iCampus Development

James Antrim, Mariusz Homeniuk, Kristijan Pokas, Wolf Rost, Niklas Simonis

November 5, 2014

Contents

Chapter 1

Introduction

This documents explains how to set up and develop in the context of the iCampus Working Group. All instructions in this document are based upon the Windows 8.1 operating system.

Chapter 2

The Development Environment

In order to develop on a personal computer, must certain steps be performed in various systems, as well as various programs installed and configured. In this chapter these items will be discussed.

2.1 Local Web Server Installation and Configuration

In order for a locally developed project to be deployed, and to guarantee access to all required components a web server must be installed locally.

2.1.1 XAMPP

In the iCampus web development group we use XAMPP from Apache Friends for our local web servers. Apache Friends describes XAMPP as follows:

> *XAMPP is a completely free, easy to install Apache distribution containing MySQL, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.*[1]

As per the description the distributions of XAMPP available from Apache Friends come included with various useful tools which are also used within iCampus, such as PHP (server-side scripting language)abd MySQL (database management system), as well as many other tools which are optional for iCampus such as FileZilla (FTP Server) and Mercury (Mail Server).

Download and install the latest XAMPP installation. Keeping in mind that at least MySQL and phpMyAdmin need to be installed to function in the iCampus environment.

[1] https://www.apachefriends.org/index.html September 19, 2014

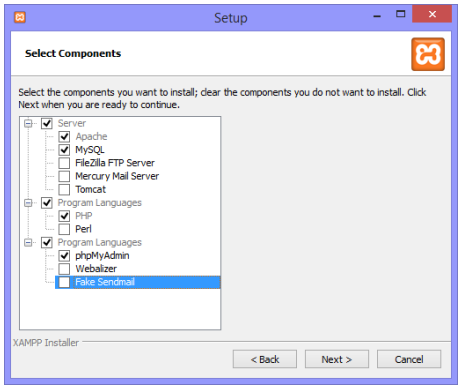

Figure 2.1: XAMPP Minimum Installation

2.1.2 PHP Configuration

The next step is the configuration of PHP for error reporting using xdebug. To do this open your php.ini file using the XAMPP control panel or the file explorer. This file is typically located at `C:/xampp/php/php.ini`.

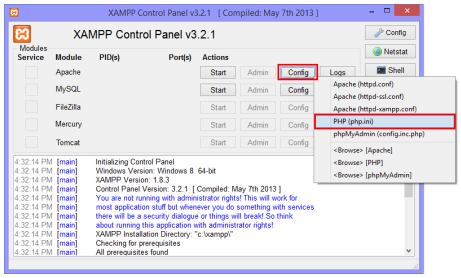

Figure 2.2: XAMPP Control Panel

The following modifications should then be made to the file:

```
// Reports all PHP Errors, Warnings, and Breaches of PHP Standards
error_reporting = E_ALL | E_STRICT

// Turns off error buffering => errors reported immediately, necessary for the debugger
output_buffering = Off
```

```
// Debugger Settings (may only need to be commented in)
[XDebug]
zend_extension="C:\xampp\php\ext\php_xdebug.dll"
xdebug.remote_enable=true
xdebug.remote_host=localhost
xdebug.remote_port=9000
xdebug.remote_handler=dbgp
xdebug.profiler_enable=1
xdebug.profiler_output_dir="C:\xampp\tmp"
```

The interface shown in Figure 2.2 above will also later used to start and stop the server service itself, Apache, as well as the the supporting database management service, MySQL, by pressing the corresponding "Start"/"Stop" button.

2.1.3 PEAR Configuration

In order to eliminate possible error sources, PEAR must next be configured. To this end run cmd.exe as an administrator and execute the following commands.

```
// Navigate to the PHP Directory
cd C:\xampp\php

// Display the PEAR Configuration
pear config-show
```

Directory Configuration

The following settings which may refer to C:\php\pear need to be changed to point to C:\xampp\php\pear should they not already do so.

```
pear config-set doc_dir C:\xampp\php\pear\docs
pear config-set cfg_dir C:\xampp\php\pear\cfg
pear config-set data_dir C:\xampp\php\pear\data
pear config-set cache_dir C:\xampp\php\pear\cache
pear config-set download_dir C:\xampp\php\pear\download
pear config-set temp_dir C:\xampp\php\pear\temp
pear config-set test_dir C:\xampp\php\pear\tests
pear config-set www_dir C:\xampp\php\pear\www
pear config-set php_ini C:\xampp\php\php.ini
```

Channel Configuration

Next PEAR itself and its channel list needs to be updated, so that later updates for extensions can be automatically pulled from their respective channels. Thereafter the auto_discover variable is set to on, this allows new channels to be discovered and dependencies to be resolved automatically.

```
// Ensures the latest PEAR version
pear upgrade pear

// Empties the PEAR Cache
pear clear-cache

// Turns on Automatic Discovery
pear config-set auto_discover 1

// Discover and Add the pear.phpunit.de Channel (usually already present)
pear channel-discover pear.phpunit.de

// Update the Channel list
pear update-channels

// Uninstalls CodeSniffer in Case a Deprecated Version is Present
pear uninstall PHP_CodeSniffer

// Installs PHP CodeSniffer
pear install PHP_CodeSniffer

// Discovers the PHP Mess Detector channel
pear channel-discover pear.phpmd.org

// Lists available packages
pear remote-list -c phpmd

// Installs PHP Mess Detector Package 1.5.0, actual as of 27 Sept, 2014
pear install phpmd/PHP_PMD-1.5.0
```

More about Code Sniffer and Mess Detector in Section 2.4.

2.2 Git

For repository and versioning iCampus uses Git. To this end Git must first be downloaded from Git, installed and configured.

2.2.1 Installation

Command Context

Part of the configuration is deciding which context you would like to be able to use Git in. In iCampus the second is used because it allows the developer later to use PHPStorm's own terminal to execute Git's commands, which eliminates the necessity for navigation to projects in Git Bash or Windows shell. More on PHPStorm in Section 2.3.

