A
Installing SilverStripe

SilverStripe is an incredibly powerful system, and yet so easy to use, once you know your way around it. However, getting started can be a little tricky—the installation in particular is sometimes a challenge for newcomers. This doesn't mean SilverStripe is hard to set up, but there are some common pitfalls.

To avoid those, we'll take a look at how to:

- Set up a development environment on Windows
- Set up a live environment on Linux
- Install SilverStripe itself

Setting up the environment

There are many possible scenarios and environments you might require, depending on the complexity of your project. However, we'll keep this short and simple:

- We'll set up a development environment, which is based on Windows (Windows XP or newer) as it is the most common operating system on desktops and laptops.
- We'll also set up one live or deployment environment, which is based on Linux as it is the most common operating system for web servers. We'll use Ubuntu in our example as it is freely available and widely used. If you prefer a different distribution some paths and commands will vary, but the general approach should be very similar.
- In case you want to use a testing or staging environment as well, we'll assume that you simply reuse the previous setup—so we don't need to configure another system.

Now that we've decided on the general purpose of the systems and their operating systems, we'll need to choose between different implementations for setting up the web server and database. Specifically, whether to use prebuilt packages or select the components ourselves.

To demonstrate both approaches we'll use a prebuilt package on our development machine (as we want to get up and running quickly) and handpick the components on our live site, where we want to have maximum control. To keep our development and live environments closely related, we'll use the same applications for both—disqualifying Windows-only solutions:

- For the web server we'll use the Apache HTTP Server (`http://httpd.apache.org`) as it's the most widely used, cross-platform server. Alternatives would be Microsoft's Internet Information Services (`http://www.iis.net`), which are only available on Windows, and Lighttpd (`http://www.lighttpd.net`) or Nginx (`http://nginx.org`), which are both fast, but not as widely used as the Apache web server, and are a little trickier to set up.
- For the database we'll use the popular MySQL (`http://www.mysql.com`). Alternatively you could use Microsoft SQL Server (`https://www.microsoft.com/sqlserver/2008/en/us/default.aspx`), which requires Windows; PostgreSQL (`http://www.postgresql.org`), SQLite (`http://www.sqlite.org`), or an Oracle database (`http://www.oracle.com/us/products/database/index.html`). However, each of these alternatives requires an additional SilverStripe module to support its specific SQL dialect, which generally receive less attention than MySQL. Unless you have a good reason for changing the database, like you're already using MS SQL for everything else, stick with MySQL.
- For a change the programming language PHP (`http://www.php.net`) doesn't require any decisions as there are no alternative implementations.
- For the prebuilt package we'll use XAMPP (`http://www.apachefriends.org/en/xampp.html`), which matches our requirements. It uses the Apache HTTP Server and MySQL and is available on Windows, Linux, Mac OS X, and Solaris. If your development machine doesn't use Windows, you should still be able to follow the installation steps with minor variations.

> On Windows there's also Microsoft's Web Platform Installer (`http://www.microsoft.com/web/downloads/platform.aspx`). It doesn't only include the web server, database and PHP, but also SilverStripe itself (`http://www.microsoft.com/web/gallery/silverstripecms.aspx`)—which can be downloaded on demand among many other web applications. However, as this is very different to our live site, we won't cover it. If you're looking for a, Windows-only solution this is nevertheless an interesting option as it is very easy to set up.

The Windows development system

Let's start off by setting up our development system, based on the freely available XAMPP package.

Time for action – installing XAMPP

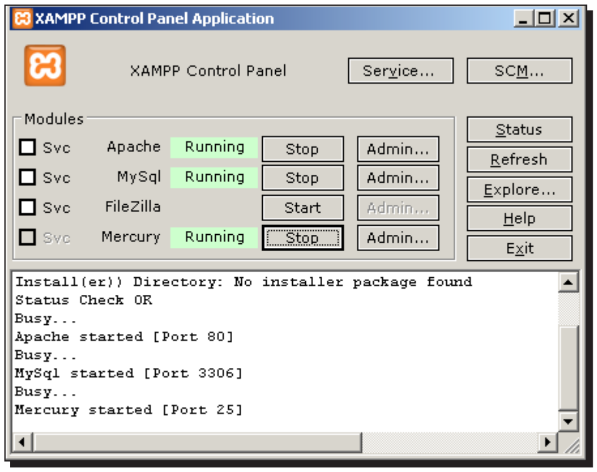
We'll assume that the operating system is Windows XP or newer—for Linux, Mac OS X, and Solaris you should only need to get a different download file and do some minor variations of the steps described:

