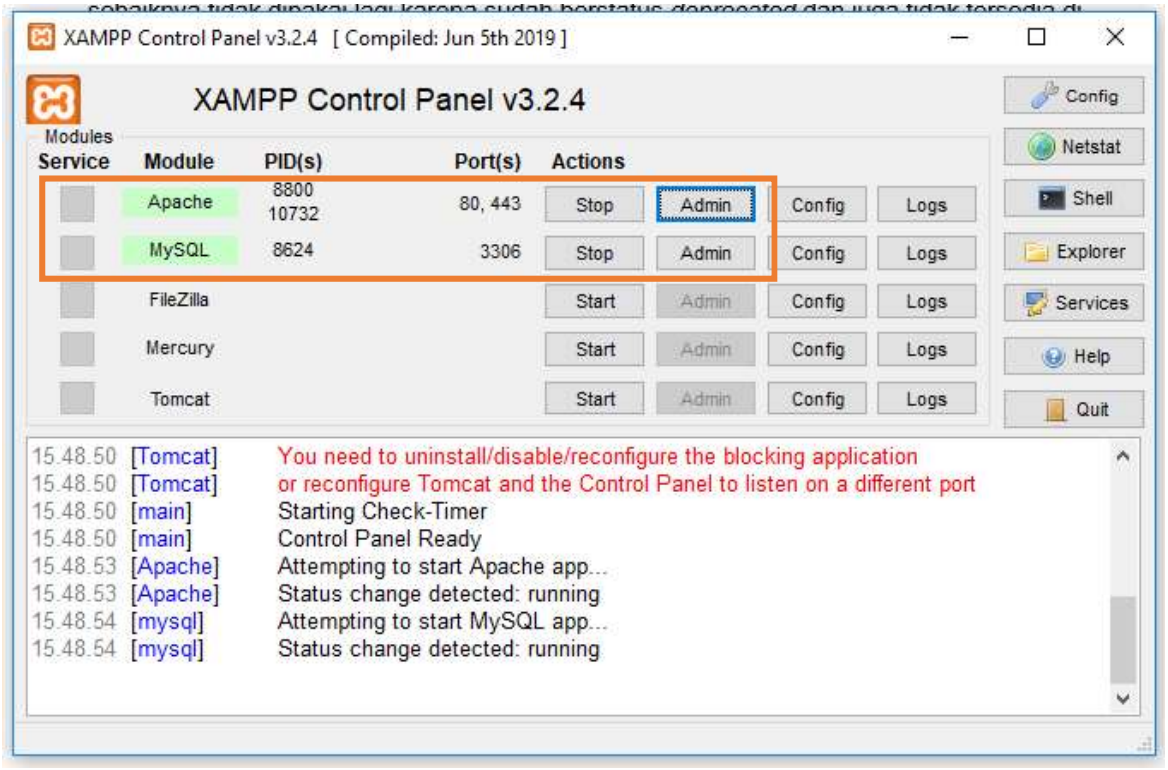
Konsep Database OOP 1

Mysqli Object

Pemrosesan database menjadi salah satu fokus utama ketika membuat aplikasi web. Dari sekian banyak pilihan, MySQL masih menjadi aplikasi database yang paling banyak di pakai dengan PHP. PHP menyediakan 3 cara untuk berkomunikasi dengan MySQL, yakni melalui **mysql extension**, **mysqli extension** serta **PDO extension**. Untuk *mysql extension* sebaiknya tidak dipakai lagi karena sudah berstatus *deprecated* dan juga tidak tersedia di PHP versi 7 ke atas. Kita disarankan untuk menggunakan ***mysqli extension*** **dan** ***PDO***. **Mysqli extension** hadir dalam 2 "varian rasa", yakni penulisan prosedural menggunakan function, serta penulisan dengan konsep pemrograman object. Pada materi OOP kita akan fokus membahas cara penggunaan *mysqli extension* menggunakan konsep pemrograman object ini (OOP).

*k*ita akan membahas 3 diantaranya, yakni **mysqli** class, **mysqli_result** class, dan **mysqli_stmt** class.
Secara singkat, **mysqli** class digunakan untuk membuat koneksi dengan database MySQL. Kemudian **mysqli_result** class dipakai untuk memproses hasil yang didapat dari query `SELECT` (menampilkan tabel). Serta **mysqli_stmt** class dipakai untuk memproses *prepared statement*.
Setiap class ini memiliki berbagai property dan method yang akan kita bahas secara bertahap.

Sebelum mulai praktek, pastikan **MySQL Server** sudah aktif. Caranya, klik tombol **start** di kolom MySQL pada XAMPP Control Panel:



Mysqli Class

Mysqli class adalah class yang dipakai untuk mengelola koneksi antara PHP dengan MySQL database server. Dalam mysqli versi prosedural, koneksi ini dibuat menggunakan function `mysqli_connect()`.
Agar bisa dipakai, **mysqli class** harus di instansiasi menjadi **mysqli object**. Proses instansiasi ini sama seperti pembuatan object biasa, yakni menggunakan perintah **new**:
`$nama_variabel = new mysqli([argument_1], [argument_2], [argument_3], ...)`

Penting!!!

```
$nama_variabel = new mysqli([$host], [$username], [$password], [$dbname], [$port], [$socket])
```

Penjelasan argument:

- `$host`: Diisi dengan alamat server atau IP address tempat MySQL server berada. Karena kita menjalankan MySQL Server di komputer yang sama dengan Web Server Apache, maka bisa diisi dengan "`localhost`" atau "`127.0.0.1`".
- `$username`: Diisi dengan nama user MySQL Server, misalnya "`root`".
- `$password`: Diisi dengan password dari user MySQL yang ada ditulis pada argument kedua.
- `$dbname`: Diisi dengan database yang ingin dipakai.
- `$port`: Diisi dengan nama port MySQL yang digunakan. Secara default, MySQL menggunakan port `3306`.
- `$socket`: Diisi dengan alamat file yang mengatur `socket`, yakni mekanisme cara komunikasi antara web server dengan database server.

Dari ke-6 argument ini, umumnya kita hanya **perlu mengisi 3 atau 4 argument saja**, yakni `$host`, `$username`, `$password` dan `$dbname`.

Ok kita akan mulai dengan membuat mysqli object, perhatikan program berikut, simpan pada folder **pertemuan8** dengan nama **mysqli1.php**

```php
<?php
$mysqli = new mysqli("localhost", "root", "");
echo "<pre>";
print_r($mysqli);
echo "</pre>";

baris kedua program
```

```php
$mysqli = new mysqli("localhost", "root", "");
```

Merupakan proses pembuatan object dari class mysqli dengan nama object **$mysqli (tidak harus nama ini ya, boleh pakai nama object yang lain).**

Lah kan kita belum pernah membuat class mysqli kok tiba tiba bisa bikin object nya??? Ada yang nanya seperti ini???