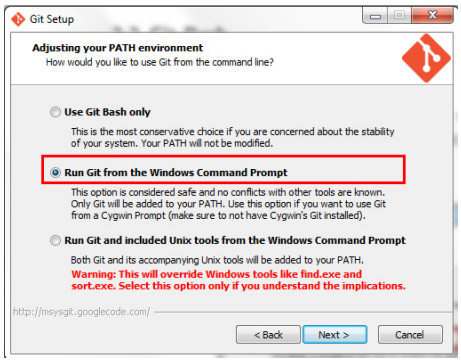

Figure 2.3: Command Context

Line Endings

The next screen lets one decide how file endings are pulled and committed. The iCampus Coding standards demand the Unix-style line endings. To this end only the second option is sufficient, because once style checking is enabled in your IDE the first and third options will, dependent on your system settings, report every single line of code as a breach of style, which removes any transparency whatsoever when searching for other errors.

2.2.2 User Configuration

In order to take part in the development process you must take several steps to configure your user account.

Global User Configuration

In Git Bash the global user name and email must be set so that repository actions can be associated with a user.

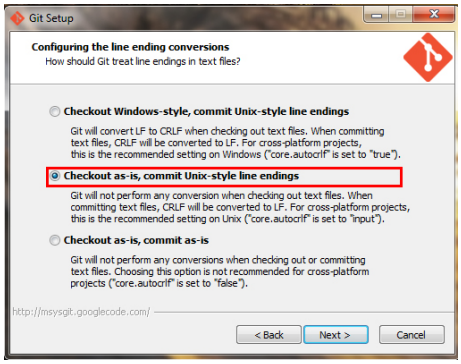

Figure 2.4: Line Endings

Start Git Bash and enter the following commands, replacing <First Name>, <Last Name>, and <Department> with your first and last names and the abbreviation of the department in which you are matriculated.

```
git config --global user.name "<First Name> <Last Name>"
git config --global user.email "<First Name>.<Last Name>@<Department>.thm.de"
```

You can confirm that the entries were saved correctly by entering the following command and comparing the associated values.

```
git config -l
```

SSH-Key Creation

After the user configuration is completed, an SSH-Key can be created using the following command in Git Bash.

```
ssh-keygen
```

The first question determines where the generated key will be saved and what its name will be. Per default this will be `C:/Users/<Windows Account Name>/.ssh/id_rsa`. This directory is important for the next steps, as well as later when repositories are cloned. Press enter to confirm the default. The next two questions will be to enter and confirm your password. You can also elect to have no password by pressing enter in answer to both questions.

SSH-Key in Gitorious eintragen

The freshly created key must now be added to the repository server in order to associate you as a user to projects for which you have access. First visit `http://scm.thm.de` and log in. After this your username will appear in the upper right hand corner between the search field and a button with a white arrow on it. Click on the arrow and select `Settings`.

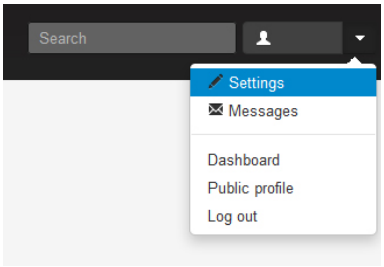

Figure 2.5: Settings Selection

Here we need to select the fourth tab `SSH Keys` and add the key using the `Add new` button.

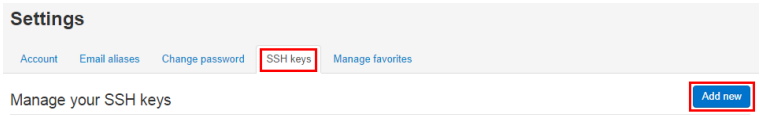

Figure 2.6: Settings Navigation

In the form that opens we have to copy the text of the public key. This can be found in the directory created during the creation of the key, `C:/Users/<Windows Account Name>/.ssh/id_rsa.pub`. Open this file with a text editor, copy the text from the file, and paste it into the available text field. At the end of the pasted text will be a `<Name>@<Domain>`. This should be replaced with the email address from the Git Configuration. When this has been completed click the `Save` button.

2.3 PhpStorm

PhpStorm is the IDE used in iCampus. While others are allowed the use of a single IDE in the working group allows for the exchange of information and tips on how to get the most out of a single IDE. This makes everyone more productive and allows for a collaborative effort to solve any problems which may be IDE related.

PhpStorm was chosen for its ease of use and integration of many tools which simplify development, including CodeSniffer, database connection tools, SASS support, Git support, and many others. PhpStorm can be downloaded from the JetBrains Website.

2.3.1 Installation

During the installation process you will be asked which file extensions should be associated with PhpStorm. As we use PhpStorm for the development with all given extensions, all can be selected. As of PhpStorm 8 this also allows for the editing of single files without the creation of a project.

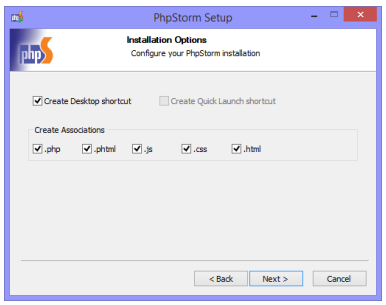

Figure 2.7: File Associations

For the most part the rest of the installation requires nothing special, however at the end you will be asked to chose between several themes and coloring/font style options. As this interface offers no preview it is best to go with the default settings. These selections can later be changed while running the PhpStorm which will actually show you what effects your selection has on its appearance.

After the installation starting PhpStorm will start the activation dialog. Here you will need to enter a "User or Company Name" and a "License Key". The name for our license is `Technische Hochschule Mittelhessen - University of Applied Sciences`. The license key will be given to you directly.

