemiseks vajaliku funktsionaalsuse loogilistesse komponentidesse luues neile komponentidele ka suhtlemiseks vajalikud sündmused (*events*). Neid liideseid kasutades on võimalik luua väga lihtsaid andmetarnijaid (*data providers*), samas on võimalik luua väga keerulisi relatsioonilisi andmebaasisüsteeme. Tänu sellele saab luua selliseid komponente, mis paistavad tavaliste tabelitena, kuid tegelikult võib kõige selle taga olla väga keeruline funktsionaalsus.

3.1.1 OLE DB omadused ja kasulikkus

OLE DB omab mitmeid omadusi, mis teevad selle kasutamise väga mugavaks ning muudab OLE DB väga kasulikuks:

- **Ligipääs kõikidele andmetele sõltumata andmete formaadist või asukohast.** See võimaldab kasutada mitte ainult andmebaase, vaid ka näiteks failisüsteemi saab vaadelda andmetena; samas ei pea andmed enam asuma samas arvutis, isegi mitte kohalikus võrgus.
- **Lihtsam programmeerida.** Kui kasutada OLE DB omadusi, ei pea iga uue andmeformaadi jaoks lisatööd tegema, kuna OLE DB andmetarnijad teevad selle töö programmeerija eest.

- **Andmekesksed komponendid.** OLE DB komponendid võimaldavad suhelda teiste komponentidega, samas ise ainult andmetega tegeledes.
- **Komponendid võivad käituda virtuaalsete tabelitena.** Enamus graafilisi arendusvahendeid loevad andmetabeleid automaatselt, samas komponentidest andmete saamine võib osutuda keeruliseks, OLE DB komponendid on võimelised end esitlema kui tabel-kujul andmeid ja arendusvahenditel on neist lihtne aru saada.
- **Integreeritus Microsofti toodetega.**
- **Täielik integreeritud ODBC-ga.** OLE DB-d kasutavad vahendid omavad täielikku juurdepääsu ODBC draiveritele ja andmetele.

3.1.2 OLE DB ei ole ODBC asendus

ODBC-d peetakse heaks relatsiooniliste andmebaasidega suhtlemise vahendiks, aga kui tegemist ei ole SQL-il põhinevate andmetega, siis võib ODBC võimalustest väheseks jääda. Samas on OLE DB andmetarbijatele loodud kõik võimalused suhtlemaks ODBC tarnijatega. Järgnevad punktid aitavad valida kumba tehnoloogiat kasutada:

- kui soovitakse kasutada standardseid relatsioonilisi andmebaase, on ODBC parim lahendus;
- kui soovitakse luua andmeliides mitte-SQL andmetele, on OLE DB parim lahendus;
- kui programmeeritakse OLE keskkonnas, on OLE DB parim lahendus;
- kui soovitakse luua andmebaasi komponente, on OLE DB ainus lahendus.

Kahe liidese tehnilised erinevused on toodud järgmises tabelis:

ODBC	OLE DB
Andmeühendus	Andmebaasi komponendid
Programmeerimiskeele (C) tase	Komponendid (*Component Object Model* – COM)
SQL-põhised andmed	Tabelipõhised andmed
SQL-põhised standardid	COM-põhised standardid
Andmebaasipõhised tarnijad	Komponent-arhitektuur

Tabel 3.1 OLE DB ja ODBC tehniline võrdlus

3.1.3 Milleks kasutatakse OLE DB-d?

Tarkvara loojad, kes kasutavad OLE DB võimalusi loovad nelja tüüpi süsteeme: andmetarnijad (*data providers*), andmetarbijad (*data consumers*), andmeteenuste tarnijad (*data service providers*) ja ärikomponendid (business components).

Andmetarnijad võivad olla suvaliste andmete edastajad rakendusele. Andmeteks ei pea olema relatsioonilised andmebaasisüsteemid (nendega suhtlemiseks võib kasutada ka ODBC-d), andmeteks võib kasutada kõike alates väga lihtsatest andmeformaadid nagu logifailid ja lõpetades väga keeruliste süsteemidega nagu näiteks e-maili süsteemid ja ADABAS süsteemid. OLE DB võimaldab luua ka kirjete (kirjehulga) liideseid andmetele, mis teeb andmetarnija kasutamise lihtsamaks.

Andmetarbijad on rakenduste osad, mis vajavad oma tööks andmeid, siin sisalduvad ka arendusvahendid, programmeerimiskeeled jms. Kõik need rakendused kasutavad mingil moel andmeid, tavaliselt kasutatakse andmete saamiseks andmetarnijaid.

Andmeteenuste tarnijad kujutavad endast andmebaasi komponente nagu päringu- või kursormootor (*cursor engine*), mis on iseseisvad tooted ja suudavad samal ajal suhelda olemasolevate OLE DB andmetarnijatega.

Sel ajal kui rakendused laienevad laivõrkudesse, muutuvad üha tähtsamaks korduvakasutusega **ärikomponendid**, mis võivad asuda nii kohalikus kui kaugarvutis. Korduvkasutusega komponentidesse koondatakse funktsionaalsus, mida on vaja mitmes rakenduses – igas rakenduses pole mõtet sama funktsionaalsust uuesti looma hakata, piisab olemasoleva komponendi kasutamisest. OLE DB ärikomponendid koondavad endas põhiliselt andmetega suhtluseks vajaminevat funktsionaalsust.

Hetkel eksisteerivatest OLE DB toodetest annab ülevaate lisa 2; kõige värskemad andmed võib leida allikast [23].

3.1.4 Kus kasutatakse OLE DB liideseid?

OLE DB liidesed on kasulikud kõikidele tarkvara loojatele, kelle tooted kasutavad mingil moel mingeid andmeid. Järgnevas on toodud vaid mõned näited, kus OLE DB osutub kasulikuks:

- **Ärimaastikul** tehakse pidevalt arvutusi, milles on vaja koguda andmeid mitmetest kohtadest – andmeid on vaja saada kiiresti, neid on vaja kiirelt analüüsida, uusi tehinguid on vaja väga kiirelt sooritada.
- **Tööstuses** kasutatakse süsteeme (CAD/CAM), mis garanteerivad, et teatud osad on alati kõige uuemad. Selleks on vaja süsteemil pidevalt suhelda nende osade tootjate andmebaasidega.
- **Kindlustuses** on tähtis, et kindlustusagendid oleksid pidevalt ühenduses oma peaserveriga. Ka siis, kui nad viibivad kliendi juures (oma kontorist eemal) – sellisel juhul on väga tähtis andmete edastuse kiirus ja samas ka turvalisus.
- **Kõikjal** kasutatakse **e-maili süsteeme** kui kriitilisi andmeallikaid (*data source*) – süsteemist saab filtreerida andmeid (maile), neid grupeerida jne.

3.2 ADO

ActiveX Data Objects (ADO) on Microsofti strateegiline kõrgtaseme liides erinevate andmetega suhtlemiseks. Täieliku ülevaate ADO-st ja tema kasutusvõimalustest annavad allikas [24] ja [25].

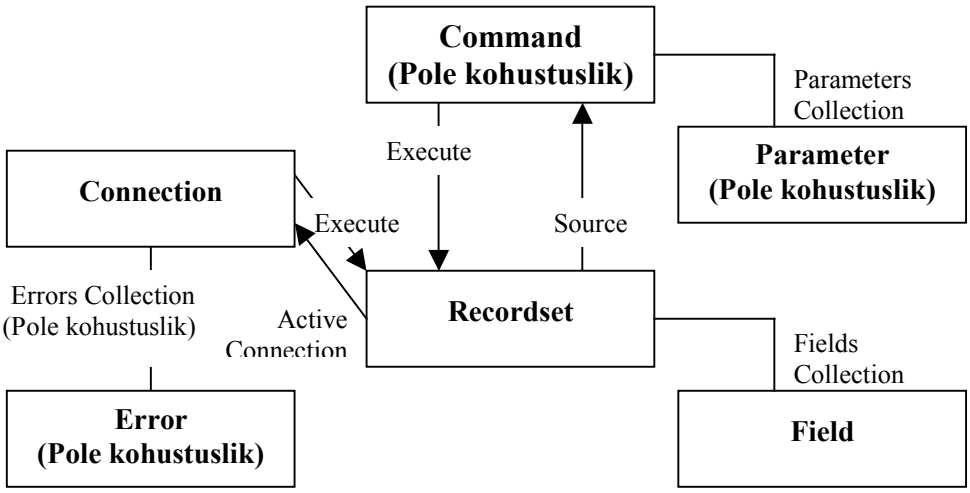
3.2.1 ADO ülevaade

ADO võimaldab kokkuhoidlikku kõrgtasemelist andmete juurdepääsu, mis omakorda lubab luua lõppkasutaja andmebaasi kliente ja äriobjekte mitmekihilises infosüsteemis kasutades selleks erinevaid rakendusi, vahendeid, programmeerimiskeeli või kasvõi Interneti brauserit. ADO on ainus andmebaasi liides mida on vaja tunda loomaks ühe kuni mitmekihilisi klient/server või veebipõhiseid andmebaasi lahendusi. ADO peamised eelised on kiirus, lihtsus, vähene mälu- ja kettaruumi vajadus.

ADO omadused, mis võimaldavad luua klient/server ja veebi süsteeme, sisaldavad järgnevat:

- Sõltumatult loodavad objektid. Kui tavaliselt tuleb objekte (ODBC ühendus, päring jt.) luua kindlas järjekorras, siis ADO võimaldab iga sellist objekti (va. vea- ja väljaobjekt) sõltumatult luua. See võimaldab kokku hoida programmeerimise aega ja ka suurendab rakenduste efektiivsust.
- Uuendatavaid andmeid võimaldatakse hoida (koguda) lokaalselt, et siis kõik korraga serverile saata.
- Eelnevalt defineeritud sisend-väljundandmetega protseduuride toetus.
- Erineva iseloomuga päringuobjektide loomine.
- Päringuparameetrite toetus (nt. kindla ridade arvuga päring).
- Mitme kirjehulga objekti (*Recordset*) tagastuse toetus eelsalvestatud protseduurides.
- Eraldiseisvad (ja töötavad) objektid rakendustes.

ADO võimaldab lihtsalt kasutada andmetega suhtlemiseks vajalikke objekte. Järgmisel joonisel on toodud ADO objektid ja nendevahelised suhted:

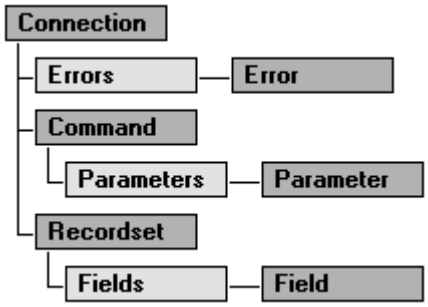


Joonis 3.3 ActiveX Data Objects objektid ja nendevahelised suhted

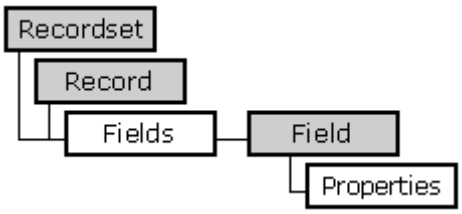
Nagu jooniselt selgub, pole kõik objektid kohustuslikud. Seejuures pole tegelikult ka näiteks *Connection* objekt kohustuslik, mis tähendab, et *Recordset* objekti võib luua kasutades *Connection* objekti meetodit *Execute*, kuid samas saab *Recordset* objekti luua ka ilma esmalt *Connection* objekti loomata (andes kogu ühenduse loomiseks vajaliku info kirjehulga objekti avamisel parameetrina).

3.2.2 ADO objektid

Kuigi ADO objektid on sõltumatult loodavad ja kasutatavad, eksisteerivad nende vahel siiski hierarhilised suhted:

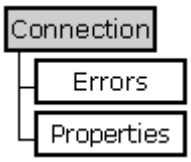


Mõnel objektil on määratud kogumid, meetodid ja atribuudid (*properties*):



ADO defineerib mitmeid objekte võimaldamaks ühendust andmebaasidega, päringute ja muudatuste tegemist andmebaasis jne. Järgnevas antakse ülevaade ADO objektidest ja nende omavahelistest seostest. Iga objekti juures on näidatud selle objekti kogumid ja kui objekt sõltud mõnest objektist, siis ka see.

