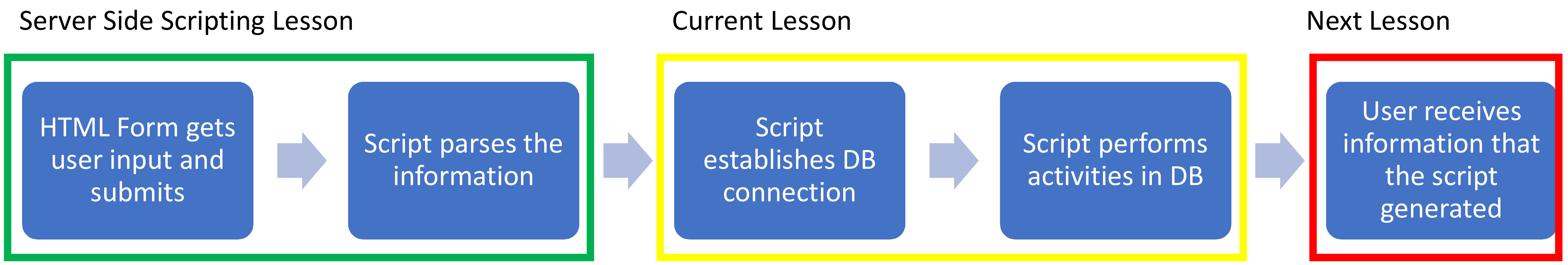
Connecting to a DB

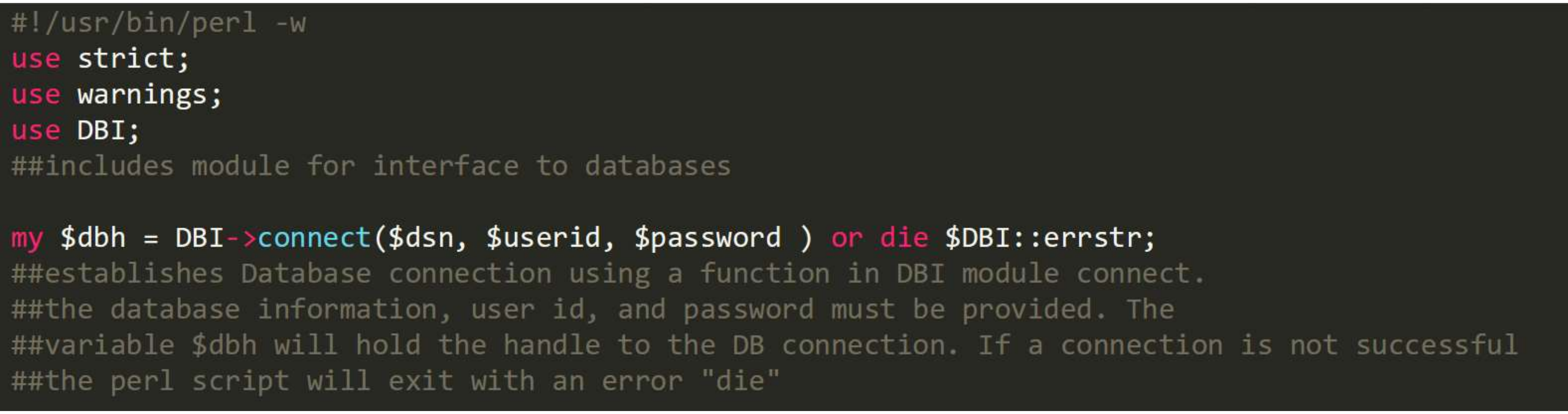
Server Side Scripts – Content

- Data Storage Request
  - Order submission
  - Contact form
- Data Retrieval Request
  - Order status
  - Login
  - Business analytics

Server Side Connection to a DB


Server Side Scripting Lesson
Current Lesson
Next Lesson
HTML Form gets user input and submits
Script parses the information
Script establishes DB connection
Script performs activities in DB
User receives information that the script generated

