CS 455
Principles of Database Systems
Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion
PHP Hypertext Preprocessor

- **PHP Hypertext Preprocessor**
  - Created by Rasmus Lerdorf in 1994
  - The first web-programming language
  - Formerly Personal Home Page Tools

- Today: Runs on > 75% of web servers
  - 8th most widely-used language
    (IEEE Spectrum, 2017)
How PHP Processing Works

Hypertext Transfer Protocol (HTTP)

- Connect port 80
- ACK
- GET /path/to/foo.php
- (contents of foo.php)
- (connection closes)

Browser (Client)

Web Server

PHP processor

HTML + PHP

HTML
Quick Guide

- Variable names start with $:

  ```php
  $var = expression;
  ```

- Getting info on variables:

  ```php
  var_dump($var);
  ```

- Printing:

  ```php
  echo expression;
  ```
Quick Guide (Cont.)

- PHP files should end in .php
  - HTML code *can co-exist* in a .php file

- Any PHP code must be enclosed in <?php ... ?> tags.
  - All other code will be interpreted as HTML!

```php
<?php $title = "David's Page"; ?>
<head>
  <title> <?php echo $title; ?> </title>
</head>
```
PHP Primitives (Boolean)

- PHP variables are *dynamically typed* and do not need to be declared
  - A variable's type is determined at **runtime**!
- They could be..  **boolean**, **int**, **float**, **string**
- Boolean Example:

```html
<html>
<p>
<?php
$largeFont = True;  //case-insensitive
if ($largeFont)
    echo '<font size="20">';
else
    echo '<font size="14">';
?>
Hello world!<br/>
</font>
</p>
</html>
```
PHP Primitives (numerics)

- **Types:** boolean, int, float, string

  - **Integers:**
    ```php
    <?php
    $a = 1234; // decimal number
    $a = -123; // a negative number
    $a = 0x1A; // hexadecimal number (equivalent to 26 decimal)
    $a = 0b1111111; // binary number (equivalent to 255 decimal)
    ?>
    ```

  - **Floats (double-precision):**
    ```php
    <?php
    $a = 1.23456789;
    $b = 1.23456780;
    $epsilon = 0.00001;
    
    // always use this instead of: if ($a == $b)
    if (abs($a - $b) < $epsilon) {
        // do something useful
    }
    ?>
    ```
PHP Primitives (strings)

- **Types:** boolean, int, float, string

- **Single-quoted Strings:** Behaves like Strings in Java

```php
<?php
$var = "cool!";
echo 'I said, "$var"';  // I said, "$var"
?>
```

- **Double-quoted Strings** evaluates variables! NICE!

```php
<?php
$var = "cool!"
echo "I said, "$var";  // I said, "cool!"
?>
```

- **Concatenation:** $str1 . $str2
  - Str1 .= $str2;  // works too!
Arrays

- Arrays in PHP are basically *hash maps*

```php
<?php
$my_arr = array(
    "foo" => "bar",
    "bar" => "foo",
    0 => 9,
);
$my_arr[1] = 'moo!'

var_dump($my_arr);
?>
```

```
array(4) {
    ["foo"] =>
    string(3) "bar"
    ["bar"] =>
    string(3) "foo"
    [0] =>
    int(9)
    [1] =>
    string(4) "moo!"
}
```

This is the output from the above code
Type Juggling

- As mentioned before, PHP is dynamically typed
  - Known as *Type Juggling* in PHP lingo

```php
<?php

$number_of_toys = 10;
$toys_category = "123 Puzzles";
$toys_age_limit = "5.5";
$toys_price = "2e2";

$result1 = $number_of_toys + $toys_category;
$result2 = $number_of_toys + $toys_age_limit;
$result3 = $number_of_toys + $toys_price;

echo $result1."<br/>";
echo $result2."<br/>";
echo $result3."<br/>";

?>
```

## PHP Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a == $b</td>
<td>Equals after type juggling</td>
</tr>
<tr>
<td>$a === $b</td>
<td>Equals, <strong>and</strong> are of the same data type</td>
</tr>
<tr>
<td>$a != $b</td>
<td>Not equals after type juggling</td>
</tr>
<tr>
<td>$a !== $b</td>
<td>Not equals, or are of different types</td>
</tr>
<tr>
<td>$a &lt; $b</td>
<td>Less than?</td>
</tr>
<tr>
<td>$a &gt; $b</td>
<td>Greater than?</td>
</tr>
<tr>
<td>$a &lt;= $b</td>
<td>Less than equals, after type juggling</td>
</tr>
<tr>
<td>$a &gt;= $b</td>
<td>Greater than equals, after type juggling</td>
</tr>
</tbody>
</table>
Comparison Operators (Cont.)

```
$foo = 10;

var_dump($foo == 10);  //true
var_dump($foo == '10'); //true!
var_dump($foo === 10); //true
var_dump($foo === '10'); //false!
var_dump($foo <= '10'); //true!
```
### Operations

