

Helsinki Metropolia University of Applied Sciences  
Degree Programme in Information Technology

**Thanh Phong Le**  
**Developing a Social Network**  
**Using the Symfony Framework**

Final Year Project. 24 February 2009  
Supervisor: Peeter Kitsnik, PhD, Senior Lecturer  
Language advisor: Taru Sotavalta, Senior Lecturer

Author Title	Thanh Phong Le Developing a Social Network Using the Symfony Framework
Number of Pages Date	62 23 February 2009
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor Supervisor	Peeter Kitsnik, PhD, Senior Lecturer
<p>The social network or the next generation of Web 2.0 technology has been one of the popular topics discussed almost everywhere in the world in recent years. Being different from the old Web model, Web 2.0 lets people interact more effectively on the Internet by building a network of people. The objective of this project was to build such a network to comic fans.</p> <p>This project used the popular technologies on the Web platform, such as PHP programming language, MySQL database system and Ajax interaction model. Furthermore, the Symfony framework, was utilized for quick and efficient development of Web applications.</p> <p>The result of this project was a website system where people can share their own comics, join in the network of comic fans and enjoy the thousands of comics books. The project was also the core project work and a product from MahShelf Ltd Inc., which is the creator of social comic website (<a href="http://www.mahshelf.com">http://www.mahshelf.com</a>)</p> <p>This project did not only contribute to the business success of the MahShelf, but also created a place for comic fans, which was still missing from the Internet.</p>	
Keywords	Social network, Web 2.0, PHP, MySQL, Symfony framework

## Contents

1	Introduction .....	6
2	Introduction to the Internet .....	8
2.1	History and Overview .....	8
2.2	Basic Concepts of World Wide Web.....	9
2.3	Hypertext Markup Language .....	11
2.4	Web 2.0.....	12
3	PHP and MySQL .....	14
3.1	PHP History and Overview.....	14
3.2	Main Features of the PHP .....	14
3.3	MySQL History and Overview .....	16
3.4	Typical Features of MySQL .....	16
4	Application Development Model .....	18
4.1	Model-View-Controller Model .....	18
4.2	Object Relational Mapping .....	19
5	Social Networks.....	20
5.1	Overview and History.....	20
5.2	Basic Concepts .....	22
6	Symfony Framework .....	23
6.1	Overview and History.....	23
6.2	Platform Architecture .....	23
6.3	Controller Layer .....	25
6.3.1	Front Controller .....	26
6.3.2	Actions .....	27

6.3.3	User Session .....	29
6.4	View Layer.....	30
6.5	Model Layer.....	32
6.5.1	Database Schema Creation.....	33
6.5.2	Table and Model Classes .....	35
6.5.3	Data Access and Criteria.....	36
6.6	Basics of Page Creation.....	37
6.7	Summary .....	38
7	Application Development .....	39
7.1	Overview.....	39
7.2	User Account Management .....	40
7.2.1	Overview.....	40
7.2.2	Implementation.....	40
7.2.3	Database Schema Design.....	42
7.2.4	Application Demo .....	44
7.3	Short Messaging System.....	45
7.3.1	Overview.....	45
7.3.2	Database Schema.....	45
7.3.3	Creating Friend Invitation.....	47
7.3.4	Creating a New Book Notification.....	48
7.3.5	Application Demo .....	49
7.4	Friends System.....	50
7.4.1	Overview .....	50
7.4.2	Database Schema.....	50
7.4.3	Creating a Friend Request.....	52
7.4.4	Accepting Friend Request.....	53
7.5	Profile Commenting System .....	55
7.5.1	Overview.....	55
7.5.2	Creating New Profile Comment.....	56
7.5.3	Removing Profile Comment .....	56
7.5.4	Application Demo .....	57

8	Conclusion.....	60
	References.....	61

## 1 Introduction

In recent years, the social network was undoubtedly the popular topic that has been addressed in almost every place on the Internet. It has raised a new wave in the community of Internet users in the first years of the twenty-first century. Different from the old model and static Web pages, whose primary purpose is to share information while users have no real way to interact with the information, the social network has made a significant change to the way people interact with information and with each other.

In the old Web model, all data on the website is maintained and updated periodically by the site administrator. The result is that the content is very static and usually poor if without maintenance. The social network has come with another approach to address this issue. The idea is to let users build and maintain the information instead of the site administrator. Users know better a particular detail in their real lives. They can come up with more accurate and helpful information. In a social network, where people interact with each other and share information, all information which is inappropriate or not correct is filtered out by the community. Thus, this type of model will give more reliable and dynamic information on the websites.

The Internet and the World Wide Web were given birth about fifteen years ago. It was a significant time for developing and changing, both in the aspect of model and technology. Nowadays, there are many software technologies available for Web development, all different in the power, ease of use and the level of abstraction, for example, JSP (Java Server Page), ASP (Active Server Page), Perl, Python, Ruby and PHP. In the environment of Web development, PHP is a good candidate that provides fast and easy development.

PHP is a server side scripting language which was primarily designed as a language for Web development. Nowadays, PHP is used by millions of websites and become one of the most popular platforms for the Web. This demonstrated the practical application PHP. Along with MySQL, it is an efficient database tool and one of the most popular database systems that have been used from personal to corporate websites. However, it

is still not sufficient to build a website with rich features and yet, with high performance, scalability and security, and with fast development time and low cost. At this level, a powerful and solid framework is a must to necessity to meet this requirement.

At the time of writing this thesis, there are many frameworks that suit well for developing powerful websites. However, all these use a very similar approach. Among those, Symfony has not existed for a long time but has proved its power and practical use. Symfony is a PHP5 MVC framework for rapid development of Web applications.

The aim of this thesis is to study the different aspects of a social network: What it is and how to apply it for a real application. Also the Symfony framework, how to use it in a Web application development will be studied. Finally, a goal is to use Symfony to build a social network for comics fan. The scope of this thesis is not limited with the user interface of the application but rather focuses on the implementation in the backend.

The project is one of core works for MahShelf Inc, which is the creator of the social comic website at <http://www.mahshelf.com>. There are several features implemented at the MahShelf website. This thesis tries to focus on the social network features.

## **2 Introduction to the Internet**

### **2.1 History and Overview**

The beginning of the Internet dates from the early years of the 1960s in the United States, when there was a big request for sharing information and research papers among the universities for the military. Soon later, in late 1962, the project was moved on to Advanced Research Projects Agency (ARPA) with the aim to develop the computer communication network further. Hence, at that time, the Internet was also known as the ARPANET. In 1969 the very first ARPANET which initially connected four major computers at the universities in the southwestern US was brought online. Nowadays the Internet has been developed and expanded to almost any place and computer all over the world, making the biggest global network. [3,490]

The Internet can be simply stated as the interconnection of networks from all around the world. This type of model is like the expansion from a single network connected together to form a bigger network. There are various computers connected to each other in a network. When growing further, each network acts as a single computer connected together with other ones, forming a bigger network. When all the networks in the world were connected together, the Internet was born to allow the possibility of a connection of arbitrary computers in the world. [3,490]



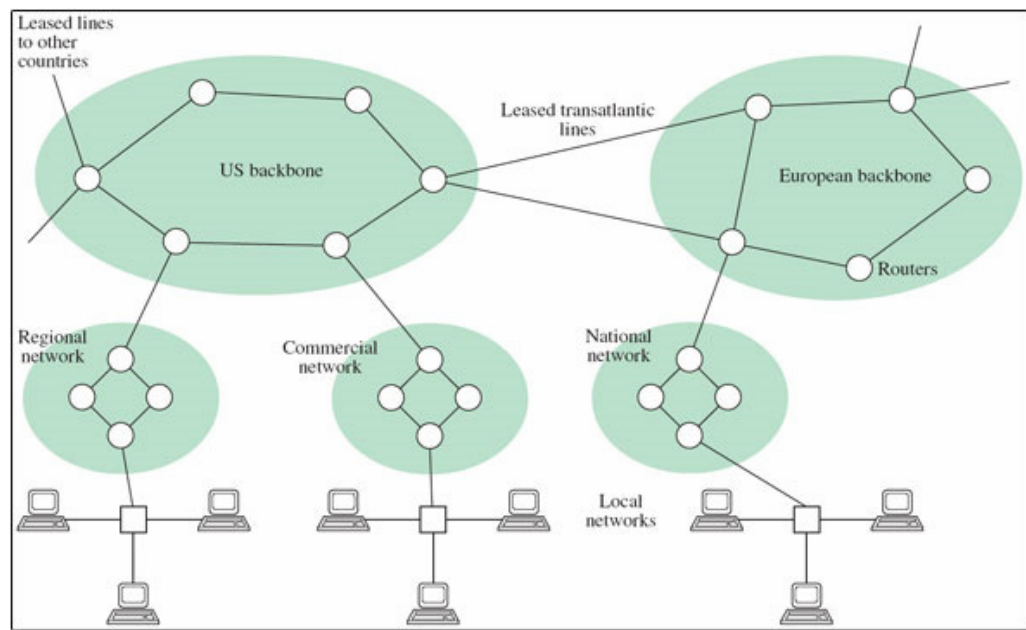


Figure 1: The topology of the Internet [4]

Figure 1 illustrates the topology of the Internet, group of computers forms a smaller network and each smaller network is connected together to form the bigger network. The computers are not connected directly but through many intermediate computers and networks. When a computer in this network is connecting and exchanging data with other computers in another network, they are actually sending out and receiving packets. A packet is a trunk of data used for exchanging information on the Internet. The packets are not exchanged directly between two computers. Instead, they are routed through many intermediate computers before being able to reach the destination. As a computer is connected to the Internet, its main task is also to route the packets to the destinations.

## 2.2 Basic Concepts of World Wide Web

World Wide Web (WWW) was invented in 1989 by an English scientist Tim Berners-Lee working for the European Organization for Nuclear Research (CERN). WWW is a system of the interconnection of hypertext documents. A hypertext document is a document that contains links to other documents. Each document can contain the normal media such as the text, image, audio, video and finally the hyperlink. The hypertext document is written in the standard of the **Hypertext Markup Language**

(HTML). A hyperlink is like a connection to another page and is allocated by a unique address, named **Uniform Resource Locator (URL)** [6, 5]. Simply stated, each document is analogous to a node in a network. Each node is connected or can be linked to other nodes by the hyperlinks defined in the hypertext document.

