Johan Falk
http://johanfalk.net/contact

Sju steg mot svart bälte i att tema Drupal 6

Den här guiden riktar sig till de som har någorlunda färdigheter i både webbdesign och Drupal, och vill lära sig att kombinera de två. Kommentarer, förslag, beröm och kritik är alltid välkommet. Frågor hänvisas till svenska Drupalforumet på drupalsverige.se. (Jag får tyvärr för många frågor för att kunna svara på dem genom min privata e-post, och frågor som ställs i forumet har också en god chans att komma andra till godo.)

Innan vi börjar

Drupal är ett oerhört kraftfullt system för att snabbt förse en webbplats de funktioner man behöver, vilket rätt snart blir tydligt för den som installerar och börjar testa olika moduler. Vad som inte är lika uppenbart är att Drupal dessutom erbjuder enorma möjligheter för att få webbplatser att se ut som man vill.

Att anpassa Drupals utseende kallas för att *tema* Drupal, och de funktioner i Drupal som sköter utseendet kallas Drupal *temalager (theme layer)*. Den här guiden kommer att ge en genomgång av stora delar av hur temalagret fungerar för det som kallas *PHP engine* – en temamotor som hjälper till att hitta och tolka funktioner och filer så att det blir enklare att anpassa teman för Drupal. PHP engine är den överlägset vanligaste temamotorn för Drupal, och är den som följer med i en standardinstallation.

Du behöver inte ha temat Drupal tidigare för att använda den här guiden. Däremot är det bra om du har grundläggande färdigheter i hur man använder Drupal – så som hur du hittar bland administrationssidorna och vet vad användarroller och nodtyper är. Guiden innehåller inte något om generella färdigheter för webbdesign, vare sig det gäller kodning (HTML, CSS, Javascript) eller mer övergripande färdigheter (så som användbarhet eller estetik). Meningen är snarare att någon som har åtminstone ett hum om Drupal och webbdesign ska lära sig att para ihop de två färdigheterna.

Guiden är uppdelad i sju steg, som börjar med det absolut mest grundläggande och avslutar med saker som kanske inte är superkomplicerade, men ändå låter dig göra så gott som allt du kan tänka dig när det handlar om att anpassa Drupals utseende. (Faktum är att de första fyra avsnitten handlar om saker som egentligen inte är att tema Drupal i strikt bemärkelse, utan bara går igenom sådant som man bör kunna för att ha nytta av temafärdigheterna.)

Upplägget ser ut som följer.

1. Vitt bälte: Principer och filosofi
2. Gult bälte: Inställningar i admin-gränssnittet
3. Orange bälte: Ladda hem nya teman
4. Grönt bälte: Hantera CSS och Javascript
5. Blått bälte: page.tpl.php (och regioner)
6. Brunt bälte: Andra template-filer
7. Svart bälte: template.php och temafunktioner
8. Bonus: Om att tema formulär

Johan Falk
http://johanfalk.net/contact

VITT BÄLTE: PRINCIPER OCH FILOSOFI

Detta är den första delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

Hemligheten med att använda Drupal effektivt är i många lägen att se till att man behöver göra så lite som möjligt. Tyvärr betyder det att man ibland bör lära om några av sina vanor, och göra saker på ett sätt som vid första anblicken inte är det snabbaste och enklaste. Varför det? För att en lite extrajobb för stunden kan betala sig flera gånger om i långa loppet.

En av sakerna som klassar in under den principen är att inte ändra i originalfiler: *Don't hack core.* är byggt så att man ska kunna skapa egna filer och skriva egna funktioner, som ersätter de filer och funktioner som finns i moduler och teman. Om man hittar den fil eller den funktion som styr precis det man vill ändra är det mycket bekvämt att göra de ändringar man vill, spara, och köra vidare med sin webbplats. Men risken är stor att det leder till extrajobb i framtiden, när man vill installera säkerhetsuppdateringar eller uppgradera bort buggar. Att se till att inte jobba i onödan handlar inte bara om ansträngningar idag, utan även att slippa huvudvärk imorgon.

(Om du är en hobbydrupalist som just laddat hem ett tema och vill ändra ett par småsaker behöver du inte skämmas om du gör det direkt i källfilerna. Det är inte det bästa sättet, men det är fullt förståeligt om du gör det i alla fall. Ändra däremot ingenting i Drupal core – det vill säga kärninstallationen – eftersom det med stor sannolikhet leder till huvudvärk i framtiden.)

En annan sak som innebär mer jobb nu men mindre på sikt är att skaffa sig rätt verktyg att jobba med. Se till att ha en bra textredigerare att skriva kod i, och om du börjar arbeta med komplexa teman vill du definitivt ha ett verktyg som kan arbeta med (och söka i) flera filer på en gång. Du vill också ha en bra webbläsare – personligen föredrar jag Firefox plus tillägget Firebug (medan andra föredrar tillägget Web developer). Om du inte testat Firebug bör du lägga den tid det tar på att ladda hem det genom verktygsmenyn, tillägg, lägg till, och söka upp Firebug. När du väl börjat kommer du inte att vilja vara utan.

Till slut bör du också se till att installera Drupalmodulen *Devel*. Det är ett packe av olika moduler som är användbara när man utvecklar webbplatser, och modulen *Theme Developer*är i vissa lägen oumbärlig när man vill anpassa teman.

Vitt bälte i att tema Drupal handlar om att inte vara rädd att göra lite extra ansträngningar för att spara tid på sikt. Det är inte kul, men det är nyttigt. *Wax on, wax off.*

Johan Falk
http://johanfalk.net/contact

GULT BÄLTE: INSTÄLLNINGAR I ADMIN-GRÄNSSNITTET

Detta är den andra delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

Att använda admin-gränssnittet för att ändra utseendet på en Drupalsajt kanske inte borde kallas theming, eftersom theming normalt handlar om att göra i ordning temafiler. Men även inställningar i det administrativa gränssnittet handlar definitivt om att få Drupals utseende att bli som man vill, och det vore fel att utelämna en sådan sak i en temaguide. För att få gult bälte i att tema Drupal ska du känna till vilka inställningar för utseende som du kan göra genom Drupals administrativa gränssnitt.