How a DB Connection is Established

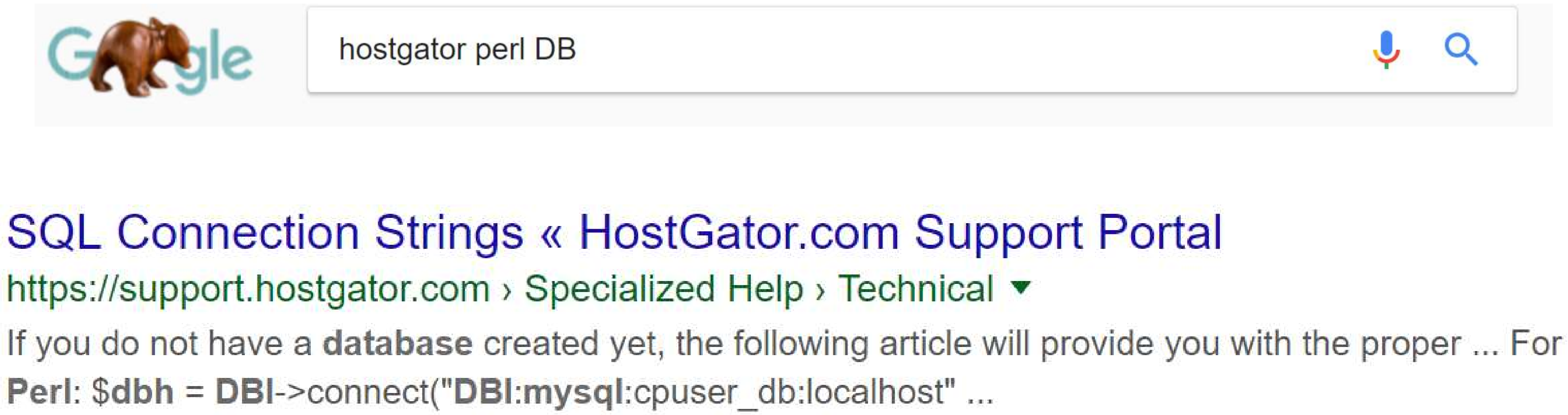


```
#!/usr/bin/perl -w
use strict;
use warnings;
use DBI;
##includes module for interface to databases

my $dbh = DBI->connect($dsn, $userid, $password ) or die $DBI::errstr;
##establishes Database connection using a function in DBI module connect.
##the database information, user id, and password must be provided. The
##variable $dbh will hold the handle to the DB connection. If a connection is not successful
##the perl script will exit with an error "die"
```

DB Connection Details

- The way to connect to a DB might be different across host providers and service plans within those. Information needs to be looked up.

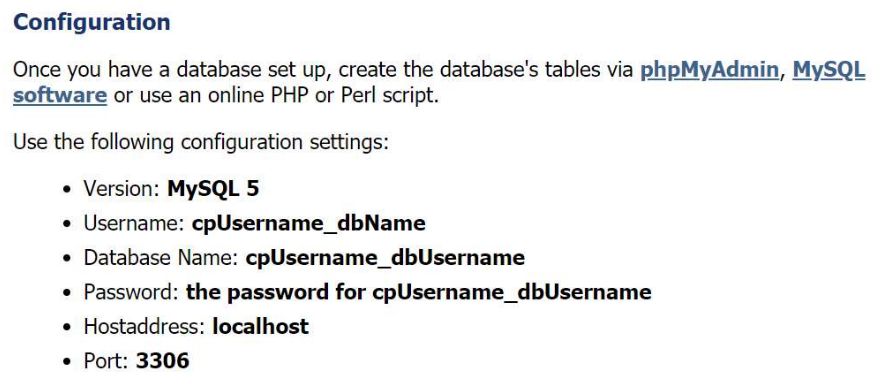


SQL Connection Strings « HostGator.com Support Portal

https://support.hostgator.com › Specialized Help › Technical ▾

If you do not have a **database** created yet, the following article will provide you with the proper ... For **Perl**: $**dbh** = **DBI**->connect("**DBI:mysql**:cpuser_db:localhost" ...

Hostgator DB Connection Details



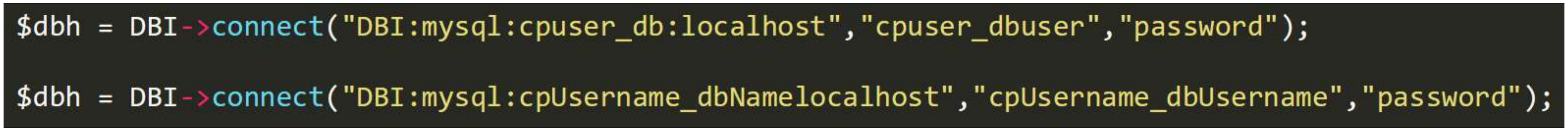
Configuration

Once you have a database set up, create the database's tables via phpMyAdmin, MySQL software or use an online PHP or Perl script.