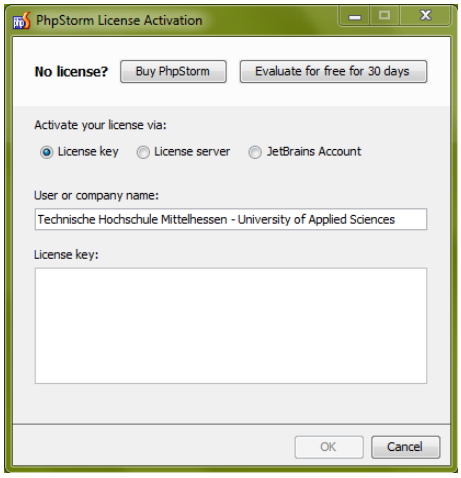

Figure 2.8: PhpStorm License Activation

2.3.2 THM Core Library

Next both for example and the inherent utility of the library we will create the project `lib_thm_core` by cloning the respository and checking out the development branch. Start PhpStorm and select `Check out from Version Control` from the "Quick Start" area and `Git` from the drop-down menu that appears. `Check out from Version Control` can later be reached under the `VCS` main menu entry.

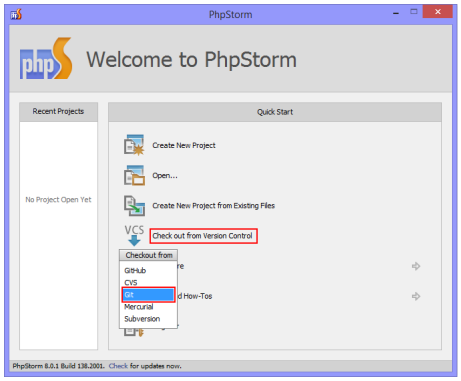

Figure 2.9: Check out a Git Project

We will want to use `git@scm.thm.de:icampus/lib_thm_core.git` as the "Git Repository Url". At this point PhpStorm will complain that the PhpstormProjects directory does not exist. Either create the default directory at the location shown or chose a different directory into which this (and your future projects if you so chose) will be saved.

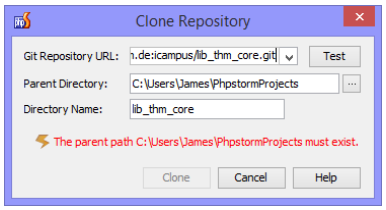

Figure 2.10: Clone Repository

While checking out you will be asked if you wish to save the server's SSH key to your list of known hosts, click ok to continue. You will then be asked if you would like to open the project you just created, click yes.

Since we did not include the branch name while cloning the repository, you will have checked out the master branch. In iCampus, with few very specific exceptions, **no one should be developing in the master branch**, for this reason we will now check out the development branch. This can be accessed in at least three ways. The way I find easiest is the branch display in the lower right hand corner. Click on the branch display, then the branch to be checked out, `origin/development`, and then `Checkout as a new local branch`.

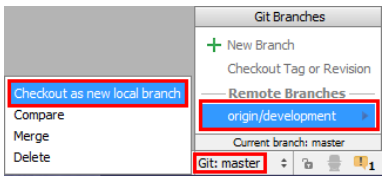

Figure 2.11: Branch Checkout

2.4 Code Sniffer and Mess Detector

Now that we have checked out `lib_thm_core`, we can use the coding standards within it to set up PHP CodeSniffer.

> PHP_CodeSniffer is a PHP5 script that tokenises and "sniffs" PHP, JavaScript and CSS files to detect violations of a defined coding standard. It is an essential development tool that ensures your code remains clean and consistent. It can also help prevent some common semantic errors made by developers.[2]

PHP Mess Detector on the other hand searches for problems more abstract and more serious such as[3]:

- Possible bugs
- Suboptimal code
- Overcomplicated expressions
- Unused parameters, methods, properties

2.4.1 Standards Directory Inclusion

First we will create symbolic links from the Joomla and THMJoomla folders to the folder which holds the existing standards. Open the windows shell as administrator and enter the following in the form `Command Link Target`.

Joomla Standard

Command `mklink /d`

Link `C:\xampp\php\pear\PHP\CodeSniffer\Standards\Joomla`

Target `C:\<System Specific Path>\lib_thm_core\coding_standards\Joomla`

iCampus Standard

Command `mklink /d`

Link `C:\xampp\php\pear\PHP\CodeSniffer\Standards\THMJoomla`

Target `C:\<System Specific Path>\lib_thm_core\coding_standards\THMJoomla`

[2] http://pear.php.net/manual/en/package.php.php-codesniffer.intro.php Stand September 22, 2014
[3] http://phpmd.org/ Stand September 27, 2014

2.4.2 PhpStorm

In order for Code Sniffer and Mess Detector to perform in PhpStorm they must first be integrated. This can be set as part of the default settings, meaning the settings will be applied to all projects, or in the project settings, where they would only valid for a specific project. The steps are exactly the same, but the entry point is slightly different. First access the settings using the menu item `File` and then selecting either `Default Settings...` or `Settings...` from the drop down menu.

PhpStorm Inclusion

To activate Code Sniffer click on `PHP` and then `Code Sniffer` or `Mess Detector`. In the text field next to "PHP Code Sniffer (phpcs) path" enter the path to phpcs.bat. In the standard installation this will be `C:\xampp\php\phpcs.bat`. After you have input the path, click on `Validate`. If the file is valid, the you will be shown a short text containing the version of the installed Code Sniffer. Although not specifically stated the steps for Mess Detector inclusion are completely analogous.