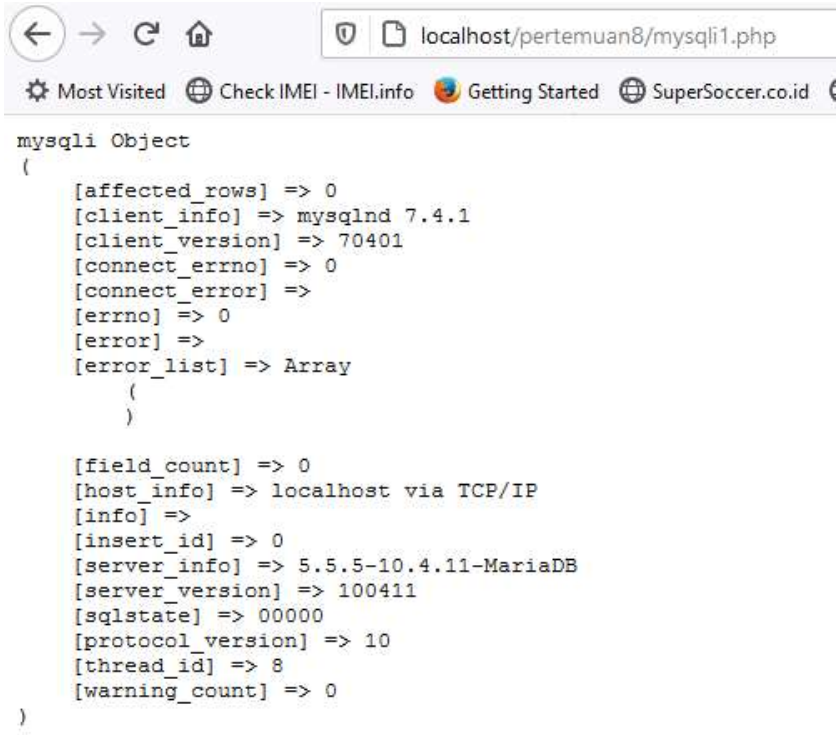
Ya karena class mysqli memang sudah dibuat oleh PHP dan kita tinggal menggunakan dan memahami bagaimana class tersebut digunakan. Paham ya. Jadi class nya sudah ada.

Sekarang perhatikan bagian argument pada saat kita membuat object mysqli, di sana terlihat kita mengisikan 3 buah argument yaitu localhost untuk $host, root untuk $username dan “” kosong untuk $password. Ketiga argument ini sudah cukup untuk membuat sebuah object dari class mysqli.

Localhost, root dan “” merupakan 3 argument standar yang kita isikan untuk melakukan koneksi ke database mysql, apabila pada kondisi tertentu kita sudah merubah host username dan password dari mysql maka ketiga nilai dri argument tersebut juga harus kita sesuaikan.

Silakan coba jalankan program diatas dan lihat hasilnya

Ingat fungsi print_r() bisa digunakan untuk menampilkan detail suatu object.



Output diatas menampilkan detail dari object $mysqli yang memiliki banyak property di dalamnya, lihat dulu saja nanti kita pelajari masing2 propertynya. Nantinya nilai property ini akan berubah setiap kali perintah MySQL di jalankan.

Catatan penting:

Detail dari property object yang kalian lihat diatas ditampilkan dalam bentuk **ASSOSIATIVE ARRAY. (ingat istilah ini ya,penting akan sering digunakan)**

Ciri-ciri penulisan dari assosiative array adalah [nama_property]=>nilai_property

Klo dilihat dari property yang dimiliki oleh object artinya class induknya yaitu class mysqli juga memiliki property yang sama pada detail diatas.

Baca pelan-pelan penjelasan dibawah ini, kita akan lihat beberapa penjelasan dari property di atas:

- Property `affected_rows` di baris 2 berisi informasi mengenai jumlah baris tabel yang terdampak oleh sebuah perintah *query*. Saat ini nilainya 0 karena kita belum menjalankan perintah *query* apapun.
- Property `client_info` dan `client_version` di baris 3-4 menunjukkan versi MySQL Client yang dipakai oleh PHP.
- Property `connect_errno` dan `connect_error` di baris 7-8 berisi nomor dan pesan kesalahan. Keduanya akan berisi nilai jika koneksi ke MySQL server gagal di proses. Dalam contoh di atas isi property ini bernilai 0 dan string kosong "" karena koneksi tidak mengalami masalah.
- Property `server_info` di baris 19 berisi informasi mengenai versi MySQL Server yang sedang terhubung. Karena alasan tertentu, MariaDB memberikan kode awal 5.5.5 sebelum versi server yang sebenarnya, yakni **MariaDB 10.1.34**.

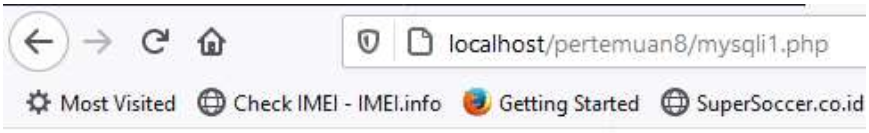
Ok seperti yang telah kita pelajari sebelumnya, tentunya kita bisa mengakses nilai dari masing-masing property dari object tersebut.

Berikut contoh mengakses beberapa property dari object mysqli, simpan dengan nama mysqli2.php

```php
<?php
$mysqli = new mysqli("localhost", "root", "");

echo $mysqli->affected_rows;  echo "<br>";
echo $mysqli->client_info;    echo "<br>";
echo $mysqli->connect_errno;  echo "<br>";
echo $mysqli->server_info;    echo "<br>";
```

baris program diatas kita ingin menampilkan nilai dari beberapa property yang kita inginkan. Silakan jalankan programnya dan lihat hasilnya.



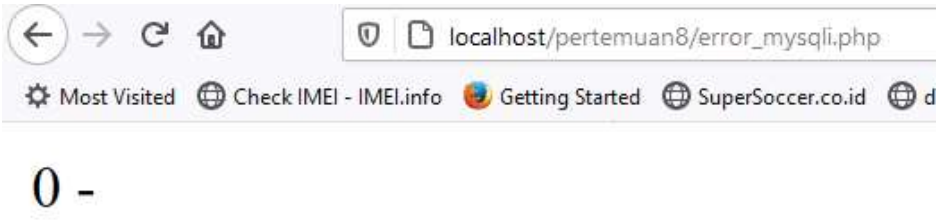
0
mysqlnd 7.4.1
0
5.5.5-10.4.11-MariaDB

Penanganan Error mysqli Object

Penanganan error merupakan hal yang sangat penting dalam pembuatan sebuah program, dengan adanya error handling kita akan bisa lebih cepat dan akurat dalam mengetahui error yang terjadi dalam pembuatan program.
Ok sekarang kita akan melihat pemanfaatan error handling yang bisa dilihat dari object mysqli, simpan program berikut dengan nama **error_mysqli.php**

```php
<?php
$mysqli = new mysqli("localhost", "root", "");
echo $mysqli->connect_errno," - ", $mysqli->connect_error;
```

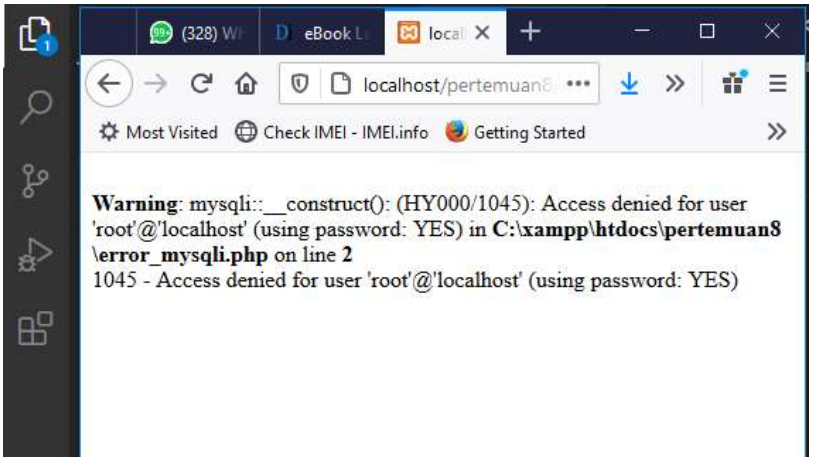
program di atas akan mengakses 2 property yang terkait dengan penanganan error yaitu [connect_errno] dan [connect_error], hasilnya



Hasilnya terlihat tidak ada error yang ditangkap oleh property karena memang kita telah melakukan konfigurasi koneksi dengan benar, nah sekarang kita akan ubah sedikit programnya dengan menambah nilai pada argument password. Pada kondisi default password dari mysql server adalah null atau kosong.

```php
<?php
$mysqli = new mysqli("localhost", "root", "x");
echo $mysqli->connect_errno," - ", $mysqli->connect_error;
```

program di atas kita tambahkan nilai “x” pada argument password. Dan seharusnya ini akan menjadi error.



