Database Driven Sites - mySQL, PHP, and MVC Overview

DB Driven Sites

- Rather than storing the content of our Web site directly within files it may be better to store the content within a datastore (XML files, Database tables, etc.)
  - We will then pull out the desired data from the database and combine it with HTML markup and CSS in the form of a template to create a complete page
    - Separation of code, presentation, and data is a good aim – though not unique to server-side dev.
  - The DB driven site provides for the possibility of dynamic or even personalized pages
    - User preferences, promotions in e-commerce, custom reports, portals, etc.
  - DB driven sites also afford us the opportunity to build content management systems (CMS)

- Important Idea – When do you build the pages in a DB driven site?
    - As they are requested? Before they are requested?
    - Your ability to prebuild pages is determined by how variable the content is. Avoid the dynamic-static trap if possible

The pieces of the DB Driven Site

- The DB
  - Oracle, MS SQL Server, mySQL, MongoDB, Redis, etc.
- Middleware / Programming Environment
  - PHP, CGI/Perl, Node.js, ASP/ASP.NET, CFM, JSP, etc.
- APIs and Drivers for DB Connectivity
  - Abstracting away the DB and specific concerns
    - Question: How often will the data store change?
  - Specifics
    - ODBC, native drivers, ADO.NET, JDBC
  - The mapping between table relations and objects also may be handled here – ORM (Object-relational mapping)
- Separate Templating Environment
  - May be part of the middleware programming environment

An App Approach

- User driven
  - UI first
- Data driven
  - Tables and data first
- Top down vs. bottom up?
- Seems is if really they are intertwined, but clearly one is more “seen” than the other

Step 1: Model Your Data

- First determine what are the types of data that your application is going to keep track of
  - Consider a table to hold user information we might have a login, password, first name, last name, and other descriptive data to keep track of our users
  - We need to assign data types to each type of information we track
    - Common data types: CHAR(LENGTH), VARCHAR(LENGTH), TEXT, MEDIUMTEXT, LONGTEXT, SMALLINT,MEDIUMINT, INT, FLOAT, DATA, DATETIME, ENUM
  - Notice that emphasis is placed upon the precision of the information stored
    - For good reason we want accuracy and we don't want to waste space!
  - We usually need to specify a primary key that is unique so that we can keep track of data properly

Step 1: Model Your Data

- It is often easier to visualize the data like so
- We could imagine a filled in set of data looking something like
- Warning: see BIG problem?

User Table

User_id (primary)	MEDIUMINT
First_name	Varchar(35)
Last_name	Varchar(35)
Login	Char(16)
Password	Char(16)

User_id	First_name	Last_name	Login	Password
347	Moe	Howard	bossman	Whyyou!
348	Larry	Howard	larryH	secret
349	Curly	Joe	curly3	Woowoo3

Step 1: Model Your Data

- Very often you will not have a single table but a collection of tables
- In this case you generally need to have something that relates data between the tables, typically this is the primary key
- Given our last example we might imagine a table called "Log" that has the system's activity log.

Log Table

User_id	Activity	Time
347	Login-fail	2004-08-04 11:46:04
347	Login-fail	2004-08-04 11:46:04
348	Login	2004-08-04 11:46:04
347	Login	2004-08-04 11:46:04
348	Logout	2004-08-04 11:46:04

Step 2 - Create the Database and Tables

- If you have shell access to a server you might use the mysql monitor to admin the database

- mysql -h *hostname* -u *username* -p *password*
  - *Hostname* will be localhost in the case you run this on the same box
  - Example

```
u39622683:~ > mysql -h db299.perfora.net -u dbo154377598 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1435237 to server version: 4.0.25-standard-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Step 2 - Create the Database and Tables

- Once you are connected you normally would create a database
  - mysql> CREATE DATABASE *datbasename;*
- In this situation we have a database specified already, so just start using it.
  - mysql> USE *databasename*;

```
mysql> use db154377598;
Database changed
mysql>
```

Step 2 – Create the Database and Tables

- Now create the table(s) you want based upon your defined schema

- CREATE TABLE users (user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,first_name VARCHAR(35) NOT NULL,last_name VARCHAR(35) NOT NULL,login CHAR(16) NOT NULL,password CHAR(16) NOT NULL,PRIMARY KEY (user_id));

- [[Watch out for returns and syntax!!]]

```
mysql> CREATE TABLE users (user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,first_name V
ARCHAR(35) NOT NULL,last_name VARCHAR(35) NOT NULL,login CHAR(16) NOT NULL,password CHAR(1
6) NOT NULL,PRIMARY KEY (user_id));
Query OK, 0 rows affected (0.01 sec)
```

Showing the Results

- Once using a DB issue SHOW TABLES; statement

```
mysql> show tables;
+------------------------+
| Tables_in_db154377598 |
+------------------------+
| users                  |
+------------------------+
1 row in set (0.00 sec)
```

- To look deeper at a table use SHOW COLUMNS from *tablename*; statement

```
mysql> show columns from users;
+------------+----------------------+------+-----+---------+----------------+
| Field      | Type                 | Null | Key | Default | Extra          |
+------------+----------------------+------+-----+---------+----------------+
| user_id    | mediumint(8) unsigned |      | PRI | NULL    | auto_increment |
| first_name | varchar(35)          |      |     |         |                |
| last_name  | varchar(35)          |      |     |         |                |
| login      | varchar(16)          |      |     |         |                |
| password   | varchar(16)          |      |     |         |                |
+------------+----------------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