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -, *, **, /, %</td>
<td>(Usual num ops) var_dump(2**3); //8</td>
</tr>
<tr>
<td>.</td>
<td>String concatenation var_dump('foo' . 'bar' . 88) //foobar 88</td>
</tr>
<tr>
<td>&amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td>$a++, ++$a</td>
<td>(Usual num ops)</td>
</tr>
<tr>
<td>$a--, --$a</td>
<td>(Usual num ops)</td>
</tr>
<tr>
<td>+=, -=, *=, /=, **=</td>
<td>(Usual num ops)</td>
</tr>
<tr>
<td>.=</td>
<td>String concat</td>
</tr>
</tbody>
</table>
Conditionals

- If-then-else

```php
<?php
if (cond) {
    echo "That was <b>true</b>\n";
}
else {
    echo "That was <b>false</b>\n";
}
?>
```

- Integration with HTML (same result as above)

```php
<?php
if (cond) { ?>
That was <b>true</b>
<?php }
else { ?>
That was <b>false</b>
<?php }
?>
```
Conditionals Else-If

- Else-Ifs

```php
if (cond) {
    //statement
}
elseif (cond) {
    //statement
}
elseif (cond) {
    //statement
}
else {
    //statement
}
```
Loops (For & While)

- While and For loops also have familiar syntax

```php
<?php
while (cond) {
    //loop statements
}
for (init; cond; progress) {
    //loop statements
}
?>
```
Loops (Cont.)

- Loops can also integrate with HTML

```php
<?php $n = 5; ?>
<ul>
<?php
    for ($i = 0; $i < $n; $i++) {
        echo "<li>List item: $i</li>
    
    ";
    }
?>
</ul>
```

- Output:

```html
<ul>
    <li>List item: 0</li>
    <li>List item: 1</li>
    <li>List item: 2</li>
    <li>List item: 3</li>
    <li>List item: 4</li>
</ul>
```
Arrays

- Recall: all PHP arrays are actually associative arrays (or HashMaps)
  - Created with the `array(...)` function

```php
$list = array(
    "foo" => "bar",
    "bar" => true,
    9 => 4,
    0 => "bla",
);
```

- Accessed as expected...

```php
var_dump($list["foo"]); // string(3) "bar"
var_dump($list[9]);    // int(4)
var_dump($list[8]);    // NULL
```
Arrays (Cont.)

- Single command to print out all contents of array: `print_r($list)`
  - Good for debugging, but not much else
  - Output:

```php
Array
(
    [foo] => bar
    [bar] => foo
    [9] => 4
    [0] => bla
)
```
Array Access (Foreach loop)

- How to access elements in an associative array?
  - No standard index... so how do we know how to loop?

- If you don't care about the array index:

```php
foreach (array_expression as $value) {
    //statement
}
```

- If you want the array index:

```php
foreach (array_expression as $key => $value) {
    //statement
}
```
Foreach Loops

<?php
$list = array(
    "foo" => "bar",
    "bar" => true,
    9 => 4,
    0 => "bla",
);

foreach ($list as $k => $v) {
    echo "$k holds $v\n";
}
?>

Output:

foo holds bar
bar holds true
9 holds 4
0 holds bla
Functions

- Functions in PHP are defined as follows:
  - Notice: no return type; just return when needed

```php
<?php
function functionName(paramList) {
    //body
}
?>
```

- Example:

```php
<?php
function max($a, $b) {
    if ($a < $b)
        return $b;
    return $a;
}

echo "The larger of 4 and 5 is: ". max(4,5); // call the function
?>
```
Good Practice

- Put related functions in their own file, then include as needed.

```php
<?php
function func0(params) {
    //body
}
//...
?>
```

```php
<?php
include "myfuncs.php";
func0(...);
?>
```

```php
<?php
function dbConnect(params) {
    //body
}
function dbQuery(params) {
    //body
}
?>
```

```php
<?php
include "myfuncs.php";
include "myDBfuncs.php";

dbQuery(...);
?>
```
Outline

