# Package Management

What are **Composer** and **Curl**?

**Composer** is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you. We tell composer which packages our project uses, and Composer downloads and keep our files up to date.

**Curl** is a Linux command line tool for making HTTP connections. We will use curl to download and install composer.

Installing Composer

SSH to the server on port 2222 (substitute your own username for "user"):

**Command:**

ssh username@username.greenrivertech.net -p 2222

Download the installer through curl:

**Command:**

curl -sS https://getcomposer.org/installer | /opt/php53/bin/php

**Output:**

All settings correct for using Composer

Downloading 1.2.2...

Composer successfully installed to: /home/username/composer.phar

Use it: php composer.phar

Create a Configuration File

Create a configuration file in the project folder *for each project*. The config file specifies which packages we want to use in our project.

* Packages can be found on [www.packagist.org](http://www.packagist.org)
* The config file is stored as composer.json file in the topmost directory of your project
* composer.json is written in JSON (JavaScript Object notation)
* Open composer.json in your editor

**Example composer.json**

{

    "name": "tina/fatfreetutorial1",

    "authors": [

        {

            "name": "Tina Ostrander",

            "email": "[tostrander@greenriver.edu](mailto:tostrander@greenriver.edu)"

        }

    ],

    "require": {

        "bcosca/fatfree-core": "dev-master"

    }

}

**Example packages**

* Fat Free Routing/Templating: "bcosca/fatfree-core": "dev-master"
* Propel ORM: "propel/propel": "~2.0@dev"
* PHP Unit: "phpunit/phpunit": "5.5.\*"
* PHP Excel: “phpoffice/phpexcel”: “1.8.1”

Install Dependencies

Navigate to your project directory, which contains composer.json. From the command line, install the dependencies located in composer.json with the following command (substitute your own username):

/home/username/composer.phar install

When you view a directory listing (ls), you should see a **vendor** directory in your project folder. A few of the key files and folders are described here.

vendor

⎣ bcosca

⎣ fatfree

⎣ lib

⎣ base.php - Defines two classes:

**Prefab** is a singleton, which allows only one instance can exist at run-time

**Base** extends Prefab

* + - * defines constants and globals, including $hive, which stores the parameters (properties) of our application context.
      * **set** allows you to set variable values
      * **get** allows you to retrieve variable values
      * **run** gets called to make F3 work (needs an instance of the base)

⎣ composer

⎣ autoload.php - Autoloads whatever 3rd party libraries you have in your json file

Composer Lock

* Composer manages a lock file which keeps track of current version numbers, called composer.lock
* Because of this lock, it is necessary to use “composer update” instead of “composer install” after any changes to composer.json

/home/username/composer.phar update

Auto-loading

Composer can manage include paths to classes in your application… only one require is needed, and autoload.php takes care of the rest!

* Autoload takes whatever 3rd party libraries you have in your json file and includes them in your PHP code
* This is helpful due to inconsistencies with nested includes in PHP (based off the original script request location)
* Auto-loading removes the need for class includes, but folders containing classes of interest must be declared in composer.json

{

"require": {

...

},

"autoload": {

"classmap": ["folder1/", “folder2/”, ...]

}

}

After changing composer.json you must dump old information and regenerate the vendor/autoload.php file using the following command:

/home/username/composer.phar dump-autoload

What is Fat Free?

Fat-Free Framework (F3) is a powerful easy-to-use PHP routing and templating framework designed to help you build dynamic web applications. F3 supports, but does not require, MVC (Model-View-Controller) Development.

API: <https://fatfreeframework.com/getting-started>

Using Fat Free

The PHP script (usually index.php) will require autoload.php, create an instance of the base, and run Fat Free.

<?php

//composer autoloader; all includes defined in   
 //composer.json will automatically be imported.  
 require('vendor/autoload.php');

//Create an instance of the Base class  
 $f3 = Base::instance();

//Run fat free  
 $f3->run();

?>

Navigating to the index page in a browser will display an **Internal Server Error: No routes specified**. Fat free is running, but it's not doing anything yet.

Defining a Default Route

Define a default route for Fat Free *after* getting an instance of the base, but *before* invoking run(). F3's routing engine will take an address (/ is the index page), and execute an anonymous function.

<?php

…

//Create an instance of the Base class

$f3 = Base::instance();

**//Define the default route**

**$f3->route('GET /',**

**function() {**

**echo '<h1>Hello, world!</h1>';**

**}**

**);**

//Run fat free

$f3->run();

?>

.htaccess

**.htaccess**is a configuration file for use on web servers running the Apache Web Server software. Adding an .htaccess file to your root folder will make any request for missing files or directories redirect to index.php, which contains our router code and analyzes the requested URL.

Add an .htaccess file to your project directory. (See Canvas 🡪 Files 🡪 Student Files)

<IfModule mod\_rewrite.c>

<IfModule mod\_negotiation.c>

Options -MultiViews

</IfModule>

RewriteEngine On

# Redirect Trailing Slashes If Not A Folder...

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule ^(.\*)/$ /$1 [L,R=301]

# Handle Front Controller...

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteRule ^ index.php [L]

# Handle Authorization Header

RewriteCond %{HTTP:Authorization} .

RewriteRule .\* - [E=HTTP\_AUTHORIZATION:%{HTTP:Authorization}]

</IfModule>