

Modern webbutveckling med PHP

PHP 6 + lite HTML 5, CSS 3, Javaskript 2, och MySQL 6

Detta utkast är ofullständigt. Kapitel saknas och somligt måste rensas bort enligt principen ”kill your darlings”.

Feedback kan skickas till Lars Gunther på adressen gunther@keryx.se

Copyright: Lars Gunther

All rights reserved

1 Om denna bok

Syfte

Att lära ut tekniken för en webbplats på ett sätt som tar hänsyn till ”best practice”¹ och alltså inte lär ut tekniker och metoder som är farliga, föråldrade eller på annat sätt undermåliga.

Den här boken går inte igenom skriptspråket PHP på det sätt som är vanligt, dvs. med utgångspunkt i språkets delar, utan dess syfte är att lära ut hur man bygger en webbplats. Dess övergripande design och tekniker är utgångspunkten. PHP är ett medel för att åstadkomma detta. Av det skälet kommer andra tekniker att beröras också: HTML 4 och 5, samt XHTML; CSS, http, SSL, SQL, m.m.

Målsättningen är inte att förmedla heltäckande kunskap om någon teknik i sig, utan fokus ligger på samspelet dem emellan. Det är inte en referensbok eller en manual. Tips på fördjupning och förslag på goda referensböcker lämnas med jämna mellanrum.

Detta tänkande går igenom de flesta kapitel. När jag jämför likartade funktioner, så behandlas inte i första hand deras syntax, utan jag försöker ge en bild av **när** respektive funktion är lämplig att använda.

Nivå och särdrag

Boken förutsätter dock inte att du är väl bevandrad i någon teknik i sig, utan allt utgår från en ren nybörjarnivå. En förhoppning är att mer erfarna webbutvecklare ändå skall finna en hel del matnyttigt.

En unik sak med denna bok är att den försöker lära ut såväl front-end, som back-end design. Bristen på kommunikation mellan utvecklarna inom dessa två områden ser jag nämligen som kanske webbutvecklingens allra största problem just nu. Back-end utvecklarna skapar idag ofta lösningar som ofelbart ger en webbplats dålig användbarhet (”usability”) och dålig tillgänglighet (”accessibility”), samt förhindrar att man drar nytta av framsteg som gjorts på områden som webbstandarder, mikroformat och CSS.

Samtidigt lärs front-end design ut som om webbsidor fortfarande vore statiska. Den layout och den interaktion som man söker åstadkomma måste ju kunna integreras i något slags innehållshanteringssystem (CMS).

Den bok som kommer närmast denna till sin idé är *PHP Solutions* av David Powers från Friends of Ed. Den boken rekommenderas varmt. Den skiljer sig dock från denna i upplägget. Den boken lär ut PHP till front-end utvecklare som vill utöka sin repertoar. Min bok lär ut PHP till back-end utvecklare som vill vara medvetna om de krav som front-end aspekterna ställer på back-end. Min

¹ Så här definieras ”best practice” på Wikipedia: **Best Practice** is a management idea which asserts that there is a technique, method, process, activity, incentive or reward that is more effective at delivering a particular outcome than any other technique, method, process, etc. The idea is that with proper processes, checks, and testing, a project can be rolled out and completed with fewer problems and unforeseen complications.

bok är mer detaljerad och går djupare in på applikationsarkitektur.

@todo Kolla: Stylin' författarens bok

Kodnivåer

Jag har märkt att många nybörjare läser exempelkod, vars uppgift är att illustrera en viss funktion, och använder denna kod som vore den färdigutvecklad. I den här boken har jag därför grupperat all kod i ett antal kategorier och nivåer, för att undvika missförstånd.

Kodnivå V: Varning! Detta är ett exempel på dålig eller rent av farlig kod.

Kodnivå D: Demo. Denna kod är inte användbar, den illustrerar bara en viss funktionalitet.

Kodnivå A: Alpha. Detta är en påbörjad praktisk användbar kod, men mycket återstår innan den är klar.

Kodnivå B: Beta. I stort sett klar kod, men ännu inte putsad eller garanterad buggfri.

Kodnivå *: "Enstjärnig kod", användbar för hobbyprojekt och enklare professionella arbeten.

Kodnivå **: "Tvåstjärnig", lämplig för små och medelstora webbplatser.

Kodnivå ***: "Trestjärnig", "enterprise", kod för stora, högtrafikerade och affärskritiska webbplatser.

Av naturliga skäl kommer de flesta exempel i boken vara på de lägsta nivåerna.

Vad denna bok inte är

Denna bok har som mål att förmedla *principer* för utveckling. Den är inte en "kokbok", även om somliga kodexempel naturligtvis är återanvändningsbara. Just möjligheten att kunna använda samma kod på fler ställen är ju en sak som utmärker "best practice"! Dock skall man betänka att den kod som ges i exemplen oftast skall betraktas som en hållplats på vägen mot slutmålet. Den illustrerar en eller ett par principer och inte en färdig lösning.

Detta är inte en bok om webbdesign. För det första menar jag att det ordet ofta är missledande. Moderna webbplatser är komplexa företeelser, där designen spelar en viktig roll, men dock bara en roll bland många. Påfallande många webbplatser är framgångsrika trots att de ur estetisk synvinkel är rent utav usla – tänk på MySpace eller Lunarstorm om du vill ha ett exempel. Jag hävdar inte att estetik är meningslöst – bara att den traditionella utlärningen av webbdesign som om det vore en i princip helt estetisk sak att göra webbplatser är för snäv och därför missvisande.

Med tanke på vad boken avser att lära ut, så är det kanske märkligt att det inte sägs mer om utvecklingsprocessen och användbarhetstestning. Någonstans måste man göra begränsningar. Denna bok fokuserar främst på den del av utvecklingskedjan som handlar om konkret programmering. Även om denna programmering sker i medvetenhet om hur den påverkar en webbplats

användbarhet, så är detta alltså inte en bok om användbarhet i sig.

Pedagogik

Den här boken är tänkt att kunna användas som läromedel på gymnasienivå eller på högskolenivå. Förslag på övningar och vägledning till hur den kan implementeras i undervisningen finns på bokens webbplats: <http://keryx.se/modern-webbprogrammering/> @todo

Oavsett om du läser boken som en del av formella studier eller inte, så rekommenderas att du svarar på de eventuella kontrollfrågor och gör de övningar som finns i avslutningen av många kapitel.

Om författaren

Jag som skriver denna bok heter Lars Gunther och jag har sysslat med webbutveckling sedan 1996. År 2001 började jag jobba som gymnasielärare i datorteknik och har haft flera kurser om webbt teknik. Hösten 2005 blev jag ombud för ”The Web Standards Project Educational Task Force” för att vara deras talesman inför Skolverket. Jag är också medlem av ”The Guild Of Accessible Web Designers”. Man kan hitta mig via min egen sida: <http://keryx.se/> och på otaliga mejlinglistor och forum som berör webbutveckling under namnet ”itpastorn”.

Jag är också teolog, ordinerad pastor i Svenska Missionskyrkan och amatörhistoriker.

Tack till

@todo

Emil Hernvall

Hans Eriksson

Roger Johansson

Tommy Olsson

Etc.

Konventioner

Det som skrivs vid en prompt

Det som skrivs som en del av en manuell http- eller ftp-session

Svar från server eller program. (Kursiv tills vidare.)

Fördjupning

Wikipedia kan ibland saboteras. Alla artiklar jag hänvisar till har nått en viss mognad. Ju fler revisioner desto bättre. Några har jag själv putsat lite på! Svenska wikipedia är sämre. Klicka på språklänken om det finns någon svenksspråkig artikel.

Övningsmiljö

Många övningar är lättast gjorda i Linux... Knoppix ett tips...

Appendix om installation av Apache, PHP och MySQL

Utvecklingsverktyg beskrivs i kapitel XX. **Minimkrav:**

- Syntax highligthing
- Radnumrering
- Möjlighet att använda både latin-1 och utf-8
- Alla slags radslut: `\r\n` `\n` och `\r`

Vad för färgkodning har din editor? Skiljer den sig från denna boks?

Kapitel

1Om denna bok.....	2
2Trender på webben.....	17
3Olika slags kodning.....	22
4Webbens grundtekniker.....	24
5Grunderna i HTML.....	29
6HTML – från grötkod till standarder.....	31
7Grundläggande CSS.....	36
8XHTML.....	42
9Att skapa god (X)HTML.....	46
10Avancerad (X)HTML.....	48
11Användbarhet och tillgänglighet.....	50
12Front-end möter back-end.....	53
13Skript på klientsidan.....	55
14PHP – webbens ”klister”.....	59
15Basal PHP.....	61
16Utvecklingsmiljön.....	64
17Variabler i PHP.....	66
18Funktioner.....	70
19Objekt i PHP.....	72
20Fel, fel, fel!.....	73
21Styr upp din PHP-miljö.....	75
22Separera innehåll från struktur med PHP.....	78
23Olika slags PHP-skript.....	81
24Lokalisering.....	84
25Språkkonstruktioner.....	88
26Indatakontroll.....	90
27Tid.....	92
28Procedurella page-controlers.....	94
29Arrayer.....	96
30Enkla databasexempel för PHP 5.2+.....	98
31Grundläggande databasteori.....	111
32MySQL.....	113
33Skapa tabeller med PHPMyAdmin.....	115
34Kommunikation mellan PHP och MySQL.....	117
35Mer om SQL.....	119
36Avancerad SQL.....	121
37Databasfel och oväntade databassvar.....	123
38Formulär – klientsidan.....	125
39Formulär – serversidan.....	127
40Skicka mejl med PHP.....	128
41Sessioner och inloggning.....	129
42En kaka, tack!.....	130
43Objekt, fördjupning.....	131
44Filhantering.....	132
45XML i PHP.....	133

46	PHP DOM kontra JS DOM	134
47	Simple XML kontra E4X	135
48	Prestanda och skalbarhet	136
49	PHP som en del av webbplatsens utveckling	138
50	Ett fullfjädrat exempel	139
51	Aldrig fullärd	141
52	Appendix 1: Installation, LAMP	142
53	Appendix 2: Installation, WAMP, WIMP, WIAP	143
54	Appendix 3: Installation, MAMP	144
55	Foobar	145

Innehållsförteckning

1 Om denna bok.....	2
Syfte.....	2
Nivå och särdrag.....	2
Kodnivåer.....	3
Vad denna bok inte är.....	3
Pedagogik.....	4
Om författaren.....	4
Tack till.....	4
Konventioner.....	4
Fördjupning.....	4
Övningsmiljö.....	5
2 Trender på webben.....	18
Front-end tekniker.....	18
Framtidens front-end?.....	18
Design-trender.....	18
Framtidens design?.....	19
Användarinteraktion.....	19
Framtidens användarinteraktion.....	19
Back-end utveckling.....	19
Framtidens back-end?.....	20
Webbläsare, användaragenter, klientplattformar.....	20
Framtiden på detta område?.....	20
Säkerhet.....	21
PHP trender.....	21
Framtiden för PHP?.....	21
3 Olika slags kodning.....	23
Imperativ eller funktionell kodning.....	23
Lågnivå – ej använt på webben.....	23
Procedurell kodning.....	23
Objektbaserad.....	23
(Äkta) objektorienterad.....	23
Deklarativ kodning.....	23
4 Webbens grundtekniker.....	25
Från URL till värd – modell 1993.....	25
Kommunikation mellan webbläsare och server – modell 1993.....	26
Tolkning av innehåll – modell 1993.....	27
Sammanfattning.....	28
Övningar.....	28
Fördjupning.....	28
5 Grunderna i HTML.....	30
Element = starttagg + innehåll + sluttagg.....	30
Dokumentstruktur.....	30
De viktigaste elementen.....	30
Attribut.....	31
Specialattributet ”id”.....	31

De vanligaste attributen.....	31
Sammanfattning.....	31
Övningar.....	31
Fördjupning.....	31
6HTML – från grötkod till standarder.....	32
Pionjäråren (1991 – 1994).....	32
Browserkriget (1994 – 1999).....	32
HTML och XHTML standarder.....	32
HTML 2.0.....	32
HTML 3.0.....	33
HTML 3.2.....	33
HTML 4.0.....	33
HTML 4.01.....	33
XHTML 1.0.....	34
XHTML 1.1.....	34
XHTML 2.0.....	34
Xforms.....	34
HTML 5 och XHTML 5.....	34
Exempel på utvidgningar av (X)HTML.....	35
Web Forms.....	35
Canvas.....	35
Annat gruppen intresserat sig för.....	35
Exkurs: Relationen till SGML.....	35
Fördjupning.....	35
XML.....	36
Några andra XML-tekniker (ämnade för klientsidan).....	36
7Grundläggande CSS.....	37
CSS-regler.....	37
Typ-selektorn.....	38
Den universella selektorn.....	38
”Global reset”.....	38
star-html hack.....	38
ID-selektorn.....	38
Avkomlingsselektorn.....	38
Klass-selektorn.....	38
Pseudoklasser.....	39
:link :visited.....	39
:active :hover :focus.....	39
Attribut-selektorer.....	39
Att lägga in CSS.....	39
Inline.....	39
Sidhuvudet.....	40
Externt.....	40
Tips.....	40
Övningar.....	40
Fördjupning.....	41
8XHTML.....	43

Teoretiska fördelar.....	43
Praktiska fördelar.....	43
Krav att uppfylla.....	43
Myter om XHTML.....	44
Fördjupning:	45
MIME-typer för XHTML.....	45
PCDATA och CDATA.....	45
Avancerade problem.....	45
ID.....	45
Namnrymder och DOM-funktioner.....	45
Övningar.....	46
Fördjupning: Debatten om XHTML kontra HTML.....	46
9Att skapa god (X)HTML.....	47
Validering.....	47
HTML Tidy.....	47
Övningar.....	48
Med validatorn.....	48
Med Tidy.....	48
10Avancerad (X)HTML.....	49
Vad du skall undvika.....	49
Lär dig elementens semantiska funktion.....	49
Lär dig elementens gruppering.....	49
Lär dig det fulla registret av element.....	50
Lär dig elementens fulla logiska register.....	50
Lär dig ”web patterns”.....	50
Utökad semantik.....	50
Mikroformat.....	50
Övningar.....	50
11Användbarhet och tillgänglighet.....	51
Använd inte frames!.....	51
Öppna inte nya fönster.....	51
Ha en vettig URI för varje resurs.....	51
Visa inte onödig information.....	51
Din sida skall fungera med bara sin (X)HTML-kod.....	52
Det finns många slags funktionsnedsättningar.....	52
Övningar.....	52
Fördjupning.....	53
12Front-end möter back-end.....	54
Olika roller.....	54
Använd inte back-end tekniker när front-end dito är bättre.....	54
Använd inte front-end tekniker när back-end dito är bättre.....	55
Dubblerad logik.....	55
Övningar.....	55
Fördjupning.....	55
13Skript på klientsidan.....	56
Faser (enligt PPK).....	56
Användningsområde.....	56

Beståndsdelar.....	56
Händelsehantering.....	57
Vad en expert bör kunna.....	57
Javaskript och PHP.....	57
”Best practices”.....	58
Övning.....	58
Fördjupning.....	58
Böcker.....	58
Hemsidor och bloggar.....	58
Personer.....	58
Toolkits.....	58
14PHP – webbens ”klister”.....	60
PHP i jämförelse med.....	60
C/C++.....	60
Java/JSP.....	60
C#/.NET.....	60
Perl.....	60
Ruby.....	60
PHP gemenskapen.....	60
15Basal PHP.....	62
PHP från kommandoraden.....	62
PHP på webbservern.....	62
Hur PHP bäddas in.....	63
Kommandot echo.....	63
Kommentarer.....	64
Övningar.....	64
Fördjupning.....	64
16Utvecklingsmiljön.....	65
Servrar.....	65
Editor.....	65
Radnumrering.....	65
En fullfjädrad IDE.....	65
17Variabler i PHP.....	67
Vad är en variabel?.....	67
De enkla värdetyperna.....	67
Heltal.....	67
Flyttal.....	67
Strängar.....	67
Boolska.....	67
Null.....	67
Variabelnamn.....	67
Skiftlägeskänslighet.....	67
Legal tecken.....	67
Bra variabelnamn.....	68
”Loosely typed”.....	68
Konstanter.....	68
(Enkla) operationer.....	68

”By reference” och ”by value”.....	69
Referensräkning.....	69
Avancerade variabeltyper.....	69
(Några) pre-definierade variabler.....	69
Övningar.....	69
Fördjupning.....	69
Pre-definierade konstanter.....	70
18Funktioner.....	71
Skiftlägeskänslighet.....	71
Variabel ”scope”.....	71
Skicka parametrar by reference.....	71
Default value för en parameter.....	71
Övning.....	71
Fördjupning.....	71
Retur by reference.....	71
Statiska variabler.....	71
De är inte objekt.....	71
De kan anropas dynamiskt.....	72
De kan skapas dynamiskt.....	72
Funktionshanterande funktioner!.....	72
Funktioner följer extension.....	72
PECL.....	72
19Objekt i PHP.....	73
Vad är ett objekt?.....	73
Klassdefinitionen.....	73
Konstruktorn.....	73
Variabler.....	73
Metoder.....	73
Konstanter.....	73
Åtkomst: Private, protected, public.....	73
Statiska metoder.....	73
20Fel, fel, fel!.....	74
Kodningsfel.....	74
I förhållande till PHP:s regler.....	74
I förhållande till din applikations logik.....	74
User error.....	75
Exceptions.....	75
Övningar.....	75
21Styr upp din PHP-miljö.....	76
Globala värden.....	76
Lokala värden.....	76
Per skript.....	76
Min standard ”init.php”.....	76
Kontrollera inställningar.....	76
Inställningar att lägga märke till.....	77
Magic quotes.....	77
Aktivera extensions i Windows.....	77

Sökvägar.....	77
Safe mode.....	77
Open basedir.....	77
Allow url-fopen.....	77
Övning.....	77
Fördjupning.....	78
22Separera innehåll från struktur med PHP.....	79
Enkelt exempel – som en ”iframe”.....	79
Språkkonstruktioner.....	79
Mer verkligt exempel: En modulär sida.....	79
En page-controler.....	79
Master-template.....	79
HTTP-huvudet allra först!.....	79
HTML-huvud.....	80
Main.....	80
Spalt.....	80
Sidfot.....	80
Jämfört med frames.....	80
Katalogstruktur.....	80
Sökvägar i PHP.....	80
Övning.....	81
Fördjupning.....	81
23Olika slags PHP-skript.....	82
Kontroller.....	82
Page-controler.....	82
Front-controler.....	82
Mallar.....	82
Kompilerade mallar.....	82
Konfiguration.....	82
i18n-filer.....	82
Dokumentation.....	83
Cache.....	83
Funktioner.....	83
Klasser.....	83
PEAR.....	83
Framework.....	83
Övning.....	83
Fördjupning.....	84
Var i övrigt ”dräller” PHP med filer?.....	84
24Lokalisering.....	85
Vad menas med i18n och lokalisering?.....	85
Aspekter.....	85
Skrift.....	85
Språk.....	85
Tid.....	85
Symboler och färger.....	85
Inställningar i PHP.....	85

Kort om teckenkodning.....	86
Teckenkodning i PHP.....	86
PHP <= 5.3.....	86
PHP 6.....	86
Säkerhetsaspekter.....	86
Övningar.....	86
Fördjupning.....	87
Resurser.....	87
Språk man inte alltid tänker på:.....	87
Mer om tecken.....	87
UTF-8 som en textsträng i PHP.....	87
Teckenkodning och (X)HTML-entiteter.....	87
Hur webbläsaren väljer teckenkodning.....	88
Text/html.....	88
application/xhtml+xml.....	88
25Språkkonstruktioner.....	89
Gruppering med ”måsvingar”.....	89
if - elseif – else.....	89
Avancerade villkor.....	89
Ternär operator.....	89
While och do-while.....	89
For.....	89
Switch.....	89
Övning.....	89
Fördjupning.....	89
26Indatakontroll.....	91
Tvinga data att bli någorlunda vettig.....	91
Typecasting.....	91
Trim().....	91
Tänkt att vara HTML?.....	91
Regelrätt kontroll.....	91
Finns värdet alls?.....	91
Min- och maxlängd av textsträng.....	92
Kontroll av vilka slags tecken som ingår.....	92
Kontroll gentemot en whitelist.....	92
Kontroll av mönster.....	92
Övning.....	92
Fördjupning.....	92
Indatafilter i PHP 5.2.....	92
27Tid.....	93
Standardformat för tid.....	93
Unix-tid.....	93
PHP 5.2.2+.....	93
Använd din DBMS!.....	93
http-huvuden och cachning.....	93
Övningar.....	94
Fördjupning.....	94