Ühenduse objekt (*Connection*) esindab unikaalset sessiooni (ühendust) andmetega (andmebaasiga).

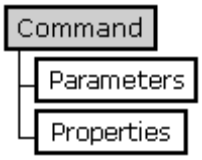


Tavalise klient-server süsteemi puhul võib ühenduse objekt esindada reaalset võrguühendust kliendi ja serveri vahel. Ühenduse objekt võimaldab:

- määrata ühenduseks vajalike atribuutide väärtused enne ühenduse loomist;
- määrata, kus luuakse kursorid (*cursor*) – kas serveris või kliendi pool;
- määrata vaikimisi valitavat andmebaasi;
- määrata transaktsioonide isoleerituse taset;
- määrata andmeallika tarnija (*OLE DB provider*);
- luua ja katkestada ühendust andmetega;

- täita käske (käsuobjekt) – teha päringud, muuta andmeid, käivitada salvestatud protseduure jne; kui käsk tagastab andmekirjeid, siis luuakse ja tagastatakse kirjehulga objekt;
- hallata transaktsioone: alustada ja lõpetada transaktsioon, luua transaktsioone üksteise sisse (kui andmetarnija seda võimaldab);
- tagastada vigade kogumi (*Errors*).

Käsuobjekt (*Command*) esindab käsku (päring, lause), mida saab rakendada andmeallikale.



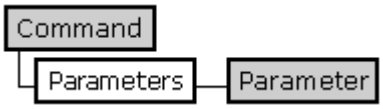
Käsuobjekt võimaldab:

- defineerida käsu teksti (näiteks SQL lauset);
- defineerida parameetritega päringuid või salvestatud protseduuride väljakutseid kasutades *Parameters* kogumit ja *Parameter* atribuute;
- käivitada käske, mis tagastavad kirjeid – selle tulemusena luuakse ja tagastatakse kirjehulga objekt;
- määrata käsu tüüpi optimiseerimaks käsu täitmist - nt tüüp võib määrata, et käsk on salvestatud protseduur või tagastab kõik read tabelis;
- määrata, kas käsk salvestatakse enne täitmist – käsk kontrollitakse (kompileeritakse) enne täitmist ja salvestatakse; see toiming võib küll esimest käsu täitmist aeglustada, kuid kui käsku täidetakse mitmeid kordi, siis järgmised täitmised on tunduvalt kiiremad;
- määrata käsu täitmise aegumist (aeg, mille jooksul peab käsk saama täidetud või sellest loobutakse);
- siduda käsuobjekti ühenduse objektiga;
- määrata objektile nimi, mille abil ühenduse objekt saab käsuobjekti välja kutsuda;
- saata käsuobjekti kirjehulga objektile saamaks andmeid.

Käsuobjekt pole tegelikult kohustuslik, kuna kirjehulga objekt on võimeline päringuid teostama kasutades SQL lauseid. Kui aga soovitakse käsu salvestamist või

kasutada parameetreid, siis on käsuobjekt vajalik. Samas ei toeta mõned andmetarnijad käsuobjekte.

Parameetri objekt (*Parameter*) esindab parameetritega päringut või salvestatud protseduuri esitava käsuobjekti parameetrit või muutujat. Parameetri objekt on kasutatav läbi *Parameters* kogumi.

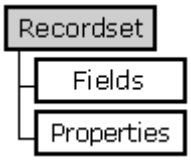


Paljud andmetarnijad toetavad parameetritega käske – need on käsud, kus tegevus defineeritakse ainult ühe korra, aga muutujate (või parameetritega) muudetakse käsku (nt saab parameetriga määrata päringu tingimuse või sorteeritava välja). Parameetrid võivad olla ka salvestatud protseduuride sisend-väljund muutujad või tagastatavad väärtused. Parameetri objekt võimaldab:

- luua parameetreid;
- lisada parameetreid *Parameters* kogumisse;
- seada või tagastada parameetri nime;
- seada või tagastada parameetri väärtust;
- seada või tagastada erinevaid parameetrit iseloomustavaid argumente (nt *Type* – tüüp, *Size* – suurus);
- edastada suuri andmemahte parameetritele.

Sõltuvalt andmetarnija funktsionaalsusest võib osa parameetri objekti funktsionaalsusest puududa.

Kirjehulga objekt (*Recordset*) esindab kogu päringu tulemusena tagastatud kirjete hulka. Samas on võimalik korraga vaadelda ainult ühte konkreetset kirjet.



Kirjehulga objekt on ADO põhiline objekt, millega toimub töö andmetega. Kõik Kirjehulga objektid koosnevad kirjetest (*Record*, tabeli rida) ja väljadest (*Field*, tabeli veerg). Kirjehulga objekt esindab andmebaasi kursorite kogu funktsionaalsust ja sellest tulenevalt on objekt ka ADO keerulisim objekt. ADO defineerib neli kursori tüüpi:

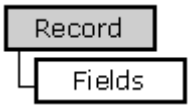
- ***Dynamic* kursor** lubab näha objekti tagastatud andmete muutusi ja lisamisi, mis toimuvad sel ajal, kui kirjehulga objekt on avatud; lubatud on ka erinevad liikumised kirjete vahel; samuti lubatakse järjehoidjast (*bookmark*) sõltuvaid liikumisi, kui andmetarnija seda võimaldab.
- ***Keyset* kursor** käitub nagu *Dynamic* kursor, ainuke erinevus on, et lisatud ja kustutatud kirjetele juurdepääs puudub; kõik liikumised kirjete vahel on lubatud.
- ***Static* kursor** tagastab staatilise koopia andmetest, st mingeid muutusi andmetes ei kajastu; kõik liikumised kirjete vahel on lubatud.
- ***Forward-only* kursor** lubab ainult edaspidi liikumist kirjetes; andmete muutusi ei ole võimalik näha.

Kursori tüüp määratakse kas enne andmete lugemist või antakse vastav parameeter andmete lugemisega koos (kui seda tehtud pole, avatakse *Forward-only* kursor). Mõned andmetarnijad ei toeta kõiki kursori tüüpe.

Kirjete lugemisel on vajalik määrata andmeallikas, selleks võib kasutada ühenduse objekti, kuid mõnede andmetarnijate puhul ei ole ühenduse objekti määramine kohustuslik – sellisel juhul määratakse andmeallikas ja tarnija kirjete lugemise meetodi väljakutsel. Tegelikult ADO loob sel juhul ka ühenduse objekti, kuid seda pole võimalik eraldi kasutada, seega kui samal andmeallikal luuakse ja kasutatakse mitmeid kirjeobjekte, siis on siiski soovitav luua eraldi ühenduse objekt.

Ka käsuobjekti kasutamine andmete lugemiseks ei ole kohustuslik, selle asemel võib SQL lause määrata kirjete lugemise meetodi väljakutsel.

Kirjeobjekt (*Record*) esindab konkreetset kirjet (rida) kirjehulga objektis.

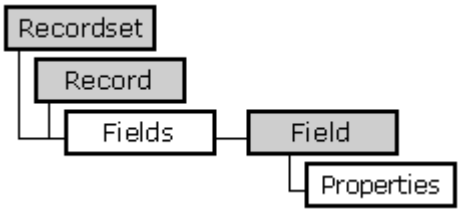


Kirjeobjekt võimaldab:

- seada või tagastada kirjega seotud ühenduse objekti;
- määrata/küsida kirje kasutamise õigusi;
- määrata/küsida kirje staatust;

- määrata/küsida kirje tüüpi;
- luua või sulgeda ühendust andmeallikaga.

Väljaobjekt (*Field*) esindab kirje välja (veergu) kirjehulga objektis. Väljaobjekt on kasutatav läbi *Fields* kogumi.

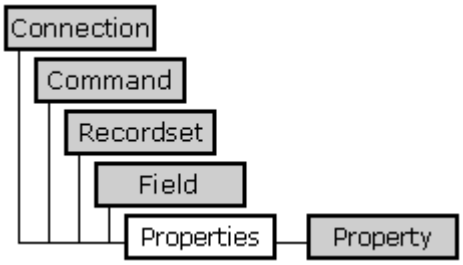


Väljaobjekt võimaldab:

- tagastada välja nime;
- vaadata või muuta väljaga esitatavaid andmeid;
- tagastada välja iseloomustavaid parameetreid (nt Type – välja andmetüüp);
- tagastada välja andmebaasis määratud suurus – sellest suuremat väärtust ei saa välja salvestada;
- tagastada välja tegelikku suurust;
- teha kindlaks, millist funktsionaalsust konkreetne väli võimaldab (*Attributes* atribuut ja *Properties* kogum);

Sõltuvalt andmetarnija funktsionaalsusest võib osa väljaobjekti funktsionaalsusest puududa.

Atribuudiobjekt (*Property*) esindab ADO objekti iseloomustavaid (dünaamilisi) atribuute. Atribuudiobjekt on kasutatav läbi *Properties* kogumi.

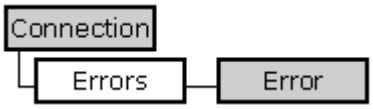


ADO objektid omavad kahte tüüpi atribuute: sisse-ehitatud ja dünaamilisi. Sisse-ehitatud atribuudid on defineeritud ADO-s ja kohe kasutatavad ja need ei ole esindatud atribuudiobjekti poolt, st nad pole kasutatavad *Properties* kogumi kaudu. Dünaamilised atribuudid defineeritakse andmetarnija poolt ja nad on kasutatavad *Properties* kogumi kaudu.

Atribuudiobjektil on neli sisseehitatud atribuuti:

- *Name* on atribuuti tekstiline identifikaator;
- *Type* on täisarv, mis tähistab atribuudi tüüpi;
- *Value* sisaldab atribuudi väärtust;
- *Attributes* sisaldab atribuuti iseloomustavaid parameetreid.

Veaobjekt (*Error*) sisaldab andmeallika või tarnija poolt tagastatud veateadet. Veaobjekt on kasutatav *Errors* kogumi kaudu.



Iga ADO objektil (või selle poolt) teostatav operatsioon võib tagastada vigu – kõik sellised ühe operatsiooni tulemusena tekkivad vead pannakse *Errors* kogumisse. Uue vea tekkimisel kogum tühjendatakse ja täidetakse uute veaobjektidega. Veaobjekt sisaldab ainult andmetarnija vigu, mitte ADO objektide vigu (mis tuleb lahendada programselt). Veaobjekt sisaldab atribuute, mis kirjeldavad viga täpsemalt; vea kohta on võimalik teada saada:

- veateksti;
- veale spetsiifilist numbrit;
- vea esilekutsunud objekti;
- SQL vigu.

Errors kogum on defineeritud ühenduse objektil, seega peab korrektseks veatöötluseks olema defineeritud ka ühenduse objekt.

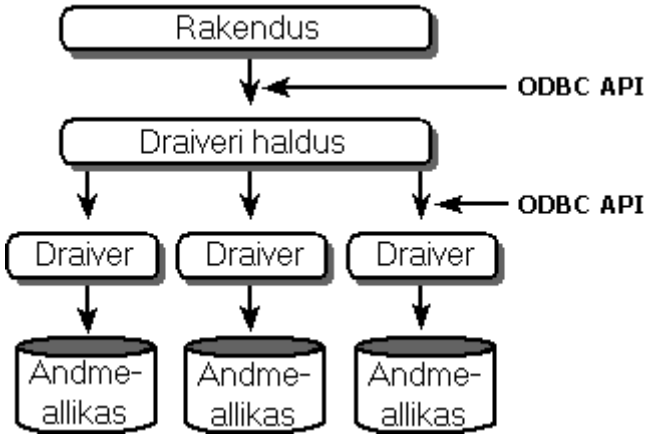
3.3 ODBC

Open Database Connectivity (ODBC) on laialt kasutatav programmeerimisliides (*Application Programming Interface* – API) andmebaasidega suhtlemiseks. ODBC põhineb X/Open ja ISO/IEC käsu-taseme liideste (*Call-Level Interface* – CLI) spetsifikatsioonil andmebaaside programmeerimisliideste jaoks ja kasutab struktuurpäringukeelt (*Structured Query Language* – SQL) andmebaasidega suhtlemiseks. Põhjalik ülevaade ODBC-st on leitav allikatest [26] ja [27].

