Schedule Utility

# Dependencies

|  |  |  |
| --- | --- | --- |
| *Runtime* | *Version* | *More info* |
| **PHP** | v5.6 and above | The php runtime. <http://php.net/> |
| **MySQL** | v5.7.\* and above | Database for storage.  <https://www.mysql.com/> |
| **Apache or NGINX** | \* | The HTTP server. This document will assume your using Apache. |
| Composer | V1.4 and above | The project dependency manager utility.  <https://getcomposer.org/> |
| Git | v2.\* and above | Version control  <https://git-scm.com/> |
| Bower | v1.8.\* and above | Web assets dependency manager.  <https://bower.io/> |

The bolded dependencies are required for production.

# Introduction

The scheduling utility is the brain child of Dr. Albert Schwarzkopf in an attempt to make the scheduling process at the University of Oklahoma more intuitive and easier to digest. The project was initially developed by Austin Shinpaugh. This document was drafted in hopes that the project will continue to evolve after his graduation.

The author of the document developed this on OSX and used Homebrew as his package manager. If the next author(s) use another Unix distro then keep in mind that the package manager cli commands will change.

For example: $brew install git

Could change to: $ sudo apt-get install git

Check your manual for further information based on your distro. In Unix to see if you already have a dependency installed (and if it’s in your PATHenvironment variable) type:

$ which <dependency>

This project was developed with Symfony, a common Model-View-Controller (MVC) PHP framework. It has extensive documentation available online. This document is intended to combine knowledge from several sources and provided as a general guide.

# Installation

## Basics

Open up the Terminal and download your project decencies:

$ brew install git

$ brew install mysql

$ brew install php56 php56-opcache php56-yaml php56-igbinary php56-mcrypt

$ brew install httpd24

$ brew install composer

Several of these will require a first-time setup. It’s recommended you refrain from changing as many defaults as possible.

## Project Setup

### Part 1: Project setup

Change Directory (CD) into where you’ll be working on the project.

$cd /Users/ashinpaugh/development/

Next download the project:

$git clone schedule git@github.com:ashinpaugh/schedule.git

$ cd schedule

The output of *pwd* is what we will refer to as the **Project’s Root Folder** (PRF):

$ pwd

Next verify that your basic dependencies are installed correctly, make changes as suggested.

$ php bin/symfony\_requirements

Install the project’s PHP dependencies:

$ composer install

Next setup the project’s SQL structure:

$ php bin/console doctrine:database:create

$ php bin/console doctrine:schema:create

Next install the project’s assets into the <Project’s Root Folder>/web directory:

$ php bin/console assetic:dump

### Part 2: Server Configuration and Host entry

Setup your Apache Virtual Host:

$ vim /usr/local/etc/apache2/2.4/extra/httpd-vhosts.conf

Insert the following:

<VirtualHost \*:80>

   ServerAdmin  yourEmail@example.com

   ServerName   scheduler.dev

   SetEnv       APPLICATION\_ENV "dev"

   DocumentRoot /Library/WebServer/schedule

   <Directory "/Library/WebServer/schedule">

      DirectoryIndex app\_dev.php

      Options        MultiViews FollowSymLinks

      AllowOverride  All

      Require        all granted

   </Directory>

</VirtualHost>

* Edit server admin to something appropriate.
* *ServerName* will be the URL you type into your browser to access the site.
* *DocumentRoot see below.*

The *DocumentRoot* can point either to a symlink you create in the *WebServer* folder that points to your <Project’s Root Folder>/web, or directly to <Project Root Folder>/web. Sometimes there are technical reasons for creating a symlink to that WebServer folder, but we won’t be exploring those in this document.

