1

Introduction

At it's heart, the role of a content management system (CMS) is to make it easier to publish content to websites and intranets. It may be used to allow the owner of a one-person company to update his or her website without needing a website developer or learning HTML skills. Or, in a multinational company, it might allow a skilled web team across various cities to manage a complex assortment of updates to products, services, promotions, and news in multiple languages. Either way, it automates various tasks, and makes building a website more efficient and reliable.

Countless products are available based on such a promise, however, all of varying sophistication, histories, programming languages, and geographical dominance. The decision-makers and developers involved in selecting a CMS work in fast-paced environments and are creatures of habit: They have personal favorites to solve the challenges their job provides. Why go through the trouble of trying a new CMS?

1.1 Why SilverStripe?

One thing up front: SilverStripe is not the answer to all problems that you might face throughout the development of a website or web application. No software is ever a silver bullet for *every* situation. You must determine selection criteria based on your needs, skills, budgets, and timeframes, and select the most relevant tool based on those criteria. That said, SilverStripe is *intended* to be both used out-of-the-box and for major customization. This makes it particularly interesting when building websites – because everyone seems to relish having a unique set of needs in terms of design,

information architecture, content, features, integration with other systems, business policies, and so on.

The following pages list some major benefits of SilverStripe, and describe how it stands out in a crowded CMS market. We acknowledge that the information here is concise, but this will help you refer back to an overview even when you're familiar with SilverStripe. We try to keep things brief here; the rest of the book will substantiate the claims made, as you begin to build stuff using the software.

1.1.1 An Application and a Framework

SilverStripe is really two products in one: The *SilverStripe CMS* and the *Sapphire Framework*. There are plenty of CMS applications and quite a few programming frameworks in the marketplace; however, SilverStripe is very rare in that it *tightly* weaves the two concepts together.

You will need to read the whole book to understand entirely the value of this unity, but the value is much greater than the sum of its parts. In other words, the CMS has plenty of good things about it, as does *Sapphire*. However, with them joined so closely, their collective value is multiplied. The closeness of the two concepts is possible because *Sapphire* and the CMS were created in tandem, by the same people.

1.1.2 CMS for Content Authors and Framework for Developers

One fundamental reason for SilverStripe being divided into two sections is to honor the needs of two quite separate types of people who use the software. The CMS backend is designed to allow non-technical users to update web pages, images, links, moderate blog comments, and so on. This is the area where content authors do their content management, and anything technical is strictly kept out of sight for the sake of good usability.

The framework is used to build the underlying website, and isn't for content authors. You 'use' the framework by actually writing HTML, CSS, JavaScript, and object-oriented PHP code. The framework enables beginning PHP programmers to achieve quite powerful things, by following recipes and tutorials as shown in this book and elsewhere. However, the framework is targeted at savvy PHP programmers. For them, *Sapphire* allows much more creative freedom in building complex websites quickly and robustly.

Many other CMS products don't have such a separation. In other words, in many systems the CMS user interface is shared by content authors and developers, which typically makes the interface too complicated for it's primary users: the content authors. In some systems, developers perform their work mainly through a graphical configuration interface, which is usually more limiting than raw programming. A graphical interface

to accommodate sophisticated needs for developers would require a bewildering set of options, which would add complexity and bloat the software, when the idea in question is likely to be far more efficiently expressed as a few lines of code.

So, this separation in SilverStripe means that the CMS backend can be designed for non-technical users, and the framework can confidently concentrate on being suitable to technically-minded web developers (see Figure 1.1 for an overview of the SilverStripe system).

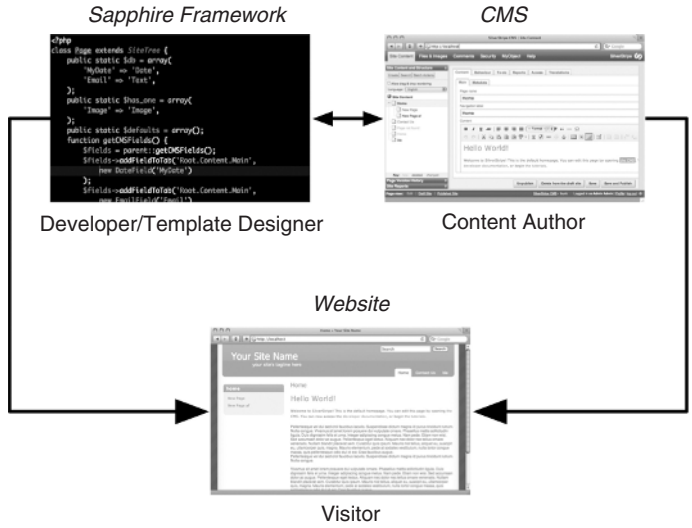

Figure 1.1 SilverStripe system overview.