Disini terlihat dengan jelas error muncul, pertama pesan **warning** merupakan bawaan dari php

Sedangkan Baris terakhir berasal dari kode

```
echo $mysqli->connect_errno," - ", $mysqli->connect_error;
```

1045 - Access denied for user…

Ini merupakan nilai dari property yang ditangkap oleh object mysqli, silakan coba pahami dulu.
Error terjadi karena kita menambahkan password saat ingein mencoba koneksi dengan database server, sedangkan secara default password tidak ada.

Dalam situasi tertentu, bisa saja PHP tidak menampilkan pesan error bawaan. Ini umum terjadi di lingkungan web hosting atas alasan keamanan (web yang sudah online).

Teknik yang sering dipakai adalah menempatkan isi property `connect_errno` atau `connect_error` ke dalam sebuah kondisi **if**.
Jika tidak ada masalah (koneksi berhasil dilakukan), property `connect_errno` akan berisi nilai 0 dan property `connect_error` akan berisi string kosong. Namun jika ternyata koneksi bermasalah, kedua property ini akan berisi "sesuatu":

nah sekarang kita akan coba menambahkan programnya agar lebih baik dalam menangkap error, simpan dengan nama **error_mysqli2.php**

```php
<?php
$mysqli = new mysqli("localhost", "root", "x");

if ($mysqli->connect_error) {
  die('Koneksi bermasalah (' . $mysqli->connect_errno . ') '
          . $mysqli->connect_error);
}

echo "Jalankan query MySQL...";
```

program diatas akan menampilkan pesan error yang lebih mudah dipahami oleh user.
Pahami penjelasan berikut.

terdapat pemeriksaan kondisi, yakni **`if ($mysqli->connect_error)`**. Kondisi ini akan bernilai **true** jika property `$mysqli->connect_error` berisi suatu string atau berarti ada error yang ditangkap, namun akan bernilai **false** jika property `$mysqli->connect_error` berisi string kosong atau tidak terjadi error.
Di dalam blok kondisi **if** terdapat fungsi `die()`. Gunanya adalah, jika ternyata koneksi ke MySQL bermasalah, tampilkan pesan error dan proses cukup berhenti sampai di sini.

Jadi inti dari program diatas adalah apabila terjadi error dalam proses koneksi ke database server maka akan ditampilkan errornya, apabila tidak ada error dan koneksi sukses maka akan muncul pesan jalankan **query MySQL…** pada browser. Silakan coba kondisi salah dan kondisi benarnya.

Exception

Exception merupakan salah satu fitur standar dari *object oriented programming*. Hampir semua bahasa pemrograman yang menerapkan prinsip OOP juga memiliki **exception** (tidak hanya PHP saja).
Exception adalah sebuah mekanisme untuk mengelola error secara lebih "elegan". Dalam prakteknya nanti, PHP juga menyediakan class yang bernama **Exception**. Karena dipakai untuk mengelola error, istilah exception ini sering juga disebut sebagai *exception handling*, yakni cara penanganan *exception*.

Pada bagian ini kita akan coba menerapkan exeption untuk program yang sebelumnya sudah kita buat, dan akan kita gunakan untuk latihan-latihan yang akan kita lakukan selanjutnya. Jadi harus benar-benar kita pahami.

Berikut konsep dasar penulisan exeption:

```
try {
// kode program yang akan menghasilkan exception/error
throw new Exception ("pesan kesalahan")
}
catch (Exception $e) {
// kode program yang akan memproses exception/error
}
```

Struktur **try - catch** ini mirip seperti struktur **if else**. Pertama, PHP akan mencoba memproses seluruh kode program yang ada di dalam block **try**. Apabila tidak ada perintah yang menghasilkan exception, maka block kode **catch** akan dilewati dan PHP lanjut memproses perintah setelah blok **catch**.

Namun jika terdapat perintah yang menghasilkan exception, maka PHP akan pindah ke bagian **catch** untuk "menangkap" exception tersebut. Jika tidak ada kode program yang memproses exception (tidak ada block kode **catch**), akan tampil "*Fatal error: Uncaught Exception*" seperti contoh kita sebelumnya.

Di awal perintah **catch**, terdapat baris `(Exception $e)`. Baris ini berfungsi mirip seperti argument di dalam sebuah function. Sepanjang block **catch**, variabel `$e` bisa dipakai untuk mengakses exception object, yakni object hasil exception yang "dilempar" dari block **try**.

Nama variabel `$e` ini boleh bebas, tidak harus ditulis sebagai `$e`, tapi bisa juga nama lain seperti `$ex`, `$exception`, atau `$objectException`. Aturan penamaan variabel ini sama seperti argument biasa pada function.

Ok untuk lebih memahami konsep exeption kita akan buat contoh sederhana, simpan dengan nama **exeption1.php**

```php
<?php
function bagi($a){
  if ($a === 0)
    throw new Exception("Argument \$a tidak bisa diisi angka 0");
  else
    return 1/$a;
}

try {
  echo bagi(0);
}
catch (Exception $e) {
  echo $e->getMessage();
}
```

Inti dari program di atas membuat sebuah fungsi dengan sebuah argument untuk melakukan proses pembagian. Pada prosesnya dilakukan proses validasi terlebih dahulu dengan validasi **if else** bila argument yang dikirim sama dengan 0 maka **exeption** akan mengirimkan pesan error yang nanti akan ditangkap oleh fungsi **catch untuk ditampilkan.**

Perbedaan yang paling terlihat dari program di atas adalah kita menambahkan struktur try-catch. Perhatikan alur programnya, yang dilakukan pertama saat program dijalankan adalah bagian **try.** Di sana, fungsi bagi dipanggil dengan mengirimkan argument 0. Karena argument yang dikirim sama dengan 0 maka nilai if adalah true dan exeption dikirimkan. Pesan exeption akan ditangkap oleh fungsi catch dan disimpan dalam object $e. salah satu lalu object $e memanggil fungsi **getMessage**() untuk menampilkan pesan yang dikirim oleh exeption. Kira kira begitu alurnya. Silakan pahami pelan-pelan.

Nah untuk pengetahuan sementara, selain fungsi getMessage, ada beberapa fungsi lain yang bisa dipanggil

method dari exception object:

1. Method `$e->getMessage()` dipakai untuk menampilkan pesan error yang ditulis sebagai argument pertama pada saat pembuatan exception. Ini sudah kita bahas sebelumnya.
2. Method `$e->getCode()` dipakai untuk menampilkan kode error yang ditulis sebagai argument kedua pada saat pembuatan exception. Di baris 4, argument kedua (setelah pesan error) berupa angka `99`, dengan demikian method `getCode()` juga akan menampilkan angka `99`. Kode error ini bisa di pakai untuk pemrosesan lebih lanjut. Misalnya jika kode error yang dihasilkan sama, berarti error yang terjadi juga berjenis sama.
3. Method `$e->getFile()` dipakai untuk menampilkan alamat path serta nama file tempat exception terjadi.
4. Method `$e->getLine()` dipakai untuk menampilkan baris tempat exception "dilempar".