Each computer on Internet is identified by a special number, called the **IP address**. The IP address consists of four numbers from 0 to 255 and separated by dot separator (.). To avoid the address duplication, each computer can be assigned to only one address and the address is unique throughout the network. [9]

Uniform Resource Locator (URL) is another form of addressing computers on the Internet. It tends to help memorizing the address by using readable and meaningful words. Actually, a URL is a translated form of an IP address which is maintained by the **Domain Name System (DNS)**, a distributed database stored on the servers throughout the world. Whenever a URL is used to address a specific server, the equivalent IP address is looked up through the local or nearest DNS servers or even to the ones in the upper levels. The IP is used internally in all the operations of the Internet working mechanism. In other words, the IP is a form of address used implicitly in the underlying Internet operations. [5, 196]

A URL address is composed of three parts, the schema, the authority and the path, as shown in the figure 2 below

Schema	Authority	Path
<a href="http://Example.org/mysite/page">http</a>	:// Example.org	/mysite/page

Figure 2: An example of an URL address

The schema defines the protocols for communication between the user computers. The common one is the Hyper Text Transfer Protocol (HTTP) which is used to retrieve the WWW resource. [5,197]

The authority has the form of *userinfo@host:port* in which the *userinfo* part is not commonly used in general cases. The *host* and the *port* part indicate the host name and the port number respectively. On the Web platform, the port 80 is used for exchanging the Hyper Text Markup Language (HTML) documents using Hyper Text Transfer Protocol (HTTP). [5,197]

The path points to the corresponding file on the server, following exactly the hierarchy of the file system. This is a common case on most servers. However, it is totally dependent on and can be customized by the developers. [5,197]

### 2.3 Hypertext Markup Language

Hyper Text Markup Language has an abbreviation as HTML, which is the language for composing and creating hyper text documents. In the computer world, the programming language is an essential means for creating computer applications. The language defines the syntax and semantics which are human-readable, so that it provides a better tool to write software. After that, the compiler provided with the language will translate the code into the machine code which can be understood only by the computer. Different from the general computer language, the markup language is not translated into a machine-readable code. Instead, it provides the syntax to format the presentation of the document and/or contains the hyper links to other documents. [10,11]

Even though HTML is a simple language that can be understood by people, it is still a plain text document. In order to visualize the document, one must render by special software, called a browser. Nowadays browsers, such as Firefox, Safari and Internet Explorer, are popular and can render almost all HTML documents quite well. In the Internet revolution, HTML plays an important role in defining the standard rules for creating hyper text documents. Due to its simplicity and ease of use, it is widely adopted as a fundamental language in the world of Web programming. [10,12]

HTML is a simple language for making different formats (markup) of the document presentation. To do that, it provides the concept of tag. A tag consists of two parts, the

starting tag and the ending tag. The starting tag is placed at the beginning of the text and the ending tag is placed at the end. All the text wrapped around by a start tag and an end tag is formatted according to the tag, as illustrated by the figure 3. [10,11]

`<b>Internet</b>` is great => **Internet** is great

Figure 3: An example of a HTML document and the corresponding output

Table 1 presents a list of the basic tags frequently used in HTML documents and their functions.

Table 1: The basic HTML tags [10,236]

Tag	Function
<code>&lt;b&gt;...&lt;/b&gt;</code>	Bold
<code>&lt;i&gt;...&lt;/i&gt;</code>	Italics
<code>&lt;u&gt;...&lt;/u&gt;</code>	Underline
<code>&lt;strong&gt;...&lt;/strong&gt;</code>	Strong
<code>&lt;strike&gt;...&lt;/strike&gt;</code>	Strike through the text
<code>&lt;h1&gt;, &lt;h2&gt;, &lt;h3&gt;, &lt;h4&gt;....&lt;/h1&gt;, &lt;/h2&gt;, &lt;/h3&gt;, &lt;/h4&gt;</code>	Heading 1, 2, 3, 4
<code>&lt;img&gt;...&lt;/img&gt;</code>	Image tag

In the beginning, HTML contained only a few tags for formatting but after years of revision and upgrades, there are now many technologies coming to replace or refine the existing one, such as XML and XHTML.

## 2.4 Web 2.0

Web 2.0 is the next generation of World Wide Web (WWW) which is aimed to improve the effectiveness of website and user interaction and offers a rich client user interface and saves bandwidth. One of the backbone technologies in Web 2.0 is Asynchronous JavaScript and XML (AJAX). Ajax allows loading and updating only a part of the website without refreshing and loading the whole website as in traditional model of Web 1.0 [7, 263]

Client/server is a well-known architecture widely used on the Internet. This model defines one computer acting as a server and the others acting as clients. The server's main task is to process the client's requests. After finishing processing, the server returns the result back to the clients for rendering. Nowadays, this model is not only used on Internet but also applied in broad categories of applications such as remote processing, database queries, terminal processing etc.

In the Internet model, the server is a computer storing and publishing the information and clients are the ones requesting for information. However, this model is not restricted to only information request as such as HTTP (Hyper Text Transfer Protocol) but extends to various services such as FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol) and database queries. These services have their own protocols built on top of the underlying Internet protocols. [11]

## 3 PHP and MySQL

### 3.1 PHP History and Overview

The PHP is a general-purpose, server side scripting language mainly used for developing a dynamic website that allows user interactions. PHP is mostly deployed on the server and its main task is to handle and process the requests from the client side. As a scripting language, the program requires an interpreter for executing. Since the interpreter is able to support any available platforms or machines if possible, that makes PHP as well as a cross-platform language. Finally, PHP is free and open source, meaning that people can get the source code and make the significant change to it. [11]

The very first PHP script was created in 1995 by Rasmus Lerdorf with the purpose of keeping track of his online resume. In the beginning, the script was merely a set of Perl scripts, and Lerdorf named it **Personal Home Page Tools**. However, he found the limitation of the script and wanted to extend it with more features. That made him write the script implementation in C language but with more functionality, such as connecting to the database and let users create a simple but dynamic web page. Finally, he decided to release the source code so that everyone could read, modify and improve it. [11]

### 3.2 Main Features of the PHP

The PHP is a simple language that can be studied and understood easily. It has several features which are designed to best fit the environment of Internet and Web development. Such features are the following:

- **Loose data type:** There is no clear data type for a variable in the PHP. A variable can contain a numeric value, a string or both without an explicit declaration. The system automatically detects and treats the variable according to what it contains. Thus, a variable is a number if it contains a number, a string if it contains text. This is the key feature that made the PHP popular in the Web development environment. On the Web, transmitted data is usually a mixing of both numbers and characters, so it is difficult to differentiate the data types.

Instead, the PHP treated all data types as one to ease the process of data handling and manipulating.

- **Database handling:** In the PHP, database handling including connecting and querying is easy due to the simplicity of the database API. Furthermore, PHP can work with various database management systems.
- **Environment variables:** The Web environment has several special variables such as request, response, session and cookie. In PHP, these are called environment variables and can be directly accessed. This feature saves developers a lot of time since these pieces of data are required on almost every web page.
- **Fast string processing:** PHP offers fairly strong functions and features that help in processing strings. A string can be easily accessed, cut, trimmed or copied by directly calling the appropriate functions.
- **Flexible array handling:** PHP provides flexible ways to handle the array. The array can have one or multi-dimensions and the values can be scalar or associative. An associative array is an array where each element is a pair of key and value, in which the value can be a scalar but it also can be another array.
- **All features of OOP with version 5:** Since version 4, Object oriented programming (OOP) has been supported in PHP. However, the support has not been strong and comprehensive. The syntax was confused and there was a lack of important features. However, in version 5, PHP has improved significantly and has better support for OOP. Most features of OOP such as polymorphism, encapsulation and inheritance, are all supported and the syntax has been improved for development and readability.

The current version of PHP now is 5 and the version 6 is up-coming. PHP 6 even tries to become the programming language for desktop applications development. That

expresses the fast growth of PHP since it was born in 1995. PHP is practically a language designed for the development of applications on the Web platform. Nowadays, PHP is used by millions of websites, supported by almost every hosting service and used by hundreds of thousands of developers. [12]

### 3.3 MySQL History and Overview

Computer applications are usually involved with data. There are always strong needs in storing, retrieving and manipulating data. However, raw data are not effective when used in critical and even trivial applications. That was the reason database was invented. Database is data but with a solid structure and can be used effectively in most applications. MySQL is that type of database. However, more than that, MySQL is also a database management system (DBMS) that helps in manipulating database more efficiently. [14]

MySQL is free and open-source software that people can use and improve it if needed. These are the characteristics made DBMS one of the most popular ones nowadays. MySQL uses the SQL, which is a language used for retrieving and manipulating data in the database. [14]

The history of MySQL dates back to 1995 by two Swedes David Anmark, Allan Larsson and a Finn Michael Monty Widenius who intended to use the database system named **mSQL** to connect to their tables using their own fast low-level routines (ISAM). However, after that, they found out that the performance of mSQL did not satisfy their requirements. It was not fast and flexible enough. They came to the decision to write a new implementation using the same interface as mSQL but faster and more powerful. The final result is the MySQL database system. [14]

### 3.4 Typical Features of MySQL

MySQL is written in C and C++ for fast and better performance. It is a fully multi-threaded system using the kernel threads and can easily support multiple CPUs if



available. MySQL can be deployed as a server in the client/server environment or embedded to standalone applications. These applications can be used in isolation and in an environment where there is no network at all. The server or the backend can work on multiple platforms. In the front end, there are many programs, libraries and administrative tools available. MySQL also supports a wide range of language interfaces, such as Java, .NET, and PHP. Besides, there are many other useful features:

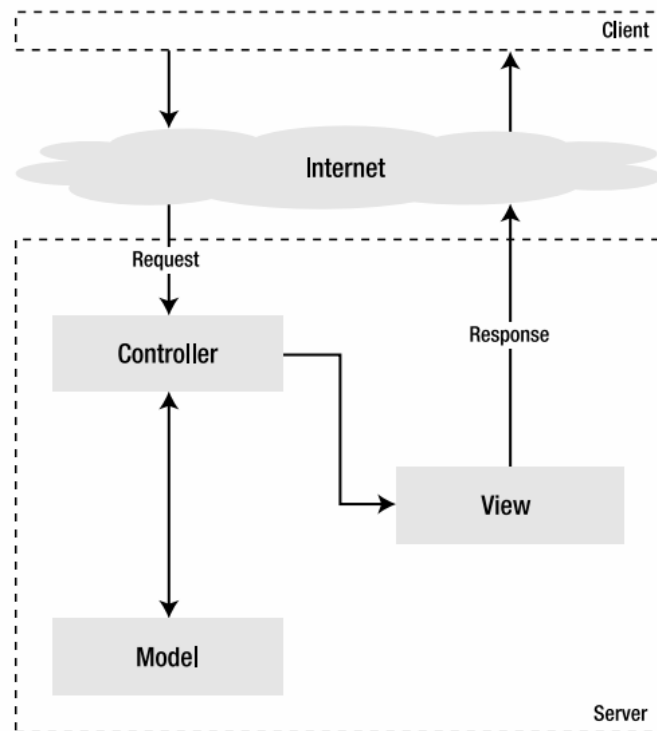
- **Data types:** MySQL has a comprehensive and very flexible data types ranging from the numeric values to string based data. Data types include integer value, float value, date, time, text, characters, set and enumerator.
- **Strong support for SQL:** Most requirements of SQL statements are all supported in MySQL. Furthermore, the implementation of these statements was even optimized to give better performance.
- **Scalability and limits:** MySQL can handle large database in which the number of records can be up to five billion. The number of indexes per table is up to 64 and each index may consist of 1-16 columns or parts of columns.
- **Connectivity:** MySQL offers various types of connectivity that is efficiently used in practice, such as TCP/IP socket on any platform. On the Windows platform, it uses the named pipes or shared connections. On a Unix platform, it can use the Unix domain socket files.

