Express Wordpress Develoment

How can WordPress be used to increase the speed and efficiency of the workflow in web design projects?

Patrik Lydman

Examensarbete / Degree Thesis

Utbildningsprogram / Degree Programme

2011

Förnamn Efternamn

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Digital multimedia
Identifikationsnummer:	
Författare:	Patrik Lydman
Arbetets namn:	Express Wordpress Development
Handledare (Arcada):	Owen Kelly
Uppdragsgivare:	

I det här pappret kommer att titta på processen att bygga webbsidor med Wordpress och grunderna till webb design som behövs för det. Vi kommer att titta på de olika komponenterna som behövs för att arbeta med Wordpress, och vi kommer att prova dem i praktiken i två riktiga webbsidor. Där vi kan se arbetet i praktiken.

Nyckelord:	WordPress, Optimization, Web Design
Sidantal:	
Språk:	engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Digital multimedia
Identification number:	
Author:	Patrik Lydman
Title:	Express Wordpress Development
Supervisor (Arcada):	Owen Kelly
Commissioned by:	

This paper looks at the process of developing sites using Wordpress and the basic web design that this needed to accomplish this. We will also research and demonstrate the different components of wordpress and what is needed to build working sites. Two actual sites demonstrate the work progress. In these examples we can see the workflow in practice.

Keywords:	WordPress, Optimization, Web Design
Number of pages:	1
Language:	English
Date of acceptance:	

OPINNÄYTE	
Arcada	
Koulutusohjelma:	Digitaalien Multimedia
Tunnistenumero:	
Tekijä:	Patrik Lydman
Työn nimi:	Express Wordpress Development
Työn ohjaaja (Arcada):	Owen Kelly
Toimeksiantaja:	

Tässä paperissa tulemme tarkastelemaan kehitys prosessia jota tarvintaan Wordpress sivujen tekemiseen ja myös Webb designin perusteitta jota tarvitaan että tämä onnistuu. Katsomme myös Wordpressin erilaisia komponentteja joista se kostuu ja mitä niistä tarvitaan toimivan sivun rakentamiseen. Katsomme myös kahta esimerkki sivua ja menemme läpi työprosessin mikä niihin kuluu.

Avainsanat:	WordPress, Optimization, Web Design
Sivumäärä:	
Kieli:	Englanti
Hyväksymispäivämäärä:	

Table of Contents

THE THESIS

1 INTRODUCTION

In this thesis I intend to take an in-depth look at basic web design and the CMS program Wordpress. The goal of the research is to provide students learning to be a designer with an analysis that covers the fundamental aspects of web design and Wordpress. I believe that there is a need for this kind of paper because most manuals and guides have concentrated their examples and tutorials around long complicated project-oriented processes that do not serve somebody new to dynamic web design. During the argument I will deconstruct the so-called waterfall model and provide an alternative framework.

What I want to do is try to eliminate some of the workload that goes into web development and optimize the work methods that we might use. This will help people new to web design, as well as helping more seasoned developers review their workflow and work habits when using Wordpress.

The first thing I will analyze is the standard model of web design that is based on a widespread model used in software design. This originates from the hardware side of development and has both strengths and weaknesses. I will look at this process step by step breaking it down to its essentials. It can be argued that the waterfall model (as it is known) is unintuitive in a design process that does not have the same strict workflow as the hardware manufacturing where it originated.

There are many applications that offer similar solutions to Wordpress but in this paper we will focus solely on Wordpress. This choice is dictated by the robust backend and easily updateable features that are incorporated into the platform. It has incorporated features that make it an attractive platform for developers and has a solid set of tools for the customer to help maintain the site. Wordpress has grown in 8 years since its launch to be the platform of choice for 13 percent of the webs 1.000.000 biggest websites, being uses by organisations ranging form Xerox to NASA.

Wordpress was chosen as the platform for this thesis over the likes of Drupal and Joomla as it offers the most robust back-end in terms of clarity and logic, and has one of the most active communities on the web producing high quality documentation and tutorials. There are many different options available in terms of structure and design choices, and most of these can be controlled using standard CSS and XHTML.

1.1 Aim of the Study

Web design is not a straight forward process and there are many ways of achieving satisfactory outcomes. This thesis aims to break down the process of web design using Wordpress into the separate stages needed for the design process. Firstly we look at building sites using the Wordpress framework and what this entails. Then we go through the process step by step and analyze what we need and what we do not.