5. Method `$e->getTraceAsString()` dipakai untuk menampilkan **trace error** dalam bentuk string. Trace error sendiri berisi penjelasan detail tentang lokasi error yang akan dibahas dalam method `getTrace()`.
6. Method `$e->getTrace()` dipakai untuk menampilkan trace error dalam bentuk array multi-dimensi, yakni ada array di dalam array di dalam array. Tampilan seperti ini diperlukan karena bisa saja sumber error tersebut "terpendam" di dalam kode program yang saling memanggil

silakan dicoba masing-masing fungsi diatas untuk gambaran apa yang akan dihasilkan dari fungsi tersebut.

```
  echo $e->getMessage()       ."<br>";
  echo $e->getCode()          ."<br>";
  echo $e->getFile()          ."<br>";
  echo $e->getLine()          ."<br>";
  echo $e->getTraceAsString() ."<br>";

  echo "<pre>";
  print_r( $e->getTrace() );
  echo "</pre>";
```

nah tadi intro untuk exeption, kita kembali lagi ke mysqli.

Sekarang kita akan menerapkan konsep try-catch untuk menangani error yang terjadi saat proses koneksi dengan merubah struktur program yang sebelumnya kita buat dengan validasi **if-else.** Simpan program berikut dengan nama **exeption2.php**

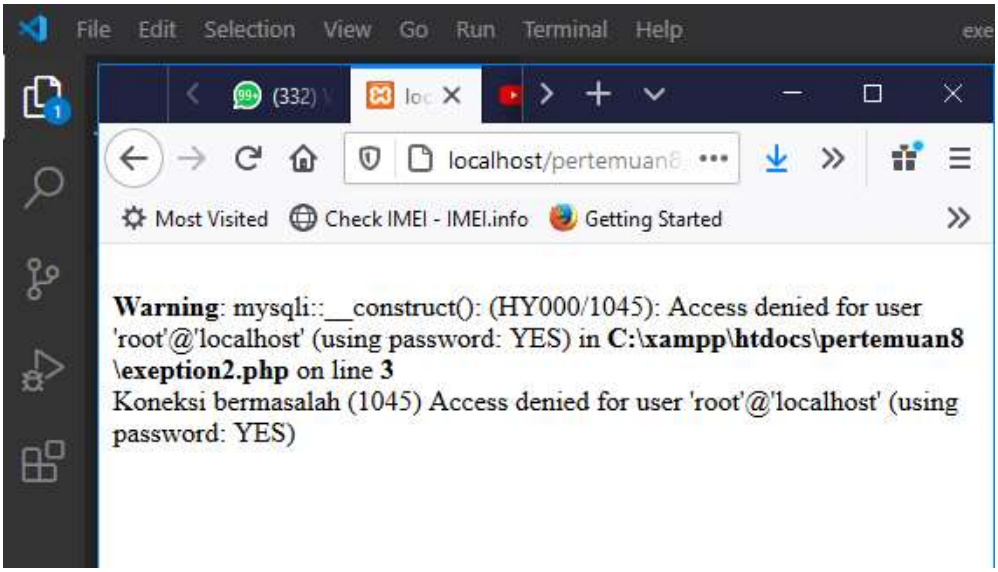
```
<?php
try {
  $mysqli = new mysqli("localhost", "root", "x");    1 buat object

  if ($mysqli->connect_error) {    2 dilakukan validasi
    throw new Exception('Koneksi bermasalah (' . $mysqli->connect_errno . ') '
            . $mysqli->connect_error);    3 bila terjadi error
  }
  echo "Jalankan query MySQL...";    Bila koneksi sukses
}
catch (Exception $e) {
  echo $e->getMessage();    4 error ditangkap dan pesan error ditampilkan
}
```

Kita pahami program diatas.
Pertama yang perlu **DIINGAT** adalah program pertama yang akan dilakukan adalah yang berada di dalam blok **try, yang mana???** Ya proses pembuatan object **$mysqli** dari class mysqli dengan mengirimkan argument host, username, dan password dari server yang digunakan.

Tahap kedua blok try akan melakukan validasi apakah terjadi error dalam proses koneksi atau tidak, apabila terjadi error maka error akan ditangkap oleh blok exeption dan pesan eror akan dilempar serta ditangkap oleh blok catch untuk dilakukan proses penampilan error apa yang terjadi pada bworser, begitu kura-kura. **PAHAM YA???**

Hasil program diatas adalah



Perhatikan baris akhir pesan errornya **koneksi bermasalah…**

Itu adalah output dari program diatas..

KOK MASIH MUNCUL WARNING BAWAAN DARI PHP !!!???

GA ENAK DILIHAT YA

Ok baca penjelasan dibawah untuk solusinya

Dalam contoh di atas kita membuat *exception* secara manual. Sebenarnya, **mysqli** object juga bisa di set agar menampilkan pesan error langsung dalam bentuk exception, namun belum aktif. Secara default, seluruh pesan error dari **mysqli** object tampil sebagai **Warning**. Untuk mengubah pesan error dari **mysqli** object menjadi *exception*, kita harus jalankan fungsi `mysqli_report(MYSQLI_REPORT_STRICT)` di awal kode program. Dengan adanya perintah ini, jika proses pembuatan mysqli object mengalami kendala, PHP akan melempar exception secara otomatis. Exception yang dilempar berasal dari class `mysqli_sql_exception`.

Nah begini solusi program agar output terlihat lebih bersih dan enak dilihat. Simpan dengan nama **exeption3.php,** cara ini akan kita gunakan untuk latihan-latihan selanjutnya jadi tolong pahami.

```php
<?php
mysqli_report(MYSQLI_REPORT_STRICT);

try {
  $mysqli = new mysqli("localhost", "root", "x");
  echo "Jalankan query MySQL...";
}
catch (mysqli_sql_exception $e) {
  echo "Koneksi bermasalah: ".$e->getMessage(). " (".$e->getCode().")";
}
```

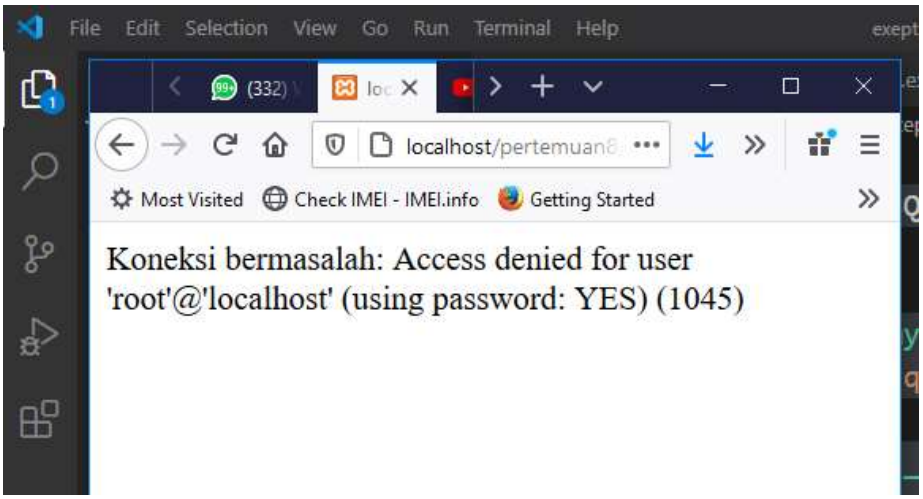
2 cek error yg terjadi dalam koneksi

1 buat object

3 bila koneksi berhasil

4 bila ada error yang ditangkap

Nah, dalam program diatas kita mengganti validasi if-else dengan fungsi bawaan **mysqli_report.** Pesan error yang tampil sekarang juga hanya berasal dari exception. Tidak ada lagi pesan error "`Warning: mysqli::__construct(): (HY000/1045): Access denied...`" seperti yang tampil jika kita memproses error **mysqli** object menggunakan fungsi `die()`. Lihat hasil program, warning sudah tidak ditampilkan.