ODBC on loodud sellisel kujul, et maksimaalselt vähendada programmeerimistööd – sama rakendus võib suhelda erinevate andmebaasisüsteemidega kasutades sama koodi. Rakendused kasutavad ODBC liidest, mis omakorda kasutab andmebaasi-spetsiifilisi draivereid. Selline lähenemisviis eraldab rakendusest andmebaasispetsiifilised funktsioonide väljakutsed – samuti nagu printeri draiver eraldab tekstitoimeti töö printeri-spetsiifiliste funktsioonide väljakutsetest. Selleks, et kasutada uut andmebaasisüsteemi rakenduses, pole vaja teha muud, kui installeerida vastav draiver, rakenduses muutuste tegemine pole vajalik.

3.3.1 ODBC arhitektuur

ODBC arhitektuur koosneb neljast komponendist: rakendus, draiveri haldus (*driver manager*), draiver ja andmeallikas. Järgnev pilt illustreerib nende nelja komponendi suhteid:



Joonis 3.4 Open Database Connectivity arhitektuur

Nagu jooniselt selgub, võib eksisteerida mitu andmeallikat, seega saab rakendus kasutada korraga mitut andmebaasi. ODBC API-t kasutatakse kahes kohas: alguses rakenduse ja draiveri halduse vahel ning seejärel draiveri halduse ja draiverite vahel. Draiveri halduse ja draiveri vahelist liidest nimetatakse ka teenuse tarnija liideseks (*Service Provider Interface*).

Rakendus on tavaline programm, mis kasutab ODBC liidest andmetega suhtlemiseks. Rakendusi võib üldistades liigitada kolmeks:

- Üldised rakendused on ette nähtud töötama mitme andmebaasiga korraga. Näiteks võib tuua tabeltöötlusprogramme, mis vajavad andmeid, et neid siis omakorda analüüsida.
- Vertikaalsed rakendused täidavad ühte kindlat tüüpi ülesannet nagu näiteks tellimuse sisestamine või andmete jälgimine. Need rakendused töötavad andmetega, mida kontrollivad rakenduse loojad, konkreetse kliendi jaoks töötavad need rakendused ühe andmebaasiga. Rakendused on loodud selliselt, et nad pole seotud ühegi konkreetse andmebaasiga, samas võivad nad toetada kindlat hulka (sarnase funktsionaalsusega) andmebaase.
- Tellitud rakendused on ette nähtud täitmaks kliendispetsiifilisi ülesandeid.

Kõik rakendused kasutavad siiski mõningaid üldisi meetodeid, mis defineerivad ODBC rakenduse üldise töö:

- andmeallika valimine ja sellega ühenduse loomine;
- täitmiseks mõeldud SQL lause saatmine;
- tulemuste saamine;
- veatöötlus;
- transaktsiooni lõpetamine;
- ühenduse katkestamine andmeallikaga.

Draiveri haldus on sisuliselt teek (*library*), mis haldab rakenduse ja draiveri vahelist suhtlust. Draiveri haldus eksisteerib põhiliselt selleks, et lahendada mõningaid paljudele rakendustele üldisi probleeme nagu näiteks andmeallika nime järgi õige draiveri valimine, draiverite laadimine ja eemaldamine, funktsioonide väljakutsed draiveritest. See kõik lihtsustab rakenduse tööd, kuna rakendus ei pea muretsema

millist draiverit kuna kasutada, ei pea hoolitsema draiveri laadimise/eemaldamise eest ega selle eest, et alati valitakse just õige funktsioon õigest draiverist.

Draiverid on teegid, mis realiseerivad funktsioone ODBC liideses. Iga draiver on andmebaasi-spetsiifiline, seega omab see ka ainult seda funktsionaalsust, mis on vastaval andmebaasil. Draiverid täidavad järgmisi ülesandeid:

- andmeallikaga ühenduse loomine ja katkestamine;
- draiveri halduse poolt mittekontrollitud vigade kontrollimine;
- transaktsioonide alustamine – see on rakenduse eest peidetud;
- SQL lausete saatmine andmeallikale täitmiseks; see hõlmab ka ODBC SQL-i modifitseerimist selliselt, et see vastaks andmebaasi SQL-ile;
- andmete saatmine rakenduselt andmeallikale ja vastupidi; see hõlmab ka andmeallikalt saadud andmete teisendamist rakenduse poolt määratud tüüpidesse;
- andmebaasi-spetsiifiliste vigade teisendamine ODBC vigadeks.

Draiverid jagunevad oma arhitektuuri poolest kaheks:

- Faili-põhised draiverid on draiverid, mis suhtlevad otse füüsiliste andmetega. Sel juhul koosneb draiver tegelikult kahest osast: ta on nii draiver kui ka andmeallikas – käsitleb ODBC väljakutseid ja täidab ise ka SQL lauseid.
- Andmebaasi-põhised draiverid suhtlevad andmetega läbi andmebaasimootori. Sel juhul käsitleb draiver ainult ODBC väljakutseid, SQL laused saadetakse andmebaasimootorile töötlemiseks.

Eksisteerib ka mittestandardseid ODBC draivereid, mis täidavad spetsiaalülesandeid.

Andmeallikas on lihtsalt andmete saamise koht. See võib olla fail, andmebaas relatsioonilises andmebaasisüsteemis või ka pidev andmevoog. Andmeallika eesmärgiks on koguda kogu andmevahetuseks vajalik tehniline info ühte kohta ja peita see lõppkasutaja eest. Seega kasutaja saab küll andmeid näha, kuid ei pea teadma, kus need andmed füüsiliselt asuvad ega ka seda, kuidas need kätte on saadud.

3.3.2 X/Open ja ISO CLI standard

ODBC (alates kolmandast versioonist) ühtlustub X/Open ja ISO CLI standardiga (on selle alamhulgaks). Selle ühtlustumise tulemusena on ODBC-le lisatud järgmised omadused:

- Üks suuremaid uusi omadusi ODBC 3.0-is on **deskriptorite** kasutamine. Deskriptor on andmestruktuur, mis sisaldab infot kas veerunimede kohta päringu tulemuses või SQL lause dünaamiliste parameetrite kohta. Deskriptorid võimaldavad otsest ja ühtset juurdepääsu veergudele või parameetritele. Nii veeru kui parameetri andmed on kirjeldatud kahe deskriptoriga: üks kirjeldab seisu rakenduse poolel, teine serveri poolel. Operatsioone, mis seovad (*binding*) parameetreid või päringu tulemuse veerge, nimetatakse kirjutamiseks deskriptorite piirkonda (*descriptor area*); operatsioone, mis loevad päringu tulemuse või parameetrite metaandmeid, nimetatakse lugemiseks deskriptorite piirkonnast.
- ODBC 3.0-s sisaldub kogu funktsioonide resultaadiks saadud tulem **diagnostikas**. Iga keskkond, ühendus, lause ja ka deskriptor omab diagnostika piirkonda. Selles piirkonnas sisalduvad ka kõik funktsioonide töö tulemusena tekkivad vead ning hoiatused. ODBC laiendab diagnostikat järgmiste omadustega:
    - diagnostika piirkond on laiendatav;
    - diagnostika piirkonnast andmete lugemine ei hävita neid;
    - staatuse info on järjestatud (tähtsuse või kriitilisuse järgi);
    - kui viga esineb ainult mõnes veerus, mitte kogu reas, siis märgitakse ära viga põhjustanud veerud.
- Päringu tulemuse **veeru nimed** ühtlustuvad X/Open ja ISO CLI standardiga.
- Lisatud on **uued atribuudid ja funktsioonid**.

Samas sisaldab ODBC ka mõningaid omadusi, mida pole veel standardiga vastavusse viidud:

- mitmerealise andmete võtmised päringu tulemusest;
- parameetrite massiivid;
- reakaupa sidumine;
- positsioneerimine;

- fikseeritud pikkusega järjehoidjad;
- ODBC kursorite tüübid;
- väljund ja sisend/väljund parameetrid;
- salvestatud protseduuride toetus.

Lisaks sellele, et ODBC 3.0 ühildub standarditega, sisaldab ta ka hulga uusi omadusi:

- Lisatud on uued andmetüübid nagu näiteks suured täisarvud.
- Andmetöötluse rühmitamine (*Batches*).
- Nimelised parameetrid.
- Laiendatud veateated.
- Laiendatud järjehoidjad.
- Tulemuse ja parameetrite ridade ignoreerimise võimalus.
- Muutujate sidumise laiendused.
- Infopäringute (spetsiaalfunktsioon *SQLGetInfo*) laiendused
- SQL-92 standardiga ühildumine.

4 Veeb kui infosüsteem

Veebi eesmärgiks oli algselt lihtsalt staatilise info edastamine, seejärel tekkis üha enam vajadus klientidega interaktiivse suhtlemise järele – veeb muutus dünaamiliseks. Võeti kasutusele andmebaasisüsteemid veebis näidatavate ja veebist saadavate andmete säilituseks – veebi võib vaadelda ühe andmebaasiliidesena, veebi kaudu saab kasutajale andmebaasi sisu suvalisel kujul esitada ja kasutajal on võimalus andmeid reaalajas muuta. Tänapäeva veebi võib vaadelda ka kui infosüsteemi, mis kasutab ühe komponendina andmebaasi.

4.1 Mitmekihiline infosüsteem

Infosüsteeme vaadeldakse tavaliselt mitmekihilistena, kus erinevad kihid tegelevad eri taseme töödega: andmebaasidega suhtlemisega, kliendiga suhtlemisega, nende kahe kihi omavahelise suhtlusega. Infosüsteem võib olla ühekihiline (kõik tegevused on ühes programmis koos) või näiteks kasvõi kuuekihiline (turvalisuse tagamiseks toimub andmebaaside ja kasutaja suhtlemine läbi mitme vahekihi).

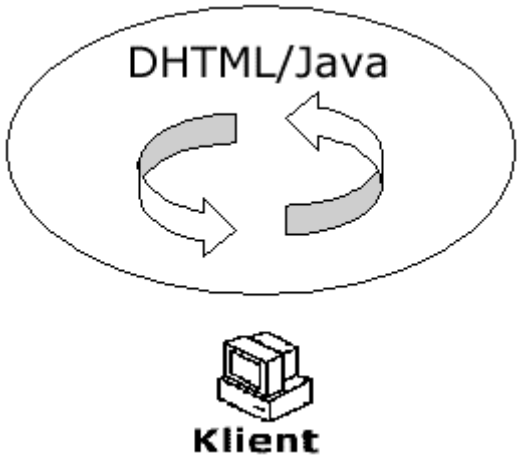
4.1.1 Ühekihilised süsteemid

Ühekihilistes rakendustes on kogu funktsionaalsus kokku kogutud ühte programmi:

1. programm suhtleb kasutajaga (kasutajaliides);
2. programm töötleb kasutajalt saadud andmeid;
3. ja programm haldab kasutajalt saadud andmeid (on ka andmebaasiks).

Veebi puhul on väga raske ühekihilisi süsteeme leida, kuid kui näiteks vaadelda kõige esimest veebi mudelit, kus kogu veeb oli staatiline, siis seda võib ette kujutada kui ühekihilist süsteemi. Samas veebiserver kui selline on sellel juhul ainult rakenduse tarnijaks. Samas kui lisada staatilisele HTML-ile mingi funktsionaalsus (dünaamilise HTMLi – DHTML – näol), siis kihte juurde ei teki, aga süsteem omandab juba ka mingi ilme. Sama efekti võib saavutada koostades Java rakendusi.