1. Download the XAMPP package at `http://www.apachefriends.org/en/xampp-windows.html`. We're using the 7zip package as it doesn't require an installation, is compact to download and also portable: you can copy it to a USB drive and use the environment on any Windows computer.
2. Extract the archive to a top-level folder on your internal drive (`C:\` for example), or your portable external drive (`U:\` for example). To do this, you'll need the free 7zip archive utility, which you can download at `http://www.7-zip.org`.
3. Open up the file `xampp/mysql/bin/my.ini` and add the following line in the `[mysqld]` section:

   ```
   lower_case_table_names=2
   ```

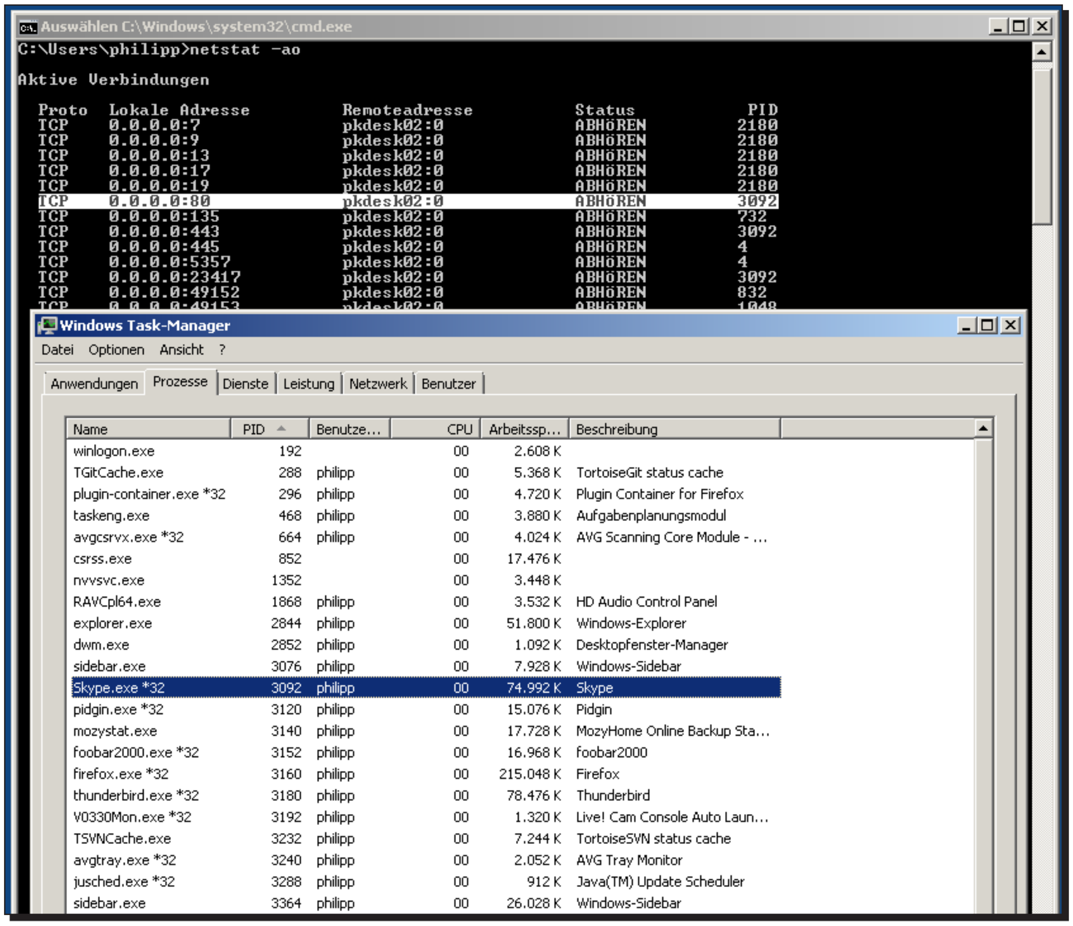
> This setting is important for exchanging database dumps between Windows and Linux systems. As Windows paths are case insensitive, MySQL lowercases database and table names by default (as they are internally stored as folders and files), but only on Windows. On Linux proper cases are preserved, which leads to problems when using dumps on different platforms. The previous statement forces MySQL on Windows to behave like on Linux.