Tidsarrayer.....	94
Klocka din applikation.....	94
28Procedurella page-controlers.....	95
Stegen för att visa en vanlig sida.....	95
Ett enkelt exempel.....	95
Stegen för att lagra data.....	95
Förhindra dubbellagring	96
29Arrayer.....	97
Jämfört med andra programmeringsspråk.....	97
Foreach.....	97
Sortering.....	97
Andra array-funktioner.....	97
Array-lik objekt.....	97
Övning.....	97
Fördjupning.....	98
Array-cursor.....	98
Sortering av multi-arrayer.....	98
30Enkla databasexempel för PHP 5.2+.....	99
Snabb säkerhetsinfo.....	100
Exempel 1: Anslutning till databas och hämta några rader data.....	100
Exempel 2: Samma som ovan, fast data visas i mallen.....	101
Exempel 3: Mysql-gränssnittet.....	104
Exempel 4: Anslutningsspecifika inställningar.....	106
Exempel 5: Förkortad version av föregående exempel.....	107
Exempel 6: SQL-fråga baserad på användardata och prepared statements.....	109
Appendix: SQL-satser för att skapa den databas som krävs för att kunna köra kodexemplen....	111
Övning.....	111
31Grundläggande databasteori.....	112
Tabeller.....	112
Datatyper.....	112
Primärnycklar.....	112
Andra index.....	112
Relationer.....	113
Vad är SQL?.....	113
Transaktioner.....	113
Övning.....	113
Fördjupning.....	113
32MySQL.....	114
Installation.....	114
Grundläggande säkerhet.....	114
PHP:s tre inbyggda moduler för MySQL.....	114
PHPMyAdmin.....	114
Inställningar i MySQL.....	115
Tabelltyper.....	115
Övningar.....	115
Fördjupning.....	115
33Skapa tabeller med PHPMyAdmin.....	116

Planera.....	116
Med PHPMysqlAdmin.....	116
Tabelltyp.....	116
auto_increment.....	116
Versaler/gemener.....	116
Aggressiv användning av index på webbapplikationer.....	116
Ändra i efterhand.....	116
Övningar.....	117
Fördjupning.....	117
Andra verktyg.....	117
34Kommunikation mellan PHP och MySQL.....	118
Schematisk förklaring.....	118
Det klassiska mysql-gränssnittet.....	118
Det kraftfulla mysql-gränssnittet.....	118
PDO.....	118
Viktiga metoder i PDO.....	118
Övning.....	118
Fördjupning.....	118
Databasabstraktion.....	118
Call-level abstraction.....	118
Data-type abstraction.....	119
SQL-abstraktion.....	119
Gränssnitt skrivna i PHP.....	119
PEAR DB.....	119
MDB2.....	119
AdoDB.....	119
Andra.....	119
Ditt eget?.....	119
35Mer om SQL.....	120
SELECT.....	120
WHERE.....	120
ORDER BY.....	120
LIMIT.....	120
Aggregatfunktioner och group by.....	120
Datumkonvertering i MySQL.....	120
INSERT.....	120
UPDATE.....	120
DELETE.....	120
Övning.....	121
Fördjupning.....	121
Hur en fråga utförs i MySQL.....	121
Having.....	121
Limit jämfört med andra DBMS!.....	121
Insert.. ”on duplicate row”.....	121
Replace into.....	121
36Avancerad SQL.....	122
GROUP_CONCAT.....	122

JOIN.....	122
Natural join.....	122
Inner Join.....	122
Left (right) join.....	123
Utan villkor.....	123
JOIN och patterns.....	123
Transaktioner.....	123
Övningar.....	123
Fördjupning.....	123
Join utan "join".....	123
"Sounds like".....	123
37Databasfel och oväntade databassvar.....	124
Applikationsfel.....	124
Tomma resultat.....	124
Dubbla POST från användaren.....	124
Transaktioner.....	125
Fördjupning.....	125
Läsning av tabeller.....	125
38Formulär – klientsidan.....	126
Fallgropar.....	126
Web Forms 2.0.....	126
39Formulär – serversidan.....	128
Samma konfigurationsfil till PHP och JS.....	128
40Skicka mejl med PHP.....	129
Grundfunktionen.....	129
Fara: Header injection.....	129
Färdiga klasser.....	129
41Sessioner och inloggning.....	130
Ett enkelt exempel.....	130
Farorna.....	130
Inloggning.....	130
Behörighetskontroll.....	130
42En kaka, tack!.....	131
43Objekt, fördjupning.....	132
44Filhantering.....	133
45XML i PHP.....	134
46PHP DOM kontra JS DOM.....	135
47Simple XML kontra E4X.....	136
48Prestanda och skalbarhet.....	137
49PHP som en del av webbplatsens utveckling.....	139
50Ett fullfjädrat exempel.....	140
51Aldrig fullärd.....	142
Mer om patterns.....	142
Unit testing.....	142
Versionshantering.....	142
Paketering.....	142
Bra resurser (i allmänhet).....	142

Böcker.....	142
Webbplatser.....	142
Podcasts.....	142
IRC.....	142
52Appendix 1: Installation, LAMP.....	143
Bokens krav:.....	143
LAMP.....	143
53Appendix 2: Installation, WAMP, WIMP, WIAP.....	144
54Appendix 3: Installation, MAMP.....	145
55Foobar.....	146

2 Trender på webben

Innan vi börjar titta specifikt på principerna för en modern webbplats, så kan det vara på sin plats att titta på historien som fört oss hit. Vad är det för utveckling som ställer krav på oss som webbutvecklare och vilka nya möjligheter har vi fått i och med den senaste tekniken? Dessa frågor kan naturligtvis endast behandlas summariskt här och det som nedan följer är kanske mer en samling stolpar, än en regelrätt text. Alla årtal som anges skall ses som ungefärliga och det finns naturligtvis en överlappning mellan olika ”epoker”.

Front-end tekniker

1. **Enkla textbaserade sidor**, med ganska semantisk HTML (1990-1993).
2. **Design-centrerad HTML**, med nästlade tabeller och ramar för layout, ”spacer gifs”, proprietär kodning (”Best viewed with”) och ofta meningslösa och irriterande DHTML-effekter (1994 – 2004).²
3. **Webbstandarder**, med semantisk (X)HTML, CSS-baserad layout, icke-inkräktande (”unobtrusive”) DOM-skript, mikroformat, etc.
 - ”Den elaka kusinen”: AJAX, när tekniken nyttjas utan tanke på användbarhet och tillgänglighet, ofta som ett självändamål.

Framtidens front-end?

1. **Deklarativ kodning**, med SVG, SMIL, etc.
2. **Utökad semantik** i (X)HTML, i form av (X)HTML 5 från WHAT-WG och förbättrad möjlighet till design genom **CSS 3**.³
3. **Professionella, ”enterprise level”, DOM-skript**, genom genomslag för avancerade kodteknik och JavaScript 2 samt DOM 3, samt formalisering av dagens viktigaste de-facto tekniker, såsom Window-objektet och AJAX.

Design-trender

Det är svårt att säga att någon nedanstående trend egentligen upphört, därför finns inga avslutande årtal.

1. **Broschyrliknande sidor**, med fixerad storlek på text och ytor, låg nivå av interaktion, i
-
- 2 Denna fas borde ha avslutats helt och hållet nu, men mindre duktiga utvecklare fortsätter att göra den här sortens sidor, och – än värre – mindre duktiga författare fortsätter att lära ut eländet. Många läser dessutom gamla artiklar som inte tagits bort, trots att deras ”bäst före datum” gått ut sedan länge.
 - 3 Jag tror alltså inte att XHTML 2 kommer att spela någon väsentlig roll. Utvecklarna av webbläsare säger att den standarden är näst intill omöjlig att få att fungera på ett vettigt sätt och den bryter totalt med dagens HTML, vilket gör övergången brutalt svår för utvecklare.

princip bara länkar och enkla formulär. Designidealen hämtas från tryckt media. (1993 -)

2. **"Flashiga" sidor**, gjorda med DHTML eller just Flash. Sidor med något mer interaktion, men fortfarande vanligast med fast storlek på innehållet. Designidealet hämtas från TV. (1996 -)
3. **Webben som sitt eget medium**, med fokus på användbarhet och hög grad av interaktion, ända upp till regelrätta applikationer. Webben som programkörningsmiljö (RTE). (2000 -)

Framtidens design?

Mest svårt att förutsäga, men troligen kommer begreppet "sida" bli mer och mer svårdefinierat. Ju mer man interagerar med sidan, så att den befinner sig i olika tillstånd, desto mindre relevant blir begreppet. För saker som tillbakaknappen och permanenta länkar innebär detta stora utmaningar.

Användarinteraktion

Jag återanvänder här tre begrepp från datakommunikationen, dock utan att hålla fast vid deras exakta definitioner, utan mer som en jämförelse.

1. **"Simplex"**: Innehåll visas för användaren som endast kan läsa det och sedan navigera vidare. (1990 – 1995)
2. **"Halv duplex"**: Innehållet presenteras och användaren kan reagera genom kontaktformulär, gästböcker, kommentarer och forum. (1995 -)
3. **"Full duplex"**: Användarna skapar webbplatsens innehåll, kanske genom att vara regelrätta författare, som i en Wiki eller på en Community, kanske genom sitt beteende ("kunder som köpte den här boken köpte också"). Även när teknikerna är desamma som i steg två, så är deras betydelse viktigare. Webbplatsens syfte sägs mer vara att möjliggöra användaraktivitet än att förmedla officiell information. (2004 -)

Framtidens användarinteraktion

En gissning är att gränserna mellan olika slags information suddas ut mer och mer. Man kommer inte längre ha dels en traditionell webbplats, dels en blogg och dels en wiki, utan webbplatsen rymmer alla delar. Information från wikis återanvänds som ett företags officiella support, inlägg i en diskussion blir marknadsföring, etc. Upphovsrätt och liknande kommer stå i stort fokus i den processen.

Back-end utveckling

1. **Statiska sidor**, där en HTTP-förfrågan direkt motsvarar en fil på servern. (1990 –)
2. **Databasdrivna sidor**, också kallade **dynamiska** sidor. (1995 -)

3. **Djuplänkade sidor**, där data från en plats syndikeras, aggregeras, bearbetas och återpubliceras på en helt annan plats. Webbtjänster, AJAX och ”mashups” är olika exempel på detta. (2003 -)

Framtidens back-end?

Tekniker som tidigare har legat helt på back-end sidan kommer alltmer att göras som en mix av front-end och back-end – och kanske vice versa. Förhoppningsvis leder detta till en korsbefruktning av expertis mellan dessa två tidigare alltför separerade läger.

Traditionella CMS:er kommer att tappa mark till bloggsystem och wikis om de inte anpassar sig snabbt. De senare å andra sidan kommer växa sig kraftfullare och få fler funktioner som tidigare endast fanns hos de traditionella ”stora” CMS:erna.

Webbläsare, användaragenter, klientplattformar

1. Rena **textbaserade** webbläsare (”WorldWideWeb”⁴, Lynx) och grafiska webbläsare med relativ enkel funktionalitet (till och med Netscape 2). Skärmupplösningar på 640x480 till 1024x768. (1990 – 1997)
2. **Kraftfulla**, skriptningsbara, webbläsare, på PC, Mac eller *nix arbetsstationer. Skärmupplösningar upp till 1920x1200 och än mer, ofta med widescreen. (1996 -)
3. **Multipla plattformar, multipla slags användaragenter.** (2004 -)
 - Traditionella datorer, fast med riktigt stora skärmar, typ 24 tum.
 - **Mobiltelefoner** och andra **bärbara** enheter med små skärmar eller ingen skärm alls i traditionell mening!
 - **Alternativa pekdon**, ej längre bara möss och musersättande pekdon.
 - **Talsyntes**, inte bara för synskadade och dyslektiker. Kanske vill man kunna lyssna på en webbplats medan man sitter i bilen eller joggar.
 - **Utskrifter.**

Framtiden på detta område?

Denna utveckling möjliggörs av genombrottet för webbstandarder och ny teknik, men den innebär också att den som inte anammar dessa inte längre kan utveckla för webben! Utvecklare fast i tänkandet ”en version per plattform, en version per typ av användare” kommer inte längre kunna konkurrera. Förhoppningsvis visar sig affärsnyttan med webbstandarder såpass snabbt att svenska företag äntligen begriper att de måste lägga om sitt tänkande.

4 ”WorldWideWeb”, utan mellanrum mellan orden, var namnet på Tim Berners-Lees ursprungliga webbläsare, världens allra första!

Det bör dock sägas att för närvarande (2007) så finns det en mängd olika webbläsare för mobiltelefoner, vilka nästan alla är mer eller mindre undermåliga och inte följer de standarder som finns särskilt väl. Därtill kommer att mobiloperatörerna ofta filtrerar och modifierar den data som skickas, så att det är näst intill omöjligt för en webbutvecklare att kontrollera exakt vad som når klienten. Så länge som detta är förutsättningarna som råder, så bör man troligen antingen endast skicka ren semantisk HTML till mobiltelefoner och ingen CSS. Vill man anpassa sin webbplats till olika telefonmodeller så är XSLT en rimlig teknik att använda.

Läs mer om detta i boken ”Mobile Web Design” av Cameron Moll, eller på adresserna <http://www.cameronmoll.com/archives/000398.html>, <http://24ways.org/2006/the-mobile-web-simplified> eller <http://www.developershome.com/wap/xhtmlmp/>

Säkerhet

Det är svårt att tala om epoker under den här rubriken. När webben skapades så var den från början tänkt för största möjliga öppenhet. Liksom för många andra tekniker på internet har säkerhet blivit något som man tagit hänsyn till i efterhand. Detta är en av de saker som måste ändras. Redan från början måste man lära sig säkra lösningar och skapa webbplatser som är användarens förtroende värda.

Webben kommer i framtiden vara den plattform som härbärgerar inte bara information, utan också socialt umgänge och rena nytto-applikationer. Troligen kommer alla som söker attackera din dator därför fokusera på webben som plattform för dessa attacker. Det enda man med säkerhet kan säga om säkerhet är att den bara kan bli viktigare i betydelse för varje år som går. Kunder kommer kräva säkra webbplatser.

Allteftersom företagen låter webben bli deras primära plattform för intern och extern kommunikation i realtid, så finns heller inte möjligheter att skapa ”säkerhet genom långsamhet”, dvs. att genomföra omfattande manuella kontroller av in- och utdata. Säkerhet måste alltså planeras in i varje system från dess första sekund av utvecklingsplaneringen.

PHP trender

1. PHP som en ”**kod ö**” i annars statiska sidor. Så konstruerades PHP från början och så används PHP än idag för somliga små dynamiska tillägg på en sida.
2. Till största delen procedurella skript som möjliggör **separation av ”intressen”** (eng: concerns) mellan design, teknik och innehåll.
3. Ofta objektorienterade ”enterprise level” applikationer, med **separation av logik** i applikationen, databasabstraktion, kodstandarder, modultest (eng: ”unit testing”), designmönster (eng: ”patterns”), återanvändningsbarhet och genomarbetat säkerhetstänkande. PHP 5 och 6 excellerar på detta område.

Framtiden för PHP?

- PHP kommer kanske tappa lite av sitt traditionella grepp om den **nedre och enklare delen** av server-side utvecklingen, där betoningen ligger på ”snabbt fixat” och ”lätt att implementera”, till Ruby eller helt färdiga applikationer.
- PHP kommer drastiskt att öka sin marknadsandel på ”**enterprise**” nivå, där tidigare JSP och i viss mån ASP tidigare dominerat.
- PHP kommer få en ökad användning på områden som tidigare varit **ovanliga** att använda just PHP på: CLI, GUI eller som interaktivt skal.
- PHP kommer mer och mer att användas för att leverera alternativa typer av information, såsom binära dataströmmar (bilder, kanske i framtiden också ljud och video), PDF, etc.

3 Olika slags kodning

@todo: hela kapitlet

@status: Stödord

Imperativ kontra funktionell kodning

Imperativ: Hur skall jag göra? Funktionell: Hur skall resultatet se ut?

PHP är imperativ, XSL är funktionell.

Lågnivå kontra högnivå

Binärkod

Assembler

Procedurell kontra objektorienterad

Procedurell kodning

Ibland kallad strukturell

Procedurella moment ingår alltid i objektorienterad kodning!

Main-objektet i Java eller C++ är i princip ett procedurellt skript.

Objektbaserad

Denna mellannivå kallas ofta för OO

Dagens Javaskript kan bara med stor möda komma förbi detta slags kodning.

Ofta en god ambitionsnivå för små och medelstora PHP-projekt. Procedurella page-controllers hanterar objekt som representerar data eller resurser.

(Äkta) objektorienterad

Av många det enda som anses duga på ”enterprise” nivå!

Men vänta, det finns ett steg till...

Deklarativ kodning

Oftast allra effektivast om man jämför antal kodrader med deras effekt.

SVG

SMIL

XUL/XAML

Och – ja! – HTML, XHTML och CSS!

4 Webbens grundtekniker

En webbläsare går igenom tre steg när den skall visa oss en webbsida:

1. Servern, på vilken sidan ligger, skall hittas.
2. Sidan skall efterfrågas.
3. Sidan skall ”målas upp” för användaren.

Mycket förenklat kan detta sägas svar mot tre tekniker: DNS, http och HTML. DNS faller utanför den här bokens område, men en viss kunskap i ämnet måste alla webbutvecklare ha. Om vi utgår från webben som den såg ut genombrottsåret 1993 så kan vi rent praktiskt konkretisera de här tre stegen som följer.

Från URL till värd – modell 1993

När en sida skall visas, så börjar webbläsaren med att utgå från dess URL. URL är en förkortning av ”Uniform Resource Locator”. Begreppet URL är egentligen en delmängd av det mer exakt definierade URI (”Uniform Resource Identifier”), men i dagligt tal används begreppen slarvigt i princip som synonymer. Begreppen formaliserades först 1994. Det kan också vara värt att nämnas att till en början stod bokstaven U inte för ”Uniform” utan för ”Universal” och i äldre litteratur är det så som förkortningarna uttyds.

Det som händer är i vart fall att en webbläsare fått en URL att hantera. Det kan ha skett genom att användaren manuellt skrev in den i adressfältet, genom att man klickat på en länk eller genom att man använt ett bokmärke.⁵ En URL ser i enklaste form ut ungefär så här:

```
<code>
```

```
http://keryx.se
```

```
</code>
```

Den har i denna form två delar: protokoll och värddamn. Dessa två delar är det minsta som måste anges. Om man inte själv anger protokollet, så förutsätter dagens webbläsare att det skall vara http.

Värddamnet skall mappas till en IP-adress. Det går rent tekniskt att ange IP-adressen direkt, men det har två nackdelar. Dels är det mycket krångligare att komma ihåg, speciellt som en webbplats ibland kan byta IP-adress men behålla sitt namn. Dels är det hart när omöjligt att låta fler än en webbplats ligga på samma server, som de ofta gör idag, om man inte anger värddamnet i i URL:en.

Lägg märke till att ett värddamn inte behöver innehålla ”www”. Än idag är det många som inte tror att det är en webbadress, dessa bokstäver förutan. Att ha dem i namnet är endast en konvention, och

⁵ Det kan också ske genom att webbläsaren *omdirigeras* av servern till en alternativ adress, eller genom att ett skript i klienten begär det.

ofta en helt onödig sådan. Man bör dock se till att båda varianterna fungerar och går till samma server. Följande två adresser ger alltså samma resultat:

```
<code>
```

```
http://keryx.se
```

```
http://www.keryx.se
```

```
</code>
```

Det finns vissa tekniska problem som bland annat berör säkerhetsmodellen för javaskript, med att ha adresser både med och utan "www". Smidigast löser man detta genom att tyst **omdiregera** all trafik som har "www" till adressen utan dessa tre i grunden onödiga bokstäver. Läs mer på <http://no-www.org/>

Kommunikation mellan webbläsare och server – modell 1993

När webbläsaren vet namnet på den värd som skall kontaktas, så ber den operativsystemets IP-funktioner om hjälp att översätta namnet till en IP-adress. Informationen i fråga eftersöks först i den textfil som på Linuxdatorer finns på platsen /etc/hosts och på windowsdatorer idag oftast på platsen C:\windows\system32\drivers\etc\hosts. Om informationen inte återfinns i denna fil, vilket den sällan gör, så frågas en *namnserver* i stället.