Generally, MySQL is a good solution for database storing and manipulation. Besides, it is an open-source software and can be used for free in any application. Currently, MySQL is used in many websites and as a database solution for most open source projects. [15]

## 4 Application Development Model

### 4.1 Model-View-Controller Model

MVC stands for Model-View-Controller which is a classic yet common design pattern in software engineering. The MVC model separates the software application architecture into three different layers called model, view and controller. The model represents the data which the application is processing. The view is for the presentation or user interface and the controller contains the application logic. The purpose of this model is to make all the layers independent of each other, in order to increase the system's flexibility in the matter of modifications. For example, the application can be easier to port to different platforms or views (i.e. a mobile or web version) by changing only the view layer while the other layers are kept unchanged. Vice versa, if there is no change in the controller, such as the internal data structure or algorithm, there is nothing to change in the view. [16]



**Figure 4:** The MVC pattern for Web application [1, 45]

Figure 4 demonstrates a MVC pattern used in Web development. The request is handled by the **controller**. The controller processes the request, updates the **view** and/or stores data to **database** accordingly. The view then outputs the response back to the client.

## 4.2 Object Relational Mapping

**Object Relational Mapping (ORM)** is a technique for translating a relational entity to an object-oriented entity. It maps the table to a class and a record to an object. Thus, instead of accessing the database directly, now it is possible to access the mapped objects. The benefit of this model is hiding the complexity of SQL statements and the underlying database system. The SQL statements are slightly different for different database systems. Hence, changing the database type requires changing the involved SQL statements. Besides, the developers can access the database without the SQL knowledge. Now, all the data manipulations are carried out by manipulating the associated objects. [8]

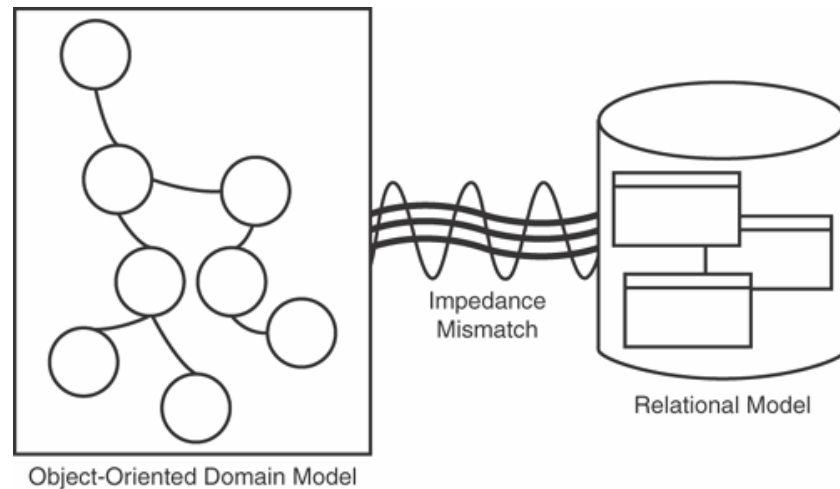


Figure 5: An illustration of the ORM model [8]

Figure 5 illustrates the translation from the records in the database tables to discrete objects. All objects contain the fields matched with the fields in the table.

## 5 Social Networks

### 5.1 Overview and History

Debuted in late 2002, the social network has made a great movement and somehow a revolution on the Internet since it has changed the way people use the Internet. Earlier, the Internet or Web contained only static information maintained by site administrators. At that time, there were no real connections at all on those websites. The term **connection** refers to a real connection in the real life, such as friends or other people who share the same interest in a specific topic. Now users are able to publish their own information on the website, making their own public profiles. Furthermore, they can make friends with people on the Internet that share the same interest and friends in real life.

Nowadays, there are thousands of social networks dedicated to a particular thing or merely social networks for your friends or colleagues. There are social networks for books, for gamers or for videos .etc... that first are built as the sharing service and then enhanced by the social network model, which aims to attract more people to the website.

The first website that can be recognizable for a social network is **SixDegree.com** launched in 1997. The website offers users abilities to create their own public profiles and open and browse the friends list. However, there are many websites that provide the same feature but there is no one that combines all these features. SixDegree.com is the first one that combines these basic features that make up the very first and basic social network. [2]

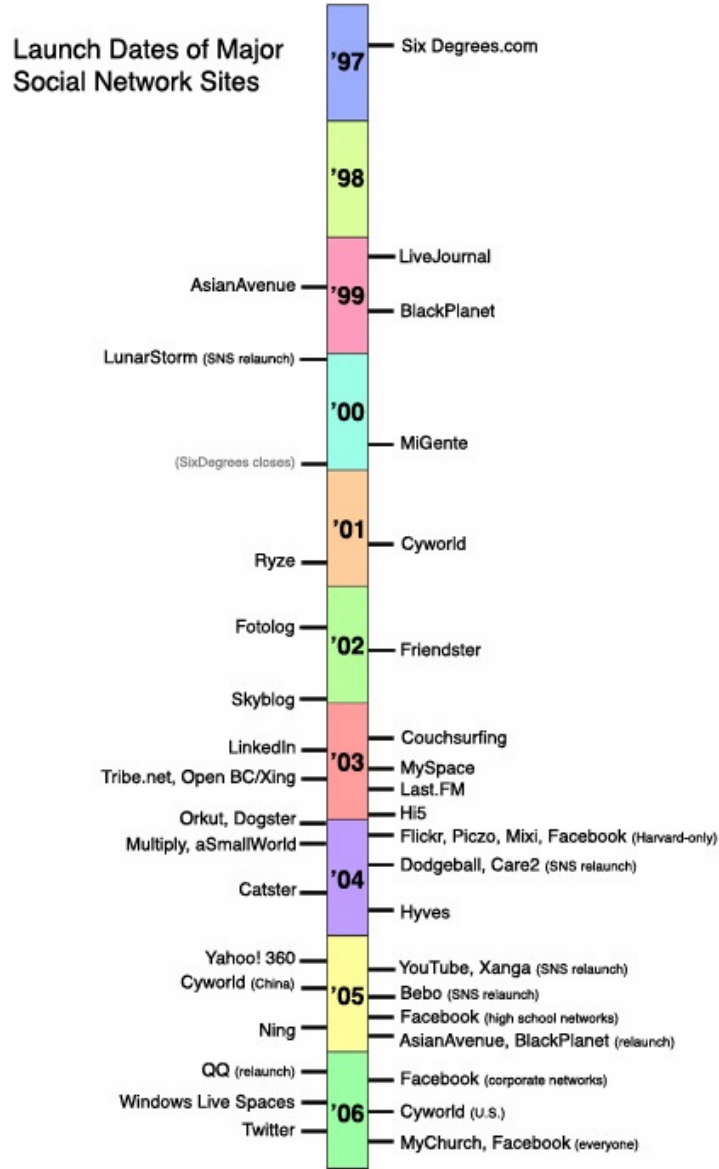


Figure 6: Timeline of the launch dates of many major social networks [2]

Figure 6 presents the timeline of the major social networks since 1997. There are websites that first come up as a sharing service, such as **Youtube.com**, the most popular video sharing service, but were later re-launched with the social network features support.

## 5.2 Basic Concepts

A website which is called a social network must have the following features. User can publish his/her own profile and user can open and browse the friends list from his/her profile. The profile can be public for everyone from outside the network or restricted to a certain network or to friends only. Depending on the website's objective, the profile can be visible to the ones that do not have the account on the website or it is restricted to users that have account. Furthermore, users can set their profile to be restricted to friends only. From a specific user profile, the visitor can see the list of this user's friends and be able to browse backward and forward this friends list. The term **browse** means that the visitor can open the profile and continue browsing the friends list.

In a social network, the user is required to make a connection with others. The term **connection** might have various meanings depending on the website context, such as friends or contacts. Hence, the connection is an abstract way to express the relationship of users in the real life. Each connection can be bi-directional or one-directional, depending on the characteristics and definition of the website. A **Friends** connection is usually bi-directional and a **Subscribers** connection is usually uni-directional.

The visibility of the user connections is another concern. Once again, it is dependent on the website, but commonly there are two different types. The public connections are visible to everyone and the restricted ones are visible to anyone who is permitted to view. The user's connections list is usually displayed in the user profile. However, there are still a few exceptional cases.

Besides the basic features of social networks, some websites have the feature that allows users to add comment to the profile or send private messages to a specific user. These features are so called profile comments and private messages respectively. Even though these features are very common on most websites, they are not considered the core features of a social network. Furthermore, some websites are also equipped with advanced features such as photo or video sharing, blogging, instant messaging and mobile interaction capability.