4. Inside the XAMPP folder, start the file `xampp-control.exe` and use it to **Start Apache**, **MySql** and **Mercury**. XAMPP's control panel should look like this after starting the three applications:


XAMPP Control Panel Application
XAMPP Control Panel
Service...
SCM...
Modules
Svc Apache Running Stop Admin...
Svc MySql Running Stop Admin...
Svc FileZilla Start Admin...
Svc Mercury Running Stop Admin...
Status
Refresh
Explore...
Help
Exit

```
Install(er)) Directory: No installer package found
Status Check OK
Busy...
Apache started [Port 80]
Busy...
MySql started [Port 3306]
Busy...
Mercury started [Port 25]
```

5. If a service doesn't start, check that its port is not already in use—this is the main reason for problems. To do this, open the command line tool and enter `netstat -ao`. This will show you the ports currently in use and the process IDs (PIDs) of the processes using them. Using Window's Task-Manager you can then find out the name of the process causing the problem, and stop or reconfigure it. The following screenshot illustrates this scenario—the web server cannot be started as Skype is already using port 80:



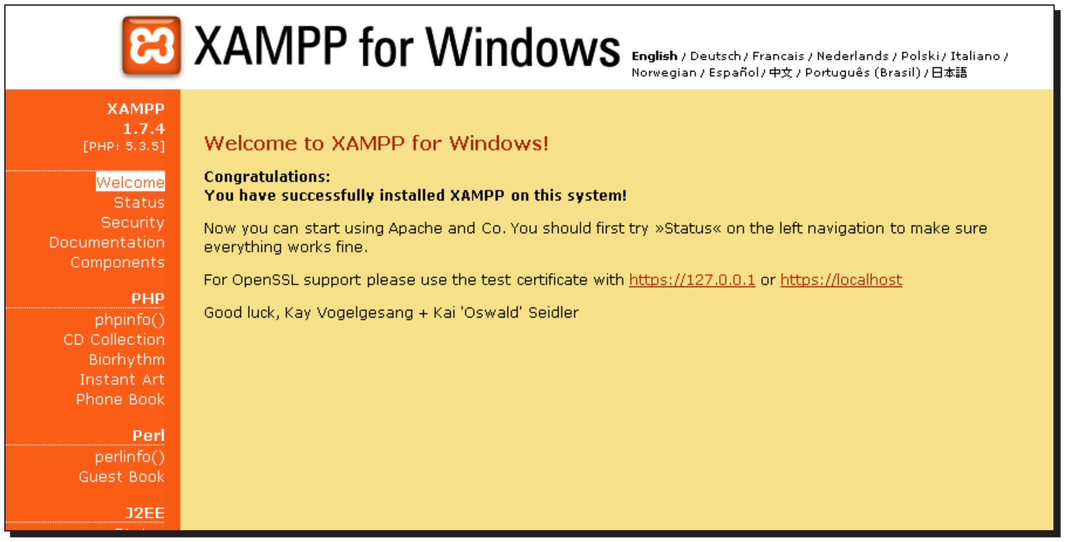
```
C:\Users\philipp>netstat -ao

Aktive Verbindungen

  Proto  Lokale Adresse          Remoteadresse          Status           PID
  TCP    0.0.0.0:7               pkdesk02:0             ABHÖREN          2180
  TCP    0.0.0.0:9               pkdesk02:0             ABHÖREN          2180
  TCP    0.0.0.0:13              pkdesk02:0             ABHÖREN          2180
  TCP    0.0.0.0:17              pkdesk02:0             ABHÖREN          2180
  TCP    0.0.0.0:19              pkdesk02:0             ABHÖREN          2180
  TCP    0.0.0.0:80              pkdesk02:0             ABHÖREN          3092
  TCP    0.0.0.0:135             pkdesk02:0             ABHÖREN          732
  TCP    0.0.0.0:443             pkdesk02:0             ABHÖREN          3092
  TCP    0.0.0.0:445             pkdesk02:0             ABHÖREN          4
  TCP    0.0.0.0:5357            pkdesk02:0             ABHÖREN          4
  TCP    0.0.0.0:23417           pkdesk02:0             ABHÖREN          3092
  TCP    0.0.0.0:49152           pkdesk02:0             ABHÖREN          832
  TCP    0.0.0.0:49153           pkdesk02:0             ABHÖREN          1048
```

Name	PID	Benutze...	CPU	Arbeitssp...	Beschreibung
winlogon.exe	192		00	2.608 K	
TGitCache.exe	288	philipp	00	5.368 K	TortoiseGit status cache
plugin-container.exe *32	296	philipp	00	4.720 K	Plugin Container for Firefox
taskeng.exe	468	philipp	00	3.880 K	Aufgabenplanungsmodul
avgcsrvx.exe *32	664	philipp	00	4.024 K	AVG Scanning Core Module - ...
csrss.exe	852		00	17.476 K	
nvvsvc.exe	1352		00	3.448 K	
RAVCpl64.exe	1868	philipp	00	3.532 K	HD Audio Control Panel
explorer.exe	2844	philipp	00	51.800 K	Windows-Explorer
dwm.exe	2852	philipp	00	1.092 K	Desktopfenster-Manager
sidebar.exe	3076	philipp	00	7.928 K	Windows-Sidebar
Skype.exe *32	3092	philipp	00	74.992 K	Skype
pidgin.exe *32	3120	philipp	00	15.076 K	Pidgin
mozystat.exe	3140	philipp	00	17.728 K	MozyHome Online Backup Sta...
foobar2000.exe *32	3152	philipp	00	16.968 K	foobar2000
firefox.exe *32	3160	philipp	00	215.048 K	Firefox
thunderbird.exe *32	3180	philipp	00	78.476 K	Thunderbird
V0330Mon.exe *32	3192	philipp	00	1.320 K	Live! Cam Console Auto Laun...
TSVNCache.exe	3232	philipp	00	7.244 K	TortoiseSVN status cache
avgtray.exe *32	3240	philipp	00	2.052 K	AVG Tray Monitor
jusched.exe *32	3288	philipp	00	912 K	Java(TM) Update Scheduler
sidebar.exe	3364	philipp	00	26.028 K	Windows-Sidebar