1.1.3 Clear Separation of Concerns

As we begin to suggest above, a core principle in SilverStripe is the *separation of concerns*. In this introduction, we use that term loosely to mean organizing the software into all sorts of logical divisions. You will be familiar with how HTML and CSS separate the concerns of markup and presentation, and that JavaScript is used to allow for the concern of behavior and interaction on a web page. This structure provides many benefits such as enabling reuse: change a single style in a CSS file, and so long as you've stuck to best practices in your HTML, styles throughout the whole website will update, saving time and maintaining consistency.

SilverStripe follows this principle: Update an isolated area of the system that deals with a particular function, and it produces logical, robust, and application-wide changes. Without this architectural principle, changing a feature of your website means changing a few lines of code here, a few lines of code there, and finding that each step of the way upsets

unrelated parts of your website. This domino effect suddenly turns a small change into a major coding and testing fiasco. SilverStripe's architecture therefore helps to keep code changes proportionate to the effort needed to implement them. However, like the HTML, CSS, and JavaScript example, this reward is only given to those who carefully architect their SilverStripe application to its best practices.

Let's highlight a few examples relating to this principle:

- ***Avoid repitition with the Sapphire framework.*** *Sapphire* supports the concept of writing the logic of your website *once*. For instance, if your website deals with selling T-shirts, you define that a T-shirt has an attribute 'color' in one place in your code. You don't have to write redundant code elsewhere to explain that T-shirts have colors; the template, CMS, and even the database will automatically be updated based on your single declaration. This principle is commonly called *Don't repeat yourself* (*DRY*).
- **No need to change database schema manually.** This important point is hinted at above: SilverStripe will inspect your code for the tables and fields it needs, and update the schema accordingly. This makes keeping your database schema in sync with your PHP code a beautiful thing.
- **Separates the PHP logic into the Model and the Controller.** You're probably familiar with separating PHP logic from HTML templates, but SilverStripe goes one step further. This is a very integral part of *Sapphire*, so let's pause for a moment to investigate this in the following box.

Model View Controller as a central concept

The *Model View Controller* (MVC) concept is one of many *design patterns* that have been identified as good programming architecture. A design pattern is essentially a structured way to document the solution to a specific problem in computer science, a term made popular by the 'Gang of Four' in their book *Design Patterns: Elements of Reusable Object-Oriented Software* (Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, published by Addison-Wesley). The three parts, *Model, View*, and *Controller*, describe the main roles of any application built with SilverStripe:

- The *Model* is responsible for maintaining the state in the application. It's basically data: Some of it might be transient and only live within a web request, whereas other data might be permanently stored in a database. You can think of the *Model* as essentially the database

schema you have for a website. We elaborate on this simplistic definition in later chapters.

- A *View* displays data that's collected from the *Model*, usually through a template on a website or application frontend. One *Model* might have many *Views*: The list of forum users on your website might have a 'public view' and a 'backend management interface view'. In both of these views, the used *Model* logic would be the same: The separation into different roles avoids code duplication and clarifies responsibilities.
- The *Controller* manages the layer in between the *Model* and *View,* and passes data between them. It's the 'glue' of an application. It might take a new forum submission from a *View,* and tell the model to store it in the database. Although the actual input form is rendered in the *View*, the *Controller* is responsible for processing it. The *Controller,* for instance, would ensure that the submitter *can* post to this forum, and might send out emails to moderators or people subscribed to the forum. To clarify, the *Model* should not be responsible for such logic, and instead just store the data once the *Controller* has figured out what needs storing and how. Using the example here, separating the *Model* and *Controller* shows how you can have different form processing logic (multiple *Controllers)* making reuse of the *Model*.