Inserting Data

- Use the INSERT statement with the following syntax:
  - INSERT INTO *tablename* (*column* 1,…,*column* n) VALUES (‘*value* 1’,…, ’*value* n’);

  - *Note: The 1 - n indication just indicates a number of columns to be used, the columns do not have to be contiguous*
  - *INSERT INTO users (first_name, last_name, login, password) VALUES (‘Thomas’, ‘Powell’, ‘tpowell’, ‘superSecret’);*

```
mysql> INSERT INTO users (first_name,last_name,login,password) VALUES ('Thomas','Powell','
tpowell','superSecret');
Query OK, 1 row affected (0.00 sec)
```

Inserting Data

- *You can also use a simple syntax like*
  - *INSERT INTO tablename VALUES ('value1',…'value n');*

  - *Note: In this case you must specify values for all columns in the table, often NULL will be used.*
  - *Note: In case of auto-increment values you still have to add something.*

```
mysql> insert into users values ('James','Kirk','capn','555destruct');
ERROR 1136: Column count doesn't match value count at row 1
mysql> insert into users values ('1','James','Kirk','capn','555destruct');
ERROR 1062: Duplicate entry '1' for key 1
mysql> insert into users values ('2','James','Kirk','capn','555destruct');
Query OK, 1 row affected (0.00 sec)
```

Inserting Data

- *It is also possible to insert multiple records at once*

```
mysql> INSERT INTO users (first_name,last_name,login,password) VALUES ('Larry','Fine','lar
ry','larbo') ,
    -> ('Moe','Howard','bossman','whyyou') ,
    -> ('Curly','Joe','curly7','woowoo');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Selecting Data

- SELECT used to retrieve info from a table.
  - Syntax: SELECT *columns* FROM *table*

```
mysql> SELECT first_name FROM users;
+------------+
| first_name |
+------------+
| Thomas     |
| James      |
+------------+
2 rows in set (0.00 sec)

mysql> SELECT login,password FROM users;
+----------+--------------+
| login    | password     |
+----------+--------------+
| tpowell  | superSecret  |
| capn     | 555destruct  |
+----------+--------------+
2 rows in set (0.01 sec)

mysql> SELECT * FROM users;
+----------+--------------+------------+----------+--------------+
| user_id  | first_name   | last_name  | login    | password     |
+----------+--------------+------------+----------+--------------+
|        1 | Thomas       | Powell     | tpowell  | superSecret  |
|        2 | James        | Kirk       | capn     | 555destruct  |
+----------+--------------+------------+----------+--------------+
2 rows in set (0.00 sec)
```

More Selecting Data

- Add the WHERE conditional to limit a selection. You can use typical operators like =,<,>,<=,>=, !=, AND (&&), OR (||), and NOT (!)
  - Special operators like IS NOT NULL, IS NULL, BETWEEN, and NOT BETWEEN are also available under mySQL

```
mysql> SELECT * FROM users WHERE login = "capn";
+---------+------------+-----------+-------+-------------+
| user_id | first_name | last_name | login | password    |
+---------+------------+-----------+-------+-------------+
|       2 | James      | Kirk      | capn  | 555destruct |
+---------+------------+-----------+-------+-------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT * FROM users WHERE user_id < 3 AND login > "d";
+---------+------------+-----------+---------+-------------+
| user_id | first_name | last_name | login   | password    |
+---------+------------+-----------+---------+-------------+
|       1 | Thomas     | Powell    | tpowell | superSecret |
+---------+------------+-----------+---------+-------------+
1 row in set (0.33 sec)
```

More Selecting Data

- In the case of strings you often need to use LIKE and NOT LIKE in conjunction with a pattern match of either _ (a single character match) or % which matches zero or more characters

```
mysql> SELECT * FROM users WHERE first_name LIKE 'T%';
+---------+------------+-----------+---------+-------------+
| user_id | first_name | last_name | login   | password    |
+---------+------------+-----------+---------+-------------+
|       1 | Thomas     | Powell    | tpowell | superSecret |
|       6 | Trevor     | Hoffman   | closer  | hbells      |
+---------+------------+-----------+---------+-------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM users WHERE login LIKE 't_owell';
+---------+------------+-----------+---------+-------------+
| user_id | first_name | last_name | login   | password    |
+---------+------------+-----------+---------+-------------+
|       1 | Thomas     | Powell    | tpowell | superSecret |
|       7 | Tony       | Rowell    | trowell | fakeone     |
+---------+------------+-----------+---------+-------------+
2 rows in set (0.00 sec)
```

Sorting Data

- Add ORDER BY to a SELECT statement
  - SELECT * from *table* ORDER BY *column*;
  - Attach ASC for ascending (default) or DESC for descending after the column
  - Join multiple columns together with commas

```
mysql> SELECT * FROM users ORDER BY first_name DESC;
+---------+------------+-----------+---------+-------------+
| user_id | first_name | last_name | login   | password    |
+---------+------------+-----------+---------+-------------+
|       1 | Thomas     | Powell    | tpowell | superSecret |
|       4 | Moe        | Howard    | bossman | whyyou      |
|       3 | Larry      | Fine      | larry   | larbo       |
|       2 | James      | Kirk      | capn    | 555destruct |
|       5 | Curly      | Joe       | curly7  | woowoo      |
+---------+------------+-----------+---------+-------------+
5 rows in set (0.01 sec)
```

Updating Data

- **UPDATE** *table* **SET** *column*='*value*';
- **UPDATE** *table* **SET** *column1*='*value1*',...*columnN*='*valueN*';
- **UPDATE** *table* **SET** *column*='*value*' **WHERE** *column2*='*value2*';

