The Definitive Guide to SugarCRM
Better Business Applications

John Mertic
# Contents at a Glance

- Contents at a Glance .................................................................................................. iv
- Contents .................................................................................................................. v
- About the Author ...................................................................................................... xi
- About the Technical Reviewers ............................................................................... xii
- Acknowledgments ..................................................................................................... xiii
- Introduction .............................................................................................................. xiv

## Part 1: The SugarCRM Platform
- Chapter 1: What Is SugarCRM? .............................................................................. 3
- Chapter 2: MVC Architecture ................................................................................. 11
- Chapter 3: Metadata Layer ..................................................................................... 35
- Chapter 4: Web Services ......................................................................................... 55
- Chapter 5: More Platform Features ........................................................................ 77

## Part 2: Customizing SugarCRM Out of the Box
- Chapter 6: Easy Customizations Using Studio ..................................................... 109
- Chapter 7: Workflows and Logic Hooks ................................................................. 125
- Chapter 8: Overriding Existing Code ..................................................................... 151

## Part 3: Building New Functionality on Top of SugarCRM
- Chapter 9: Using Module Builder to Build Custom Modules .............................. 171
- Chapter 10: Building a Custom Module Manually ................................................. 195
- Chapter 11: Designing a Complete Application .................................................... 219
- Index ....................................................................................................................... 219
# Contents

- Contents at a Glance ........................................................................................................ iv
- Contents ........................................................................................................................... v
- About the Author ........................................................................................................ xi
- About the Technical Reviewers ................................................................................ xii
- Acknowledgments ....................................................................................................... xiii
- Introduction ................................................................................................................ xiv

## Part 1: The SugarCRM Platform ............................................................................. 1

- **Chapter 1: What Is SugarCRM?** ........................................................................ 3
  - The Company ............................................................................................................ 3
  - Rapid Growth—Harnessing Open Source and SaaS ................................................... 4
  - The Product .............................................................................................................. 4
  - The Community ...................................................................................................... 7
    - Sugar Forums and Wiki ........................................................................................ 7
    - SugarExchange and SugarForge ......................................................................... 7
    - Sugar Developer Zone ......................................................................................... 8
    - Getting SugarCRM .............................................................................................. 8
  - Summary ................................................................................................................ 9

- **Chapter 2: MVC Architecture** .......................................................................... 11
  - What Is MVC? ........................................................................................................ 11
  - MVC the Sugar Way ............................................................................................. 12
    - SugarApplication ............................................................................................... 13
    - SugarController ................................................................................................. 15
    - SugarView .......................................................................................................... 18
<table>
<thead>
<tr>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrypoints</td>
</tr>
<tr>
<td>What About the Model?</td>
</tr>
<tr>
<td>Bean Classes and SugarObjects</td>
</tr>
<tr>
<td>Vardefs</td>
</tr>
<tr>
<td>Fields</td>
</tr>
<tr>
<td>Indexes</td>
</tr>
<tr>
<td>Relationships</td>
</tr>
<tr>
<td>Database Layer</td>
</tr>
<tr>
<td>Sugar Before MVC</td>
</tr>
<tr>
<td>Summary</td>
</tr>
<tr>
<td>Chapter 3: Metadata Layer</td>
</tr>
<tr>
<td>DetailView and EditView</td>
</tr>
<tr>
<td>How the DetailView and EditView Classes Work</td>
</tr>
<tr>
<td>SugarFields</td>
</tr>
<tr>
<td>ListViews</td>
</tr>
<tr>
<td>Defining Search Fields</td>
</tr>
<tr>
<td>Field Overlays</td>
</tr>
<tr>
<td>Subpanels</td>
</tr>
<tr>
<td>Summary</td>
</tr>
<tr>
<td>Chapter 4: Web Services</td>
</tr>
<tr>
<td>What Are Web Services?</td>
</tr>
<tr>
<td>Connectors</td>
</tr>
<tr>
<td>Building the Connector Source</td>
</tr>
<tr>
<td>Defining the Fields from the Web Service</td>
</tr>
<tr>
<td>Formatters</td>
</tr>
<tr>
<td>Localization</td>
</tr>
<tr>
<td>Pulling It Together</td>
</tr>
<tr>
<td>Web Services API</td>
</tr>
<tr>
<td>SOAP</td>
</tr>
<tr>
<td>Make a Connection</td>
</tr>
<tr>
<td>Get Lists and Counts of Records</td>
</tr>
</tbody>
</table>
Add a New Record .......................................................................................................................... 68
Saving and Retrieving Attachments ............................................................................................... 70
Relate Records .................................................................................................................................. 71
REST ................................................................................................................................................... 72
Logging in Using REST ....................................................................................................................... 73
Custom Web Services ......................................................................................................................... 73
  Create a Custom Registry .................................................................................................................. 74
  Provide an Implementation Class ..................................................................................................... 74
  Add soap.php and rest.php Files ...................................................................................................... 75
Summary ............................................................................................................................................... 75