- History of the Web
- Introduction to HTML
- **Dynamic Web Programming with PHP**
  - PHP Basics
  - Superglobals
  - PDO Database Connectivity
- Conclusion
Variable Scope in PHP

- Different PHP blocks *within* file:

```php
<?php
    $x = 100;
?>

<!-- some HTML -->

<?php
    echo $x;  // This works
?>

<?php
    echo $x;  // $x is not defined
?>
```

- Across file:

```php
<?php
    $x = 100;
?>
```

```php
<?php
    echo $x;  // $x is not defined
?>
```

File1.php

File2.php
Variable Scope in PHP

- Across file (using include):

```php
<?php
    $x = 100;
?>

<?php
    include "File1.php";
    echo $x;  // This works again
?>
```
PHP Superglobals

- **Superglobals** are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

Here are a few important ones:

- `$_GLOBALS[...]`: user-defined (think public static variables in Java)
- `$_COOKIE[...]`: cookies (variables) we set on the client (browser)
- `$_GET[...]`: variables passed from URLs
- `$_POST[...]`: variables passed from HTML forms
- `$_SERVER[...]`: information about the web server
PHP Superglobals

- **Superglobals** are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

Here are a few important ones:

- `$_GLOBALS[...]`: user-defined (think public static variables in Java)
- `$_COOKIE[...]`: a cookie we set on a browser
- `$_GET[...]`: variables passed from URLs
- `$_POST[...]`: variables passed from HTML forms
- `$_SERVER[...]`: information about the web server
Cookies

- If HTTP is stateless, how do sites like Amazon and Facebook remember that I'm logged in?

- *Cookies* are data that websites can store on your browser so that it can remember you in a later HTTP session.

- PHP has built-in cookie handling mechanisms
Setting Cookies

- Setting a cookie (browser has to accept them)
  - Caveat: Cookies are a part of the HTTP header, and **must be set before** any other content is sent to the browser

```php
<?php
    setcookie("userID", "dchiu", time() + (86400 * 30)); // 86400 = 1 day
?>
<!DOCTYPE html>
<html>
    <!-- blah blah blah -->
</html>
```

**Expiration (duration):** Time from now in seconds. Value of 0 means end of session (when browser closes)
Later, a user browses back to your web page... to remember who they are, we need to see if the **userID** cookie is set!

Enter the `$_COOKIE[...]` superglobal

```php
<?php
    // do we know this user?
    if (isset($_COOKIE['userID'])) {
        $firstName = getName($_COOKIE['userID']);
        echo "Welcome back $firstName!";
    }
    else {
        // don't know this person (or cookie expired)
        printLoginForm(); // make them login again
    }
?>
```
Superglobals are variables that are accessible in all scopes.

- They are all associative arrays (hashmaps)

Here are a few important ones:

- $_GLOBALS[...]: user-defined (think public static variables in Java)
- $_COOKIE[...]: a cookie we set on a browser
- $_GET[...]: variables passed from URLs
- $_POST[...]: variables passed from HTML forms
- $_SERVER[...]: information about the web server
Ever wonder what ?, = and & mean in a URL?
What's in a URL?

- **URL Syntax**

  \[\text{protocol:}[/[/\text{user:password@}]\text{host}[::\text{port}]][/]\text{path}[/?\text{query}][\#\text{fragment}]\]

- **Examples:**
  - Locates a file on my local machine
    \[\text{file://localhost/Users/David/Documents/foo.txt}\]
  - Locates a directory on another machine using FTP
    \[\text{ftp://ftp.at.debian.org/debian-cd/8.2.0/i386/iso-dvd}\]
What's in a URL? (Cont.)

- URL Syntax

```
protocol://[user:password@]host[:port][/path[?query][#fragment]]
```

- Examples:
  - Get Lecture 1 from my course page (login automatically)
    
    `http://CS455:p4ssword@cs.pugetsound.edu/~dchiu/CS455/notes/CS455_1-intro.pdf`

  - Sends a "query" (i.e., variables) to the server
    
Inside showGetvars.php

- Just use the $_GET[...] superglobal to access any variable and its value that was passed via URL!