Järgmine pilt illustreerib sellist veebisüsteemi:



Joonis 4.1 Veebileht ühekihilise infosüsteemina

Sellised süsteemid on näiteks mõned veebis tegutsevad kalkulaatorid, pangad kasutavad sellist süsteemi näiteks laenutestides – lihtsad programmid, mis ei vaja oma tööks serverit.

4.1.2 Kahekihilised süsteemid

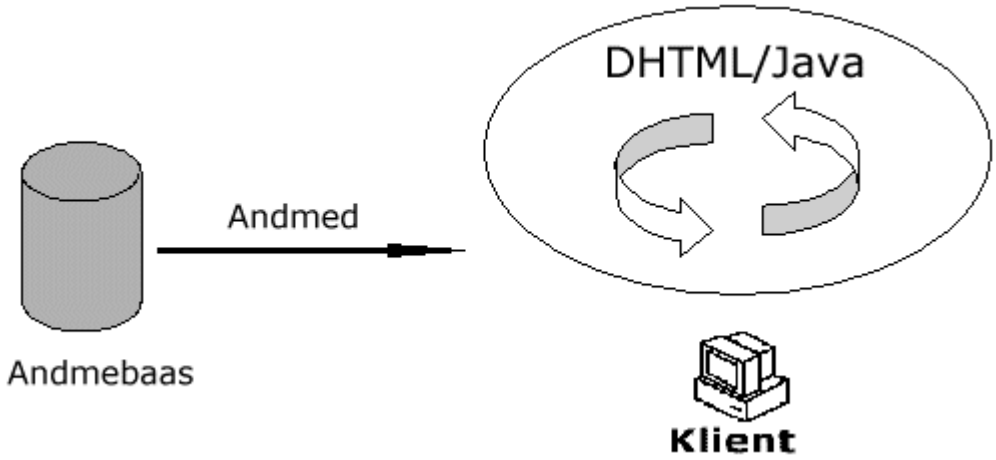
Kahekihilise süsteemi puhul eksisteerib tavaliselt mingi andmebaas ja kasutajaliides, mis suhtleb selle andmebaasiga. Seega esineb siin kaks kihti: andmebaasisüsteem haldab andmebaasi; programm suhtleb kasutajaga (kasutajaliides) ja töötleb kasutaja poolt saadetud andmeid ning vahendab neid andmebaasi.

Veebi puhul on kahekihilise süsteemi ülesehitus natuke teine. Nimelt on sel juhul veebisüsteemi töö jaotatud kaheks:

1. Andmebaas, mis teatud aja tagant moodustab oma andmetest väljundi ja paigutab selle veebi. Selleks on andmebaasisüsteemis defineeritud teatavad protseduurid andmete saamiseks ja nende veebi jaoks kujundamiseks.
2. Kasutajaliideseks on selle andmebaasi poolt väljastatud HTML, mis võib sisaldada ka näiteks dünaamilisi elemente (DHTML, Java) andmetega kliendi pool manipuleerimaks.

Veebiserveri ülesandeks on järjekordselt olla ainult vahendaja, mingit sisulist ülesannet tal pole.

Järgnev pilt illustreerib antud olukorda:



Joonis 4.2 Kahekihiline veebiinfosüsteem

Näiteks Microsoft SQL Server (alates versioonist 6.5) omab spetsiaalset liidest andmete veebis avaldamiseks (*Web Assistant*), mis lubab andmeid HTMLina avaldada ja seda protseduuri ka automaatselt teatud aegadel käivitada. Seda tüüpi süsteeme on väga kasulik kasutada olukordades, kus andmebaasidest tehakse palju sarnaseid päringuid. Siis on kasulik luua süsteem, mis teatud aja tagant saadab andmebaasi väljundi veebi staatiliste HTML lehtedena. Sellist süsteemi on soovitatav kasutada näiteks rahvaloenduse reaalajas jälgimiseks – andmebaasi koormuse vähendamiseks tehakse iga minuti tagant valmis kõik HTML lehed ja vahetatakse need veebiserveris.

4.1.3 Kolmekihilised süsteemid

Kolmekihilise infosüsteemi puhul lahutatakse kasutajaliides ja äriloogika (kogu töö andmetega on kasutajaliidesest eraldi komponenti viidud). Sellisel juhul eksisteerivad kolm kihti on järgmised:

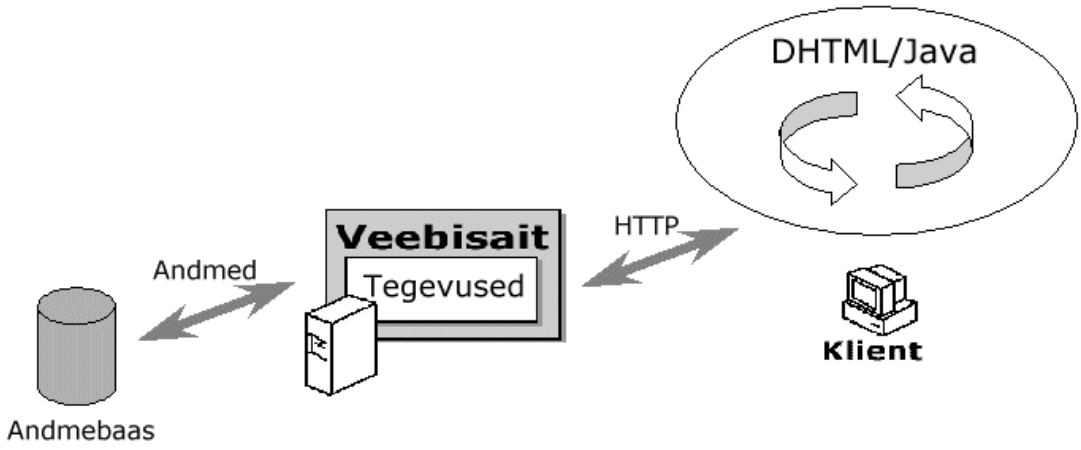
1. Andmebaasi kiht, mille ülesandeks on andmete haldamine ja ärikihile juurdepääsu võimaldamine.
2. Ärikiht sisaldab ärireeglistikku, mille alusel toimub andmebaasist andmete leidmine ja nende sinna panemine. Kihi ülesandeks on andmete vahendamine kasutajaliidese ja andmebaasi vahel. Sellesse kihti ehitatakse tavaliselt kogu süsteemi funktsionaalsus.

3. Kasutajaliides, mille ülesandeks on andmete näitamine ja muutmise jaoks liidese loomine. Kõik andmed saadetakse ja saadakse ärikihilt.

Veebi süsteemides on kolmekihiline süsteem väga tavaline. Sellisel juhul on kihtideks:

1. Andmebaas, mis haldab andmeid.
2. Veebiserver koos seal töötavate programmidega (CGI, ISAPI, serveripoolsed skriptid), mis loevad andmebaasist andmeid, kujundavad neid ja saadavad kujundatud tulemuse kliendile; lisaks loevad nad kliendi poolt saadetud andmeid ja võimaldavad nende andmete salvestamist andmebaasi.
3. Kasutajaliideseks on tavalised HTML lehed kliendi brauseris.

Järgnev pilt illustreerib kolmekihilist infosüsteemi veebis:



Joonis 4.3 Kolmekihiline veebiinfosüsteem

Kolmekihilist süsteemi kasutavad väga paljud veebisüsteemid (otsingumootorid, andmebaasid, ajalehtede veebiväljaanded, jne). Veebiserveris võib sel juhul kasutada suvalist veebi programmeerimise vahendit, tänapäevastes süsteemides toimub kliendi pool andmete kujundamine DHTMLi vahenditega vähendamaks serveripoolset koormust – sel juhul saadetakse suur hulk andmeid korraga kliendile ja seal näidatakse neid vastavalt kliendi soovidele. Sellise süsteemi puhul hajub ärikiht küll kliendi ja serveri vahele.

4.1.4 Mitmekihilised süsteemid

Kui süsteem muutub suuremaks, andmeid tekib üha enam, andmemahud suurenevad, funktsionaalsust on vaja suurendada jne võib kolmekihilisest süsteemist väheseks jääda. Kolmekihilise süsteemi korral on kogu funktsionaalsus ja andmetega suhtlemine koondatud ühte vahekihti, kui aga andmete mahud lähevad suureks, kõikide andmetega on vaja midagi teha (funktsionaalsus suureneb), siis saab vahekiht väga suure koormuse ja selle haldamine võib muutuda raskeks. Sellisel puhul luuakse tavaliselt neljakihiline süsteem:

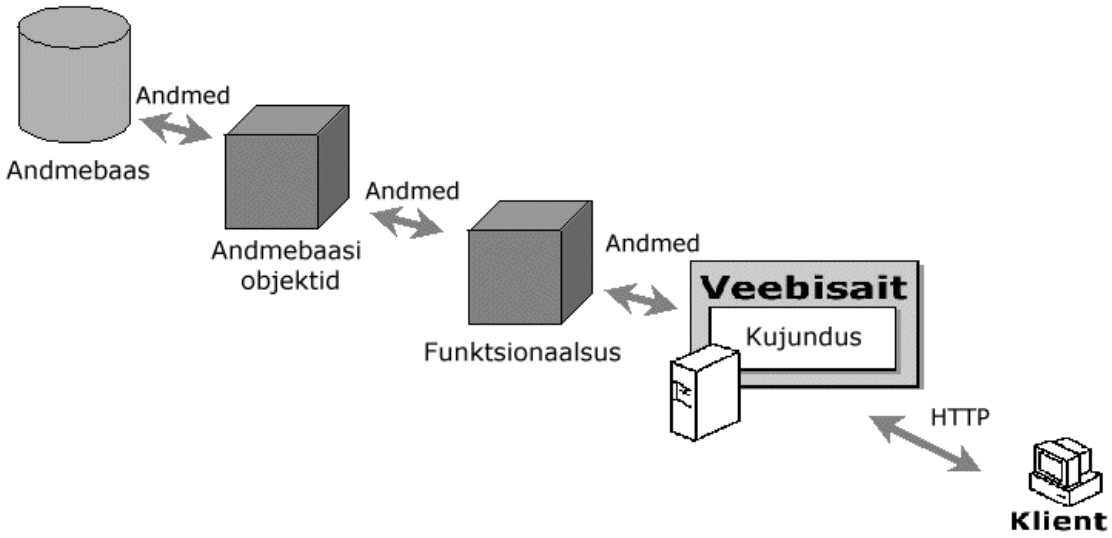
1. Andmebaas.
2. Andmebaasidega suhtlemise komponendid tegelevad ainult andmete vahendamisega andmebaasi ja ärikihi vahel.
3. Ärikiht sisaldab ainult süsteemi funktsionaalsust, seda kihti ei huvita, kus tabelis andmed reaalselt asuvad ja kuidas nad andmebaasi ja sealt tagasi saavad, tema töötleb andmeid ja saadab (või saab) need andmebaasi komponentide kihile ning teiselt poolt saadab andmed otse kliendile.
4. Kasutajaliides.

Veebis leiab ka selline süsteem küllaltki tihti kasutamist. Näiteks *Active Server Pages* (ASP) keskkonna korral kasutatakse tavaliselt andmetega suhtlemiseks *ActiveX Data Objects* (ADO) objekte, seega on siis ASPi puhul neli kihti võimalik jaotada järgmiselt: Andmebaas, ADO, ASP veebiserveris, brauser.

Veebi puhul leiab kasutamist ka viiekihiline süsteem. Nimelt kui soovitakse lahku lüüa äriloogika ja veebi disainimine, siis luuakse eraldi kiht, mis tegeleb süsteemi funktsionaalsusega. Kihid on sel juhul:

1. Andmebaas.
2. Andmebaasidega suhtlemise komponendid.
3. Ärikiht tegeleb süsteemi funktsionaalsuse realiseerimisega.
4. Kujunduse kiht tegeleb ärikihi poolt tagastatud andmete kujundamisega ja veebi saatmisega ning kasutajalt saadud info ärikihile vahendamisega.
5. Kasutajaliides.

