REPORT GENERATOR AND SERVER DATABASE

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Monica Uthappa Maleyanda

Fall 2012
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Monica Uthappa Maleyanda:

Report Generator and Server Database

Carl Eckberg, Chair
Department of Computer Science

William A. Root
Department of Computer Science

J. Carmelo Interlando
Department of Mathematics and Statistics

Nov 5, 2012
Approval Date
DEDICATION

I would like to dedicate this thesis to my parents for their unconditional love and abundant blessings that have made me everything I am today.
ABSTRACT OF THE THESIS

Report Generator and Server Database
by
Monica Uthappa Maleyanda
Master of Science in Computer Science
San Diego State University, 2012

The thesis project was developed for Playdom Inc. (Disney Interactive Media Group) to assist in the cost calculation associated with the company resources, mainly the computer servers used by the various teams in the company for their respective games. The process was regularly a manual process, involving multiple spreadsheets and a single employee who manually calculated the costs and created reports that could be sent to the accounting team for processing. This resulted in a lot of time consumption which could otherwise be used for more productive work.

The proposed application tool was intended to simplify the complex method of manual calculation and develop a tool which automatically generated reports of the server details instead. It also allows active user interaction so that any updates could be easily made by the authorized users. The tool creates and maintains a database that is a structured compilation of data from the various spreadsheets available, by using a CSV (comma separated value) file which on submit would retrieve the data from the back end and insert in onto the UI page depending on the initiated query. The database is required to contain all of the data that is necessary to gauge the status of a given project/game at any given instant. The data available to view, add or update is defined according to the role of the user involved. There are customized tools available on the application to list contents of a particular table, add or delete values to name a few.

The tool is currently live and is actively being used by the Studio Engineering team of Playdom Inc. at the Palo Alto site to maintain server details.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT</th>
<th>LIST OF FIGURES</th>
<th>ACKNOWLEDGEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>viii</td>
<td>ix</td>
</tr>
</tbody>
</table>

## CHAPTER

1 INTRODUCTION ................................. 1
   1.1 Overview .................................. 1
   1.2 About the Application ................. 2

2 SPECIFICATIONS .............................. 4
   2.1 Proposed Solution ....................... 4
   2.2 Dependencies .............................. 5
   2.3 First Prototype ......................... 5

3 TECHNOLOGIES ............................... 6
   3.1 Model View Controller (MVC) .......... 6
      3.1.1 Component Interactions .......... 6
      3.1.2 Use of MVC in Developing Web Applications .......... 7
   3.2 PHP ........................................ 8
   3.3 Eclipse IDE ............................... 9
   3.4 HTML ..................................... 9
   3.5 CakePHP .................................. 10
   3.6 SVN ...................................... 11
   3.7 phpMyAdmin .............................. 12

4 SOFTWARE PROTOTYPE ....................... 14

5 ARCHITECTURAL DESIGN ..................... 19

6 SETTING UP DEVELOPER’S ENVIRONMENT 22
   6.1 Downloading CakePHP ................. 22
   6.2 Permissions .............................. 22
   6.3 Setup .................................... 24
6.4 Development .................................................................24

7  ILLUSTRATION OF MVC FRAMEWORK USING CAKEPHP ..............26

8  WORKING OF THE APPLICATION TOOL ...............................................28
  8.1 Importing a CSV File .................................................................28
  8.2 Using the Available Menu Items to Add a Studio .........................28
  8.3 Listing the Server Records Already in the Database .................29
  8.4 Listing the Contacts in the Database ..........................................29

9  CUSTOM ACTIONS AND THEIR SIGNIFICANCE ...............................32
  9.1 View Action ............................................................................32
  9.2 Edit Action .............................................................................32
  9.3 Delete Action ..........................................................................32