Use the following configuration settings:

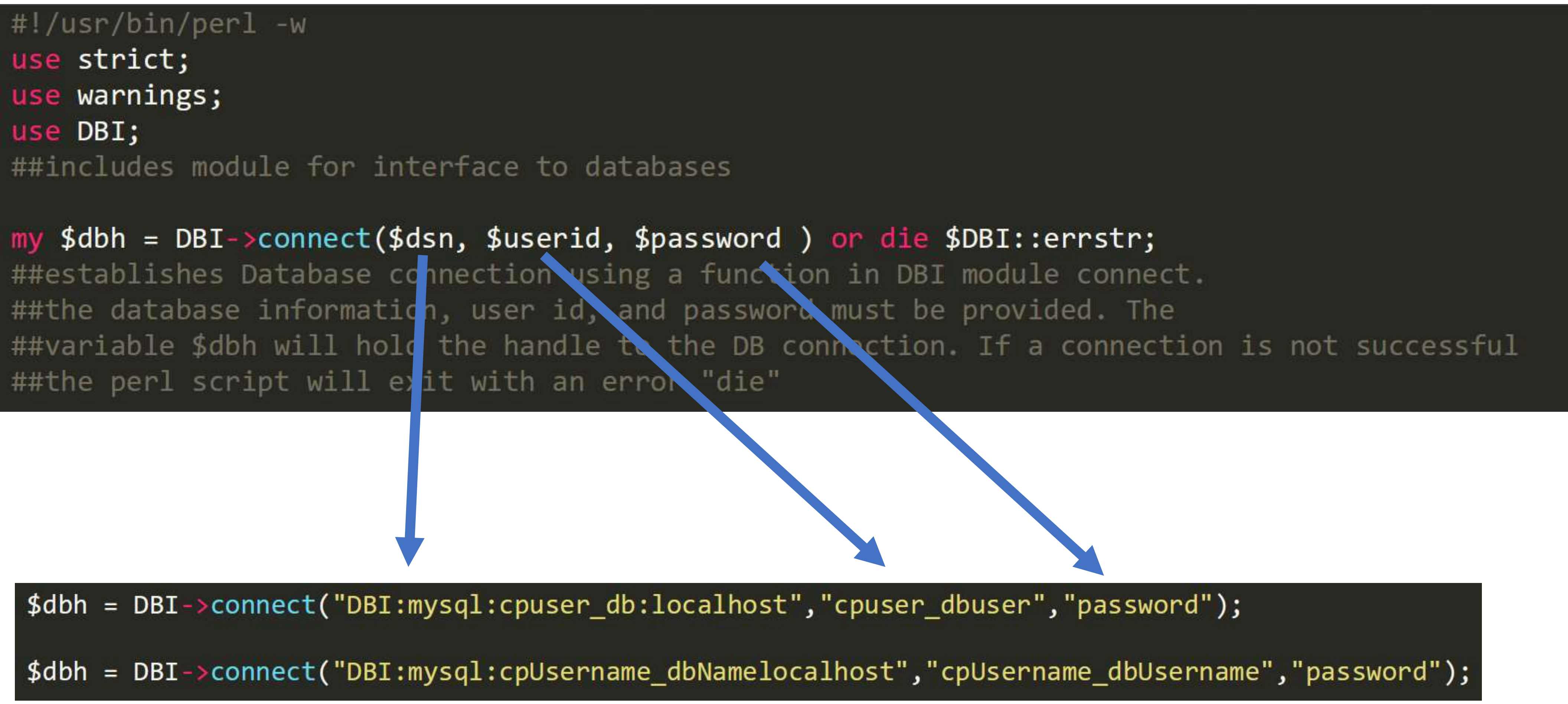
- Version: **MySQL 5**
- Username: **cpUsername_dbName**
- Database Name: **cpUsername_dbUsername**
- Password: **the password for cpUsername_dbUsername**
- Hostaddress: **localhost**
- Port: **3306**

Hostgator offers 2 examples of how its possible to access DB on their server



```
$dbh = DBI->connect("DBI:mysql:cpuser_db:localhost","cpuser_dbuser","password");

$dbh = DBI->connect("DBI:mysql:cpUsername_dbNamelocalhost","cpUsername_dbUsername","password");
```