Inställningar för sajten

På sidan administration > inställningar > information om webbplatsen finns några av de mer grundläggande saker som visas på webbplatsen – så som webbplatsens huvudtitel.

På samma sida hittar du också lite kryptiska inställningar som heter *slogan* och *syfte* (*mission).* De är (enligt min enkla mening) fossiler från en tid då man tänkte att varje webbplats vill ha ett speciellt meddelande som visas på förstasidan (syfte) eller alla andra sidor (slogan). Som vi kommer att se senare sköts dessa uppgifter – i den mån man är intresserad av dem – lika väl genom Drupals funktioner för block, men om du skulle vilja ge din webbplats en slogan och ett syfte kan du testa det här.

Samma sak gäller egentligen också sidfoten som du kan hitta på samma sida, även om sidfot är en lite mer rimlig funktion på en webbplats.

Inställningar för block

En mer intressant funktion i Drupals gränssnitt är möjligheten att flytta block mellan olika regioner. Du sköter detta på sidan administration > uppbyggnad > block. Vilka regioner som finns tillgängliga varierar för olika teman, och därför ställer man också in visning av block för varje tema för sig.

De listade blocken kommer mer eller mindre färdiga från installerade moduler. Men om man vill går det utmärkt att skapa block med ett handskrivet innehåll, så som bilder eller korta textmeddelanden. Det görs genom fliken *Lägg till block*.

Det kan vara värt att notera att block som vanligtvis är vertikala – som huvudmenyn eller loginblocket – blir horisontella om de placeras i en region som är inställd på att visa innehåll inline.

Styra visning av block

Varje block har en länk för inställningar, där man bland annat kan ändra när blocket ska visas. Visningen går att begränsa på ett par olika sätt. Det som är enklast är att antingen begränsa så att endast vissa användarroller ser blocket, eller så att blocket endast visas (eller inte visas) på vissa sidor på sajten. Den som kan PHP har också möjlighet att sätta upp mer avancerade villkor för när blocket ska visas, så som varannan torsdag eller om en användare redigerar sin egen profil.

På samma sida kan man också ange en egen rubrik för blocket. Ett tomt fält ger defaultrubriken för blocket, medan man ska skriva <*none*> för att få en blank rubrik.

Menyer
Menyer handlar mer om gränssnitt än utseende. Men i vilket fall kan det vara bra att veta att man skapar och redigerar på ett ställe (kolla administration > uppbyggnad > menyer), och bestämmer om och var menyblocket ska synas på ett annat (bland inställningarna för block).

Inställningar för teman

Det sista stället att känna till när det gäller inställningar i Drupals online-gränssnitt är inställningar för teman. Du hittar det under administration > uppbyggnad > tema. Sidan innehåller en lista över de teman som finns tillgängliga, och inte minst innehåller den också länkar till inställningar för varje tema.

På sidan med inställningar kan du ändra ett antal saker, så som bildfil för logotyp eller bokmärkesikon, eller stänga av visning av inbyggda temaelement som sökruta eller webbplatsens namn.

En inställning som är speciellt värd att nämna är det som heter *primary menu* och *secondary menu*. Många (men inte alla) teman har speciella platser där man kan skriva ut huvudvalen i en eller två utvalda menyer. De platserna kallas *primary* respektive *secondary menu*. Man bör känna till att om båda dessa hänvisar till samma meny kommer secondary menu visa eventuella underval till det aktiva valet i primary menu – de kommer alltså inte båda att visa huvudvalen i samma meny. Inställningar för detta finns under administration > uppbyggnad > meny > inställningar.

Vissa teman innehåller extra inställningar, och därför har varje tema en egen sida för inställningar. Med core-temat Garland kan du till exempel ställa in färgscheman, och med bastemat Zen kan du exempelvis ändra hur breadcrumben ser ut och även stänga av cachning av temat (vilket är praktiskt under utveckling). Under de *globala* inställningarna finns ytterligare en inställning du kan göra, nämligen att bestämma vilka nodtyper som ska visas tillsammans med en textrad som berättar när och av vem noden skapades *(“submitted by”)*.

Med fler moduler kan det dyka upp fler möjligheter att styra utseendet genom onlinegränssnitt, men detta är en bas som man bör känna till. Gult bälte i att tema Drupal innebär att du inte gör onödigt temaarbete för att åstadkomma sådant som du kan göra genom onlinegränssnittet.

Johan Falk
http://johanfalk.net/contact

Orange bälte: Ladda hem nya teman

Detta är den tredje delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

De teman som följer med Drupal core är inte särdeles imponerande. (Garland kanske inte är så fult eller tråkigt som de flesta säger, men eftersom det är defaulttemat i Drupal är det lätt att tröttna på det.) Rätt snart börjar man leta efter andra teman att ladda hem och installera, och att göra det på rätt sätt är orange bälte i att tema Drupal.

Var ska man leta?

Alla som lägger ett större jobb på att bygga ett tema, och gör det fritt tillgängligt, bör lägga upp temat på Drupals huvudsajt, drupal.org. På portalens startsida finns en länk som går direkt till listan över teman, och det är en bra plats att leta efter teman.

En annan plats som är värd att besöka är themegarden.org. Där finns många (men kanske inte alla) teman från drupal.org installerade, och man kan bläddra mellan dem för att se hur de ser ut in action. Det går inte att se alla aspekter av temorna, eftersom webbplatsen har ett fixt innehåll, men man kan åtminstone få en uppfattning om hur ett tema ser ut.