Sekarang silakan ubah program dengan menghilangkan password yang sebelumnya kita isi, maka seharusnya akan menghasilkan output `Jalankan query MySQL...`

Ok sekarang kita akan menuju ke tahap selanjutnya, tapi sebelumnya pastikan kalian benar-benar memahami materi yang telah kita bahas diatas.

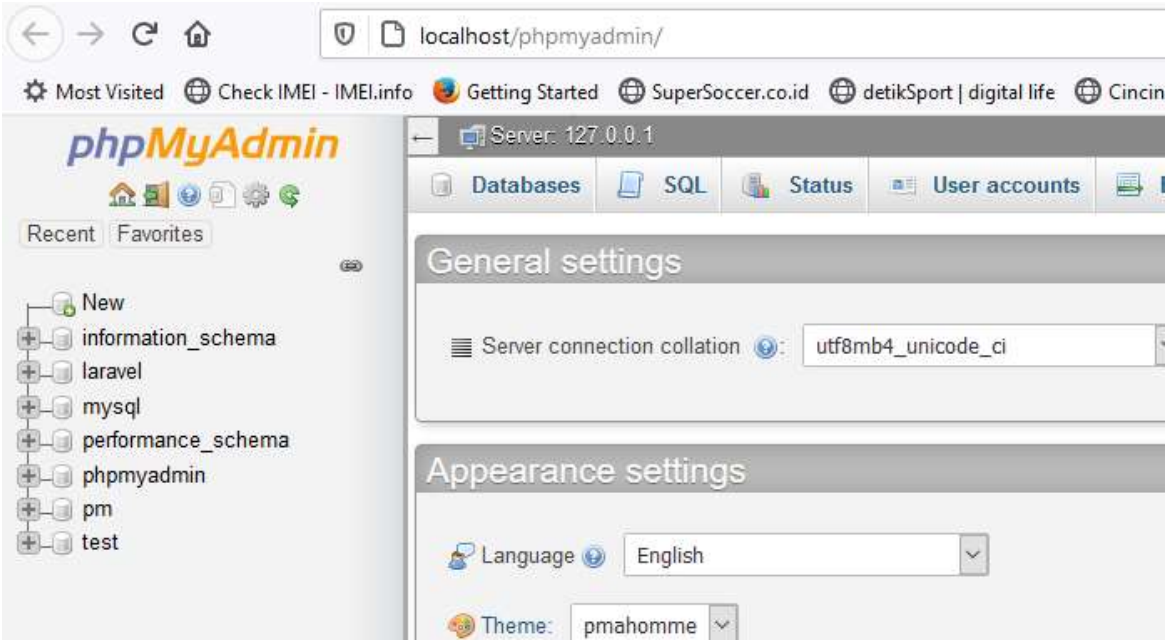
Membuat Database Dengan mysqli Object

OK, apabila proses koneksi antara PHP dengan database MySQL sudah aktif dengan program yang telah kita jalankan sebelumnya. Selanjutnya kita akan bahas cara menjalankan query MySQL menggunakan **mysqli** object.
Sebagai bahan praktek, kita ingin merancang sebuah mini project sederhana, yakni membuat database, tabel dan mengisinya secara langsung dari PHP. Proses pembuatan database dan tabel ini sebenarnya lebih sering dibuat dari **phpmyadmin, dan tentunya proses ini lebih mudah**. Namun pada praktik kali ini kita akan menulis kodenya secara

langsung di PHP, hal ini akan membawa keuntungan tersendiri. Nantinya kita memiliki sebuah file yang bisa dipakai untuk meng-*generate* database dan tabel secara otomatis. Pengetahuan ini sangat berguna dalam pembuatan aplikasi komersil. Katakanlah anda sudah selesai merancang aplikasi **Sistem Informasi Sekolah**. Cukup repot jika setiap kali menginstall aplikasi, databasenya harus dibuat dulu dari phpmyadmin. Akan lebih praktis jika kita menyediakan semacam file *booting* atau file *generate* yang ketika dijalankan, otomatis membuat semua database dan tabel. Proses ini memang butuh persiapan yang lebih sedikit lebih lama, tapi sangat praktis dalam jangka panjang.

Jadi intinya kita akan membuat file untuk proses untuk generate database beserta tabel di dalamnya secara manual tanpa melalui aplikasi server PHP myAdmin.

Sebelumnya kita cek dulu database server kita untuk memastikan sudah aktif. Ketik **localhost/phpmyadmin/** pada browser.



Tampilan diatas menunjukkan server database mysql yang akan kita gunakan sudah aktif, dan kita lihat pada deretan database belum ada nama database **ilkoom** yang akan kita buat.

Untuk praktek ini saya akan membuat kode program untuk meng-*generate* database **ilkoom** dan tabel **barang**. Kita akan buat database "**ilkoom**" terlebih dahulu, simpan dengan nama **create_DB.php**:

```php
<?php
mysqli_report(MYSQLI_REPORT_STRICT);   2 cek error apakah ada error koneksi

try {
  $mysqli = new mysqli("localhost", "root", "");   1 membuka koneksi

  // Buat database "ilkoom" (jika belum ada)
  $query = "CREATE DATABASE IF NOT EXISTS ilkoom";   3 buat query
  $mysqli->query($query);   4 jalankan query
  if ($mysqli->error){   5 cek error query yang dijalankan
    throw new Exception($mysqli->error, $mysqli->errno);
  }
  else {   6 bila query sukses muncul pesan
    echo "Database 'ilkoom' berhasil di buat / sudah tersedia <br>";
  };
}
catch (Exception $e) {
  echo "Koneksi / Query bermasalah: ".$e->getMessage(). " (".$e->getCode().")";
}
finally {
  if (isset($mysqli)) {
    $mysqli->close();
  }
}
```

Ok kita bahas program di atas. Pahami pelan-pelan!!!

```php
$mysqli = new mysqli("localhost", "root", "");
```

Program di atas adalah proses membuat object **$mysqli** dari class **mysqli**, sekaligus bertujuan membuka koneksi dengan database server, seperti yang sudah pernah kita lakukan pada materi sebelumnya. Apabila koneksi sukses maka program dilanjutkan dengan mumbuat query yang diinginkan, karena kita akan membuat database ilkoom maka query-nya

```php
$query = "CREATE DATABASE IF NOT EXISTS ilkoom";
```

$query (namanya boleh diganti bebas) merupakan sebuah variable yang digunakan untuk menampung query yang kita buat. If not exists adalah perintah untuk melakukan cek kedalam database apakah sudah ada atau belum, bila belum ada maka database akan dibuat, apabila sudah ada maka proses pembuatan database akan diabaikan.