## **6 Symfony Framework**

### **6.1 Overview and History**

Symfony is the PHP 5 MVC framework designed for the rapid development of Web applications. The first version was released in October 2005. It is free to use and an open-source project that people can read and improve. Symfony provides the infrastructure for developing projects using MVC model with capabilities to use ORM for secure and fast database access functionalities. The MVC model is a common and efficiently proven design pattern in the software industry. The Symfony framework is written entirely in PHP 5 to utilize all the benefits of new improvements in performance and the new support for object-oriented programming. The current version of Symfony is 1.2 but in the scope of this thesis, version 1.0 is used. [1,7-9]

Symfony was invented in 2005 by Fabien Potencier, the CEO of a French web agency for innovative views on Web developments. In 2003, he spent some time to find an existing open source project as a development tool for Web applications, but none of them fulfilled the requirements. At the time, PHP 5 had just been released and he believed the new version of PHP was good enough for building a new framework for Web development. He spent a year to write the Symfony core in the MVC model. After successfully using it in Web application development, Potencier decided to release the framework as an open source project, so that people could contribute to the project by fixing bugs and/or improving it. Nowadays, Symfony has been used in many critical applications and reached its maturity as a framework for Web development. [1,5]

### **6.2 Platform Architecture**

The Symfony architecture is separated into two stacks, the framework and the platform. Both layers contain cohesive but decoupled classes. The classes in the framework layer are divided into different blocks, following the MVC model. In contrast, there is no dependency between the classes on the platform layer.

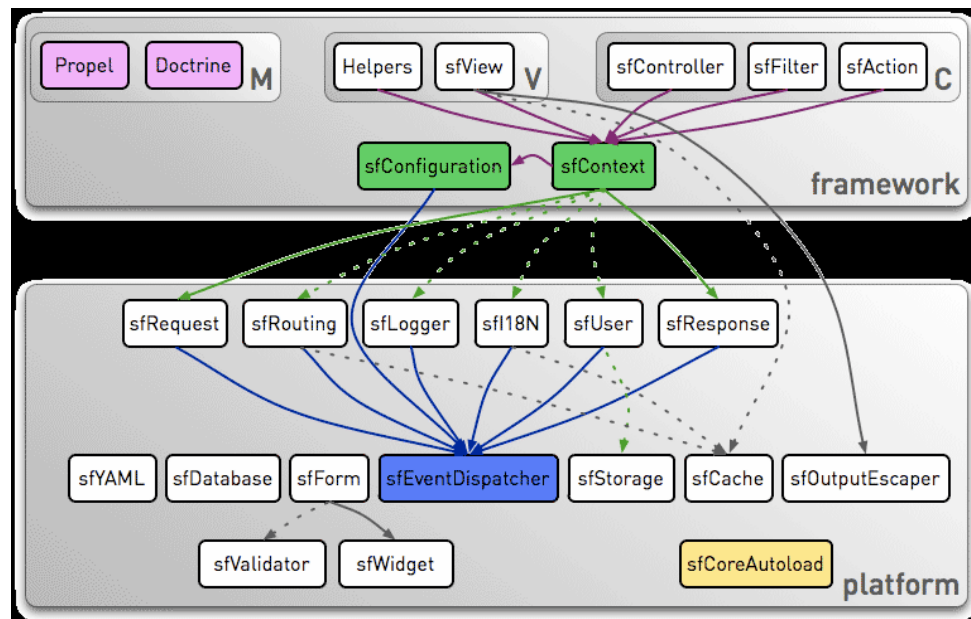


Figure 7: The Symfony architecture [17]

From the figure 7, all the usable classes in the upper layer belong to the framework, and are divided into blocks of Model, View and Controller in MVC model. These classes use the core classes in the lower layer which belong to the platform. [17]

Symfony follows the MVC model quite strictly since it is the core pattern in the framework. The developers do not have choice and must use the MVC model as default in the entire application. The purpose is also to keep the code in structure and quality. On the other hand, this will make consistency throughout the entire system.

Commonly, in a Web application, the data is usually stored in the database, mostly separated from the rest of the application. That somehow can be thought of as a model layer. The requirement is to separate the view and controller layer which is usually mixed together if not designed beforehand. This is not an easy task since the view and the logic usually interact with each other and it is somehow hard to distinguish these parts. That also makes developers confused with which part should appear as logic and which part should appear as view.



In order to ensure that the view and controller are separated, Symfony requires that the application or a module of an application must have the controller and view separated completely into different files. Even at the higher level, all the files belonging to the view part are saved into the **template** folder and all the files belonging to the controller are saved into the **action** folder. However, this does not help if the developers do not have in mind the clear distinction between the view and controller. A rule of thumb is that the view should only take care of rendering data and there should be nothing related to the data manipulation. The logic behind or the controller should only take care of the part of data manipulation. [1,22-24]

### 6.3 Controller Layer

Generally, the controller layer in an MVC model contains the code for processing the business logic of the application. In Symfony, this layer is even separated into more components to enhance the flexibility in writing the application. These components include:

- **Front controller**, the entry point to the application. Its task is to load the configuration and determine the action to execute
- **Actions**, defines the business logic of the application. It usually validates the inputs from the request and prepares data for the presentation (view) layer.
- **Environment variables**, the objects named request, response and sessions. They are to give access to the Web special environment variables such as request parameters, response headers and persistent user data.
- **Filters**, the special part executed before and after every request. An example of a filter is the security and validation check before each request to ensure the data are correct and valid. [1,83]

These features are handled automatically by Symfony. The developer's task is to define the clear action for the application logic.

### 6.3.1 Front Controller

Front controller is the unique entry point to the application in a predefined environment. When there is request to the system, the front controller first uses the routing system to find the module name and the action name with a particular URL. For an instance, the following URL executes the action name **myAction** of the module **mymodule**

**http://domainname/mymodule/myAction**

After the domain name, there is the module name and then the action name. All are separated by slash (/). The front controller's other jobs are handled automatically by Symfony. Here is the list of job details of front controller:

1. Define the core constants.
2. Locate the Symfony libraries.
3. Load and initiate the core framework classes.
4. Load the configuration.
5. Decode the request URL to determine the action to execute and the request parameters.
6. If the action does not exist, redirect to the 404 error action.
7. Activate filters (for instance, if the request needs authentication).
8. Execute the filters, first pass.
9. Execute the action and render the view.
10. Execute the filters, second pass.
11. Output the response.

Even though there are many tasks belong to the front controller but mostly they are handled automatically by Symfony framework. The developer main work is to define the template and the action behind it. [1,83-84]

### 6.3.2 Actions

Action is the heart of the application since it contains the application logic. All the actions are defined in the file named **actions.class.php**, stored in the folder **actions** of the module directory. Each action is defined by a method named **executeActionname** of the default class created by Symfony named **moduleNameActions**. The **moduleNameActions** class is inherited from the **sfActions** class. Listing 1 shows the snippet of code demonstrating how an action is defined.

```
class mymoduleActions extends sfActions
{
    public function executeIndex()
    {
        ...
    }
}
```

Listing 1: The sample code for defining an action

In the action method, there is access to the common Web parameters such as request, response or session, as shown in the listing 2

```
class mymoduleActions extends sfActions
{
    public function executeIndex()
    {
        // Retrieving request parameters
        $password    = $this->getRequestParameter('pa

        // Retrieving controller information
        $moduleName  = $this->getModuleName();
        $actionName  = $this->getActionName();

        // Retrieving framework core objects
        $request     = $this->getRequest();
        $userSession = $this->getUser();
        $response    = $this->getResponse();
        $controller  = $this->getController();
        $context     = $this->getContext();

        // Setting action variables to pass information to the
        template
        $this->setVar('foo', 'bar');
        $this->foo = 'bar';           // Shorter version
    }
}
```

Listing 2: The parameter accesses from the action method

As shown in the listing 2, all the common Web parameters can be accessible through the calls of Symfony framework methods.

### 6.3.3 User Session

A user session is handled automatically by Symfony, which is built on the PHP session mechanism. In the action method, the session data can be accessed by calling the method **getUser()**. The return value is the object of the class **sfUser**. This object contains a parameter holder that can be used to store or retrieve any attributes of the session. The data type of the attribute can be any (integer, string, array or even the object).

```
class mymoduleActions extends sfActions
{
    public function executeFirstPage()
    {
        $nickname = $this->getRequestParameter('nickname');
        // Store data in the user session
        $this->getUser()->setAttribute('nickname', $nickname);
    }
    public function executeSecondPage()
    {
        // Retrieve data from the user session with a default value
        $nickname = $this->getUser()->getAttribute('nickname', 'Anonymous Coward');
    }
}
```

Listing 3: An example of storing and retrieving an attribute in a session

Listing 3 shows an example of how to store and retrieve data from the attributes defined in the session. In PHP, to access these attributes, it can be a hard and tedious work but with Symfony, it can be done in easier way.

The persistent data stored in the session somehow expose the security vulnerability since it will keep data alive too long. There is an option requiring that the data stored is valid only in next action. That comes with the flash attribute, which is existing only in the next action. [1,96-97]

## 6.4 View Layer

As discussed in the MVC model section, the view layer takes care of rendering the data to the website. This can be thought of as the user interface of the application. In Symfony, the view layer is divided into smaller components to improve flexibility and scalability. These components are:

- **Layout and templates:** Similar to the fixed portion of the web page that usually does not change or changes somewhat from page to page. In contrast, the template contains the dynamic content and represents the data for each action.
- **Code fragment:** Portions of the same view but different in the data. Symfony introduces three types of fragments in the view: layer, partial, component and slot.
- **View configuration file:** Configuring the rendering of all aspects on the website. This includes the setting of output escaping, which is a feature to escape all the special characters of the output data for security purposes.

Each template file is corresponding to a specific action, named as **actionSuccess.php** and saved in the **templates** folder. The template file usually contain an HTML code mixed with an embedded PHP code. The embedded code usually contains only an **echo** statement with the variables defined from the action method. [1,113]

```
<h1>Welcome</h1>
<p>Welcome back, <?php echo $name ?>!</p>
<ul>What would you like to do?
    <li><?php echo link_to('Read the last articles',
'article/read') ?></li>
    <li><?php echo link_to('Start writing a new one',
'article/write') ?></li>
</ul>
```

Listing 4: A sample of a template file

The listing 4 demonstrates a sample of a template file, which contains a statement echo to display the variable **\$name**. Besides, the code contains a helper, for example such as **link\_to()** which is a replacement of a chunk of a HTML code for a specific task. There are some variables that can give accesses to the common needed information:

- **\$sf\_context**: The whole context object (instance of sfContext)
- **\$sf\_request**: The request object (instance of sfRequest)
- **\$sf\_params**: Parameters of the request
- **\$sf\_user**: The current user session object (instance of sfUser)

The page layout is defined in the file named **layout.php** in the **templates** folder of the application root folder. This is the default layout Symfony will use if no other layout is specified. Each layout has a name associated with it and can be customized for different actions by creating different layout files. The action method can choose to use the layout that matched with it.