Bedöma teman: the good, the bad and the ugly

Att skilja bra teman från dåliga är inte en helt lätt uppgift. Det kan till och med vara svårt att skilja snygga teman från fula teman, eftersom många av de mest flexibla och genomarbetade termorna som default har extremt avskalade (och därmed rätt tråkiga) utseenden – men samtidigt kan de i rätt händer förvandlas till mycket snygga kreationer.

När du väljer mellan olika teman att ladda hem ska du först fråga dig om du vill ha ett i princip färdigt tema att använda, eller om du vill utgå från ett “tomt” tema och bygga upp utseendet själv. I det senare fallet bör du kolla in listan nedan, som innehåller teman som är designade för att bygga vidare på. (Min personliga favorit är Zen.)

Om du däremot är ute efter ett färdigt tema att bara göra mindre anpassningar i finns det några saker du bör tänka på innan du gör ditt slutgiltiga val. Det första är att temat ska funka för Drupal 6. Det andra, som är lite knepigare, är att temat bör vara snyggt även i den HTML som matas ut. Det finns ett par olika sätt att kolla det på. Ett första test är att kolla in källkoden för en sida som använder temat (antingen på themegarden.org eller på en installation du gör själv). Kontrollera att temat *inte*är baserat på tabeller, utan att positionen på olika element styrs med hjälp av CSS. De flesta teman har kastat ut tabelltänkandet, men det finns fortfarande kvar på vissa håll. (Tabellbaserad layout är betydligt mindre flexibel än CSS-baserad, och dessutom är den svårare att tolka för sökmotorer – vilket gör webbplatsen svårare att hitta för folk som googlar.)

Bryr man sig mycket om att webbplatsen ska komma högt upp på sökmotorresultat finns det ytterligare ett par saker man kan kolla in. Det första är att huvudinnehållet på sidan kommer så högt upp som möjligt i sidans HTML-kod. Allra helst ska rubriken för sidan (noden) som presenteras vara det första textelementet i HTML-bodyn – och inte till exempel hamna efter en hel lång lista med länkar i en meny. Man bör också se till att header-taggar används på ett konsekvent sätt i temat. (De flesta brukar säga att h1 ska användas för sajtens namn och rubrik, h2 för aktuell sidas rubrik, och h3 för underrubriker på sidan. Det råder delade meningar om rubriker i block bör vara h2 eller h3, och man behöver över lag inte tro på allt som sägs när det gäller sökmotoroptimering.)

Ett bra och snabbt test är också att köra en webbsida genom w3schools verktyg för HTML-validering. Om du inte är mycket noggrann med både tema och innehåll kan du förvänta dig att få några mindre fel i valideringen, men om ett tema redan från början ger ett antal allvarligare varningar finns det anledning att undvika det temat.

Basteman
Om du läser den här guiden finns det rätt stor sannolikhet att du är intresserad av att snickra egna teman. Här är en lista på ett antal av de teman som är gjorda som utgångspunkter för att bygga egna teman.

- Zen (http://drupal.org/project/Zen)
- Genesis (http://drupal.org/project/genesis)
- Advanced Theme Construction Kit (http://drupal.org/project/atck)
- Basic (http://drupal.org/project/basic)
- Beginning (http://drupal.org/project/beginning)
- Blueprint (http://drupal.org/project/blueprint)
- Clean (http://drupal.org/project/clean)
- Flexible (http://drupal.org/project/flexible)
- Foundation (http://drupal.org/project/foundation)
- Framework (http://drupal.org/project/framework)
- Hunchbaque (http://drupal.org/project/hunchbaque)
- Mothership (http://drupal.org/project/mothership)
- Tendu (http://drupal.org/project/tendu)

En jämförelse mellan temorna (utom Mothership) kan hittas på http://adaptivethemes.com/starter-theme-comparison.html.

Ladda hem och installera

När man väl bestämt sig för vilket tema det är som gäller är vägen till att börja använda det rätt kort. Ladda hem temat, packa upp det, och lägg hela katalogen med temat i mappen sites/all/themes i din Drupalinstallation. Katalogen finns inte från början, utan får skapas för hand.

När temat väl ligger där är det bara att gå till administration > uppbyggnad > tema, välja lämpligt tema som default och spara.

Bakgrunden till den lite märkliga sökvägen är att en och samma Drupalinstallation kan ta hand om ett antal olika webbplatser på en gång. Det sköter man bland annat genom att skapa mappar sites/www.example1.com, sites/www.example2.com, och så vidare. Av prestandaskäl är det bra att lägga specifika moduler och teman i respektive webbplats katalog. De extrasaker man installerar som ska finnas tillgängliga för samtliga sajter läggs i mappen sites/all – vilket också är det rimliga om man bara driver en webbplats.

Den spontana platsen att lägga teman på är rotmappen themes, där man även hittar ett antal andra teman. Tyvärr är det opraktiskt att lägga teman där. När du uppdaterar Drupal nästa gång

finns det risk att ditt eget tema försvinner, eftersom det blivit placerat tillsammans med temorna i Drupal core.

Hade jag varit kung skulle hela Drupal core ligga i en rotmapp som heter core, för att förhindra att folk oavsiktligt ändrar i core-installationen. Men jag känner inte till alla märkliga och obekväma följdändringar som skulle ske av det. Och jag är inte kung.

Summan av kardemumman: Orange bälte i att tema Drupal får du om du på ett stabilt sätt kan leta upp, jämföra och installera utvalda teman på din sajt. Rätt ställe är intethemes-katalogen.

Johan Falk
http://johanfalk.net/contact

Grönt bälte: Hantera CSS och Javascript

Detta är den fjärde delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

Hur får man runda hörn till blocken på sin Drupalsajt? Hur får jag menyblocket, men inte något annat block, att få en viss bakgrundsfärg?

Frågorna ovan är rätt viktiga saker om man håller på att utveckla ett eget tema, men de har mycket lite med Drupal att göra. Oavsett vilket system man använder för att driva webbplatser kommer sådana saker att skötas med CSS, och jobbet för att åstadkomma ändringarna handlar om att hitta rätt CSS-fil, rätt CSS-klasser eller ID:n, och lägga in rätt CSS-kod för att påverka utseendet på webbsidan.

