**ITC260 Gig Central PHPDocumentor Documentation Example**

**File Level DocBlock:** At the top of every file,

Below is an example file sample DocBlock:

**/\*\***

**\* Gig.php manages all gig data**

**\***

**\* @package GIG\_CENTRAL**

**\* @subpackage GIG**

**\* @author Your Name <you@example.com>**

**\* @version 1.0 2016/06/09**

**\* @link http://newmanix.com/**

**\* @license http://www.apache.org/licenses/LICENSE-2.0**

**\* @see Gig\_model.php**

**\* @see views/gigs/index.php**

**\* @todo none**

**\*/**

**Class DocBlock:**

**/\*\***

**\* Gig Controller manages gig data**

**\***

**\* @see Gig\_model**

**\* @todo none**

**\*/**

**Function/Method DocBlock:**

**/\*\*  
 \* Constructor for Gig controller.   
 \*  
 \* @param none  
 \* @return void   
 \* @todo none  
 \*/**

BELOW IS THE ORIGINAL DATA FOR PHPDOCUMENTOR----------------------------------------------------------

When we work on our group projects, we’ll be using the PHPDocumentor documentation standards for our code:

*“phpDocumentor 2 is a tool that makes it possible to generate documentation directly from your PHP source code. With this you can provide your consumers with more information regarding the functionality embedded within your source and not just what is usable to them from your user interface.”*

*“Documentation generated by phpDocumentor 2 does not aim to be a replacement for conventional documentation but is rather supplemental, or reference, documentation.”*

**Resources:** Here are some links related to the use of PHPDocumentor and an example of how to use PHPDocumentor with CI:

[Your first set of documentation](http://phpdoc.org/docs/latest/getting-started/your-first-set-of-documentation.html) - How to get started with PHPDocumentor

**DocBlocks:** A DocBlock is a specifically formatted comment inside our code that is parsed by a third party app, such as PHPDocumentor, which turns it into formatted HTML pages.  DocBlocks are the building blocks for a well documented application.

*“Using a DocBlock you are able to effectively document your application’s API (Application Programming Interface) by describing the function of, and relations between, elements in your source code, such as classes and methods.”*

*“In reality a* [*DocBlock*](http://phpdoc.org/docs/latest/glossary.html#term-docblock) *is in fact the name for a combination of a, so-called,* [*DocComment*](http://phpdoc.org/docs/latest/glossary.html#term-8) *and a block of the* [*PHPDoc*](http://phpdoc.org/docs/latest/glossary.html#term-phpdoc) *Domain Specific Language (DSL). A DocComment is the container that contains documentation that can be formatted according to the* [*PHPDoc Standard*](http://phpdoc.org/docs/latest/references/phpdoc/index.html)*.”*

**DocBlock Levels:** DocBlocks should be placed at specific levels in the code, for example:

* **File Level DocBlock -**  Identifies the purpose of the file, ownership, version and contingencies
* **Class Level DocBlock -**  Identifies the purpose of the class, ownership and version
* **Function/Method Level DocBlock -**  Identifies the purpose of the function/method as well as ownership/version.  Also includes definitions of parameters and should always include code samples

**Examples:** Below are some examples of code that is documented per the PHPDocumentor standards.  Consider printing and using these while working.

**File Level DocBlock:** At the top of every file,

Below is an example file sample DocBlock:

**/\*\***

**\* file.php does blah, blah, blah**

**\***

**\* In addition it does blah.**

**\***

**\* @package LARGE\_PIECE\_OF\_PROGRAM**

**\* @subpackage CONTROLLER**

**\* @author Your Name <you@example.com>**

**\* @version 1.0 2015/04/30**

**\* @link http://www.example.com/**

**\* @license http://www.apache.org/licenses/LICENSE-2.0**

**\* @see related\_file.php**

**\* @see other\_related\_file.php**

**\* @todo none**

**\*/**

**Class DocBlock:**

**/\*\***

**\* Survey Class retrieves data info for an individual Survey**

**\***

**\* The constructor an instance of the Survey class creates multiple instances of the**

**\* Question class and the Answer class to store questions & answers data from the DB.**

**\***

**\* Properties of the Survey class like Title, Description and TotalQuestions provide**

**\* summary information upon demand.**

**\***

**\* A survey object (an instance of the Survey class) can be created in this manner:**

**\***

**\*<code>**

**\*$mySurvey = new Survey(1);**

**\*</code>**

**\***

**\* In which one is the number of a valid Survey in the database.**

**\***

**\* The showQuestions() method of the Survey object created will access an array of question**

**\* objects and internally access a method of the Question class named showAnswers() which will**

**\* access an array of Answer objects to produce the visible data.**

**\***

**\* @see Question**

**\* @see Answer**

**\* @todo none**

**\*/**

**Function/Method DocBlock:**

**/\*\*  
 \* Constructor for Answer class.   
 \*  
 \* @param integer $AnswerID ID number of answer   
 \* @param string $Text The text of the answer  
 \* @param string $Description Additional description info  
 \* @return void   
 \* @todo none  
 \*/   
   function \_\_construct($AnswerID,$answer,$description)  
 {#constructor sets stage by adding data to an instance of the object  
 $this->AnswerID = (int)$AnswerID;  
 $this->Text = $answer;  
 $this->Description = $description;  
 }#end Answer() constructor**