10 STRUCTURE AND FUNCTIONALITY OF THE APPLICATION TOOL ......36

11 LIVE STATUS OF THE PROJECT ......................................................40

12 CONCLUSION AND CHALLENGES ...............................................42

13 FUTURE ENHANCEMENTS ............................................................43

BIBLIOGRAPHY .................................................................................44
LIST OF FIGURES

Figure 3.1. MVC workflow........................................................................................................7
Figure 3.2. MVC usage in web applications.................................................................................8
Figure 3.3. Typical cake request................................................................................................11
Figure 3.4. Flowchart describing SVN workflow.........................................................................12
Figure 4.1. Working copy of CakePHP after installation............................................................15
Figure 4.2. First GUI screen of the application............................................................................16
Figure 4.3. Code snippet of the defined parsing rules.................................................................16
Figure 4.4. Code snippet of the parse function............................................................................17
Figure 4.5. Final report generated...............................................................................................18
Figure 5.1. Application tool architecture diagram.................................................................20
Figure 5.2. GUI with the functional menu items listed...............................................................21
Figure 6.1. Installing PDT Eclipse............................................................................................23
Figure 7.1. Basic MVC request.................................................................................................26
Figure 8.1. Initial screen to import a CSV file...........................................................................28
Figure 8.2. Adding a new server using the custom add menu item.........................................29
Figure 8.3. Displaying the server details available in the database.........................................30
Figure 8.4. Listing contacts from the database..........................................................................31
Figure 9.1. Custom actions defined..........................................................................................32
Figure 9.2. Custom view action...............................................................................................33
Figure 9.3. Custom edit action...............................................................................................34
Figure 9.4. Custom delete action............................................................................................35
Figure 10.1. Structure of the project in Eclipse IDE...............................................................36
Figure 10.2. Model folder in the application directory...........................................................37
Figure 10.3. Structure of view folder in the application directory........................................38
Figure 10.4. Structure of the controller branch in the project...............................................39
Figure 11.1. User login on the live status page......................................................................40
Figure 11.2. Sample page from the report on the live application page.................................41
ACKNOWLEDGEMENTS

The euphoria and satisfaction that is attached to the completion of a technical project would be incomplete without thanking the people responsible for this venture.

I would like to acknowledge Playdom, Inc. (Disney Interactive Media Group) and everyone who works here for providing me with an opportunity to put my knowledge and presentation skills to test and also for providing me with the necessary facilities and the right atmosphere to work in. I would also like to shout out a special note of thanks to Mr. Ed Shirey and Mr. Sonjaya Tandon for their constant guidance in completing this project. My heartfelt gratitude to Prof. Carl Eckberg, for being so patient and helpful, every step of the way. I am indebted to Prof. Root and Prof. Interlando for making time in their busy schedules to be part of my thesis committee.

Last, but certainly not the least, I would like to thank my parents for their blessings and for believing in me throughout the course of my Masters. I am also grateful to my brother and all my friends for their constant love and support.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Playdom is an online social network game developing company, widely popular on Facebook, Google+ and MySpace; it is currently the largest social game developer on MySpace and number 15 on Facebook. The company was founded in the San Francisco Bay Area by University of California, Berkeley graduates Ling Xiao and Chris Wang and Swarthmore College graduate Dan Yue. In 2009, the market for games played on social networking sites was valued at $300 million, consisting mostly of online sales of virtual goods. In May 2010, Playdom was recognized as one of the “2010 Hottest Silicon Valley Companies” by Lead411.

It is a wholly owned subsidiary of The Walt Disney Company as part of the Disney Interactive Media Group.

Playdom has over a total of 27 million players as of July 2011. The following are few of the famous games developed by Playdom:

- Disney’s Ghosts of Mistwood
- Disney’s Gnome Town
- Sorority Life
- Wild Ones
- Marvel Avengers
- Gardens of Time
- Mobsters

Off late, Playdom has also introduced their social games on the iOS platform and a few in the Android Market [1].

Playdom was acquired by Disney Interactive Media Group in 2010. Playdom produces a diverse portfolio of casual games for the rapidly growing platform of social networks, including Facebook and MySpace. With over 47 million monthly users and #1 rated games, Playdom is one of the fastest-growing social game developers, offering high
quality community entertainment experiences. Disney Interactive Media Group (DIMG), the interactive entertainment affiliate of The Walt Disney Company, creates immersive, connected, interactive experiences across console, online, mobile and social network platforms to entertain and inform audiences around the globe. DIMG’s mission is to deliver Disney content to fans, whenever and wherever they want it, through numerous interactive media platforms [2].

The Report Generator and Server Database application tool is developed using CakePHP which is an open source code distributed under the MIT License. The most important aspect of using this tool is that it allows for an organized maintenance of server details and cost allocation based on the CPU usage, which before this tool, was a long and difficult manual process of maintaining multiple excel spreadsheets. This manual process in turn resulted in a lot of approximation and lacked efficiency.