6. After successfully starting all three services, navigate to `http://localhost` in your browser. This should redirect you to XAMPP's welcome page.



Welcome to XAMPP for Windows!

Congratulations:
You have successfully installed XAMPP on this system!

Now you can start using Apache and Co. You should first try »Status« on the left navigation to make sure everything works fine.

For OpenSSL support please use the test certificate with https://127.0.0.1 or https://localhost

Good luck, Kay Vogelgesang + Kai 'Oswald' Seidler

7. In case you need to send e-mails for testing purposes, you'll need to perform two additional steps to configure your SMTP server:

   - Enable SMTP relaying of non-local mail. However, ensure that no one can access your mail server from outside or you might be abused as a spam relay (your router or firewall will protect you). In XAMPP's control panel, click on **Admin...** next to the **Mercury** label. Next go to **Configuration**, **MercuryS SMTP Server** and on the **Connection control** tab uncheck **Do not permit SMTP relaying of non-local mail**.

   - Provide a DNS server so domain names can be looked up: Under **Configuration**, **MercuryE SMTP Client** fill in the field **Name server** with your provider's DNS server.

8. That's it for our general purpose development machine. In case you want to run automatic tests or generate translation files, you'll also need to install PHPUnit through PEAR:

   - While XAMPP includes PEAR and PHPUnit, the bundled versions are hopelessly outdated. Furthermore the update process doesn't work reliably, so we'll better start anew.

   - First remove `xampp/php/PEAR/`, `xampp/php/pear.bat`, `xampp/php/pear.ini`, `xampp/php/peardev.bat` and `xampp/php/pear-update.bat`.

- Next enable PHP's cURL extension: Open `xampp/php/php.ini`, find the line `;extension=php_curl.dll` and remove the leading semicolon.
- Then download `http://pear.php.net/go-pear.phar` into `xampp/php/`.
- Next install PEAR on the command line (start it as an admin user). Note that you must be in the folder `xampp/php/` for the following commands to work:

  ```
  php go-pear.phar
  ```

- Now we can use PEAR to update its channels and upgrade all packages:

  ```
  pear update-channels
  pear upgrade --alldeps
  ```

- Finally you can install PHPUnit and all of its dependencies:

  ```
  pear channel-discover pear.phpunit.de
  pear channel-discover components.ez.no
  pear channel-discover pear.symfony-project.com
  pear install phpunit/PHPUnit
  ```

- That's it, you've successfully installed PHPUnit!

The Linux live system

Next we'll set up our live system. Thanks to package managers this isn't much harder—you just shouldn't be afraid of the shell.

Time for action – installing the requirements by hand

In our example we're using Ubuntu, so we'll rely on its package manager Apt, abbreviation of Advanced Package Tool. Note that we won't only install the bare necessities, but also some more tools to make our system ready for production.

1. Open the terminal and install the Apache HTTP Server together with PHP and PHP's GD library (needed for image manipulation) through the following commands. Note that all required dependencies are added automatically, though you may need to manually accept them:

   ```
   sudo apt-get install apache2 php5 php5-gd
   ```

2. Next we'll install MySQL and its binding to PHP, again through the terminal. You'll be prompted for a password, which we'll need again later on.

   ```
   sudo apt-get install mysql-server php5-mysql
   ```

3. Enable Apache's rewrite module, which SilverStripe requires to use pretty URLs:

   ```
   sudo a2enmod rewrite
   ```

4. Edit the file `/etc/apache/sites-available/default` and replace the line `AllowOverride None` with `AllowOverride All` inside the `<Directory /var/www/>` block. This enables the rewrite module inside the `/var/www/` folder.

5. Install **Alternative PHP Cache** (**APC**), which will accelerate PHP programs—SilverStripe benefits a lot from this. While there are some alternatives, it is currently planned that APC will be included in PHP 6, which makes it sound like a future-proof decision for our installation:

   ```
   sudo apt-get install php-apc
   ```

> If you want to see APC in action, you'll need its management tool. By default it is located in the file `/usr/share/doc/php-apc/apc.php.gz`. Unpack and copy it to your web-root to see detailed statistics on how well it's working for you, but protect it against unauthorized access (you can set credentials inside the file).