Chapter 5: More Platform Features ................................................................................................. 77
User Management ................................................................................................................................ 77
  ACL ................................................................................................................................................... 79
  Password Management ....................................................................................................................... 81
  Tracking User Actions .......................................................................................................................... 82
Team Management ................................................................................................................................ 84
  Dashlets ............................................................................................................................................. 85
    ListView Dashlet ............................................................................................................................... 86
    Chart Dashlets .................................................................................................................................. 89
    Iframe Dashlets ................................................................................................................................. 92
    Custom Dashlets ............................................................................................................................... 93
Sugar Feeds .......................................................................................................................................... 94
Record Importing and Exporting .......................................................................................................... 98
Themes ................................................................................................................................................. 102
Logger .................................................................................................................................................. 104
Summary ............................................................................................................................................... 106

Part 2: Customizing SugarCRM Out of the Box ................................................................................ 107
Chapter 6: Easy Customizations Using Studio ................................................................................. 109
  Starting with Studio ............................................................................................................................ 109
  Adding New Fields to a Module .......................................................................................................... 111
  Customizing View Templates ........................................................................................................... 113
Part 3: Building New Functionality on Top of SugarCRM ................................................. 169

Chapter 9: Using Module Builder to Build Custom Modules ............................................. 171

Getting Started ........................................................................................................................ 172
  Designing a New Module ......................................................................................................... 174
    Fields .................................................................................................................................. 177
    Metadata .............................................................................................................................. 180
  Relationships .......................................................................................................................... 184
  What Do I Do with This Package? .......................................................................................... 187
    Deploy to the Current Instance ........................................................................................... 187
  Publishing Your Changes and Installing Them with Module Loader ................................... 188
    Exporting the Package ........................................................................................................ 191
    Removing an Installed Package .......................................................................................... 191

Summary .................................................................................................................................. 193

Chapter 10: Building a Custom Module Manually ............................................................... 195

Why Would You Do This? ....................................................................................................... 195

What You Need to Do ............................................................................................................. 196
  Construct Directory Structure ................................................................................................. 196
    Register Module with the Instance” .................................................................................... 198
    Add Bean Class File” .......................................................................................................... 200
    Build vardefs ....................................................................................................................... 202
  Build Any Metadata Templates ............................................................................................. 204
  Add Language Files ................................................................................................................ 211
  Create a Dashlet ..................................................................................................................... 212
  Add Any Additional Views Needed ....................................................................................... 215

Summary .................................................................................................................................. 217

Chapter 11: Designing a Complete Application .................................................................... 219

Background of the Application ............................................................................................... 220

Breakdown of the Various Components .................................................................................. 221
  Studio ...................................................................................................................................... 222
  Accounts and Contacts ........................................................................................................... 222
  Module Builder ....................................................................................................................... 224
  Customers ............................................................................................................................... 224
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jobs</td>
<td>226</td>
</tr>
<tr>
<td>Suppliys</td>
<td>226</td>
</tr>
<tr>
<td>Vendors, Vendor Contacts, and Vendor Orders</td>
<td>227</td>
</tr>
<tr>
<td>Custom Code</td>
<td>228</td>
</tr>
<tr>
<td>Accounts and Contacts</td>
<td>229</td>
</tr>
<tr>
<td>Jobs</td>
<td>229</td>
</tr>
<tr>
<td>Supplies</td>
<td>233</td>
</tr>
<tr>
<td>Vendors</td>
<td>239</td>
</tr>
<tr>
<td>Invoicing Hooks</td>
<td>243</td>
</tr>
<tr>
<td>Final Touches</td>
<td>246</td>
</tr>
<tr>
<td>Summary</td>
<td>248</td>
</tr>
<tr>
<td>Index</td>
<td>251</td>
</tr>
</tbody>
</table>
About the Author