All three roles work closely together. It's just important to understand what goes where on a conceptual and practical level. If that sounds a bit too abstract for you at the moment, don't despair. MVC is a central concept in SilverStripe, and we've dedicated an in-depth explanation to it in Chapter 3, Section 3.2, 'MVC – Model View Controller'.

- **Object Relational Mapper.** The PHP code throughout SilverStripe is heavily object-oriented rather than procedural code, and this architecture extends to how you access the database. This maintains a unified approach to processing data both in and out of your database: you deal with *objects* rather than flipping constantly between arrays, methods, objects, SQL syntax, and so on. This lessens the lines of code used to create, remove, update, and delete data in your database, and improves readability. It also marries the database schema more closely to your PHP code, thereby lessening the risk that these are out of sync. The concept is explained in Chapter 3, Section 3.3, 'ORM – Object Relational Mapping' in more depth.
- **A very flexible data-model.** Because of the ORM and the MVC patterns, SilverStripe is able to consume quite complex data structures.

You might have all sorts of different 'types' of pages, each with different fields. Then you might add products, members, product reviews, coupon codes, and all sorts of other objects to your website. All of this data needs to be stored in your database, managed in the CMS, and presented on your public website. SilverStripe allows you to have a complex entity model, full of relationships, while ensuring the structure and data granularity is elegant.

- **HTML, CSS, and JavaScript.** Being the *lingua franca* of the web, SilverStripe respects that competent website developers need to have control over and separate their website into these three languages. In other words: SilverStripe doesn't make you add CSS into your HTML. Nor does it prevent you from intelligently using JavaScript to *progressively enhance* your website with extra functionality. It stays out of your hair – as a developer, you can craft these files and incorporate them into your project in an elegant fashion. This provides for fantastic flexibility in your markup, therefore providing freedom in visual design.
- **Simple template language.** The 'View' mentioned in the MVC explanation above means that SilverStripe templates don't contain logic. More than simply insisting pedantically that the *Controller* instead should contain this logic – it means that the templates deliberately resemble standard HTML syntax and that they can be created and edited by people without PHP skills. This lets you unleash your zealous HTML- and CSS-skilled designer on SilverStripe templates. Compared to other templating languages, SilverStripe's seems overly simplistic. But you might come to appreciate the benefit of this in the same way you appreciate extracting your CSS out of your HTML: When you need to change something you have only one place to look, making changes easy and encouraging consistency.
- **Class inheritance as a means of customization.** This is really a topic for a later chapter in the book. But it's important, and so we whet your appetite with a preview. If you have the concept of a 'Product' in your application, how do you customize that to make *some* 'Products' a 'Book', which might require an *Author* field? Extensions like this are fundamental to any CMS software package. SilverStripe uses the natural capabilities of an object-oriented language to *extend* Products into Books. This is nothing special in any object-oriented language, and all MVC frameworks do this, but it's rare for a CMS to get out of your way and just let you use these natural features of object-oriented code.
- **Clean file system structure.** SilverStripe's file and folder structure is neatly organized in several ways. The core application lives in a couple of top-level folders. All your project-specific changes go into

your own top-level folders, providing convenient separation from the core package. Files containing code are named specifically based on the PHP classes they define. This means a lot of files, but also makes the role of each file specific and deliberate. Template files follow similar naming conventions, which makes SilverStripe websites portable: You can just move files and folders from one installation to another and be confident about what functionality in the application you're shifting.

- **Modules, themes, and widgets.** The core functionality of SilverStripe can be extended by modules (for example 'forum' functionality), themes (changing the visual appearance of your website), and widgets (small features you can add to a page). We will build all of these types throughout this book. Extensions allow the core to be kept compact and succinct, so you don't have a product with features you don't need. Although the ability to have fully-featured modules isn't a surprise for a CMS, having them tightly integrated at a framework level is a unique characteristic.
- **Automated tests.** SilverStripe provides a neat structure for testing any custom code you write. This means that you can test that your application works, automatically (we talk about this a bit more in Chapter 9, 'Testing'). Automated tests allow you to make a code-level change to your website and test that you've not broken other features in your website. This reduces the *friction* of making changes to your application. In turn, this translates to being more able to continuously update your application, which is very appropriate in the world of 'continuous beta', and supports an *Agile development* methodology (see the following box). By the way, SilverStripe uses automated tests to ensure the consistency of its core functionality as well.