Chapter 2 summarizes the requirements the tool should meet, as put forward by the various users from the game studios. Chapter 3 discusses the technology used for developing this application. PHP, CakePHP and Html are explained in detail along with the Basic MVC (Model View Controller) framework and principles. Chapter 4 contains the software prototype and discusses the initial version of this tool. Chapter 5 discusses the architecture of the application and the main components of this application development. Chapter 6 summarizes how to set up the java environment by importing CakePHP. Chapter 7 explains the tools available in this application and their significance. Chapter 8 discusses the final working copy of the project, including some code snippets.

### 1.2 ABOUT THE APPLICATION

This project was developed as a tool to assist the various teams and game studios at Playdom with cost calculation and server maintenance. Playdom has a number of game studios and manually creating a spreadsheet every week to report to accounting caused a lot of confusion and the cost calculation ended up being very flaky as a result of all the approximation that manual calculation brings. In order to make the entire process effective, I proposed to develop a tool which would provide a way to create and maintain a database with Server Information, allocation, costs, CPU usage and the name of the person in charge of a
particular studio, which in turn would make the entire costing model less prone to manual errors and inefficiency.

Effectively a “studio” is a small development team within the company and the accounting concerns are with the quantity of use of various company resources, primarily computer servers by these numerous studios.

The author of this thesis is employed by Playdom Inc., and discovered the manual algorithm as a consequence of working at this Palo Alto site. My supervisor agreed that this project had merit and authorized me to pursue this software using company resources, but with the coding and design time not paid for by the company.
CHAPTER 2

SPECIFICATIONS

To create a service that allows a user to view and update the data available, depending on the role of the corresponding user without disrupting any other services. The service should also have a log of the changes that may have been made to the database.

- The service should present a unified view of all the servers and the corresponding details grouped under the respective datacenter groups.
- We should be able to create or edit filter rules for sorting server charges to various studios.
- We should be able to provide an audit phase where studio engineering (or assigned team(s)) could audit the automated filtering/grouping process to include manual adjustments.
- The Accounting (or assigned team(s)) should be able to view and generate reports which could be customizable.
- The Ops (or assigned team(s)) should be able to download server lists and feed it into their system as input files. Rules to parse these input files should be flexible and customizable so that our datacenters are allowed to change their output format without requiring too much input from the engineering end.

The intended audience is the various game studios and the people associated with them.

2.1 PROPOSED SOLUTION

- To generate a report that is a compilation of data from the various spreadsheets available. The report should contain all the data that is required by the finance department necessary for accounting purposes. The UI could contain simple html forms which would return the data retrieved from the back end and inserted to the page depending on the query that was initiated by the user on the html form.
- Also, the accounting team could be provided with access to the database directly with a query database instead of an automated report so they could customize it as per their needs.
- Engineering Managers should receive automated reports on a timely basis.
2.2 Dependencies

- Finance team
- Tech-Ops team
- Studio Engineering

2.3 First Prototype

- Creating a single spreadsheet with a unified UI to edit/view data.
- The service allows the user to view data based on the data center involved (Softlayer or NAP7) and the filtered regex. The filters could be customized as per the requirement.
- There should be a log of the changes that have been applied to the database and by whom.
CHAPTER 3

TECHNOLOGIES

This chapter introduces the technologies and platform used to develop the application tool. It also describes why these respective technologies were considered to develop this project and their advantages. Platform independence and easy deployment were the top two reasons why PHP was chosen to develop most of the application tool. The other technologies used are CakePHP, HTML and JavaScript. These are in turn explained in detail later. A majority of the concept of this project revolves around the MVC Design Pattern of software engineering. I begin the description of technologies used with a brief introduction to MVC Development first, as it is essential and is the backbone of this entire project:

3.1 MODEL VIEW CONTROLLER (MVC)

Model View Controller (MVC) is a computer software design pattern that separates the representation of information from the user’s interaction with it. The model consists of application data and business rules, and the controller mediates input, converting it to commands for the model or view. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a pie chart for management and a tabular view for accountants. The central idea behind MVC is code reusability and separation of concerns.

3.1.1 Component Interactions

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them as follows:

A controller can send commands to its associated view to change the view’s presentation of the model (for example, by scrolling through a document). It can send commands to the model to update the model’s state (e.g. editing a document).