John Mertic is a software engineer at SugarCRM, and has several years of experience with PHP Web applications. An avid writer, he has been published in php|architect, IBM Developerworks, and in the Apple Developer Connection, and has been a speaker at several developer conferences. He has also contributed to many open source projects, most notably the PHP project where he is the creator and maintainer of the PHP Windows Installer. John is happily married to his lovely and exceptionally supportive wife Kristy. Together they have a daughter Mallory and a dog Dominic.
Roger Smith is currently an Engineering Manager and Staff Engineer at SugarCRM. Prior to SugarCRM, Roger held Software Engineering positions at Strikelron, a Web Services company focused on providing commercial data as a service. He also served as a Software Engineer at Cisco Systems on their E-Commerce and Government Solutions teams. Roger holds a Masters degree in Machine Learning from Columbia University and a BS in Mathematics from the University of North Carolina at Chapel Hill.

Collin Lee has been working at SugarCRM as a platform developer for over three years. He has also previously worked for IBM and Xerox Corporation. In his spare time, he enjoys cooking, running, and experimenting with software technologies. He currently lives with his wife in New York City.

Matthew Heitzenroder, SugarCRM’s Community Manager, is a fanatic about the power of community and democratization of software. Four years ago, Matt joined SugarCRM’s Support team, dedicating himself to provide an outstanding customer support experience. He made a transition to become a Senior Professional Services consultant, implementing SugarCRM in some the companies largest clients and most demanding projects. Today, Matt’s passion for Open Source and it’s ideals has naturally lead to a career of empowering and advocating the SugarCRM community of developers, consultants, and users. He happily lives life with his wife in sunny Miami, Florida, sailing and diving every chance he gets.
Acknowledgments

One thing that I’ve learned over the past 8 years is software development is hard work. But I never realized that writing about software development is even harder.

Looking back on writing this book, I am simply amazed at the gift God has given me to accomplish such an amazing feat. He stuck by my side, even when it was 2 a.m. and I was trying to make it through a few more paragraphs, and for that I am the most grateful. There is also an amazing group of individuals that he put here who I am in tremendous debt to that made this book possible.

I’ll begin by thanking the entire team at SugarCRM for all their help and support with the book. I’d like to personally thank SugarCRM’s founders, John Roberts, Clint Oram, and Jacob Taylor, for starting the project and the entire company from scratch five years ago. I want to personally thank the reviewers Matt Heitzenroder, Roger Smith, and Collin Lee for giving me great advice and direction in making the book such a resounding success. And, I want to thank all those at Sugar who have put their time and energy into developing SugarCRM into a world class application. My hope is that this book puts a light to the high quality engineering that has been put into the product.

I would also like to thank Apress, especially Steve Anglin, Michelle Lowman, James Markham, and Dominic Shakeshaft for giving me the opportunity as a new author to write this book. The group at Apress put a ton of effort into making this book a reality, even when the schedule seemed to work against us at times.

And last, but certainly not least, I would like to thank all my friends and family that have provided me support and encouragement over the years, especially during the writing of this book. The biggest thank you goes to my exceptionally supportive wife Kristy, who put up with all the long writing nights and weekends (and family vacation) in my efforts to put together this book. I could never have done this without you.
Introduction

I started this book with the intention of bringing a new side of SugarCRM to light. Since I began working at SugarCRM, I saw the flexibility and extensibility that the application could provide. I looked back on my previous position developing internal business applications, and saw that many of the features I added and design issues I would wrestle with were problems that SugarCRM had already solved. The engineering team at SugarCRM had built the application to solve this problem, yet few developers outside of SugarCRM really knew how powerful the underlying platform was. I knew there were other developers in this same boat, and that if I could reach them it would make their jobs much easier.