6. Edit the file `/etc/php5/apache2/php.ini`, using whatever text-editor you prefer (this can be quite a sensitive topic with programmers). Replace the line `;date.timezone =` with `date.timezone = US/New_York` or your current location. See `http://php.net/manual/en/timezones.php` for all possible values. Note that removing the semicolon at the beginning is important, otherwise the line will be considered a comment.

7. Restart the web server to apply all the settings we've just made:

   ```
   sudo /etc/init.d/apache2 reload
   ```

8. Now it's time to test our installation. Go to the directory `/var/www/` which is the default web-root directory:

   ```
   cd /var/www/
   ```

9. It should already contain a file `index.html`, which you can view if you visit `http://127.0.0.1` in your browser (or whichever IP address you can access your server under).

10. However, this is only a static HTML file. To see if PHP is also working, first set the file permissions, assuming our current user is `ubuntu` (which we'll use to edit files) and the web server is run as `www-data` (which is the default). So we'll make our user the owner of the files with full permissions, and set the group to the one of the web server with read and execute permissions only:

```
sudo chown ubuntu:www-data -R /var/www/
sudo chmod 0755 -R /var/www/
```

After adding new files, you'll need to rerun these two commands.

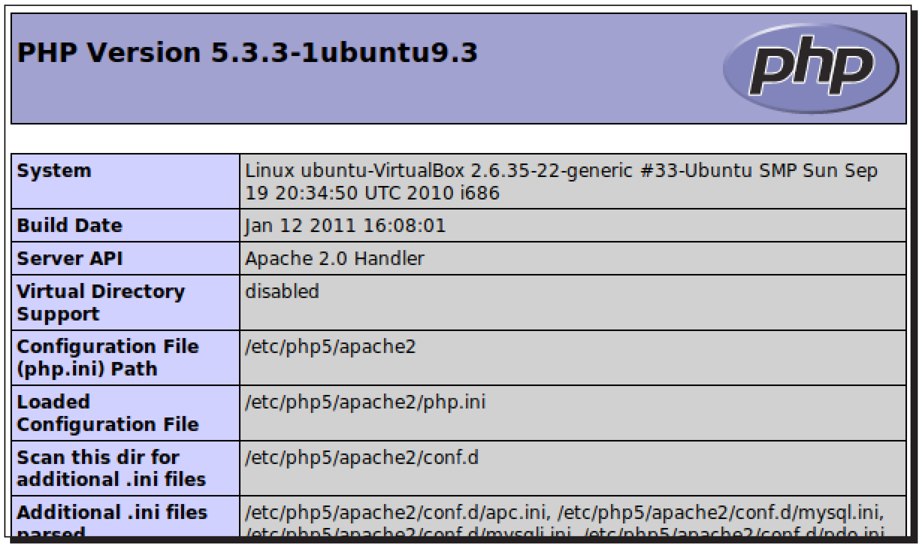
11. Next create a file `index.php` in `/var/www/`:

```
touch index.php
```

12. Add the following code to it:

```
<?php
  phpinfo();
?>
```

13. In your browser load the page `http://127.0.0.1/index.php` (again use your specific IP address). The output should look something like this:



PHP Version 5.3.3-1ubuntu9.3

System	Linux ubuntu-VirtualBox 2.6.35-22-generic #33-Ubuntu SMP Sun Sep 19 20:34:50 UTC 2010 i686
Build Date	Jan 12 2011 16:08:01
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/apc.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini

14. This shows that PHP is working. Also check that you can find **mod_rewrite** and a full section on **APC** in the output to be sure they are enabled.

15. That's it, our live system is now ready for action. We haven't installed PEAR as we'll assume that testing and translating is done in the development environment, not on the live server.

> We've also left out how to install and configure an SMTP server. Unfortunately this is pretty complicated and beyond the scope of this book—especially since you don't want to become a spam relay. If your provider or distribution hasn't already set this up for you, take a look at a specific SMTP server how-to and documentation—Postfix (`http://www.postfix.org`) is widely used for example.

Installing SilverStripe

The following steps are the same on both systems. You can follow these instructions on either one and then copy your site (that is the code, your database dump and any files you've added to the `assets/` folder) to the other—but we'll come to that a little later.

