PHP: Databases and Classes

CS174

Chris Pollett

Sep 29, 2008.

Outline

- Databases
- Classes

Connecting to MySQL from PHP

- To start a connect to a MySQL database one can issue the command:

  $db = mysql_connect();
- This function actually takes three parameters: the host, the username, and the password.
- These default to localhost, the process name PHP runs under, and blank.

  $db = mysql_connect(host, uname,pword);
- Depending on how mysql is configured, the first example above might work and saves some typing.
- This function returns false if a connection is not made.
- To close a database, one can call mysql_close();

Selecting a Database and queries

- To select a database one calls:

  mysql_select_db(“cars”);
- One can then do a query with a command like:

```
$query =“SELECT * FROM Corvettes”;
$result = mysql_query($query);
$num_rows = mysql_num_rows($result);
$num_fields = mysql_num_fields($result);
for($j =1; $j <=$num_rows; $j++)
{
    $row = mysql_fetch_array($result);
    print $row[0].$row[“some_attr”]. “<br />”;
}
```

- mysql_query can also be used to do inserts, etc.

Classes in PHP

- Classes in PHP are in many ways similar to classes in Java.
- To define a class one uses the keyword class as in:

```
class MyFirstClass {
    var $myVariable = 0;
    function getMyVariable() {
        return $this->myVariable;
        //note need to use $this
    }
}
```

- To create an instance of a class and invoke methods I can then use:

```
var $myClass = new MyFirstClass();
echo “My 1st var: {$myClass->getMyVariable()}”;
```

Including Classes

- Typically, you put the code for your class into a file and then use a line like:

```
require("MyClass.inc");
//or more likely
require_once("MyClass.inc");
```

- The require function is similar to include except when it fails it gives a fatal error rather than a warning. Also you cannot use require to include remote files even if allow_url_fopen is enabled.
- require_once will not re-include the file if it has already been included.

Constructors/Destructors

- A couple slides back we saw we could set up an initial value of a field variable of a class when we declare it: var $myVariable = 0;
- You can also have a functions __construct and __destruct to do initialization and clean-up.

```
function __construct($n=0) {
    $this->myVariable = $n;
}
```

Private, Protected, Public

- Member variables and member functions can be declared private, protected or public:

```
private var $myField;
protected function myMethod() { /* some code*/}
```

- Methods without any declaration are the same as public.
- Private means only visible within the class.
- Protected is visible within the class or within subclasses.
- Public means variable or method is visible to anyone.

Static and Const

- The static keyword creates one instance of the field or method for the object.

  class Foo{ static $bob=1;}

  echo “bob: {Foo::$bob}”;
- Within the class use self:: to refer to static members.
- The const keyword can be used to define constants for a class:

  class Goo{ const blob=1;}

  Notice no dollar sign. Can refer to this using Goo::blob
- Values of constants cannot be changed.

Cloning

- PHP has a command clone for cloning objects:

  $my_copy = clone $my_obj;

- To specify how the copying is done you can write a __clone() method for your class.

Inheritance

- Inheritance in PHP is very similar to Java.
- A PHP class can extend one other class.

```
class A{}
class B extends A {}
```

- PHP also has a notion of interface:

```
interface myInterface
{
   function method1($a, $b);
}
```

- A PHP class can implement multiple interfaces:

```
class C implements myInterfaceA, myInterfaceB {}
```

- If you want to have a class with some but not all of its methods defined. You can use the keyword *abstract* on those methods which will be overriden in subclasses.

Referring to Parents, Final

- Consider:
  1. class A {function foo(){} }
  2. class B extends A {function foo(){} }
- Within B this::foo() refers to the redefinition of foo given in (2).
- Within B parent::foo() refers to class A's version of foo.
- To prevent a function from being overridden in a subclass you can use the keyword final.

```
class A {final function foo() {} /* can't override*/ }
```

Exceptions

- PHP supports try catch blocks like Java.

```
try {} catch(MyException $e){} catch(Exception
   $ee){}
```

- You can use the keyword throw to throw an exception

```
if($denom == 0){
    throw new Exception("divide by zero");
}
```

- You can subclass Exception to create custom exceptions.