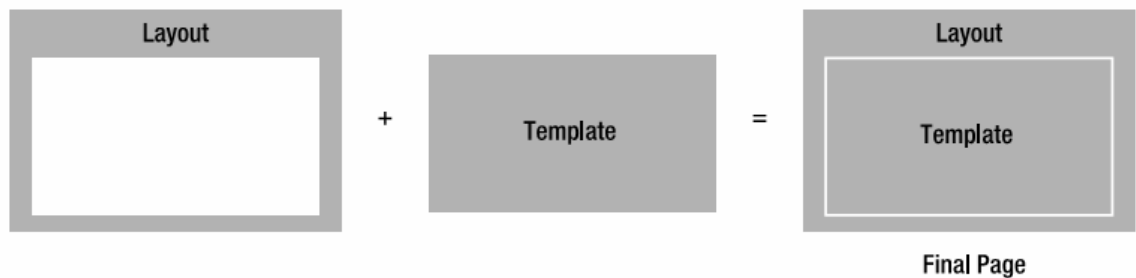


Figure 8: Decorating a template with a layout [1,117]

The template with a layout is deployed using the decorator design pattern. The layout contains the HTML code which is the same for every template. [1,117]

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
<head>
  <?php echo include_http_metas() ?>
  <?php echo include_metas() ?>
  <?php echo include_title() ?>
  <link rel="shortcut icon" href="/favicon.ico" />
</head>
<body>
<?php echo $sf_data->getRaw('sf_content') ?>
</body>
</html>

```

Listing 5: A sample of the layout file

Listing 5 shows an example of a layout file which contains the full structure of a HTML document. The page will be processed and parsed by PHP before sending to the user client browser.

## 6.5 Model Layer

In Symfony, the model layer is separated into two sub-layers: the data access layer and the database abstraction layer. The database abstraction layer makes the code independent of the database system so in the change of the underlying database system, there is no need to change the code in the other layer. This technique is implemented using object relational mapping (ORM) where the main idea is to map each record in a table to an object.



In Symfony version 1.0 uses Propel as the default ORM framework for interacting with the database. Propel provides a lightweight and simple API for storing and retrieving the data. Here are the typical features of the Propel framework:

- Underlying database system and SQL statements query are handled automatically.
- The data has escaping and type casting.
- Database is built by the Propel based schema files in YAML format.
- All model classes are built automatically matching to the schema file.

Propel provided fairly good features to help developers dealing with database. All these features aim to free developers from the complicated tasks and focus on the logic of the application. [1,141]

### 6.5.1 Database Schema Creation

The database is automatically created by defining the schema file which complies with the Propel standard. All tables can be created in a single schema file, but also can be divided into many different files with one table per file. All the schema files are stored under folder **config** in the Symfony root folder. Each schema file must be named with the prefix **\_schema** before the extension, for instance **users\_schema.yml**. All the schema files are stored using the standard of the YAML structure.

```

---
propel:
  feedback:
    id:
      user_id: { type: integer, foreignTable: sf_guard_user,
foreignReference: id, required: true, onDelete: cascade, index:
true }

    content: longvarchar
    email: varchar(128)
    website: varchar(128)
    name: varchar(128)
    created_at:

```

Listing 6: A sample of the feedback schema file

Listing 6 shows an example of the schema file for creating a table called **feedback**. The first part is to indicate that this is the schema file for Propel framework. In the next line is the name of the table, in this case, it is **feedback**. The following lines are the fields of the tables and each field is defined in one line.

Each line usually contains two parts that are separated by a colon (:). The first part is the name of the field such as ID, name or email and the second field is the data type and/or the attributes associated with it. Each database system has its own set of data types. Due to that, Propel does not use any data types set of a specific database system. Instead, it builds its own set of data such as an interface to different classes of databases. However, the data types defined by Propel are very similar to other database types such as MySQL.

The first part is compulsory for every line, but the second part is not. There are lines where the second part is missing. Those are recognized as the fields that will be automatically filled with the predefined types. For example, if any field name contains the string **id** at the end and the second part is omitted, then it is understood that the field is an integer, primary key and auto incremented. The other cases go to the field name that has the string **at** in the end. Those fields are automatically generated with the type

of **TIMESTAMP**. In other cases, the second part must be clearly declared the data type with all available attributes. If there is an attribute missing, it is assigned with a default value.

There are various attributes needed to be defined in the second part of each field. The typical one is to define the foreign key to a field of another table. It is declared in two keys **foreignTable** and **foreignReference**. The **foreignTable** specifies the table to which the key points to and the **foreignReference** refers to the field of the foreign table. Another attribute **index** has two possible values (true and false). This attribute sets the field to be indexed or not. If indexed, it improves the searching performance but in exchange it decreases the inserting operation. [1,143-144]

### 6.5.2 Table and Model Classes

When the schema file is completely defined and ready, the table and equivalent model classes are automatically created by calling the batch script packaged with Symfony.

In the Symfony root folder, a user has to enter the following command

```
$ php symfony propel-build-all
```

This script will detect all the file names that have the **\_schema** prefix and generate the equivalent tables and model class names. All the model classes are stored in the folder **lib/model/om**. For a specific table, there are always two model classes which are created automatically. For instance, if the table name is feedback, then two files

**BaseFeedback.class.php** and **BaseFeedbackPeer.php** will be created. These are called the basic model classes which contain all the basic functionalities to access all the pieces of data in that table. There are also two other classes created in the folder **lib/model** that have the same name as the ones created in **om** folder but without **Base** suffix. However, these files just contain a blank class which extends from the class in the base model classes. The purpose is to provide a way to extend and customize the existing base class. All these classes are automatically loaded into the main application and are ready to use in the action methods created afterward. [1,145-147]

### 6.5.3 Data Access and Criteria

As stated in the ORM model, the data access is done via the objects of model classes instead of interacting directly with the database. The first procedure is always to instantiate a new object from the equivalent model class. The operations such as retrieving, storing, updating and deleting can be executed using the class's methods getter and setter. These are the automatically generated methods for retrieving and updating the fields in the object. All the fields in a table will have all the equivalent getter and setter methods. [1,147]

A query criteria is a set of conditions to examine if a record is matching or not. The purpose is to retrieve a list of records that meets the specific requirement of users. A simple example can be a list of feedbacks that comes with a specific date. That specific date is called the criteria. Symfony provides a set of methods and classes for dealing with criteria in the database. The criteria is like a replacement of a WHERE clause of SQL language in the ORM model. In general use, criteria fulfils all the needs for retrieving data. [1,151-153]

Table 2: SQL and Criteria Object Syntax

SQL	Criteria
WHERE column = value	->add(column, value);
WHERE column <> value	->add(column, value, Criteria::NOT_EQUAL);
<b>Other Comparison Operators</b>	
> , <	Criteria::GREATER_THAN, Criteria::LESS_THAN
>=, <=	Criteria::GREATER_EQUAL, Criteria::LESS_EQUAL
IS NULL, IS NOT NULL	Criteria::ISNULL, Criteria::ISNOTNULL
LIKE, ILIKE	Criteria::LIKE, Criteria::ILIKE
IN, NOT IN	Criteria::IN, Criteria::NOT_IN
<b>Other SQL Keywords</b>	
ORDER BY column ASC	->addAscendingOrderByColumn(column);
ORDER BY column DESC	->addDescendingOrderByColumn(column);
LIMIT limit	->setLimit(limit)
OFFSET offset	->setOffset(offset)
FROM table1, table2 WHERE table1.col1 = table2.col2	->addJoin(col1, col2)
FROM table1 LEFT JOIN table2 ON table1.col1 = table2.col2	->addJoin(col1, col2, Criteria::LEFT_JOIN)
FROM table1 RIGHT JOIN table2 ON table1.col1 = table2.col2	->addJoin(col1, col2, Criteria::RIGHT_JOIN)

The table 2 shows the list of SQL queries with WHERE, LIMIT, OFFSET and ORDER statement and the equivalent Criteria methods. A Criteria object must be created before using these methods. Criteria is an object that defines all the criteria incorporated in the retrieval statement. This object will be passed as a parameter to the retrieval method of the model classes.

```
$c = new Criteria();
$c->add(FeedbackPeer::EMAIL, "abc@domain.com");
$results = FeedbackPeer::doSelect($c);
```

Listing 7: An example of retrieving the feedback that has email of [abc@domain.com](mailto:abc@domain.com)

Listing 7 demonstrates a sample of how to define a criteria and retrieve the records by that criteria. In this example, the purpose is to retrieve the feedbacks that have email address of [abc@domain.com](mailto:abc@domain.com)

## 6.6 Basics of Page Creation

Page creation in Symfony is not as simple as creating a single HTML file, linked to the PHP file in the backend. It requires that every page must belong to a specified module. A module can be thought of as a group of pages that share the same functionality. Each module is identified by its name and stored in a folder with its name under the **application/modules** folder. A module folder will have at least two subfolders which are actions and templates. The other optional folders are **config**, **lib** and **validate**. These folders are automatically generated when the module is created with the batch script.

