L1: PHP Object Oriented Programming

PHP Object Oriented Programming

We may not need to use objects to add dynamics to our PHP web applications until we are faced with data requirements that force the issue. Before we get there, it would be good to go over a few fundamentals about using classes in PHP.

Entity Classes: When we look to create database tables we identify **entities** **(http://en.wikipedia.org/wiki/Entity-relationship_model)** (customers, orders, products, etc.) which figure prominently in our application. These entities become candidates for classes in a complex web application. If we decide on an entity named "**Customer**" to store our customer data such as phone numbers, credit cards, etc., we would then create a database table named **Customers** (note the plural) then build a class named **Customer** (singular) to store all the entity data (database fields) as **properties** of the objects we'll create. Usually there is a direct map between the properties of the objects and the fields in the database table. Placing the data in the object in this way allows us to work with the object instead of directly accessing SQL every time we need to work with data. The relationships superimpose, each representing the same data given a different context:

- Entity -> attribute (abstract model)
- Table -> field (as stored in a database)
- Object ->property (as used in program)

PHP OOP Handout: The above is continued in the **PHP OOP Handout**
(http://www.southlore.net/upload/userfiles/wnewman/c0xuwmvCK3z1z478.doc)