What a CRM application does or doesn't do isn't authoritatively defined; instead, its goal is to fill in the gap where a company needs to solve problems in their relationships with their customers. Sometimes this means keeping track of meetings and phone calls. Other times, this means tracking the progress of an ongoing project. It could also mean managing support cases and product defects. Yet sometimes an application may not completely cover this. Just as every business or organization is unique, so must be what CRM will mean to them. Up until SugarCRM, this application space was full of players who thought they had the CRM problem solved, and built large proprietary applications that were expensive to implement and support and notoriously difficult to customize to meet their needs. SugarCRM came in and changed that scene, making CRM something that is inexpensive to implement, easier to customize, and more approachable for end-users to work with. It's designed to be a CRM that your users won't hate, which is the philosophy that the founders of SugarCRM set as their paramount goal when building it.

This book is designed to take this easy-to-use and customizable application and show you what you need to do with it. The contents of this book are unique as they come directly from the engineering experience of SugarCRM, giving you as the reader an insight into the application that you can’t find anywhere else. I've broken the book down into three distinct parts:

- Part 1: The SugarCRM Platform
- Part 2: Customizing SugarCRM Out of the Box
- Part 3: Building New Functionality on Top of SugarCRM

While this book is designed to be read from beginning to end, it's also useful as a general reference manual when developing on SugarCRM. Once you have the knowledge of how the application works internally, you can go back to the book easily to pick up any tidbit of information you might need as you work with the platform. All of the information contained within is current with SugarCRM version 5.5, and most of the examples are built upon the community edition of SugarCRM. I encourage you as you read through the book to download and install SugarCRM on your local machine and try the examples out to see how easy it is to work with. This book is only the tip of the iceberg in what can be done with SugarCRM. Therefore, I also encourage you to also visit the Sugar Forums (http://www.sugarcrm.com/forums) and the Sugar Developer Zone (http://developers.sugarcrm.com) for more about what Sugar can do for you and where we are going in the future.

Thanks for picking up this book and taking a chance on SugarCRM. My hope is that it can help you out in your future applications for your business or organization.
In this first part of the book, you’ll learn all about SugarCRM; from the company and community to the various features of the platform. You’ll see in depth how the MVC and metadata frameworks drive the core of the application. You’ll also see how you can integrate SugarCRM with various other applications using the feature rich web services platform, and learn about many other features the Sugar platform offers the user and the developer.
What Is SugarCRM?

SugarCRM is a commercial open source company. It’s not often that “commercial” and “open source” go together. When we think of commercial software we think proprietary, closed-source software. On the flipside, when we think of open source software, we think of free or “libre” software that is community driven and community oriented. SugarCRM is unique in that it breaks the expectation of how commercial software works, leveraging the best of how open-source software is designed and built to create a product that is focused on the end-users and developers, creating a positive experience for both groups. But it also has the advantages of a commercial company, which includes world-class support, comprehensive end-user training, and end-to-end quality assurance testing to ensure high product stability.

Let’s look at SugarCRM from a few different perspectives: as a company, product, and community.

The Company

SugarCRM was founded in 2004 as an open source project on SourceForge, http://www.sourceforge.net, one of the Internet’s largest open source development sites. SugarCRM’s three founders, John Roberts, Clint Oram, and Jacob Taylor, had a combined experience of over 50 years building proprietary Customer Relationship Management (CRM) applications for Silicon Valley companies. They had grown frustrated with the lack of innovation in CRM and the high failure rates of proprietary CRM applications. SugarCRM’s founders took an unusual approach in building a CRM solution. Rather than write the code in secret and keep the product proprietary, the founders released the code with an open source license and allowed for any interested party to download, modify, and redistribute SugarCRM.
In just a few months, the application was downloaded 50,000 times and translated into ten languages. In November 2004, the Sugar Open Source Project was selected as Project of the Month by SourceForge.net. The popularity of the application allowed SugarCRM founders to incorporate a business around the open source project and receive $2 million in venture capital funding from Draper Fisher Jurvetson, a leading Silicon Valley venture capital firm.