Agile development

This term stands for a software development methodology that fosters a process of frequent inspection and adaption after an initial planning phase. It's in contrast to a more rigid methodology, commonly known as *Waterfall*, where you devise a detailed specification upfront and proceed to build to that specification, generally with fixed budgets and timeframes.

The agile approach embraces the idea that it's impossible to predict exactly what you want ahead of seeing your software take shape. Speaking metaphorically, it's very difficult to foresee the impact of a picture in your living room based on the floor plan. Perhaps the lighting is totally different from what the window placement suggests?

Instead of resisting *change requests*, an agile approach involves constant communication, demonstrations, and changes, as well as re-prioritizing what should be worked on. Agile maintains this notion throughout the process: Your software should be working and 'deliverable' most of the time, as opposed to just being launched at the 'eleventh hour'.

A major benefit of an agile approach is that you're more likely to finish with a *useful* set of features – the features implemented are the most important ones. Done well, agile also serves to maintain project schedules and budgets.

Agile is a way of thinking that involves everyone in a project: It impacts project managers and stakeholders just as much as developers – and it requires discipline from each of them. Agile works best with software that can quickly prototype features and that can support change easily.

More information on agile methodologies is available at `http://agilemanifesto.org/` and `http://agilealliance.org`.

1.1.4 Interoperability and Open Standards

SilverStripe makes substantial use of open and common technologies as first-class citizens. The choice of PHP, MySQL, HTML, CSS, and JavaScript for example is very deliberate: They score well in terms of being well understood by the web industry, are open rather than in the control of a private company, and therefore are likely to be supported for a long time into the future. Here are some specific examples of SilverStripe's treatment of interoperability and open standards:

- **Embracing browser compatibility.** SilverStripe does a lot of work to ensure that your website and administration system hums along in different web browsers. We've already covered the main factor here: SilverStripe gives you full control of HTML and CSS in your templates. It's an indictment on other CMS products that we even have to mention this: But yes, content authors using the SilverStripe backend can use a Mac, Windows, or Linux computer for their work because it performs well in Internet Explorer, Firefox, Safari, and Chrome. With all these browsers competing for supremacy, your website and CMS backend need to be cross-browser compatible now if they're to have any hope of being cross-browser in the future. The core developers of SilverStripe also have a watchful eye on HTML 5, which is quickly gaining traction after a long period of stagnation in the HTML standard.
- **JavaScript and progressive enhancement.** Ideally a website should work with JavaScript disabled, thereby ensuring accessible

functionality and content for both users and search engines. This paradigm has two approaches: *graceful degradation* and *progressive enhancement*. SilverStripe supports these for your frontend templates, both by giving you full control over the templates as explained earlier, and by ensuring that automatically generated markup complies with the principles of progressive enhancement. Although the CMS backend does require JavaScript to function, it shies away from excessive Javascript-generated HTML, making debugging easier, allowing the backend user interface HTML to be edited as template files, and encouraging robust customizations.

- **Database Abstraction.** SilverStripe currently only supports MySQL in a stable release, but the underlying architecture allows for other database drivers. Because of this, SilverStripe has been shown to work on other databases such as PostgreSQL or the Microsoft SQL server. Note that both are currently in an alpha state.

- **Server-side interoperability.** SilverStripe requires a PHP webserver but not much else. It supports running on the two most popular webservers, *Apache* and *Microsoft IIS*. Being able to install SilverStripe onto Windows, Mac, and Linux servers means that you can trial or deploy it on an architecture that you're familiar with or that will better support integration into your IT environment.

- **Accessibility.** Having control over HTML, CSS, and JavaScript means that it's easy to write the markup that validates against *Web Content Accessibility Guidelines* (WCAG) as published by the W3C (see `http://www.w3.org/TR/WCAG20/`), and therefore can be processed easily by screen readers. This is a necessity for government sector work, and in case the standard commercial justification is instead important to you: Good markup positively influences your search engine rank!

