Institutt for informasjonssystemer

Innebygde funksjoner

IS-115 Webprogrammering i PHP | Peter André Busch

Innhold

Leksjon 2: innebygde funksjoner

I denne leksjonen ser vi på innebygde funksjoner, som vi kommer til å benytte ofte. PHP har svært mange innebygde funksjoner som du finner via www.php.net/manual/. Kapittel 3 i læreboka behandler denne tematikken.

1. Om innebygde funksjoner

Innebygde funksjoner er en sentral del av PHP-rammeverket. Det er funksjoner utviklet for gjenbruk av andre programmere. De fungerer litt som π gjorde i skolen – du visste at den var 3.14, men behøvde ikke å vite hvorfor (de spesielt interesserte visste det). De innebygde funksjonene gjør slik dokumentasjonen viser og hva som skjer under panseret er ikke nødvendig for oss å vite (men mindre vi er spesielt interesserte 😊).

1.1 Hensikt

Hensikten med innebygde funksjoner er å tosidig: (1) det skal ikke være nødvendig å finne opp hjulet to ganger og (2) praksis med beste løsninger etableres. På den måten kan vi dra nytte av arbeidet som andre dyktige PHP-programmerere har gjort før oss. Spesielt, så sparer det oss for tid som vi kan bruke til å løse andre problemer.

1.2 Dokumentasjon

De innebygde funksjonene i PHP er godt dokumenterte, så dersom du lurer på noe kan du bare slå opp i manualen som du finner på www.php.net/manual. Som regel finner du flere eksempler som du kan se på samt kommentarer fra brukere. Denne dokumentasjonen kommer du til å bruke flittig (du har kanskje allerede gjort det?). Så kan du selvsagt google enkelte problemstillinger – det er sannsynlig at andre har spurt om det samme før.

2. Strengbehandling

Det finnes en rekke funksjoner som kan hjelpe deg å behandle tekststrenger. Mulighetene er mange; telle antall tegn i en tekststreng, gjøre store bokstaver om til små bokstaver eller søke etter et tegn (eller flere tegn) i en tekststreng.

La oss ta et eksempel med et skjema for registrering av medlemmer i et medlemssystem. I alle skjemaer vil det være nyttig å kontrollere (og kanskje manipulere) informasjonen som blir sendt inn via skjemaet. Et eksempel på kontroll er følgende:

> Du er administrator på systemet og lagrer en rekke data om medlemmet. Et av feltene dreier seg om utfylling av mobilnummer. Du skriver feil og taster inn ni sifre istedenfor åtte. Her kan du bruke PHP-funksjonen `strlen()` til å undersøke om lengden på strengen er riktig. Ettersom vi er i Norge, kan du også kontrollere om nummeret kan være riktig (f.eks. utelate numre som begynner på 1 og 2). Det finnes enda mer avanserte måter å gjøre dette på, men det utelater vi for nå.

Dette er et eksempel på hvordan en strengfunksjon kan brukes for å kontrollere innhold i skjemafelt. La oss ta et annet eksempel, denne gangen hvordan informasjon kan manipuleres:

> Du er fremdeles administrator på systemet og skriver inn fornavn og etternavn til medlemmet. Dersom du skriver alt med små bokstaver blir navnet lagret feil i databasen. Du bruker PHP-funksjonene `strtolower()` og `ucfirst()` til å forsikre deg om at navnet blir lagret riktig. Ved å bruke disse funksjonene ville det også blitt riktig om du skrev alt med store bokstaver først (eller hadde en stor bokstav midt i navnet for den saks skyld osv.). Hensikten er altså å sikre seg mot at brukeren skriver feil i skjemaet. Funksjonen `htmlentities()` vil erstatte HTML-tagger med ASCII-kode slik at vi unngår sikkerhetsproblemer med det som tastes inn (dette går vi gjennom i en senere modul). Funksjonen `strip_tags()` fjerner HTML- og PHP-tagger fra en streng.