Rapid Growth—Harnessing Open Source and SaaS

The popularity of SugarCRM on SourceForge and an infusion of capital from Silicon Valley investors allowed the company to begin expanding. SugarCRM established a headquarters in Cupertino, CA and began building out its engineering team. In early 2005, SugarCRM introduced Sugar On-Demand, which is a “Software-as-a-Service” (software is provided for a user via a hosting service) that allows customers to use Sugar without installing software on premise.

The adoption of an Open Source and On-Demand product offering positioned SugarCRM at the nexus of two major technology waves. The first—On-Demand—promotes more flexibility and control over how the application is installed, customized, and used. The second—On-Demand—gives users the ability to use CRM software without having technical expertise in-house.

SugarCRM’s innovation in the marketplace was to champion both approaches. Previously, companies had offered On-Site or On-Demand, rarely both. In all cases, the code was kept proprietary which means it was very difficult for customers to understand what they were purchasing, and nearly impossible to modify the code without large investments in professional services and long project timelines.

The Product

Customer Relationship Management is a well-established industry that has evolved over the past two decades. Simply put, CRM is about using information technology to gain a better understanding of customers and deliver a differentiated customer experience across the entire relationship. Think of it as turning your customers inside out, giving businesses a tool to learn what their history is, buying trends are, and interactions have been, allowing you to use this knowledge to plan ahead for your interactions with them. CRM suites, such as SugarCRM, provide tools to all customer-facing employees—marketing, sales, customer support—as well as provide collaboration tools to ease communications and reporting functionality, so managers can understand what is happening in their business. CRM suites also provide administration tools to manage users, information flow, customizations, and other “behind-the-scenes” operations of the CRM system.
SugarCRM began as a sales force automation tool and quickly expanded to include marketing automation functionality and customer support, as well as collaboration and reporting across all parts of the application. It is a web-based application written mostly in the PHP programming language, supporting version 5.2.1 and greater as of SugarCRM version 5.5.0, as well as using the latest javascript and flash techniques to enhance the user experience. It supports running on the MySQL, Microsoft SQL Server, and Oracle database servers (Oracle is only supported in the Enterprise Edition) as well as deployments on Windows, Linux, Mac OS X, and Solaris. Being a browser-based application, it allows the end-users to use either Internet Explorer, Mozilla Firefox, or Apple’s Safari browser.

SugarCRM comes in three product editions: Sugar Community Edition contains core CRM functionality designed for small businesses. Sugar Professional contains additional functionality to manage the needs of small-and-medium-sized business. Sugar Enterprise contains the CRM features and support for large enterprises. The following discusses each edition in detail:

- **Sugar Community Edition**: A Free Open Source Software (FOSS) licensed under the GNU General Public License Version 3 (GPLv3), the newest and one of the many prevalent open source licenses in the software world. Sugar Community Edition is available for free download at SugarCRM’s development environment at http://www.sugarforge.org. Customers are free to download, modify, and use Sugar Community Edition without restriction.

- **Sugar Professional**: SugarCRM’s flagship product, targeted at small and medium-sized businesses. It contains additional functionality, such as team management, access control, reporting, and wireless device access. It is offered under a commercial license. Sugar Professional is offered under an annual subscription.

- **Sugar Enterprise**: Contains enterprise-grade functionality for large enterprises. It adds additional features that most large scale deployments require, such as Oracle database support and Advanced SQL reporting. It is also commercially licensed and offered under an annual subscription, just like Sugar Professional.

Figure 1-1 provides a graphical outline of what is included with the various editions of SugarCRM.
SugarCRM is further customizable by allowing multiple deployment options, which allows even companies without IT expertise to use SugarCRM.

- **Sugar On-Demand**: Allows users to set up and run SugarCRM within minutes, using SugarCRM Managed Data Centers. Advantages include guaranteed uptime and management of your SugarCRM instance, which takes away all the upgrade and server maintenance worries.

- **Sugar On-Site**: Allows users to install SugarCRM on their existing hardware. In addition, SugarCRM has a Faststack installer that is designed to install all the required components (Web Server, Database Server, PHP) for SugarCRM as well.