Förutsatt att en IP-adress gick att uppåbåda och att protokollet är just http, så kopplas nu webbläsaren upp mot servern. Kommunikationen ser, något förenklad, ut ungefär så här:

```
<code>
```

```
GET / HTTP/1.0
```

```
<samp>
```

```
HTTP/1.0 200 OK
```

```
Content-Type: text/html; charset=iso-8859-1
```

```
<html>
```

```
  <head>
```

```
    <title>Demosida</title>
```

```
  </head>
```

```
  <body>
```

```
    Något slags innehåll här...
```

```
  </body>
```

```
</html>
```

</samp>
</code>

Först säger webbläsaren tre saker:

1. "GET" är själva kommandot. Det betyder i princip "ge mig". Vanliga alternativ är POST och HEAD.
2. "/" är namnet på den *resurs* som efterfrågas. Just den enkla slashen står för hemsidan, dvs. den första sidan på webbplatsen, som inte behöver namnges i frågan. Oftast heter den i själva verket "index.html", "index.htm", "index.php", "default.htm" eller något liknande. **Tips!** Tag för vana att aldrig avslöja vad hemsidan egentligen heter, så kan du i framtiden byta från exempelvis en statisk sida (.html) till en dynamisk (.php) utan att det påverkar alla som har dess adress som bokmärke eller liknande.
3. "http/1.0" är en angivelse av vilken version av http som används. Om den utelämnas förutsätts version 0.9. Idag är version 1.1 vanligast och den har stötts av samtliga på marknaden förekommande servrar och webbläsare i många år nu.

Webbläsaren markerar att den "talat färdigt" med två radbrytningar.

Servern svarar med att ange en statuskod. 200 betecknar att resursen fanns och kunde skickas utan problem. Servern berättar sedan i vårt fingerade exempel vad för slags resurs som returneras, med en s.k. MIME-deklaration. "text/html" skall förstås som att innehållet är i textformat, till skillnad från en ljudsnutt, en bild, video eller något annat binärt format. Det skall tolkas som just html. Dessutom anges vilken teckenkodning som använts. I det här fallet "ISO-8859-1", vilket är en kodning som tillåter bokstäver som å, ä och ö.

Denna varudeklaration är mycket viktig ur webbläsarens synvinkel. Utan den skulle den kanske tolka bild som text, eller ljud som bild. Visserligen har webbläsare metoder att gissa innehållets art, om MIME-deklarationen uteblir, men det är omöjligt för den att alltid gissa rätt.

Det som föregår det egentliga innehållet kallas *http-huvud*, ofta kallade "headers" rätt och slätt. HTML-koden kan också ha en huvudsektion, men det är likväl en del av dokumentet som sådant.

När vi börjar koda PHP skall vi också titta på hur man som programmerare kan styra vad webbservern säger om det skickade innehållet.

Tolkning av innehåll - modell 1993

HTTP-huvudet skiljs från det egentliga innehållet med två nyradstecken. År 1993 förutsattes i princip innehållet antingen vara enkel text eller html. Ett äldre system, gopher, användes i viss mån fortfarande och i och med att webbläsaren Mosaic lanserats kunde också bilder visas.

Just Mosaic kunde också visa olika typsnitt och olika storlekar på texten. Det var helt enkelt den första *grafiska* webbläsaren. Html-koden började i och med detta bli mer och mer grafiskt orienterad. Märkord för fet stil, kursiv text, textstorlek och olika typsnitt introducerades raskt och

den tidigare uppmärkningen – vars enda syfte hade varit att ange ordens *logiska* funktion – började försvinna ur webbutvecklarnas tänkande. Detta framsteg skulle komma att visa sig vara en återvändsgränd.

I nästa kapitel skall principerna för HTML behandlas, liksom utvecklingen av detta märkspråk.

Sammanfattning

Webbens grundtekniker är:

- URI:er, i folkmun kallade URL:er.
- DNS
- HTTP
- HTML

Övningar

1. Använd verktygen *dig* eller *nslookup* eller webbplatsen <http://samspace.org> för att ta reda på IP-adressen till några kända webbplatser.
2. Tag reda på vilka protokoll, utöver http, som stöds av din webbläsare.
3. Pröva att surfa med den icke-grafiska webbläsaren Lynx (alternativt elinks). Testa gärna att göra en sökning på en sökmotor så att du får fylla i ett enkelt formulär.
4. Använd verktygen *netcat* (kommandot skrivs ”nc”) eller, om det inte finns tillgängligt, *telnet* för att simulera några http-förfrågningar mot en webbserver. Testa både metoderna GET och HEAD.

Ex:

```
<code>
$ <kbd>nc -v msn.com 80</kbd>
<samp>Connection to msn.com 80 port [tcp/http]
succeeded!</samp>
<kbd>GET / HTTP/1.0 [+dubbla enterslag]</kbd>
</code>
```

```
<code>
$ <kbd>nc -v ibm.com 80</kbd>
<samp>Connection to ibm.com 80 port [tcp/http]
succeeded!</samp>
<kbd>HEAD /no_such_page_exists.html HTTP/1.0 [+dubbla
enterslag]</kbd>
```

</code>

Fördjupning

Wikipedia har artiklar om alla ovan nämnda tekniker, webbläsaren Mosaic, etc.

Webbens pionjärtid finns beskrivna på bl.a. följande platser:

- @todo

Några webbplatser från pionjäråren finns bevarade på ”The Wayback Machine”:
<http://web.archive.org/collections/pioneers.html> (Tyvärr saknas de allra äldsta versionerna.)

Se också vidare i denna boks appendix för ytterligare information om URI:er, DNS och http.

”Architecture of the World Wide Web”: <http://www.w3.org/TR/webarch/>

I kortform: <http://www.w3.org/TR/webarch/summary.html>

5 Grunderna i HTML

Ett märkspråk, inte ett gängse programmeringsspråk.

Deklarativ programmering

Aktiviteten inte kallad ”programmering” eftersom resultatet inte var ett program. Annorlunda idag...

Att koda HTML anses ofta vara för enkelt eller sakna egentlig potential för att kunna göra något häftigt och därför sällan värt att fördjupa sina kunskaper om. Det är en tråkig attityd som innebär att många webbplatser blir mycket sämre än vad de annars kunde ha varit. Även om vi nu börjar enkelt och inte heller kan gå på djupet i en bok som denna, vars syfte främst inte är att lära ut HTML, så kom ihåg att det finns mycket att bemästra innan man – om någonsin – blir fullärd.

Element = starttagg + innehåll + sluttagg

När nybörjare talar om HTML så kallas ofta allt för ”taggar”. Tag följande exempel:

```

```

En del säger att detta är ”en bildtagg” och att den har en ”alt-tag”. Rätt språkbruk är att detta är ett bildelement, som i sin tur har ett alt-**attribut**. Elementet i fråga är skrivet i XHTML-syntax och inkluderar sin egen slut-tag. Den avslutande slashen är XML:s sätt att korta ned följande:

```
</img>
```

Eftersom just bildelementet alltid är tomt – då det saknar innehåll mellan taggarna – så är ovanstående onödigt långt. I traditionell HTML behöver man inte ens avsluta med en sluttagg. Elementet är ”självavslutande” (self-closing). Detta är alltså exakt samma:

```

```

Normalt sätt består dock ett element av en starttagg, som kan ha olika attribut, innehåll, vilket kan bestå också av andra element, samt slutligen en sluttagg. Det kan exempelvis se ut så här:

```
<a class="foobar" href="somepage.html">En sida till</a>
```

Elementet består av allt detta, enligt formeln: element = starttagg + innehåll + sluttagg. Roger Johansson reder förtjänstfullt ut dessa begrepp på adressen:

http://www.456bereastreet.com/archive/200508/html_tags_vs_elements_vs_attributes/

Dokumentstruktur

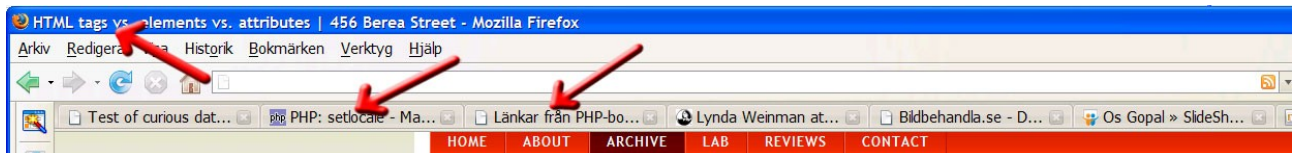
HTML förekommer i två **serialiseringar**: ”Vanlig” HTML och XHTML. Därutöver finns det ett antal versioner av båda, men i grunden ser ett dokument ut enligt följande:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Kapitel 5, exempel 1</title>
  </head>
  <body>
    <h1>Rubrik</h1>
    <p>Stycke</p>
  </body>
</html>
```

Först skall det vara en s.k. **doctype**. Den har ingått i specifikationen ända sedan version 2.0 av HTML och den fyller två syften:

1. När ett dokument kontrolleras, så anger den vilken version av HTML eller XHTML som man avsett att följa. Ovan anges den doctype som används av HTML 5.
2. När webbläsare ritlar upp en sida på skärmen så kan olika tolkningar ske av främst CSS-koden. Mozilla Firefox kallar dessa två lägen för ”quirks mode” respektive ”standards compliant mode”. Utan doctype hamnar man i quirks-läge – och det är avsevärt sämre!

Det element som utgör hela dokumentet i övrigt – html-elementet – kallas också **dokumentrot** eller, för XHTML ”document element”. Detta element innehåller två andra dokument: ”head” och ”body”. Head är dokumentets meta- och styrdata. Det skall inte innehålla information som skall visas för användaren, sånär som att ”title” hamnar på programfönstrets namnlist eller på flikarna i moderna webbläsare, enligt markeringar på bilden.



Body-elementet innehåller den information som skall visas i **dokumentfönstret** (”viewport”). I formell teknisk mening är detta en sanning med modifikation. Dels är html-elementet detsamma som viewport i XHTML, dels varken body-elementet eller html-elementet obligatoriska i HTML, utan webbläsaren skall själv infoga det om det fattas. Följande är alltså ett komplett HTML-dokument:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<title>Exempeldokument 2</title>
<h1>Rubrik</h1>
<p>Stycke</p>
```

Dokumentet ifråga har alltså både ett html- och ett body-element, men de är underförstådda. Försöker man hitta dem via CSS eller javascript, så finns de där – förutsatt att sidan är i HTML-läge! (Mer därom längre fram @todo: specifik angivelse.)

De viktigaste elementen

Här följer en introduktion till de vanligaste elementen. En (förhoppningsvis) komplett förteckning finns på <http://keryx.se/resources/html-elements.xhtml>.

Strukturerande element

@todo ”Definitionslista” i Open Office? Hur?

<html> Taggarna omsluter hela dokumentet.

<head> Taggarna omsluter styrinformation och meta-information.

<title> Dokumentets namn. Detta visas på namnlistan, det syns i sökmotorresultat och det läses upp för användare med talsyntes. Alla dokument bör ha ett unikt title-värde som ger en klar bild av sidans innehåll. Sånär som på information om teckenkodning bör detta element placeras först i head-elementet.

<link> Hänvisar till en relaterad resurs som CSS-filer, alternativa format som RSS, etc. Placeras i head-elementet.

<style> CSS-kod. Placeras i head-elementet.

<body> Taggarna omsluter allt som skall visas på dokumentytan, ”viewport”.

Grundläggande språkliga funktioner

<h1> till <h6>. Rubriker på nivå 1 (huvudrubrik) till nivå 6. Sällan behövs fler än tre nivåer. Använd rätt nivå utifrån innehållets logiska funktion och styr typsnittets storlek med CSS!

<p> Stycke, ”paragraph”.

 Betonad text. Visas vanligen kursiverad. Talsyntes höjer volymen. Vill du ha kursiv text av något annat skäl än att orden är betonade, så hitta ett element som täcker din semantiska mening om möjligt eller använd CSS.

 Starkt betonad text. Visas vanligen som fetstilt text. Talsyntes höjer rösten rejält. Skall inte heller missbrukas till annat än just betoning.

<abbr> En förkortning. Med attributet title kan du – om det inte är självklart vad förkortningen betyder – skriva ut den i sin helhet.

Hypertext

<a> Goda länktexter ingår som en naturlig del i språket. Med andra ord, det här är ett dåligt exempel:

```
För att läsa mer om våra produkter <a href="foo.html">klicka här</a>.
```

Just ”klicka här” är ett riktigt nybörjaraktigt sätt att formulera sin länktext. Det här är ett bättre exempel:

```
<a href="foo.html">Läs mer om våra produkter</a>.
```

Sidans struktur

```
<div>
```

```
<span>
```

XHTML 2 och (X)HTML 5 utökning:

```
<section>
```

HTML 5 utökningar:

```
<article>
```

```
<aside>
```

```
<header>
```

```
<footer>
```

Bilder och media

`` Bild, ”image”. Ett **ersatt** element, dvs. webbläsaren visar den bild som hänvisas till med attributet *src*. Attributet *alt* anger information som anger bildens funktion för de som inte kan se den, exempelvis användare av Lynx eller talsyntes. Läs om detta på <http://www.sitepoint.com/article/html-37-steps-perfect-markup/2> punkt 33.

```
<object> + <param>
```

Återkommer i HTML 5:

```
<embed>
```

Införs i HTML 5:

```
<video>
```

```
<audio>
```

<canvas>

Listor

ul, ol, li

dl, dt, dd

HTML 5:

dialogue

Element vars syfte endast är grafiskt har ingen plats i modern webbutveckling, utan ersätts med CSS. Vi återkommer längre fram till formulär.

Attribut

Hur de skrivs

Logisk funktion: lang, title, href, type, maxlength

Fysisk funktion: width, height (reserverar plats och därför ok!), bgcolor...

Användardefinierad funktion: class

Specialattributet "id"

Måste vara unikt

nybörjare använder det som "class" för CSS

De vanligaste attributen

Standard

i18n

Händelsehanterare (kort presentation + att de inte skall användas heller i modern kodning!)

Sammanfattning

Xx

Övningar

1. Skapa ett enkelt HTML-dokument, där du använder alla element som gåtts igenom i detta kapitel. Titta på det i en grafisk webbläsare och i en textbaserad⁶.
2. Skapa ytterligare ett dokument. Länka mellan de båda så att du kan surfa fram och tillbaka.

Fördjupning

<http://www.w3.org/MarkUp/>

http://en.wikipedia.org/wiki/Separation_of_concerns

”Architecture of the World Wide Web 4.3. Separation of Content, Presentation, and Interaction”

<http://www.w3.org/TR/webarch/Overview.html#pci>

Se följande kapitel!

⁶ Använd Lynx, Elinks, eller det textbaserade visningsläget i Opera.

6 HTML – från grötkod till standarder

@status: Stödord

Olika slags HTML. Tre faser: Pionjäråren, browserkriget, standarder.

Tanken: "HTML is designed to be interoperable across the widest possible range of platforms and devices of varying capabilities." (rfc 2854, sektion 2)

Praktiken: Olika företag har försökt låsa in användarna på just deras specifika plattform. En utlovad förbättring kom i stället att bli en hämsko.

Pionjäråren (1991 – 1994)

Logisk uppmärkning, informellt samarbete, experimentlystnad.

Browserkriget (1994 – 1999)

Proprietära tillägg

Fysisk uppmärkning:

- Element vars enda funktion är grafisk
- Attribut vars enda funktion är grafisk
- Missbruk av element
 - Blockquote för indrag
 - Tabeller
 - Nästlade tabeller
 - "Spacer gifs"
- Felaktigt valda element
 - Huvudrubriken som "h3" för att "h1" blev för stor

Onödiga effekter (DHTML)

"How to build killer websites" Bibeln för "bad practice".

HTML och XHTML standarder

HTML 2.0

November 1995

Tabeller, m.m. kompletterade efter hand.

I praktiken bestämde Netscape vid denna tid.

Formaliserad som SGML med krav på doctype (sic!), men ingen brydde sig.

HTML 3.0

Kom och gick och ingen märkte den. Netscape bestämde i praktiken, som sagt.

HTML 3.2

W3C rekommendation i januari 1997.

Sammanfattade rådande praxis vid denna tidpunkt. Det som MSIE 4 och Netscape 4 hade gemensamt, plus/minus någon detalj.

Doctype – vad slags HTML är det?

Krävs enligt specifikationen från och med denna version. Används sällan i praktiken. Påverkar inte funktionen.

HTML 4.0

W3C rekommendation i december 1997.

Upprensningsaktionen inleddes!

Grundtanken: Separera innehåll från design. Låt HTML styra det första och CSS det andra. Tre versioner:

1. Strikt
2. Transitional
3. Frameset

Syftet var att all nyutveckling skulle ske med den strikta versionen, men ännu var webbläsarnas stöd för CSS för dåligt.

Doctypes? Använd aldrig denna version! Den innehöll buggar.

HTML 4.01

REC: December 1999, senast ändrad maj 2001.

Buggfixad version av 4.0. Samma tre versioner som HTML 4.0. När man kort säger "HTML 4" så avses i praktiken för det mesta "4.01".

Fram till dess att HTML 5 har fått ett utbrett stöd bland marknadens webbläsare är detta ditt ena huvudalternativ som utvecklare.

Doctypes – från och med nu har de också en praktisk funktion. Se kapitlet om CSS.

XHTML 1.0

REC: Januari 2000.

HTML 4 omformulerad som XML. Dvs. det är en tillämpning av XML, vars elementnamn och attributnamn sammanfaller med HTML 4.01.

Ditt andra huvudalternativ.

Doctypes

XHTML 1.1

REC: 31 maj 2001

Uppdelad i moduler

Stöd för Ruby

Skall – utan undantag – skickas som application/xhtml+xml och så länge som MSIE version 5, 6 eller 7 är en målplattform förblir denna version därför oanvändbar! MSIE kanske inte heller kommer stödja äkta XHTML.

Andra utgåvan. WD 16 januari 2007

Tillåter "text/html". Se <http://www.w3.org/TR/xhtml-media-types/>

XHTML 2.0

Ett evighetsprojekt...

Inte bakåtkompatibelt

Ingen entusiasm bland utvecklare

Stöd bland webbläsare är långt borta. Utan stöd från någon ledande webbläsare inom överskådlig framtid.

- href på nästan alla element.
- Men: role-attributet (Stöds av gecko)

Xforms

Otroligt kraftfullt, men inte lätt att använda.

Ett eget namespace (förklaras i nästa kapitel)

HTML 5 och XHTML 5

Ett initiativ som inte kommer från W3C, men under hösten 2006 ”erkändes” av W3C

Detta är framtidens HTML troligare än XHTML 2!

Bakåtkompatibelt

Två *serialiseringar*: HTML och XHTML (men illusionen om att det är SGML har övergivits!)

<http://www.whatwg.org/specs/web-apps/current-work/>

wiki.whatwg.org/wiki/HTML5_Presentations

http://wiki.whatwg.org/wiki/Differences_from_HTML4

Exempel på utvidgningar av (X)HTML

<aside>

<date>

Etc.

Web Forms

Den del som kommit längst

<http://code.google.com/p/webforms2/>

Canvas

En skriptad teknik för grafik, ursprungligen från Apple

Annat gruppen intresserat sig för

DOM och CSS

Doctype

<http://www.alistapart.com/stories/doctype/>

<http://www.w3.org/QA/2002/04/valid-dtd-list.html>

<http://hsivonen.iki.fi/doctype/>

http://en.wikipedia.org/wiki/Document_Type_Declaration

Exkurs: Relationen till SGML

I praktiken: *SGML-aktigt* språk, snarare än äkta SGML

HTML 5 överger också tanken på att det skall vara äkta SGML

Gecko stödjer vissa SGML regler med roligt resultat:

<http://www.thespanner.co.uk/2007/08/13/firefox-weird-javascript-execution/> Detta kan skapa stora säkerhetsproblem (lätt att missa något) och att inskräpa XHTML-syntax är ett skydd.