- **Unicode.** Not everybody has only 26 letters in their alphabet. New Zealanders typically need the *Macron* so that they can spell 'Māori'. Europeans have various diacritics. The Chinese have tens of thousands of *Han* characters. The *Unicode* standard enables SilverStripe templates and the content entered into the CMS to be in multiple languages and alphabets.

- **Supporting the 'web of data' and it's APIs.** There's an inexorable trend away from websites only providing their content and features in a human-readable format. If your website catalogs a hundred books, or adds a dozen fresh news articles every day, this content shouldn't just be accessible in a gleefully designed HTML and CSS webpage. The information should also be available to download as a CSV document. Or an RSS feed. Or better yet, a full *Web Services API*, allowing the content to be machine-readable in a robust and versatile fashion. This

lets other websites and tools take information from its authoritative source – your website – and promote it elsewhere. However, you're unlikely to bother with such a feature on your website if it adds time or complexity to your project. Fortunately, as Chapter 6, Section 6.10, 'Web services using RESTfulServer' explains, SilverStripe gives you robust and feature-rich APIs with very little code.

1.1.5 Conventions

As you might have begun to determine already, SilverStripe makes a lot of use of conventions. There are also documented PHP coding standards, conventions for HTML and CSS, and naming conventions for functions, methods, and filenames. As a result, you have a well-considered structure to follow for your projects – this allows teams of people to work on SilverStripe projects, and helps you share your work with others in the overall developer community.

Related to this is the principle of *convention over configuration*, which has two major benefits: Efficiency and robustness to the developer and user-friendliness to content authors, as illustrated by these examples:

- The base package 'just works' out-of-the-box. It provides a helpful installer, and immediately can be used to add pages and run a website. A default theme means that it's feasible to write *no* code and have this as a public website – it would just look a little bland. Extension modules such as the blog and forum adhere to this 'works out-of-the-box' principle, too, which makes getting started with SilverStripe quicker.
- Options are only visible in the user interface where they're commonly used. Functions with less frequent usage exist on the sidelines without cluttering up the interface, and rarely used functions are either possible through extension modules or may be excluded altogether. Those rare features might still be useful to you, and are basically waiting for a developer to write them using one of the many extension points in the framework. Although this may make you consider using an older, more 'feature rich' alternative to SilverStripe, do be careful: In general there's a major switch toward using more lightweight tools that are user-friendly. A good CMS is one that doesn't have a ton of functionality you never use.
- The same is true at the code level, in the *Sapphire* framework. The core package focuses on providing commonly needed features easily. Generally, if functionality is missing, it provides easy means for adding, replacing, or changing behavior – by writing code. This is in contrast to having a multitude of built-in options that serve to complicate and bloat the software. This focus is rare in CMS packages.

1.1.6 CMS Interface

So far we've discussed principles and architecture. They're important to mention because they might not be as readily apparent as the functionality that's visible within the CMS backend functions provided by the SilverStripe backend is to try it out. The best way to understand the functions provided by SilverStripe is to try it out, for instance the official public demo at http://demo.silverstripe.com/ or the screencast at http://silverstripe.org/assets/video/cms.html.

Chapter 4 serves to explain the CMS interface in detail, so we'll just mention some interesting features here.

In comparison to other CMSs, the backend is quite clean and uncluttered, and supports intuitive usage without specialized knowledge. Long training sessions should be a thing of the past with SilverStripe for the average computer user who is familiar with using *Microsoft Word* or similar text processing solutions.

Behind the scenes, SilverStripe was an early adopter of highly interactive interfaces built around *JavaScript* and *AJAX*, which means certain areas of the interface can dynamically adapt to user actions without reloading the entire application. AJAX powers key tasks such as opening and saving pages, inserting images and links, and most strikingly, using drag&drop to organize your sitemap. The use of AJAX makes performing these standard tasks much quicker than the traditional approach of having to reload the entire webpage. Other interesting features include the following:

- A site-tree interface instantly provides users with the idea of the organization of the website's pages, and how to edit and update pages.
- Each page has a private 'draft' view, which you can work on until you're happy to publish it live. Each page version is saved, allowing you to revert to and compare older versions of pages.
- There's support for websites having multiple languages and subsites.
- The user interface used by content authors is translated into many different languages, which makes it more accessible to non-English markets.
- There's powerful and automatic image manipulation. If you insert a 10 megapixel image onto your homepage, SilverStripe automatically reduces it to a filesize appropriate for the web.
- The URLs are friendly URLs, of the format *http://yoursite.com/pagename/*.