Installation

To keep it simple, we'll assume that you know how to get SilverStripe's core files to your development and live system. The options include:

- Downloading it from SilverStripe's website in your browser to your development machine: `http://silverstripe.org/stable-download/`
- Downloading it through the shell with `Wget` from the above address to your live environment
- Uploading the files from your development to your live site through FTP or SSH
- Getting the code from the official repository or your own one—further directions for this can be found at `http://silverstripe.org/version-control/`

Time for action – installing SilverStripe

We're assuming you've set up your environment. Additionally, you've added a folder `silverstripe/` to your web-root containing all of SilverStripe's core files. In our examples this would either be `xampp/htdocs/silverstripe/` or `/var/www/silverstripe/`.

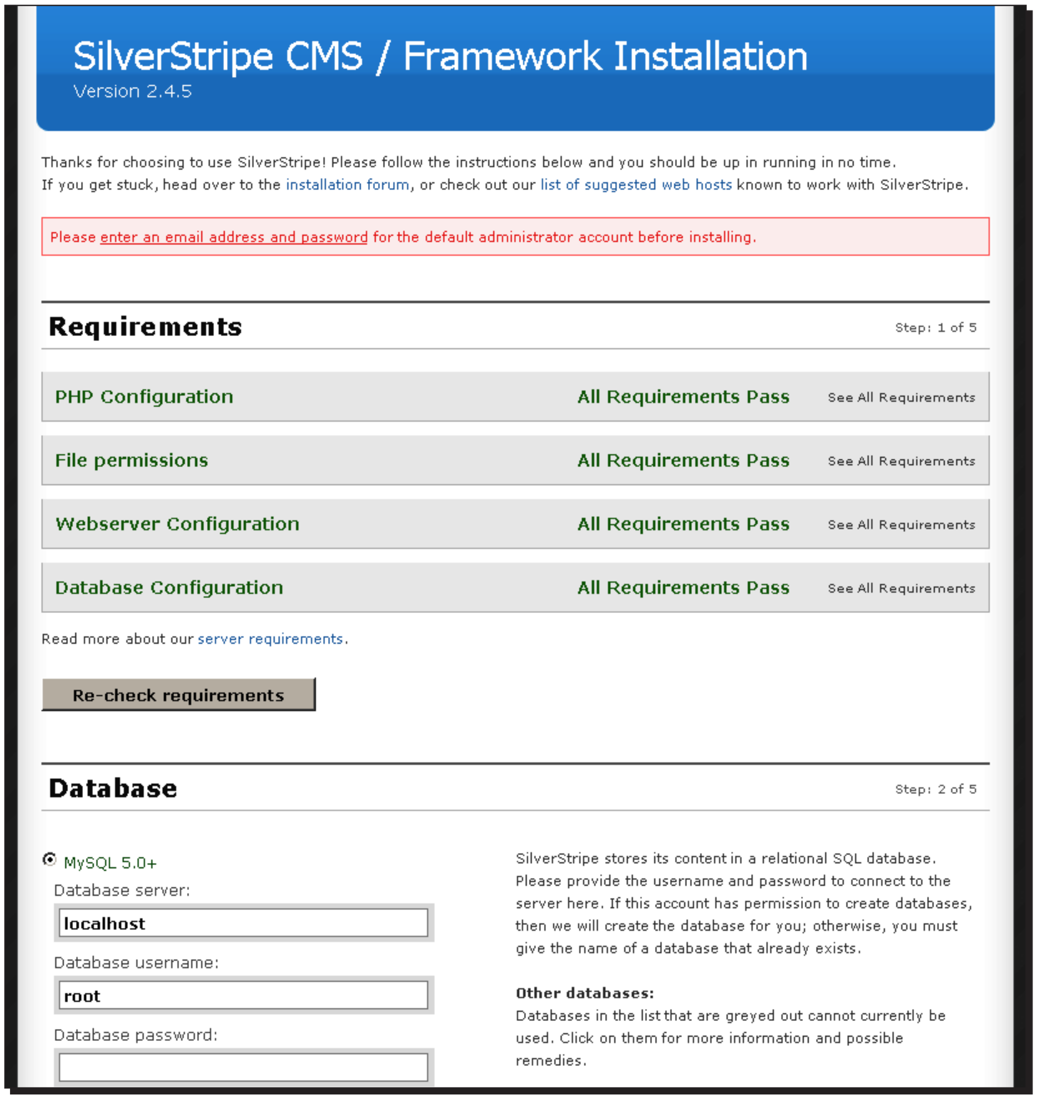
1. On Linux, make sure you've set the correct file permissions:

```
sudo chown ubuntu:www-data -R /var/www/
sudo chmod 0755 -R /var/www/
```

2. Additionally you'll need to allow the web server to write to the following files and folder. This isn't normally necessary on Windows due to its laxer permission settings:

```
chmod 0775 -R /var/www/silverstripe/assets/
chmod 0775 -R /var/www/silverstripe/.htaccess
chmod 0775 -R /var/www/silverstripe/mysite/_config.php
```

3. Open `http://127.0.0.1/silverstripe/` in your browser—if you're not installing on localhost, replace the IP address with the one of your server. You should then be redirected to the installation page which looks like this:



SilverStripe CMS / Framework Installation
Version 2.4.5

Thanks for choosing to use SilverStripe! Please follow the instructions below and you should be up in running in no time.
If you get stuck, head over to the installation forum, or check out our list of suggested web hosts known to work with SilverStripe.