Example DocBlock from SurveySez project:

**<?php**

**/\*\***

**\* Survey\_inc.php data access classes & other related code for SurveySez project (position of line ids how it’s used) Name of the file, short description of what it does.**

**\***

**\* Data access for several of the SurveySez pages are handled via Survey classes**

**\* named Survey,Question & Answer, respectively.  These classes model the one to many**

**\* relationships between their namesake database tables.**

**\***

**\* @package SurveySez**

**\* @author William Newman**

**\* @version 2.0 2010/08/16**

**\* @link http://www.billnsara.com/advdb/**

**\* @license http://opensource.org/licenses/osl-3.0.php Open Software License ("OSL") v. 3.0**

**\* @see survey\_view.php (involved files)**

**\* @see response\_view.php**

**\*/**

**/\*\***

**\* Survey Class retrieves data info for an individual Survey**

**\***

**\* The constructor an instance of the Survey class creates multiple instances of the**

**\* Question class and the Answer class to store questions & answers data from the DB.**

**\***

**\* Properties of the Survey class like Title, Description and TotalQuestions provide**

**\* summary information upon demand.**

**\***

**\* A survey object (an instance of the Survey class) can be created in this manner:**

**\***

**\*<code>**

**\*$mySurvey = new Survey(1);**

**\*</code>**

**\***

**\* In which one is the number of a valid Survey in the database.**

**\***

**\* The showQuestions() method of the Survey object created will access an array of question**

**\* objects and internally access a method of the Question class named showAnswers() which will**

**\* access an array of Answer objects to produce the visible data.**

**\***

**\* @see Question**

**\* @see Answer**

**\* @todo none**

**\*/**

**class Survey**

**{**

**public $SurveyID = 0;**

**public $Title = "";**

**public $Description = "";**

**public $isValid = FALSE;**

**public $TotalQuestions = 0; #stores number of questions**

**public $TotalResponses = 0; # v5: stores number of responses**

**public $aQuestion = Array(); #stores an array of question objects - changed to public in v5**

**/\*\***

**\* Constructor for Survey class.**

**\***

**\* @param integer $id The unique ID number of the Survey**

**\* @return void**

**\* @todo none**

**\*/**

**function \_\_construct($id)**

**{#constructor sets stage by adding data to an instance of the object**

**$this->SurveyID = (int)$id;**

**if($this->SurveyID == 0){return FALSE;}**

**$iConn = IDB::conn(); #uses a singleton DB class to create a mysqli improved connection**

**MORE SURVEY CONSTRUCTOR CODE GOES HERE...**

**}**

**}# end Survey() constructor**

**/\*\***

**\* Reveals questions in internal Array of Question Objects**

**\***

**\* @param none**

**\* @return string prints data from Question Array**

**\* @todo none**

**\*/**

**function showQuestions()**

**{**

**if($this->TotalQuestions > 0)**

**{#be certain there are questions**

**foreach($this->aQuestion as $question)**

**{#print data for each**

**echo $question->Number . ') '; # We're using new Number property instead of id - v2**

**echo $question->Text . ' ';**

**if($question->Description != ''){echo '(' . $question->Description . ')';}**

**echo '<br />';**

**$question->showAnswers() . '<br />'; #display array of answer objects**

**}**

**}else{**

**echo 'There are currently no questions for this survey.';**

**}**

**}# end showQuestions() method**

**/\*\***

**\* Allows read only access to Question Array**

**\***

**\* Added in v3 - Result object**

**\***

**\* @param none**

**\* @return array an array of Question objects**

**\* @todo none**

**\*/**

**function getQuestions()**

**{**

**return $this->aQuestion;**

**}# end getQuestions() method**

**//MORE CODE WENT HERE...**

**}# end Survey class**

**Resources**

[Anatomy of a DocBlock](http://phpdoc.org/docs/latest/guides/docblocks.html) - A guide from the PHPDocumentor website on DocBlocks

[how to use phpdocumentor for codeigniter](http://stackoverflow.com/questions/12606839/how-to-use-phpdocumentor-for-codeigniter) - a couple of nice thoughts related to PHPDoc and CI on stackoverflow

*All text in italics is copied from the* [*PHPDocumentor website*](http://phpdoc.org/)