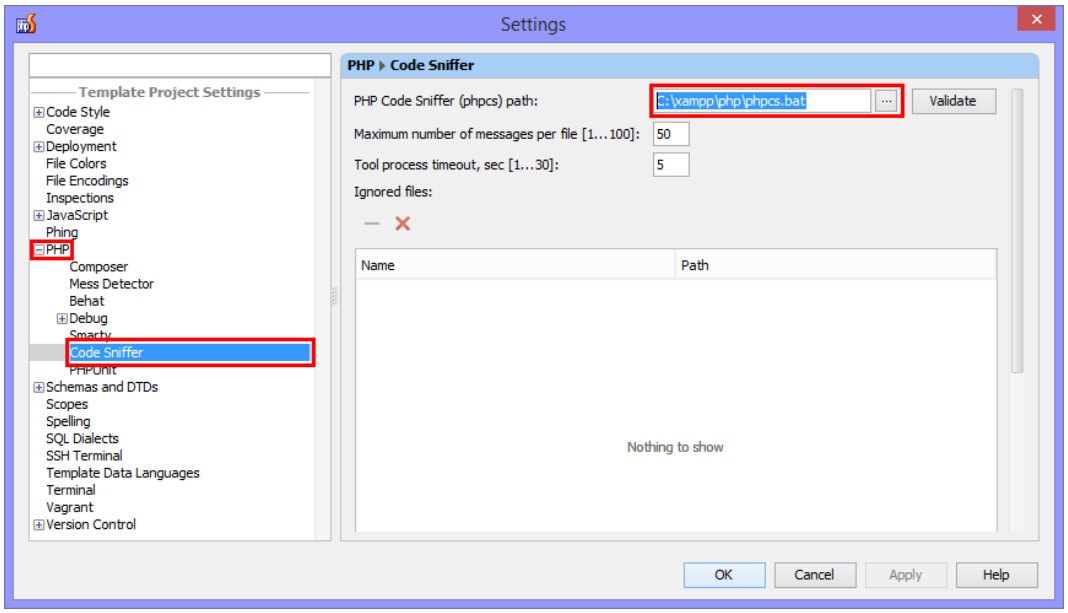

Figure 2.12: Code Sniffer - PhpStorm Inclusion

PhpStorm Configuration

Next we need to add PHP_CodeSniffer to the inspections performed. With the settings still open, click on `Inspections`, then in the list to the right `PHP`, and finally on `PHP Code Sniffer validation` to open the configuration settings for Code Sniffer.

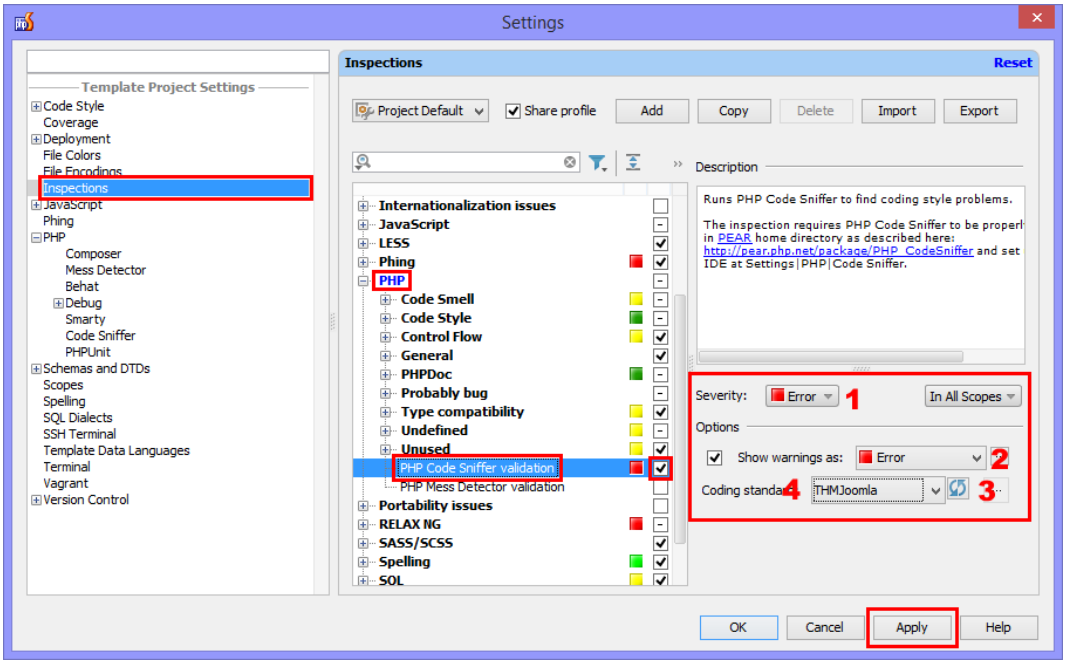

Figure 2.13: Code Sniffer - PhpStorm Configuration

To activate the inspection we must first set the checkbox next to `PHP Code Sniffer validation`. This activation enables the further settings to the right which we see numbered with one through four in Figure 2.13.

1. `Inspection severity`:"indicates how seriously the code issues detected by the inspection impact the project and determines how the detected issues should be highlighted in the editor."[4]

2. `Display severity`: defines how the infractions found are marked.

3. `Refresh`: updates the list of available coding standards. Should the desired standard not be found in the folder defined in the section on symbolic links, you can also manually search for the standard on your file system.

4. `Available standards`: a list of standards found. In iCampus we use the THM Joomla Standard which inherits or extends many of the definitions in the Joomla standard.

It is recommended that both inspection and display severity be set to errors to make the standard infractions more noticable. Settings take effect when the `Apply` button has been pressed.

Inspection settings for Mess Detector are also analogous here, with the distinction that one need not select a standard, instead selecting which rule sets should be applied.

[4]http://www.jetbrains.com/phpstorm/webhelp/configuring-inspection-severities.html Stand September 22, 2014

2.5 SASS

In iCampus we strive to maintain a standardized appearance both within views of the same extension as well as between extensions developed by the iCampus Group as a whole. To this end we utilize SASS.

> Sass is an extension of CSS that adds power and elegance to the basic language. It allows you to use variables, nested rules, mixins, inline imports, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized, and get small stylesheets up and running quickly, particularly with the help of the Compass style library.[5]

The above mentioned variables, nested rules, mixins, and inline imports allow the centralization of style structures and recurring themes, while still allowing for individualized style elements as required.