För den som inte är speciellt bekant med CSS ter sig språket som ett sätt att få alla länkar att bli i en viss färg, eller sätta texten alla paragrafer till ett visst typsnitt. Men det går att göra mycket mer än så. Så länge elementen på sidan är uppmärkta på ett trevligt sätt går det att påverka endast utvalda delar av webbsidan, och till exempel ge loginblocket en annorlunda formatering. Det går också att använda CSS för att välja ut de första tre orden i den första paragrafen på sidan och ge dem en egen stil. Och inte minst går det att använda CSS för att styra hur olika element ska placeras på sidan.

Innan du djupdyker i hur man temar Drupal bör du känna till hur allmänna webbspråk som (X)HTML, CSS och Javascript fungerar. Om du vet hur dessa språk fungerar – eller allra minst vad man kan och inte kan göra med dem – kommer du att ha bra förutsättning att använda mer avancerad Drupal-temning *endast när det behövs*. Om du däremot inte vet hur man åstadkommer skuggeffekter med hjälp av CSS spelar det ingen roll hur mycket tid du lägger på att leta runt bland Drupals temafunktioner – i slutändan kommer de skuggorna att komma från CSS-kod.

Sensmoralen är klar: Se till att lära dig de breda designfärdigheterna innan du ger dig i kast med de mer avancerade verktygen. Med CSS och ett bra grundtema (exempelvis Zen) kan man åstadkomma 95 procent av det som krävs för ett färdigt tema.

Redigera teman: skapa underteman

Det korrekta sättet – *best practice* – för att redigera teman är att skapa ett *undertema (sub theme)* till temat man har laddat hem. När man precis börjar lära sig att bearbeta teman känns det som onödigt arbete, men fjärde eller femte gången upptäcker man att det finns en del vinster.

Att skapa ett undertema är enkelt gjort, och går till på följande vis:

- Skapa en mapp i temakatalogen, och ge den ett namn som matchar undertemat. (Underkatalogens namn kommer att bli systemnamnet för undertemat.)
- Kopiera filen <temanamn>.info till underkatalogen, och byt namn på den till <undertema>.info.
- Öppna .info-filen och ändra alla <temanamn> till <undertema>. Det är också lämpligt att ändra beskrivningarna i filen till saker som stämmer för ditt undertema.

Har du gjort detta rätt borde ditt undertema dyka upp i webbplatsens lista över teman, eftersom .info-filen säger åt Drupal att det finns ett nytt tema att använda. Undertemat borde – än så länge – ge precis samma utseende som modertemat. Det är för att Drupal först letar efter temainställningar i ditt undertema, men om det inte finns några där faller den tillbaka på modertemat. Och än så länge finns det inte några temainställningar i katalogen.

När du hittar CSS-filer som du vill ändra i ändrar du inte i modertemats filer, utan kopierar dem först till undertemats katalog. Väl där kan du göra alla ändringar du vill, och Drupal kommer att välja den redigerade filen framför filen i modertemat. Fördelen med denna approach är dels att du kan gå tillbaka till det orörda modertemat när du vill, dels att du kan installera uppdateringar till modertemat utan att dina egna justeringar påverkas.

Hitta rätt CSS-filer

Om man inte använder rätt verktyg är det ett utdraget arbete att ta reda på varifrån CSS-koden som ger ett visst utseende kommer. Med rätt verktyg är det däremot en enkel match.

Rätt verktyg heter Firefox och Firebug. Med dessa kan du högerklicka på vilket element som helst på en webbsida och välja "inspect element". Detta öppnar en panel för Firebug, där du dels kan se HTML-strukturen för elementet, dels vilka olika CSS-attribut som styr elementets utseende. För varje attribut står det vilken fil attributet kommer ifrån, och på vilken rad i filen du hittar just de kodraderna. Som grädde på moset kan du dessutom gå in i CSS-koden live och göra teständringar. (De ändringarna görs inte i källfilerna, så du måste redigera dem själv för att göra ändringar permanenta.)

Många CSS-filer kommer inte från temat som används, utan från olika moduler i Drupalinstallationen. En sådan CSS-fil redigerar du på samma sätt som en CSS-fil i modertemat – du kopierar den till undertemats mapp, och redigerar sedan efter behag.

Om Firebug skulle berätta att sin CSS-fil har ett namn som ser ut som ett temporärt filnamn (så som 16 slumpvis valda tecken) beror det med stor sannolikhet på att Drupal börjat slå samman och cacha CSS-filer. Stäng av den inställningen genom *optimize CSS* på administration > inställningar > prestanda.

Lägga till fler CSS- och js-filer

I ett temas .info-fil finns en förteckning över de CSS- och js-filer som används. Så länge de finns med i .info-filen kommer de att länkas in korrekt av Drupal.

Om du vill lägga till fler CSS- eller js-filer gör du detta genom att lägga till fler hänvisningar i .info-filen, enligt samma mönster som de andra hänvisningarna i filen (så som `stylesheets[print][] = print.css` för CSS och `scripts[] = scripts.js` för Javascript).

Om ändringar du gör inte får effekt är det värt att rensa cachen för webbplatsen. Det görs enkelt genom ett block som modulen Devel tillhandahåller, eller lite mer omständligt genom administration > inställningar > prestanda.

Två genvägar

Om man tycker att det är jobbigt att redigera CSS-filer, eller man av någon anledning har svårt att nå filsystemet för sin webbplats, är det värt att kolla in de två modulerna CSS Injector och Block Class. De låter en skjuta in extra CSS-kod på utvalda platser på sajten genom ett onlinegränssnitt.

Lösningen kan också vara praktisk om man bara vill göra mycket små/få ändringar i ett tema.

Modulerna når du på http://drupal.org/project/css_injector respektive http://drupal.org/project/block_class.