Järgmine pilt illustreerib seda süsteemi:



Joonis 4.4 Viiekihiline veebiinfosüsteem

Lisaks neile kihtidele võivad mõned spetsiifilised süsteemid (näiteks panga infosüsteem) nõuda spetsiaalset lisakihti (või ka mitut) turvalisuse tagamiseks (kasutajate tuvastamine, õiguste kontroll, kasutajategevuste logimine).

4.2 Milline vahend valida?

Millist veebiinfosüsteemi siis luua? Ja millest see koosnema peaks? Kas kasutada andmebaasi? Kui, siis kui palju kasutada andmebaasi? Sellised küsimused tekivad kindlasti iga veebi süsteemi looja peas kõigepealt. Kõigepealt peab panema paika vahendid ja server-süsteemid, mida kasutatakse: andmebaasisüsteem, veebiserver, veebiserveris programmeerimise vahendid, andmebaasidega suhtlusvahendid.

Tänapäeval on **andmebaasiservereid** niivõrd palju, et valik ei tohiks mingil juhul väike olla. Kasutada võib alates väga lihtsatest süsteemidest nagu *MySQL* ja lõpetades suurte andmebaasisüsteemidega nagu *Oracle* ja *Microsoft SQL Server*. Valikul saavad tavaliselt määravateks toote hind, hallatavus ja lisavõimaluste rohkus. Näiteks MySQL süsteem on Linux keskkonnas kasutamiseks tasuta, Windows keskkonnas aga mitte. Suurte süsteemide korral nagu Oracle ja MS SQL Server tuleb tähele panna, et lisaks andmebaasiserverile tuleb osta veel ka spetsiaalne litsents andmebaasi veebirakendustest kasutamiseks (MS SQL Serveri puhul tuleb näiteks osta *Internet Connector*i nime kandev litsents). Seega võib öelda, et

andmebaasiserveri valik on täiesti vabalt valitav, kuna näiteks Windows keskkonnas on pea kõikidele andmebaasisüsteemidele olemas ODBC draiverid.

Veebiserveri valik tehakse põhiliselt isikliku meeldimise järgi. Windows keskkonnas kasutatakse väga palju *Microsoft Internet Information Server*it, kuid kes on varem töötanud UNIX keskkonnas, eelistab tavaliselt kasutada Apache serverit. Netscape serverit kasutatakse ka palju, aga selle juures on määrav asjaolu, et see server ei ole vabavara. Kõik veebiserverid võimaldavad serveri laiendamist, serveripoolseid skripte ja loomulikult CGId.

Veebi programmeerimise vahend valitakse tavaliselt harjumuse järgi. Kui veebi looja omab CGI harjumust, siis ta selle ka valib. Siin tasuks aga mainida, et CGI lahenduste kasutamise vajalikkust tuleb väga tõsiselt arutada, kuna CGI on kõige aeglasem vahend veebi programmeerimiseks. Kindlasti leidub süsteeme, kus ka CGI end õigustab, aga kui süsteem omab vähegi suuremat funktsionaalsust, tasuks kaaluda mõne ISAPI sarnase lahenduse kasutuselevõttu. Peamiseks põhjenduseks võib tuua just kiiruse; lisaks võimaldab ISAPI palju enamat kui CGI, näiteks serveri laiendamise võimalus. Ka serveripoolsete skriptide kasutuselevõtt on mõttekas. Kuigi skriptid on aeglasemad kui puhas ISAPI, on neid palju kergem luua ja hallata.

Andmebaasidega suhtlemisel Windows keskkonnas on valida kas ODBC, OLE DB, ADO või andmebaasi vahendite vahel. ODBC üheks suurimaks miinuseks peetakse ühenduse loomise aeglust. Kuid sellele puudusele võib pakkuda lahenduseks tööskeemi, kus rakenduse töö alguses luuakse vajalikud ODBC ühendused ja neid kasutatakse kogu töö käigus. Näiteks võimaldab ISAPI rakenduste loomine sellist skeemi kasutada, kuna ISAPI rakendused laetakse tavaliselt ainult ühe korra. OLE DB ja ADO on väga võimsad ja palju kasutust leidvad vahendid. Nende suurteks plussideks on mugavalt kasutatavad üldised liidesed ja objektid – andmebaasiserveri vahetusel tuleb muuta väga vähe. Andebaasiserveri vahendite kasutamise kasuks räägib asjaolu, et need on väga kiired, kuid miinuseks on see, et andmebaasiserveri vahetuse korral tuleb süsteemi uuesti töölesaamiseks väga palju lisatööd teha.

5 Eesti veebimaastik

Sel ajal kui kogu maailmas areneb veeb väga võimsalt e-äri suunas, ei jää ka Eesti veeb sellest arengust maha. Eestisse on tekkinud juba mitmeid Internetiportaale, Internetikaubamaju, enamus ajalehti omab võrguväljaannet, mõned neist ka reaalajas tegutsevat väljaannet. Lisaks eksisteerib ka mitmeid andmebaase, millele on liidetud ka veebiliides. Pea iga firma ja asutus peab oma aukohuseks omada veebilehte – kasvõi sellist, kus on ainult firma logo ja kontaktandmed, peaasi et see oleks olemas. Portaalid ja kaubamajad põhinevad loomulikult andmebaasidel, ajakirjad-ajalehed arenevad ka selles suunas. Isegi mõned firmad on oma veebi teinud andmebaasipõhiseks. Tundub, et areng on selles suunas, et veeb ei ole enam tavaline lingikogu – see on kujunemas suureks infosüsteemiks.

Järgnevas püütakse anda põgus ülevaade Eesti veebi olukorrast võttes aluseks veebiserverit poolt väljastatud infot ning kasutades ka veebiserverite haldajate/loojate käest saadud infot.

5.1 Eesti veebi hetkeolukord

Selleks et saada mingit ülevaadet Eesti veebiserveritest, ja neis kasutatavast tarkvarast sai 2000. aasta aprilli alguses tehtud väike uuring. Nimelt sai läbi vaadatud kõik Eesti serverid, mille nimes sisaldub lühend '*www*'. Kokku oli küsitluse tegemise hetkel Eestis registreeritud 3426 Interneti aadressi, mille nimes sisaldus '*www*'. Uurimuse eesmärgiks oli kõikide aadresside taga olevate arvutite veebiserveri kohta info teadasaamine. Selleks sai koostatud lühike programm, mis võttis järjest kõigi arvutite veebipordiga ühendust ja päris neilt juurkataloogi päiseinfot. Konkreetselt saadeti igale arvutile järgmine päring:

```
HEAD / HTT/1.0
Host: aadress
```

Päringus oli *aadress*i kohal iga uuritava arvuti Interneti aadress. Kõikidest arvutitest kõigest 18 kas ei töötanud või puudus nendes veebiserver – nende info kohal märgiti 'Pole teada'.

Veebiserver peab vastavalt HTTP standardile sellisele päringule tagastama moodustatud HTTP päised. Päises sisaldub tavaliselt HTTP staatus, veebiserveri tüüp, MIME infot ja lisainfo (mida veebiserver peab vajalikuks lisada). Näiteks võib väljund olla järgmine:

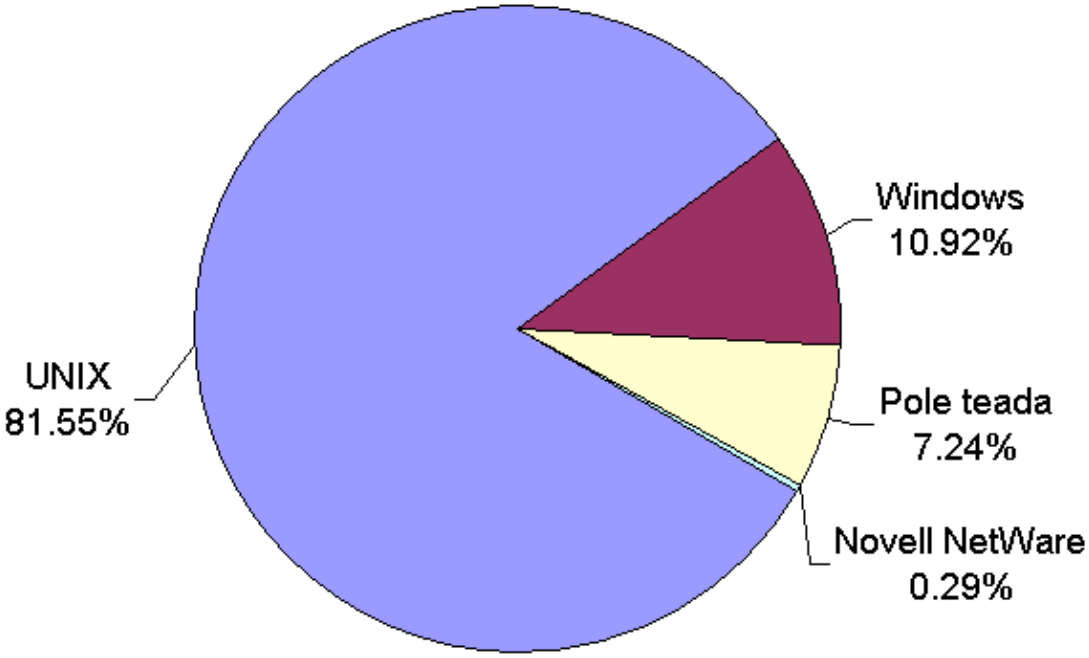
```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Fri, 12 April 2000 06:14:23 GMT
Content-type: text/html
```

Vastuseks saadud info salvestati faili. Seejärel teisendati saadud infot selliselt, et sellest jäi järele ainult analüüsimiseks vajalik info (näiteks saadavad mõned veebiserverid eirates HTTP standardit päisepäringule tagasi ka kogu väljundi – see tuli välja jätta). Peale teatavaid muutusi andmete formaadis tekkis kogu sellest infost *Microsoft Access*i ja *Microsoft Excel*i abiga järgmine tulemus, mille kohta ei saa küll öelda, et see peegeldab Eesti veebi tegelikku olukorda, kuid mingisuguse ülevaate see siiski annab.

5.1.1 Serveri platvorm

Selleks et teha kindlaks veebiserveri operatsioonisüsteemi, tuleb uurida serveri nimeinfot. Näiteks kui veebiserveri nimes sisaldub näiteks '*Win32*', '*Microsoft*', '*Savant*', '*Website*', '*Sambar*' või '*OmniHTTP*', siis võib suure tõenäosusega öelda, et tegemist on Windows-platvormiga, kuna neid sõnaosasid sisaldavad veebiserverid töötavad ainult Windows-platvormil. Kui aga veebiserveri nimes sisaldub näiteks '*UNIX*', '*FreeBSD*', '*Red Hat*', '*Linux*' või '*Debian*', võib suure tõenäosusega öelda, et tegemist on UNIX platvormiga. Samas '*Netware*' või '*NOVELL*' sisaldavate veebiserverite kohta võib öelda, et need põhinevad Novell NetWare platvormil. Kõikidel muudel juhtudel (näiteks Netscape server eksisteerib mitmel platvormil ja serveri nimesse platvormi ei kirjutata) ei olnud võimalik veebiserveri nime järgi platvormi määrata ja need märgiti kui 'Pole teada'.

