

Pertemuan 09

Sistem Login dan Registrasi Menggunakan PHP

Pendahuluan

Nah kemarin kita sudah belajar PHP dan Javascript paling dasar, sistem code splitting, koneksi ke database, control flow, OOP, nah sekarang kita mau mempraktekan pengetahuan tersebut ke dunia nyata, ini hanya contoh semata jadi bisa di kembangkan dengan kreatifitas anda.

Sebenarnya cara penyelesaian permasalahan sistem login dan registrasi ada banyak jadi ini hanya salah satu contoh sistem login/registrasi di dunia nyata, kita akan menggunakan sistem soal cerita lalu dipecah menjadi list permasalahan.

Permasalahan

Misal kita mempunyai website yang membutuhkan setiap user untuk login, sebelum login si user harus dapat melakukan registrasi terlebih dahulu, dan setelah login si user dapat menjelajahi semua halaman website kita tanpa halangan (area user yang sudah login).

Oke nih kita sudah permasalahan utama kita bisa pecah permasalahan ini menjadi kecil-kecil

1. Dapat Login dan Registrasi
2. Ada halaman yang hanya dapat diakses oleh user yang sudah login

Nah setelah sudah kita sederhanakan permasalahannya kita bisa pecah lagi nih permasalahan jadi lebih kecil

1. Dapat Login dan Registrasi
 1. Melakukan koneksi ke Database.
 2. Melakukan *Data Fetching* dari Database untuk melakukan pencocokan password (Login).
 3. Melakukan pengiriman data ke Database untuk disimpan (Registrasi).
 4. Kalau bisa login/registrasi kenapa tidak bisa logout.
2. Ada halaman yang hanya dapat diakses oleh user yang sudah login.
 1. Saat login/registrasi user yang sukses otomatis melakukan penyimpanan suatu data kunci sebagai patokan bahwa user tersebut telah login dengan menggunakan sistem cookie/session.
 2. Ketika ada halaman yang hanya bisa dibuka oleh user yang sudah login kita tinggal mengambil data dari cookie/session dan lalu melakukan sedikit pencocokan apakah user memang benar-benar sudah login.

Karena kita ingin sederhana jadi kita akan menggunakan session untuk menyimpan data user.

Oke nih jadi soal cerita tadi ternyata ada banyak permasalahan-nya, tapi ini masih di bagian sederhana, normalnya ketika kita melakukan registrasi ada operasi yang bernama *hashing*, *hashing* ini merupakan suatu cara untuk melakukan “enkripsi” terhadap data, dalam permasalahan ini *hashing* akan digunakan ketika user melakukan registrasi yang nantinya akan disimpan di database, kenapa? Agar lebih aman (kalau di lomba melakukan fitur hashing untuk password dapat nilai plus).

Sebelum kita membuat, kita harus membuat *table schema*-nya dulu

```
CREATE TABLE IF NOT EXISTS users (  
  userId INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  username VARCHAR(255) UNIQUE,  
  password VARCHAR(255)  
);
```

Oke jangan panik, ini merupakan sql query untuk membuat tabel users (kalau phpmyadmin tinggal buka tab query lalu copy ini query lalu di paste ke phpmyadmin lalu klik go).

Query diatas itu maksud-nya membuat table users jika table-nya tidak ada dengan kolom userId yang bertipe INT (integer), NOT NULL (tidak boleh null/kosong ketika nambah data ke database), AUTO_INCREMENT itu bermaksud kita tidak perlu mengisi kolom userId ketika kita nambah data karena mysql secara otomatis akan menambahkan angka-nya, lalu PRIMARY KEY yang bermaksud kunci utama yang membedakan antara data yang lain. Kolom selanjutnya adalah username yang bertipe VARCHAR yang panjang maksimal 255 yang unik. Lalu dikolom terakhir password bertipe VARCHAR yang panjang maksimal 255.

Note : Kebanyakan query ataupun bahasa pemrograman itu menggunakan bahasa inggris jadi dilatih ya bahasa inggris-nya.

Kalau masih tidak paham ini bentuk yang sama dengan di phpmyadmin.

The screenshot shows the 'Structure' tab in phpMyAdmin for a table named 'users'. The table has three columns: 'userId' (INT, PRIMARY KEY, AUTO_INCREMENT), 'username' (VARCHAR(255), UNIQUE), and 'password' (VARCHAR(255)). The interface includes a top navigation bar with tabs like Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, Tracking, Designer, and More. Below the navigation bar, there's a 'Table name' field with 'users' and a 'Go' button. The main area shows the column definitions with various options like 'Pick from Central Columns', 'Type', 'Length/Values', 'Default', 'Collation', 'Attributes', 'Null', 'Index', 'A_I', and 'Comments'. At the bottom, there are fields for 'Table comments:', 'Collation:', and 'Storage Engine:'.