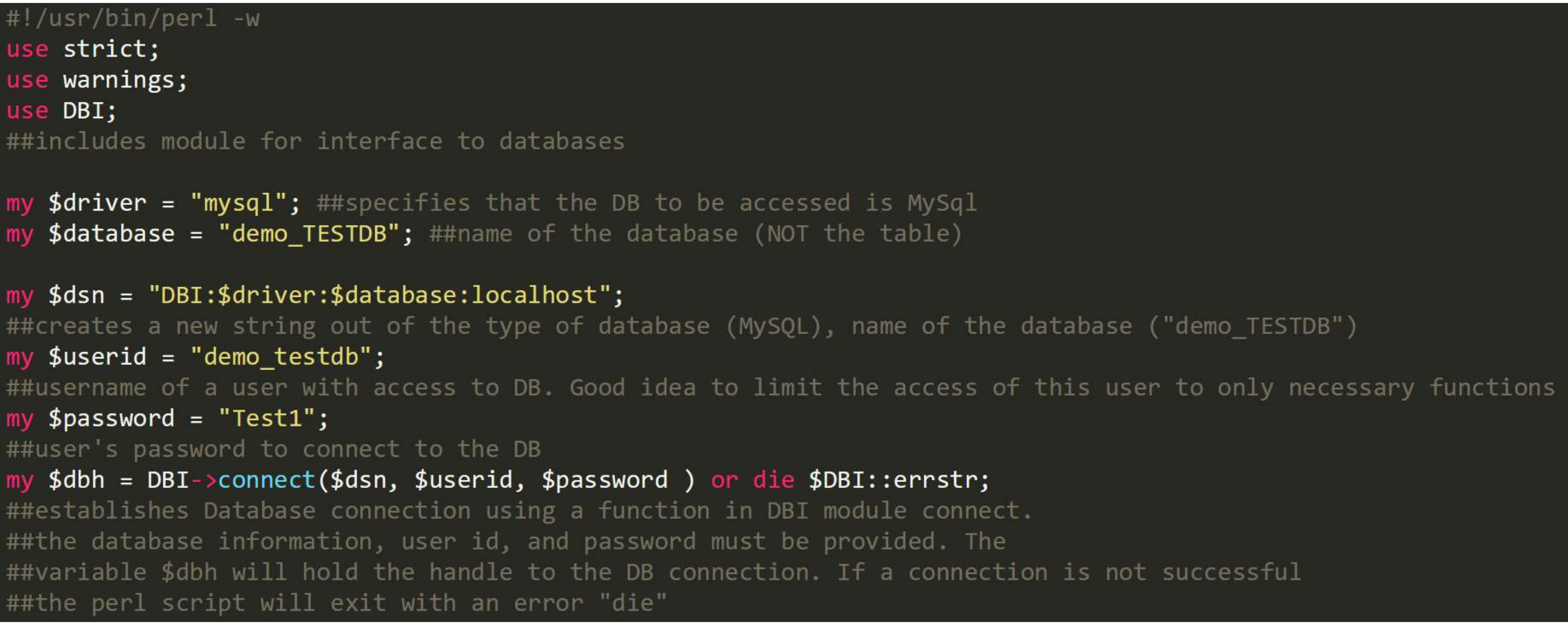


```
#!/usr/bin/perl -w
use strict;
use warnings;
use DBI;
##includes module for interface to databases

my $dbh = DBI->connect($dsn, $userid, $password ) or die $DBI::errstr;
##establishes Database connection using a function in DBI module connect.
##the database information, user id, and password must be provided. The
##variable $dbh will hold the handle to the DB connection. If a connection is not successful
##the perl script will exit with an error "die"
```

```
$dbh = DBI->connect("DBI:mysql:cpuser_db:localhost","cpuser_dbuser","password");

$dbh = DBI->connect("DBI:mysql:cpUsername_dbNamelocalhost","cpUsername_dbUsername","password");
```