In iCampus we use the SCSS SASS syntax which closely resembles most .css you will have seen, and, stand alone, are actually valid CSS3 documents. This has as a consequence that our SASS files developed later will actually end with the extension .scss. For more information read the SASS syntax explanation.

2.5.1 Installation

Logically SASS must first be installed on the local system in order to function. This in turn requires the Ruby programming language, for which sass and compass are packages. While we need Ruby on the local system for SASS to function, we will not be using it to program, for this reason only the minimal requirements as pertain to SASS installation will be discussed.

Ruby can be installed by visiting the ruby installer downloads page and choosing the download that best suits your operating systems needs, as described in the right hand column.[6]

After you have downloaded and started the Ruby installation executable and accepted the license agreement you will be confronted with choices regarding the installation directory, support features, path variables, and file associations. We recommend using the default file path as it makes finding the installation easier, should any problems arise. You should only choose to install Td/Tk support if you know that you will be developing GUI applications in Ruby. The path variable is required in order for SASS to function. The association of .rb and .rbw files with this installation is recommended.

[5] http://sass-lang.com/documentation/file.SASS_REFERENCE.html Stand 27 September, 2014
[6] Stand 27 September, 2014

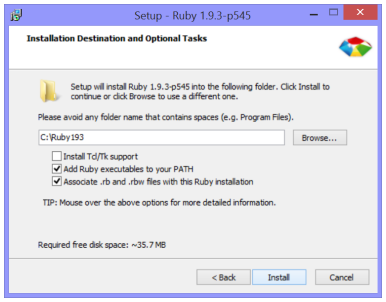

Figure 2.14: Ruby Installation

After ruby is installed we can begin to install SASS itself. Open the command line tool cmd, and enter the following command: `gem install sass`. Installation completes without an explicit noticed but can be verified by entering: `sass -v`. The result should resemble the output of Figure 2.15.

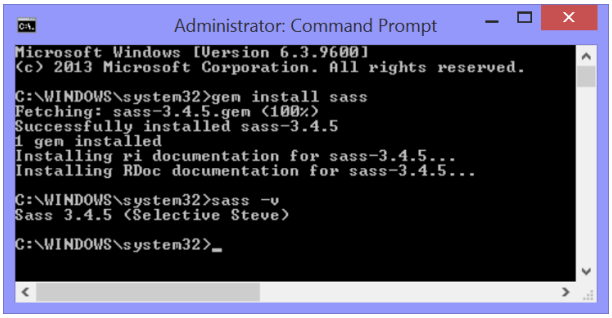

Figure 2.15: SASS Installation

Finally in order to extend the functionality of SASS, we install the Compass css authoring library package using the following command: `gem install compass`. Similar to SASS, no explicit message indicating the success of the installation is output, but can be verified with the command: `compass -v`.

2.5.2 PhpStorm Configuration

In iCampus our SASS infrastructure, or SCSS in our case, are stored at iCampus level in `lib_thm_core\scss`, at project level in `<Project Name>\\scss`, and lastly the compiled CSS files are stored in `<Project Name>\<Project Name>\media\css`.

The iCampus level SCSS files ensure standardized styles accross extensions. The project level styles allow for variance between individual extensions as required. To ensure that these files are not installed with the extension itself and thereby unnecessarily taking up system space these files are stored at root level in the project's repository.

Further complicating matters, SASS files are of two basic types, signified by the beginning of the file name. Templates are complete SASS files which are normally used to generate .css files, whereas partials contain reoccurring style information used by multiple style sheets. One of the many SASS conventions regards the naming of such files. Where templates have normal names, partials begin with a "_" to signify them as such.

The configuration of the PhpStorm file watchers allows for all changes made to templates in the scss folder of the project to be compiled to the its css folder. Opening a template in PhpStorm will trigger a prompt asking if you would like to create a file watcher (Figure 2.16).

Figure 2.16: File Watcher Notification

To configure a file watcher either click on `Add watcher` in the notification, or navigate to `File` > `Settings` then click on `File Watchers`, the `+` button on the right hand side of the interface, and finally on `SCSS` in the selection box that appears.

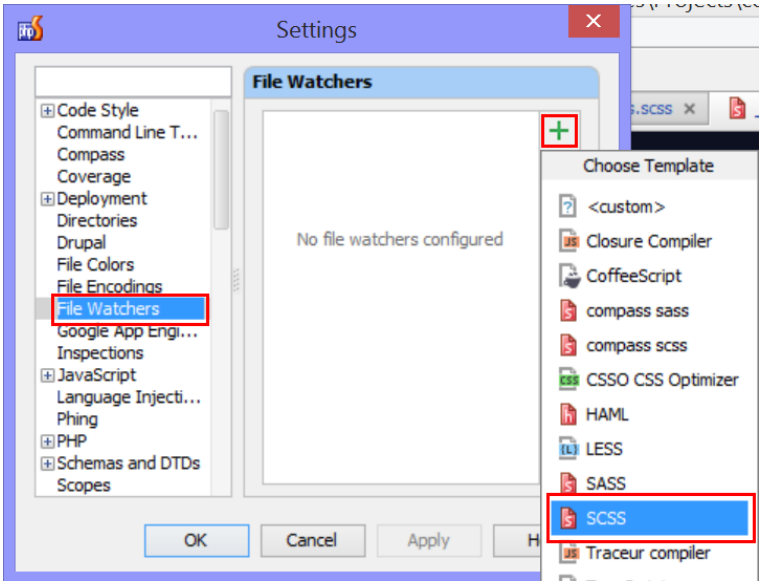

Figure 2.17: Add a New SCSS File Watcher

This brings us to the watcher edit interface seen in Figure 2.18. This will already have default values for all fields with the exception of `Program`. Fill out the fields as described below.