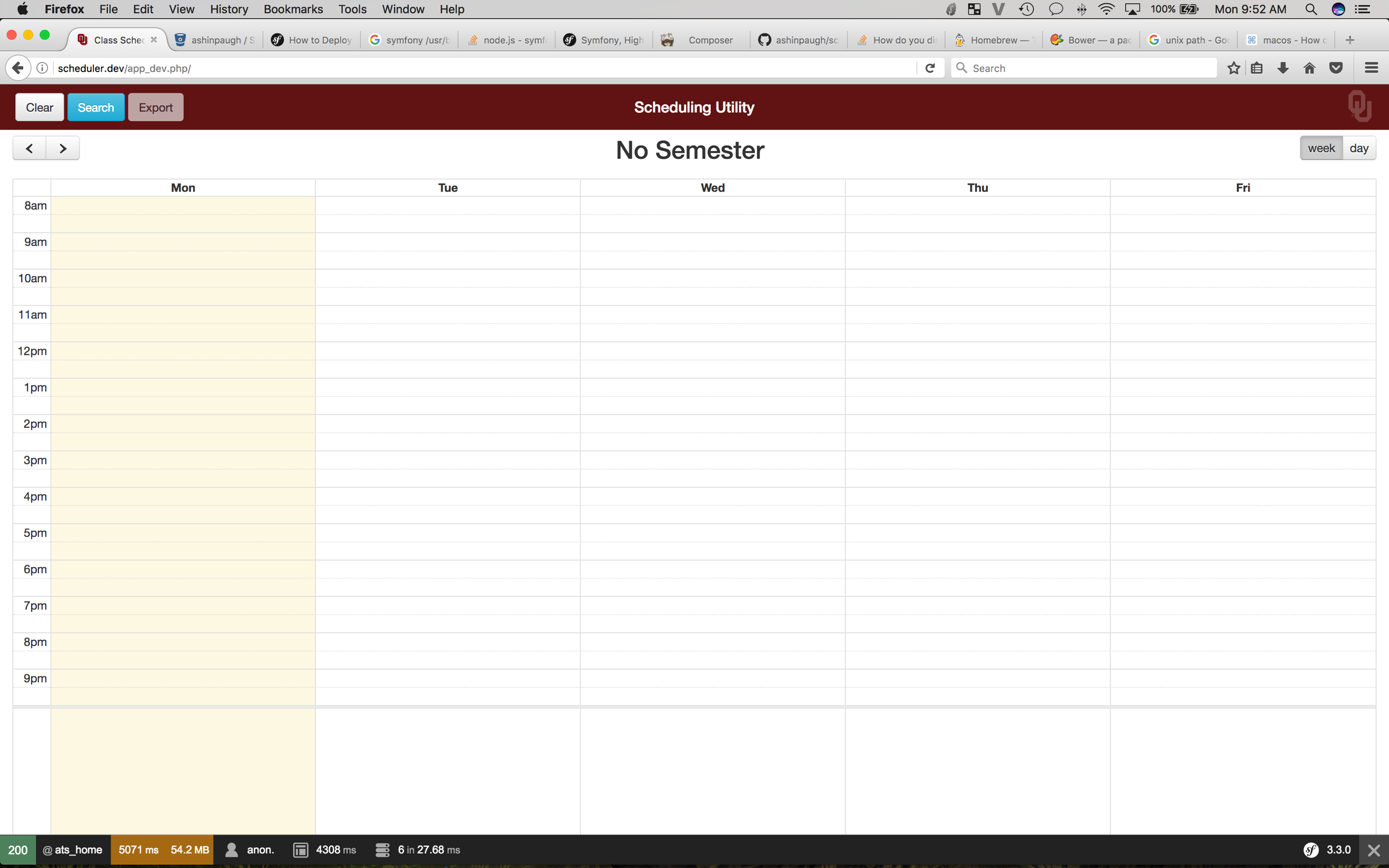
Edit your host file to point to home:

$ sudo vim /etc/hosts

And paste:

127.0.0.1 scheduler.dev

At this point you should be able to visit the landing page for the app:



Note the /app\_dev.php at the end. This means your accessing the “dev” version of the project, which will enable the Symfony’s Web Profiler feature to give you useful information regarding the performance of the app. /app\_dev.php also loads the “dev” config files and routing resources.

### Part 3: Development

At this point you need to import data to work with. You can get the data by running:

$ php bin/console scheduler:parse-book

This wasn’t high optimized (it takes a while to run) as it was hoped that we would be able to hook into API’s provided by OU IT; at the time of writing this document, those have yet to surface.

From here you’re ready to start development. At this point you should familiarize yourself with Symfony’s command line. To get a list of available commands type:

$ php bin/console

The author prefers PHPStorm by JetBrains for as an IDE. A one year free trial is available through their website for students. PHPStorm has an extremely useful Symfony plugin that is highly recommend.

When editing your CSS / Javascript run the :watch command in an open terminal window to compile those assets automatically when you reload your page:

$ php bin/console assetic:watch

To see a list of existing project APIs visit /api/doc, ie: <http://scheduler.dev/app_dev.php/api/doc>

# Maintenance

## Purging Data

Thanks to referential integrity, by deleting one *Semester* entry all related entries will be taken with it.