```
mysql> UPDATE users SET login="captain" WHERE user_id="2";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM users WHERE user_id="2";
+---------+------------+-----------+---------+-------------+
| user_id | first_name | last_name | login   | password    |
+---------+------------+-----------+---------+-------------+
|       2 | James      | Kirk      | captain | 555destruct |
+---------+------------+-----------+---------+-------------+
1 row in set (0.01 sec)
```

Deleting Data

- **DELETE FROM** *table* **WHERE** *column*='*value*';
  - Avoid **DELETE FROM** *table*; as it would blow out the whole table

```
mysql> DELETE FROM users WHERE user_id="3";
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM users;
+---------+------------+-----------+---------+-------------+
| user_id | first_name | last_name | login   | password    |
+---------+------------+-----------+---------+-------------+
|       1 | Thomas     | Powell    | tpowell | superSecret |
|       2 | James      | Kirk      | captain | 555destruct |
|       4 | Moe        | Howard    | bossman | whyyou      |
|       5 | Curly      | Joe       | curly7  | woowoo      |
+---------+------------+-----------+---------+-------------+
4 rows in set (0.01 sec)
```

Deleting Data – Big Changes

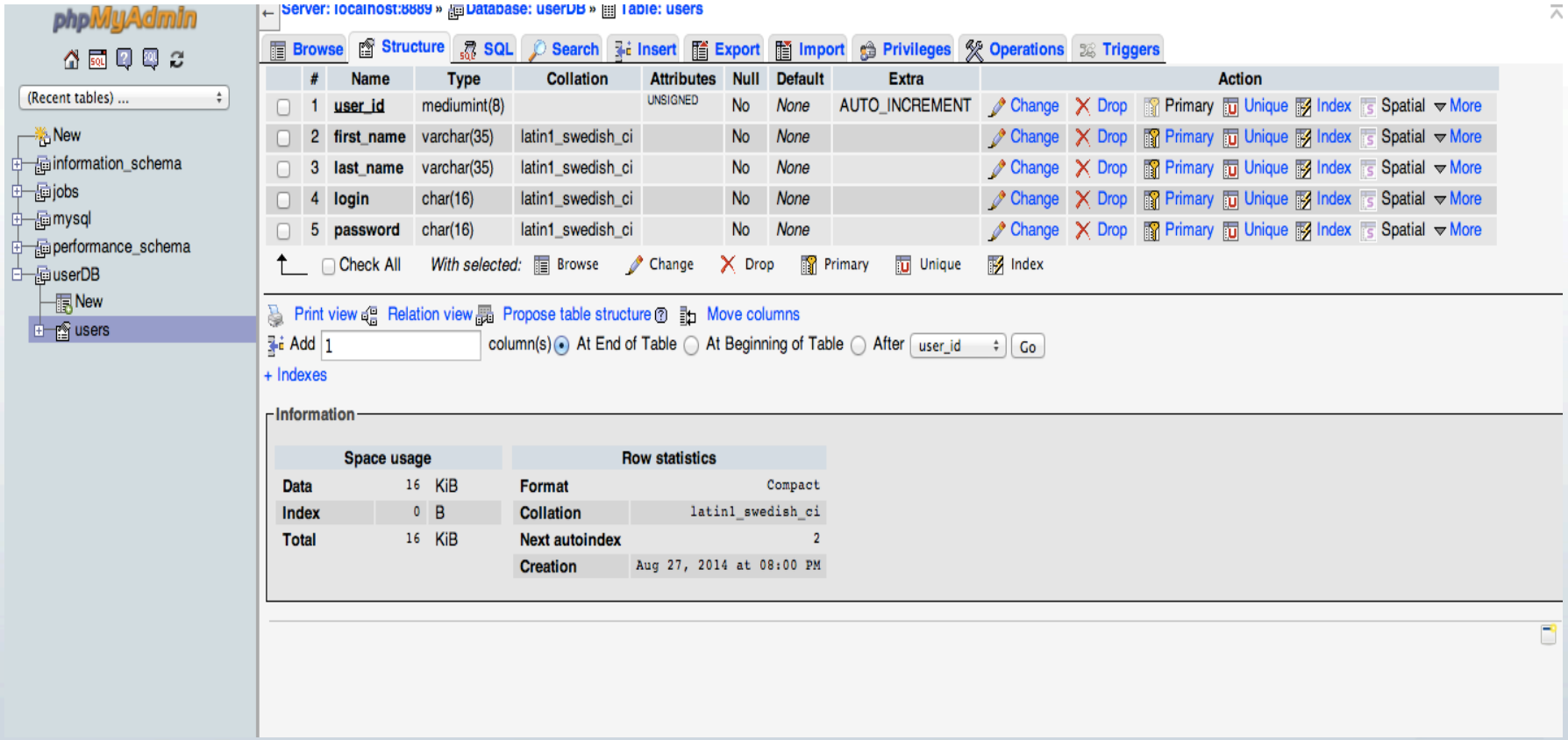
- **DROP TABLE** *tablename;* is the appropriate way to destroy a table and its contents
- **DROP DATABASE** *databasename;* allows you to drop a whole DB (tables and all) – be careful in some shared environments as you may not have permissions to make a replacement database