A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these
notifications, because the application does not require them or the software platform does not support them.

A view requests from the model the information that it needs to generate an output representation.

Diagrammatic illustration of a MVC design pattern is shown in Figure 3.1 [3].

![Diagram of MVC workflow](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#Use_in_web_applications)

**Figure 3.1. MVC workflow. Source:**
Wikipedia. MVC in Web Applications, 2008

### 3.1.2 Use of MVC in Developing Web Applications

Model View Controller has been adapted as architecture for World Wide Web applications. Several commercial and noncommercial application frameworks have been created to enforce the design. These frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server. See Figure 3.2 [4].

Early web MVC frameworks such as Java EE took a thin client approach that placed view and controller logic almost entirely on the server. In this approach, the client sends either hyperlink requests or form input to the controller and then receives a complete and
updated web page (or other document) from the view; the model exists entirely on the server. As client technologies have matured, frameworks such as CakePHP, Zend, CodeIgniter, Symfony and Agile Toolkit have been created that allow the MVC components to execute partly on the client [3].

3.2 PHP

PHP is a general-purpose server-side scripting language originally designed for Web development to produce dynamic Web pages. It is one of the first developed server-side scripting languages to be embedded into an HTML source document, rather than calling an external file to process data. Ultimately, the code is interpreted by a Web server with a PHP processor module which generates the resulting Web page. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most Web servers and also as a standalone shell on almost every operating system and platform free of charge. A competitor to Microsoft’s Active Server Pages (ASP) server-side script engine and similar languages, PHP is installed on more than 20 million Web sites and 1 million Web servers.
Rasmus Lerdorf originally created PHP in 1995. The main implementation of PHP is now produced by The PHP Group and serves as the formal reference to the PHP language. PHP is free software released under the PHP License, which is incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP. While PHP originally stood for “Personal Home Page”, it is now said to stand for “PHP: Hypertext Preprocessor” [5].

### 3.3 Eclipse IDE

Eclipse is a multi-language software development environment comprising and integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java, and by means of various plug-ins, other programming languages like Ada, C++, C, Perl, PHP, Python. It can also be used to develop applications for the software Mathematica. Development environment include the Eclipse Java Development tools (JDT) for Java, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others. Eclipse provides views, perspective and editors. There is a debug perspective which includes break points which in turn make debugging comparatively easier. This thesis project is developed using Eclipse IDE with the PDT plug-in. It provides a flexible development environment for developers by indicating auto-suggestions and bug fixes [6] Netbeans allows for the same interactive features like in Eclipse, such as auto-generating and auto-completion, along with providing a fairly nicer UI which makes it extremely user-friendly. But Eclipse was used in this project because of the familiarity with it and for the fact that Eclipse execution is much faster. Also, Eclipse has a wide variety of plug-ins which makes it more convenient to use for various development environments. It starts up faster and reacts faster than a simple Netbeans execution which in turn makes Eclipse comparatively efficient [6].

### 3.4 HTML

HTML, Hypertext Markup Language is the basic foundation in developing web pages. The HTML provided tags are used by web browser to interpret the statements written inside HTML tags. This is a way of writing a structured document with heading, text style, paragraph, lists. HTML5 which is the latest release allows including images and videos.
These days the internet is evolving so rapidly that there is a lot of competition for creating interactive web pages. It helps to organize the web pages and its components.

There are HTML editors that will help to write HTML lines faster by providing all the tags available and suggesting while you are writing. CSS is a cascading style sheets which works together with HTML. It is used to work on the presentation of a page and apply styles to it. Initially all the styling tags were included with in HTML code, but CSS provides a separate styling file which helps for better readability and brevity.

3.5 CakePHP

CakePHP is an open source web application framework. It is written in PHP, modeled after the concepts of Ruby on Rails, and distributed under the MIT License. CakePHP uses well-known software engineering concepts and software design patterns such as Convention over configuration, Model View Controller, Active Record, Association Data Mapping and Front Controller [7].