Nah setelah table-nya ada semua kita baru bisa mulai ngoding, kita akan mempraktekan apa yang sudah kita pelajari kemaren.

1. Membuat file db.php untuk meng-handle semua hal yang berhubungan dengan database

```
1 <?php
2 // biasanya username-nya itu root lalu password-nya kosong
3 $connection = new PDO('mysql:host=localhost;dbname=ngoding', 'username-nya', 'password-nya');
4
```

Nah setelah kita buat variabel connection, ini nanti ketika kita require kita bisa pakai variabel tersebut untuk melakukan komunikasi dengan database, kalau kemarin sudah menginstall xampp berarti tinggal ketik username-nya root lalu password-nya kosong.

2. Membuat file navigation.php untuk menyimpan semua hal tentang navbar/navigasi

```
1 <!-- File ini digunakan seperti komponen di framework besar agar tidak melakukan pengulangan kode -->
2 <div>
3     <a href='./home.php'>Home</a>
4     <!-- Check apakah user sudah login, kalau belum di beri tombol login sama register -->
5     <!-- kalo sudah dikasih tombol logout -->
6     <?php if (!isset($_SESSION['user'])) : ?>
7         <a href='./login.php'>Login</a>
8         <a href='./register.php'>Register</a>
9     <?php else : ?>
10         <span> User : <?= $_SESSION['user']['username'] ?></span>
11         <a href='./dashboard.php'>Dashboard</a>
12         <a href='./logout.php'>Logout</a>
13     <?php endif; ?>
14 </div>
```

Kenapa kita buat file ini karena kita ingin mengurangi duplikasi kode nanti-nya, terus kenapa kita ngecek variabel `$_SESSION['user']`, nah variabel ini nanti kita pakai untuk menyimpan data user, perintah `!isset($_SESSION['user'])` itu bermaksud kalau variabel ini tidak ada jalankan kode dibawah ini kalau ada jalankan kode yang di sebelah else, di file ini tidak ada `session_start()` karena file ini hanya digunakan di perintah include/require

3. Membuat file `home.php` / `index.php` (Halaman utama website), file ini akan men include `navigation.php`.

```
1  <?php
2  session_start();
3  ?>
4  <!DOCTYPE html>
5  <html lang="en">
6
7  <head>
8      <meta charset="UTF-8">
9      <meta http-equiv="X-UA-Compatible" content="IE=edge">
10     <meta name="viewport" content="width=device-width, initial-scale=1.0">
11     <title>Home</title>
12 </head>
13
14 <body>
15     <h1>Home</h1>
16     <?php require './navigation.php' ?>
17 </body>
18
19 </html>
```

Nah kok ada `session_start()`; karena kita mau ada sistem login/register kita mengimplementasikan dengan menggunakan session (lihat di permasalahan tadi) terus elemen meta itu buat apa, nah itu tidak terlalu penting, yang harus tau itu hanya yang viewport ini akan penting ketika kita mulai membuat website untuk hp/tablet.

4. Membuat file register.php, karena kalau belum punya user cara-nya login gimana
Pertama kita buat html-nya dulu kenapa? Karena lebih mudah me-referensi sesuatu yang sudah ada daripada ber-imajinasi

```
1 <?php
2 // Inisialisasikan session-nya (Agar bisa menggunakan fitur session)
3 session_start();
4
5 require './db.php';
6
7 ?>
8
9 <!DOCTYPE html>
10 <html lang="en">
11
12 <head>
13     <meta charset="UTF-8">
14     <meta http-equiv="X-UA-Compatible" content="IE=edge">
15     <meta name="viewport" content="width=device-width, initial-scale=1.0">
16     <title>Register</title>
17 </head>
18
19 <body>
20     <h1>Register</h1>
21
22     <!-- Menggunakan isi file navigation.php -->
23     <?php require './navigation.php' ?>
24
25     <form action="" method="post">
26         <input type="text" name="username" placeholder="username">
27         <input type="password" name="password" placeholder="password">
28         <input type="submit" value="Register" name="register">
29     </form>
30 </body>
31
32 </html>
```

Nah di file ini kita juga men-require navigation.php dan juga db.php, karena file ini akan berkomunikasi dengan database untuk melakukan registrasi user.

