Automatisk generering af kodeord via PHP

Grundig gennemgang af hvorledes man via et simpelt bibliotek, kan oprette nærmest enhver type kodeord. Ialt 6 "sværhedsgrader" inkluderet ! Lige til at ligge i ethvert login, forum, bruger system.

Skrevet den **10. Feb 2009** af **limemedia** I kategorien **Programmering / PHP** | ★★★☆☆

Ofte står man i en situation, hvor en bruger skal kunne identificeres unikt overfor systemet. I en sådan situation, kan man vælge om man ønsker brugeren selv skal indtaste et kodeord, modtage et inspirerende kodeord fra systemet eller fastligges på et fast kodeord som systemet så generer til brugeren.

Herunder har jeg forsøgt at skabe et generel funktions bibliotek, der kan generere alt fra yderst simple til meget komplekse kodeord, samt en funktion der kan skabe "udtalelige" kodeord, der i test har vist sig nemmere for brugerne at huske; men på samme tid også mere forudsigelige for brugere der ønsker at kompromittere et kodeord.

Systemet vil også kunne anvendes til at generere tekststrenge, til anvendelse i "verificer min brugerprofil" scenarier, hvor tilfældige tekststrenge skal anvendes.

Biblioteket består af tre funktioner, password_generate, password_generate_advanced og password_generate_pronouncable, hvor password_generate er ment som en wrapper funktion, der kalder de to andre. Der er ingen krav til at anvende denne wrapper funktion, dens formål er dog at simplificere brugen. Den mere advancerede bruger ønsker evt at anvende de specialiserede funktioner.

Password_generate, har tre parametre, $nice, $length og $allowchars. $nice angiver "sværhedsgraden" af kodeordet, som er et heltal mellem 1 og 6, hvor sværhedsgraden stiger med et højere tal. $length angiver den ønskede længde af kodeordet - hvis ingen længde er angivet, vil der genereres en tilfældig længde mellem 5 og 9 tegn. $allowchars kan anvendes til at angive hvilke specifikke tegn der må indgå i kodeordet, parameteren her forventer en streng. Det er muligt at prioritere visse tegn ved at angive disse flere gange i strengen, måtte man have sådanne ønsker.

Sværhedsgraderne er inddelt som følger:
1: Generer et udtaleligt kodeord
2: Generer et kodeord bestående af lowercase a-z bogstaver og tal mellem 0 og 9
3: Ligesom 2, men tegnene 'O01lI5S' giver ofte problemer med læsbarheden og afvises som gyldige tegn
4: Ligesom 3, men tillader uppercase A-Z bogstaver
5: Ligesom 4, men tillader specialtegn '!#$%&()*+-./;<=>@_'
6: Tillader tal, små og store bogstaver samt specialtegn, uden at korrigere for læsbarhed problemer
Hvis sværhedsgrad er angivet forkert, vil 3 være valgt som en medium løsning.

Held og lykke med oprettelsen af kodeord. Funktionen vil i sin helhed inklusiv indrykninger og formatering kunne findes på http://www.ljweb.com/usefulscripts/

Kode eksempel

```
<?php
  # Seed the random generator - consider doing this in a config file once for the entire site
  mt_srand((double)microtime()*1000000);
```

```
 function password_generate($nice = 1, $length=0, $allowchars = "") {
    # Find random password length
    if (!$length) $length = mt_rand(5, 9);

    # pronouncable password
    if ($nice == 1) return password_generate_pronouncable($length);
    # lowercase only, fix similar
    else if ($nice == 2) return password_generate_advanced($length, 0, 1, 0, 0, 1, $allowchars);
    # lowercase and numbers only, fix similar
    else if ($nice == 3) return password_generate_advanced($length, 0, 1, 1, 0, 1, $allowchars);
    # both lower and uppercase chars and numbers , fix similar
    else if ($nice == 4) return password_generate_advanced($length, 1, 1, 1, 0, 1, $allowchars);
    # all types of letters, including special chars, fix similar
    else if ($nice == 5) return password_generate_advanced($length, 1, 1, 1, 1, 1, $allowchars);
    # oh my :) the real deal - get it all and dont fix similars
    else if ($nice == 6) return password_generate_advanced($length, 1, 1, 1, 1, 0, $allowchars);

    # $nice contained illegal value, go for the easy 3
    else return password_generate_advanced($length, 1, 1, 1, 0, 1);
 }


 function password_generate_advanced($length = 8, $allow_uppercase = 1, $allow_lowercase = 1,
$allow_numbers = 1, $allow_special = 1, $fix_similar = 0, $valid_charset = "") {
    # Create a list of usable chars based upon the parameters
    if (!$valid_charset) {
      if ($allow_uppercase) $valid_charset .= 'ABCDEFGHIJKLMNOPQRSTUVXYZ';
      if ($allow_lowercase) $valid_charset .= 'abcdefghijklmnopqrstuvxyz';
      if ($allow_numbers) $valid_charset .= '0123456789';
      // if lowercase and uppercase, the chance for a number is less, and thus doubled in the char array
      if ($allow_numbers && $allow_lowercase && $allow_uppercase) $valid_charset .= '0123456789';
      if ($allow_special) $valid_charset .= '!#$%&()*+-./;<=>@\_';
    }

    # Find the charset length
    $charset_length = strlen($valid_charset);

    # If no chars is allowed, return false
    if ($charset_length == 0) return false;

    # Initialize the password and loop till we have all
    $password = "";
    while(strlen($password) < $length) {
      # Pull out a random char
      $char = $valid_charset[mt_rand(0, ($charset_length-1))];

      # If similar is true, check if string contains mistakeable chars, add if accepted
      if (($fix_similar && !strpos('O01lI5S', $char)) || !$fix_similar) $password .= $char;
    }

    # Return password
    return $password;
 }
```

```
  function password_generate_pronouncable($length = 8) {
     # Initialize valid char lists
     $valid_consonant = 'bcdfghjkmnprstv';
     $valid_vowel = 'aeiouy';
     $valid_numbers = '0123456789';

     # Find the charset length
     $consonant_length = strlen($valid_consonant);
     $vowel_length = strlen($valid_vowel);
     $numbers_length = strlen($valid_numbers);

     # Initialize the password and loop till we have all
     $password = "";
     while(strlen($password) < $length) {
       # Pull out a random set of pronouncable chars
       if (mt_rand(0, 2) != 1) $password .= $valid_consonant[mt_rand(0,
($consonant_length-1))].$valid_vowel[mt_rand(0, ($vowel_length-1))].$valid_consonant[mt_rand(0,
($consonant_length-1))];
       else $password .= $valid_numbers[mt_rand(0, ($numbers_length-1))];
     }

     return substr($password, 0, $length);
  }
?>
```

Kommentar af dmcn d. 24. Mar 2004 | 1

Virkelig brugbar artikel med grundig forklaring af funktionerne og gode pointer omkring "gode" kodeord. Att skize_someone: Lars har skrevet begge artikler. ;) (http://www.eksperten.dk/spm/481557)

Kommentar af skizo_someone d. 27. Oct 2004 | 2

Jamen i få fald... rigtig god artikel :D

Kommentar af sandbox d. 18. Jan 2004 | 3

God og anvendelig artikel Især godt at der gives mulighed for forskellige typer password.