```
mysql> DROP TABLE users;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from users;
ERROR 1146 (00000): Table 'db154377598.users' doesn't exist
mysql> DROP DATABASE db154377598;
Query OK, 0 rows affected (0.00 sec)
```

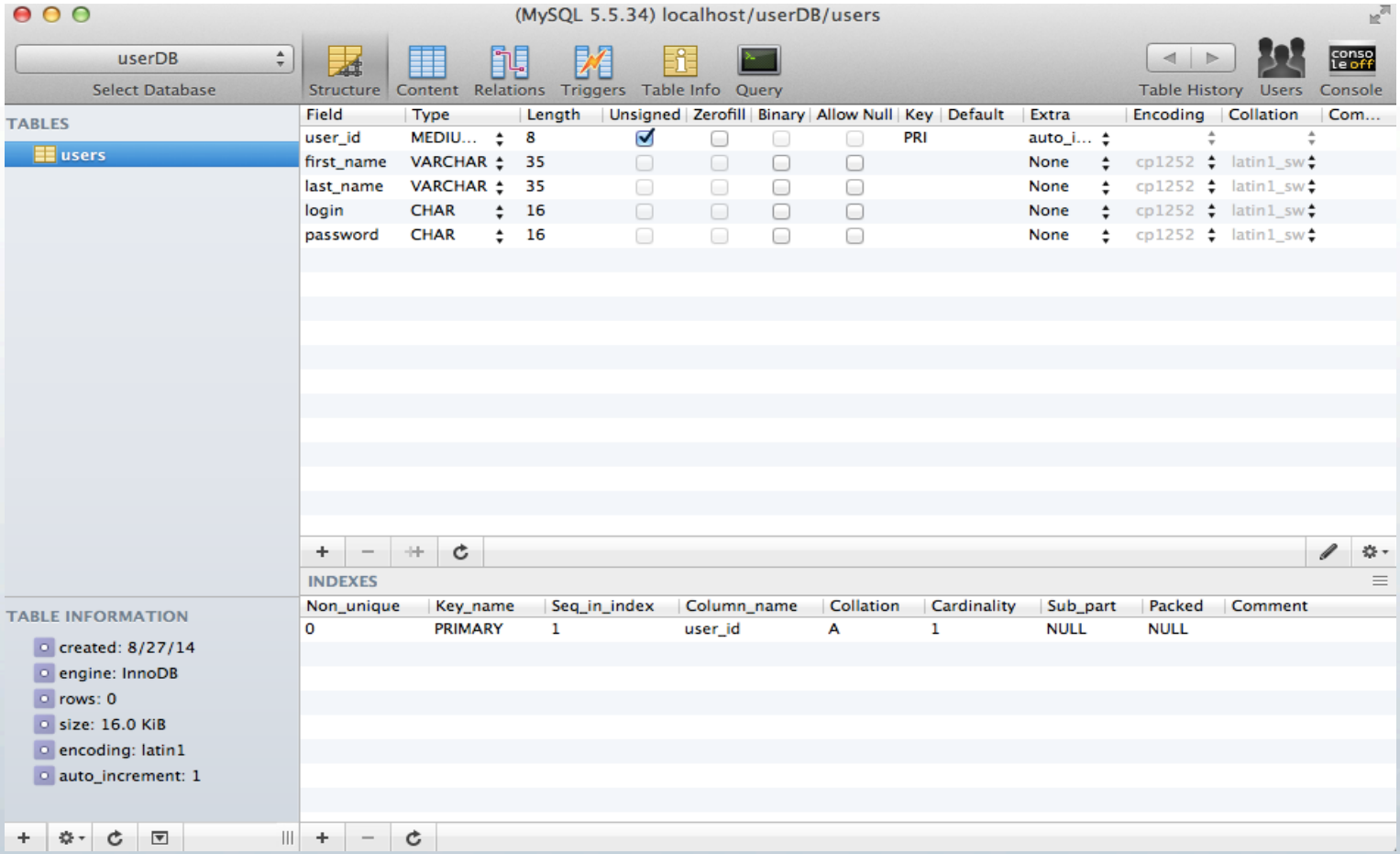
An Easier Way to Admin MySQL

- phpMyAdmin - http://www.phpmyadmin.net



Easier Way

- GUI clients



Talking to mySQL from PHP

- First you need to connect with **mysqli_connect** and the select the database using **mysql_select_db**
  - Better to use some constants to make like easy
  - Add in error checking to be on the same side

```
define('DB_USER','cse135demo');
define('DB_PASSWORD','notsecret');
define('DB_HOST','127.0.0.1');
define('DB_NAME','userDB');

// CONNECT TO DB
$conn = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
```

Talking to mySQL from PHP

- Following our last example in the mySQL monitor let's create a table

```
$createTable = "CREATE TABLE users (user_id MEDIUMINT UNSIGNED
NOT NULL AUTO_INCREMENT,first_name VARCHAR(35) NOT
NULL,last_name VARCHAR(35) NOT NULL,login CHAR(16) NOT
NULL,password CHAR(16) NOT NULL,PRIMARY KEY (user_id))";

$result = @mysqli_query($createTable);
echo "Database create: $result <br />";
```

- Now we can insert some data in a similar fashion

```
 $insertStatement = "INSERT INTO users
(first_name,last_name,login,password) VALUES ('Thomas',
'Powell','tpowell','supersecret');";
 $result = @mysqli_query($insertStatement);
 echo "Database insert: $result <br />";
```

Running a Select Query from PHP

- Once you have done this run a query or other SQL statement