- showGetvars.php:

```php
<!DOCTYPE html>
<html>
<body>
<?php
    if ($_GET['foo'] == 10)
        echo "Foo!";
    if ($_GET['bar'] == 20)
        echo "Bar!";
?>
</body>
</html>
```
PHP Superglobals

- *Superglobals* are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

- Here are a few important ones:
  - $_GLOBALS[...]: user-defined (think public static variables in Java)
  - $_COOKIE[...]: a cookie we set on a browser
  - $_GET[...]: variables passed from URLs
  - $_POST[...]: variables passed from HTML forms
  - $_SERVER[...]: information about the web server
HTML Forms

- You can make forms with HTML:

  Where does it take you when you click the `submit` button?
  Which HTTP method to use to send data?
  Possible values: `post` or `get` *(USE POST ALWAYS)*

```html
<form action="formHandler.php" method="post">
  Name: <input type="text" name="name"/><br/>
  E-mail: <input type="text" name="email"/><br/>
  <input type="submit"/>
</form>
```

The `submit` button
HTML Forms (Cont.)

- You can make forms with HTML:

```html
<form action="formHandler.php" method="post">
  Name: <input type="text" name="name" /><br/>
  E-mail: <input type="text" name="email" /><br/>
  <input type="submit" />
</form>
```

Draw a textbox

Name of the variable
HTML Forms (Cont.)

- **Password Field**

  ```html
  Enter your password: <input type="password" name="pwd"/>
  ```

- **Checkbox**

  ```html
  Today I am:<br/>
  <input type="checkbox" name="happy"/> Happy<br/>
  <input type="checkbox" name="angry"/> Angry<br/>
  <input type="checkbox" name="sad"/> Sad<br/>
  ```

- **Dropdown List**

  ```html
  <select name="country">
    <option value="ca">Canada</option>
    <option value="zn">China</option>
    <option value="fr">France</option>
    <option value="in">India</option>
    <option value="us" selected>U.S.</option>
  </select>
  ```
HTML Forms (Cont.)

- **File**
  
  ```html
  Upload a file:<br/>
  <input type="file" name="filename"/>
  ```

- **Hidden**
  
  ```html
  <input type="hidden" name="var" value="val" />
  ```

- **Radio Options**
  
  ```html
  Your pet is a:<br/>
  <input type="radio" name="species" value="cat"/> Cat<br/>
  <input type="radio" name="species" value="dog"/> Dog<br/>
  <input type="radio" name="species" value="fish"/> Fish<br/>
  <input type="radio" name="species" value="lizard"/> Lizard<br/>
  ```
We need a (PHP) script to process the form data!

- The superglobal $_POST[...] hold all those variables from the form
  - Assuming you used the "post" method in your form

```php
<?php
var_dump($_POST[name]);
var_dump($_POST[email]);
?>
```

- Typically, this PHP script would insert the collected data into a database...
Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion
PHP Database Connectivity

- There are many free PHP database libraries...
  - We focus on PHP Data Objects (PDO)
  - Need to be installed as an add-on library to PHP

- From your (Ubuntu) Linux shell:
  
  ```
  $ sudo apt-get install php-pdo
  
  or (if on CentOS)
  
  $ sudo yum install php-pdo
  ```

- PDO is not the only way... other libraries exist
Assumptions

‣ Caveat: This tutorial written for SQLite3

‣ Assumptions:
  • SQLite3 database already exists on filesystem (i.e., you used .save or .backup to create the file)
  • Apache web server needs write access to both the database file and the directory where it's located

‣ The PDO library is object-oriented. Pro-tip:

```php
$obj = new Class(..);  //instantiation
$obj->method(..);  //method call
```
(Dis)Connecting to/from the Database

PDO Object Instantiation:

```php
<?php
try {
    // open the sqlite database file
    // assumes airport.db is in the myDB directory and has read/write permissions
    $db = new PDO('sqlite:./myDB/airport.db');

    // Set errormode to exceptions
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // >>> queries and stuff right here <<<

    // disconnect from database
    $db = null;
}
catch(PDOException $e) {
    die('Exception : '.$e->getMessage());  // die will quit the script immediate
}
?>
```
Important: Set Permissions on DB