Fördjupning

HTML:s historia: <http://tools.ietf.org/html/rfc2854>

W3C:s standardiseringsprocess: http://en.wikipedia.org/wiki/W3C_recommendation

1. WD: Working Draft (Idé) – somliga browsers kanske redan stödjer tekniken!
2. CR: Candidate recommendation: Börja implementera nu! + Skapa tester.

Det är ett missförstånd att man inte kan börja använda tekniken.
”Significant features are locked.” Innan minst två större webbläsare stödjer tekniken kommer man inte vidare från denna nivå.

3. PR: Proposed recommendation.
4. REC: W3C recommendation. Färdig version.

5. Errata

HTML:s officiella sida på W3C: <http://w3.org/markup/>

WHAT-WG

<http://en.wikipedia.org/wiki/XForms> m.fl. artiklar på Wikipedia

XML

XML som sådant

XML 1.0 senaste specifikationen <http://www.w3.org/TR/xml/>

XML:s hemsida på W3C: <http://www.w3.org/XML/>

Javaskript för att implementera (X)HTML 5

1. Hur testas man generellt stöd?
2. Hur testas man specifikt stöd
3. Färdiga skript som implementerar funktioner

Några andra XML-tekniker (ämnade för klientsidan)

XHTML Basic – ”minsta gemensamma nämnare”: <http://www.w3.org/TR/xhtml-basic/>

XHTML for Print – för ett specialfall av utskrift endast. <http://www.w3.org/TR/xhtml-print/>

XHTML Mobile <http://www.openmobilealliance.org/> - ersätter WML för WAP 2.0

XFrames – ett försök att lösa problemen med ramar, används av ingen:
<http://www.w3.org/TR/xframes/>

XSL gör sig kanske bäst på servern, men kan användas direkt i webbläsaren:
<http://www.w3.org/Style/XSL/>

RDF

RSS

SMIL

SSML <http://www.w3.org/TR/speech-synthesis/>

7 Grundläggande CSS

Varför?

- Separation
 - Tänk inte ”med divvar”, utan säg ”CSS grids”
- Kraft

CSS Zen Garden

”Det kan man göra med mallar” -- men det är en sämre lösning!

En (X)HTML-kod för alla användare och alla plattformar:

- Stora skärmar
- Små skärmar (mobilen)
- Utskrift
- Talsyntes
- Blindskrift
- Etc.

Tänk dig att du mejlar en länk till en kompis. Han får versionen för stora skärmar, men tittar i sin mobil...

Tänk dig att någon hittat din sida via Google. Av något skäl – lätt begripligt för den som kan SEO – så hamnar utskriftsversionen före bildskärmsversionen. Den som anländer ser inte dina menyer och får en ganska tråkig design. Är det vad du önskar?

”Använda divvar” - fel tänkt! Använd semantisk (X)HTML-kod. Språkets inbyggda logik. ”Div” = en del av en sida, en ”division”.

CSS-regler

```
<code>
```

```
selektor {  
  egenskap: värde;  
}
```

</code>

Det som skiljer nedanstående från vanliga genomgångar är att tonvikten lagts vid **när** du skall använda en viss selektor.

<http://www.w3.org/TR/2006/WD-CSS21-20061106/selectors.html> (Permalänk till senaste?)

<http://www.w3.org/TR/css3-selectors/>

Typ-selektorn

Skriv elementets namn som selektor

Äkta XHTML är känslig för versaler kontra gemener. Tips: Använd alltid små bokstäver.

Ändra standardvisning av ett element.

Tips! Studera Firefox "html.css" noga för att se hurdan standardvisningen är.

Den universella selektorn

Täcker alla element.

"Global reset"

Se kapitlet om fördjupad CSS --> Browserstöd

Ett förnuftigt alternativ från YUI

star-html hack

Tanken: Bara till MSIE

Använd en bättre metod!

Se kapitlet om fördjupad CSS --> filter

ID-selektorn

Kom ihåg att ID är unikt för ett enda element. Använd aldrig som "klass"!

Avkomlingsselektorn

Alla element som finns inuti ett annat element.

Börja med avkomling och ta bara till klass om inget annat är möjligt.

Klass-selektorn

Nackdel kontra avkomling - ”coupling” till (X)HTML-koden.

”Klassitis” = överdrivet bruk av klasser

Grafiska namn är inte bra – använd logiska (se mer i avsnittet om mikroformat)

Pseudoklasser

:link :visited

Använd gärna som avkomlingar, typ inga understrykningar på menyns länkar.

:active :hover :focus

Bugg i MSIE behandlar :active som :focus

Använd inte :hover och CSS för att skapa ett beteende. Fascinationen gav oss en del exempel på spel och menyer endast gjorda med CSS. De demonstrerar kraften, skall ses som experiment, men är inte ”best practice”. Låt DOM-skript sköta beteendet!

Attribut-selektorer

Inte använda då inte MSIE 6 hade något stöd, men de stöds av MSIE 7 och framåt!

Dags att börja använda dem!

Kan reducera ditt behov av att använda klasser ytterligare.

Många fler i CSS 3, och ganska bra stöd i moderna webbläsare.

Att lägga in CSS

Inline

Bara under utveckling, för att snabbt testa ett alternativ!

```
<code>
```

```
Jag heter <span style="font-style: italics">Lars</span>
```

```
</code>
```

Är precis lika illa som – eller rent utav sämre – än

```
<code>
```

```
Jag heter <i>Lars</i>
```

```
</code>
```

Sidhuvudet

Också endast under utveckling.

Du bryr dig väl om kostnaden för bandbredd och serverbelastning? Att besökarens väntetid minskar? Att din lösning är så enkel som möjligt att vidareutveckla eller ersätta med en ny?

Externt

JA!

Det finns två sätt:

- link
- @import
 - Också från en CSS-fil till en annan

Tips

- Undvik ”divvitis”
- Kortformerna
 - Spar rader att skriva
 - Kan ibland återställa en regel utan att du tänkt på det
color: black; border-color: red; border: 2px solid; => Ger **svart** kantlinje!
- Generiska typsnitt
- Validera din CSS-kod
 - Utveckla inte i MSIE 6, helst inte i MSIE 7 heller!
-

Övningar

Titta på HTML-koden...

Skriv en CSS-kod som

- Gör all text till ”Verdana, sans-serif”
- Ge andra stycket en streckad kantlinje
- Några till.

Studera ett färdigt förslag på två-kolumn layout som kommer att användas i bokens meta-projekt

Fördjupning

Wikipedia

- http://en.wikipedia.org/wiki/Tableless_web_design
- http://en.wikipedia.org/wiki/Style_sheet_%28web_development%29
- Massor av artiklar som berör CSS

W3C

- ”CSS Under Construction” <http://www.w3.org/Style/CSS/current-work.html>

Referens

Forum, mejlinglistor

Browserstöd

CSS-versioner

- 1.0 (REC: Dec 1996) <http://www.w3.org/TR/REC-CSS1>
- 2.0 (REC: maj 1998) <http://www.w3.org/TR/REC-CSS2/>
- 2.1 – en anpassning av 2.0 till verkligheten <http://www.w3.org/TR/CSS21/> (”Last Call” innan CR status i november 2006 = snart REC!)
 - En delmängd av CSS 2.0 + några få tillägg.
 - @aural är uppdelad i två moduler. En för talsyntes (@speech) och en för ljudeffekter.
- 3.0 – delas upp i moduler som utvecklas relativt självständigt och är olika långt komna.

Somliga delar har gått tillbaka från CR till WD, då processen (och förhoppningsvis därmed kvalitetskontrollen) har blivit hårdare Några mycket efterlängtade tillägg:

- Fler selektorer
- Rundade hörn
- Multipla bakgrundsbilder
- Boxmodellen kan styras (se doctype switch)
- Media queries – skicka olika CSS till små och stora skärmar! (Idag måste man använda JS)

8 XHTML

Vad det är...

Vad det ämnades bli, men inte blev:

- I stället för HTML
- Skickat till XML-tolkar
 - MSIE har inget sådant tolkningsläge!
 - Majoriteten av alla påstådda XHTML-sidor sänds som text/html
 - Majoriteten av dessa sidor är inte ”well formed” och skulle inte kunna skickas som XML!

Teoretiska fördelar

Namnrymder: XHTML + MathML + SVG

Snabbare tolkning (JA! Från och med våren 2007 också för långa dokument.)

Praktiska fördelar

Du tvingas ha god disciplin – förutsatt att koden validerar!

Det mesta som krävs i XHTML, är önskvärt också i HTML. Exempelvis well-formedness.

Lättare att bygga en applikation *ovanpå* XHTML-dokumentet eller att utvinna data från det med tekniker som Xpath.

Som PHP-utvecklare vill du gärna ha välformad XHTML att bearbeta på servern!

Dock: Disciplin är viktigare än teknikval: Disciplinerad HTML 4.01 är mycket bättre än skräpig kod som påstår sig vara XHTML. Att koda i XHTML är ingen garanti för god semantik, inte ens om koden validerar.

Krav att uppfylla

Bara små bokstäver på elementens och attributens namn.

Liksom i HTML version 3.2 och framåt: Doctype är obligatorisk

Namnrymd skall anges

Välformad kod – önskvärt i HTML – ett krav i XHTML

Inga odefinierade entiteter – önskvärt i HTML – ett krav i XHTML

Citationstecken på alla attributvärden – önskvärt i HTML – ett krav i XHTML

Sluttagg på alla element

- Frivilligt, men önskvärt i HTML för:

- li
- td
- th
- tr
- script, med extern källa

- Helt onödigt i HTML för:

- br
- input
- img

OBS! PHP:s funktion nl2br skapar XHTML-aktiga br-element.

I HTML 5, serialiserad som just HTML, kommer detta troligen dock inte vara förbjudet.

Myter om XHTML

”Du måste använda XHTML om du vill ha en CSS-baserad design.”

”XHTML är framtiden” - svar: Båda serialiseringarna kommer leva i många, många år till.

”Visual Studio skapar god XHTML...” Ingen säger det, men på tok för många utvecklare som om det vore sant.

”Du måste ha ett mellanslag före avslutande 'slash större än” Ej för Gen 4 webbläsarna och framåt!

”HTML är inte framtidssäkert” - alla webbläsare kommer stödja all befintlig HTML under överskådlig framtid! XHTML 2.0 är **inte** bakåtkompatibelt!

”Du måste ha en XML-prolog.” Nej! Det är frivilligt och den förstör DOCTYPE-switchen på MSIE

6, dock ej på MSIE 7.

”Tillgängligare” för SEO och talsyntes. Nej!

Fördjupning:

MIME-typer för XHTML

application/xhtml+xml

text/xml

Etc.

<http://www.w3.org/TR/xhtml-media-types/>

Webbläsarstöd

<xml:base> och <xml:lang> i stället för <base> och <lang>

<http://wiki.whatwg.org/wiki/HtmlVsXhtml>

PCDATA och CDATA

Vad är det?

Problem som kan uppstå. Ex. Opera ignorerar din CSS.

Lösning – också bra ur andra synvinklar – ha all CSS och alla JS externt.

Avancerade problem

ID

ID är inte kort och gott ett namn, utan något som definierats i en DTD

Namnrymder och DOM-funktioner

Javaskripten slutar att fungera!

- createElementNS
- createAttributeNS
- Opålitligt stöd för innerHTML

- document.write alltid oanvändbart
- Etc

Inget på marknaden förekommande JS-toolkit fungerar med äkta XHTML!

Övningar

Konvertera HTML till XHTML

- Manuellt
- Med Tidy
- Validera!

Fördjupning: Debatten om XHTML kontra HTML

”Sending as ... considered harmful” www.hixie.ch/advocacy/xhtml (2002-09-08, men det dröjde till 2004 innan debatten vaknade på allvar)

Fler uppmärksammade inlägg

Mot XHTML

- annevankesteren.nl/2004/08/xhtml
- www.autisticcuckoo.net/archive.php?id=2005/03/14/xhtml-is-dead
- lachy.id.au/log/2005/12/xhtml-beginners
- www.webdevout.net/articles/beware-of-xhtml
- www.elementary-group-standards.com/html/why-do-you-use-html-4.html (Flera kände front-emd utvecklare uttalar sig)

Pro XHTML, innan den ”nya debatten” drog igång:

- www.nypl.org/styleguide/xhtml/benefits.html
- www.w3.org/MarkUp/2004/xhtml-faq
- www.alistapart.com/stories/betterliving/
- www.webstandards.org/learn/articles/askw3c/oct2003/

Pro XHTML, som en del av debatten:

- h3h.net/2005/12/xhtml-harmful-to-feelings/
- www.kurafire.net/articles/case-for-xhtml
- tantek.com/presentations/2005/09/elements-of-xhtml/

Medelvägen/klargörande artiklar:

- www.456bereastreet.com/archive/200501/the_perils_of_using_xhtml_properly/
- www.456bereastreet.com/archive/200601/html_or_xhtml_does_it_really_matter/
- www.sitepoint.com/article/html-37-steps-perfect-markup

Räcker inte detta, så finns det en skrälldus länkar på adressen hyperactive.com/h.php?doc=xhtml som alla handlar om HTML kontra XHTML.

Lägg märke till att i princip det inte finns några ”back-end” utvecklare i diskussionen.

HTML:s killer över XHTML? Javaskript!

XHTML:s killer? Mikroformat.

I HTML 5 kan man fortfarande använda sina XML-verktyg efter att indata har ”tokeniserats” så att de funkar. <http://code.google.com/p/html5lib/>

9 Att skapa god (X)HTML

Välformad

Enligt standarden – validerande enligt DTD

Enligt standarden – dess syfte, som inte kan maskinkontrolleras

Detta validerar, men är dålig kod:

```
<code>
```

```
<div class="heading_one">Huvudrubrik</div>
```

```
</code>
```

Exempel 2: "Bed and breakfast"

Validering

Validera tidigt, validera ofta!

Använd den som verktyg, inte bara kontroll.

HTML Tidy

Extension till Firefox

Extension till jEdit

Från kommandoraden

Tillägg till PHP

Tidy kan:

- Kontrollera enligt standard och WCAG
- Föreslå förbättringar av koden
- Konvertera till XHTML
- Rensa upp. Ex. Font-taggar blir klasser för CSS
- Genom att rensa upp i användardata kan man med Tidy på servern undvika XSS-attacker

- Kan *inte* skapa semantisk kod.

Dokumentation finns på...

Ett par exempel.

Övningar

Med validatorn

Hitta felet och åtgärda det

Med Tidy

Hitta felet och åtgärda det

Konvertera till XHTML

Self-closing `<script>`? Sällan uppfattat korrekt.

10 Avancerad (X)HTML

Hur kan man någonsin påstå att en back-end lösning är bra om dess *slutprodukt* inte är bra?

Merparten av dina PHP-skript skall producera (X)HTML. Hur skall det gå till om du inte kan HTML?

Vad du skall undvika

Allt som endast har grafiskt värde

Name, annat än som variabelnamn i formulär. Använd ID.

A-taggar som ankare. Onödigt. Använd ID.

Proprietära koder. (Gäller också CSS)

Bygg först enligt standarder – anpassa sedan till verkligheten (dvs. främst MSIE!)

Lär dig elementens semantiska funktion

Tänk dig att någon läser upp sidan för dig:

- Det gillar Google!
- Tillgänglighet
- Enklare handhavande med inre logik – inkl. självdokumenterande kod!
- Spin-off effekter
 - Få dina nyhetssidor upplästa för dig i MP3-spelaren

Lär dig elementens gruppering

Block – äkta och block-liknande (tabell, listor)

Inline

Ersatta

Tabell

Etc.

Lär dig det fulla registret av element

dfn, q, cite, label, etc.

Lär dig elementens fulla logiska register

Ex "scope" för <th>

Ex. "alt", "title" och "longdesc" på bilder. Vad är ett *gott* alt-attributvärde?

Lär dig "web patterns"

Återkommande strukturer

Utökad semantik

Med kloka class, role (XHTML 2.0), rel, rev, etc.

Mikroformat

Gemensamt överenskommen utökad semantik

Övningar

Gör om följande icke-semantisk kod till semantisk...

Bygg upp en "lorem ipsum" sida med en web pattern

Skriv en adress som hCard

Skriv en händelse som hCal

Mikroformat extension till Firefox

11 Användbarhet och tillgänglighet

Basfakta

Använd inte frames!

Vilket problem sökte man lösa?

Varför är det en dålig lösning?

Hur gör man idag? SSI eller PHP

Undantag? Det finns i princip alltid en bättre lösning, men: iframe som skriptas in för att Firefox inte tillåter *redigeringsläge* på annat än hela sidor

Öppna inte nya fönster

Vilket problem sökte man lösa?

Varför är det en dålig lösning?

Undantag? För icke-html dokument och efter en tydlig förvarning.

Ha en vettig URI för varje resurs

Distinkt

Lätt att säga, mejla (under 80 tecken) och bra för sökmotorerna

Hackningsbar

Utan www

Helst bara US-ASCII och a-z, 0-9, bindestreck, slash, understrykning och punkt

Dölj dina variabelnamn!

Dölj definitivt sessionsid! Säkerhetsproblem.

Dölj din back-end arkitektur

Exkurs: ”doPostBack” bland det dumaste som skådats på webben!

Aldrig ”post” som länk – aldrig ”get” för att utföra en åtgärd. Google Web Accelerator => Alla inlägg raderade (Man var ju inloggad!)

Visa inte onödiga information

Ogifta skall inte behöva se ett fält med bröllopsdatum...

Expanderande formulär en god idé

Din sida skall fungera med bara sin (X)HTML-kod

Så kan den te sig i en mobiltelefon

Så är den för sökmotorer

Så är den för talsyntes

Så kan den skickas till Lynx och gamla webbläsare

All design i extern CSS

Alla javaskript enligt ”icke-inkräktande” paradigmet.

Det finns många slags funktionsnedsättningar

Blindhet

Nedsatt syn

Färgblindhet

Nedsatt rörlighet (funkar din meny utan musen?)

Epilepsi

Dyslexi (kan också hjälpas av talsyntes)

Språksvaghet (annat modersmål till exempel)

Övningar

Testa verktyg

- Cynthia says
- TAW
- Fangs
- aDesigner <http://www.alphaworks.ibm.com/tech/adesigner>

- <http://colorfilter.wickline.org/> (Du måste testa en frame i taget på en sida med frames!)

Fördjupning

Vägledningen 24-timmarswebben

WAI

Webbplatser:

- GAWDS
- Etc.

Böcker:

- Användbarhetsboken
- Jakob Nielsen
- Steve Krug
- ”... and regulatory compliance”
- ”Tillgängliga webbplatser”

12 Front-end möter back-end

Samma designprincip: Separation of Concern

Samarbete i kontaktytorna!

På klienten:

- Innehåll och struktur
- Design och layout
- Interaktion, animation, beteende

På servern

- Presentationslogik
- Affärslogik
- Dataåtkomstlogik
 - Objekt-persistens
 - Databasabstraktion

MVC – Model, View, Controller

Olika roller

(en gång i tiden fanns det en ”webmaster” som gjorde allt, så icke idag!)

- Informatör
 - Kan vara dina användare (web 2.0)
- Designer
- Utvecklare
- Driftstekniker

Använd inte back-end tekniker när front-end dito är bättre

Ex. ”browscap” - ofta använd helt i onödan

Ex. Hellre CSS-filer än browsersniffning på servern

Använd inte front-end tekniker när back-end dito är bättre

Frames

Simulera inte frames med AJAX!

Undvik att aggregera information med AJAX

Dubblerad logik

Validering av formulär.

På klienten för att:

- Avlasta servern och minska bandbreddsutnyttjandet
- Öka användbarheten med snabb feedback

På servern för att:

- Somligt kan inte göras på klienten
- Säkerhet: Allt måste dubbekollas. Klientsideskontroll kan alltid förbigås.

Övningar

??

Fördjupning

Separation av innehåll från struktur

Mer om MVC

Diskutera ”PHP Application Design” på Sitepoint

13 Skript på klientsidan

Faser (enligt PPK)

1. Tunn fas (NS 3)
2. Tjock fas (Gen 4), DHTML
3. Tunn fas (åtstramning)
4. Tjock fas (AJAX)
5. En kommande tunn fas...?

Användningsområde

Förbättrad användbarhet!

- Förbättrad interaktion med drag och släpp...
- Rullgardinsmenyer (rätt gjorda) kan dölja onödiga alternativ tills de behöver bli sedda
- Formulärkontroll
 - Måste kollas på servern också!