Tahap selanjutnya adalah memanggil fungsi untuk menjalankan query yang telah disiapkan

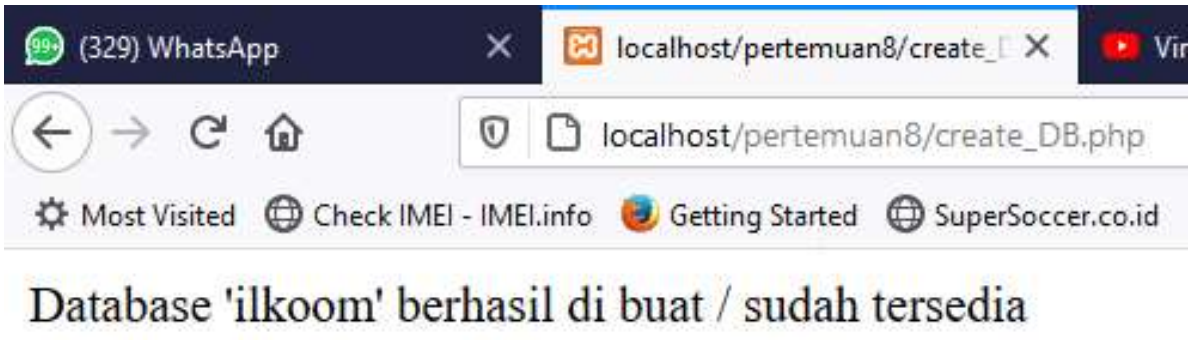
```php
$mysqli->query($query);
```

$mysqli merupakan object dari class **mysqli** memanggil method **query()** dengan menginputkan parameter **$query** yang telah kita siapkan sebelumnya. **PAHAM YA???**

Apabila query berhasil dijalankan tanpa error, maka akan ditampilkan pesan di browser

```php
echo "Database 'ilkoom' berhasil di buat / sudah tersedia <br>";
```

tapi apabila proses koneksi atau query yang telah kita buat ada terjadi error, maka akan diproses oleh exeption untuk ditampilkan error apa yang ditemukan.



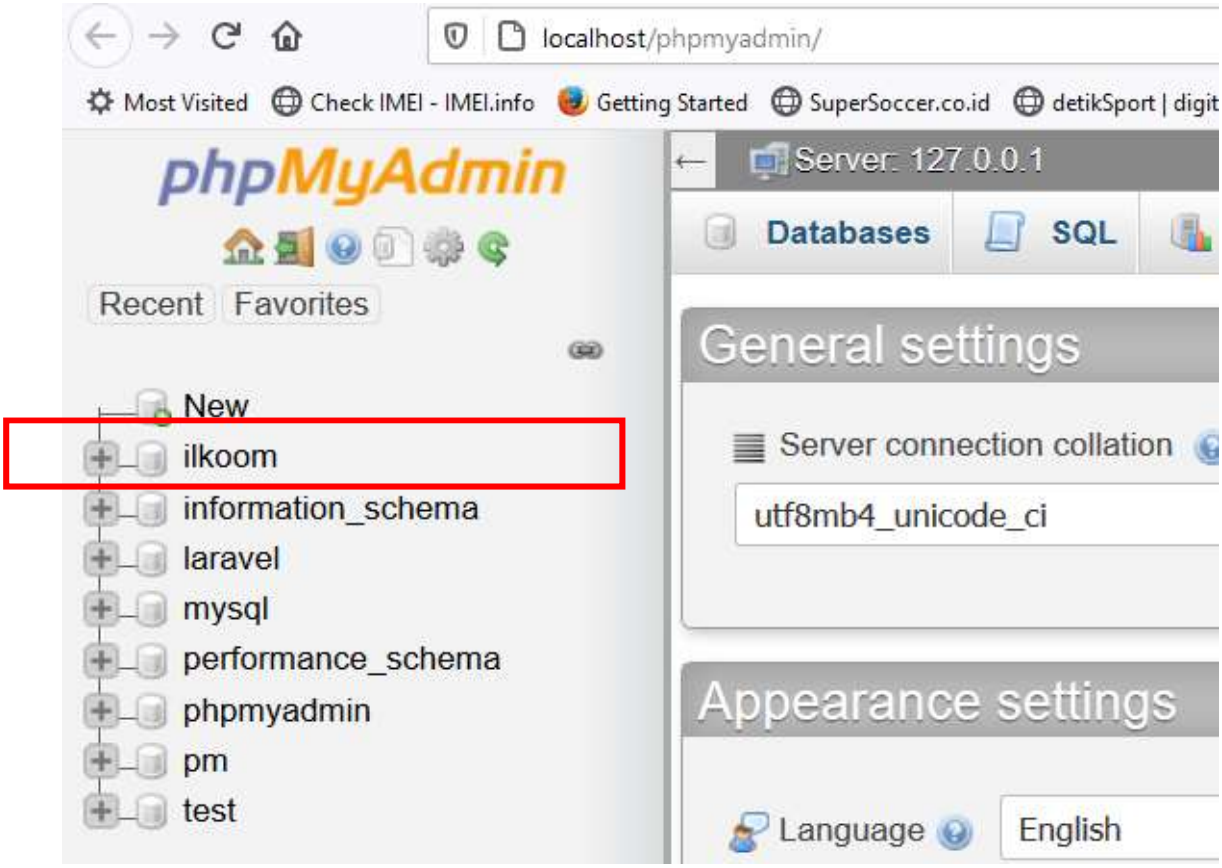
Nah sekarang silakan kalian coba untuk membuat query menjadi error misal dengan menghapus huruf E pada Create lalu simpan dan jalankan lagi programnya

```php
$query = "CREAT DATABASE IF NOT EXISTS ilkoom";
```

Salah satu pelajaran yang perlu diingat disini adalah penulisan query harus diperhatikan sesuai dengan aturan yang ada, tidak boleh typo

Sampai tahap ini kita telah berhasil membuat database secara manual menggunakan perintah oop PHP, untuk memastikan hasilnya kita akan coba cek.

Kita buka lagi server database mysql untuk memastikan database sudah terbentuk



Ok ya???

Memilih Database Dengan mysqli Object

Dari kode program sebelumnya kita telah berhasil membuat database **ilkoom**. Langkah berikutnya adalah memilih database ini sebagai database aktif dengan menjalankan method `mysqli->select_db()`. Method ini butuh sebuah argument berupa nama database yang akan dipakai. Kenapa harus dipilih karena syarat agar tabel dapat dibuat pada sebuah database maka database yang dituju harus berada pada kondisi aktif.

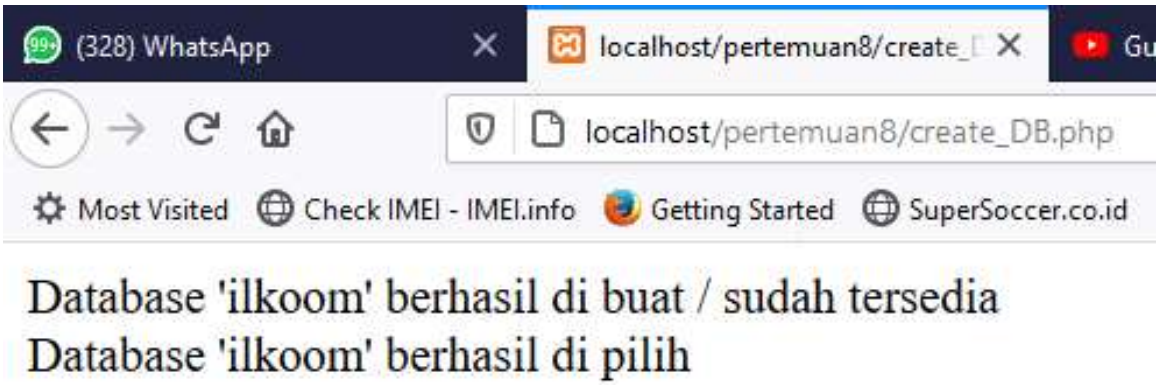
Karena sifatnya melengkapi program yang sudah kita buat sebelumnya maka saya tidak akan menulis ulang semua programnya kalian tinggal menambahkan baris program dibawah ini

```
 echo "Database 'ilkoom' berhasil di buat / sudah tersedia <br>";
}; // tadi kita sampai disini, tulis tambahan baris program dibawah ini


// Pilih database "ilkoom"
$mysqli->select_db("ilkoom");
if ($mysqli->error){
  throw new Exception($mysqli->error, $mysqli->errno);
}
else {
  echo "Database 'ilkoom' berhasil di pilih <br>";
};
```

Fungsi select_db() digunakan untuk mengaktifkan database yang akan digunakan dengan menambahkan argument dari nama database yang dimaksud.