Dette var to praktiske eksempler på hvordan strengfunksjoner kan brukes i PHP. Det finnes flere eksempler i læreboka som dere kan se på. Et annet eksempel som jeg ønsker å nevne er passordgenerator. Dette kan dere få bruk for i et medlemssystem dersom dere ønsker at hvert medlem skal kunne logge seg på (selv om dette ikke er et krav i prosjektet i dette kurset). Et tips her er å lage en egen funksjon som genererer passord ved bruk av `rand()`- og `md5()`-funksjonene. Den første funksjonen genererer en tilfeldig verdi mellom 0 og RAND_MAX. Den andre funksjonen genererer en hashet streng på 32 heksidesimale sifra (uansett hva som sendes til den).

Denne funksjonen ble også tidligere benyttet når passordet skal lagres i databasen. Da ble passordet lagret i databasen som en md5-streng og når brukeren logget inn ble md5-hashet av det som skrives inn i passordet sjekket mot strengen (hashet) i databasen. Helt like strenger vil alltid generere samme streng ved bruk av `md5()`-funksjonen, mens en liten forskjell vil føre til helt forskjellige resultater. Denne funksjonen brukes imidlertid ikke lengre for passord ettersom den samme strengen alltid genererer samme hash. På grunn av dette finnes det lister over md5-hash og deres originale passord – og det er dermed lett å knekke ofte brukte passord. Vi går gjennom passord senere i modul 10 om sikkerhet.

Det finnes så mange streng-funksjoner at vi kan ikke gå gjennom dem her. Gjør dere kjent med de mulige funksjonene i PHP-manualen (https://www.php.net/manual/en/ref.strings.php). For øvrig kommer dere borti dem når dere programmerer og behøver funksjonalitet for å løse ulike problemstillinger.

3. Behandling av dato og tidspunkt

En viktig del av programmering er å håndtere datoer og tidspunkt. Det kan vi gjøre ved å bruke innebygde funksjoner. Dere har allerede blitt introdusert for `date()`-funksjonen som i stor grad brukes til dette. Når du slår opp `date()`-funksjonen i PHP-manualen, kan du se alle mulige formateringer som er mulig å benytte med denne funksjonen. Her er noen eksempler:

```php
$today = date("m.d.y"); // 03.31.20
$today = date("F j, Y, g:i a"); // March 31, 2020, 5:16 pm
$today = date("Y-m-d H:i:s"); // 2001-03-10 17:16:00
                              (the MySQL DATETIME format)
$today = date("d.m.Y k\l. H:i"); // 31.03.2020 kl. 17:16
```

I det siste eksemplet må vi vise `date()`-funksjonen at vi er ute etter bokstaven l (liten L) og ikke formateringen l som vil gi oss en tekstuell representasjon av ukedagen i dag (på engelsk). Dette gjør vi ved å sette en backslash i forkant av bokstaven. Dette er bare noen eksempler blant mange. Denne funksjonen kan brukes til å arbeide med dato og tidspunkter på mange forskjellige måter. Merk at det

er forskjell på store og små bokstaver. Funksjonen `checkdate()` vil hjelpe deg med å finne ut om en dato er gyldig.

En annen nyttig funksjon er `mktime()`-funksjonen som lager et Unix-tidsstempel av hvilken som helst dato. Dersom du er født 31. januar 1994, så kan du bruke `mktime()`-funksjonen til å lage et tidsstempel for deretter å bruke `date()`-funksjonen for å finne hvilken ukedag du ble født på. Eller, enda enklere, du kan bruke `mktime()`-funksjonen inni `date()`-funksjonen:

```
echo date("l", mktime(0, 0, 0, 1, 31, 1994));
```

Legg merke til at `date()`-funksjonen tar to argumenter hvor tidsstemplet er det andre. Du må ikke ta med tidsstemplet. I de første eksemplene ovenfor gjorde vi ikke det, men da tar `date()`-funksjonen også utgangspunkt i tidspunktet akkurat nå (dvs. til serveren).

I medlemssystemet er det kanskje mer fornuftig å bruke en predefinert liste over datoer (altså dag, måned, år) i nedtrekksmenyer. Dette er mer brukervennlig og kan føre til mindre feil. I et bestillingssystem for f.eks. hoteller, kan det være lurt å sette dagens dato som standard. Læreboka viser deg hvordan du kan gjøre dette med bruk av PHP og HTML.