Whube Development Guide

Program without Suck

Authors:
Paul Tagliamonte
Thomas Martin

June 4, 2010

Chapter 1

MVC Basics of Whube

This chapter will cover the basics of the whube MVC basics. This will outline how Whube works under the hood. So, let's start off slow, and take a look at a simple incoming request.

1.1 Simple Request

So. Let's deal with the incoming request `http://whube.com/t/foo`. This request is sent to the server `whube.com` over the `http` protocol. CS 101, I know. Let's now look at the part handled by the Whube application. The request to the Whube webapp is `/t/foo`. This is handled by the .htaccess file in the project directory. The rule reads:

```
RewriteRule ^t/(.*+)      controller.php?p=$1   [NC]
```

This means that any requests that start with `/t/` will be passed to controller.php with everything after the last slash as the GET['p'] argumnt.

The `/t/foo` request is just an alias for `controller.php?p=foo`
the `controller.php?p=foo` script then goes on to do a few things. The `controller.php` script includes the `conf/site.php` and `libs/php/globals.php` files for use by the content script. `controller.php` then searches the `content/` directory for a content script that matches the request `foo`. `controller.php` resolves this to `content/foo.php`. If this script does not exist, it will default to `/t/default`.

If `content/foo.php` exists, it will be included by `controller.php`. This script defines two varables. `$CONTENT` and `$TITLE`. `$CONTENT` is the "meat"

of the page, and `$TITLE` is the HTML page title (`<title>`). After the controller includes the content script, in invokes `view/view.php`. This will take all the varables that the `content/foo.php` script sets up, and echos them to the screen in a pre-created HTML template.

1.2 Hello, World!

Let's take a look at the most simple content script. It's also pretty cool.

```
<?php
    $TITLE   = "Hello, World!"; // title of the page
    $CONTENT = "Hello, World!"; // content on the page
?>
```

Well, if you don't understand this, you might want to go back and review PHP. This is a very straight forward example, and it only goes downhill from here!

Chapter 2

Digging in deeper to Whube

In this chapter we will discuss more advanced features of the Whube platform.

2.1 $SITE_PREFIX

This is the full URL Path (INCLUDING `http://`) to the server and ending with a trailing slash (IMPORTANT!). ALWAYS use this when refering to another whube project file. If you fail to do this, it WILL fail. When you don't know the absolute path (how many arguments were passed into the script) you don't know how many times to call `../`!! Failing to use $SITE_PREFIX will cause your code to be rejected without review.

2.2 Kicking ass with $argv

Let's take a look at a new request. `http://whube.com/t/foo/bar/baz`. Breaking this back up, the request turns into `controller.php?p=foo/bar/baz`. If you missed this, go back and review the MVC Basics. `controller.php` will strip out the first item (`foo`) to detect the content script. `content.php` will take the rest of the URL past a forward slash and put it into an argument list. `content.php` sets up $argv and $argc just like this:

```
// handling request http://whube.com/t/foo/bar/baz

$argv = array(   // it's not actually done this way
    'bar',       // but it makes things really clear
```

```
        'baz'        // compared to the push back system
    );               // it uses.

    $argc = sizeof( $argv ); // this is actually how it's done.
```

A simple script using this would look something like:

```
<?php
    $TITLE   = 'Argument Listing';
    $CONTENT = "<h1>Arguments passed in:</h1>\n";

    foreach( $argv as $arg ) {
        $CONTENT .= $arg . "<br />\n";
    }

?>
```

If the foreach loop is new to you, you are missing out! It's really handy if you want to avoid cookiecutter code like a normal for loop.

2.3 Pimping out with useScript()

`useScript()` is a pretty kickass function of the whube codebase. It does lots of really fun things when you wish to manage what scripts should be included. When you call `useScript("jQuery.js");` it does some nifty stuff under the hood. It pushes the identifer back in an array `$SCRIPT`. When the `view/view.php` script is called, the `view/head.php` script should go through the `$SCRIPT` array and echo out the full path to the script with the correct base.

`useScript("jQuery.js");`
will produce output that looks a bit like

`<script src = 'http://whube.com/libs/js/jQuery.js' type = 'text/javascript'></`
The bit `http://whube.com/` will of course, be replaced with `$SITE_PREFIX`.
This also allows for the `useScript()` to rewrite URLs on the fly, and handle conditionals outside of the view (content) code.