Please enter an email address and password for the default administrator account before installing.

Requirements Step: 1 of 5

PHP Configuration	All Requirements Pass	See All Requirements
File permissions	All Requirements Pass	See All Requirements
Webserver Configuration	All Requirements Pass	See All Requirements
Database Configuration	All Requirements Pass	See All Requirements

Read more about our server requirements.

Database Step: 2 of 5

[✓ MySQL 5.0+]

Database server: localhost

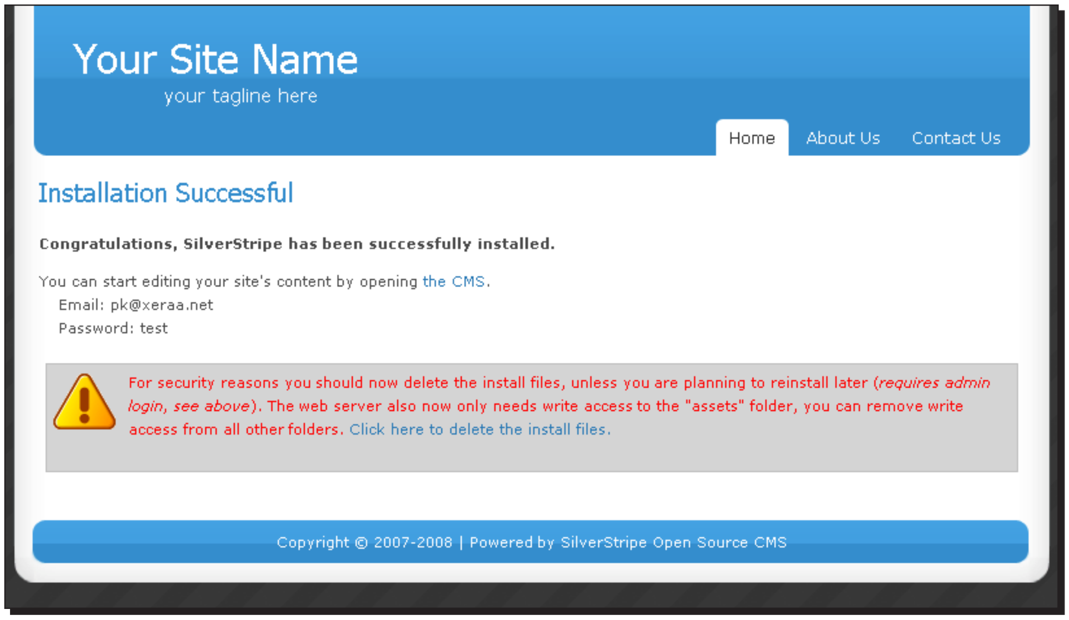
Database username: root

Database password:

SilverStripe stores its content in a relational SQL database. Please provide the username and password to connect to the server here. If this account has permission to create databases, then we will create the database for you; otherwise, you must give the name of a database that already exists.

Other databases:
Databases in the list that are greyed out cannot currently be used. Click on them for more information and possible remedies.

4. If there are any problems with your installation, SilverStripe will tell you about them and your four **Requirements** points will be yellow or even red instead of green.
5. Fill out the database credentials and the database name you want to use for your installation.
6. Fill out your e-mail address, password, and select your default language.
7. Hit **Install SilverStripe**.
8. This might take a little while as SilverStripe is doing a lot of work in the background. After finishing, you should be redirected to the following page:



Installation Successful

Congratulations, SilverStripe has been successfully installed.

You can start editing your site's content by opening the CMS.
Email: pk@xeraa.net
Password: test

For security reasons you should now delete the install files, unless you are planning to reinstall later (*requires admin login, see above*). The web server also now only needs write access to the "assets" folder, you can remove write access from all other folders. Click here to delete the install files.

Copyright © 2007-2008 | Powered by SilverStripe Open Source CMS

9. As advised, remove the installation file `install.php` in the web-root either manually or by calling `http://127.0.0.1/silverstripe/home/deleteinstallfiles` in your browser (requires you to login).
10. Your installation is now ready. You can access the frontend at `http://127.0.0.1/silverstripe/` and log into the backend at `http://127.0.0.1/silverstripe/admin`.