Det orangea bältet kräver en del av temaninjan, nämligen att man ska ha koll på verktygslådan utanför Drupal. I orange bälte ingår också det första steget i att redigera temafiler för Drupal – att skapa ett nytt undertema, inklusive .info-fil och eventuella förändrade CSS- och js-filer.

Johan Falk
http://johanfalk.net/contact

Blått bälte: page.tpl.php (och regioner)

Detta är den femte delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

Det blåa bältet i att tema Drupal handlar om en av de mest centrala filerna i Drupalteman, nämligen page.tpl.php. I ett tema behövs endast två filer. Det ena är .info-filen, som berättar för Drupal att mappen innehåller ett tema (för rätt Drupalversion). Det andra är page.tpl.php, som fungerar ungefär som temats index.html – alla sidor som visas på webbplatsen matas ut genom page.tpl.php.

Att redigera page.tpl.php

Om man öppnar en page.tpl.php ser man att det i princip är en vanlig (X)HTML-fil, men där själva innehållet på sidan är utbytt mot PHP-variabler. I början av filen finns exempelvis variabeln $language->language och $head_title, som berättar vilket språk sidan har och vad titeln på sidan är. Senare i filen hittar man variablen $content för huvudinnehållet på sidan, och ofta variablerna $left och $right för vänster respektive höger sidospalt. Alla dessa variabler fylls med hjälp av mallfiler och temafunktioner, som tas upp i senare avsnitt, men deras placering sinsemellan kommer från huvudfilen page.tpl.php.

Vill man göra omfattande ändringar i ett tema vill man förmodligen ändra i page.tpl.php. Som med andra ändringar ska du se till att skapa ett undertema till temat du utgår från, och kopiera page.tpl.php till undertemats mapp. (Kolla in grönt bälte – redigera teman: skapa underteman om du inte redan gjort det.)

Om du bygger en page.tpl.php från grunden bör du se till att få med vissa variabler. Några som kan vara lätta att glömma är:

- $messages, som bland annat innehåller felmeddelanden.
- $tabs, som visar flikar men mest bara används på adminsidor.
- $closure, som kan innehålla viktig kod och bör skrivas ut efter allt annat dynamiskt innehåll.

Fler variabler

Det finns en rad variabler som kan användas i page.tpl.php – betydligt fler än de som brukar synas i de page-filer man öppnar. I temat Zen inleds page.tpl.php-filen med en utmärkt dokumentation över vilka variabler som finns tillgängliga (så som eventuell administratörstatus hos besökaren och sökväg till aktuell temakatalog).

Variabler kan förstås inte bara användas för att mata ut innehåll, utan också för att utföra logiska tester. (“Om användaren är administratör, skriv ut extralänkar.”) Man ska dock se till att hålla all komplicerad logik i en separat fil – vanligtvis template.php. Logik i page.tpl.php bör begränsa sig till enkla if-satser. (Se *svart bälte – lägga till nya variabler* om du har behov av mer avancerad logik.)

Olika mallar för olika webbsidor

En finess med page.tpl.php är att man genom att namnge varianter av filen på olika sätt kan man använda olika mallar på utvalda delar av sin sajt. Exempelvis kommer page-node-1.tpl.php att an-

vändas när man besöker nod nummer 1 på sajten, medan page-user.tpl.php kommer att användas när någon användarprofil laddas.

I regel kommer Drupal att leta efter en temafil med ett namn som motsvarar den interna sökvägen till sidan som visas (där node/1 motsvarar page-node-1.tpl.php). Om filen inte hittas letar Drupal efter mer och mer generella mallfiler (som page-node.tpl.php), för att till slut falla tillbaka på temats default – page.tpl.php. Eventuella specialanpassade page-filer skapar man förmodligen lättast genom att kopiera originalfilen och sedan göra de ändringar man vill, men teoretiskt kan man förstås skriva dem från grunden också.

Det finns ett par undantag till regeln om sökväg/filnamn som är värda att nämna. Den första är redigeringssidan för noder, som specialtemas genom page-node-edit.tpl.php (och inte exempelvis page-node-1-edit.tpl.php). Det andra är mallnamnet page-front.tpl.php, som om den finns alltid används för sajtens förstasida, oavsett vilken den är. Ett praktiskt sätt att se vilka möjliga mallfiler man kan använda istället för page.tpl.php är att utnyttja Theme Developer i modulen Devel.

Om man skulle vilja utöka reglerna för hur Drupal letar efter mallar går det att genomföra med lite PHP. Kolla in http://drupal.org/node/223440 för information om hur det funkar.

En viktig sak att känna till är att defaultfilen måste finnas i samma katalog som eventuella varianter för att Drupal ska bry sig om den. Det går alltså inte att lägga in page-node.tpl.php i sitt undertema utan att dessutom ha med page.tpl.php. (Som sagt måste alla teman innehålla en page.tpl.php-fil för att fungera, men underteman kan klara sig utan den så länge modertemat innehåller filen.)

Regioner

Regioner är platser på webbsidan där dynamiskt innehåll kan placeras – vanligtvis block. Det brukar betyda spalter av olika slag, men det är förstås upp till formgivaren att bestämma hur de olika regionerna ska synas.

I page.tpl.php representeras varje region av en enkel variabel, och som default finns $head, $content, $left, $right och $footer. Man kan enkelt lägga till fler regioner genom att redigera i temats .info-fil och skriva rader i stil med `region[min_region] = Min region`. Det kommer att ge upphov till regionvalet *Min region* i listan med block, och allt som läggs i den regionen kommer att kunna matas ut genom regionen $min_region.

Defaultregionerna ovan finns bara med *om man inte deklarerar någon region över huvud taget*. Vill man lägga till en enda region till dessa ska man också lägga till alla standardregioner, om man inte vill sitta med endast en region i slutändan. Om man använder regioner som motsvarar standardregionerna kan det vara klokt att använda standardregionernas namn, eftersom det i vissa lägen gör det lättare att samordna blockinställningar mellan olika teman.