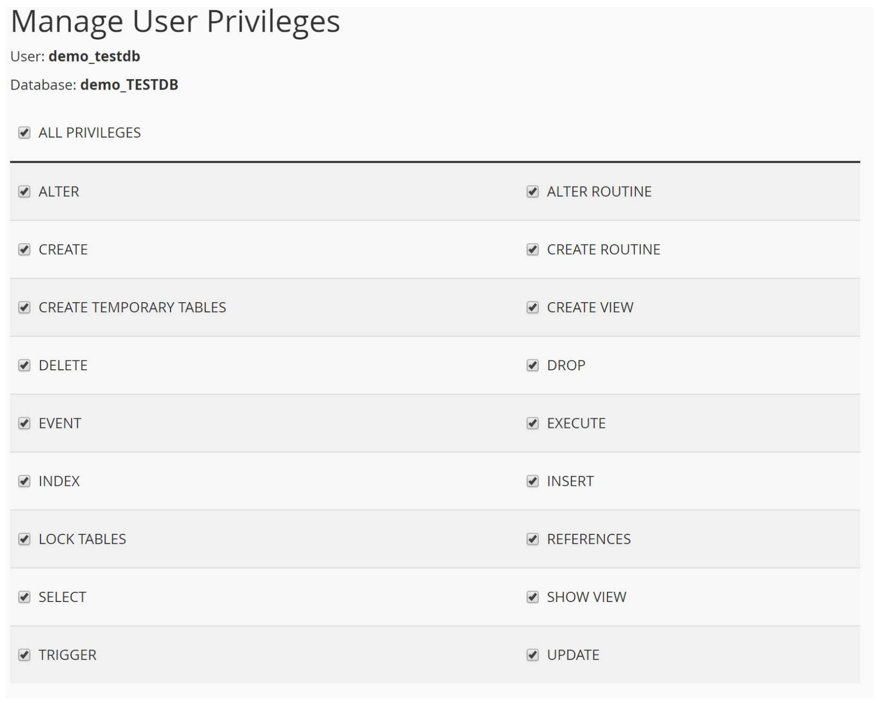
```
#!/usr/bin/perl -w
use strict;
use warnings;
use DBI;
##includes module for interface to databases

my $driver = "mysql"; ##specifies that the DB to be accessed is MySql
my $database = "demo_TESTDB"; ##name of the database (NOT the table)

my $dsn = "DBI:$driver:$database:localhost";
##creates a new string out of the type of database (MySQL), name of the database ("demo_TESTDB")
my $userid = "demo_testdb";
##username of a user with access to DB. Good idea to limit the access of this user to only necessary functions
my $password = "Test1";
##user's password to connect to the DB
my $dbh = DBI->connect($dsn, $userid, $password ) or die $DBI::errstr;
##establishes Database connection using a function in DBI module connect.
##the database information, user id, and password must be provided. The
##variable $dbh will hold the handle to the DB connection. If a connection is not successful
##the perl script will exit with an error "die"
```

Ensuring correct user permissions

- Limit users to permissions that are required to perform the necessary task



Manage User Privileges

User: **demo_testdb**

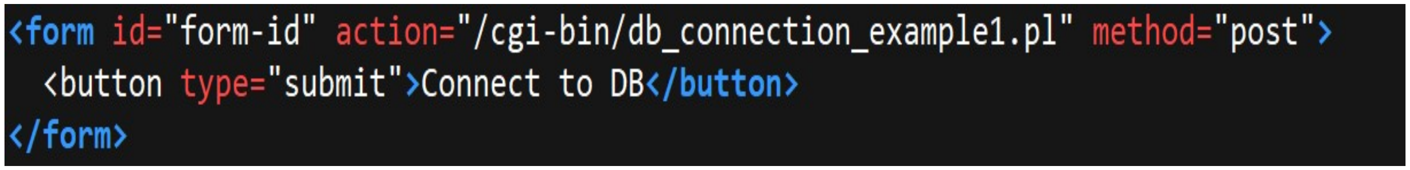
Database: **demo_TESTDB**

☑ ALL PRIVILEGES

☑ ALTER	☑ ALTER ROUTINE
☑ CREATE	☑ CREATE ROUTINE
☑ CREATE TEMPORARY TABLES	☑ CREATE VIEW
☑ DELETE	☑ DROP
☑ EVENT	☑ EXECUTE
☑ INDEX	☑ INSERT
☑ LOCK TABLES	☑ REFERENCES
☑ SELECT	☑ SHOW VIEW
☑ TRIGGER	☑ UPDATE

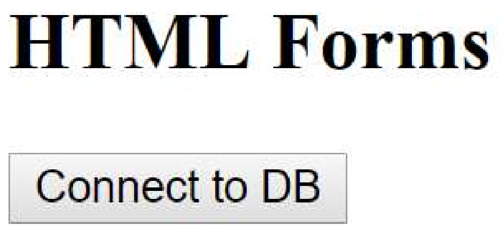
DB Connection Tutorial

- Create a database called: **Contact_Information**
- For the database, create a user: **contactus**
- For the user contact_us_form, create a password: **password**
- Take the perl script (db_connection_example1.pl) and change the variable definitions as appropriate for the newly created database
  - Double check your actual DB name. There might be a prefix before the name you designated
  - Double check your username.
- Create an html form that will POST data to the perl script
- Observe the results!