CakePHP has several features that make it extremely adaptive and useful in developing interactive web applications. See Figure 3.3 [8]. Some of them are as follows:

- Flexible Licensing
- Compatibility with PHP4 and PHP5
- Integrated CRUD for database interaction and simplified queries
- Application Scaffolding
- Model View Controller (MVC) Architecture
- Request dispatcher with good looking, custom URLs
- Built-in Validation and Fast and flexible templating (PHP syntax, with helpers)
- View Helpers for AJAX, JavaScript and HTML Forms
- Security, Session, and Request Handling Components
- Flexible access control lists and Data Sanitization
- Flexible View Caching
- Works from any web site subdirectory, with little to no Apache configuration involved.
3.6 SVN

Apache Subversion (SVN) is a software versioning and revision control system distributed under an open source license. Developers use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation. SVN File system comprises of the 2 main items:

- Path
- Revision

Each revision in a Subversion filesystem has its own root, which is used to access contents at that revision. Files are stored as links to the most recent change; thus a Subversion repository is quite compact. The system consumes storage space proportional to the number of changes made, not to the number of revisions. The Subversion filesystem uses transactions to keep changes atomic. A transaction operates on a specified revision of the filesystem, not necessarily the latest. The transaction has its own root, on which changes are
made. It is then either committed and becomes the latest revision, or is aborted. The transaction is actually an object; a client does not need to commit or abort a transaction itself, rather it can also begin a transaction, exit, and then can re-open the transaction and continue using it. Multiple clients can access the same transaction and work together on an atomic change, though no existing clients expose this capability [9]. See Figure 3.4 [10].

![Flowchart describing SVN workflow](http://www.istruttoredigitale.com/lo-strumento-indispensabile-per-i-programmatori/subversion_workflow/, accessed October 2012.)

**Figure 3.4. Flowchart describing SVN workflow. Source:** Luca Grillo. Instructor Digital 2.0, 2011.

### 3.7 phpMyAdmin

phpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a Web browser. It can perform various tasks such
as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions.

The various features of phpMyAdmin that make it favorable to work with especially with regard to this project are [11]:

- Web Interface
- MySQL database management
- Data can be imported from SQL
- Data can be exported to various formats: CSV, SQL, XML, PDF
- Administering multiple servers
- Creating PDF graphics of the database layout
- Creating complex queries using QBE (Query-By-Example)
- Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link
CHAPTER 4

SOFTWARE PROTOTYPE

Software prototype is a necessary component in a software development process. It is basically a sequence of draft versions of the final outcome of the application. A prototype in the development process serves multiple purposes:

• It drives the development process in a direction and gives a perspective into the challenges ahead.
• It invites feedback from both supervisors and customers.
• Software prototype serves as an initial model which is refined along the development process to meet the requirements.
• The precursor to the development of software prototype is to gather requirements and specifications.
• Prototyping is considered an important component of agile software development.

The initial prototype for this application was developed as shown in the following figures.

The first milestone was to install and get a working copy of CakePHP to show on the screen and this is as illustrated in Figure 4.1.

The next step in the initial prototype was to display an index page for the application which facilitated a user to import any file (preferably CSV) to be processed and submitted as defined by the parsing rules (see Figure 4.2).

The first screen of the application went on to remain the same as the initial prototype after passing a variety of prototypes so that the UI was simple to follow and execute for any first time users as well.

The parsing rules were defined as part of the initial prototype and this included rules defined by regular expressions which primarily parsed the CSV file based on the name of the server. A code snippet of the defined patterns and the method used to parse are as shown in Figure 4.3.
The methods used to define the classification of the servers based on the regular expressions so that they are inserted into the right tables in the database are shown in Figure 4.4.

Initial report generated: The initial prototype also had a report generated on submitting the CSV file as input. It contained a tabulated representation of the data that is available in the database for the user. This data could further be used for billing and cost calculation by the various game studios and the finance department (see Figure 4.5).
Figure 4.2. First GUI screen of the application.

Figure 4.3. Code snippet of the defined parsing rules.
Figure 4.4. Code snippet of the parse function.
Figure 4.5. Final report generated.
CHAPTER 5
ARCHITECTURAL DESIGN

This chapter defines and explains the architecture details of this application. The architecture shows the software components involved in this application, and the interaction between the components.

Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [12]. It serves as a blue print indicating what tasks and development have to be done. In other words it is an early analysis of the end product and designing structural models for it. Figure 5.1 shows the architecture diagram for the tool implemented as part of this thesis project.