```
$query = 'SELECT login,first_name,last_name FROM
users ORDER BY login';
$result = mysqli_query($query);
```

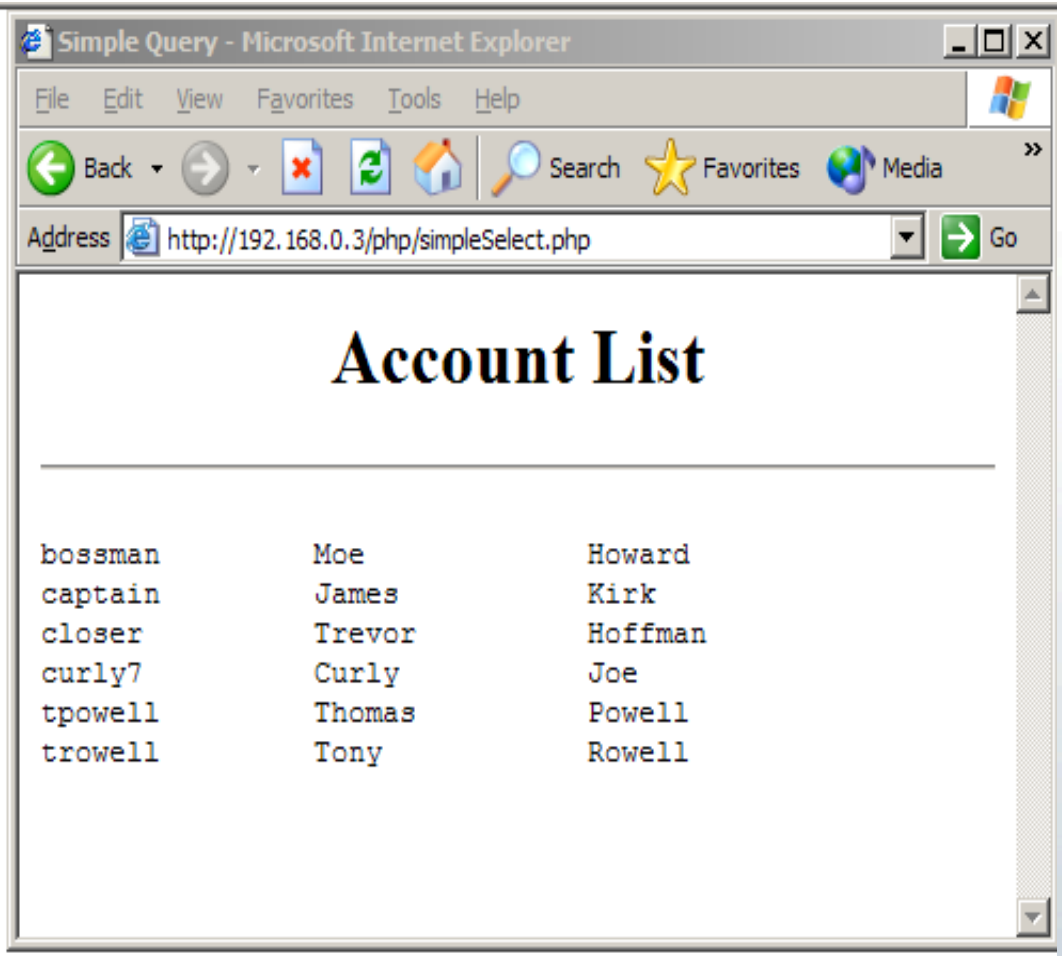
- **After you run a query loop through the result set to output the returned rows one by one, but make sure you address the empty set**

```
$query = 'SELECT login,first_name,last_name FROM users ORDER BY
  login';
   $result = mysqli_query($query);
$numAccts = mysqli_num_rows($result);
print “<pre>”;
if ($numAccts == 0)
  print ("No accts");
else
  while ($row = mysqli_fetch_array($result,MYSQL_NUM))
    print($row[0] ."\t\t". $row[1] ."\t\t". $row[2] ."\n");
print “</pre>”;
```

Running a Select Query from PHP

```
<html><head><title>Simple Query</title></head>
<body><h1 align="center">Account List</h1><hr>

<pre>bossman             Moe             Howard
captain             James           Kirk
closer              Trevor          Hoffman
curly7              Curly           Joe
tpowell             Thomas          Powell
trowell             Tony            Rowell
</pre></body></html>
```



Running a Select Query from PHP

- Using tables for query results is often appropriate

```
$query = 'SELECT login,first_name,last_name FROM users
  ORDER BY login';
    $result = mysqli_query($query);
    $numAccts = mysqli_num_rows($result);
    print "<table>";
    if ($numAccts == 0)
      print ("<tr><td>No accts</td></tr>");
    else
     while ($row = mysqli_fetch_array($result,MYSQL_NUM))
        print("<tr><td>".$row[0] ."</td><td>". $row[1]
."</td><td>". $row[2] ."</td></tr>");
    print "<table>";
```

Delete Items

- An easy pattern to follow is to add delete and edit links next to item with the unique-id for the object embedded. Then we can tie the delete link to call a page that runs a SQL statement to delete
- For example in the example printout use
  - print ("<td>Delete</td>");
- Should these be GETs!? NO!!!! - watch out common issue - ease over correctness?
- Then in deleteitem.php you would have typical DB connection code followed by