Förbättrad åskådlighet

- Animationer kan förtydliga

”Plugga igen hålen” i dagens CSS

Beståndsdelar

- ECMAScript
 - Används också av ActionScript
 - Kan (i teorin) användas på servern
 - Delar av Firefox är skrivna i JS
- ”Netscape 3 de-facto standarder” (DOM 0)
- W3C DOM

- Window-objektet (BOM)
- XMLHttpRequest

Händelsehantering

Inline – känn igen, men använd aldrig!

DOM 0 – på mindre projekt, med ingen annan kod som kan tänkas störa

W3C

MS

”addEventListener” = abstraktion av W3C och MS

Överkurs: XML Events

Vad en expert bör kunna

- Toolkits (DOJO, YUI, etc)
- Objekt-*baserad* kodning (inte objektorienterad förrän ECMAScript 4)
- JSON
- Canvas
- E4X
- Javaskript 1.7
- (Koll på) ECMAScript 4 aka. Javaskript 2

Javaskript och PHP

PHP kan framställa javaskript

PHP kan interagera med javaskript (”remote scripting”/AJAX)

Liknande bekymmer

- Globala variabler kan förstöra en applikation fullständigt.
- Dragkamp i ”gemenskapen” mellan procedurell och objektbaserad/orienterad filosofi

Denna bok lär inte ut JS, men har många ”jämfört med JS” stycken, för att underlätta inläringen

för den som redan kna JS, eller för att hjälpa den som kan PHP att också koda JS.

"Best practices"

Capability testing = object detection (not code forking)

Nammrymder i objekt/Inkapsling (i väntan på libraries)

JSDoc

JSLint

Etc.

Övning

Testa kodexemplet

Ändra det så att...

Testa ett mer avancerat kodexempel

Ändra det så att

Skriv ett eget skript

Fördjupning

Böcker

Hemsidor och bloggar

http://en.wikipedia.org/wiki/DOM_scripting

Personer

- Brendan Eich
- Douglas Crockford
- David Flanagan
- Dean Edwards

- John Resig
- Christian Heilmann
- PPK och WASP DOM-scripting Task Force

Toolkits

YUI

JQuery

Dojo

Etc.

Några artiklar som beskriver nackdelar/faror med toolkits.

Många PHP-utvecklare tycker inte om JS och ser toolkits som ett sätt att slippa språket och utnyttjar dem därför dåligt!

Kan man ha andra språk än javaskript på klienten?

VBScript

IronMonkey: <http://wiki.mozilla.org/Tamarin:IronMonkey>

<http://ejohn.org/blog/the-browser-scripting-revolution/>

- Python
- Ruby
- PHP? (Också andra projekt finns.)

ECMAScript 4 på andra plattformar än Mozillas?

Konverteringsprogram: Koda JS 2 -> konvertera till JS 1.x

Screamingmonkey: <http://wiki.mozilla.org/Tamarin:ScreamingMonkey>

14 PHP – webbens ”klister”

Vad är PHP och vad kan det göra

Innehåll hämtas i huvudsak från motsvarande kapitel i ”PHP in a nutshell”

PHP i jämförelse med

C/C++

Syntax – användningsområde

Java/JSP

Syntax – användningsområde

C#/.NET

Syntax – användningsområde

Troligen ditt huvudalternativ

Perl

Syntax – användningsområde

Ruby

Syntax – användningsområde

Ett framework (ingen använder Ruby utan Rails...) kontra många frameworks

”Det går snabbt” kontra ”det kan gör allt”

Konventioner

PHP gemenskapen

Rasmus

Zeev, Andi - Zend

Tongivande: Sclossnagle, Sara G, Wez, Ilia, Chris Schiflett, etc.

Utkast: Modern webbutveckling med PHP
Version: 07-08-31

Sid 68 av 164

Stora: Yahoo, Omi TI

Webbplatser: Yahoo, Flickr, Digg, etc.

15 Basal PHP

Installation: Se appendix

PHP från kommandoraden

php -a Interaktivt läge <http://se2.php.net/manual/sv/features.commandline.php>

- e "Extended information for parser/debugger"
- f PHP-kod i en fil
- i Information
- l Syntax check (lint)
- m Lista moduler

Tips: "pajpa" "php -i" och "php -m" till grep eller less

man php

php -h

PHP på webbservern

Även från kommandoraden (eller PHP-Gtk+) så skapas en slags "server". Allt går via ISAPI!

1. Anrop till Apache (eller IIS)
2. Filter: Ex. "Mod rewrite"
3. PHP
 - a) Parse
 - b) Opcode
 - Kan cachas!
 - c) Exekvering inkl. kommunikation med andra servrar/datakällor
 - Databas
 - Webbtjänster

- Lokala filer

4. Lämna resultatet till Apache (IIS), som sänder till klienten

Nybörjartabbe: Att tro att webbläsaren kan tolka PHP!

Hur PHP bäddas in

Processinstruktionen:

- `<?php ?>`
- Kort form:
 - Kan strula med XML
 - Kan vara avstängd
 - Bekväm variant: `<?= ?>` men den har samma problem
 - `<?php= ?>` blir troligen inte av
- ASP-form
- `<script language="php">` Används aldrig i praktiken

Man kan hopp in och ut hur många gånger som helst.

Tips!

- Blanda inte bearbetning med output.
- Om dokumentet slutar med php-kod, så strunta i att avsluta med `?>`

Kommandot echo

Grundläggande syntax

- echo är inte skiftlägeskänslig
- echo är **inte** en funktion, utan en *språkkonstruktion*.
- echo "sträng"
- echo x (siffra)
- echo \$foo (variabel)

Tips

- Parenteser är onödiga och ökar sällan läsbarheten
- PHP saknar ett writeln kommando. Lägg på `”\n”`.
- **print** funkar till 99 % likadant
 - Har returvärde
 - Kan inte ta emot komma-separerade argument

Kommentarer

C-stil

C++ stil

Bash stil

PHPDocumentor

Övningar

”Hello world” från kommandoraden med `”php -a”` och `”php -f”`

Tips: Nyradstecken som avslutning på alla `”echo”`

Hello world inbakad i en webbsida.

Fördjupning

Böcker om PHP

In a nutshell

Etc

Webbsidor som introducerar PHP (och inte är för gamla)

Foo

16 Utvecklingsmiljön

Serverar

Utveckla inte på den offentliga servern!

1. Utveckling
2. Ev. Testning (gemensam)
3. Skarp

Bara trean är *offentligt* tillgänglig.

Editor

Radnumrering

- Syntax highlighting
- Hitta matchande parentes, måsvinge, dubbelfnutt, etc.
- Markera fel

Editor, ytterligare krav

- Balansera måsvingar och gärna (X)HTML-taggar
- Smart hantering av indrag
 - Tips! Använd fyra mellanslag. Inte tabb.
- Dölj/visa kodblock
- Sök/ersätt med regex
- hints - vem sjutton kommer ihåg parameterordningen?
- Teckenkodning
 - Tips! Akta dig för ”Byte Order Mark” (BOM)
- Etc

En fullfjädrad IDE

Allt ovan plus:

- Förhandsvisning
- Debug
 - Stega sig igenom koden
 - Stacktrace
 - Breakpoints
 - Etc
- Integrering med CVS/subversion eller liknande.
- Projekthantering
- Etc

ASP har Visual Studio...

1. Eclipse
2. Zend studio (snart flyttad till Eclipse?)

17 Variabler i PHP

Vad är en variabel?

Som en låda

Med en etikett

De enkla värdetyperna

Heltal

Flyttal

Strängar

Boolska

Null

Variabelnamn

Skiftlägeskänslighet

`$foo != $Foo`

Legala tecken

Inte börja med siffra

alnum + underscore

ääö OK, men kanske inte smart (förr eller senare vill du ha hjälp på ett forum eller irc och den som hjälper dig kan inte skriva tecknen med sitt tangentbord)

Bra variabelnamn

Beskrivande – långa!

Livslängd?

- Sant globala: ANVÄND STORA BOKSTÄVER
- Långlivade: Mycket tydliga namn
- Kortlivade: Kortare namn
- Konventioner: \$i för loopar

”Loosely typed”

Vad som menas

Oväntade resultat

Manuell typecasting

Konstanter

(Enkla) operationer

Matematik

Konkatenering

Varför älska PHP? Slå ihop en sträng och en siffra!

I JS vet jag inte (?) om det blir en sträng eller siffra

I MySQL blir det en ”BLOB”

I C måste man skriva en speciell funktion för åtgärden!

I PHP har jag kontroll, tack vare skilda operatorer.

Jämförelser

Kontroll av typ

- isset
- empty
- is_numeric
- is_integer
- gettype
- Etc

"By reference" och "by value"

Referensräkning

Du behöver inte städa (liknar JS)

Du har inte exakt kontroll över när en variabel förstörs (OO-kodning)

Globala variabler

Varning! Kollisioner.

Vad göra? (Dessa tips gäller också för dina funktionsnamn.)

- Kodningskonventioner. Äkta globala variabler skall ha STORA bokstäver endast i sitt namn.
- Namnrymder av C-typ: \$BOKDEMO_MIN_GLOBALA_VARIABEL

Varning: "Coupling"

Vad göra? Lär dig patterns!

Avancerade variabeltyper

Snabbt uppräknade bara:

- Arrayer
- Objekt

- Resurser

(Några) pre-definierade variabler

\$_GET

\$_POST

\$_SERVER

\$GLOBAL

Övningar

Främst övningar som kan göras från interaktivt läge

Fördjupning

Variabler är inte objekt!

De är referenser

Inte ens objektvariabler!

Pre-definierade konstanter

Vanliga

Smarta (typ ”ny rad”. Hat tip Emil Hernvall)

Felhantering

Per tillägg

18 Funktioner

Xx

Skiftlägeskänslighet

Nej!

Variabel "scope"

Globalt

Lokalt

\$GLOBAL

Skicka parametrar by reference

Default value för en parameter

Övning

Fördjupning

Retur by reference

Xx

Statiska variabler

Xx

De är inte objekt

Som i JS

De skickas som strängar. Ex array_map()

De kan anropas dynamiskt

```
$func = 'strpos';
```

```
$func('foobarfoo','foo');
```

Fungerar inte med *språkkonstruktioner*.

<http://se.php.net/manual/sv/functions.variable-functions.php>

De kan skapas dynamiskt

```
create_function
```

Ovanligt

Funktionshanterande funktioner!

<http://se.php.net/manual/sv/ref.funchand.php>

Funktioner följer extension

500 i grund-extensionen

Inkluderade som standard

Windows: Inkluderad \neq aktiverad

PECL

19 Objekt i PHP

Vad är ett objekt?

Något som har egenskaper och metoder.

Objektterminologi: Instanser tillhör klasser.

Helt annorlunda i JS: Lambda.

Exkurs: PHP stödjer lambda, men inte closures. I praktiken ovanligt.

Klassdefinitionen

Klassnamn

- Namnrymder för att undvika kollisioner
- Ett av skälen till varför du bör använda objekt är att slippa kollisioner (variabelnamn, funktionsnamn, konstantnamn)

Konstruktorn

Variabler

Metoder

Konstanter

Åtkomst: Private, protected, public

Statiska metoder

- Deklarera dem statiska, annars tappar du fart!

20 Fel, fel, fel!

När man börjar utveckla i PHP dröjer det inte länge innan något fel uppstår. PHP har en nivå-uppdelning av sina felmeddelanden, beroende på svårighetsgrad:

- ”parse error” - inläsningen av skriptet misslyckas. Opcode kan inte skapas. Ingen output.
- ”fatal error” - exekveringen är igång, men avbryts när felet uppstår.
- ”recoverable error” (ny i PHP 6)
- ”warning”
- ”notice”
- ”strict standards” (ny i PHP 5)

Kodningsfel

I förhållande till PHP:s regler

- Balans i måsvingar
- Balans i parenteser
- Balans i citationstecken
- Paamayim nekutadayim
- Etc.

Tips! Ibland syns inte felet på den rad som det verkligen finns på. Ibland säger PHP att det är fel på rad 63 men du har bara 62 rader! Nästan alltid en måsvinge som saknas. Fuska inte med indragen.

I förhållande till din applikations logik

Ex. Miss av stora/små bokstäver i ett variabelnamn

Ex. Skickat fel slags objekt till en funktion.

Ex. Glömt att ge en variabel ett värde.

Tips! Utveckla alltid med högsta nivån på felrapporteringen.

Tips! Bygg in kontroller i din egen kod.

User error

Fel man kan skapa själv.

Exceptions

Nytt i PHP 5

Alltid "fatal" men kan "fångas"

Övningar

Hitta felet

Olika "ini_set" och jämför output

print_r och var_dump

21 Styr upp din PHP-miljö

Länge leve å, ä och ö. Våra tre (nästan) unika svenska bokstäver gör livet värt att leva! Nja, kanske inte, men nog vill jag som svensk att också PHP skall förstå mina intentioner fullt ut. När jag säger åt PHP att alla bokstäver är ok i användarnamn, så vill jag att hänsyn skall tas till språket som för stunden används. Jag vill att ”nu” skall vara där jag är (eller rättare sagt, där min applikation är virtuellt placerad) – inte där servern är placerad. Det senare kanske på ett amerikanskt webbhotell. Jag vill rentav kunna flytta från ett svenskt till ett amerikanskt webbhotell utan att behöva skriva om min kod.

PHP har en mängd inställningar som hjälper mig på vägen. I PHP 6 är en av nyheterna att lokalisering har blivit enklare, buggfriare och mer exakt. Men det finns andra inställningar som jag gärna vill bestämma över: Hur skall fel hanteras? Var ligger de filer jag vill inkludera med funktioner och klasser? Vilka funktioner skall avaktiveras av säkerhetsskäl.

Låt oss ta en titt på **var** PHP hämtar sina inställningar, och **vilka** inställningar man bör fundera på att alltid göra.

Globala värden

Om man kör kommandot `phpinfo()`, så visas två slags värden upp: globala och lokala. Globala inställningar är de standardinställningar som gäller hela servern.

När PHP startas

`php.ini`

`httpd.conf`

Du måste starta om webbservern innan ändringen ”tar”!

Lokala värden

Per-directory i `httpd.conf` (omstart krävs)

`.htaccess`

Per skript

Förhandstitt

- Locale
- tidszon

Det är lämpligt att ha dessa här så att portabiliteten maximeras

Felhantering

Autoladdning av klasser

spl_autoload_register är bättre än __autoload

Etc.

Min standard "init.php"

Som ett exempel

+ En mindre version för boken

Om magic_quotes_gpc är på – *vägra* köra vidare!

Kontrollera inställningar

phpinfo() har globala och lokala värden

php -i

get_defined_functions()

get_loaded_extensions()

php -m

ini_get

Kontroll av enskilda saker

- if (magic_quotes_gpc) ...
- date_default_timezone_get()
- \$_ENV

Inställningar att lägga märke till

Magic quotes

Finns inte längre i PHP 6!

Aktivera extensions i Windows

Gör dll aktiv

Fel? Tag bort en i taget tills det funkar.

Sökvägar

Gås igenom i detalj i kapitlet ”Separera innehåll från struktur med PHP”

Safe mode

Hur det funkar

Open basedir

Do

Allow url-fopen

Do

Övning

Tag reda på...

Ändra i httpd.conf (gör backup först!)

Ändra i php.ini

Ändra i .htaccess

Fördjupning

Göm känsliga variabler i httpd.conf

”Lazy loading” t.ex. av databasanslutning

”The registry pattern”

- Undvik globala variabler
- Undvik ”coupling”

22 Separera innehåll från struktur med PHP

Presentationslogik

Använd PHP som mall

Enkelt exempel – som en "iframe"

Första utkast

Säkerhetsproblem!

Whitelist

OK!

Språkkonstruktioner

include

require

include_once

require_once

Mer verkligt exempel: En modulär sida

En page-controller

Filtrera indata

Bearbetning

Visa med hjälp av mallarna

Master-template

Inkluderar delarna. På sätt och vis motsvarande ett frameset.

HTTP-huvudet allra först!

MIME

Teckenkodning

Felkoder

HTML-huvud

Variabel för title

Main

Spalt

Sidfot

Jämfört med frames

Varje frame ett komplett dokument! (Fast somliga fuskar tyvärr med det och förvärrar skadan än mer!)

Varje frame kan anropas via http. De delar som sätts ihop av PHP bör skyddas från den möjligheten!

Ytterligare fördelar kontra frames:

- Färre http-anrop
- Mindre bandbredd

Katalogstruktur

Webbrot

Ej http-anropbara resurser

Sökvägar i PHP

Absoluta

Relativa

PATH_SEPARATOR

DIRECTORY_SEPARATOR

Försök ha absoluta sökvägar för fartens skull, men tänk på portabiliteten också!

get/set_include_path()

Övning

Bygg din egen modulära sida

Fördjupning

Dedikerade mallsystem

- Smarty
- Savant
- Flexy
- Etc

XSLT

23 Olika slags PHP-skript

Kontroller

Page-controler

Ett skript per typ av sida eller uppgift

Front-controler

Ett skript för alla sidor – ej behandlat i den här boken

Laddar in moduler för varje slags sida eller typ av uppgift

”The command pattern”

Mallar

Master mallar

Del-mallar

Kompilerade mallar

template_c

Konfiguration

En nödvändigtvis PHP-syntax

För PHP-miljön (”init.php”)

Databasuppgifter

Applikationens inställningar

i18n-filer

Fraser på alla tänkbara språk.

Tips! Följ iso-standarden för språknamn i URL:er, men PHP:s språkfunktioner (PHP 6+) använder CLDR: <http://www.unicode.org/cldr/>

Tips! Lägg inte allt i en enda groteskt stor array!

Dokumentation

PHPDocumentor får ett eget kapitel

Cache

Indata från andra servrar och databaser

Utdata till besökarna

- Inbyggd funktion i mallsystemet (smarty)
- Kan finnas i ditt framework
- PEAR
 - PEAR2
- Egenhändigt gjord

Olika slags utdata-cachning

- Vanlig (tidsbaserad)
- Statiska kopior
- Server-side proxy

Funktioner

Bibliotek

Klasser

PEAR-konventionen för namngivning

PEAR

Exempel bara här

Framework

Ex. Zend

Övning

Bygg upp ditt katalogträd

Fördjupning

Var i övrigt ”dräller” PHP med filer?

Temp-katalogen

Sessionsdata

Var finns php.ini?

24 Lokalisering

<http://www.gravitonic.com/talks/>

Vad menas med i18n och lokalisering?

Aspekter

Skrift

Vilket ”alfabet”?

Vilken riktning?

Språk

- Kollationering – i vilken ordning skall saker sorteras?
- Decimaltecken och tusentalsavgränsare
 - Alltid punkt internt för PHP!
- Valuta
- Etc.

Tid

- Hur den skrivs
- Tidszoner

Symboler och färger

Följer kulturer

Inställningar i PHP

PHP 6 nedvärderar `setlocale()` som ersätts av `locale_set_default()` samt `locale_get_default()` vilken byter från POSIX-locales till ICU-locales internt.

Tidszon i PHP 5.1 och framåt

Tidszon före PHP 5.1

strftime följer locale

Kort om teckenkodning

ASCII

Latin-1 (Windows motsvarande)

UTF-8

Teckenkodning i PHP

Indata (från användare eller databas)

Skriptet i sig

Utdata

PHP <= 5.3

Förutsätter latin-1

XML-funktionerna använder UTF-8

utf8_encode/utf8_decode

iconv

mbstring

Överladdning

PHP 6

Förutsätter UTF-8, men använder UTF-16 internt.

Ini-inställningar för indata, skriptets egen teckenkodning och utdata

Kan styras med .htaccess och ini_set?

Säkerhetsaspekter

Ex. UTF-7 användes för XSS mot Google

Övningar

Sortera med olika locales

Skriv datum med olika locales

Skicka "ääö" som utf-8 och latin-1

Fördjupning

Resurser

W3C

Unicode

Etc

Språk man inte alltid tänker på:

- Teckenspråk
- Invandrarspråk (även på en "svensk" webbplats!)

Mer om tecken

Codpage

Encoding

Glyf

Font

UTF-8 som en textsträng i PHP

`\nnn`

`\xnnn`

Teckenkodning och (X)HTML-entiteter

Från början: Bara US-ASCII

HTML 4.01 har latin-1 som standard!

Många skriver fortfarande i onödan å

XHTML har UTF-8 som standard.

Med UTF-8 behövs nästan inga entiteter.