Nah sekarang baru bisa buat php-nya, kode-nya kita taruh dibawah-nya require “./db.php”

```

1  <?php
2  // Inisialisasikan session-nya (Agar bisa menggunakan fitur session)
3  session_start();
4
5  require './db.php';
6
7  // Check apakah user sudah pernah login/registrasi, kalau sudah pernah
8  // Pindah ke dashboard.php
9  if (isset($_SESSION['username']))
10     header('Location: ./dashboard.php');

```

Oke jangan panik, saya akan jelaskan sedikit demi sedikit dibagian bawah require itu merupakan *guard*, guard ini digunakan untuk menjaga ketika apakah user sudah pernah login dengan mengecek \$ *SESSION*['user'].

```

1  // Cek apakah sudah diklik register
2  if (isset($_POST['register'])) {
3      // Kalau diklik buat query dengan menggunakan koneksi dari db.php
4      $query = $connection->prepare(
5          "INSERT INTO users (username, password) VALUES (:username, :password)"
6      );
7
8      // Melakukan hashing password agar lebih aman
9      $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
10
11     // Di masukan value-nya
12
13     $query->bindParam('username', $_POST['username'], PDO::PARAM_STR);
14     $query->bindParam('password', $password, PDO::PARAM_STR);
15
16     // Eksekusi query tersebut lalu cek apakah sukses atau tidak
17     if ($query->execute()) {
18         // Kalau sukses otomatis pindah ke dashboard.php dan disave kredensi user ke session
19         // Sebelum di save ambil data user yang baru di registrasi dan diambil id-nya
20         $query = $connection->prepare("SELECT * FROM users WHERE username=:username");
21         $query->bindParam('username', $_POST['username'], PDO::PARAM_STR);
22         $query->execute();
23         $user = $query->fetch();
24
25         $_SESSION['user'] = [
26             'id' => $user['userId'],
27             'username' => $_POST['username'],
28         ];
29
30         header('Location: ./dashboard.php');
31     } else {
32         // Kalau gagal tinggal di echo agar si user tau kalau gagal
33         echo "Registrasi gagal!";
34     }
35 }
36
37 ?>

```

Selanjut-nya seperti biasa mengecek apakah tombol submit sudah di klik atau belum kalau sudah melakukan query ke database untuk disimpan dan di server nge-set session user yang tadi baru login, kalau sudah pindah ke dashboard.php, di setiap potongan kode terdapat komentar untuk referensi lebih dalam.

5. File logout.php kalo bisa registrasi kenapa tidak bisa logout.

A screenshot of a code editor with a dark background and light-colored text. The code is a PHP script for a logout function. It starts with a PHP opening tag, followed by a comment and a session_start() call. Then, it checks if the 'user' session variable is set, and if so, it unsets it. Finally, it sets a header to redirect to a local file named 'home.php'. The code is numbered from 1 to 12.

```
1 <?php
2
3 // Jangan lupa session harus di start dulu
4 session_start();
5
6 // Melakukan penghapusan kredensi user di session
7 if (isset($_SESSION['user']))
8     unset($_SESSION['user']);
9
10 // Lalu pindah ke halaman localhost/nama-folder-mu/home.php
11 header('Location: ./home.php');
12
```

Kode untuk logout sangat simpel kita tinggal ngecek apakah user sudah login atau belum kalau sudah variabel `$_SESSION['user']` di kosongkan/hapus, lalu pindah ke <http://localhost/nama-folder/home.php>.

6. Membuat file dashboard.php

```

1  <?php
2  // Inisialisasikan session-nya (Agar bisa menggunakan fitur session)
3  session_start();
4
5  // Check apakah user sudah pernah login/registrasi, kalau belum
6  // Pindah ke login.php
7  if (!isset($_SESSION['user']))
8      header('Location: ./dashboard.php');
9
10 ?>
11
12 <!DOCTYPE html>
13 <html lang="en">
14
15 <head>
16     <meta charset="UTF-8">
17     <meta http-equiv="X-UA-Compatible" content="IE=edge">
18     <meta name="viewport" content="width=device-width, initial-scale=1.0">
19     <title>Dashboard</title>
20 </head>
21
22 <body>
23     <h1>Dashboard</h1>
24     <?php require './navigation.php' ?>
25     Hai <?= $_SESSION['user']['username'] ?>!
26 </body>
27
28 </html>

```

Membuat file dashboard.php hampir sama dengan register.php di bagian *guard*-nya bedanya kebalikan-nya (cara membedakan-nya dengan tanda !).