- You can use a WYSIWYG editor to update content, sidestepping the need for content authors to learn HTML.
- There's a graphical user interface to manage files and other documents, as well as security permissions.
- Some extension modules provide rich additional functionality to the CMS. For instance, the *Userforms module* lets non-technical people create forms for their website. The *Workflow* module provides the ability for larger groups of people to work on a website, ensuring that content goes through an approval process before being published.
- It's highly customizable. Developers can substantially alter the user interface so that content authors have the ability to manage their unique website. This is much more than being able to add a custom field – you can create entirely new data management interfaces, making the CMS backend capable of managing all sorts of data.

1.1.7 Commercial Open Source Software – Best of Both Worlds?

SilverStripe is open source software, which means that you benefit from access to the source code, and of course that it's free to use. It also serves to build a global community of developers who take interest in the software, sharing best practices, code, documentation, roadmap ideas, and so on.

The source code is managed and predominantly produced by a privately held company, SilverStripe Ltd. This in contrast to a community ownership model that often typifies open source. Despite SilverStripe being produced by a company, which would often suggest there being a 'commercial' and a 'free community' version, the SilverStripe company instead focuses on making a single (free) version great.

Having a company behind an open source project typically allows the best of both worlds: On the one hand it provides access to code, a good price, and minimizes lock-in to a vendor. On the other hand, it also provides assurance that there is a number you can call if you want to pay for professional services, such as resolving an issue or commissioning a feature. You can visit http://silverstripe.com/ for more detail on the professional services that SilverStripe and its partner companies offer, or keep an eye on the community forums on http://silverstripe.org/ for freelancers.

However, the devil is in the detail: Open source has become a coveted term and there's variation in just how 'open' different 'open source content management systems' are. Some companies profess to provide open source software, but fail to honor some key principles beyond just providing access to the source code. In SilverStripe's case, the company

does the right thing and provides a number of attributes necessary for the open source litmus test:

- Source code is available in a public versioning repository. All past and upcoming versions of the software are available, with all changes between them shown in a convenient manner. Every change on every line of code can be tracked to a person, a moment in time, and a release.
- The project accepts (and encourages!) contributions in the form of code back to the project. Although nobody outside of the company has *commit* rights to the core package at the moment, contributed patches are regularly added to the core, and many members in the community have commit rights to extensions. So far, the company's control appears to have done a good job prioritizing sound architecture and usability.
- Core developers regularly appear publicly on support forums and use a development mailing list to advise of the roadmap, and actively encourage discussion and input by the wider developer community. A healthy sign of this is that some features rise in the priority list of the core team after community discussion. Multiple language support is an example of an often requested feature that was implemented by SilverStripe based on community feedback.
- The bug tracker and feature request system are all publicly visible and open to the public to use at http://open.silverstripe.com/. A public roadmap and a real-time list of changes to the product are also found here.
- SilverStripe is distributed under the commonly used *BSD* license, which is a *permissive* license approved by the *Open Source Initiative* (http://opensource.org/). SilverStripe refrained from crafting their own open-source license, which means that your legal team will have an easier job reviewing the license details (assuming they know about open source!).

BSD license

BSD is short for *Berkeley Software Distribution,* which is a liberal license model originally developed by the American university of the same name. The BSD license allows copying, modifying, and distributing of software without having to grant access to the altered source code, and is by definition a 'permissive' license. The only requirement is that the original (fairly short) license note has to stay in place. The complete license can be found at http://opensource.org/licenses/bsd-license.php.

In practice, this means that service providers can develop specific extensions for paying customers without obligations to license this work as open source. The created intellectual property stays with the service provider.

SilverStripe is one of very few CMSs that take this liberal approach of BSD licensing. Most open source alternatives use a variant of the *GNU Public License* (GPL), which enforces greater responsibility around contributing your changes back into the core product.