Jika database berhasil dipilih maka akan muncul pesan sukses,



Membuat Tabel Dengan mysqli Object

Setelah sukses memilih database saatnya kita untuk membuat tabel, kita akan kembali menggunakan method mysqli::query(), namun query yang dijalankan adalah **`CREATE TABLE`**. Berikut tambahan kode programnya, masih file yang sama dengan yang sebelumnya ya biar ga repot:

Untuk tabel **barang** kita rancang dengan 5 kolom:

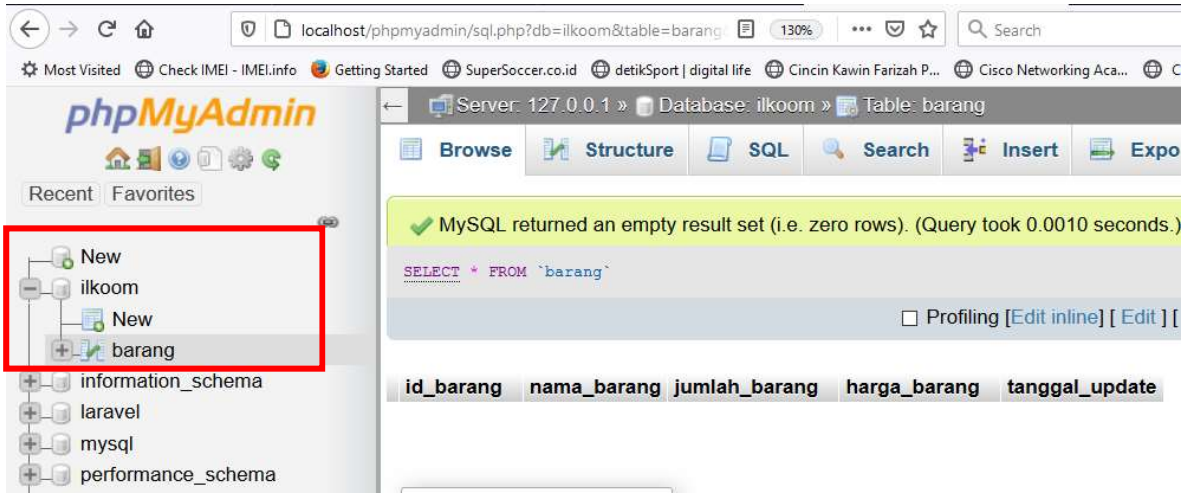
- `id_barang sebagai primary key bertipe INT`
- `nama_barang bertipe VARCHAR`
- `jumlah_barang bertipe INT`
- `harga_barang bertipe DECimal`
- `tanggal_update bertipe TIMESTAMP yg akan otomatis terisi sesuai tanggal dan waktu pada sistem saat data di input`

```
// blok kode membuat database "ilkoom"
// blok kode memilih database ilkoom

// Buat tabel "barang"
  $query = "CREATE TABLE barang (
            id_barang INT PRIMARY KEY AUTO_INCREMENT,
            nama_barang VARCHAR(50),
            jumlah_barang INT,
            harga_barang DEC,
            tanggal_update TIMESTAMP
            )";
  $mysqli->query($query);
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }
  else {
    echo "Tabel 'barang' berhasil di buat <br>";
  };
```

Tidak ada yang baru dengan program diatas, hanya perubahan query yang berisi pembuatan tabel, saya harap kalian sudah paham akan hal ini karena diajarkan juga pada materi matakuliah basisdata.

Jalankan kembali programnya dan lihat apakah tabel sudah terbentuk pada database,



Dengan bentuk program diatas kita hanya bisa menjalankan program 1 kali, apabila program dijalankan kembali akan tertangkap error pada proses pembuatan tabel karena kita sudah peernah membuat sebelumnya.

Karena pada proses latihan nanti kita akan sering mengotak atik data pada tabel maka, agar lebih efisien tidak perlu memasukan data berkali-kali kita akan menambah sedikit query untuk menghapus tabel apabila tabel sudah ada dalam database. Kita menambahkan query DROP TABLE IF EXISTS sebelum proses pembuatan tabel yang sebelumnya telah kita buat, jadi ingat ini masih rangkaian program yang sama dan kita hanya menambah beberapa baris baru untuk melengkapi, Begini perubahan program nya

```php
// tambahkan blok Hapus tabel "barang" (jika ada) sebelum blok buat tabel barang
  $query = "DROP TABLE IF EXISTS barang";
  $mysqli->query($query);
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }

  // Buat tabel "barang"
  $query = "CREATE TABLE barang (
            id_barang INT PRIMARY KEY AUTO_INCREMENT,
            nama_barang VARCHAR(50),
            jumlah_barang INT,
            harga_barang DEC,
            tanggal_update TIMESTAMP
            )";
  $mysqli->query($query);
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }
  else {
    echo "Tabel 'barang' berhasil di buat <br>";
  };
```

Dengan penambahan blok diatas maka error pembuatan tabel akan teratasi

Berikut gambaran lengkap program kita sampai terakhir

```php
<?php
mysqli_report(MYSQLI_REPORT_STRICT);

try {
  $mysqli = new mysqli("localhost", "root", "");

  // Buat database "ilkoom" (jika belum ada)
```

```php
$query = "CREATE DATABASE IF NOT EXISTS ilkoom";
$mysqli->query($query);
if ($mysqli->error){
  throw new Exception($mysqli->error, $mysqli->errno);
}
else {
  echo "Database 'ilkoom' berhasil di buat / sudah tersedia <br>";
};

// Pilih database "ilkoom"
$mysqli->select_db("ilkoom");
if ($mysqli->error){
  throw new Exception($mysqli->error, $mysqli->errno);
}
else {
  echo "Database 'ilkoom' berhasil di pilih <br>";
};

// Hapus tabel "barang" (jika ada)
$query = "DROP TABLE IF EXISTS barang";
$mysqli->query($query);
if ($mysqli->error){
  throw new Exception($mysqli->error, $mysqli->errno);
}

// Buat tabel "barang"
$query = "CREATE TABLE barang (
         id_barang INT PRIMARY KEY AUTO_INCREMENT,
         nama_barang VARCHAR(50),
         jumlah_barang INT,
         harga_barang DEC,
         tanggal_update TIMESTAMP
         )";
$mysqli->query($query);
```

```
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }
  else {
    echo "Tabel 'barang' berhasil di buat <br>";
  };


}
catch (Exception $e) {
  echo "Koneksi / Query bermasalah: ".$e->getMessage(). " (".$e-
>getCode().")";
}
finally {
  if (isset($mysqli)) {
    $mysqli->close();
  }
}
```