- Say you're in the document root directory...

```bash
$ sudo chown -R apache myDB/
$ sudo chmod -R 755 myDB
$ sudo chmod -R 700 myDB/airport.db
$ ls -l
```

```bash
total 88
-rw-r--r-- 1 dchiu 10000 285 Feb 27 2015 cookieread.php
-rw-r--r-- 1 dchiu 10000  70 Feb 27 2015 formHandler.php
-rw-r--r-- 1 dchiu 10000 519 Feb 27 2015 form.php
-rw-r--r-- 1 dchiu ctweb03-access 2048 Jul 26 15:53 hi.db
-rw-r--r-- 1 dchiu ctweb03-access  417 Oct 11 21:31 insert.html
-rw-r--r-- 1 dchiu ctweb03-access  698 Oct 11 21:55 insertPassenger.php
-dwxr-xr-x 2 apache ctweb03-access 4096 Oct 11 21:58 myDB
-rw-r--r-- 1 dchiu 10000  332 Oct 10 14:56 setcookie.php
-rw-r--r-- 1 dchiu ctweb03-access  165 Oct 10 16:17 showGetvars.php
-rw-r--r-- 1 dchiu ctweb03-access 321 Oct 14 2015 showPassengers.html
-rw-r--r-- 1 dchiu ctweb03-access  752 Oct 11 21:56 showPassengers.php
```

```bash
$ ls -l myDB/
```

```bash
total 64
-rwx------ 1 apache ctweb03-access 28672 Oct 11 21:58 airport.db
```
"Read" Queries: Select

- With select, we don't care about number of rows affected, we want the result set that was returned!

- **Syntax:**
  ```php
  public PDOStatement query(string $statement)
  ```

- **Return Value:** An array of tuples
  - Each tuple is an associative array of `attribute => value` pairs

```php
//select all passengers
$result = $db->query('SELECT * FROM passengers;');

foreach($result as $tuple) {
    echo "$tuple[ssn] $tuple[f_name] $tuple[l_name] <br/>";
}
```
<!DOCTYPE html>
<html>
<body>
<h2>List of all passengers</h2>
<?php
    try {
        // open the sqlite database file
        $db = new PDO('sqlite:./myDB/airport.db');
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        // select all passengers
        $query = "SELECT * FROM passengers";
        $result = $db->query($query);

        // loop through each tuple in result set
        foreach($result as $tuple) {
            echo "<font color='blue'>$tuple[ssn]</font> $tuple[f_name] $tuple[m_name] $tuple[l_name]<br/>
";
        }
    } catch(PDOException $e) {
        die('Exception : '.$e->getMessage());
    }
?>
</body>
</html>
## List of all passengers

<table>
<thead>
<tr>
<th>Phone Number</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-11-1111</td>
<td>Homer J Simpson</td>
</tr>
<tr>
<td>444-44-4444</td>
<td>Bart H Simpson</td>
</tr>
<tr>
<td>222-22-2222</td>
<td>Lisa G Simpson</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>Frank Lovejoy</td>
</tr>
<tr>
<td>666-66-6666</td>
<td>Robert N Quimby</td>
</tr>
<tr>
<td>777-77-7777</td>
<td>Ned T Flanders</td>
</tr>
<tr>
<td>333-33-3333</td>
<td>Frank Ryerson</td>
</tr>
<tr>
<td>000-00-0000</td>
<td>Test t Testing</td>
</tr>
<tr>
<td>000-00-1234</td>
<td>Test t Testing</td>
</tr>
</tbody>
</table>
"Write Queries" (Insert, Delete, Update)

- Use this: `public int exec(string $statement)`
  
  - Executes given SQL statements and returns number of affected rows

```php
<?php
try {
    $db = new PDO('sqlite:./myDB/airport.db');
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    //insert some new tuples into the passenger relation
    $db->exec("insert into passengers values ('David', NULL, 'Chiu', '888-88-8888');");
    $db->exec("insert into passengers values ('Brad', NULL, 'Richards', '999-99-9999');");

    //now put Brad and David on the same flight
    $db->exec("insert into onboard values ('888-88-8888',4,'32B')");
    $db->exec("insert into onboard values ('999-99-9999',4,'32C')");

    //disconnect from database
    $db = null;
}
catch(PDOException $e) {
    die('Exception : '.$e->getMessage());
}
?>
```
How to Get Form Data from Users

- See insert.html on course page:

```
<!DOCTYPE html>
<html>
<head>
  <title>Insert Passengers</title>
</head>
<body>
  <p>
    <form action="insertPassenger_insecure.php" method="post">
      SSN: <input type="text" name="form_ssn" /><br/>
      First Name: <input type="text" name="form_fname" /><br/>
      Middle Name: <input type="text" name="form_mname" /><br/>
      Last Name: <input type="text" name="form_lname" /><br/>
      <input type="submit"/>
    </form>
  </p>
</body>
</html>
```

- Demo: http://cs.pugetsound.edu/~dchiu/cs455/webstuff/insert.html
How to Get Form Data from Users