Antud uurimuse tulemus on näha järgmisel joonisel:

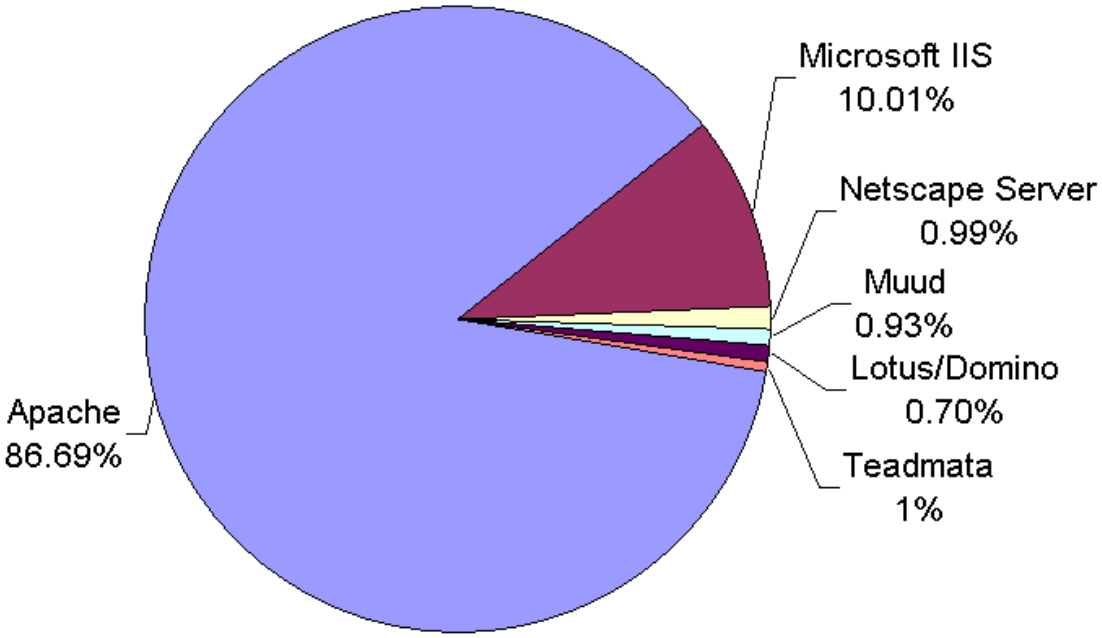


Joonis 5.1 Veebiserveri platvormide jagunemine Eesti veebis

Nagu tulemusest selgub kasutatakse suures osas UNIX platvormi. Väga suure tõenäosusega võib oletada, et tegemist on vabavaraga, mis on ka arusaadav, kuna Eestis üritatakse tavaliselt väga odavalt läbi ajada.

5.1.2 Veebiserverid

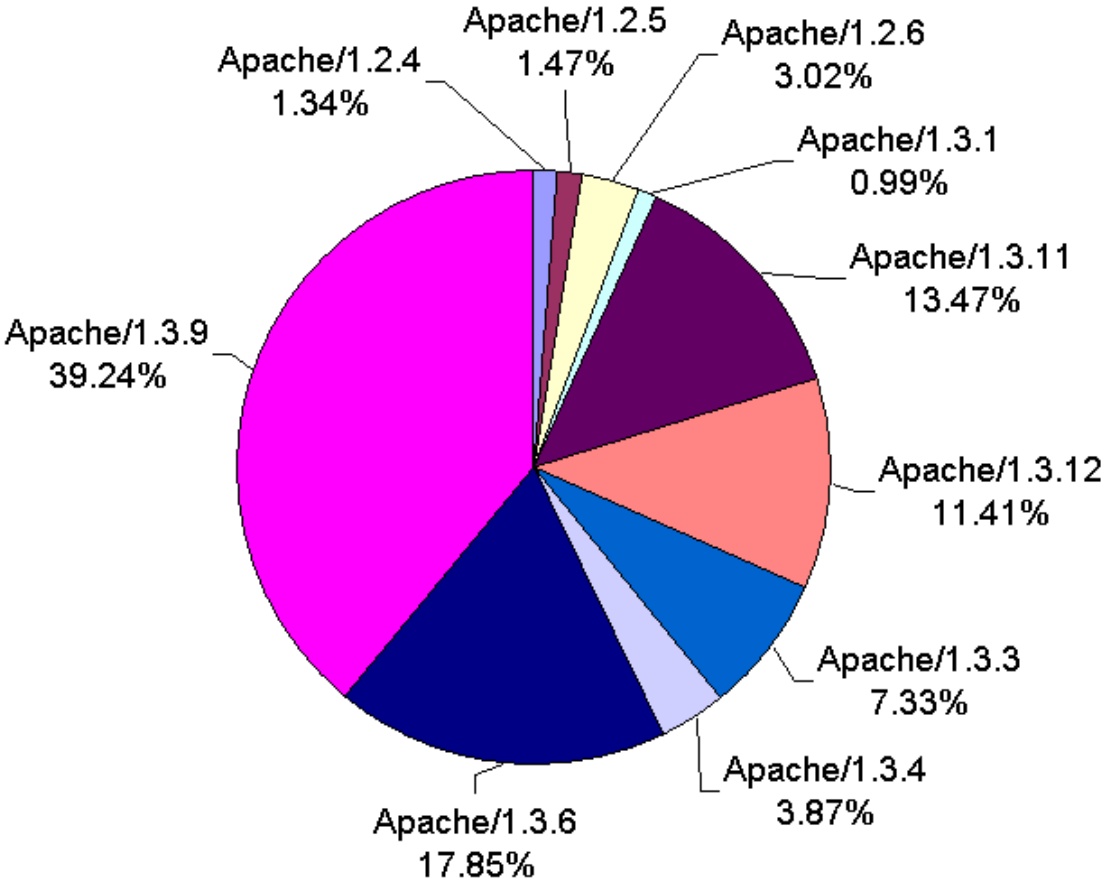
Erinevaid veebiservereid oli kokku 73, nende seas oli näiteks Apache serveri eri versioone 21. Kuna eri serverite eri versioone oli liiga palju, sai kõik versioonid liidetud üheks tervikuks. Eraldi pole välja toodud veebiservereid, mida oli alla 20 eksemplari (näiteks Oracle veebiserver). Tulemusest annab ülevaate järgmine joonis:



Joonis 5.2 Veebiserveri tüüpide jagunemine Eesti veebis

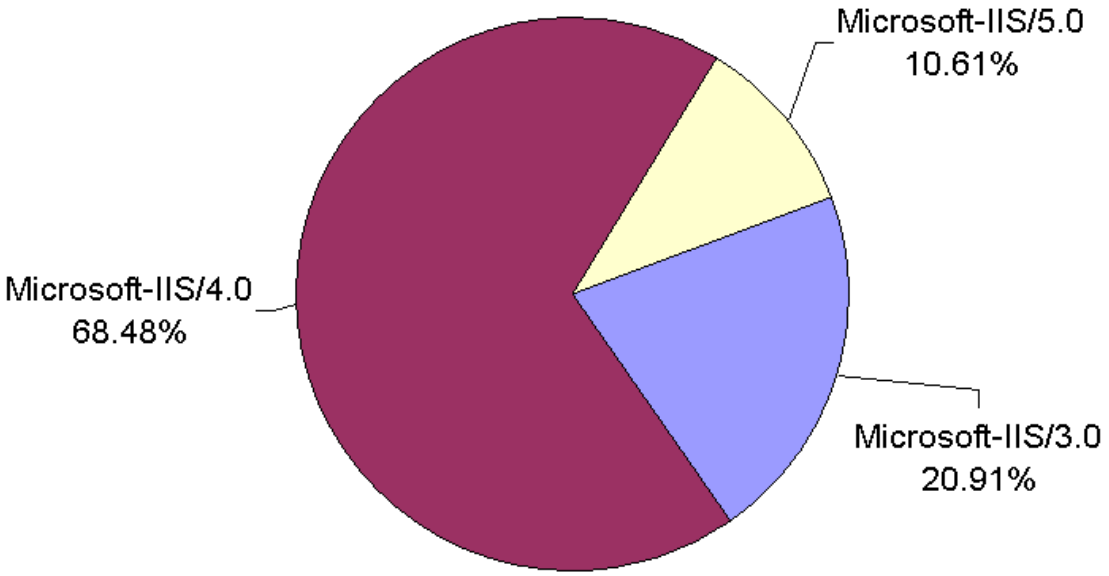
Kui võrrelda neid tulemusi näiteks firma E-Soft Incorporated ülemaailmse uurimuse Eesti osaga (vaadeldav sama aja kohta aadressil allikas [28]), siis on tulemused küllaltki sarnased.

Kui vaadelda populaarsemaid servereid eraldi, siis näiteks Apache serverid jagunesid järgmiselt (vähem kui 20 esinemisega versioonid välja jäetud):



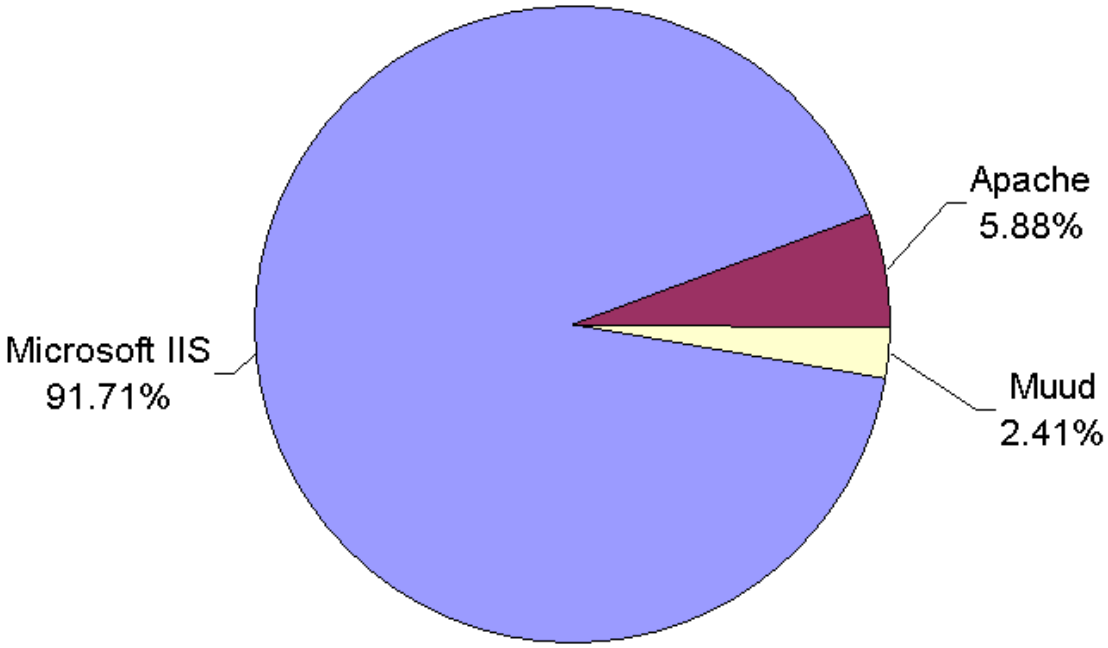
Joonis 5.3 Apache serveri versioonide jagunemine Eesti veebis

ja *Internet Information Server* versioonid järgmiselt:



Joonis 5.4 Internet Information Server versioonide jagunemine Eesti veebis

Kui aga vaadelda eraldi näiteks Windows-platvormi, siis jagunevad veebiserverid tüüpide järgi järgmiselt (hõlmatud on veebiserverid, mille nimest oli võimalik välja lugeda, et tegemist on Windows platvormiga):