This thesis provides two practical examples of building websites with Wordpress. The first is a website for the organization Gerocompetence which is a joint venture project in Scandinavia between geriatric care students and teachers. The site will be used for marketing the organization, for communication within the group, and also for the publication of related material. The site demonstrates the different steps in developing a medium size Wordpress site for an organization with social features. This site also offers a good example of the easily updateable features that are included in the Wordpress package.

The second example site is a web portal for the Finnish punk band Blossom Hill. This is a site that needs to communicate effectively with its audience as it is the primary information outlet for the band. The site was designed and executed very quickly as a way of testing findings and methods described below.

The practical examples are followed by a review of the strengths and weaknesses of Wordpress, and a look at ways to optimize the workflow. The thesis finishes with a summary of the issues and some conclusions.

1.2 Goal of the thesis

This thesis suggests a way of working that optimizes work methods regarding web design. It provides guides in basic concepts in web design and using Wordpress to construct pages. It aims to leave the reader with a good basic understanding of what

Wordpress and WCMS systems are, and a working knowledge of web design and design concepts that can be applied to other projects. The thesis therefore covers Wordpress from basic installation through to custom expansions and security issues.

The design process is demonstrated with two websites that were developed as parallel research projects.

The thesis concludes with a lengthy set of appendices that provide a comprehensive curated tutorial and information resource. This is not an official part of the thesis and the reasons for its inclusions are described below.

1.3 About the manual

The thesis is written in the standard format and ends with a conclusion. However the thesis itself is followed by a manual that was created in the process of studying the different aspects of the content management system Wordpress.

The sheer volume of information available on the subject is staggering and not at all well-organized. It therefore became necessary to compile a coherent set of notes to help with the development of the example sites and to make sense of the different parts of Wordpress and how they worked. These notes were later compiled and edited into the form of a manual.

The bulk of the text in the manual is not written by me. It has been edited from the original sources, to include only the essential core information. The origins of the sources are clearly noted. Where necessary I have written linking passages.

Although the manual is not part of the thesis I believe strongly that leaving this information out of the publication would be illogical, as it provides crucial insights into the inner workings of Wordpress.

I believe that the manual could have many uses for someone reading this thesis, and for that reason I have included it.

2 METHODOLOGY

This thesis is based on both theoretical and practical research. We have explored the theories of basic web design and how it is structured. We have also researched the workflow that goes into building sites utilizing Wordpress and how Wordpress is structured.

The ideas proposed in the thesis were tested during the development of two websites in 2010 and 2011 with a site for the Arcada Gerocompetence project and for a punk band Blossom Hill. The process of designing and testing these sites are discussed Part Three.

The research is divided into three basic parts. First there is a general look at web design and what can be improved in the workflow, as step two we look at the two example pages and go through the design process and in the last faze we look at the components of Wordpress.

We also saw the need to provide a comprehensive appendix of articles edited to be more accessible and to reduce the amount of redundant information that these kinds of articles contain. These help to give depth to the information section that this thesis contains and also provide tutorials to help explain the basic functionality of the different components.

3 RESEARCH

3.1 WEB DESIGN THEORY

The web design process is not a well-documented guidebook of steps that we can use to build a website as we would construct a car or a house. It is not unheard of that the design process is based on concepts that do not support the optimal structure of the website and is instead built on some other design aesthetic. The waterfall model is applied widely in the field by design companies. It gives a structure to the workflow and is a viable model to base a company structure on.

This is a model more applied by companies that individual designers. Companies need an organized workflow that suits a structured environment, and large work groups. Freelance designers can usually work in a more organic manner and do not need such a strict framework to work in.

We will look at the linear Waterfall model that is widely used in software development by the likes of NASA, and the United States army where it is known as US military standard DoD-2167. We will test this model as it is the most widely used system for development and see if holds up to scrutiny, we will also see if we can get a more non-linear model to work.

3.2 WEB DESIGN CRITERIA

To be able to successfully determine what is critical to our workflow when designing, and using Wordpress to do this. We have condensed what a good model should be able to do from smashingmagazine.com 2008 Dmitry Fadeyed, Dustin Wax and Vitaly Friedman articles on the subject. The rules are condensed into seven points.

Clearly define goals for the design: what is the site supposes to do and who will it reach.

Set a timetable: How long will it take to complete the project.

Goal driven design: Set design goals that must be met for the site. For example a working online shopping feature or a photo gallery.

Construction: How we put the components together and that everything works.