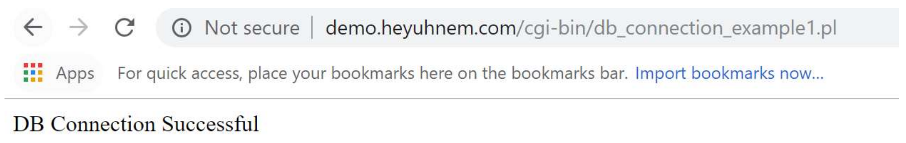


```
<form id="form-id" action="/cgi-bin/db_connection_example1.pl" method="post">
  <button type="submit">Connect to DB</button>
</form>
```

Result

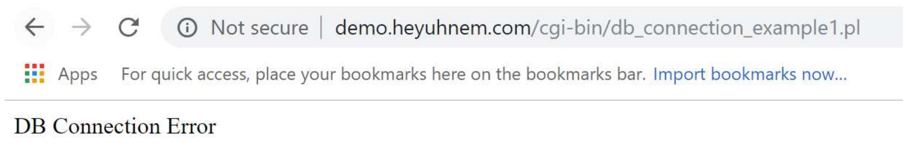

HTML Forms

Properly Connected


Not secure
demo.heyuhnem.com/cgi-bin/db_connection_example1.pl
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