Det blå bältet i att tema Drupal får du om du kan hantera page.tpl.php. Det betyder att du både ska kunna använda varianter av page.tpl.php för att ge olika delar av webbplatsen avvikande struktur och att kunna lägga till fler regioner i temat.

Johan Falk
http://johanfalk.net/contact

Brunt bälte: Andra template-filer

Detta är den sjätte delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

Med det bruna bältet i att tema Drupal når vi det kanske mest intressanta i Drupals temalager – tpl.php-filer (hädanefter ibland kallade tippelfippar, efter förslag i boken Front End Drupal). Med hjälp av dem kan man påverka stora delar av Drupals utseende, utan att behöva grotta ner sig i komplicerade temafunktioner eller knepig PHP.

Man skulle kunna säga att det är här det roliga börjar.

Hur tippelfippar fungerar

En tippelfipp är en liten (eller ibland stor) mall som Drupal använder för att formatera en viss sorts data. Det finns tippelfippar som styr rutan med kommentarer, det finns tippelfippar som styr en nods uppbyggnad, och det finns tippelfippar som ger ramen runt varje block.

Till sin uppbyggnad liknar tippelfippar page.tpl.php (som kan ses som tippelfipparnas hövding) – de innehåller (X)HTML-kod med delar av innehållet utbytt mot PHP. För varje tippelfipp finns det ett antal variabler man kan använda – som skiftar mellan olika tippelfippar – och dessa kan användas både för direkt utmatning och för logiska villkor. (Dokumentation om vilka variabler som finns tillgängliga borde finnas i respektive tippelfipp.)

Även när det gäller redigering fungerar tippelfippar precis som page.tpl.php. Man skapar helt enkelt en kopia av tippelfippen man vill ändra, lägger den i katalogen för sitt undertema, och redigerar så mycket man orkar. Enkelt, snabbt och effektivt!

Att hitta rätt tippelfipp

Något som däremot kan vara lite svårare när det gäller tippelfippar är att hitta originalfilen. Det finns i princip två sätt att göra det på. Det ena är att leta runt bland alla moduler och include-kataloger som Drupal har, tillsammans med eventuella contrib-moduler som man laddat hem. Det andra är att använda Theme Developer i Devel-modulen, klicka på ett element och få reda på vilken mall som användes när det skapades.

Det senare är att föredra.

Men. Inte all data som visas på en sida är skapad genom tippelfippar. Många av core-modulerna och en del av de större contrib-modulerna använder tippelfippar för det mesta av datan som ska visas, men inte alla. De som inte använder tippelfippar förlitar sig istället på temafunktioner – PHP-funktioner som gör i ordning data för utmatning. Det går faktiskt ganska enkelt att göra om temafunktioner till tippelfippar, men det blir föremål för nästa bälte.

Varianter av tippelfippar

Med page.tpl.php kan man variera filnamnet för att få varianter av filen att användas på utvalda delar av webbplatsen. Samma sak går att göra med *vissa* tippelfippar. Här är ett urval av några som förhoppningsvis är användbara.

- node.tpl.php kan ges tillägget -nodtyp (typ node-story.tpl.php), för att tema utvalda nodtyper på ett speciellt sätt. Om du använder nodtyper uppbyggda med hjälp av CCK har du en bra hjälpreda genom modulen Contemplate/Content Template. Genom den modulen kan du få Drupals nuvarande mall för nodtypen (även om den inte finns som en fil), och spara ner i en fil.
- Tippelfippar för nodtyper kan dessutom få tilläggen -body, -teaser eller -rss, om du vill tema utmatning av noden i fulltext, teaserläge respektive som RSS-flöde. (Använder du inte något av dessa tillägg får du en mall som gäller för samtliga utmatningsformer – vilket betyder att teasers och fulltextnoder ser likadana ut.)
- comment.tpl.php och comment-wrapper.tpl.php kan också ges tillägg för nodtyp, för att tema varje nodtyps kommentarer för sig.
- block.tpl.php kan ges en rad olika tillägg för att tema enligt olika sammanhang. De filnamn som Drupal letar efter per default är block-*modulnamn-delta*.tpl.php, block-*modulnamn*.tpl.php, block-*regionnamn*.tpl.php och till slut block.tpl.php. Modulnamn är namnet på den modul som skapar blocket (handgjorda block kommer från modulen Block), *delta* är ett löpnummer som moduler ger varje block som skapas, och regionnamn är alltså systemnamnet på regionen där blocket visas. Genom att variera namnen på tippelfipparna kan man alltså låta enstaka block, block från vissa moduler eller block i vissa regioner få olika mallar.

Om du vill veta fler sätt att ändra namn på tippelfippar rekommenderas Theme Developer. Om du klickar på ett element får du inte bara reda på vilken mall som används för elementet, utan också vilka filnamn som skulle kunna ha använts istället.

När helst du använder varianter av tippelfippar ska du se till att du har defaultfilen i samma katalog som specialfilen – annars kommer specialfallet inte att hittas. (Det funkar alltså inte att *bara*ha node-story.tpl.php i sin temafil om man vill skapa en ändra utseendet på story-noderna – du måste också ha node.tpl.php i samma katalog.)

Du bör också se till att hålla tippelfippar fria från avancerad logik, vilket ibland kan vara lättare sagt än gjort. Om innehållet i din tippelfipp kräver avancerad logik bör du kolla in avsnittet *lägga till variabler* under svart bälte.

Rött bälte i att tema Drupal har du förtjänat om du kan hantera tippelfippar – det vill säga hitta, redigera och vid behov skapa specialvarianter av tpl.php-filer. Som överkurs kan du också skapa nya variabler, för att hålla tippelfipparna så rena och lättlästa som möjligt.