2.4 Using the models

The core of what makes Whube easy to use is the `dbobj` superclass, and all the models that inherit from that. `dbobj` provides a unified object interface to the Whube backend. It provides a lot of methods that take pre-templated SQL querys and generlizes them. Let's take a look at a basic listing example for all bugs, and also querys projects and users.

```
<?php
    $Count = 20; // How many bugs we want on the page

    include( "model/bug.php" );       // Home of the bug klass
    include( "model/user.php" );      // Home of the user klass
    include( "model/project.php" );   // Home of the project klass

    $b = new bug(); // instantiate bug klass
    $b->getAll();   // same as a SELECT * FROM bugs;

    $u = new user();     // instantiate user klass
    $p = new project();  // instantiate project klass

    $TITLE = "Latest $Count bugs"; // descriptive title

    $i = 0; // Oldschool C99 style.

    $CONTENT .= "
<table>
    <tr>
        <th>ID</th> <th>Owner</th> <th>Project</th> <th>Title</th>
    </tr>
"; // Basic HTML code to start off the loop.

    while ( $row = $b->getNext() ) { // while we have bugs to go thru

        $u->getAllByPK( $row['owner'] );
        // ^ same as: SELECT * FROM users WHERE uID = '$OWNER';

        /*
         * You can see how easy it is to forget a col name here
```

```
         * or a table name there. Not if you use the models. Not
         * to mention you won't have to rewrite every Query if
         * you need to move a table or something.
         *
         */

        $owner = $u->getNext();
        // Get the first row ( should only be one )

        $p->getAllByPK( $row['package'] );
        // ^ same as: SELECT * FROM projects WHERE pID = '$PACKAGE';
        $package = $p->getNext(); // should only be one :)

        if ( $i < $Count ) { // check to see if we need another bug
            $CONTENT .= "\t<tr>\n<td>" . $row['bID'] // This is just
                . "</td><td>"                        // Display code.
                    . $owner['real_name']            // just outputs
                . "</td><td>"                        // a single line
                    . $package['project_name']       // of the table
                . "</td><td><a href = '"
                    . $SITE_PREFIX . "t/bug/" . $row['bID']
                . "' >" . $row['title']
                . "</a></td>\n\t</tr>\n";
        } else {
            break; // leave the while loop if we are > $Count
        }
        $i++;
    }

    $CONTENT .= "
</table><br /><br />
"; // end the basic HTML stuff

?>
```

You can see how it's easy to search for an object by it's internal ID.

Chapter 3

API Cheat Sheet

This chapter is full of how some of the internal stuff works. Refer to this often.

3.1 dbobj - the Database Object

```
class dbobj {
    function dbobj( $table, $pk_field );
    /*
     * dbobj ( constructor ) sets up
     *   the table and pk to query against
     */
    function getAll();
    /*
     * preforms a 'SELECT * ' against the
     *    table set up in the constructor
     */
    function createNew( $items );
    /*
     * preforms an 'INSERT'. The argument taken
     *     is an array. The array should look as follows:
     *
     * array(
     *     "package"  => $projectID,
     *     "reporter" => $userID,
     *     "title"    => $title,
     *     "descr"    => $descr
```

```
     * );
     *
     * That would create a new record on the top table
     * with the array data in each of it's fields.
     */
    function updateByPK( $PK, $tables );
    /*
     * preforms an `UPDATE` on the table the object
     *    uses. Array should look like an insert.
     *    $PK should be the PK of the row you want to update.
     */
    function getAllByPK( $pk );
    /*
     * this does a `SELECT *` WHERE the primary key
     *    = the $pk given. Very useful.
     */
    function getByCol( $cID, $id );
    /*
     * does a select on the column you want
     *    with the value you want. super useful
     *    for fetching similar things.
     */
    function numRows();
    /*
     * Number of rows returned on the last
     *    query set.
     */
    function getNext();
    /*
     * Get the next row of the last query
     *    sent out on this instance
     */
}
```

3.2 conf/site.php

```
$CONTENT            // Default site content
$TITLE              // Default site title
$SCRIPT             // Default array of JavaScript to include
```

```
$PRELOAD            // Default images to pre-load
$PAGE_MAX_COUNT     // Max items on a page.
$GUILT_ME           // Put the donate wrapper on the top
$TWEETER            // put the twitter banner on the top bit
$SITE_PREFIX        // the full path to the install ( from outside )
```

3.3 libs/php/core.php

```
$VERSION              // Whube Version String ( 3.141 as of Jun 4th )
$VERSION_STRING       // a cute "v" prepended to the $VERSION string.
                      //   might also include codenames.
$WHUBE_PROJECT_LEAD   // The Full Name of the Whube project Leader
$WHUBE_PROJECT_URL    // The full URL to the Canonical Whube site.
$ABOUT_WHUBE          // Brief about message from the Whube Project Leader
```

3.4 libs/php/globals.php

```
function useScript( $id );
/*
 * Pushes back the script to be included
 *    by the view code.
 */
function preload( $l, $w, $src );
/*
 * Preload an image to be included
 *    by the view code
 */
function breakUpLine( $line );
/*
 * Parse apart a line by slash
 */
function requireLogin();
/*
 * Bounce a user if they are not logged in
 *    this will bounce to /t/login
 */
function loggedIn();
/*
 * Check if the user is currently logged into
```

```
 *    whube.
 */
function getStatus( $status );
/*
 * Get the name of the status by the status ID
 */
function getAllStatus();
/*
 * Get all of the statuses that are in the Whube backend
 *    full select, all cols
 */
function getSeverity( $status );
/*
 * Get the name of the severity by the severity ID
 */
function getAllSeverity();
/*
 * Get all of the severities that are in the Whube backend
 *    full select, all cols
 */
```