The site's cache folder

Following the above steps, SilverStripe will automatically create the folder `/tmp/silverstripe-cache-var-www-silverstripe/` on Linux where it will put all of its cache files. This is the recommended behavior. If this doesn't work on your system (which sometimes happens) or you want to use caching on Windows as well, create a folder `silverstripe-cache/` inside your web-root—so it should be located right next to `mysite/`, `sapphire/`, and so on. And don't forget to allow the web server to write to the folder on Linux—just like the `assets/` folder.

Troubleshooting

In case something doesn't work out as expected despite all our efforts, don't panic! Common problems include getting an error message or the dreaded white page of death with no output at all.

Debugging

There are three things you should do to debug errors:

1. Enable SilverStripe's development mode, as it will show you what's going wrong. It isn't enabled by default to guarantee maximum security for default installations. Add the following line to `mysite/_config.php`:

   ```
   Director::set_environment_type('dev');
   ```

2. Take a look at the PHP error log file. If it's empty or you can't find it, correct the log level and its location. This is done in PHP's configuration file, located at `xampp/php/php.ini` or `/etc/php5/apache2/php.ini`. The following settings are important when debugging:

   - Log all errors with the following setting: `error_reporting = E_ALL | E_STRICT`
   - Actually write errors to the log file: `log_errors = On`
   - Define in which file errors should be logged. The example assumes you're using XAMPP on the `C:\` drive: `error_log = "C:\xampp\apache\logs\php_error.log"`

   Don't forget to restart the web server after changing the configuration, otherwise your changes won't be applied.

3. Carefully read the error message. In most cases it will clearly tell you what's wrong.

PHP requirements

SilverStripe is a modern and powerful system. While its hardware requirements are not very demanding, you may have problems on very cheap hosting platforms. These generally manifest in PHP errors. Ensure that you fulfill the following requirements:

- You need at the very least 40 MB of memory (`memory_limit` in the PHP configuration file we used in the previous section). However 64 MB is recommended and you might need even more in some situations: This depends on your site-specific code or in case you want to generate translation files. In case of problems, PHP will choke up an **Out of memory** error.
- Another critical value is the maximum execution time of a PHP script (`max_execution_time` in the configuration). The default of 30 seconds should be enough in most situations, but if you run into **Maximum execution time** errors, you'll need to increase this value.

For a general purpose website we're done. However, if you expect a lot of traffic you might need to fine-tune and configure your live installation even further—SilverStripe is capable of scaling with your demands.

Copying data between the development and live site

This is actually pretty simple. Let's assume you've created your development site and are ready to put the first version of your page into production.

- You'll need to copy all of your files to your live environment—including the core files, your custom code, and anything you've added to the `assets/` folder. This can either be done through FTP, SSH, or a version control system such as Git or Subversion.
- Additionally we need to copy the database. You can either rely on a tool such as phpMyAdmin (`http://www.phpmyadmin.net`) or use MySQL's built-in command line client, at which we'll now take a short look:
    - On the development machine, start the command line and change into the folder `xampp/mysql/bin/`. Assuming the database is called `silverstripe`, we could dump the database to the file `silverstripe.sql` with the following command:

      ```
      mysqldump -u root -p silverstripe > silverstripe.sql
      ```

    - On the server create the database. First you'll need to change into the MySQL environment (you'll be prompted for the password), then we'll create the database and finally we'll leave the environment again:

      ```
      mysql -u root -p
      CREATE database silverstripe;
      \q
      ```

  - Upload the dump to the server.
  - Import the dump with this command:

    ```
    mysql -u root -p silverstripe < silverstripe.sql
    ```

  Now that we've done that, we'll continue to work on the site locally. Once we or our client are satisfied with the result, we'll need to upload it to the server again.

- As the core files shouldn't be changed during site development, we'll only need to upload our custom code and any changes to the `assets/` folder in future.
- Assuming the live site's database hasn't changed, we can simply overwrite it with a dump from our development machine.
- If the live site has been changed, which is the more common situation, we cannot simply overwrite the changes. In this situation you'll most likely need to synchronize any code changes with the "old" database. After adding your code changes call `http://127.0.0.1/silverstripe/dev/build?flush=all` in your browser. You'll obviously need to replace the above address with your own IP address or domain name and the correct folder (if you're using a subfolder). You'll be prompted to log in as an administrator to apply the changes. After that you can add all your other database changes from the development version to the live site—generally by redoing the required steps like adding new pages or other content.

Summary

Getting SilverStripe up and running isn't so hard after all.

In this appendix we've taken an in-depth look at how to install a development system on Windows, including all the components necessary for creating a successful website. And we've also shown how to set up a live environment on Linux (Ubuntu specifically).

Finally we've also explored how to install SilverStripe itself, how to troubleshoot possible problems and how to upload your development site to your live server.