Johan Falk
http://johanfalk.net/contact

SVART BÄLTE: TEMPLATE.PHP OCH TEMAFUNKTIONER

Detta är den sjunde delen i guiden Sju steg mot svart bälte i att tema Drupal 6.

Med det svarta bältet tar vi oss in i vad som ibland beskrivs som *the theming ninja's weapon of choice:* template.php. Det är en fil som brukar innehålla olika funktioner och lite mer avancerad logik för temat.

I detta avsnitt kommer vi att titta på temafunktioner som alternativ till tippelfippar. Vi ska se hur man kan skapa sina egna versioner av temafiler, och även hur man kan ersätta temafiler med tippelfippar. Till slut ska vi också gå igenom hur man kan flytta knölig logik från tippelfippar till template.php.

Temafunktioner

De element på en Drupalsajt som inte temas genom tippelfippar använder i stället *temafunktioner*– som sköter logik och bearbetning av data, för att skicka ut snyggt formaterad (X)HTML-kod.

Temafunktioner är ett lite mer omständligt sätt att formatera utdata än vad tippelfippar är. Temafunktioner är ren PHP, men i utdatan från funktionerna finns (X)HTML med i de allra flesta fall.

Precis som med tippelfippar vill man inte ändra i originalen, utan istället skapar man egna kopior som man redigerar i. Men istället för att ge företräde åt funktioner genom deras placering i filstrukturen blir man tvungen att ändra namnet på funktionen för att åsidosätta defaultfunktionerna.

Ett standardrecept för att skapa egna temafunktioner ser ut som följer:

- Du använder Theme Developer för att ta reda på hur ett visst element fått sin formatering/struktur. Theme Developer säger att det kommer från en temafunktion (theme_xyz) och inte en tippelfipp (xyz.tpl.php).
- Du letar upp originalfunktionen på ett av fyra sätt:
  - Genom en php-fil i lämplig modul eller include-fil. (Theme Developer borde ge information om var du ska leta.)
  - Genom funktionsregistret i Devel-modulen.
  - Genom api.drupal.org, som innehåller dokumentation om alla funktioner och filer som används för drupal.org.
  - Genom ett eget API-register som du skapat genom att installera API-modulen. Det kan vara fiffigt om du har många moduler som inte finns med på drupal.org.
- När du väl hittat funktionen kopierar du *hela* funktionen, inklusive funktionsnamn och avslutande måsvinge, till template.php i katalogen för ditt undertema. Vanligtvis brukar man lägga funktionen sist i filen, men det ska inte spela någon roll så länge du inte råkar lägga den mitt i en annan funktion.
- Byt namn på funktionen, från *theme_xyz* till *temanamn_xyz*. (Det är detta som ser till att Drupal ger företräde för din funktion framför originalfunktionen.)
- Genomför de ändringar du vill i funktionen.

- Spara. Testa. Rensa cachen om du inte ser någon effekt. (Det görs på administrera > inställningar > prestanda, eller genom länkar i ett Devel-block.)

Att förvandla temafunktioner till tippelfippar

I många lägen kan det vara mer praktiskt att ha en tippelfipp istället för en temafunktion. Det gäller inte minst om en designer vill arbeta vidare med formateringen, utan att behöva svepa in all utdata i PHP-kod. (Däremot gäller det förmodligen inte om temafunktionen innehåller tung logik.)

Att förvandla en temafunktion till en tippelfipp är en överraskande snabb och enkel process. Det hela går ut på att skapa en tippelfipp med rätt namn och fylla den med den output som kommer från temafunktionen (inklusive eventuella villkor). Filnamnet ska vara samma som funktionsnamnet, men utan theme_ (eller motsvarande) och med understreck utbytta mot bindestreck. Exempelvis kan en funktion `theme_xyz_abc() {print “<h2>Hello world!</h2>”;}` mot en tippelfipp med namnet *xyz-abc.tpl.php* och innehållet `<h2>Hello world!</h2>`.

Inte alla temafunktioner gör sig bra som tippelfippar. Temafunktioner som använder *`for each`*-loopar är till exempel svåra att omvandla till tippelfippar på ett bra vis. En annan svårighet är att hålla avancerad logik utanför mallfilerna. Det blir föremål för nästa avsnitt.

Lägga till variabler

En likhet mellan page.tpl.php och tippelfippar är att man bör hålla avancerad logik borta från tippelfipparna. Det beror på flera saker. Det mest uppenbara är att mallen blir svår att överblicka om det dyker upp flera rader långa if-satser eller variabler som byggs upp med många olika villkor. Men det kan också påverka webbplatsens prestanda på otrevliga sätt. (Om din comment.tpl.php innehåller villkor som kräver att data hämtas från databasen kan det bli jobbigt om en nod har 200 kommentarer eller så.)

I stället för att lägga in avancerad logik i tippelfippar bör man flytta den till en separat php-fil, nämligen template.php. I den filen kan man vanligtvis hitta diverse PHP-sjok och deklarationer som avancerade teman använder.

Ett praktiskt sätt att hålla tippelfippar rena från knölig kod är att ersätta all den knöliga koden med en enda variabel. Om man exempelvis har omständliga villkor för hur användarnamn ska matas ut tillsammans med noder ersätter man alla dessa villkor med en variabel, exempelvis $formatted_user_data, och låter template.php befolka den variabeln med den data som behövs. Resultatet är en tippelfipp som är renare och snyggare, och om man har tur också en del vinst i prestanda.

För att skapa nya variabler till en tippelfipp behöver man använda temalagrets preprocess-funktioner. Mallen för detta ser ut som följer:

```
temanamn_preprocess (&$vars) {
$vars['variabelnamn'] = uttryck;
}
```

Alternativt:

```
temanamn_preprocess_tippelfipp (&$vars) {
$vars['variabelnamn'] = uttryck;
}
```

I det första fallet blir variabeln tillgänglig i *samtliga* tippelfippar (inklusive page.tpl.php), men i det senare fallet blir den bara tillgänglig i mallfilen *tippelfipp*.tpl.php. Om variabeln bara kommer att användas i en enda tippelfipp är det senare att föredra, eftersom det minskar risken för krockar i variabelnamn och dessutom medför små prestandavinster.