The architecture of the application is explained as follows:

- The CSV file is imported on the first page of the application.
- The UI has three parts, the left panel has the menu items where the various options of the available functions are listed, the right panel contains the actual functionality or displays the output of the function selected and the lower half of the screen displays the query executed when the respective function is selected.
- The user interface in this application is developed using CakePHP. Menu items could be added easily once the layout of the form was decided. See Figure 5.2.
Figure 5.1. Application tool architecture diagram.
Figure 5.2. GUI with the functional menu items listed.
CHAPTER 6

SETTING UP DEVELOPER’S ENVIRONMENT

Eclipse IDE is one of the easiest environments for a developer to work in. As stated previously, this project was developed using Eclipse environment. Download and install eclipse (see Figure 6.1) and add the PDT plug-in as follows [12]:

1. Install Eclipse from the official Eclipse website:
2. Install PDT SDK with the Help -> Install New Software menu item.
   
Next, install CakePHP. The requirements to install a working CakePHP developing environment are:

- HTTP Server
- PHP 5.2.8 or greater
- Database Engine (MySQL)

The following steps facilitate easy download and installation of CakePHP.

6.1 DOWNLOADING CAKEPHP

There are two ways to get CakePHP. We could either download an archive copy (zip/tar.gz/tar.bz2) from the main website, or check out the code from the GIT repository. To download the latest major release of CakePHP, visit the main website http://www.cakephp.org and follow the “Download Now” link. All current releases of CakePHP are hosted on Github. Github houses both CakePHP itself as well as many other plugins for CakePHP. The CakePHP releases are available at Github downloads. Alternatively one can get fresh off the press code, with all the bug-fixes and up to the minute enhancements. These can be accessed from Github by cloning the Github repository: git://github.com/cakephp/cakephp.git [13].

6.2 PERMISSIONS

CakePHP uses the app/tmp directory for a number of different operations. Model descriptions, cached views, and session information are just a few examples.
Figure 6.1. Installing PDT Eclipse.
As such, make sure the directory app/tmp and all its subdirectories in your cake installation are writable by the web server user.

### 6.3 Setup

Setting up CakePHP can be as simple as slapping it in your web server’s document root, or as complex and flexible as you wish. This section will cover the three main installation types for CakePHP: development, production, and advanced.

- **Development**: easy to get going, URLs for the application include the CakePHP installation directory name, and less secure.
- **Production**: Requires the ability to configure the web server’s document root, clean URLs, very secure.
- **Advanced**: With some configuration, allows you to place key CakePHP directories in different parts of the filesystem, possibly sharing a single CakePHP core library folder amongst many CakePHP applications.

### 6.4 Development

A development installation is the fastest method to setup Cake. This example will help you install a CakePHP application and make it available at http://www.example.com/cake_2_0/. The document root is usually set to /var/www/html.

Unpack the contents of the Cake archive into /var/www/html. You now have a folder in your document root named after the release you have downloaded (e.g. cake_2.0.0). Rename this folder to cake_2_0.

The development setup will look like this on the file system:

```
/var/www/html/
 cake_2_0/
  app/
  lib/
  plugins/
  vendors/
  .htaccess
  Index.php
  README
```

If the web server is configured correctly, the Cake application is accessible at: http://www.example.com/cake_2_0/.

To configure the Cake installation, we have to make some changes to the following files:
There are 3 constants that we need to edit: ROOT, APP_DIR and CAKE_COREINCLUDE_PATH:

- ROOT should be set to the path of the directory that contains your app folder.
- APP_DIR should be set to the name of your app folder.
- CAKE_CORE_INCLUDE_PATH should be set to the path of your CakePHP libraries folder.
CHAPTER 7

ILLUSTRATION OF MVC FRAMEWORK USING CAKEPHP

Figure 7.1 [14] shows a bare-bones MVC request in CakePHP. In order to illustrate this, I will use an example. Assume a client named “Jane Doe” clicked on a “Buy a dress!” link on the application’s home page.

- Jane Doe clicks the link pointing to http://www.example.com/dresses/buy, and her browser makes a request to your web server.
- The dispatcher checks the request URL (/dresses/buy), and hands the request to the correct controller.
- The controller performs application specific logic. For example, it may check to see if Jane Doe has actually logged in or similar form validations.
- The controller also uses models to gain access to the application’s data.

Models usually represent database tables, but they could also represent LDAP entries, RSS feeds, or files on the system. In this example, the controller uses a model to fetch Jane Doe’s last purchases from the database.
• Once the controller has worked on the data, it hands it to a view. The view takes this data and gets it ready for presentation to the client.