```
> symfony init-module myapp mymodule
```

In the **actions** folder, the file named **actions.class.php** is automatically created and the content is the extension of the **sfAction** class. This file is required to customize different actions or pages in this module. For example, if the new action has the name XXX then in the **templates** folder, a template file with name the **xxxSuccess.php** must be created for this action. The page then can be accessed via the address

<http://domain.com/module/xxx> or if there is a route predefined for this specific action, the address will be different according to the settings in the routing system. [1,49-50]

## **6.7 Summary**

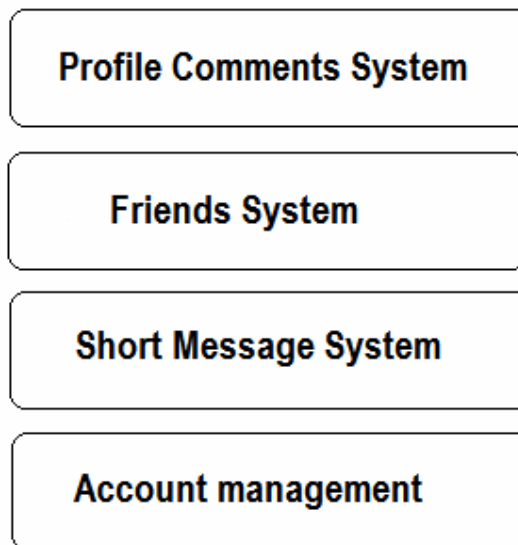
Symfony offers a comprehensive and robust framework for Web application development. The MVC model forces the developers to follow the strict yet effective design pattern to achieve the better quality code. Each layer is even separated further to improve the flexibility for modification. One of the key features is support of a rich set of helpers. Another one is the abstraction of database layer deployed by the Propel ORM framework. This not only hides the complexity of the underlying database system but also improves the scalability of the system. Symfony is a good framework for Web development. It significantly reduces development time and brings richer and more robust features to the application.

## 7 Application Development

### 7.1 Overview

The whole website <http://www.mahshelf.com> is a large system and contains several complex features that are out of the scope of this thesis. This thesis tries to focus on the implementation of the core features of a social network system. The core features include:

- User account management
- Facebook-like wall info system
- Friends system
- Profile commenting system



**Figure 9:** The Application Stack

The figure 10 demonstrates block diagram of the main features of the application. For each feature, this report will show the specification (requirement), the database schema, the block diagram of key components, the algorithm, the core implementation and finally a demo of user interface.

## 7.2 User Account Management

### 7.2.1 Overview

User account management is a basic feature in implementing a user-centric website. This requirement is very important since the system needs to know how the users are and what they are doing. Especially in a social network, the user account must be managed in a more effective and flexible way. A common solution is to distinguish each user by his/her username. This can be accomplished by the way of providing the username and the password. After logging on, every action will be recorded under the username. Furthermore, when user browses to other page, there is no need to log in again. [1,98-99]

The user account management in this project is very similar to the general one. However, there are just two more advanced features. The first one is the time limitation for each user's log on. If there is no action within 15 minutes then the account is forced to log off automatically. The second one is the automatic log in, which let user log in without the need to enter again the username and password. This feature is enabled from user by marking the **Remember me** check box in the login box.

### 7.2.2 Implementation

When the user opens the website, the system will automatically create a new user session. To distinguish each user, the session can set up to store two pieces of information as two attributes. The first one is the username and the second one is the authentication state. The authentication state can be true or false. The **true** value indicates that this user is logging in, and the **false** indicates that this user is a visitor. In order to set the authentication state, the user must pass the authentication phase that occurs at the login page. The attribute username must be set manually, but the authentication state attribute is automatically handled by Symfony. This is the special attribute of the class **sfBasicUserSecurity** which is an extended class from **sfUserSecurity**.



The login page requires the user to enter the username and the password. At the server side, the password is then encrypted by the MD5 hashing algorithm. The system will use the username and the encrypted password to look up in the **Users** table. If there is a match, then the user is entitled as authenticated. Otherwise, it is a fail authentication and the user is required to enter the username and password again. The figure 11 illustrates the block diagram of the user login process.

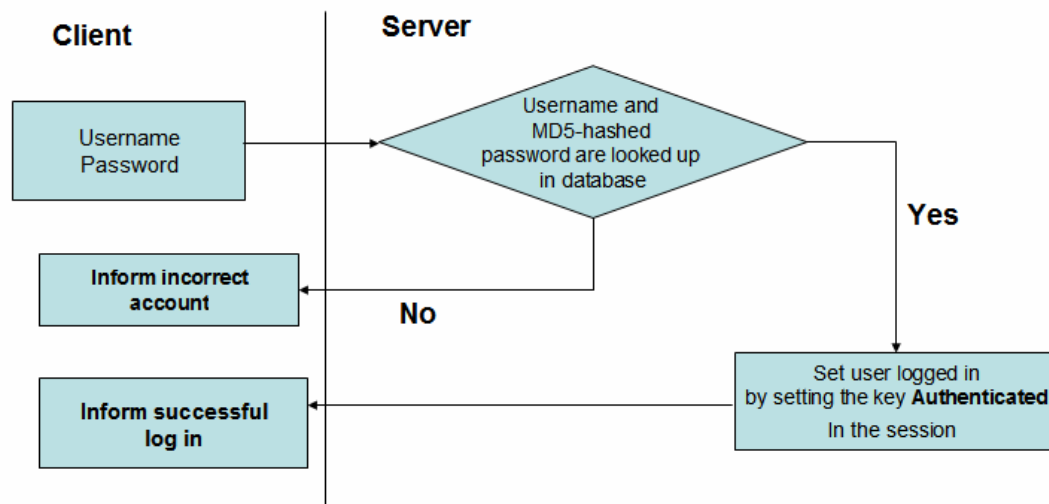


Figure 10: Block diagram of the user login process

The automatic login feature is implemented with the help of a special key, called **remember key**. This key is generated when the user logged in the first time and marked the remember check box option. This key is stored in two places, in the database of the server side and in the cookie of the client side. Whenever the user opens the website again, the system check if there is the cookie name **remember\_key**. If there is, this key's value will be read and at the server side, the key value will be looked up through the **Users** table if there is record contains this key value. If the match is found, the user is automatically logged in. However, due to security reasons, the key must be regenerated again and stored in both sides.

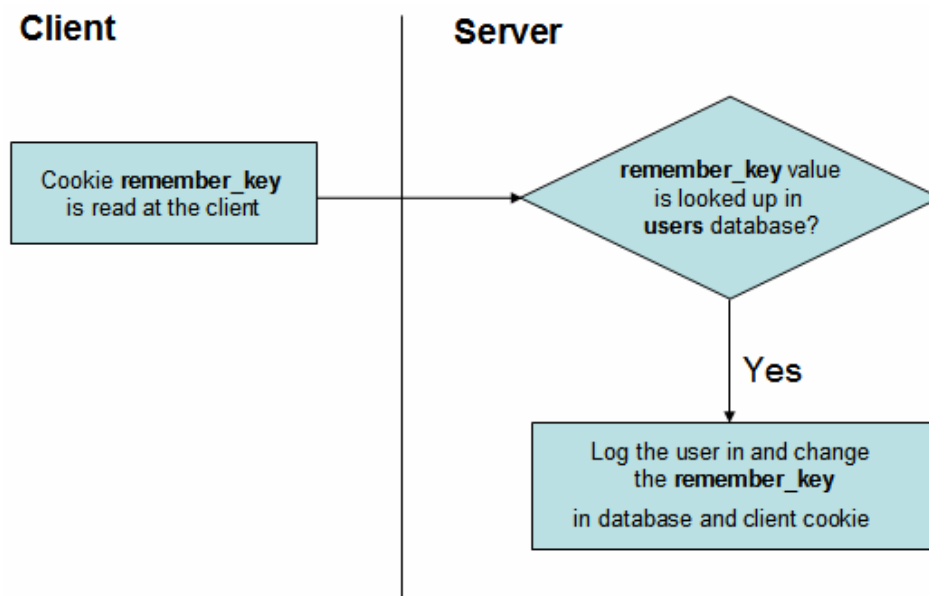


Figure 11: The block diagram of automatic login feature

Figure 12 demonstrates the block diagram of automatic login feature. The `remember_key` is read from the client and then looked up in the `Users` database. If the key can be found, the user is automatically logged in.

### 7.2.3 Database Schema Design

Listing 8 is the schema file for the `users` table. This table is to keep track each user's data.

```

users:
  id:      ~
  username: { type: varchar, size: 128, required: true, index: unique }
  password: { type: varchar, size: 32, required: true }
  created_at: ~
  last_login: { type: timestamp }
  is_active: { type: boolean, required: true, default: 1 }
  remember_key: { type: varchar, size: 32 }
  
```

Listing 8: The users table schema file

The fields have the following meanings:

- **id** is the primary key of the table.
- **username** and **password** respectively store the username and the encrypted password
- **created\_at** and **last\_login** are to keep track of user data, such as the account registration time and the last login time
- **is\_active** is to set if this account is active or not since there is a case this user can be banned due to violating the terms and conditions of the website.
- **remember\_key** is a special key stored to each user for auto login feature

The user authentication process is well implemented with the support of Symfony. Listing 9 shows the snippet of code for authentication process.

```
// Get the request parameters
$username = $this->getRequestParameter('username');
$password = $this->getRequestParameter('password');
// Hash the password
$hashedPassword = md5($password);
// Define the criteria and look up in the users database
$c = new Criteria();
$c->add(UsersPeer::USERNAME, $username);
$c->add(UsersPeer::PASSWORD, $hashedPassword);

If (UsersPeer::doSelectOne($c)) {
    $this->getUser()->setAttribute('username', $username);
    $this->getUser()->setAuthenticated(true);
}
```

Listing 9: The source code of user authentication and log in function

In this snippet of code, after retrieving the request parameters, the password is hashed by MD5 hashing function. Both values are looked up in the **users** database, if there is a match, the attributes **username** and **authenticated** will be set accordingly.

### 7.2.4 Application Demo

Figure 13 is the screenshot of the dialog box for the user to log in. There is the check box for marking the option to **remember me** on the computer.

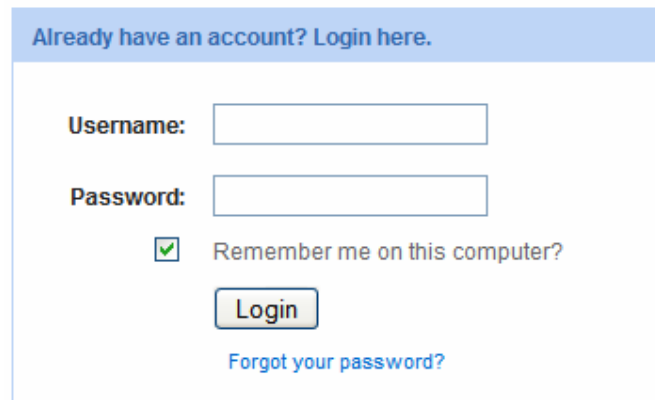


Figure 12: The account log in dialog box

After logging in successfully, the user is redirected to the home page. The status now is authenticated and the username is displayed in the header of the web page, as illustrated in the figure 14.

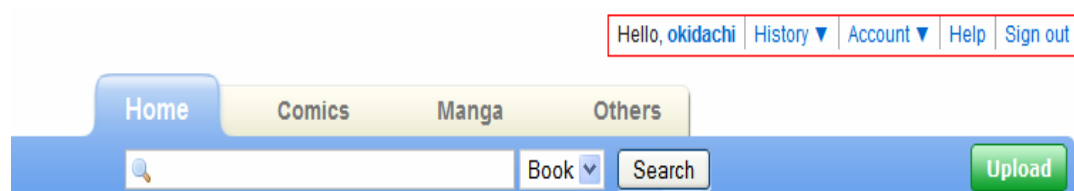


Figure 13: The status of the user after logging in

The login feature and the authentication process is very basic in users-centric website. If the process is not well designed, it can cause significant impact in the future feature and the system design.

## **7.3 Short Messaging System**

### **7.3.1 Overview**

The Short Messaging System (SMS) is a mechanism to deliver short messages from the system to users in an effective way. It informs users of the updates from their friends and from the system. This feature is very common and mostly a must in a social network. The features specifications are the following:

- When there is a new friend request, the system should inform the user about it.
- When the user accepts a friend request, the other end should be informed of it
- When the user uploads a new book, all his/her friends should be informed of it
- All users should be informed from the system if there is new message
- A user has the capability to remove a specific message

Since there might be many short messages delivered to users per day from the system and from other users. The component design should be lightweight and fast. All other main features of a social network will use this component as a mechanism to deliver the messages to users.

### **7.3.2 Database Schema**

Each message might have different types, such as announcements, notifications, friend requests, friend acceptance and new book upload. However, all the messages share the same attributes, which are who dispatched this message and who is the receiver.

Depending on the type, each message can have different parameters associated with it.

Listing 10 illustrates the schema file for the table storing short messages:

```

propel:
  flash_message:
    id:
      type: { type: smallint, default: 0 }
    created_at:
    viewed_at: { type: timestamp }
    params: varchar(512)
    receiver_id: { type: integer, default: 0 }

```

Listing 10: The schema file for a SMS table

The fields have the following meanings:

- **id**: the id of the message (primary key)
- **type**: the info message type
- **created\_at, viewed\_at**: to track the time the message was generated and the receiving time
- **params**: stores the parameters of the message
- **receiver\_id**: the user id of the receiver

The message type can have the following values:

**0 - SYSTEM**: The message is dispatched from the system with the aim to send it to all users on the website. The purpose is sending the messages from the site administrator to all users as news or announcements.

**1 – FRIEND**: This type of message is to inform a user about a friend request made by other users and wait for a response.

**2 – NEW BOOK**: This type is to inform a user that his/her friend has just uploaded a new book.

The **params** field may have different data depending on the message type. There might be many parameters included in one single string separated by a special separator (**&nbsp;**):

- **SYSTEM**: the **params** field is left blank since the content field usually has all the information for the message.
- **FRIEND REQUEST**: the **params** field contains two attributes, the username and user id of the requestor.
- **NEW BOOK** info: the **params** field contains three attributes, the username of the author, the book title and the book id.

The **receiver\_id** stores the user id of the receiver. The value 0 can be used as a special value indicating that the info is aimed to be sent to all users registered on the website, instead to that single user.

### 7.3.3 Creating Friend Invitation

Listing 11 illustrates the method used to create a message for a friend invitation request. The method takes the host user id, the host username and the friend user id as parameters.

```
public function createInvitation($hid, $username, $fid)
{
    $this->setType(self::INVITATION);
    $this->setReceiverId($fid);
    $params = $username . self::DELIT . $hid;
    $this->setParams($params);
}
```

Listing 11: The method for creating a new info for friend request

The method sets the type to **INVITATION** to indicate that this is the message for creating the friend invitation. Then it sets the friend id **\$fid** as the receiver id since that is the one who will receive the invitation request. The **params** field in this request contains two pieces of data, the username and the host user id. The host user is the one who made the invitation request.

#### 7.3.4 Creating a New Book Notification

The following method, shown in listing 12 is to create a message for a new book notification to a specific friend id. The method takes the author's username **\$username**, the book title **\$title**, the new book id **\$bookId** and the friend user id **\$fid**.

```
public function createNewBook($username, $title, $bookId, $fid)
{
    $this->setType(self::NEW_BOOK);
    $this->setReceiverId($fid);
    $params = $username . self::DELIT . $title.self::DELIT . $bookId;
    $this->setParams($params);
}
```

Listing 12: The method for creating info for a new book notification

The method first sets the type to **NEW\_BOOK** to indicate that this info is to notify users that there is a new book uploaded by one of their friends. The user's friend is set as the receiver. The **params** contains three pieces of data, the author username, the new book title and the new book id. This message is supposed to be sent to all friends of the author. Here is the snippet of code to send to all friends of the author.



```
// Send the flash message to this user's friends informing there is a new book
foreach (FriendlistsPeer::getFriendList($user->getId()) as $friend)
{
    $fm = new FlashMessage();
    $fm->createNewBook($user->getUsername(),    $book->getTitle(),    $book->
    getIdCode(), $friend->getFriendId());
    $fm->save();
}
```

Listing 13: The snippet of code to send info to all author's friends

First, all the friend IDs of the author are retrieved. To each friend, the code creates the info for new book notification with the corresponding author's username, book title and new book id.

### 7.3.5 Application Demo

The figure 15 illustrates a sample view of the user account page, **updates** section. The updates section displays all the latest info sent to the user.

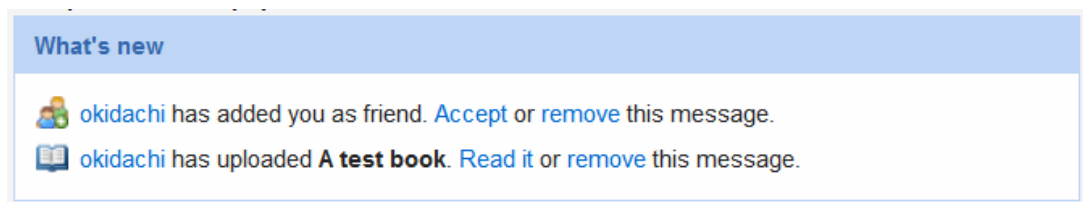


Figure 14: The updates section in the user account page

As shown in figure 15, the user received two different types of messages. The first one is the friend invitation request and the second one is the new book notification.

## 7.4 Friends System

### 7.4.1 Overview

The friends system is the core feature of a social network. A social network is a network of connected users on the Internet. The term **connected** refers to the relationship between users and the friends system letting users to establish such connections. The common features are:

- Adding a friend from the friends list of your friend
- Adding an arbitrary friend
- Accepting a friend invitation request
- Denying a friend request from other user
- Removing an existing friend connection

Since a friend connection is a two-directional connection that requires the acceptance from both sides, the user should be able to accept or deny a friend request from other users. Furthermore, the friend connection can be removed any time by any party.

### 7.4.2 Database Schema

The system distinguishes each user by user ID, and a connection with two ends can be thought of as a pair of two user IDs. Since a friend connection is a mutual connection, it requires storing made in both directions. On the other hand, the connection might have various statuses depending on the different contexts, for example waiting for the request response, or a connection is already accepted.

Listing 14 illustrates the schema file for the table storing a friend connection.

```
---
propel:
  friendlists:
    host_id: { type: integer, foreignTable: sf_guard_user, foreignReference: id, required:
true, onDelete: cascade, index: true }
    friend_id: { type: integer, foreignTable: sf_guard_user, foreignReference: id,
required: true, onDelete: cascade, index: true }
    status: { type: tinyint, default: 0 }
```

Listing 14: The schema file for friends list table

The fields have the following meanings:

- **host\_id**: the user id of the requestor
- **friend\_id**: the user id of the receiver
- **status**: connection status

The value of the status field can be:

- 1** – to indicate that the connection is an invitation (INVITATION)
- 2** – to indicate that the connection is already made (ACCEPTED)

As discussed in the social network section, friend connection is bi-directional and the database should store the connection from both sides. In other words, one user must be stored as host user and the other one is stored as a friend user. Vice versa, there must be another connection where the host user becomes the friend user and the friend now becomes the host user. Hence, there will always be another record with the values of **host\_id** and **friend\_id** swapped but the status field is kept unchanged. This is required since a friend connection must be from two sides, so when checking the friends list of a specific user, it can be done on only one field of the table, rather than on both fields.

### 7.4.3 Creating a Friend Request

The listing 15 demonstrates the method to create a friend request invitation. It takes two parameters. The first one is host user id **\$hid**, which is the id of the user who makes the invitation. The other one is the friend user id **\$fid**, which is the id of the user who receives the invitation.

```
public static function makeInvitation($hid, $fid)
{
    // Verify first if these users are not friends before
    $c = new Criteria();
    $c->add(FriendlistsPeer::HOST_ID, $hid);
    $c->add(FriendlistsPeer::FRIEND_ID, $fid);
    // If it does not exist, create the invitation
    if (!FriendlistsPeer::doSelectOne($c)) {
        $fcon = new Friendlists(); // friend connection
        $fcon->setHostId($hid);
        $fcon->setFriendId($fid);
        $fcon->setStatus(self::INVITATION);
        $fcon->save();

        return true; // indicate that invitation has been created ok
    } else return false; // indicate that the invitation has been created before
}
```

Listing 15: The method for creating a friend request invitation

The method first checks if the connection of host user id **\$hid** and friend user id **\$fid** exists. This is to ensure that there are no duplicated connections which can occur when the user has made multiple requests. If there is no such connection, a new connection will be created with the corresponding value, and the status will be set to **INVITATION** state.



**Figure 15:** A demo of making the friend request invitation

The figure 16 illustrates a demo of the action making friend request before and after. After clicking on the link **Add as friend**, the result message is returned to inform the current state of the request.

#### 7.4.4 Accepting Friend Request

After making the friend request, a response will be sent to the requester to inform the result. The receiver needs to accept the request to make the friend connection. Otherwise, the connection will be removed. The following method will demonstrate how to accept a friend request. The method takes the host user id **\$hid** which is the user id of the one who made the request. The friend user id **\$fid** is the user id of the one who will receive the request.