1.1.8 What You Don't Get

Another way to define something is by what it *doesn't* have:

- There's no graphical user interface to manage templates, website logic, alter your database schema, provide a 'control panel', and so forth. These are all managed by editing code at the framework level, offering more flexibility to developers.
- The template files and PHP files are stored on the file system. This means that the responsibility of maintaining a history of old versions of these files, allowing collaboration by multiple people editing these files, and pushing files from 'development' to 'production' is up for you to manage. There's plenty of powerful tools to provide this facility (e.g., *Subversion* and *Rsync*) but SilverStripe doesn't do it for you.
- SilverStripe is not a *document management system*, a term that normally refers to a more heavy-weight product capable of managing many thousands of documents for perhaps just as many staff, and robustly ensuring versioning, integrity, and security of the files. Any document management system worth its salt will have APIs to provide integration with SilverStripe, allowing for instance the uploading of files into the CMS from your document store. SilverStripe does have a powerful 'Files and Images' system that lets website content authors manage thousands of publicly accessible images and documents on your website, and it's well suited to *that* purpose.
- You don't get a Microsoft application with SilverStripe, but if want to want to leverage knowledge or investment in a Microsoft environment, take advantage of SilverStripe being able to run on IIS and talk to SQL Server databases. Use APIs if you need integrate with third party applications such as *Sharepoint.*
- There's no inline editing. Some content management systems allow you to browse the website, find a section of content you want to edit, click it, and edit it right then and there – 'inline'. SilverStripe

has chosen to put more effort into making a 'backend' CMS interface polished and user-friendly. You'll see in this book that forms can be added to the 'frontend', which update pages and content in the CMS, but these are focused on providing specific functions, such as letting members update their profile. However, you *can* have a link shown on webpages that loads the CMS backend for the current page.

- There's not gazillions of modules, themes, widgets, translations, and developers able to give you free support 'yesterday'. If you want something modern you need to be prepared to have less of these amenities. Bear in mind that you don't need so much pre-built code because it should be easy to write it yourself.
- We don't supply an application that runs on the x386 PC you have earmarked for the local museum. SilverStripe is built in an environment where dual-core machines are the lowest-end computers available. That's not to say SilverStripe shouldn't concentrate on client-side performance in the browser, but it makes pragmatic choices. On server-side performance, you can expect several page views to be dynamically generated per second; hundreds if you use static caching.

1.2 History

After giving you a high level overview of what SilverStripe has in store technically, we'd like to give you a little insight into where it comes from. Welcome to our little SilverStripe history lesson!

The website development company SilverStripe Ltd was founded in 2000 in the beautiful capital of New Zealand, Wellington. The three founders Tim Copeland, Sam Minnée, and Sigurd Magnusson originally developed and marketed a closed-source PHP4-based CMS to local small and medium-sized business customers.

In 2005, the business decided to rebuild this CMS from scratch in PHP5, using the experience gathered throughout five years of commercial CMS development. This rewrite was largely motivated by the need to support larger customers with more complex website requirements. The rewrite came with a twist, too: This time it would be open sourced. SilverStripe hoped to foster goodwill and a wider adoption by this move, assuming it would make SilverStripe a better and more mature product in the long run. If they could pull it off and create a thriving community around the newcomer, the feedback and ideas generated would mean welcome improvements to the overall system.

The software was rewritten substantially in 2006, with beta downloads available late that year, and v2.0.0 stable released in February 2007. Just months after the stable release, the project was accepted into the *Google*

Summer of Code, followed shortly later that year by being accepted into the *Google Highly Open Participation Contest* (see later box 'Google's open source initiatives').

These two programs brought SilverStripe to the attention of web developers around the world, and led to many new features being contributed to the software. In 2008, the software won 'Most Promising CMS' at the *PacktPub Open Source CMS awards*, having been a finalist the year before. The software also featured at the annual *MySQL Conference* in California, won at the *New Zealand Open Source Awards*, and was used to power http://demconvention.com/, a high traffic official website in the recent and very visible US Presidential Elections.