Hur webbläsaren väljer teckenkodning

Text/html

1. Http
 - a) Att INTE ange är också en angivelse!
2. Meta-tag
3. Byte order mark
4. Intern ”heuristik”

application/xhtml+xml

1. Http
2. XML-prolog
3. meta-taggen skall ignoreras!

25 Språkkonstruktioner

C-lik syntax

Tips! ”Lint” ditt program från kommandoraden

Block

Gruppering med ”måsvingar”

Jämfört med Python: PHP:s indrag är för min skull, inte tolkens.

Jämfört med C: Inget ”block-scope”

if - elseif - else

Normal syntax

Med kolon-slutord

Jämfört med BASH

Alternativa sätt att göra if-strukturer

`funk1() && funk2()` detsamma som `if (funk1()) funk2`

`funk1() || funk2()` detsamma som `if (! funk1()) funk2`

Men: `$foo = funk1() || funk2()` evaluerar till sant eller falskt – och fungerar alltså inte som i javaskript.

Avancerade villkor

`||` AND XOR etc

Ternär operator

While och do-while

For

Switch

Syntax

Mycket switch? Dags för OO och *delegeringsmönstret*!

Övning

Fördjupning

Break

Continue

Multipla startvärden för ”for”

switch true – case expression

26 Indatakontroll

Mantra: Filter input, escape output

SQL-injection

XSS

Kom ihåg att vi inte förlitar oss på `magic_quotes_gpc`!

Tvinga data att bli någorlunda vettig

Typecasting

Trim()

`ltrim()`

`rtrim()`

Tänkt att vara HTML?

`strip_tags()`

`htmlspecialchars()`

`htmlentities()`

tidy får eget kapitel

Hur ta bort attribut?

- Regexp
- XSLT

<http://htmlpurifier.org/>

Regelrätt kontroll

Finns värdet alls?

`empty()`

`isset()`

Min- och maxlängd av textsträng

Med utf-8 är antal bytes != antal tecken!

`strlen()`

Kontroll av vilka slags tecken som ingår

Ctype

(Regexp ofta ett sämre alternativ)

Kontroll gentemot en whitelist

`in_array()`

`array_key_exists()`

Kontroll av mönster

Använd `preg` – inte `ereg`!

Genomgång i eget kapitel

Indatafilter

`Filter_var()`

Strategi

1. Basala förändringar, typ `trim`, `stripslashes` och `tidy-first run`.
2. Validering
3. Definitiv förändring av indata

Sedan bearbetning

Lagra icke-escapad data

Undantag: htmlspecialchars på rubriker och annat liknande?

Escape-ning när något visas.

I kontroll-skripten. Den som gör mallen skall inte behöva bry sig om säkerhet!

Övning

Testa funktioner

Bygg in kontroller i dina mallar

Fördjupning

Avancerat bruk av indatafilter (PHP 5.2+)

Filter_array()

27 Tid

Standardformat för tid

Cookies, http, Etc

- ISO formatet
- hCal-formatet

Etc

Unix-tid

PHP tid

Utgå från Unix-tidsstämpel

Xx

Till och från

date()

strftime()

strtotime()

Etc.

Tids-arrayer

DateTime-objektet

PHP 5.2.2+

DateTime-objektet: <http://se.php.net/manual/en/function.date-create.php>

Använd din DBMS!

Eftersom PHP:s tidsfunktioner är mindre pålitliga (PHP <= 5.0)

Eftersom PHP har svårt med datum för långt bak eller för långt fram.

Eftersom DBMS:en har mer lättlästa tidsformat

Men: Somliga lagrar tiden som UNIX-timestamp i sin DBMS... Det gillar inte jag.

http-huvuden och cachning

När skapades dokumentet egentligen?

Måste det skickas, det kanske finns cachat?

Cookies

Övningar

Skriv ut dagens datum i följande format:

- Fredag 12 september 2007
- Etc

If-modified-since

Setcookie

Fördjupning

Tidsarrayer

Klocka din applikation

Leta efter flaskhalsar

Premature optimization...

28 Procedurella page-controlers

Styrskript

Stegen för att visa en vanlig sida

1. Sätt upp miljön
2. Kontrollera användar-indata (GPC + sessioner)
3. Hämta data från sina källor
 - a) Databaser
 - b) Textfiler
 - c) XML-filer
 - d) Webbtjänster
 - e) Etc
4. Bearbeta denna data
5. Presentation av data
 - a) Felsidor
6. Ev. Ytterligare loggning

Dessa steg gäller också om det är en webbtjänst eller data skickad till ett AJAX-skript, etc.

Ett enkelt exempel

Visa en artikel

Simulerad databas än så länge

Stegen för att lagra data

Man kan kombinera och både lagra och visa i samma skript.

I princip samma flöde

1. Sätt upp miljön

2. Kontrollera användar-indata (GPC + sessioner)
3. Ev. Bearbeta denna data ytterligare
4. Kommunicera med databasen (eller skriv till textfilen, etc).
5. Presentation av resultatet
 - a) Ev. genom att omdirigera till en vanlig page-controler
 - b) Felsidor
7. Ev. Ytterligare loggning

Förhindra dubbellagring

reloadknappen/F5

29 Arrayer

Xx

Numeriska

Associativa

Blandade

Multi

Jämfört med andra programmeringsspråk

De är inte objekt

De är inte av skilda sorter

Blir ”struct” för XML-RPC om arrayen är associativ

De är otroligt kraftfulla. Kan ofta ersätta objekt.

De har en enorm mängd funktioner. Byter man till JS känner man sig handikappad!

Foreach

Sortering

Andra array-funktioner

Array-lik objekt

SPL

Ex. loopa igenom ett databasresultat

Övning

Konstruera arrayer

var_dump av arrayer

”Allsvenskan”

Fördjupning

Array-cursor

list()

each()

rewind()

Etc

Sortering av multi-arrayer

array_multisort()

30 Enkla databasexempel för PHP 5.2+

Detta kapitelutkast är ett utdrag från en planerad bok av Lars Gunther om webbutveckling med PHP. Det är fritt att citera och redistribuera detta kapitel, under förutsättning att:

1. *Det sker ideellt, dvs. att ingen betalning utgår.*
2. *Att källan tydligt anges.*
3. *Att det inte modifieras.*
4. *Kodexemplen får modifieras och användas enligt i desamma angiven licens.*

Licensform: Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License

Detaljer om licensen: <http://creativecommons.org/licenses/by-nc-nd/3.0/>

Copyright: Lars Gunther

Förändringar:

- 2007-07-01 Statusbeskrivning av säkerheten med PDO prepared statements och när det är aktuellt att avaktivera emulerade dito för MySQL. Rör kommentarer i exempel 4 och tipsruta till exempel 6.

PHP är berömt för att kunna ansluta till i princip varje tänkbar databas som finns. Traditionellt har PHP dock använts till övervägande del, med den databashanterare (DBMS) som heter MySQL. Det gränssnitt som används av tradition känns lätt igen på att alla kommandon börjar med *mysql_*.

Sedan version 5.0 av PHP så finns det ett förbättrat gränssnitt som kallas *mysqli*, där bokstaven *i* står för ”improved”. Det gränssnittet fick ingen större spridning och du kommer troligen inte hitta så många kodexempel med det. I och med version 5.1 av PHP så finns nämligen ett tredje gränssnitt, PDO, som har alla fördelar från *mysqli* och sedan ytterligare ett antal. Utvecklarna av PHP är helt överens om att framtiden är PDO!

I detta kapitel skall jag presentera ett antal exempel på hur man kan ansluta till en databas och utföra några enklare uppgifter. Stegen för att använda PDO är:

1. Anslut till DBMS och vald databas. Skapa ett objekt som håller koll på denna anslutning.
2. Skapa en fråga, tag hänsyn till funktionalitet, säkerhet och fart.
 - a) Enkel indatakontroll, filtrering.
 - b) ”Escapa” indata, om man inte använder ”prepared statements”.
3. Utför frågan. Ett ”frågeobjekt” returneras.

4. Hämta och ev. loopa igenom resultatet
 - a) Resultatet kan bearbetas (affärslogik)
 - b) Resultatet kan visas (presentationslogik)
5. Hantera fel

Snabb säkerhetsinfo

Inget brukar vara känsligare i en webbapplikation än den data som ligger i databasen. Skulle någon manipulera PHP-skripten, är de ofta lätt återställda från en backup, men manipulerad data är ofta mycket svårare att återställa. Regelbundna backuper av databasen kan minimera skadan, men inte alltid till 100 %. Affärskritiska applikationer måste naturligtvis ha mer tillförlitlig säkerhet än ett hobbyprojekt, men att tänka rätt från början skadar aldrig.

Du bör aldrig förvara det användarnamn och det lösenord som skripten använder för att koppla upp sig mot en DBMS⁷, så att de kan läsas av andra. Under inga omständigheter skall dessa uppgifter förvaras i en fil som återfinns under webbroten. (Se kapitel 21.)

Använd heller aldrig ett webbhotell som inte har en s.k. chroot-miljö för alla användare på en delad server. Om webbhotellet erbjuder SSH-access, förvissa dig om att också den uppkopplingen är ”chrootad”! Om dessa villkor är uppfyllda, så är miljön säker nog för dina lösenord att ligga i klartext i en settingsfil, för ett hobbyprojekt eller inledande semi-professionella projekt.

Eftersom vi måste hantera lösenorden i klartext, så är det viktigt att du använder ett **unikt lösenord för varje applikation**, så att det inte kompromissar en annan applikation om det skulle råka komma på avvägar. Det har exempelvis hänt att folk av misstag postar sina lösenord på hjälpforum, när de ber om just hjälp. Är det lösenordet samma på applikation efter applikation, så blir det ett hiskeligt jobb att ändra alla.

En god regel är dessutom att låta det databaskonto som PHP-applikationen körs med ha ett kritiskt minimum av behörighet. Det skall vanligtvis kunna köra SQL⁸-kommandona SELECT, INSERT, UPDATE och DELETE, men sällan mer än så.

Exempel 1: Anslutning till databas och hämta några rader data

<code>

```
<?php
/**
 * bokdemo/enkla-databasexempel/exempel-1.php
 *
 * Koppla up mot databas, utför en enkel fråga och dumpa resultatet.
 *
 * @author    Lars Gunther <gunther@keryx.se>
 * @copyright Lars Gunther <gunther@keryx.se>
```

7 DBMS = Database Managements System, databasserver, det program som hanterar databaser.

8 SQL = Structured Query Language, det språk som man använder för att ”tala med” en relationsdatabas.

```
* @license Creative Commons Attribution-Noncommercial-Share Alike 3.0
*          {@link http://creativecommons.org/licenses/by-nc-sa/3.0/}
* @package bokdemo
* @subpackage databas
* @filesource
*
* @todo Detta skript är ett demo-skript (D) och det har allvarliga
brister (V)
*/
// Lösenord och användarnamn förutsätts finnas i variablerna
// $dbpass och $dbuser
// Varning: Lägg inte dessa uppgifter i en fil som återfinns
// under webbroten!
$dbuser = 'bokdemo_php';
$dbpass = 'hard_to_guess';

// Variabeln $dsn - utläst "Data Source Name" - talar om
// vilken slags DBMS och vilken databas som skall användas
// och på vilken värd (här samma som webbservern) DBMS finns.
$dsn = 'mysql:dbname=bokdemo;host=localhost';

// Steg 1: Själva anslutningen, den kastar som standard
// en PDOException om den inte skulle lyckas
try {
    $db = new PDO($dsn,$dbuser,$dbpass);
}
catch (PDOException $e) {
    // Ett fel uppstod, anslutningen kunde inte göras
    // Varning! Skriv inte ut felmeddelanden till hela världen på
    // detta sätt!
    echo '<h1>Uppkopplingen till databas misslyckades</h1>'.PHP_EOL;
    echo '<pre>'; // Ifall meddelandet formatterats över flera rader
    echo $e->getMessage();
    echo '</pre>';
    exit;
}

// Steg 2: En enkel SQL-fråga
$sql = 'SELECT * FROM users';
// Steg 3: Utför frågan
$stmt = $db->query($sql);

// Lyckades frågan? I denna demo tas det för givet.
// Steg 4: Det statement-objekt som returnerats som svar kan
// kan behandlas som vore det en array
echo '<pre>'.PHP_EOL;
foreach ( $stmt as $row ) {
    var_dump($row);
}
echo '</pre>';

</code>
```

Exempel 2: Samma som ovan, fast data visas i mallen

Vi skapar en begynnande uppdelning mellan å ena sidan ”model” (dataaccess) och ”controler”

(huvud-skriptet som körs) och å andra sidan ”view” (presentationslogiken), här i form av en simulerad mall.

Den oerhört behändiga metoden `PDOStatement::fetchAll` används. Man bör komma ihåg att den inte är lämplig på stora resultat, då den kan skapa en gigantisk array, med allt vad det innebär av minneskrav. Eftersom ingen bearbetning görs alls av informationen från databasen, så hade man kunnat skicka statementobjektet direkt till mallen och loopa runt det som i förra exemplet.

Personligen är jag heller inte förtjust i att kontrollera om fel uppstod i efterhand, som i detta exempel. Se vidare i exempel 4 för en lösning på detta.

<code>

<?php

```
/**
 * bokdemo/enkla-databasexempel/exempel-2.php
 *
 * Koppla up mot databas, utför en enkel fråga och lämna resultatet till en mall
 *
 * @author    Lars Gunther <gunther@keryx.se>
 * @copyright Lars Gunther <gunther@keryx.se>
 * @license   Creative Commons Attribution-Noncommercial-Share Alike 3.0
 *            {@link http://creativecommons.org/licenses/by-nc-sa/3.0/}
 * @package   bokdemo
 * @subpackage databas
 * @filesource
 *
 * @todo      Detta skript är ett demo-skript (D) och det har allvarliga
brister (V)
 */

// Kursiva rader är identiska med det förra exemplet
$dbuser = 'bokdemo_php';
$dbpass = 'hard_to_guess';
$dsn    = 'mysql:dbname=bokdemo;host=localhost';
try {
    $db = new PDO($dsn,$dbuser,$dbpass);
}
catch (PDOException $e) {
    // Varning! Skriv inte ut felmeddelanden till hela världen på
    // detta sätt!
    echo '<h1>Uppkopplingen till databas misslyckades</h1>'.PHP_EOL;
    echo '<pre>';
    echo $e->getMessage();
    echo '</pre>';
    exit;
}

$sql = 'SELECT username,auth_level FROM users';
$stmt = $db->query($sql);

// Här börjar det nya
// Lyckades frågan? PDO:s standardinställning är "tyst" vid misslyckanden
// Låt oss testa resultatet
// Varning! Skriv inte ut felmeddelanden till hela världen på detta sätt!
if ( false === $stmt ) {
```

```
// Kommer vi hit har frågan inte gått att utföra
// Testfråga som skapar fel: 'ZSELECT * FROM users'
$fel_array = $db->errorInfo(); // En array med tre poster
echo <<<FEL
  <dl>
    <dt>SQLSTATE error Code (ANSI-felkod)</dt>
    <dd>{$fel_array[0]}</dd>
    <dt>Drivrutinsspecifik felkod</dt>
    <dd>{$fel_array[1]}</dd>
    <dt>Drivrutinsspecifikt felmeddelande</dt>
    <dd>{$fel_array[2]}</dd>
  </dl>
FEL;
  exit;
}

// Vi flyttar svaret över till en multi-array
$userdata = $stmt->fetchAll();

// Frågan kunde utföras, men den har kanske inte returnerat något innehåll
// Testfråga som lämnar tomt innehåll: 'SELECT * FROM users WHERE 0'
if ( empty($userdata) ) {
  // Ett uteblivet resultat kan tänkas föranleda att man
  // skickar HTTP-statuskoden 404 (Not Found)
  // Varning! Detta är bara "proof of concept", felmeddelanden till användaren
  // bör vara (1) mindre tekniska och (2) mer användbara!
  header('HTTP/1.1 404 Not Found');
  echo '<h1>Databasfrågan returnerade ett tomt svar</h1>';
  exit;
}

// Varning! Följande kod är tänkt att illustrera en mall och eftersom man
// oftast vill koda sina mallar separat, så bör inte koden finnas
// i denna page-controler
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Exempel 2 - enkla databasexempel - PHP bok</title>
    <style type="text/css">
      /* Demo - annars hade CSS-koden legat i en extern CSS-mall */
      body {
        margin: 100px;
        font-family: Verdana, sans-serif;
      }
      table, th, td {
        border: 2px solid #777;
        border-collapse: collapse;
      }
      th,td {
        padding: 3px 10px;
      }
      .adminstatus {
        text-align: right;
      }
    </style>
  </head>
  <body>
```

```
<h1>Användardata</h1>
<table>
  <tr>
    <th>Användarnamn</th>
    <th class="adminstatus">Adminstatus</th>
  </tr>
<?php foreach ( $userdata as $row ): ?>
  <tr>
    <td><?php echo $row['username']; ?></td>
    <td class="adminstatus"><?php echo $row['auth_level']; ?></td>
  </tr>
<?php endforeach; ?>
</table>
</body>
</html>
</code>
```

Exempel 3: Mysql-gränssnittet

Detta exempel skapar precis samma resultat som det föregående, men det gamla MySQL

```
<code>
<?php
/**
 * bokdemo/enkla-databasexempel/exempel-3.php
 *
 * Koppla up mot mysql-databas, utför en enkel fråga och lämna resultatet till
en mall
 *
 * @author      Lars Gunther <gunther@keryx.se>
 * @copyright   Lars Gunther <gunther@keryx.se>
 * @license     Creative Commons Attribution-Noncommercial-Share Alike 3.0
 *             {@link http://creativecommons.org/licenses/by-nc-sa/3.0/}
 * @package    bokdemo
 * @subpackage databas
 * @filesource
 *
 * @todo       Detta skript är ett demo-skript (D) och det har allvarliga
brister (V)
 */

$dbuser = 'bokdemo_php';
$dbpass = 'hard_to_guess';

// Mysql-gränssnittet är helt procedurellt, men har resursvariabler

// Att koppla upp och att välja databas är två skilda steg
$dbconn = mysql_connect('localhost', $dbuser, $dbpass);
if ( !$dbconn ) {
    // Varning! Man bör inte skriva ut sina felmeddelanden på detta vis.
    // Här görs det endast som demo av funktionaliteten.
    header('HTTP/1.1 500 Internal Server Error');
    echo 'Databasuppkoppling misslyckades. ' . mysql_error();
    exit;
}
// Nästa steg utförs med en syntax som är mycket vanlig, men som lämnar
```

```
// i princip inget utrymme för vettig felhantering.
// Här sänds bara ett felmeddelande, men ingen statuskod.
// Medtaget som ett varnande exempel
$db = mysql_select_db('bokdemo')
    or die('Kunde inte jobba mot vald databas. ' . mysql_error());

$sql = 'SELECT username,auth_level FROM users';
 svar = mysql_query($sql);

if ( false === $svar ) {
    // Kommer vi hit har frågan inte gått att utföra
    // Varning! Man bör inte skriva ut sina felmeddelanden på detta vis.
    // Testfråga som skapar fel: 'ZSELECT * FROM users'
    header('HTTP/1.1 500 Internal Server Error');
    echo mysql_error();
    exit;
}

// Frågan kunde utföras, men den har kanske inte returnerat något innehåll
// Testfråga som lämnar tomt innehåll: 'SELECT * FROM users WHERE 0'
if ( mysql_num_rows($svar) == 0 ) {
    // Ett uteblivet resultat kan tänkas föranleda att man
    // skickar HTTP-statuskoden 404 (Not Found)
    // Varning! Detta är bara "proof of concept", felmeddelanden till användaren
    // bör vara (1) mindre tekniska och (2) mer användbara!
    header('HTTP/1.1 404 Not Found');
    echo '<h1>Databasfrågan returnerade ett tomt svar</h1>';
    exit;
}

// Varning! Följande kod är tänkt att illustrera en mall och eftersom man
// oftast vill koda sina mallar separat, så bör inte koden finnas
// i denna page-controller.
// Denna mall är mycket lik det föregående exemplet.
// Ändrade rader är fetstilta.
// Lägg märke till att vi tvingas använda ett kommando som egentligen inte hör
// hemma i
// mallen, nämligen mysql_fetch_assoc(). Det kommandot hör hemma i "modellen"
// (MVC).
// Om man i ett professionellt projekt tvingas jobba med med mysql-gränssnittet,
// så
// bör man bygga ett eget mer avancerat gränssnitt runt det.
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Exempel 3 - enkla databasexempel - PHP bok</title>
    <style type="text/css">
      /* Demo - annars hade CSS-koden legat i en extern CSS-mall */
      body {
        margin: 100px;
        font-family: Verdana, sans-serif;
      }
      table, th, td {
        border: 2px solid #777;
        border-collapse: collapse;
      }
      th,td {
```

```
        padding: 3px 10px;
    }
    .adminstatus {
        text-align: right;
    }
</style>
</head>
<body>
    <h1>Användardata (ex 3)</h1>
    <table>
        <tr>
            <th>Användarnamn</th>
            <th class="adminstatus">Adminstatus</th>
        </tr>
<?php while ( $row = mysql_fetch_assoc($svar) ): ?>
        <tr>
            <td><?php echo $row['username']; ?></td>
            <td class="adminstatus"><?php echo $row['auth_level']; ?></td>
        </tr>
<?php endwhile; ?>
    </table>
</body>
</html>

</code>
```

Exempel 4: Anslutningsspecifika inställningar

Ofta vill man göra en serie inställningar för sin databas, i samband med att man kopplar upp. Speciellt om man själv inte administrerar serverna, så hjälper detta till att skapa portabel kod. Om man däremot har full kontroll över sin DBMS och sin webserver, så görs dessa inställningar bättre direkt på just servern.