```

public static function acceptInvitation($hid, $fid)
{
    // create a new friend connection from host to friend
    $fcon = new Friendlists();
    $fcon->setHostId($hid);
    $fcon->setFriendId($fid);
    $fcon->setStatus(self::ACCEPTED);
    $fcon->save();

    // for reversed connection, we must set it to accepted.
    $c = new Criteria();
    $c->add(FriendlistsPeer::HOST_ID, $fid);
    $c->add(FriendlistsPeer::FRIEND_ID, $hid);
    $fcr = FriendlistsPeer::doSelectOne($c);
    $fcr->setStatus(self::ACCEPTED);
    $fcr->save();
}

```

Listing 16: The method for accepting the friend request

Listing 16 demonstrates the method for accepting the friend request. The method first sets the status of the connection of \$hid and \$fid to **ACCEPTED**, and then makes reverse connection as discussed in the friends system implementation. The reverse connection has the same status.

Figure 17 shows the screenshot of the friend request message received at the user account page.

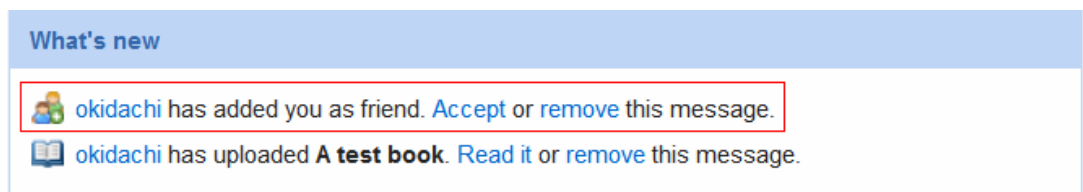


Figure 16: Friend request message

The change after the user clicked on the **Accept** link, as shown in figure 18

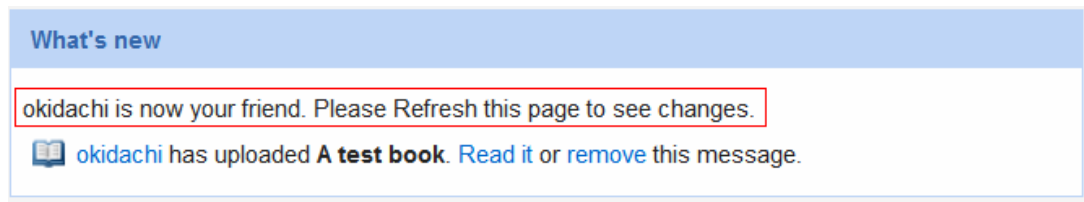


Figure 17: Friend acceptance message

An Ajax call is made to the server to accept the friend request. The result message is returned and displayed right away to the users.

## 7.5 Profile Commenting System

### 7.5.1 Overview

Profile commenting is a feature allowing users to put comments on a specific profile. This feature is to improve the communications between users. The features specifications include:

- User is able to put comment on other user profiles
- User is able to remove the comment posted by other users

The profile comments data stored in the separate table with all information of who posted it and which profile this comment has been posted to. Listing 17 demonstrates the schema file for the table storing the profile comments:

```
propel:
  profile_comment:
    id:
      profile_id: { type: integer, foreignTable: sf_guard_user_profile, foreignReference: id,
required: true, onDelete: cascade, index: true }
      poster_id: { type: integer, foreignTable: sf_guard_user, foreignReference: id,
required: true, onDelete: cascade, index: true }
    content: longvarchar(1024)
    created_at:
```

Listing 17: The schema file for profile comments table

The fields have the following meanings:

- **id**: the profile comment id (the primary key)
- **profile\_id**: the profile id, almost the same as the user id
- **poster\_id**: the user id of the comment poster
- **content**: the comment content
- **created\_at**: the time of posting

### 7.5.2 Creating New Profile Comment

Listing 18 demonstrates the method to add a new profile comment to a specific profile ID.

```
// Take the parameters from the web form in the client
$userId = $this->getUser()->getId();
$content = $this->getRequestParamter('content', null);
$profileId = $this->getRequestParamter('pid', null);Some explanation of the code
// Add to new profile comment to the database
$pro_cmt = new ProfileComment();
$pro_cmt->setProfileId($profileId);
$pro_cmt->setContent($content);
$pro_cmt->setPosterId($userId);
$pro_cmt->save();
```

Listing 18: The demonstrated code for adding a new profile comment

As shown in the listing, the comment content and the profile are retrieved from the Web form as parameters. The user id is retrieved via the browser session data.

### 7.5.3 Removing Profile Comment

Listing 19 demonstrates the way to remove a profile comment. Only the profile author can have the right and permission to remove the profile comment.



```

// Data input
$pCmtId = $this->getRequestParameter('pcid');
// Data validation
$pCmt = ProfileCommentPeer::retrieveByPK($pCmtId);

if ($pCmt) { // if there is this profile
// Get the profile from this comment
    $profile = $pCmt->getsfGuardUserProfile();
    $profile->setCommentCount($profile->getCommentCount() - 1);
    $profile->save();
    $pCmt->delete();
} else {
    $this->forward404(); // Prevent hijacking
}

```

Listing 19: The demonstrated code for profile comment removal

The function gets the parameter **pcid** as the profile comment id. The function looks up in the database if there is this profile comment id. If there is, then it will delete the comment from the database. Otherwise, it will return the 404 HTTP error code to prevent the hijacking.

#### 7.5.4 Application Demo

Figure 19 is the screenshot of input dialog box for the profile commenting. This block is only visible in the user profile page.

Comments (0)

**Add new comment**

Hi Subasa, you are my number one fan. :)


40/500

Add comment Clear

Figure 18: The profile comment input dialog box

The dialog contains two buttons. **Add comment** button is to add a new comment and **Clear** button is to clear the message. After pressing the Add comment button, a message is added automatically to the profile comments list as shown in figure 20.

Comments (0)

 **okidachi** (less than a minute ago)  
Hi Subasa, you are my number one fan. :)

**Add new comment**

40/500

Add comment Clear

**Figure 19:** The profile comment list and the add comment dialog box

The profile is added immediately to the list and the profile comment input box is empty for entering a new comment. The comment just entered can be viewed and removed from the author's account page, as shown in figure 21.

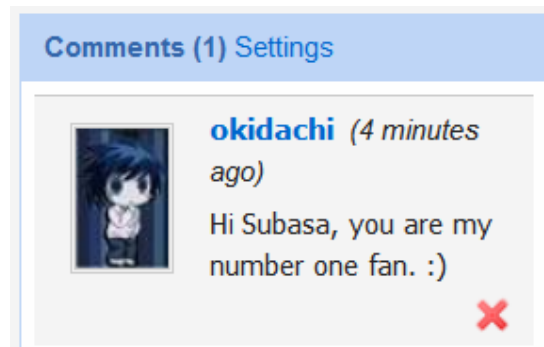


Figure 20: The profile comment list in the author account view page

Due to the scope of the project, the application specification is limited to only the core features of the social network. As stated in the social network section, the basic features required in a social network website are the public profiles and the friends system. However, in building these basic features, the system requires even more underlying features. User management, short messaging system and friends system are the ground features which are used to implement the basic requirements of a social network. The project also introduced a simple but advanced feature, profile commenting with the aim to give a sample of a social network extension and an advanced feature which can be used to extend the system and/or attract and increase the users base.

## 8 Conclusion

The goal of the project was to use the Symfony framework in developing a social network for comics. There are many websites nowadays claimed to be social networks but in fact they lack many basic features. This phenomenon is very common and it is due to the lack of studying the subject before building it. Thus, this thesis has tried to point out the basic definitions and features to build a site with the social network. Due to the scope of the thesis, there are many advanced features that could not be addressed. However, the study laid a solid ground to build a basic and proper social network.

The result of the project was the features of the social network of the social comics website <http://www.mahshelf.com>. At the time of the project, the website has been published and has been used by thousands of comics fans. The social network played an important role and contributed a lot to the success of the website in the earlier days. All the studies of the social network were used and applied in building the Mahshelf website.

Symfony is a good framework for building robust and scalable websites. Applying the proven industry patterns such as MVC and ORM made the code quality better and more scalable. Furthermore, it improved the productivity and significantly reduced the development time. Using Symfony in building a social network based websites was a good solution that helped in implementing many key features and maintaining the system.

## References

- 1 Fabien Potencier, François Zaninotto. The Definitive Guide to Symfony. United States: Apress; 2007.
  
- 2 Boyd, D. M., & Ellison, N. B. Social network sites: Definition, history, and scholarship [serial online] 2007. Journal of Computer-Mediated Communication, Volume 13(1), article 11.  
URL: <http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html>  
Accessed 9 January 2009
  
- 3 Bates, Regis J. Internet. McGraw-Hill Professional; 2002.
  
- 4 The Open University. TCP/IP protocol architecture [online]. United Kingdom;  
URL: <http://openlearn.open.ac.uk/mod/resource/view.php?id=175847>  
Accessed 8 December 2008
  
- 5 Eric van der Vlist, Danny Ayers, Erik Bruchez, Joe Fawcett, Alessandro Vernet. Professional Web 2.0 Programming. Indiana Polis: Wiley Publishing; 2007.
  
- 6 Oppliger, Rolf. Security Technologies for the World Wide Web. 2<sup>nd</sup> ed. Artech House. 2002.
  
- 7 Thau, Dave. Book of JavaScript : A Practical Guide to Interactive Web Pages. 2<sup>nd</sup> ed. No Starch Press, Incorporated; 2006.
  
- 8 Choosing an Object-Relational Mapping Tool [online]  
URL: <http://jacobcantwell.com/blog/?p=27>  
Accessed 23 February 2009
  
- 9 What is an IP address [online]  
URL: <http://computer.howstuffworks.com/question549.htm>  
Accessed 17 May 2009

- 10 Mercer, David. Schaum's Outline of HTML. McGraw-Hill Trade. 2001.
- 11 Client/Server Model [online]  
URL:  
[http://searchnetworking.techtarget.com/sDefinition/0,,sid7\\_gci211796,00.html](http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci211796,00.html)  
Accessed 17 May 2009
- 12 Overview and history of PHP [online]  
URL: <http://fi.php.net/manual/en/history.php.php>  
Accessed 17 May 2009
- 13 PHP Manual in English [online]  
URL: <http://www.php.net/manual/en/>  
Accessed 17 May 2009
- 14 Overview and history of MySQL [online]  
URL: <http://dev.mysql.com/doc/refman/5.0/en/history.html>  
Accessed 17 May 2009
- 15 Main features of MySQL [online]  
URL: <http://dev.mysql.com/doc/refman/5.0/en/features.html>  
Accessed 17 May 2009
- 16 Model-View-Controller Pattern [online]. eNode 2002.  
URL: <http://www.enode.com/x/markup/tutorial/mvc.html>  
Accessed 17 May 2009
- 17 The Symfony Architecture [online]. Symfony 2008  
URL: <http://www.symfony-project.org/blog/2008/06/23/the-symfony-1-1-architecture>  
Accessed 17 May 2009