Funktionerna läggs in i template.php, som kommer att hittas och köras automatiskt. Om template.php tidigare var tom bör man se till att inleda filen med `<?php` – däremot är det best practice att inte avsluta filen med `?>`.

Johan Falk
http://johanfalk.net/contact

Bonusavsnitt: Om att tema formulär

Detta är bonusavsnitt i guiden Sju steg mot svart bälte i att tema Drupal 6.

Den som arbetar med att tema Drupal viss förmodligen förr eller senare ändra på hur formulär i Drupal ser ut. Till viss del görs detta på samma vis som all annan theming – exempelvis när det gäller CSS. Men till stor del byggs (X)HTML-innehållet i formulär inte upp med hjälp av tippelfippar och temafunktioner, och följaktligen måste man arbeta på ett annorlunda sätt när man vill ändra i dem.

Detta är ett kort avsnitt som berättar hur man kan gå tillväga för att ändra formulär i Drupal. Avsnittet bygger i stort på kapitel 9 i boken *Drupal 6 Themes*.

Formulärets id

För att påverka formulär i Drupal är det mycket praktiskt att känna till formulärets id. Alla formulär på en Drupalsajt har ett id, och lättast hittar man det genom att besöka sidan som visar formuläret och leta i källkoden. Mot slutet av formuläret ska det finnas ett input-element av typen `hidden` och med namnet `form_id`. I value-fältet för elementet står formulärets id, som brukar vara av formen `user_login_block`.

Den som hellre letar i källkoden har stor chans att hitta formulärets id i funktionen som bygger formuläret (så som `user_login_block()`).

Formulärets array

En annan viktig sak att känna till med formulär man vill ändra är den array som bygger upp formuläret. I arrayen finns all information för elementen i formuläret – typ av element, namn, innehåll, med mera. För att ändra formuläret behöver man i många lägen ändra på formulärets array.

Det lättaste sättet att hitta arrayen är förmodligen gå till sidan där formuläret visas och med hjälp av Theme Developer läsa bland de arrayelement och värden som användes för att bygga upp formuläret.

En annan metod är att använda PHP-funktionen print_r för att skriva ut hela arrayen. Om du föredrar det kan du använda kodsnutten här nedan, under *använd temafunktioner*.

Fyra sätt att ändra i formulär

Vanliga temametoder
En hel del går att åstadkomma genom att använda CSS och/eller redigera de tippelfippar som ibland är används för att mata ut de färdiga formulären.

Använd temafunktioner
Vill man ändra inuti formuläret blir man tvungen att gå in och mecka i den array som Drupal använder för att bygga upp formuläret. Det är inte skitenkelt, men inte heller skitsvårt. För att lyckas behöver vi dels bygga en funktion som ändrar på arrayens innehåll, dels säga åt Drupal att funktionen finns och bör anropas.

Ett sätt att genomföra detta på är att lägga in följande två funktioner i template.php:

```
// Säg åt Drupal att en ny temafunktion finns tillgänglig
```

```
function temanamn_theme() { // Byt ut temanamn mot temats namn
  return array (
    'form_id' => array(     // Byt ut form_id mot formulärets id
      'arguments' => array('form' => NULL),
    ),
  );
}

// Funktion för att modifiera formulärets array
function temanamn_form_id($form) { //Byt ut temanamn och form_id
  // Lägg till dina ändringar här!
  // print_r($form); // Skriver ut formulärets array
  $output .= drupal_render($form);
  return $output;
}
```

Bygg en minimodul

Ett effektivt, men kanske något omständligt sätt att ändra formulär är att bygga en minimodul för ändamålet. Modulen bör, som alla andra custom-moduler, läggas som en underkatalog till sites/all/modules (och alltså inte till rotkatalogen modules).

I katalogen behöver du ha två filer. Dels en .info-fil, som berättar för Drupal att katalogen innehåller en modul för rätt Drupalversion, dels en fil som innehåller modulkoden. Namnet på modulens underkatalog bestämmer namnet på .info-filen och .module-filen.

.info-filen bör innehålla följande:

```
;$id$
name = Ändring av formulär
description = Ändrar utvalda formulär på sajten
package = other
core = 6.x
```

.module-filen bör innehålla följande:

```
<?php
//$id$
/**
*
* Ändrar i utvalda formulär på sajten.
*
*/
function formmod_form_alter(&$form, $form_state, $form_id) {
  // Ändra villkor och kod som du önskar.
  If ($form_id == 'user_login_block') {
    $form['submit']['#value'] = t('Logga in mig!');
  }
}
```

Observera funktionen t() som omsluter texten i formuläret. Den används för att kunna dra nytta av Drupals funktioner för översättningar.

Använd fiffiga genvägar

Många av de ändringar som man vill göra i formulär är sådana som andra velat göra tidigare, och det redan finns snabba lösningar för. Här är tre exempel:

- Om du vill dölja valda element i ett formulär (så som loggfältet när man redigerar noder) rekommenderas modulen Formfilter (http://drupal.org/project/formfilter).
- Om du vill lägga till en spara-knapp överst i formuläret för att redigera noder rekommenderas modulen
- Om du vill ändra ordning på eller beskrivning av utvalda fält kan modulen Form Defaults (http://drupal.org/project/formdefaults) vara en bra lösning. (Detta är dock en rekommendation som jag gör utan att ha testat modulen själv!)
    - Om du har en sajt på svenska är ett annat alternativ att utnyttja Drupals funktioner för översättningar för att ge fältet ett annat namn – men det ändrar samtidigt alla ställen där exakt samma textsträng förekommer. Översättning förutsätter att du har Locale-modulen aktiverad, och du genomför översättningen på administrera > uppbyggnad > översätt gränssnitt > sök.