För att dessa inställningar skall fungera så måste man ha MySQL version 4.1 eller senare.

De förslag på inställningar jag här lämnar är lämpliga för en utvecklingsmiljö, då de ställer högre krav på att applikationen fungerar perfekt. MySQL kan nämligen vara ”förlåtande” mot felaktig indata, vilket låter bra men ofta leder till obehagliga överraskningar.

I och med detta exempel har vi en rimligt fungerande uppkoppling. För enklare projekt skulle man kunna lägga detta i en egen fil, som sedan inkluderas av alla kontroll-skript som behöver en databasuppkoppling – förutsatt att vi fixar felhanteringen!

```
<code>
<?php
/**
 * bokdemo/enkla-databasexempel/exempel-4.php
 *
 * Koppla up mot databas och gör diverse inställningar
 *
 * @author     Lars Gunther <gunther@keryx.se>
 * @copyright  Lars Gunther <gunther@keryx.se>
 * @license    Creative Commons Attribution-Noncommercial-Share Alike 3.0
```

```
*          {@link http://creativecommons.org/licenses/by-nc-sa/3.0/}
* @package bokdemo
* @subpackage databas
* @filesource
*
* @version 0.02 alpha
* @todo    Detta skript är på pre-alpha-nivå, men det har ännu
*          allvarliga brister (V)
*/

// Kursiverade rader är identiska med exempel 1
$dbuser = 'bokdemo_php';
$dbpass = 'hard_to_guess';
$dsn    = 'mysql:dbname=bokdemo;host=localhost';

try {
    $db = new PDO($dsn,$dbuser,$dbpass);
}
catch (PDOException $e) {
    echo '<h1>Uppkopplingen till databas misslyckades</h1>'.PHP_EOL;
    echo '<pre>'; // Ifall meddelandet formatterats över flera rader
    echo $e->getMessage();
    echo '</pre>';
    exit;
}

// Nytt börjar här

// Gör så att PDO alltid kastar PDOExceptions om regelrätta fel uppstår
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// Eftersom exceptions kastas från och med nu vid fel så använder vi try-catch

try {
    // Gör så att PDO använder emulerade prepared statements för MySQL
    // Se adressen http://netevil.org/blog/2006/apr/using-pdo-mysql
    // för en förklaring.
    // Ändra detta om du (1) använder icke-ascii kompatibelt teckensnitt
    // eller (2) använder MySQL 5.1.17 eller senare.
    $db->setAttribute(PDO::ATTR_EMULATE_PREPARES, true);

    // För att jag personligen aldrig använder sifferindexerade svar från
    // databasens tabeller och därför vill minska ner vad PDO::fetch()
    // och PDO::fetchAll() returnerar till endast den associativa arrayen.
    // Detta kräver PHP 5.2.0 eller senare.
    $db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);

    // Gör så att PDO som standard använder "buffrade" frågor ihop med MySQL
    // Detta krävs om man skall använda två samtidiga frågor i sin kod
    $db->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, true);

    // Ställ in så att MySQL-förbindelsen går på svensk tidszon
    // Detta gör koden portabel om man har sin server i en annan tidszon
    // Det kräver också att man fyllt MySQL:s tidszonstabeller med information
    // http://dev.mysql.com/doc/refman/5.0/en/time-zone-support.html
    $sql = "SET time_zone = 'Europe/Stockholm'";
    $db->query($sql);
}
```

```
// Ställ in så att MySQL blir mindre feltolerant och mer ANSI-lik
$sql = "SET SESSION sql_mode =
      'STRICT_ALL_TABLES,NO_ZERO_DATE,NO_ZERO_IN_DATE'";
$db->query($sql);
}
catch (PDOException $e) {
    // Varning! Dålig felhantering.
    header('HTTP/1.1 500 Internal Server Error');
    echo '<h1>Ett fel uppstod i databasinställningarna.</h1>'.PHP_EOL;
    echo $e->getMessage();
}
?>
<!DOCTYPE html>
<title>Exempel 4 - enkla databasexempel - PHP bok</title>
<h1>Alla inställningar gjorda!</h1>
<p>Ser du detta kunde alla inställningar göras utan problem.</p>
</code>
```

Exempel 5: Förkortad version av föregående exempel

Föregående exempel innabar att vi började närma oss användbar kod, men för att vara pedagogisk var det onödigt långt. Här kommer exakt samma funktionalitet, fast i nedkortad form.

```
<code>
<?php
/**
 * bokdemo/enkla-databasexempel/exempel-5.php
 *
 * Koppla up mot databas och gör diverse inställningar, förkortad version, men
 med större
 * flexibilitet. Resultatet av uppkopplingen är identiskt med föregående
 exempel.
 *
 * @author      Lars Gunther <gunther@keryx.se>
 * @copyright   Lars Gunther <gunther@keryx.se>
 * @license     Creative Commons Attribution-Noncommercial-Share Alike 3.0
 *             {@link http://creativecommons.org/licenses/by-nc-sa/3.0/}
 * @package     bokdemo
 * @subpackage  databas
 * @filesource
 *
 * @todo       Detta skript är på alpha-nivå, men det har ännu allvarliga
 brister (V)
 * @todo       Den avslutande HTML-koden är endast till för att visa att
 skriptet fungerar
 *             och skall tas bort innan det används "på riktigt"
 * @todo       Att ha kontonamn och lösenord på detta sätt i skriptet kan vara
 en säkerhetsrisk. beroende på
 *             miljön i övrigt. Och ihop med filesource-direktivet kommer
 inställningarna synas i dokumentationen!
 * @todo       Alla variabler som styr förbindelsen är globala. De bör flyttas
 bort från den globala namnrymden.
 */
```

```
$dbuser = 'bokdemo_php';
$dbpass = 'hard_to_guess';
$dsn = 'mysql:dbname=bokdemo;host=localhost';
$options = array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                 PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
                 PDO::ATTR_EMULATE_PREPARES => true,
                 PDO::MYSQL_ATTR_USE_BUFFERED_QUERY => true);
$conn_tz = 'Europe/Stockholm';
$conn_sql_mode = 'STRICT_ALL_TABLES,NO_ZERO_DATE,NO_ZERO_IN_DATE';

try {
    $db = new PDO($dsn,$dbuser,$dbpass,$options);
    if ( isset($conn_tz) ) {
        $sql = "SET time_zone = '{$conn_tz}'";
        $db->query($sql);
    }
    if ( isset($conn_sql_mode) ) {
        $sql = "SET SESSION sql_mode = '{$conn_sql_mode}'";
        $db->query($sql);
    }
}
catch (PDOException $e) {
    // Varning! Fortfarande dålig felhantering.
    header('HTTP/1.1 500 Internal Server Error');
    echo '<h1>Ett fel uppstod i databasuppkopplingen.</h1>'.PHP_EOL;
    echo $e->getMessage();
}
// Egentligen slut här, nedanstående rader bör tas bort efter att man testat
demon.
?>
<!DOCTYPE html>
<title>Exempel 5 - enkla databasexempel - PHP bok</title>
<h1>Alla inställningar gjorda!</h1>
<p>Ser du detta kunde alla inställningar göras utan problem också med nedkortad
kod.</p>
</code>
```

Exempel 6: SQL-fråga baserad på användardata och prepared statements

Det är mycket vanligt att de SQL-frågor man skickar till sin databas är baserade på indata från webbplatsens användare. Detta medför en potentiell risk för attacker i form av s.k. SQL-injektion. Följande exempel är tänkt att visa användardata om en enda användare i databasen, och använder *förberedda uttryck* ("prepared statements") för att gardera mot detta. Det finns inget känt sätt att smugla SQL-injektion förbi ett (icke-emulerat) förberett uttryck, vilket gör denna metod till den allra bästa för rena nybörjare.

OBS! Just nu används dock emulerade förberedda uttryck i min kod, vilket ger 99 % skydd mot SQL-injektion. För PHP 5.2.1 och framåt är det också standardinställningen. Det är endast om man har en icke-ASCII kompatibel teckenkodning som problem kan uppstå. *Till dess att boken släpps*, så kommer detta (troligen) vara åtgärdat, eftersom MySQL 5.1 stödjer "query cache", så behövs inte längre emuleringen. Förutsatt att man använder just MySQL 5.1.17 eller senare.

Under testerna för att skriva denna bok, så fann jag dock att PHP på Windows har en bugg som gör att också med äkta, icke-emulerade, förberedda uttryck, så kan SQL-injektion ske.⁹ Ett ”försvar på djupet”, med noggrann indatafiltrering är naturligtvis alltid varmt rekommenderat.

Jag har tagit bort de sista raderna i föregående exempel och återanvänder den koden för min uppkoppling. Den ändrade filen har jag sparat under namnet ”pdo-cx-alpha-1.php” och den ligger i katalogen ”bokdemolib/model”.

<code>

<?php

```
/**
 * bokdemo/enkla-databasexempel/exempel-6.php
 *
 * Utgå från URL:ens GET-variabler för att skapa en fråga och visa data.
 *
 * @author    Lars Gunther <gunther@keryx.se>
 * @copyright Lars Gunther <gunther@keryx.se>
 * @license   Creative Commons Attribution-Noncommercial-Share Alike 3.0
 *           {@link http://creativecommons.org/licenses/by-nc-sa/3.0/}
 * @package   bokdemo
 * @subpackage databas
 * @filesource
 *
 * @version   Demo
 *
 * @todo      Ordentlig felhantering saknas fortfarande.
 */

// Webbroten är den katalog som är publikt anropbar,
// våra bibliotek ligger parallellt med densamma
require_once dirname($_SERVER['DOCUMENT_ROOT']) . DIRECTORY_SEPARATOR .
    'bokdemolib' . DIRECTORY_SEPARATOR . 'model' .
    DIRECTORY_SEPARATOR . 'pdo-cx-alpha-1.php';

// Har vi kommit hit så finns det ett PDO-objekt som heter $db

// Kontrollera om användardata finns
if ( isset($_GET['username']) && strlen($_GET['username']) ) {
    // Denna variabel används för att slippa återupprepa villkoret ovan
    $username_chosen = true;
    // I denna demo hoppas indatafiltrering över, vi är ändå säkra
    // I vanliga fall kan det finnas andra skäl än rena säkerhetsskäl
    // att filtrera sin indata, såom ökad användbarhet, etc.
    try {
        $stmt = $db->prepare('SELECT username,auth_level FROM users WHERE
username = :username');
        // I händelse av att vår URL skulle innehålla fler GET-variabler än vad
vi önskat
        // så flyttar vi användarnamnet till en ny array
        $params = array( 'username' => $_GET['username'] );
        $stmt->execute($params);
        $row = $stmt->fetch(); // Hämta en rad som en associativ array är *MIN*
```

⁹ <http://www.phpportalen.net/viewtopic.php?t=83977>

```
standard!
}
    catch (PDOException $e) {
        // Varning! Fortfarande dålig felhantering.
        header('HTTP/1.1 500 Internal Server Error');
        echo '<h1>Ett fel uppstod i databasuppkopplingen.</h1>'.PHP_EOL;
        echo $e->getMessage();
    }
} else {
    $username_chosen = false;
}

// Emulerad mall följer
?>
<!DOCTYPE html>
<html>
    <head>
        <title>Exempel 6 - enkla databasexempel - PHP bok</title>
        <style type="text/css">
            /* Demo - annars hade CSS-koden legat i en extern CSS-mall */
            body {
                margin: 100px;
                font-family: Verdana, sans-serif;
            }
            dt, dd {
                margin-top: 1em;
            }
        </style>
    </head>
    <body>
        <h1>Användardata (ex 6)</h1>
        <?php if ( !$username_chosen ): // För tekniskt felmeddelande för verkligt bruk
        ?>
            <p>Du måste ange ett användarnamn som GET-variabel!</p>
        <?php elseif ( empty($row) ): ?>
            <p>Det valda användarnamnet matchar ingen post i databasen.</p>
        <?php else: ?>
            <dl>
                <dt>Användarnamn</dt>
                <dd><?php echo htmlentities($row['username'], ENT_QUOTES, 'UTF-8'); ?></dd>
                <dt>Administrativ behörighet</dt>
                <dd><?php echo $row['auth_level']; ?></dd>
            </dl>
        <?php endif; ?>
        <h2>Testlänkar</h2>
        <ul>
            <li><a href="exempel-6.php?username=foo">Foo (vanlig användare)</a></li>
            <li><a href="exempel-6.php?username=admin">Admin</a></li>
            <li><a href="exempel-6.php?username=bad-name">Användarnamnet finns
inte</a></li>
            <li><a href="exempel-6.php?username=">Inget namn</a></li>
            <li><a href="exempel-6.php">Ingen variabel</a></li>
        </ul>
    </body>
</html>

</code>
```

Appendix: SQL-satser för att skapa den databas som krävs för att kunna köra kodexemplen

<code>

```
CREATE TABLE `users` (  
  `username` varchar(10) character set utf8 collate utf8_swedish_ci NOT NULL,  
  `password` varchar(40) character set utf8 collate utf8_swedish_ci NOT NULL,  
  `auth_level` tinyint(4) NOT NULL default '1',  
  PRIMARY KEY (`username`)  
) ENGINE=InnoDB;
```

```
INSERT INTO `users` (`username`, `password`, `auth_level`) VALUES  
( 'admin', 'd033e22ae348aeb5660fc2140aec35850c4da997', 3),  
( 'bar', '60518c1c11dc0452be71a7118a43ab68e3451b82', 1),  
( 'foo', '8843d7f92416211de9ebb963ff4ce28125932878', 1);
```

</code>

Övning

Laborera med exemplens SQL-satser och uppkopplingsinställningar och se vad som händer om något blir fel.

Ingen fördjupning till detta kapitel. Ämnet fortsätter att behandlas i de följande kapitlen.

31 Grundläggande databasteori

Vad är en databas?

Vad är en relationsdatabas?

Vad är en DBMS?

Exempel på DBMS:

- SQLite
- MySQL
- PostgreSQL
- DB2
- Oracle
- Access (Jet)
- MS SQL

Tabeller

Post – rad

Fält - kolumn

Datatyper

PHP – MySQL – SQLite – Oracle/DB2/MSSQL/Access/Postgresql/Firebird

<http://dev.mysql.com/doc/refman/5.0/en/data-types.html>

Primärnycklar

Konvention: Kalla ett fält för ”tabellnamn_ID”

Andra index

Unika

Tips! Kan omspanna flera kolumner

Fulltext

Relationer

Tips! Referenintegritet möjlig med InnoDB och Falcon

Klient-server

PHP är klienten.

Vad är SQL?

ANSI + tillägg

Tips! "LIMIT" är inte ANSI, men ack så praltiskt

Transaktioner

Tips! Med InnoDB och Falcon

Övning

Fördjupning

Tompas bok och webbplats

Andra böcker

32 MySQL

Installation

Se appendix (säkra, tidszon, konton, databaser, tabeller)

Kräv *minst version 4.1* – byt annars webbhotell!

Grundläggande säkerhet

Tre konton:

- Root
- Priviligierad PHP-användare
- Vanlig PHP-användare

Inställningar i MySQL

Tips: Glömt rootlösenordet?

1. Gå till körläge ett så att ingen annan kommer åt din DB under tiden
2. Starta utan priv-tabellerna
3. Skriv in nytt lösenord
4. Stäng demonen
5. Gå till körläge 3 eller 5 och starta ev. Mysqld om det inte gjorts automatiskt

PHP:s tre inbyggda moduler för MySQL

- mysql
- mysqli
- PDO

PHPMysqlAdmin

Kan (ännu) inte använda PDO, men väl mysqli

Autentiseringsmetod

Hjälptabeller

Tips: Symlänka till den version du använder.

Inställningar i MySQL

Server default

Per-connection – utnyttja dessa till max så skapar du portabla skript!

Tips! Gör servern så *intolerant* mot fel som möjligt!

Tabelltyper

- MyISAM
- InnoDB
- HEAP
- Falcon

Övriga som fördjupning

Övningar

Skapa databas

Skapa användare med rätt privilegier.

Fördjupning

Man kan replikera mellan flera maskiner.

33 Skapa tabeller med PHPMyAdmin

Planera

Systemutveckling

ER-diagram

Rimlig tillämpning av normalformlerna.

Med PHPMyAdmin

Lägg märke till att den visar CTREATE TABLE frågan

Tips! Du kan exportera din DB från PHPMyAdmin

Tabelltyp

Transaktioner?

Referensintegritet?

Fulltext-index?

auto_increment

Versaler/gemener

MySQL bryr sig om detta IBLAND, och ibland inte. Varierar med version, inställningar och operativsystem.

Var konsekvent!

Jfr. Oracle returnerar alltid kolumnnamnen som versaler. Så inte MySQL.

Aggressiv användning av index på webbapplikationer

Ett index används per fråga!

Nybörjartabbe: Att inte tro att primärnycklar är index.

Ändra i efterhand

Multikolumn index är svåra att göra direkt.

ALTER TABLE

PHPMyAdmin kan det också!

Övningar

Skapa tabeller enligt spec.

Fördjupning

Manualen till MySQL

Webbplatser

Böcker

Certifiering

- Utvecklare
- Drift

Andra verktyg

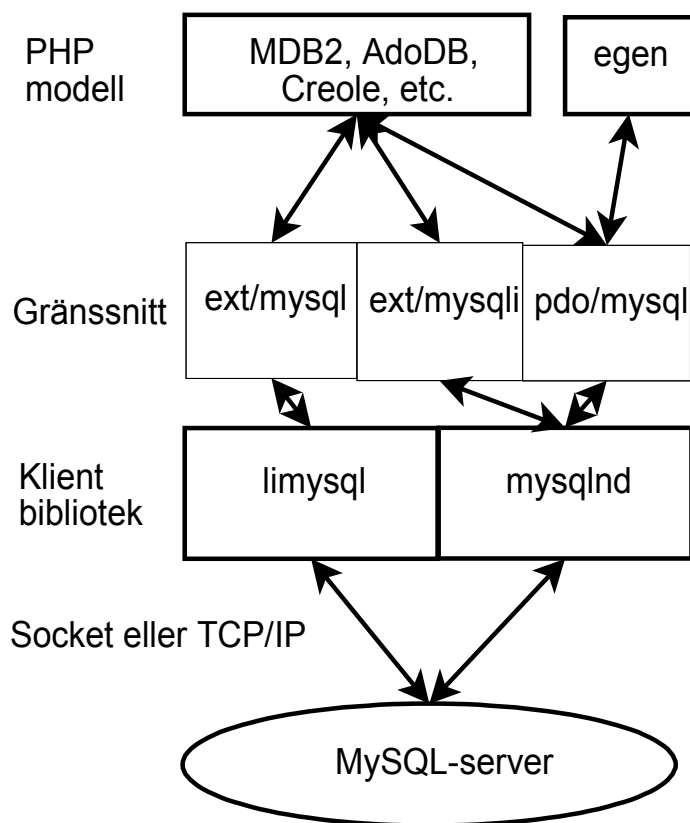
CLI

MySQL CC

DBDesigner

34 Kommunikation mellan PHP och MySQL

Schematisk förklaring



När din PHP-applikation kommunicerar med en MySQL-databas så går kommunikationen genom fyra lager.