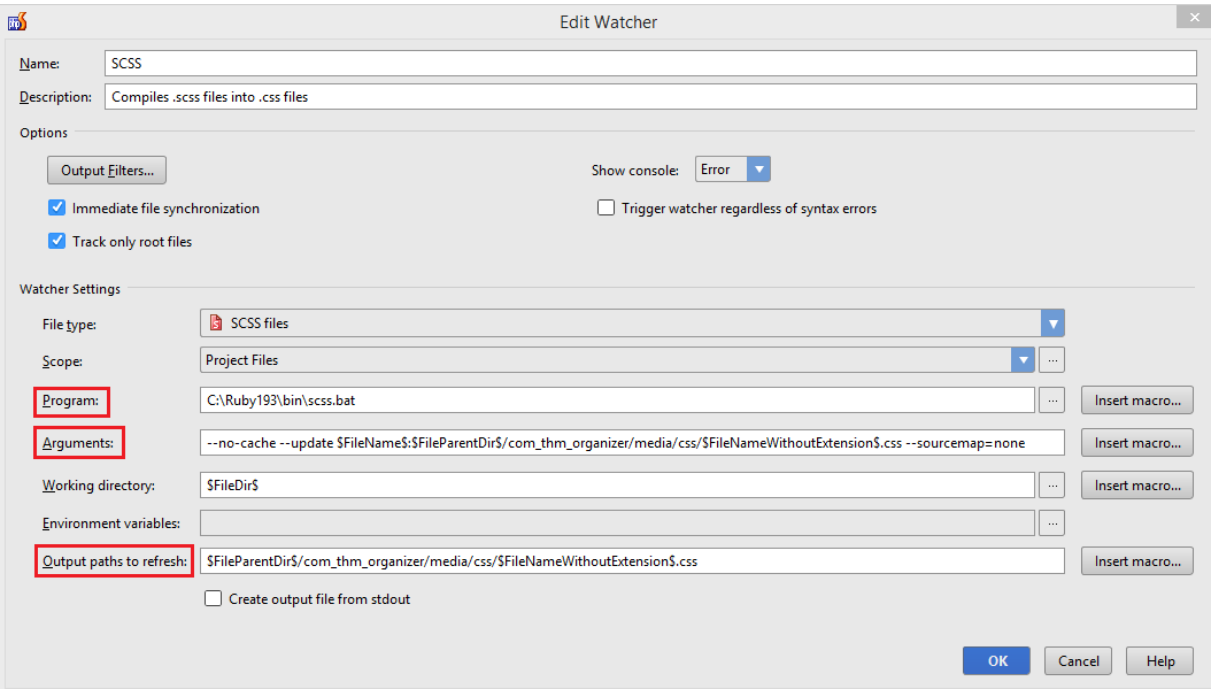

Figure 2.18: Edit File Watcher

Program: `<Ruby Installation Path>\bin\scss.bat`

Arguments: `--no-cache --update $FileName$:$FileParentDir$/<Project Name>/media/css/$FileNameWithoutExtension$.css --sourcemap=none`

Output Paths: `$FileParentDir$/<Project Name>/media/css/$FileNameWithoutExtension$.css`

Upon completion click "OK" to save the watcher. As necessary in the `Settings` interface put a check mark in the box in front of the watcher and click "Apply" for your changes to take effect.

SASS version 3.4+ automatically generates <File Name>.css.map files when used as a plugin, as well as adds comments to the .css files which reference these. While this may at some point be used for debugging purposes, I would really like to turn it off. If anyone figures out how I would be very grateful. I could not get the method explained here *to work.*

The inclusion of iCampus partials from `lib_thm_core` is done in the scss files themselves using `@import`. If your project repository is in the same directory as `lib_thm_core`, the imports would constructed as follows:

```
@import '../../lib_thm_core/scss/<Partial Name without Underscore>';
```

Here I would welcome any help in being able to add the directory with the partials to the SCSS path. This would reduce the import to the name of the partial to be imported.

2.6 Joomla!

Almost all software developed by iCampus is for use within the context of the Joomla! content management system (CMS) used by the Department of Mathematics, Natural and Informatics (MNI) at the Central Hessen University of Applied Science.

> Joomla is an award-winning content management system (CMS), which enables you to build Web sites and powerful online applications. Many aspects, including its ease-of-use and extensibility, have made Joomla the most popular Web site software available. Best of all, Joomla is an open source solution that is freely available to everyone.[7]

As of September 2014 iCampus is slightly between worlds as concerns Joomla!. The department's website currently still uses Joomla! version 2.5.x whereas the current Joomla! version as of 27 September, 2014 is 3.3.4.

For feature development and bugfixes for the department's website a dump of the site will be provided for you. Otherwise development should be performed in the context ofJoomla! 3.x which can be downloaded from the Joomla website. If you didn't deviate from the standard XAMPP installation you will want to extract the ZIP-Archive to the `C:\xampp\htdocs` directory.

At this point both the Apache and the MySQL services need to be running. To do so open the XAMPP control panel, Figure 2.2, and click the "Start" button next to both of these services.

2.6.1 Database Creation

For either of the development instances to function we first need to create databases for them to store their data in. To do this enter `http://localhost/phpmyadmin` in the address bar of your browser. This will open up the home page for your phpMyAdmin tool.

To create a database, click on the `Databases` tab,the interface for database management will then appear. The first item on the page will be "Create Database" under which will be an option for "Database name" and "Collation". Enter a name for your database and select the collation `utf8_general_ci` and then click `Create`. Then repeat as necessary for any further databases.

Figure 2.19: Database Creation

[7] `http://www.joomla.org/about-joomla.html` Stand 27 September, 2014

2.6.2 MNI Site Development

First we will discuss the installation of the MNI backup. If you will only be developing for Joomla! 3.x you should skip ahead to the next subsection. First extract the backup and give it an URL-friendly name such as "mni". Next place this folder in your web directory. If you followed the default installation steps this should be located at `C:\xampp\htdocs`.