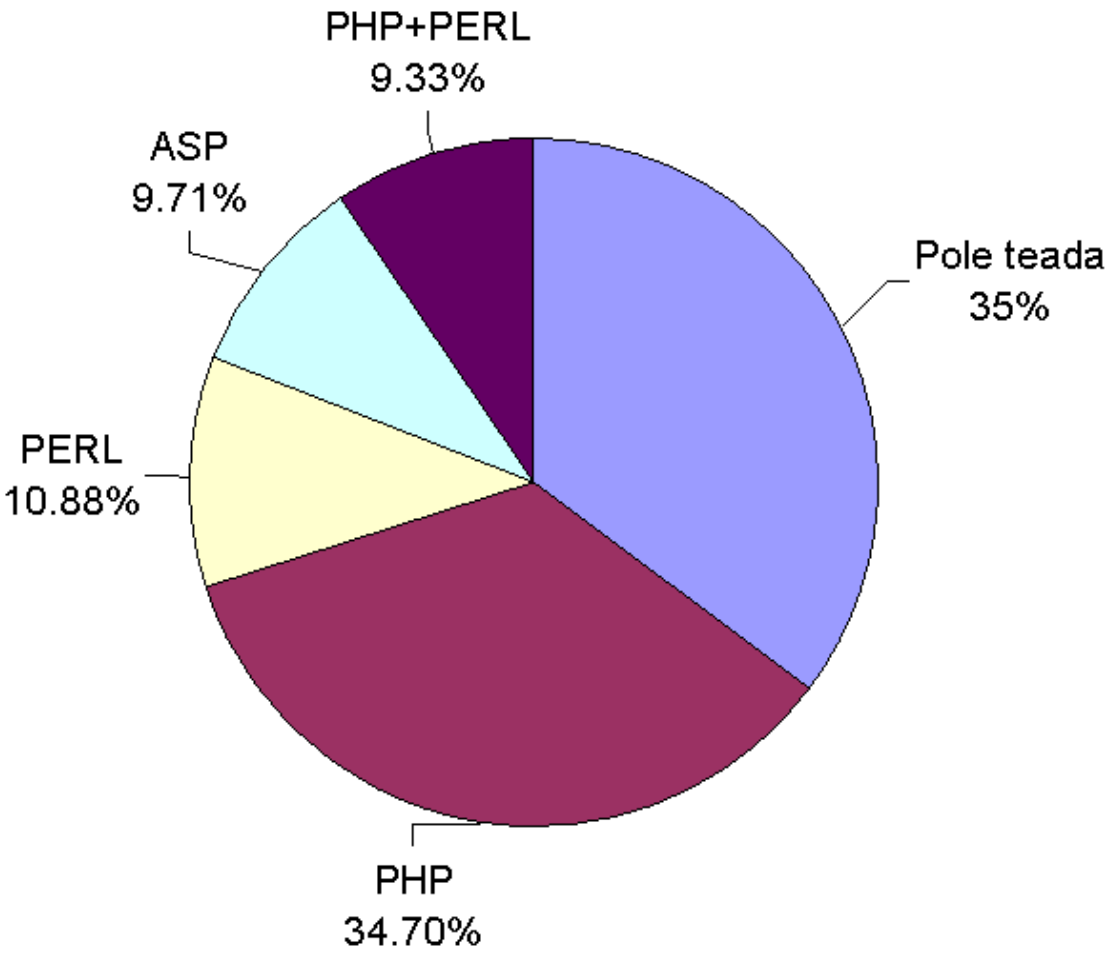


Joonis 5.5 Windows platvormi veebiserverite jagunemine Eesti veebis

5.1.3 Serveripoolsed skriptid

Veebiserveri poolt edastatava info põhjal on võimalik teha järeldusi ka selle kohta, milliseid serveripoolseid skripte on võimalik seal kasutada. Näiteks *Internet Information Server* sisaldab alates kolmandast versioonist vaikimisi *Active Server Pages* (ASP) kasutamise võimalust. Kui aga Apache serverisse on moodulitega lisatud PHP või näiteks ActivePerl, siis kajastub see tavaliselt ka serveri nimes. Samuti on võimalik serveri poolt saadetavate küpsiste järgi kindlaks teha kasutatavat skriptivahendit – sellist moodust kasutab näiteks ASP ja ColdFusion. Lisaks võib näiteks PHP lisada veebiserveri väljundisse iseloomulikke lisapäiseid. Ja otse loomulikult esines servereid kus oli võimalus kasutada mitut skriptivahendit.

Kogu selle info põhjal tekkis teatav ülevaade serveripoolsete skriptide kasutamisest. Jooniselt on jällegi välja jäetud liiga väikese esinemisega (alla 20 eksemplari) skriptivahendid (näiteks jääb siit välja ColdFusion). Väljend ‘Pole teada’ määratleb neid servereid, mille päise infost polnud võimalik skriptivahendit leida:



Joonis 5.6 Serveripoolsete skriptivahendite jagunemine Eesti veebis

5.2 Asjaosaliste arvamus

Selleks, et saada veelgi lähemat ülevaadet, sai koostatud küsitlus, mida saadeti erinevatele veebi aktiivselt kasutavatele firmadele. Küsitlusega loodeti saada arvamusi asjaga tegelejatelt. Küsitluse said eesti pangad, portaalid, Internetikaubamajad, mõned ajalehed, mõni firma ja mõne veebiliidesega andmebaasi haldaja. Küsitluse said kokku 27 veebi haldurit, tagastas selle 13, andmete avaldamisest keeldus üks. On täiesti loomulik, et need numbrid on liiga väikesed tegemaks mingeid põhjapanevaid järeldusi, kuid loodetavasti annab see küsitlus siiski mingi ülevaate.

5.2.1 Küsitluse tekst

Täieliku ülevaate huvides on järgnevas esitatud küsitluse täielik tekst (edaspidise viitamise huvides on küsimused nummerdatud):

1. Kas Teie veebis on kasutatud andmebaasi?
2. Mis platvormil kasutate andmebaasiserverit?
3. Millist andmebaasiserverit kasutate?
4. Mis platvormil kasutate veebiserverit?
5. Millist veebiserverit kasutate?
6. Kas veebiserver ja andmebaasiserver asuvad samas füüsilises serveris?
7. Milline järgnevatest on täpseim Teie veebilahenduse iseloomustamisel (lisaks tavalistele HTML-lehtedele):
   a) Serveripoolsed skriptid
   b) ISAPI/NSAPI (või mõni muu analoog – paluks märkida)
   c) CGI
8. Kui kasutate serveripoolseid skripte, siis miks?
   8.1. Kui kasutate serveripoolseid skripte, siis mida täpsemalt? (Näiteks PHP, ASP, CFM, Perl)
   8.2. Miks otsustasite just selle skript-vahendi kasuks?
   8.3. Kas valitud serveripoolne skript-vahend rahuldab Teie nõudmisi? (kui ei, siis miks)
   8.4. Kas pidite lisaks skript-vahendi poolt pakutud funktsionaalsusele ka ise midagi juurde programmeerima?
9. Kui kasutate CGI lahendust, siis miks?
10. Kui kasutate ISAPI/NSAPI lahendust, siis miks?
11. Mis vahendeid kasutate andmete saamiseks andmebaasist? (Näiteks ADO, ODBC, JDBC)
    11.1. Miks otsustasite just selle vahendi kasuks?
    11.2. Kas valitud vahend rahuldab Teie nõudmisi? (kui ei siis miks)
12. Kui vaadelda veebi mitmekihilise infosüsteemina, siis mitmekihiliseks hindate oma veebi? Kena oleks, kui tooksite ka kihid eraldi välja. Näiteks on väga tavaline kolmekihiline süsteem: 1. andmebaas; 2. programm (veebiserveris); 3. kasutajaliides (brauseris).
13. Kas Teie arvates on Teie veeb piisavalt turvaline – st kas Te ei karda andmete turvalisuse pärast? (võib ka lisada, miks)
14. Kas Teie arvates on üldse mõistlik koostada veebi andmebaasipõhisena? Miks?
15. Teiepoolne lõppkommentaar oma veebilahendusele:
16. Teiepoolne lõppkommentaar antud küsimustikule:

Peab kohe mainima, et mõne küsimuse sõnastus ei olnud võibolla kõige parem ja tekitas seetõttu teatavat segadust. Näiteks Perli paigutamine serveripoolsete skriptidega ühte oli võibolla viga, kuna Perli skriptid saavad olla ka näiteks CGI skriptid, samas ActivePerl leiab kasutust serveripoolse skriptina.

5.2.2 Tulemused

Järgnevas antakse ülevaade küsimuste kaupa neile saabunud vastustest.

1. See küsimus oli määrava tähtsusega ja sellele eitavalt vastamine oleks tähendanud edasisest vastamisest loobumist. Ükski saabunud vastustest ei olnud eitav.
2. Andmebaasiserver asub erinevatel platvormidel alustades *Red Hat Linux*ist ja lõpetades *Windows2000 Advanced Server*iga.
3. Andmebaasiserveritest oli mitmel pool mainitud *MySQL* süsteemi, kuid samas esines ka näiteks *Microsoft SQL Server* ja *Oracle* serverit.
4. Veebi platvormina kasutatakse rohkem kui pooltel juhtudel Linux distributsioone, ülejäänud juhtudel on kasutuses Windows-platvorm.
5. Veebiserverina mainiti põhiliselt *Apache* serverit, Windows platvormil aga *Internet Information Server*it.
6. Suuremate süsteemide korral asuvad andmebaasiserver ja veeb eri serverites, väiksematel juhtudel samas serveris.
7. Põhiliselt kasutatakse veebi rakenduste loomiseks serveripoolseid skripte, mõni süsteem kasutab ka ISAPI lahendust, samas aga leidub ka CGI lahendusi.
8. Serveripoolsete skriptidena kasutatakse põhiliselt PHPd ja ASPi, kuid ka ColdFusion leiab kasutamist. Skriptivahendi plussidena mainiti põhiliselt selle lihtsust ja haldamise mugavust. Enamusel juhtudel tuli lisaks standardvahenditele ka ise midagi juurde luua.
9. Kui kasutatakse CGId, siis põhiliselt kasutatakse Perl skripte ja põhjendusena tuuakse Perli mugavusi.
10. ISAPI rakenduse kasutamise põhjendusena tuuakse asjaolu, et see on standard.
11. Andmete saamiseks andmebaasist kasutatakse kas andmebaasiserveri vahendeid, *Microsoft ActiveX Data Obejcts* vahendeid või ODBCd.
12. Veebi hinnatakse väga erinevate tasemetena, seega mainitakse, et veeb võib olla kahe kuni kuuekihiline. Kahekihilise süsteemi korral peetakse üheks kihiks

andmebaasi ja teiseks kõike muud; kuuekihilise süsteemi korral toimub andmete saamine andmebaasist läbi mitme eri kihi.

13. Veebi turvalisuse hindamisel kõiguvad arvamused küllaltki palju: osa arvab, et kõik, mis on inimese poolt loodud, on ka inimese poolt murtav, teised seevastu peavad oma süsteemi väga turvaliseks.
14. Veebi loomisel andmebaasipõhisena ollakse ühisel arvamusel, et kui veeb on vähegi dünaamilist laadi, siis on andmebaasi põhine veeb ainuke lahendus.
15. Enda veebi hinnatakse tavaliselt heaks, aga mööndakse, et alati saab paremini.

Kokkuvõte

Veebiinfosüsteemid on tänapäeval kujunenud väga tavalisteks – pea iga süsteemi on võimalik veebipõhisena üles ehitada, veeb on kujunenud üheks tähtsaks (et mitte öelda tähtsaimaks) väljundiks erinevatele süsteemidele.

Et selline süsteem korralikult töötaks, on vaja see ka korralikult luua –veeb pakub selleks mitmeid vahendeid: loodud on mitmeid veebi programmeerimise vahendeid nagu CGI, ISAPI ja serveripoolsed skriptid. Kuigi CGI programmeerimine on juba peaaegu aegunud, leidub kindlasti süsteeme, kus ka seda kasutada; seega jäävad tänapäevase veebi koostamiseks ISAPI ja serveripoolsed skriptid. Kumba neist valida, on igaühe enda otsustada; kuigi ISAPI rakendused on kiiremad, on skriptivahendit mugavam ja lihtsam kasutada.

Andmebaasidega suhtlemise osas on Windows keskkonnas võimalik valida kolme vahendi vahel: OLE DB, ADO ja ODBC. Siin otseseid soovitusi anda ei saa, kuna igaüks neist võimaldab väga võimsaid vahendeid andmebaasidega suhtlemiseks.

Lisaks vahendite valikule tuleb iga infosüsteemi loomise juures mõelda mitmekihilisena süsteemi luua. Veeb luuakse tavaliselt kolmekihilisena, kuid kindlasti tasuks mõelda ka enamate kihtide loomisele – selline jaotamine on kindlasti lihtsamini hallatav ja samas ka turvalisem.

Tulevikus areneb veeb veel kindlasti, XML ja traadita võrgu arenedes suurenevad veebi kasutusvõimalused veelgi.