1. MySQL-servern är åtkomlig genom en socket-fil, eller via TCP/IP. Detta är nätverkstrafiken, som kan bli en flaskhals på tungt belastade webbplatser. Dessutom kan själva databasservern bli en flaskhals, när PHP-skriptet tvingas göra halt och bara vänta på svar från servern. Möjligheten att kunna **cache** resultat på klienten är ett sätt att råda bot på dessa problem.
2. Den kod, skriven i C, som har det direkta ansvaret för kommunikationen på klient-sidan, har länge varit **libmysql**, men numera finns också **mysqlnd** som alternativ. **mysqlnd** utläses ”MySQL Native Driver” och är alltså en drivrutin specialskriven för att fungera optimalt med just PHP, medan **libmysql** gjordes för att passa med många språk. **mysqlnd** innehåller ett flertal förbättringar som gäller prestanda och utvidgad funktionalitet.¹⁰
3. PHP som språk tillhandahåller tre gränssnitt för att kommunicera med MySQL. Dessa presenteras mer i detalj i egna stycken nedan.

¹⁰ <http://www.derickrethans.nl/files/andrey-mysqlnd.pdf>

<http://blog.ulf-wendel.de/?p=149> ”PHP: What is mysqlnd, do I need it?”

4. Den del av din applikation som specifikt ansvarar för kommunikationen med databasen är, som tidigare sagts, ”modellen”. Det finns ett antal färdigskrivna klasser som du kan återanvända, eller så skriver man sin egen kod, förslagsvis mot PDO-gränssnittet.

Det klassiska mysql-gränssnittet

Ännu skrivs det massor av kod som använder detta gränssnitt. Det beror på ett flertal faktorer, såsom:

- Man har en gammal applikation, vars historia sträcker sig långt tillbaka i tiden. Det finns mängder av sådana applikationer igång, just för att de blivit populära och spelar en kritisk roll för en organisation.
- Man har ett behov av att kunna köra sin applikation på servrar som endast stödjer PHP version 4. Ännu fem år efter att PHP version 5 lanserats så erbjuder de flesta webbhotellen PHP 4 endast, eftersom man menar att kundernas befintliga applikationer annars slutar fungera.

För att kunna bygga en modern seriös webbapplikation, så behöver man bygga ett antal hjälpklasser runt detta gränssnitt för att få till stånd logikseparation, återanvändningsbarhet och prestanda. Den här boken berör därför inte detta gränssnitt. Skulle du tvingas ta över en gammal applikation som bygger på detsamma så finns det en mängd böcker, artiklar och diskussionsforum till ditt förfogande.

Det kraftfulla mysqli-gränssnittet

I och med version 5.0 av PHP så introducerades ett förbättrat gränssnitt: mysqli – där i:et står för improved. Funktionaliteten utvidgades med bl.a. prepared statements och möjlighet infördes att använda gränssnittet objektorienterat.

I den händelse att man vill utnyttja varenda uns prestanda och funktionalitet i sin MySQL-databas, så är detta gränssnitt att föredra. Annars är det inte heller nödvändigt längre!

PDO

Version 5.1 av PHP gav oss detta gränssnitt, med ett i mitt tycke bättre API än mysqli. PDO har all funktionalitet som behövs för de allra flesta webbapplikationerna, och är det gränssnitt som i framtiden utvecklarna av PHP kommer bygga vidare på. Det är kort och gott standardalternativet för modern webbutveckling med PHP.

PHP är inte specifikt för MySQL, utan kan användas för de flesta DBMS. Somliga tror att det inte är någon fördel, för ”min kod skall ju ändå aldrig köras på någon annan databas”. Själv tycker jag att det finns en fördel att göra **utvecklaren** lätt flyttbar, även om den aktuella kod han/hon skriver inte behöver vara det. Om jag i framtiden skriver ett nytt projekt, så är det mindre att lära om och mindre att slå upp i manualen! Din kod kanske inte skall vara flyttbar – men du som programmerar bör nog vara det!

Jämfört med det äldsta mysql-gränssnittet är dock den viktigaste fördelen att PDO erbjuder en lätthet att skapa en fungerande struktur för hela sin applikation. Det finns ett antal färdiga metoder som tar hänsyn till det tänkande som man idag bör ha.

Viktiga metoder i PDO

Använd alltid prepared statements!

- Skydd mot SQL-injection
- (Ofta) bättre prestanda

John Coggeshall:

Those who do not use Prepared statements should be flogged with a rubber hose

They are

- *Faster*
- *Easier to maintain*
- *Considerably more secure*

ALL database write operations should be done through prepared statements, Period. ¹¹

FETCH_CLASS

Scenario: Du har tagit över en befintlig databas (=opålitlig data). När data hämtas skall den kontrolleras.

Genom att bygga in kontrollen i klassen så kan den användas också när datakällan är exempelvis \$_POST eller när den är från en webbtjänst.

Mysqlnd

Förbättrad prestanda, speciellt vad gäller ”persistent connections”

Utvidgad funktionalitet

Klientsidescachning

Bättre licensmodell

Övning

(Magic_quotes är off!)

Hämta data och visa i tabell: ”Allsvenskan” fast nu från databas.

¹¹ http://blog.coggeshall.org/exit.php?url=aHR0cDovL3ByaXZhdGUuY29nZ2VzaGFsbC5vcmcvdG9wMzBfemVuZC56aXA=&entry_id=214 (PPT fil inuti ZIP)

Fördjupning

Databasabstraktion

Call-level abstraction

Erbjuds av PDO

”Data *access* abstraktion”

Data-type abstraction

Inte i PDO, men stöd för *sekvenser*

SQL-abstraction

Inte i PDO

SQL-skapande PHP-kod

Data-persistens lager. När objekten hämtas/lagras så behöver de inte veta hur det går till. (De kan rentav lagras på annat sätt än i databas.)

Gränssnitt skrivna i PHP

PEAR DB

På väg ut.

MDB2

Ny standard i PEAR sedan 2006

Liknar

AdoDB

Xx

Andra

- Xx

Ditt eget?

Troligen onödigt...

35 Mer om SQL

SELECT

Syntax

Tips! Använd aldrig *

WHERE

Också för andra frågor än select

ORDER BY

Har du rätt kollationering?

LIMIT

Syntax

Aggregatfunktioner och group by

Bör användas maximalt

Datumkonvertering i MySQL

Till/från Unix-tid

Tidsintervall

OBS! DATE_FORMAT är ofta använt vid SELECT-frågor, trots att datumformat inte hör hemma i modellen, utan i ”vyn” (presentationslogiken).

INSERT

UPDATE

DELETE

Glöm aldrig where

Tips! Hängslen och livrem. Använd gärna LIMIT här också!

Övning

Skriva SQL-frågor

Pröva dem i PHPMyAdmin

Fördjupning

Hur en fråga utförs i MySQL

Flödesschema – se http://hades.phparch.com/files/tek07/jay_pipes-kill_mysql_performance.pdf

1. Optimering (prepare)
 - a) I *teorin* skall alla *sedan* gå lika snabbt. Två förbehåll!
 - b) Val av index
2. Exekvering

Having

Limit jämfört med andra DBMS!

36 Avancerad SQL

GROUP_CONCAT

Suveränt för web 2.0 taggar!

```
SELECT a.articleID, a.title, A.text, GROUP_CONCAT(t.tag) AS tags
FROM articles AS a
INNER JOIN article_tags AS t USING (articleID)
WHERE a.articleID = :articleID
```

Eller för att lista flera medförfattare till en artikel

```
SELECT a.articleID, a.title, a.text,
       GROUP_CONCAT(CONCAT(u.first_name,',u.last_name)) AS full_names
       GROUP_CONCAT(u.username) AS writer_ids
FROM articles AS a
NATURAL JOIN authorship
NATURAL JOIN users AS u
WHERE a.articleID = :articleID
```

Med denna fråga slipper vi flera databasuppslag och kan i PHP-land göra om till länkar.

```
$writer_fullnames_arr = explode(',', $row['full_names']);
$writer_names_arr     = explode(',', $row['userid']);
$writer_link_arr      = array(); // initialisera (God practice!)
foreach ( $writer_names_arr as $key => $userid ) {
    $writer_link_arr[] = '<a
href="/author/'. $writer_id. '">'. $writer_names_arr[$key]. '</a>';
}
// Om det är fler än en författare skall de sista separeras med ordet "och"
if ( count($writer_link_arr) > 1 ) {
    $last = array_pop($writer_link_arr);
    $writers = implode(',', $writer_link_arr);
    $writers .= ' och ' . $last;
}
echo $writers;
```

JOIN

Natural join

Inner Join

ON

USING

Left (right) join

Utan villkor

Den kartesiska (stavning?) produkten

JOIN och patterns

Många frameworks (Rails!) förutsätter en tabell per objekt i sin grundkonfiguration.

Transaktioner

ACID – teorin

I SQL

I PDO

Stored procedures

Triggers

Övningar

Skriva SQL-frågor

Testa dem i PHPMyAdmin

Fördjupning

Join utan "join"

D.v.s. bara med where villkor

"Sounds like"

Tolerera stavfel, olka stavningar (Carlson, Carlsson, Karlson, Karlsson), etc

REPLACE INTO och ON DUPLICATE KEY

Ej standard, men bekvämt

SQL-fu!

<http://jpipes.com/index.php?/archives/177-Common-Questions-and-Answers-from-Performance-Tuning-Webinars.html>

37 Databasfel och oväntade databassvar

Vanliga misstag:

- Visa inte användarna dina fel!
 - Låt användaren få ett **hjälsamt** felmeddelande!
 - Felmeddelanden är mumma för hackers som vill göra intrång
- Visa inte sökmotorer dina felkoder!

Grundläggande regler:

- 404 "Not found" fel om URL inte mappas mot resurs.
- 301 "Moved permanently" om URL:en kan tolkas, men inte är optimal
- 500 "Internal Server Error" om databaskommunikationen inte fungerat
- 304 "Not modified" kan också vara ett alternativ som förbättrar användbarheten och avlastar servern.
- 503 "Service Unavailable" kan vara ett alternativ till 500, om felet beror på överbelastning eller om data skall hämtas från tredje part och den förbindelsen är bruten.

Applikationsfel

Visa det under utveckling – eller om systemadministratör är inloggad

HTTP/1.1 500 Internal Server Error

Hjälsamt felmeddelande

Det är inte dina användares jobb att meddela dig om att fel uppstår!

Logga felet och skicka meddelande till teknikansvarig!

Tomma resultat

Ta inte för givet att du får ett svar!

Överväg HTTP/1.1 404 Not Found

Dubbla POST från användaren

Ex. Uppdatering av webbläsaren.

Transaktioner

Om flera SQL-satser skall göras i följd och de är beroende av varandra.

Fördjupning

Låsning av tabeller

ACID

Tabellnivå

Radnivå

Cellnivå (ej i MySQL)

38 Avancerade PDO-tricks

Skydda dina uppkopplingsuppgifter

Ilia A: The DSN string can be an INI setting and you can “name” as many DSNs are you like.

```
ini_set("pdo.dsn.ilia", "sqlite::memory");
```

```
try {
```

```
$db = new PDO("ilia");
```

```
} catch (PDOException $e) {
```

```
echo $e->getMessage();
```

```
}
```

http://ilia.ws/files/quebec_PDO.pdf

Fetch into (existing object)

Scrollable cursor

Lazy fetching

Fetch into callback

39 Formulär – klientsidan

Element

CSS

Javaskript-kontroller

AJAX (är ett användarnamn upptaget?)

Fallgropar

- ”GET” för operationer som utför något
- Maskerad submit som länk
 - Googlebot kan radera din databas!
- POST för vanlig navigation
 - Följs däremot inte av sökrobotar
- Javaskript för att skicka
 - OK, om du har en ”fallback”

Web Forms 2.0

<http://www.whatwg.org/specs/web-forms/current-work/>

Princip: Bakåtkompatibilitet, ”graceful degradation”/feltolerans

- Nya data-typer funkar i gamla webbläsare
- Befintlig datamodell

Nya typer, snabbexempel

- Datum
- Mail
- URL
- Range

- Datalist
- Pattern
- Templates för expanderande formulär

Allt detta kan göras med JS idag, men detta är smidigare.

40 Formulär – serversidan

Ej kompletta exempel

Total validering kontra AJAX

- Inloggning
- Blogginlägg
- Kommentar till en blogg
 - Gravatar
- Kontaktformulär
- Användardata (ex. Kunduppgifter)

Samma konfigurationsfil till PHP och JS

Automatiserade kontroller utifrån parametrar

- Krävs, ja/nej
- Typ
 - Sträng
 - Datum
 - Nummer
 - Etc
- Max-/min- längd/värde
- Etc

41 Skicka mejl med PHP

Grundfunktionen

Fara: Header injection

array_filter

smtp_header_injection_check

HTML-mail

Varför?

Stora problem skapa standardbaserad kod.

XSS

Färdiga klasser

PEAR

Swift

Etc.

åäö i ärenderaden? imap_8bit <http://www.webforum.nu/showthread.php?t=159447>
<http://se.php.net/manual/en/function.imap-8bit.php>

Hur blir detta med Unicode?

42 Sessioner och inloggning

Ett enkelt exempel

Formulär – kontroll – ”du är inloggad”

Sida – kontroll huruvida personen är inloggad – till formuläret annars.

Farorna

Sessionskidnappning

Sessionsfixering

Delad värd? Sessionsdata kan bli stulen

Stäng av session via URL

`session_regenerate_id()`

Lagra sessionsdata i DB

Skalbarhetsbonus! Flera maskiner kan dela på samma DB.

Inloggning

Nämn HTTP, men visa via formulär

Behörighetskontroll

Både i kontroll-skript och mallar

43 En kaka, tack!

Http only

44 Objekt, fördjupning

Polymorfism

Interfaces

Abstrakta klasser och metoder

Patterns

45 Filhantering

Streams

Uppladdade filer

46 XML i PHP

47 PHP DOM kontra JS DOM

48 Simple XML kontra E4X

49 "Maintainable code"

Konventioner

Dokumentation

Tråkigt!

PHPDocumentor

Doxygen

Namnrymder

Det gamla sättet i PHP

Det nya sättet

Det gamla sättet i javaskript

- direktexkeverande "closures"
- Yahoo.util...
- "with" är däremot en dålig ide.

Det nya sättet i JS

- Let-block
- Bibliotek
- Namnrymder "per se"

50 Prestanda och skalbarhet

Varning för ”premature optimization”

Snabbt vz långsamt i språket

- Återskapa inte inbyggda funktioner
- Undvik rekursiva funktioner
 - Kombination: db_escape....
- Enkel- och dubbelfnuttar
- Etc

Lazy loading

Databasoptimering

Opcode-cache

<http://www.slideshare.net/oscon2007/os-gopal/>

<http://t3.dotgnu.info/blog/php/>

Cache

- Statiska kopia?
- Client-side
- HTML-cache
- DB-cache
- Objekt-cache

”Profiling”

- Hemmabyggd
 - Tidtagning
 - `memory_get_peak_usage()`
- Professionella

Server-side proxy

Kluster

- Sessioner
- Databas

51 PHP som en del av webbplatsens utveckling

Separerade roller underlättar omdesign, förbättring, utvidgningar, etc.

52 Ett fullfjädrat exempel

En blogg

URL-alternativ

- Enstaka inlägg, med kommentarer, via
 - inlaggsid
 - snygg url
- Flera inlägg via
 - Taggar
 - Datumintervall
 - År
 - År-månad
 - År-månad-dag
 - År-vecka? (Just for fun)
 - Datum1 [- datum 2]
 - ”Senaste”
 - Typ
 - ”vanligt” (POSH)
 - Händelse
 - Recension
 - Författare
 - Populäraste (hur mäta?)
 - Sök
 - Hjälp med stavningen (dyslektiker)

Sidtyper

- Startside
- Enstaka inlägg, med *trådade* kommentarer, ”fler i samma ämne”, etc.
- Lista av inlägg
- Några med ingress
- Resten bara med rubrik
- ”Om oss”

- Kontakt

Egenskaper

- Mikroformat (vad slags inlägg är det)
- Blogroll
- Flickr-feed
- Gravatarer
- Hindra CSRF och bloggspam

API:er

- Pingback
- Trackback
- OpenID
- In-place editing med AJAX
- RSS
- Ical-feed?
- Etc

Om möjligt: Konfiguration via *mallen!*

1. Kontrollern läser mallen för att ta reda på vilken data som skall läsas in – utöver vad som framgår av URL:en
2. Data hämtas och bearbetas
3. Sidan visas

53 Aldrig fullärd

Avancerade funktioner som nämns som tänkbar vidareutbildning:

Alternativa sätt att använda PHP

CLI

PHP-Gtk+

(Ogg Vorbis!)

Bättre matematik

- Bitwise
- BC Math
- Gnu MP library funktioner

PHP moduler/extensions

Bildbehandling

PDF

JSON

SOAP

COM

SCA (IBM/Caroline Maynard)

SPL

Filsockets/streams

Curl

Lågnivåsockets

SQLite, Oracle, DB2, MSSQL, etc

<http://www.php.net/manual/en/funcref.php>

Mer om patterns

Unit testing

Inga regressionsbuggar när du uppdaterar din kod.

Snabb kontroll om din kod klarar en uppgradering av PHP-versionen.

Versionshantering

Paketering

Bra resurser (i allmänhet)

Böcker

Foo

Webbplatser

Podcasts

IRC

54 Appendix 1: Installation, LAMP

Bokens krav:

- PHP 6.0 eller senare (just nu 5.2.3) med följande extensions
 - mysqlnd
 - Tidy
 - Etc.
- MySQL 6.0 eller senare (just nu 4.1)
 - Bara ett index per fråga i versioner ≤ 4.1
- Apache 2.2 eller senare (mina rewrites följer senaste versionens metod, men med få omställningar kan man använda 1.3 eller IIS)
- PHPMyAdmin
- HTML 5 kunnig webbläsare för somliga exempel
 - Just nu bara Opera

LAMP

Ej komplett installationsguide, bara några tips.

55 Appendix 2: Installation, WAMP, WIMP, WIAP

Ej komplett, bara några tips

<http://www.corephp.co.uk/archives/41-A-Guide-to-running-Apache-2,-PHP-4-PHP-5-on-Windows-XP.html>

<http://www.corephp.co.uk/archives/36-A-Guide-to-using-PHP-5-Extensions-on-Windows.html>

Access via ODBC

56 Appendix 3: Installation, MAMP

Macintosh. Ej komplett, bara några tips

57 Appendix: Kodbibliotek

Kolla miljön

magic_quotes_runtime()

Finns alla moduler

PHP-version

Läs och skrivrättigheter

Ställ in miljön

Init-skriptet i sin helhet

Installationsskript

Fopen till bokens sajt

Hämta

Spara

Svara på frågor...

Skriv inställningar

58 Appendix: Vad är nytt i PHP 6

Unicode

<http://www.php.net/manual/en/ref.unicode.php>

<http://www.php.net/manual/en/ref.i18n.php>

<http://php.net/unicode>

Namespaces

<http://cvs.php.net/viewvc.cgi/php-src/README.namespaces?revision=HEAD>

<http://php100.wordpress.com/2007/08/17/namespaces-faq/>

<http://blog.agoraproduction.com/index.php?archives/47-PHP-Namespaces-Part-1-Basic-usage-gotchas.html>

Opcode-cache (APC) inbyggt från början

"Spring cleaning"

<http://www.php.net/~derick/meeting-notes.html#cleanup-of-functionality>

magic_quotes_gpc finns inte längre

Nytt i PHP 5.1-3

http://cvs.php.net/viewvc.cgi/php-src/NEWS?revision=PHP_5_2

http://cvs.php.net/viewvc.cgi/php-src/NEWS?revision=PHP_5_1

Ej sorterat ännu

Mobile Web Best Practices: <http://www.w3.org/TR/mobile-bp/>

<http://www.css3.info/>

http://en.wikipedia.org/wiki/Category:World_Wide_Web

http://wiki.pear.php.net/index.php/PEAR2_Standards

http://hades.phparch.com/ceres/public/page/index.php/tek_live::slides

<http://developer.mozilla.org/es4/> ECMAScript 4 Wiki

Interaktiv PHP

<http://www.fischerlaender.net/php/phpa-norl>

<http://david.acz.org/phpa/>