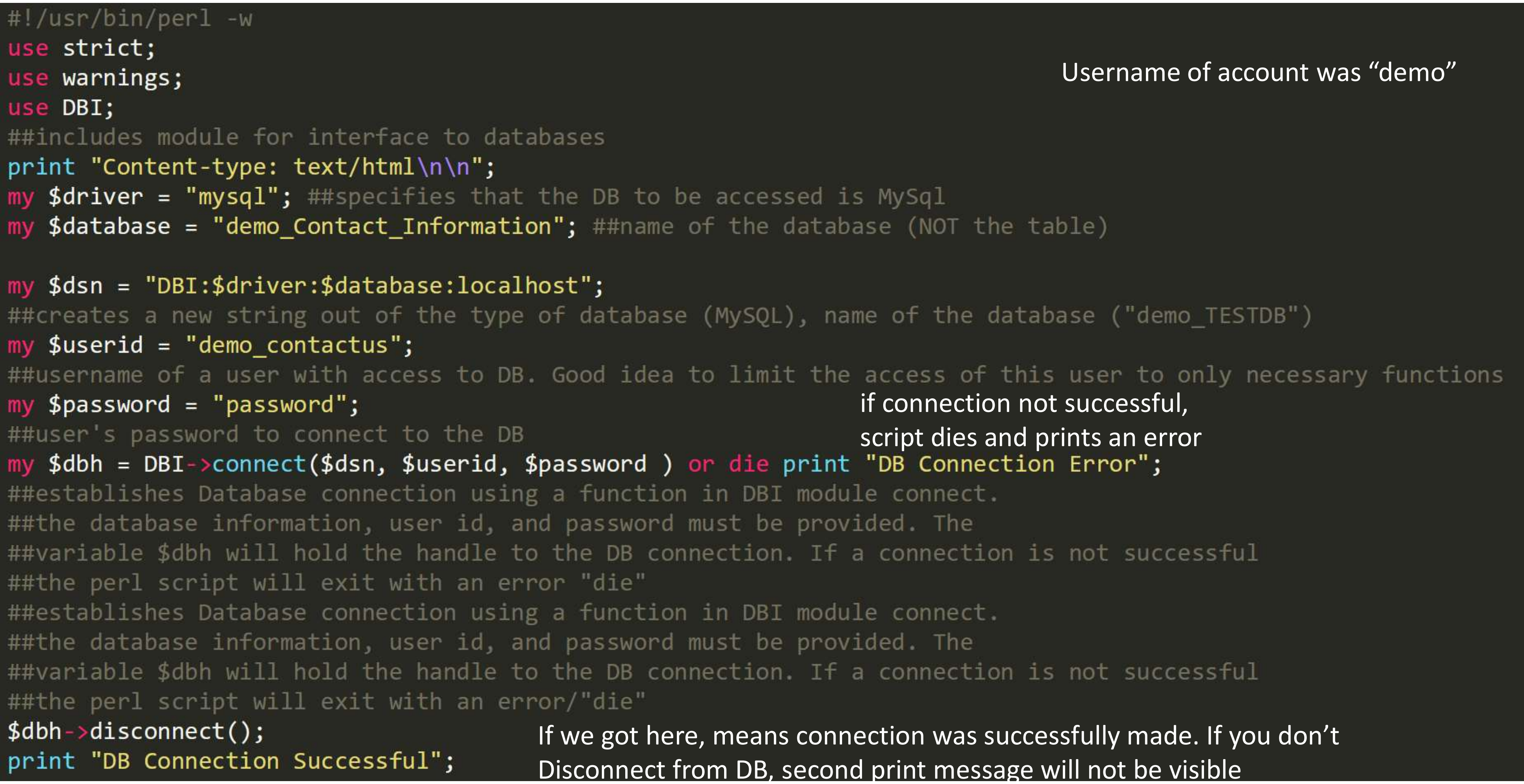
```
DB Connection Successful
```

Improperly Connected


Not secure
demo.heyuhnem.com/cgi-bin/db_connection_example1.pl
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

```
DB Connection Error
```

Link



```perl
#!/usr/bin/perl -w
use strict;
use warnings;
use DBI;
##includes module for interface to databases
print "Content-type: text/html\n\n";
my $driver = "mysql"; ##specifies that the DB to be accessed is MySql
my $database = "demo_Contact_Information"; ##name of the database (NOT the table)

my $dsn = "DBI:$driver:$database:localhost";
##creates a new string out of the type of database (MySQL), name of the database ("demo_TESTDB")
my $userid = "demo_contactus";
##username of a user with access to DB. Good idea to limit the access of this user to only necessary functions
my $password = "password";
##user's password to connect to the DB
my $dbh = DBI->connect($dsn, $userid, $password ) or die print "DB Connection Error";
##establishes Database connection using a function in DBI module connect.
##the database information, user id, and password must be provided. The
##variable $dbh will hold the handle to the DB connection. If a connection is not successful
##the perl script will exit with an error "die"
##establishes Database connection using a function in DBI module connect.
##the database information, user id, and password must be provided. The
##variable $dbh will hold the handle to the DB connection. If a connection is not successful
##the perl script will exit with an error/"die"
$dbh->disconnect();
print "DB Connection Successful";
```

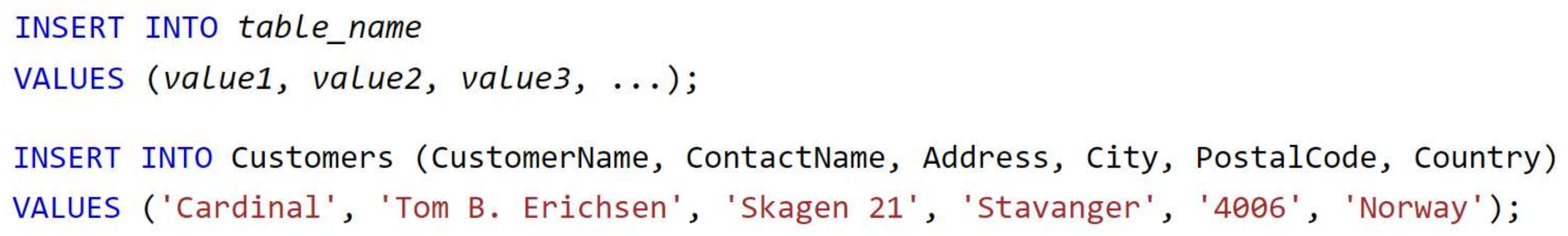
Username of account was “demo”

if connection not successful,
script dies and prints an error

If we got here, means connection was successfully made. If you don’t
Disconnect from DB, second print message will not be visible