The beauty of flexible deployment options is that even users using Sugar On-Demand can customize how SugarCRM works through the powerful, yet easy-to-use Module Builder and Studio tools. (We’ll learn more about these in Parts 2 and 3 of
this book.) This allows anyone using SugarCRM to customize it to meet their needs without having the burden of supporting a server and installation. In addition, they can leverage both the On-Demand and On-Site deployment options at no additional charge. This is handy so that one system can be used as a backup of the other, or one can be used as a development/staging system and the other as a production system.

SugarCRM’s growth has continued month after month since being founded as an open source project in 2004. To date, SugarCRM has been downloaded over 5.2 million times, with over 55,000 active systems in over 195 countries supporting nearly half a million users. In addition, SugarCRM has 4,500 paying customers that use the commercial versions of the product to power their enterprise.

The Community

The key to SugarCRM’s success lies in the large community of users and developers around the world. Their feedback and contributions have helped shape the product, and have been an invaluable resource for helping SugarCRM grow. SugarCRM as a company has recognized this and have built several tools for the community to interact with SugarCRM and other users of SugarCRM around the world. Let’s take a look at them.

Sugar Forums and Wiki

The easiest place to interact with the Sugar community is at the Sugar Forums (http://www.sugarcrm.com/forums/). Here is where members of the SugarCRM team, end-users, partners, and developers interact to discuss issues with the product, learn about upcoming features and releases, and learn how others are using SugarCRM within their industry. The forums are driven by both members of the SugarCRM team and community volunteers and is the easiest way to participate with the SugarCRM community.

The Sugar Wiki (http://www.sugarcrm.com/wiki/) is an invaluable resource for tips and tricks when working with SugarCRM. This is a mostly community-driven wiki, but also provides official SugarCRM documentation on the product and podcasts from SugarCRM team members on a variety of SugarCRM and industry topics.

SugarExchange and SugarForge

SugarExchange (http://www.sugarexchange.com/) is the SugarCRM marketplace where any Sugar user wishing to extend core Sugar functionality can choose among hundreds of module extensions, themes, and language packs provided by Sugar community members and partners. It is the go–to place when you are looking for functionality that can be easily added to your SugarCRM instance. SugarExchange
contains both free and non-free add-ons to SugarCRM. While SugarExchange is facilitated by SugarCRM, all transactions and support for the add-ons offered is completely independent of SugarCRM, which provides a forum for the SugarCRM developer community to showcase add-ons to the product.

SugarForge (http://www.sugarforge.org/) is the developer side of the SugarExchange, which provides developer and project collaborations tools for those developing on the SugarCRM platform. This is designed to offer features similar to SourceForge or Google Code, and provides forums, and documentation space for your add-ons. It is often used in conjunction with SugarExchange, where SugarExchange is used to help feature the add-ons to the Sugar Community as a whole. There are over 600 active projects on SugarForge to date, including over 80 language translations offered for free download.

Sugar Developer Zone

If you do any development on top of SugarCRM, this is the place to be. It is a comprehensive resource for any Sugar developer, with links to the official SugarCRM developer guide, developer forums, and tutorials on common customizations and topics. It also features a blog that is run by the SugarCRM team, which provides developers insight on upcoming developer features in SugarCRM or tips and tricks on developing applications with SugarCRM.

As you can see, SugarCRM isn’t like typical commercial software, but it has the more polished feel of typical open source software. This book focuses on this distinction, so in Part 1 we will take an in-depth look at the SugarCRM platform and the features of the product that make it ideal for building a business application.

Getting SugarCRM

In order to best follow along with the examples in the remaining chapters, you should probably download and install SugarCRM. The community edition of SugarCRM is available for download at http://www.sugarcrm.com/crm/download/sugar-suite.html. You have two options for installing SugarCRM from this site. One option is to download the zip archive that contains the application, and install it on your local machine or Web Server. In order to do this, you’ll need the following components installed and configured:

- **Web Server**: Either Apache 1.3 or later or IIS 6 or later with FastCGI installed if you are using Windows.
- **PHP**: Version 5.2.1 or later installed and configured to be used with the above Web Server.
- **Database Server**: Either MySQL 5.0 or later or SQL Server 2005 or later.
To install SugarCRM, simply open a web browser and point it to the location where SugarCRM was unzipped. If you unzipped it into the sugar directory in the root of the Web Server on your local machine, point your web browser to http://localhost/sugar, and then the Sugar interactive installer will guide you to the remainder of the setup process.