• Views in CakePHP are usually in HTML format, but a view could just as easily be a PDF, XML document, or JSON object depending on client requirements.

• Once the view has used the data from the controller to build a fully rendered view, the content of that view is returned to Jane Doe’s browser.
CHAPTER 8

WORKING OF THE APPLICATION TOOL

8.1 IMPORTING A CSV FILE

The index page of the application tool allows the user to import the invoice that is used to generate the final report (see Figure 8.1).

![Figure 8.1. Initial screen to import a CSV file.]

8.2 USING THE AVAILABLE MENU ITEMS TO ADD A STUDIO

The menu item Add Studio from the main menu could be directly used to add any new server or a new contact according to the requirement at the time (see Figure 8.2).
8.3 LISTING THE SERVER RECORDS ALREADY IN THE DATABASE

The List Studios menu item displays a list of the available server records in the database along with the date and time of creation (see Figure 8.3).

8.4 LISTING THE CONTACTS IN THE DATABASE

The contact list is a feature used to display the contacts in the database (see Figure 8.4).
Figure 8.3. Displaying the server details available in the database.
Figure 8.4. Listing contacts from the database.
CHAPTER 9

CUSTOM ACTIONS AND THEIR SIGNIFICANCE

Custom actions are defined in the project to facilitate easy updates and editing to the database directly via the application tool instead of having to add/edit or update from command line or SQL queries. See Figure 9.1.

9.1 VIEW ACTION

As the name suggests view action displays the entire details about the particular item selected. For example, if one clicks on the view button in a Contacts Table (see Figure 9.2).

9.2 EDIT ACTION

As the name suggests edit action allows a user to edit already available. For example, if one clicks on the edit button in a Contacts Table (see Figure 9.3).

9.3 DELETE ACTION

As the name suggests delete action allows a user to delete a column from the database after displaying a confirmation message about the deletion (see Figure 9.4).
Figure 9.2. Custom view action.
Figure 9.3. Custom edit action.
**Figure 9.4. Custom delete action.**
CHAPTER 10

STRUCTURE AND FUNCTIONALITY OF THE APPLICATION TOOL

As previously stated in the previous chapters, the application is structured on the MVC framework development using CakePHP. The structure of the tool in Eclipse is illustrated in Figures 10.1-10.4.

Figure 10.1.
Structure of the project in Eclipse IDE.
Figure 10.2. Model folder in the application directory.
Figure 10.3. Structure of view folder in the application directory.
Figure 10.4. Structure of the controller branch in the project.
CHAPTER 11

LIVE STATUS OF THE PROJECT

Figures 11.1 and 11.2 show a sample page from the current report that is live. Figure 11.1 contains a tabulated representation of data including the server name, CPU memory usage and actions that could be applied on the respective rows.

Figure 11.1. User login on the live status page.
Figure 11.2. Sample page from the report on the live application page.
CHAPTER 12

CONCLUSION AND CHALLENGES

This thesis was proposed and presented as an interactive application tool which facilitates a well working tool to provide the necessary information for the company when required. The main purpose was also maintain an updated version of the servers being used in the database with the most up to date information, which in turn would provide the most accurate information necessary for cost calculation and server allocation.

The main challenges I faced while developing this tool is that it was my first ever individual professional project out of school and it was challenging to single-handedly collect all the necessary information and talk to the managers of different studios in order to collect all the details that they required and to keep updating the requirements presented by various users. It was a challenge to be able to try and satisfy all the user requirements while keeping in mind the complexity and resources that I could handle on my own. Also, a major part of my research time was spent studying and trying to figure out the functionality and robustness of CakePHP since it was new to me and I had never used it to develop code before. So it took me a while to structure and write code so that the application was efficient in execution.
CHAPTER 13

FUTURE ENHANCEMENTS

The application tool was initially developed to have a record of the updated versions of the servers being utilized by the various game studios. As I approached completion, I had a few more ideas in order to enhance user experience on this tool:

1. One could facilitate a method to email the updated database records timely to the respective team managers, so they had a working copy of the data available every time there was an update.

2. One could develop a script to record a log of any changes made by users, so there would be no chance for any data to be misrepresented.

3. There could be more CSS changes implemented for a better looking GUI.