- See insert.html on course page:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Insert Passengers</title>
</head>
<body>
  <form action="insertPassenger_insecure.php" method="post">
    SSN: <input type="text" name="form_ssn" />
    First Name: <input type="text" name="form_fname" />
    Middle Name: <input type="text" name="form_mname" />
    Last Name: <input type="text" name="form_lname" />
    <input type="submit"/>
  </form>
</body>
</html>
```

- Demo: http://cs.pugetsound.edu/~dchiu/cs455/webstuff/insert.html
How to Get Form Data from Users

- See insert.html on course page:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Insert Passengers</title>
</head>
<body>
    <p>
        <form action="insertPassenger_insecure.php" method="post">
            SSN: <input type="text" name="form_ssn" /><br/>
            First Name: <input type="text" name="form_fname" /><br/>
            Middle Name: <input type="text" name="form_mname" /><br/>
            Last Name: <input type="text" name="form_lname" /><br/>
            <input type="submit"/>
        </form>
    </p>
</body>
</html>
```

Uses the HTTP POST command to send values to apache

- Demo: [http://cs.pugetsound.edu/~dchiu/cs455/webstuff/insert.html](http://cs.pugetsound.edu/~dchiu/cs455/webstuff/insert.html)
How to Get Form Data from Users

- See insert.html on course page:

```html
<!DOCTYPE html>
<html>
<head>
  <title>Insert Passengers</title>
</head>
<body>
  <p>
    <form action="insertPassenger_insecure.php" method="post">
      SSN: <input type="text" name="form_ssn" />
      First Name: <input type="text" name="form_fname" />
      Middle Name: <input type="text" name="form_mname" />
      Last Name: <input type="text" name="form_lname" />
      <input type="submit" />
    </form>
  </p>
</body>
</html>
```

Form input can be accessed with PHP's $_POST[name] superglobals e.g., $_POST[form_ssn]

- Demo: http://cs.pugetsound.edu/~dchiu/cs455/webstuff/insert.html
<?php
    try {
        //open the sqlite database file
        $db = new PDO('sqlite:./myDB/airport.db);
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        //insert the passenger (UNSAFE!)
        //order matters (look at your schema) -- fname, mname, lname, ssn
        $stmt = "INSERT INTO passengers VALUES
            ('$_POST[form_fname]', '$_POST[form_mname]', '$_POST[form_lname]', '$_POST[form_ssn]');";
        $db->exec($stmt);

        //disconnect from database
        $db = null;
    }
    catch(PDOException $e)
    {
        die('Exception : '.$e->getMessage());
    }

    //redirect user to another page
    header("Location: showPassengers_secure.php");
?>
Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
    - Dealing with the SQL Injection Vulnerability
- Conclusion
Why Insecure? (SQL Injection)

- One of the classic XKCD comics (Exploits of a Mom)
  - Oct 10, 2007

- Demo:
  - http://cs.pugetsound.edu/~dchiu/cs455/webstuff/injection_demo.html
What Happened?

- Attacker guesses (correctly) that the form will take users to a page that does an INSERT.

```php
// from insertPassenger_insecure.php
$stmt = "INSERT INTO passengers VALUES
    ('$_POST[form_fname]', '$_POST[form_mname]', '$_POST[form_lname]', '$_POST[form_ssn]');"
$db->exec($stmt);
```

- After POST variables are evaluated. Now:

```php
$stmt = "INSERT INTO passengers VALUES
    ('David', 'Blah', 'Chiu', '123-45-6789'); delete from Passengers; -- ');
$db->exec($stmt);
```

- Essentially running two statements:

```sql
INSERT INTO passengers VALUES ('David', 'Blah', 'Chiu', '123-45-6789');
delete from Passengers; --
```
How to Combat SQL Injection?

- David isn't going to tell you
  - The preferred way:
    - Use PDO's *prepared statements*
  - Another way (not recommended in real-world)
    - "Sanitizing" inputs. Check every POST variable for suspicious stuff like:
      --, ', ), DROP TABLE, DELETE FROM, ...
    - Why not recommended? Limits what users can/can't enter. Some DB fields might want to accept any input (like a review)

- Project 2 must handle inputs securely
  - David will try to access/destroy your database as part of grading
Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion
Conclusion

- Dynamic web programming boot camp
  - PHP is a huge language... highly recommend that you learn more on your own

- Many of today's websites follow the 3-tier architecture:
  - Presentation: HTML + CSS
  - Logic: PHP, C#, JSP, ASP, Rails, ...
  - Database: MySQL, SQLite3, ...

- Further topics for exploration for the Web-curious:
  - JavaScript, NodeJS, Ajax, MongoDB, XML (DTD, XPath, XQuery)