```
$query = "DELETE FROM users WHERE
user_id='".$_GET['user_id']."'";
  $result = mysqli_query($query);


  /* bounce back to the list page */
  header('Location: list.php');
```

Update Item

- Similar to the delete pattern we add a link to trigger the edit. This link sends us to a form pre-populated with the DB entry we will update
- We then take this updated text and send to a PHP file that performs the update and bounces us back to the main page

```
$user_id = $_REQUEST['user_id'];
$first_name = $_REQUEST['first_name'];
$last_name = $_REQUEST['last_name'];
$login = $_REQUEST['login'];
$password = $_REQUEST['password'];

$query = "UPDATE users SET first_name='" .$first_name."'
,last_name='".$last_name."' ,login='".$login."'
,password='".$password."' WHERE user_id='".$user_id."'";
$result = mysqli_query($query,$dbc);
```

Add Item

- Similar to the update example we now create a link “Add item” which then points to a blank form and runs an insert command. Given the similarity of add and update you could imagine combining them

```
$first_name = $_REQUEST['first_name'];
  $last_name = $_REQUEST['last_name'];
  $login = $_REQUEST['login'];
  $password = $_REQUEST['password'];

$query = "INSERT INTO users (first_name,last_name,login,password)
    VALUES ('$first_name' , '$last_name' , '$login' , '$password')";
$result = mysqli_query($query,$dbc);

/* bounce back to the list page */
header('Location: list.php');
```

Adding More

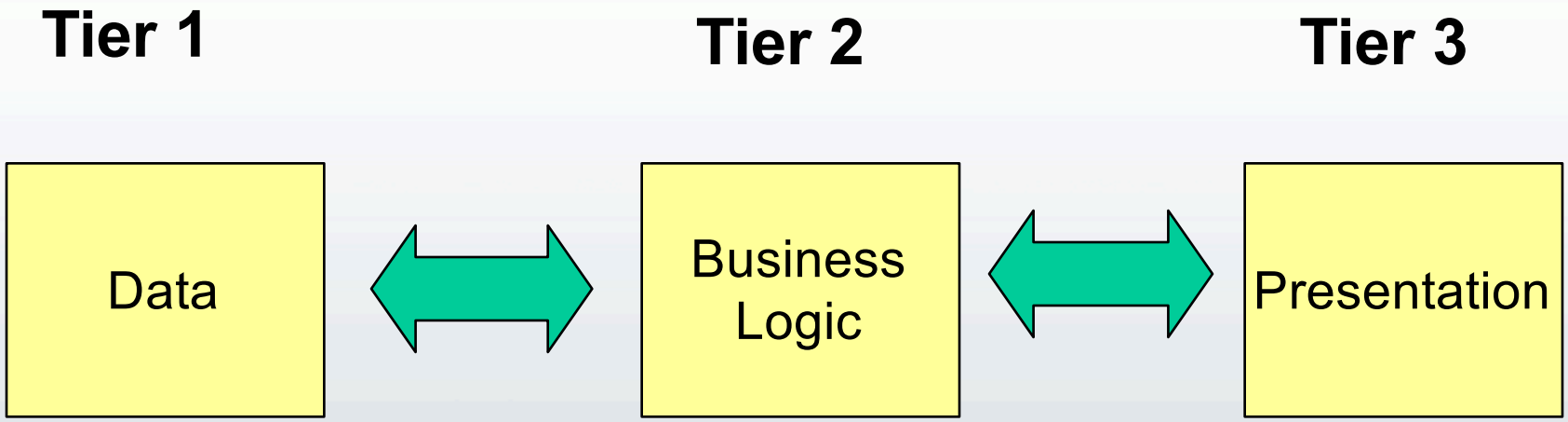
- One common feature is to add sorting by columns
- Another possibility is to add paging of results
  - Show by 5, 10, etc.
- Visually you might do zebra stripping for readability – alternate row colors
- You also might want to integrate validation and the various JavaScript usability improvements we have been discussing along the way

Web Site Template Systems

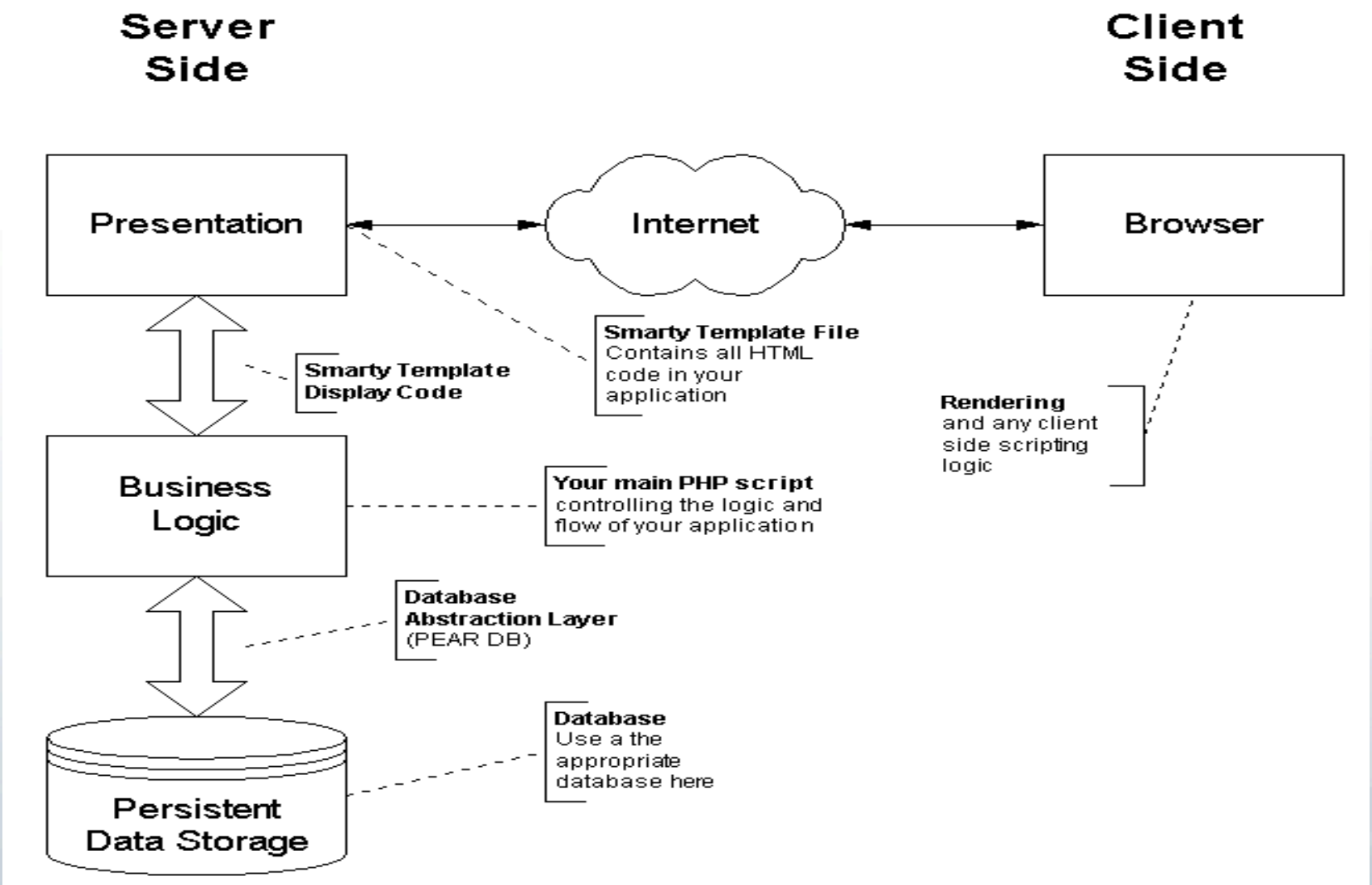
Goals of a Three Tier Web Architecture

- There can be numerous benefits to a well designed n-tier Web site/application
  - Each tier could be run on a separate machine(s) leading to better performance?
  - Better structure leading to better maintenance
  - Better division of labor due to structure also leading to improved maintenance
- Of course like any design pattern it is very possible to cause yourself great trouble if it is poorly applied or shudder simply not as necessary as some would have you believe
  - A "cargo cult" problem often seems to abound in programming at times
    - Feynman speech
      http://en.wikipedia.org/wiki/Cargo_cult_science
    - Read application to SE by McConnel
      http://www.stevemcconnell.com/ieeesoftware/eic10.htm

Typical 3-Tier Separation



Typical 3-Tier Separation in PHP



Database Abstraction Layer

- In the case of PHP there are lots of choices to abstract for example
  - http://pear.php.net/package/MDB2
- The advantage is to abstract away certain details of an underlying database
  - Portability
    - Allow for easy switching of DB
    - Provide for easier code distribution
  - Provide more of an OO approach to the DB connections in PHP
  - At what cost? Remember your tradeoffs!

Template Example: Smarty

- Numerous template systems out there for PHP with the most notable being Smarty http://www.smarty.net/
- Reasons promoted for using template langs like Smarty
  - Big one: Separate code from template
  - Caching of templates, white space striping, other delivery prepping facilities
  - Avoid using PHP for presentation use a less powerful language and thus improve security (really?)
- Note: There are also libraries for specialized HTML output that you might want to take a look

Smarty Hello World