In 2009, the software was chosen as one of ten packages bundled with *Microsoft Web Platform Installer*, an automated way to install PHP applications on Microsoft's IIS webserver. The Microsoft installer now accounts for nearly half of SilverStripe's downloads each month. Since launch, the SilverStripe project team have maintained at least one major and a couple of minor releases each year.

On the commercial side, the SilverStripe company has more than tripled its company size since going open source, currently employing over 30 staff. To help grow the company, in 2007 the founders appointed a new CEO, Brian Calhoun, who had spent considerable time working in the Silicon Valley software industry.

All employed developers can dedicate a certain percentage of their work time to open source development. Many new features also find their beginnings in commercial work for specific clients with enough foresight to allow open source usage of their commissioned work. Much of the company's work has come from international clients who became aware of the open source product without any significant marketing effort on behalf of SilverStripe, by virtue of the community spreading the word.

Google's Open Source Initiatives

Through the *Google Summer of Code* (GSOC) and *Google Highly Open Participation* (GHOP) *Contest* programs, the search giant shows generous support for open source software. Both programs encourage students to contribute to established open source projects.

The *Summer of Code* is a reference to the *Summer of Love* that took place in San Francisco, 1967 – and implies similarly youthful and revolutionary energies. The annual event began in 2005, and provides hundreds of paid internships to university students. The students work for about three months over the northern hemisphere summer, and are mentored by members of an open source project to contribute a

specific feature or effort to that project. More information is available at http://code.google.com/soc/.

The *Google Highly Open Participation Contest* connects high-school students with open source projects in a similar fashion, but with paid tasks that take only hours, not months, to complete. Further information is at http://code.google.com/ghop/.

The support from Google not only boosted SilverStripe development, but also led to a good chunk of attention from the global open source community, as well as potential clients. SilverStripe is now recognized beyond the southern hemisphere, and finds a lot of adopters in Europe as well as North America.

1.3 Future

SilverStripe is a fairly young CMS. The price you pay for any modern system is that the documentation, features, and support may be more raw than in an older system – but then, an older system clearly has its share of disadvantages too. It is therefore comforting that the project cares a lot about embracing and furthering website development best practices, and that the core developers and the company show regular commitment to growing both the community and the software.

In most cases, the active developer community is able to respond quickly to any questions. Overall, the development velocity is quite high, and this is the main factor that made this underdog a great CMS. It doesn't look as if development is slowing down any time soon; SilverStripe is seeing increased contributions from the community in the form of themes, modules, and new core developers.

The community is definitely at the heart and pulse of the web, which means that new requirements and technologies are embraced very quickly. This is also the reason for sparse long-term roadmap planning – the next SilverStripe version is always dependent on community contributions and appropriate responses to new requirements. Before writing this book, we decided not to include a comprehensive and detailed SilverStripe API reference for this very reason. The young product is moving a lot quicker than the dead trees that compose these pages. If you're looking for a complete reference, the web guarantees you up-to-date information: http://api.silverstripe.com/.

Looking at download figures, the trend is pointing upwards. In late 2008, two years after the first betas, 100,000 downloads were reached. Less than a year later, downloads hit 200,000. A good indicator is also the adoption rate among third party developers: Numerous commercial

service providers have included SilverStripe in their service portfolio. SilverStripe is currently transforming from the 'secret weapon' for early adopters to a broadly used application.

1.4 Conclusion

SilverStripe is a user-friendly and versatile CMS, and gets real strength through its MVC underpinnings and developer-friendly framework.

It's not yet at a point where it can compete feature-by-feature, extension-by-extension with some of the more established systems in the market, but it doesn't really have to: The underlying architecture compensates for many missing bullet points because it's comparatively easy to extend and customize the out-of-the-box experience. SilverStripe's focus on using open standards and common technology not only helps to make extending the software easier, but also helps to future-proof projects you make with it.

The BSD license is particularly interesting for commercial service providers who are interested in developing extensions for their clients, both open and closed source. The ability for technical support to come from commercial and community sources makes it suitable for both small and large budget projects.

SilverStripe is able to score well in terms of cost-effective development: Websites and web applications are easy to get started, and importantly, can be quickly customized in an agile fashion. This agility can be a significant competitive advantage.