7. File login.php

Pertama-tama seperti biasa kita buat html-nya dulu buat referensi nanti pas kita mulai ngoding php-nya


```

1  <?php
2  // Inisialisasikan session-nya (Agar bisa menggunakan fitur session)
3  session_start();
4
5  require './db.php';
6
7  // Check apakah user sudah pernah login/registrasi, kalau sudah pernah
8  // Pindah ke dashboard.php
9  if (isset($_SESSION['user']))
10     header('Location: ./dashboard.php');
11
12  ?>
13
14  <!DOCTYPE html>
15  <html lang="en">
16
17  <head>
18     <meta charset="UTF-8">
19     <meta http-equiv="X-UA-Compatible" content="IE=edge">
20     <meta name="viewport" content="width=device-width, initial-scale=1.0">
21     <title>Login</title>
22  </head>
23
24  <body>
25     <h1>Login</h1>
26
27     <!-- Menggunakan isi file navigation.php -->
28     <?php require './navigation.php' ?>
29
30     <form action="" method="post">
31         <input type="text" name="username" placeholder="username">
32         <input type="password" name="password" placeholder="password">
33         <input type="submit" value="Login" name="login">
34     </form>
35  </body>
36
37  </html>

```

Hampir mirip dengan register.php cuman nanti kita tidak melakukan query penyimpanan ke database tetapi kita akan mengambil data ke database lalu melakukan pencocokan dengan data yang dikirim oleh user.

```

1  if (isset($_SESSION['user']))
2      header('Location: ./dashboard.php');
3
4  if (isset($_POST['login'])) {
5      // Ambil data user dengan menggunakan data yang disubmit
6      $query = $connection->prepare('SELECT * FROM users WHERE username=:username');
7
8      // Di bind data-nya
9      $query->bindParam('username', $_POST['username'], PDO::PARAM_STR);
10
11     // Di eksekusi query-nya lalu di fetch data-nya
12     $query->execute();
13     $user = $query->fetch();
14
15     // Check apakah data yang di fetch ada atau tidak
16     if ($user) {
17         // Kalo user-nya ada
18
19         // Verifikasi password
20         if (password_verify($_POST['password'], $user['password'])) {
21             // Kalau benar, masukan kredensi user ke session lalu pindah ke dashboard
22             $_SESSION['user'] = [
23                 'id' => $user['userId'],
24                 'username' => $_POST['username'],
25             ];
26             header('Location: ./dashboard.php');
27         } else {
28             // Kalau salah
29             echo "Password salah!";
30         }
31     } else {
32         // kalo user-nya tidak ada
33         echo "Username salah!";
34     }
35 }
36 ?>

```

Nah seperti biasa kita ngecek apakah tombol yang kita sediakan di form tadi diklik submit belum kalau diklik kode yang ada di `if($_SESSION['login'])` akan ke-trigger, di bagian pertama kita buat query untuk mengambil data user yang memiliki `username=:username` kenapa kok `:username` karena kita menggunakan fitur PDO (agar query lebih aman), karena dengan fitur ini kita harus bind value-nya ke query tadi dengan `→bindParam` setelah di bind di execute lalu difetch karena kita ngambil data setelah itu kita kasih if statement lagi untuk ngecek apakah data-nya memang ada lalu kalau ada melakukan pencocokan password dengan function `password_verify(password, versi-hash-nya)` function ini return boolean (true/false) yang dapat dipakai di if else dengan mudah, kalau `password_verify`-nya sukses/true kita langsung save kredensi user ke session kalau gagal/false kita echo password salah.

Note : Contoh ini bisa digunakan untuk tugas yang membuat website blog


Note : keyword ! Didalam suatu kode bisa bermaksud kebalikan-nya kalau misal

!true berarti ini false

!!true berarti true

Jangan kebalik ya

Note : Return di function itu bermaksud



```
1 function namafungsi()
2 {
3     return 'sesuatu';
4 }
5
6 $data = namafungsi();// $data akan menyimpan 'sesuatu'
```

Note : Jangan lupa kalau di pemrograman memiliki sistem tipe data

1. Integer → Angka yang tidak memiliki coma.
2. Float → Angka yang memiliki coma (0,2).
3. Double → Sama dengan Float tetapi lebih akurat.
4. Char → Sebuah huruf.
5. String → Kumpulan Char/Huruf (kalau di bahasa pemrograman yang ngarah ke low level tidak ada kata string tetapi array of string).
6. Boolean → True / False.
7. Array → Suatu block memory yang dapat menyimpan value banyak.
8. Object → Ini biasanya dengan konsep OOP dengan (apa apa yang dimulai dengan *new*, kalo diatas tadi *new PDO*).

Note : Untuk kode keseluruhan ada di github jadi kalau misal ada yang pengen liat tinggal kirim username github temen-temen ke saya.