Open a browser of your choosing and enter `localhost/<Your MNI Backup Folder>` in the address bar. If you renamed the backup "mni" this would then be `localhost/mni`. This will automatically start the Akeeba installer. The first page shown should look like Figure 2.20. As illustrated in the figure, we deviate from the recommended settings in one point "Display Errors", since we changed it in Subsection 2.1.2. Should your settings not match those displayed, reconfigure your PHP settings to do so.

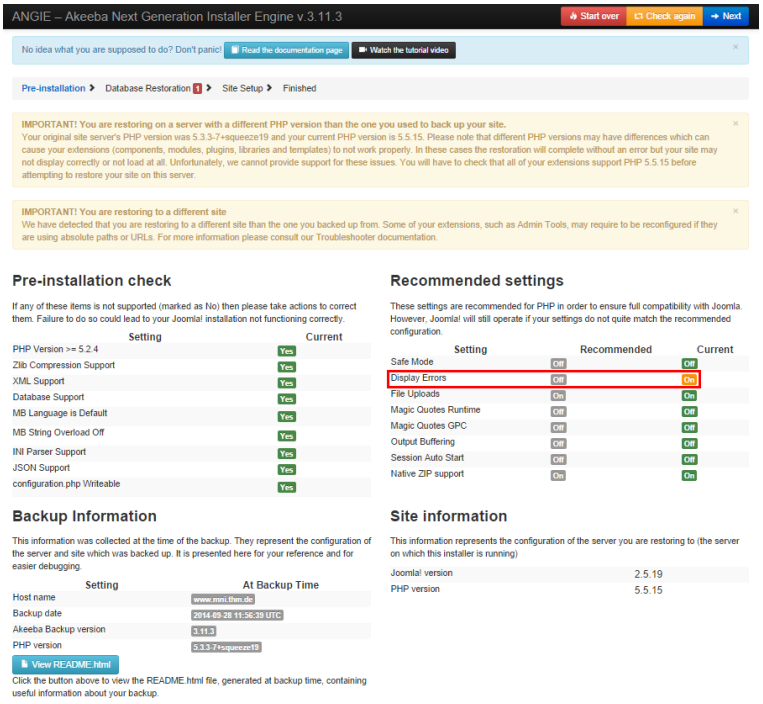

Figure 2.20: Akeeba Pre-installation

Click "⇒ Next". This brings us to the database restoration page seen in Figure 2.21.

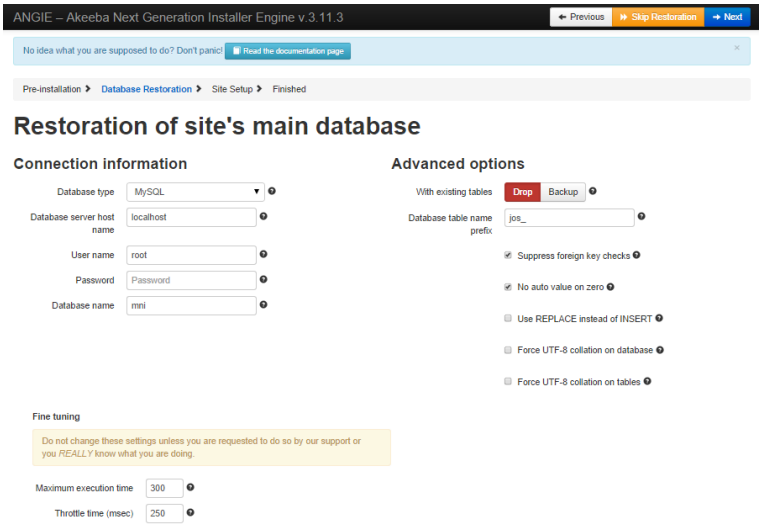

Figure 2.21: Akeeba Database Restoration

Here use the following values:

`Database type`	MySQL
`Database server host name`	localhost
`User name`	root
`Password`	(empty, unless set)
`Database name`	mni (unless named otherwise)
`Maximum execution time`	300

Clicking "⇒ Next" will trigger the restoration of the database. Then progress will then be displayed in a pop-up window. This may take up to several minutes dependent upon your computer. This brings us to the database restoration page seen in Figure 2.22.

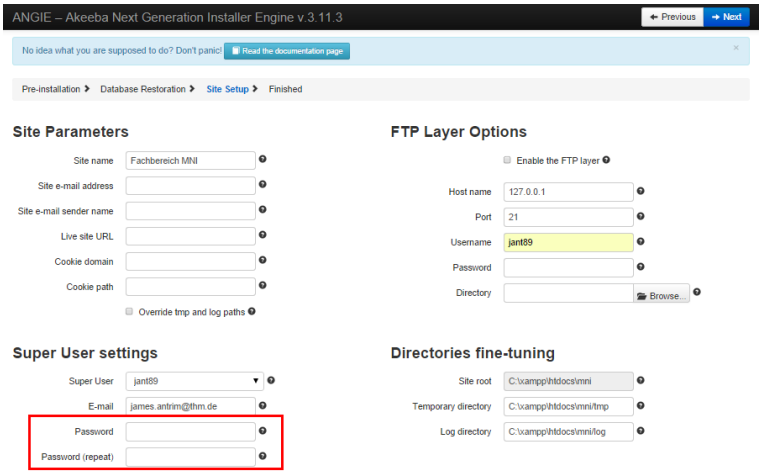

Figure 2.22: Akeeba Site Setup

Here you need to choose a `Super User`. The available options are the users which have super user access to the live web site. Here it is not important which user you chose, but rather that you remember which user you chose. Also important is setting a local `Password` for the chosen user, this prevents the system from validating against the data available to the CAS server. Set a new local `Password` and repeat it in `Password (repeat)`.

When you click Clicking "⇒ Next" you will be directed to the final confirmation page. Here you will be notified that you need to delete the `installation` directory to complete the installation. Do so by clicking on the blue button. You will then automatically be redirected to the site's frontend.

2.6.3 Joomla! 3.x Development

First abstract the archive file downloaded from the Joomla! website and give it a URL friendly name such as "j3". Next place this folder in your web directory. If you followed the default installation steps this should be located at `C:\xampp\htdocs`.