Web info systems

Aulis Sibola

Abstract

Today we can't imagine ourselves without web: we read newspapers from the web, we read and write our e-mails in the web, we watch movies on the web, we pay our bills on the web, we buy our clothes and even food from the web. Web has become a part of our everyday life. We have got web-stores, Internet banks, portals, search engines – we have got almost everything in the web. The web has become a part of so called e-business.

All those web systems relay on the similar architecture: somewhere there is a database and the data in this database can be viewed and as need arises even modified through the web.

The web info system can be built using one of the following techniques: CGI, ISAPI, server-side scripting. Although CGI is the oldest and slowest there are still places where it can be used. ISAPI applications and server-side scripting are the most popular techniques today and most web sites are built based on them.

Database access in the Microsoft Windows environment can be achieved by using *Microsoft Universal Data Access* components: OLE DB, ADO and ODBC – all of them are widely used and they all give developers a lot of options to work with different type of data.

Web info system can be viewed as multi tier application: there is database, there are database components, there is web programming and there is user interface.

Kasutatud kirjandus

1. Web Services, Microsoft Corporation, 2000 (http://www.microsoft.com/ntserver/web/default.asp)
2. Internet Information Server 4.0, Microsoft Corporation, 2000 (http://msdn.microsoft.com/workshop/essentials/forstarters/starts031698.asp)
3. Apache Server Frequently Asked Questions, Apache, 2000 (http://www.apache.org/docs/misc/FAQ.html)
4. iPlanet Web Servers, Sun-Netscape allians, 2000 (http://www.iplanet.com/products/infrastructure/web_servers/index.html)
5. The WWW Common Gateway Interface Version 1.1, 1999 (http://Web.Golux.Com/coar/cgi/draft-coar-cgi-v11-03-clean.html)
6. CGI Programming Unleashed, Eugene Eric Kim, 1996, (http://www.atoofi.com/doc/books/mcp/cgi-programming-unleashed/index.htm)
7. The Common Gateway Interface, NCSA, (http://hoohoo.ncsa.uiuc.edu/cgi/)
8. Special Edition Using ISAPI, 1996, (http://www.fintech.ru/Library/isapi/indextoc.htm)
9. Developing ISAPI Extensions and Filters, Microsoft Corporation, 2000 (http://msdn.microsoft.com/library/psdk/iisref/isgu6j5f.htm)
10. NSAPI Programmer's Guide for iPlanet Web Server, Sun-Netscape Alliance, 2000 (http://docs.iplanet.com/docs/manuals/enterprise/41/nsapi/contents.htm)
11. Chili!Soft Home, Chili!Soft, Inc, 2000 (http://www.chilisoft.com/)
12. Active Server Pages, Microsoft Corporation, 2000 (http://msdn.microsoft.com/workshop/c-frame.htm?/workshop/server/asp/aspover.asp)
13. Active Server Pages Guide, Microsoft Corporation, 2000 (http://msdn.microsoft.com/library/psdk/iisref/aspguide.htm)
14. Introduction to Active Server Pages, Kevin Cooke, 30 september 1998 (http://hotwired.lycos.com/webmonkey/98/39/index2a.html?tw=programming)
15. ColdFusion Product Information, Allaire Corporation, 2000 (http://www.allaire.com/Products/ColdFusion/productinformation/)

16. CFML Language Reference, Allaire Corporation, 1998 (http://www.allaire.com/documents/cf4/CFML_Language_Reference/contents.htm)
17. ColdFusion tutorial, Charles Mohnike, 2000, http://hotwired.lycos.com/webmonkey/programming/coldfusion/tutorials/tutorial2.html)
18. PHP Manual, PHP Documentation Group, 2000 (http://www.php.net/manual/)
19. DevShed – PHP, ngenuity, 2000 (http://www.devshed.com/Server_Side/PHP/)
20. Microsoft Universal Data Access - Product Information, Microsoft Corporation, 1999 (http://www.microsoft.com/data/prodinfo.htm)
21. Microsoft OLE DB, Microsoft Corporation, 1999 (http://www.microsoft.com/data/oledb/)
22. Platform SDK – OLE DB, Microsoft Corporation, 2000 (http://msdn.microsoft.com/library/psdk/dasdk/oled0cs7.htm)
23. OLE DB Products on the Market Today, Microsoft Corporation, 2000 (http://www.microsoft.com/data/partners/products.htm)
24. Microsoft ActiveX Data Objects, Microsoft Corporation, 1999 (http://www.microsoft.com/data/ado/)
25. Platform SDK – ActiveX Data Objects, Microsoft Corporation, 2000 (http://msdn.microsoft.com/library/psdk/dasdk/ados4piv.htm)
26. Microsoft ODBC, Microsoft Corporation, 1999 (http://www.microsoft.com/data/odbc/)
27. Platform SDK – ODBC, Microsoft Corporation, 2000 (http://msdn.microsoft.com/library/psdk/dasdk/odbc4vcn.htm)
28. Web Server Survey, E-Soft Incorporated, 2000 (http://www.e-softinc.com/survey/data/200003/ee/index.html)

Lisa 1 ColdFusion serveri kasutamine eri veebiserverite ja andmebaasidega

	Express Windows	Express Linux	Professional Windows	Professional Linux	Enterprise Windows	Enterprise Solaris	Enterprise HP-UX	Enterprise Linux
Veebiserverid								
Microsoft Internet Information Server (IIS) v4.0	✓		✓		✓			
Netscape Enterprise Server (NSAPI) v3.5.1 ja 3.6	✓		✓		✓	✓	✓	
Apache Web Server v1.3.6 ja 1.3.9	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Personal Web Server (PWS) v4.0	✓		✓					
O'Reilly WebSite (WSAPI)	✓		✓		✓			
Internet Server API (ISAPI)	✓		✓		✓			
Common Gateway Interface (CGI)	✓		✓		✓	✓		
Andmebaasid								
ODBC								
MERANT INFORMIX 7.x/9.x draiver				✓		✓	✓	✓
MERANT Sybase 11 draiver				✓		✓	✓	✓
MERANT dBase/FoxPro draiver						✓	✓	
MERANT IBM DB2/6000 draiver						✓	✓	
MERANT OpenIngres 1.x draiver						✓	✓	
MERANT OpenIngres 2.x draiver						✓	✓	
MERANT Oracle 7 draiver						✓	✓	
MERANT Oracle 8 draiver				†		✓	✓	†
MERANT Text draiver						✓	✓	
MERANT Microsoft SQL Server draiver				†		✓		†
MERANT MySQL draiver		†		†				†
Microsoft Access draiver	✓		✓		✓			
Microsoft SQL Server draiver			✓		✓			
Microsoft dBase draiver	✓		✓		✓			
Microsoft FoxPro draiver	✓		✓		✓			
Microsoft Visual FoxPro draiver	✓		✓		✓			
Microsoft Excel draiver	✓		✓		✓			
Microsoft Text draiver	✓		✓		✓			
FileMaker ODBC draiver	✓		✓		✓			
Andmebaasidega otse suhtlus								
Sybase System 11					✓	✓	✓	✓
Oracle 7.3, 8.0 ja 8i					✓	✓	✓	✓
Informix 7.3					✓	✓	✓	
DB2 Universal Database 5.2/6.1					✓	✓	✓	✓
OLE DB								
SQLOLEDB			✓		✓			
Microsoft.Jet.OLEDB			✓		✓			

Märkused

✓ Kasutatav † Alates 2000 aastast saadaval.

Lisa 2 Hetkel olemasolevad OLE DB tooted

Andmete tüüp/platvorm	OLE DB tarnija	Tootja
(Unisys® OS 2200 ja ClearPath IX Systems)	AIS UniAccess	Applied Information Sciences, Inc.
DB2/400 AS/400	Acceler8-DB and DataGate/400	ASNA
DB2/400 (Microsoft Windows NT Server)	HiT OLE DB Server/400 ja Developer Edition	HiT Software, Inc.
DB2/400 (Microsoft Windows NT Server)	HiT OLE DB/400 ja Developer Edition	HiT Software, Inc.
DB2/400 AS/400	Client Access for Windows 95/NT SDK	IBM
OLE DB tarnijad: Relatsioonilised andmeallikad - Oracle, Sybase, Informix , SQL Server, DB2, Rdb, Ingres, Red Brick, Non-Stop SQL/MP, Non-Stop SQL/MX Mitte-relatsioonilised andmeallikad - VSAM, CISAM, RMS, DBMS, MUMPS, Btrieve, Jasmine, Adabas, Enscribe, Flat file data sources, Text file data sources	ISG Navigator	ISG
Suvaline MAPI-põhine andmehoidla	Connect OLE DB	Merant

IBM DB2 perekond, Informix, Microsoft SQL Server, OpenIngres, Oracle, Sybase Adaptive Server 11.5 ja System 10/11; sisaldab toetust ka Oracle ja DB2-le IBM OS/390-l	SequeLink OLE DB Edition	Merant
AS/400 ja VSAM (Microsoft Windows NT 4.0 Workstation, Windows NT 4.0 Server või Windows 95)	MetaWise DP	MetaWise
LDAP version 2, NetWare 4, NetWare 3, ja Windows NT kataloogid	Microsoft Active Directory	Microsoft
HTML, tekst, Microsoft Office dokumendid, SQL (NT)	Microsoft Index Server	Microsoft
Microsoft Access tarnija	Microsoft Jet Database Engine	Microsoft
Microsoft SQL Server tarnija	Microsoft SQL Server	Micrsosoft
ODBC tarnija	Microsoft OLE DB SDK	Microsoft
VSAM ja AS/400	Microsoft SNA Server	Microsoft
ObjectStore andmebaasid	ObjectStore Active Toolkit 1.5	Object Design, Inc.
OLAP Microsoft SQL Server, Oracle, Red Brick, ja Sybase	Sagent Data Mart Server	Sagent Technology

xBASE (FoxPro, Clipper ja dBASE) (Windows 95, 3.1, NT, CE, DOS, OS/2, Macintosh ja UNIX platvormid nagu Solaris, SunOS, HP/UX, AIX, SCO, Linux)	CodeBase OLE DB Data Provider	Sequiter
OSS süsteemid	Data Explorer	X-Tension
dBASE, FoxPro, CA-Clipper (DBF, FPT, CDX, NTX, NSX) (Microsoft Windows NT/2000 Server)	Apollo OLE DB Provider	Vista Software

Andmete tüüp/platvorm	**OLE DB teenuse tarnija**	**Tootja**
Oracle, Sybase, Informix , SQL Server, DB2, Rdb, Ingres, Red Brick, Non-Stop SQL/MP, Non-Stop SQL/MX, CISAM, RMS, DBMS, MUMPS, Jasmine, Adabas, Enscribe, Flat files, Text files, ODBC data, OLEDB data (Intel Win 95, Intel NT, HP Unix, Sun Unix, Data General Unix, IBM RS/6000 Unix, Digital Alpha Unix, Digital Alpha OpenVMS, Digital Alpha NT, Tandem NSK)	ISG Navigator (Multi-Platform Distributed Query Processor)	ISG

SQL ja mitte-SQL andmed nagu MAPI e-mail, Microsoft Exchange, Lotus Notes ja Lotus cc:Mail	DataDirect Reflector (Query Processor)	Merant

Andmete tüüp/platvorm	OLE DB toode	Tootja
Instrumentaalpakett (*toolkit*) kiirete OLE DB tarnijate arenduseks	OpenAccess OLE DB SDK	Automation Technology, Inc. (ATI)
Dünaamiline OLE DB OLAP MDX generaatori jaoks (Windows 95, NT)	AntMDX	Geppetto's Workshop
Sisaldab OLE DB SDK madalataseme OLE DB tarnija tegemiseks ning ODBC SDK (Windows NT)	SimbaProvider™ Professional Edition	Simba Technologies
Instrumentaalpakett valmistamaks andmetarnijaid OLE DB for OLAP jaoks (Windows 95, NT)	SimbaProvider™ for OLAP	Simba Technologies
OLE DB for OLAP MDX analüsaator (Windows 95, NT)	MDX Parser for OLE DB for OLAP	X-Tension