Adding data to the database Workflow

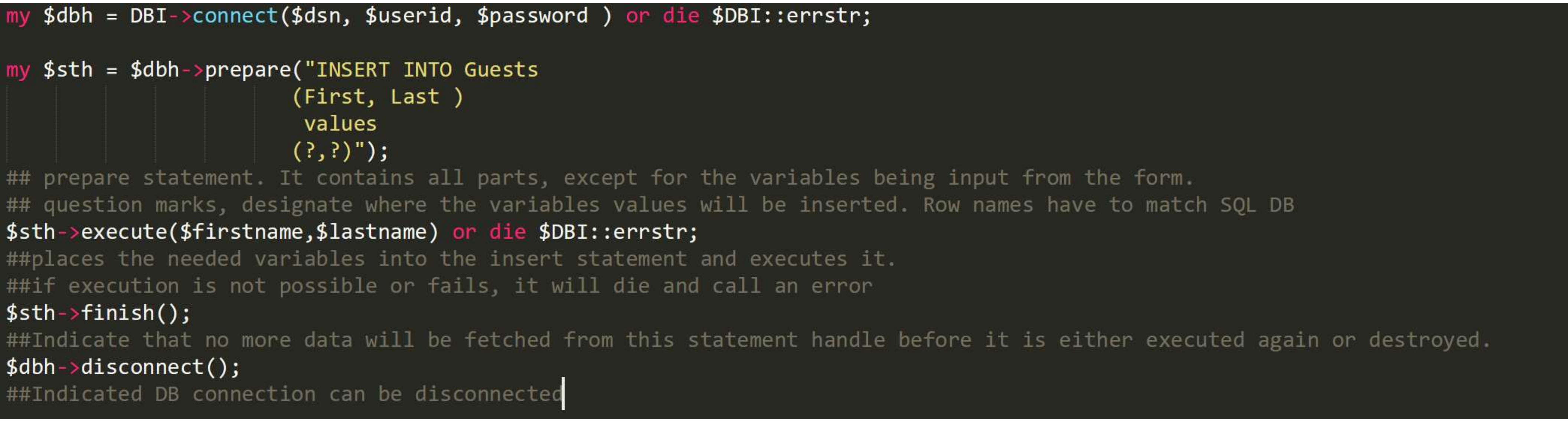
- A DB connection, gives access to the database console
- An INSERT statement needs to be generated in order to add data to the database using a server side script
- INSERT statement needs to be executed
- DB connection needs to be closed (disconnected)



```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');
```

Perl Statement Generation



```
my $dbh = DBI->connect($dsn, $userid, $password ) or die $DBI::errstr;

my $sth = $dbh->prepare("INSERT INTO Guests
                       (First, Last )
                        values
                       (?,?)");
## prepare statement. It contains all parts, except for the variables being input from the form.
## question marks, designate where the variables values will be inserted. Row names have to match SQL DB
$sth->execute($firstname,$lastname) or die $DBI::errstr;
##places the needed variables into the insert statement and executes it.
##if execution is not possible or fails, it will die and call an error
$sth->finish();
##Indicate that no more data will be fetched from this statement handle before it is either executed again or destroyed.
$dbh->disconnect();
##Indicated DB connection can be disconnected
```

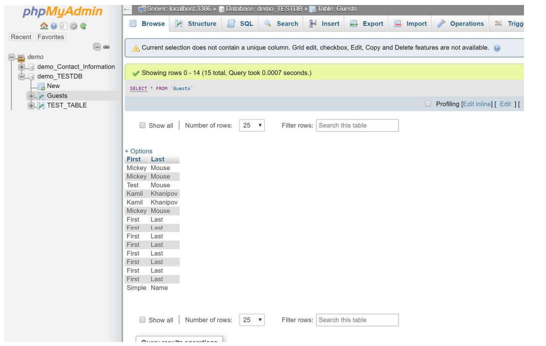
Take a look at an example

2 Files in Example 2 (HTML Form and Perl Script)

demo.heyuhnem.com/simple_form_db.html

DIY

- Create a table within your database with 2 VARCHAR columns:
  - First
  - Last
- Change the login details in the perl example file
- Launch the html form and the perl script on hostgator
- Perform form submissions and track if the data is getting added
  - Go to Cpanel on Hostagator
  - Go to phpMyAdmin under databases
  - Navigate to the right database and table to see what is in it



Server: localhost:3306 » Database: demo_TESTDB » Table: Guests

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 14 (15 total, Query took 0.0007 seconds.)

```
SELECT * FROM `Guests`
```

Profiling [Edit inline] [Edit] [

First	Last
Mickey	Mouse
Mickey	Mouse
Test	Mouse
Kamil	Khanipov
Kamil	Khanipov
Mickey	Mouse
First	Last
First	Last
First	Last
First	Last
First	Last
First	Last
First	Last
First	Last
Simple	Name

DIY: Make the Jump

- Create a database for the form you were working on before
  - 4 input fields: First Name, Last Name, Email and Phone
- Take the form_extended.html (don’t forget to change action attribute of the form element to submit to the correct perl script)
- Create a Perl Script to take the data and parse it
- Insert the data into your MySQL database
- Print a message at the end of the script, stating that the connection has been made and data added.
- Submit to all files used and screenshot of database content