```
<?php

// My path to Smarty
require ('./smarty/libs/Smarty.class.php');

$smarty = new Smarty;

// Set a path to your templates
$smarty->template_dir = './templates/';

$smarty->assign('Username', 'Mr. Smarty');
$smarty->display('index.tpl');

?>
```

Smarty Hello World Template

```
<html>
<head>
<title>Hello Smarty</title>
</head>
<body>

Hello world {$Username}!

</body>
</html>
```

Smarty DB Example

```
<?php
    require ('./smarty/libs/Smarty.class.php');
    $smarty = new Smarty;
    $smarty->template_dir = './templates/';


    define('DB_USER','root'); define('DB_PASSWORD','sniggle');
    define('DB_HOST','localhost');define('DB_NAME','bookmarks2');


    $dbc = mysql_connect(DB_HOST,DB_USER,DB_PASSWORD) OR die('Cound not connect to
    MySQL: '.mysql_error());
    @mysql_select_db(DB_NAME) OR die('Could not select database: '.mysql_error());
     $query = "SELECT name,url,description FROM bookmarks";
     $result = mysqli_query($query);


     $bookmarks = array();
     $i=0;
     while ($row=mysqli_fetch_array($result)) {
        $abookmark = array('name' => $row['name'],'url' => $row['url'],'description' =>
    $row['description']);
        $bookmarks[$i++] = $abookmark;
     }


    $smarty->assign('results', $bookmarks);
    $smarty->display('displaybookmarks.tpl');
?>
```

Displaybookmarks.tpl

```
<!DOCTYPE html>
<html>
<head>
<title>List Bookmarks Smarty Style</title>
<link rel="stylesheet" href="/styles/main.css”>
</head>
<body>
<h1>Bookmarks Smarty Style</h1>
<hr>
<table cellspacing="0" width="75%">

<tr>
 <th>Name</th> <th>URL</th> <th>Description</th>
</tr>
```

Displaybookmarks.tpl Contd.