Mengisi Tabel Dengan mysqli Object

Setelah tabel barang selesai dibuat, saatnya kita isi dengan query `INSERT`:
Sama seperti sebelumnya kita akan menyisipkan blok program insert ke dalam program yang telah kita buat sebelumnya.

```
// Isi tabel "barang"
  $sekarang = new DateTime('now', new DateTimeZone('Asia/Jakarta'));
  $timestamp = $sekarang->format("Y-m-d H:i:s");
  $query = "INSERT INTO barang
    (nama_barang, jumlah_barang, harga_barang, tanggal_update) VALUES
      ('TV Samsung 43NU7090 4K',5,5399000,'$timestamp'),
      ('Kulkas LG GC-A432HLHU',10,7600000,'$timestamp'),
      ('Laptop ASUS ROG GL503GE',7,16200000,'$timestamp'),
      ('Printer Epson L220',14,2099000,'$timestamp'),
      ('Smartphone Xiaomi Pocophone F1',25,4750000,'$timestamp')
    ;";
```

```
  $mysqli->query($query);
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }
  else {
    echo "Tabel 'barang' berhasil di isi ".$mysqli->affected_rows."

         baris data <br>";
  };
```

Kita bahas program diatas

Pertama yaitu baris

```
$sekarang = new DateTime('now', new DateTimeZone('Asia/Jakarta'));
  $timestamp = $sekarang->format("Y-m-d H:i:s");
```

Sebelum query `INSERT`, kita menyiapkan data tanggal saat ini. Caranya dengan membuat **DateTime** object, lalu di format sebagai "`Y-m-d H:i:s`". Jika **$sekarang** adalah tanggal 04-01-2019 pukul 14:18:25 maka isi variabel **`$timestamp`** adalah `2019-01-04 14:18:25`. Format ini akan dipakai untuk tipe data `TIMESTAMP` pada kolom `tanggal_update`.

Lalu baris query insert, format umum yang telah kita ketahui untuk query insert adalah

```
INSERT... (nama_kolom1, nama_kolom2,...) VALUES (data1, data2,... )
```

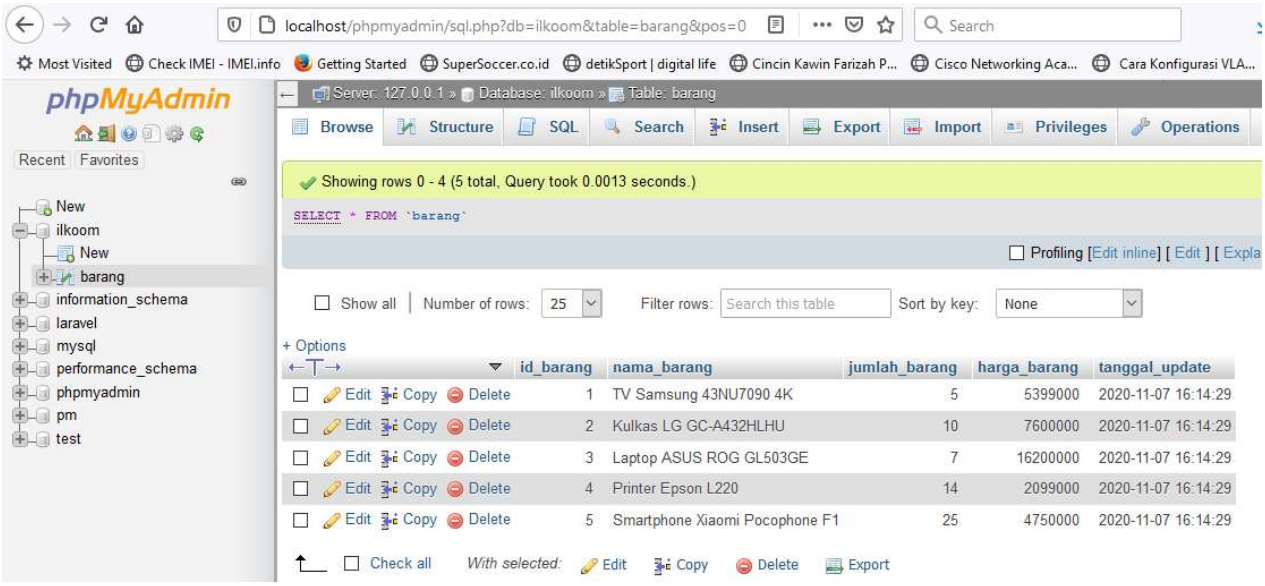
```
Dengan format di atas kita bisa langsung sekaligus mengisikan data dengan
jumlah banyak.
```

```
$query = "INSERT INTO barang
    (nama_barang, jumlah_barang, harga_barang, tanggal_update) VALUES
      ('TV Samsung 43NU7090 4K',5,5399000,'$timestamp'),
      ('Kulkas LG GC-A432HLHU',10,7600000,'$timestamp'),
      ('Laptop ASUS ROG GL503GE',7,16200000,'$timestamp'),
      ('Printer Epson L220',14,2099000,'$timestamp'),
      ('Smartphone Xiaomi Pocophone F1',25,4750000,'$timestamp')
    ;";
```

Bila program diatas berhasil dijalankan maka outputnya:

```
Database 'ilkoom' berhasil di buat / sudah tersedia
Database 'ilkoom' berhasil di pilih
Tabel 'barang' berhasil di buat
Tabel 'barang' berhasil di isi 5 baris data
```

Lalu cek pada database barang kita, maka akan ada 5 tambahan data yang kita input:



Sukses ya??

Berikut program lengkapnya

```php
<?php
mysqli_report(MYSQLI_REPORT_STRICT);
try {
  $mysqli = new mysqli("localhost", "root", "");
  // Buat database "ilkoom" (jika belum ada)
  $query = "CREATE DATABASE IF NOT EXISTS ilkoom";
  $mysqli->query($query);
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }
  else {
    echo "Database 'ilkoom' berhasil di buat / sudah tersedia <br>";
  };

  // Pilih database "ilkoom"
  $mysqli->select_db("ilkoom");
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }
```

```
  else {
    echo "Database 'ilkoom' berhasil di pilih <br>";
  };

  // Hapus tabel "barang" (jika ada)
  $query = "DROP TABLE IF EXISTS barang";
  $mysqli->query($query);
  if ($mysqli->error){
    throw new Exception($mysqli->error, $mysqli->errno);
  }

  // Buat tabel "barang"
  $query = "CREATE TABLE barang (
    id_barang INT PRIMARY KEY AUTO_INCREMENT,
    nama_barang VARCHAR(50),
    jumlah_barang INT,
    harga_barang DEC,
    tanggal_update TIMESTAMP
    )";
$mysqli->query($query);
if ($mysqli->error){
throw new Exception($mysqli->error, $mysqli->errno);
}
else {
echo "Tabel 'barang' berhasil di buat <br>";
};

 // Isi tabel "barang"
 $sekarang = new DateTime('now', new DateTimeZone('Asia/Jakarta'));
 $timestamp = $sekarang->format("Y-m-d H:i:s");

 $query = "INSERT INTO barang
   (nama_barang, jumlah_barang, harga_barang, tanggal_update) VALUES
     ('TV Samsung 43NU7090 4K',5,5399000,'$timestamp'),
```

```
    ('Kulkas LG GC-A432HLHU',10,7600000,'$timestamp'),
    ('Laptop ASUS ROG GL503GE',7,16200000,'$timestamp'),
    ('Printer Epson L220',14,2099000,'$timestamp'),
    ('Smartphone Xiaomi Pocophone F1',25,4750000,'$timestamp')
  ;";
 $mysqli->query($query);
 if ($mysqli->error){
   throw new Exception($mysqli->error, $mysqli->errno);
 }
 else {
   echo "Tabel 'barang' berhasil di isi ".$mysqli->affected_rows."
        baris data <br>";
 };
}
catch (Exception $e) {
  echo "Koneksi / Query bermasalah: ".$e->getMessage(). " (".$e->getCode().")";
}
finally {
  if (isset($mysqli)) {
    $mysqli->close();
  }
}
```