Open a browser of your choosing and enter `localhost/<Your Joomla 3.x Folder>` in the address bar. If you renamed the backup "j3" this would then be `localhost/j3`. This will open the main configuration page as seen in Figure 2.23.

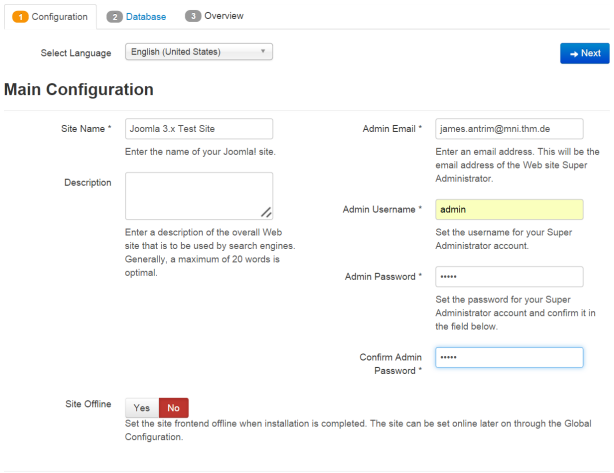

Figure 2.23: Joomla Main Configuration

Here you will need to enter the `Site Name`, `Admin Email`, `Admin Username`, `Admin Password`, and `Confirm Admin Password`. Click "⇒ Next" to move on to the database configuration.

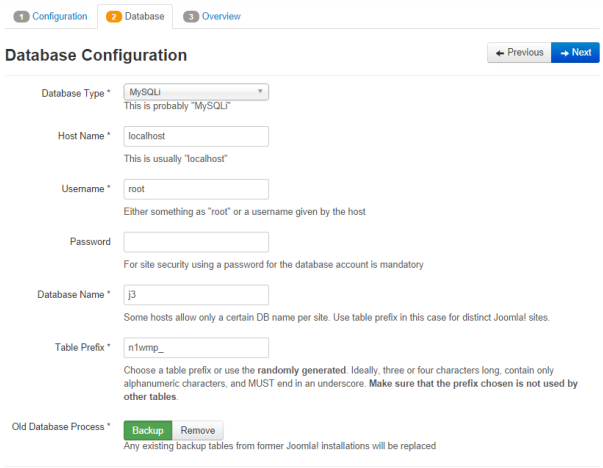

Figure 2.24: Joomla Database Configuration

Use the following values to complete the form, then click "⇒ Next" to continue on to the finalization.

`Database type`	MySQLi
`Host Name`	localhost
`Username`	root
`Password`	(empty, unless set)
`Database name`	j3 (unless named otherwise)

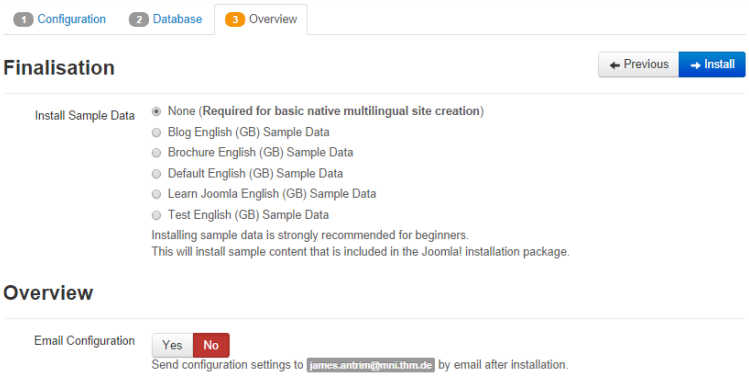

Figure 2.25: Joomla Finalization

Here you can choose whether or not you wish to install sample data. Leave "None" selected and click "⇒ Install" to complete the installation. The installation will then show the progress of the database construction. When this is finished you will be told that Joomla! is installed. You should now press the orange button "Remove installation folder" to complete the installation. You can choose to visit the frontend or backend of the site by clicking the appropriate button.

Chapter 3

Extension Development

3.1 Installation Files

Joomla has two files used for installation, upgrade and deinstallation of extensions: the manifest file and a so-called script file.

3.1.1 Manifest File

The manifest file is an XML file common to all extensions. It tells Joomla! everything the site itself needs to know about the extension, such as which files to install, and the name of the extension. Joomla's own documentation to this file can be found here. Although this documentation is relatively thorough there are some finer points which may be of interest to the developer.

Extension Tag

This is the root tag for the document and contains important information as attributes. The type tag should be self-explanatory and the individual extension types will be covered later in this chapter.

The method tag tells Joomla how to handle the 'installation' of the extension. The two methods available are The greatest difference here is the

3.1.2 Script File

The so-called script file is an optional file which can perform actions or display information during extension installation, upgrade, or uninstall process. This file can be named anything as the name is given as a parameter of the manifest file in the `scriptfile` tag. It must however follow the convention of containing a class with the name <`Extension Name`>`InstallerScript`.

The main functions of this file correspond with the process names they accompany: install, upgrade and uninstall. Additionally, the developer can define preflight and postflight functions which will be performed before or after each call to one of the three main functions.

3.2 Components

Components are the main Joomla Extensions responsible for the management of resources inclusive of both the functionality offered to the user and the data upon which this functionality rests.

3.3 Libraries

Libraries offer functionality to extensions.

3.4 Templates

Templates define the page presentation structure.

3.5 Modules

Modules are lightweight views based upon data offered by components and can be bound to many varying positions within a template.

3.6 Plugins

Chapter 4

User's Guide

4.1 Git

Revert to the previous commit

From the repository directory in Git Bash or the PhpStorm Terminal enter:

```
git reset --hard HEAD
```

Warning! This will erase any and all uncommitted changes!

Create a new local branch

Add a local branch to remote

From the repository directory in Git Bash or the PhpStorm Terminal enter:

```
git push -u origin <Branch Name>
```