To make it easier to get the stack installed, SugarCRM provides several “faststack” installers that will install SugarCRM along with the complete Apache, MySQL, (or SQL Server Express for Windows), and PHP stacks, so you can be ready to run in no time.

Summary

In this chapter, you looked at SugarCRM, and saw how the open source and community-driven nature of the company has helped the product grow into such a success. You then looked at the product, learning the various editions available as well as the many different ways that SugarCRM can be deployed and used by businesses. Next, you turned toward the SugarCRM community, seeing how the SugarCRM as a company and the Sugar community can interact through both forums and wikis to add-on package repositories and marketplaces. Finally, you took a brief look at installing Sugar, so you can follow through with the examples in the remainder of this book.

Let’s continue on with a deep dive into SugarCRM’s underlining platform. In Chapter 2, you’ll look at the MVC framework that Sugar is built upon.
With the advent of SugarCRM 5.0, a new MVC architecture was born. This architecture was designed to eliminate the painful tasks in building a module in SugarCRM. Instead of having to manually lay out templates and set up object interactions and relationships, you can easily leverage the framework by using standardized templates and definition files for building the various views. The system is also very extensible, allowing new views and custom templating to be built on top of it as well.

Before digging into how the SugarCRM MVC model works, let’s take a step back and see what MVC actually is.

What Is MVC?

MVC stands for Model View Controller, and is a very common architectural pattern used in both web and desktop application design. The goals of using the MVC pattern is to separate the user interface logic from the application logic, having a layer in between to facilitate the communication between them. Each one of the Model, View, and Controller components are tasked with handling certain roles within an application (see Figure 2-1).

![Diagram of the MVC pattern](image-url)
The model layer represents the application logic layer. The goal of this layer is to handle the communication with any external resources, such as databases, Web Services, and files. It also contains any business logic in the application, such as ways to calculate field values. A good model provides a clean interface to the guts of the application, providing methods and functions to easily interact with the lower level services and provide any needed transformation or interpretation so that other parts of the application can easily use it.

The view layer represents the user interface. This is where any display logic is dealt with, such as form layout and data display. It is also designed to how it’s consumed, so for a web application a view would be a typical web page. The view layer is specific to what role it is meant to have, so a data entry view would be different than a record display view, even though they may represent the same model.

The controller layer is the glue between the model and view layer. A good controller will accept the request from the user, calls upon a model for the information it needs, and then calls upon a view to return that information to the user. This layer is meant to be a thin layer. It shouldn’t contain business logic, communicate with a database directly, or deal with the how to display information to the user.

Sugar has used the MVC pattern to replace the aging architecture used before version 5.0. Although it was based upon many of the principles of the MVC architecture, it was not truly optimized in a way to really take advantage of it. Let’s look now at how SugarCRM does MVC.

**MVC the Sugar Way**

Sugar uses the MVC pattern to handle requests from the users. Each request to the primary entrypoint (index.php) will specify HTTP request variables indicating the module (which maps to the MVC definition of controller) and the action (which maps to the MVC definition of view) as follows:


For the preceding URL, the request to Sugar would return the EditView action of the Contacts module. Figure 2-2 shows what happens internally when that request is made.
Let’s take a look at each component in the MVC flowchart.

SugarApplication

The first step in the request handling happens at the SugarApplication level. This class handles much of the prerequisites needed for the Sugar application, including session checking, user authentication, setting the theme, and making the connection to the database. It also handles much of the preprocess logic of the application. For example, setting the user time zone on first login and letting the user know when their password is expired. The class is not designed to be modified or extended by developers. However, a preprocess method does exist in the SugarController class which allows developers to interject logic into the handling of the request before the controller is executed.