```
{section name=nr loop=$results}
 <tr {if $smarty.section.nr.iteration is odd} bgcolor="#efefef"{/if}>
      <td>
         {$results[nr].name}
      </td>
      <td>
         {$results[nr].url}
      </td>
      <td>
         {$results[nr].description}
      </td>
 </tr>
{/section}

</table>
</body>
</html>
```

Pretty cool…but

- Some developers are skeptical of template systems like Smarty
  - Have to learn a new language
    - Is it that much different than just plain PHP?
      - { } instead of <? ?>
  - Can't you just do it in PHP anyway?
    - Maybe you can

Notsosmarty.php

```
<?php
    /* Global DB settings */
define('DB_USER','root');define('DB_PASSWORD','sniggle');define('DB_HOST','l
   ocalhost');define('DB_NAME','bookmarks2');


    $dbc = mysqli_connect(DB_HOST,DB_USER,DB_PASSWORD,DB_NAME) OR die('Cound
   not connect to MySQL: '.mysql_error());
    $query = "SELECT name,url,description FROM bookmarks";
     $result = mysqli_query($query);


     $bookmarks = array();
     $numbookmarks=0;
     while ($row=mysqli_fetch_array($result))  {
        $abookmark = array('name' => $row['name'],'url' =>
   $row['url'],'description' => $row['description']);
        $bookmarks[$numbookmarks++] = $abookmark;
     }


    // Now pass the results to the template
    include('./templates/displaybookmarks2.tpl');
?>
```

A Template without Smarty

```
<!DOCTYPE html>
<html>
<head>
<title>List Bookmarks Not Smarty Style</title>
<link rel="stylesheet" href="/styles/main.css" media="screen" />
</head>
<body>
<h1>Bookmarks Not So Smarty Style</h1>
<hr>

<table cellspacing="0" width="75%">

<tr>
 <th>Name</th> <th>URL</th> <th>Description</th>
</tr>
```

A Template without Smarty Contd.

```
<?php for ($nr = 1; $nr < $numbookmarks; $nr++) {  ?>
 <tr <?php if ($nr % 2 != 0) print('bgcolor="#efefef"');?> >
      <td>
         <?=$bookmarks[$nr]['name'] ?>
      </td>
      <td>
         <?=$bookmarks[$nr]['url'] ?>
      </td>
      <td>
         <?=$bookmarks[$nr]['description'] ?>
      </td>
 </tr>
<?php } ?>

</table>
</body>
</html>
```

Template Conclusions

- "… the point of template engines should be to separate your business logic from your presentation logic, not separate your PHP code from your HTML code."

- Still the idea of templates is still a good idea, despite how it may be aggressively implemented
  - Easier to "re-skin" an application
  - Helps divide work from programmers and designers to a degree

- Don't get carried away with the separation of presentation and logic
  - Presentation may have logic associated with it
    - Particularly if you are trying to do an adaptive interface for mobile device

Switching to High Gear

- Since we build so many CRUD applications a number of efforts have been made to build them faster and easier
  - Examples: Rails framework for Ruby, Cake for PHP, Sails for Nodejs, Meteor, etc.
- These frameworks often contain tremendous demo deception and hide the tradeoffs made
  - Don't interpret this as me not wanting to use them - interpret this as knowing what it is being provided and what you will have to address

From Table to Object

- ORM
  - From the all knowing Wikipedia - "**Object-Relational Mapping** (aka **ORM**, **O/RM**, and **O/R mapping**) is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages."
  - Lots of implementations exist
    - http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software
- Example:
  - In Cake take a Database table named "Pets" that has columns like name, type, breed, color and a primary key and an ORM system will create a relation to an object called "Pet" with similar properties
  - Question: What's the primary key? Answer: Convention for most systems is "id"

Scaffold

- In the real world
  - "**Scaffolding** is a temporary framework used to support people and material in the construction or repair of buildings and other large structures."

- In the Web and programming world
  - Generally term given to a quick and dirty framework built by inspecting database tables or various config files
  - Example: In Ruby you might create the scaffolds with

```
ruby script\generate model Pet
ruby script\generate controller Pet
```

or just `ruby script\generate scaffold Pet`

Quick Crud

- With scaffold in hand you likely have a list, edit, add, and delete page that allows you to make changes to the DB
- You also will find that a change to the DB can be quickly recognized in the app
  - Sometimes automatically sometimes via a trigger to re-scaffold
- Now you have to go and improve the views to make it look pretty and add any other customized business logic desired

There's the rub…

- So you may have to eventually do plenty of work to:
  - Get the pretty templates in place
  - Do anything out of the ordinary at all

- I also wonder if we do the same thing with a "wizard" to make a sample application would you have been impressed?

- One thing then you might say that programs built with such systems really have going for them is that they will be familiar to another programmer familiar with that system, but then again we have another name for that – *a coding standard*.
  - Final Consideration – what if you don't know the standard?