After loading the controller, the following tasks are done before executing the controller:

- User authentication takes place. If the user is already authenticated then you’ll continue on with the request, otherwise you’ll redirect the login form so the user can authenticate. I’ll talk in more detail about user authentication in Chapter 5.
- Using the settings from the user, any Access control list (ACL) rules are applied for the cases of what modules to expose to the user.
• Any preprocess rules from that are defined on the application level or are applied by the controller. Setting the time zone for the user on their first login is handled here.

• The correct language strings are loaded, as set by the user upon login. In Chapter 5, I’ll talk about how SugarCRM can be internationalized for any language.

• The theme that is currently selected by the user is initialized and loaded. I’ll talk more about how themes work in Chapter 5.

The preProcess() method in the controller is best used when there needs to be some initial logic before you do anything with a module. Listing 2-1 provides an example of what that would be.

**Listing 2-1. Example of a preProcess() Method in the Controller**

```php
public function preProcess()
{
    global $current_user;

    if ( !is_admin($current_user) ) {
        echo 'This module is for admin use only!';
        sugar_die();
    }
}
```

You would use this if the module is going to be an admin-only module, so you could check initially in the controller to see if the user has access to it or not. This avoids the need to add this logic to every view in the module.

Once all of these prerequisites are handled, you can then move on to the SugarController, which is designed to take the request and execute it.
SugarController

The SugarController class handles the main flow of the request, and is designed to control all requests for the module specified. The SugarController implementation maps very well to the MVC model's interpretation of what a controller will do, and contains several action hooks for common actions you would have in a typical module. These actions, available out of the box by default, include complete implementations for DetailView, EditView, ListView, and saving and deleting records. Several of these have definition files, or metadata, that is used to define what they look and act like, which you'll learn more about in Chapter 3.

The SugarController will provide representation for each of the views of the module for which it controls. It does this is by providing two ways of handling the mapping to a view. The first way is by having an actual action method in the controller class which represents the action and transfers control over to the view. At a minimum, this method will specify the view to use with this action, but it can also do some logic at this level that isn't really related to the view layer. Listing 2-2 shows an example view, which I'll call 'getthemail'.

Listing 2-2. action_getthemail()

```php
public function action_getthemail()
{
    if ( mailExists() )
        $this->view = 'getthemail';
    else
        $this->view = 'hasnomail';
}
```

The action hook will first add a check to see if mail exists before redirecting to the 'getthemail' view. If there is no mail, then it will use 'hasnomail' view instead. This kind of situation is a common one, and can provide hooks to direct different views to use depending upon the state of the application or the type of request being made. For example, you could have different views based upon whether json or HTML data is being requested, but the core logic could use the same action hook.

If you have additional controller logic that needs added, such as handling extra request parameters, you can subclass the SugarController class by naming it ModulenameController and saving the file as controller.php in the module's directory. Listing 2-3 is an example where you will override the Meetings module EditView action to allow marking a meeting completed by passing the request variable 'close'.
Listing 2-3. MeetingsController Class

```php
class MeetingsController extends SugarController
{
    public function action_editview()
    {
        if ( isset($_REQUEST['close']) )
            $this->bean->status = 'Held';
    }
}
```

In the preceding, you simply added a new piece of logic to the EditView action which will set the status of the Meeting to 'Held', if you have requested to close it. The bean object corresponds to the data record of the module being requested. The record id is grabbed from the request variable 'record', and it is initialized and loaded automatically in the SugarController class as a part when it processed the request, prior to calling the actual action logic.

Sometimes you may want to be selective about overriding a controller action, by adding logic before or after the action is called. The SugarController class provides hooks for this, by having pre_action() and post_action() methods for each action, which are called before and after the action is called, if they are defined. There are multiple uses for this. Listing 2-4 shows an example where you will check for a changed value of the status field during a save of a Bugs module record in the BugsController class.

Listing 2-4. BugsController Class

```php
class BugsController extends SugarController
{
    protected $_prevStatus = '';

    public function pre_save()
    {
        $bugFocus = new Bug;
        $bugFocus->retrieve($_REQUEST['record']);
        if ( isset($bugFocus->id) )
            $this->_prevStatus = $bugFocus->status;
        parent::pre_save();
    }

    public function post_save()
    {
        if ( ($bugFocus->status != $this->bean->status)