Does the site work: Does the site do what it's supposed to do and is it reaching the customer and fulfilling its client's needs?

Flexibility: How well can we accommodate change into the site, if needed?

Post-launch maintenance: How easy is the site to maintain to make changes if necessary?

In these seven points we have now condensed the principles we need to look at when we design a site in the conceptual stage and when we implement it into the Wordpress structure. These seven points, if met, should ensure a good stable product of high quality that we can deliver to clients on time and working.

3.2.1 The Waterfall Design Model

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation and Maintenance.

The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce, though Royce did not coin the phrase "waterfall" in this article. Royce presented this model as an example of a flawed, non-working model (Royce 1970). This, in fact, is how the term is generally used in writing about software development to describe a critical view of a commonly used software practice.

3.2.1.1 THE ROYCE MODEL

In Royce's original waterfall model, the following phases are followed in order:

1. Requirements Specifications

2. Design

3. Construction (AKA implementation or coding)

4. Integration

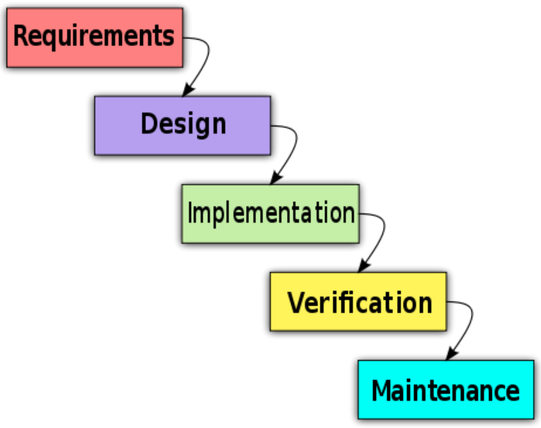
5. Testing and debugging (AKA Validation)

6. Installation

7. Maintenance

Thus the waterfall model maintains that we should move to a phase only when it's proceeding phase is completed and perfected. However, there are various modified waterfall models (including Royce's final model) that may include slight or major variations upon this process.

As this model is applied for a wide range of software platforms to model can wary greatly in its structure but in most cases it follows this basic flow.



The unmodified "waterfall model". Progress flows from the top to the bottom, like a waterfall. figure.1

3.2.1.2 ADVANTAGES OF THE WATERFALL MODEL

The advantages of the waterfall model are that it has a clear workflow and a readymade structure that enables the building of a coherent workflow model for software design and enables the estimation of time and energy that is committed to the different work phases, as an example the time spent early in the software production cycle can lead to greater economy at later stages of the project.

McConnell shows that a problem found in the early stages (such as requirements specification or design) is cheaper in money, effort, and time, to fix than the same problem found later on in the process. ([McConnell 1996], p. 72, estimates that "...a requirements defect that is left undetected until construction or maintenance will cost 50 to 200 times as much to repair as it would have cost to repair at requirements time.") To take an extreme example, if a program design turns out to be impossible to implement, it is easier to fix the design at the design stage than to realize months later, when program components are being integrated, that all the work done so far has to be discarded because of a broken design. This is an again a textbook example of generalized software design but by using Wordpress as our platform we sidestep a lot of problems.

3.2.1.3 DISADVANTAGES OF THE WATERFALL MODEL

Many argue the waterfall model is a bad idea in practice, believing it impossible for any non-trivial project to finish a phase of a software product's life cycle perfectly before moving to the next phases and learning from them. For example, clients may not know exactly what requirements they need before reviewing a working prototype and commenting on it. They may change their requirements constantly. Designers and programmers may have little control over this. If clients change their requirements after the design is finalized, the design must be modified to accommodate the new requirements. This effectively means invalidating a good deal of working hours, which means increased cost, especially if a large amount of the project's resources has already been invested in Big Design Up Front.

Designers may not be aware of future implementation difficulties when writing a design for an unimplemented software product. That is, it may become clear in the implementation phase that a particular area of program functionality is extraordinarily difficult to implement. In this case, it is better to revise the design than persist in a design based on faulty predictions, and that does not account for the newly discovered problems.

Even without such changing of the specification during implementation, there is the option either to start a new project from scratch, "on a green field", or to continue some already existing, "a brown field" (from construction again). The waterfall methodology can be used for continuous enhancement, even for existing software, originally from another team. As well as in the case when the system analyst fails to capture the customer requirements correctly, the resulting impacts on the following phases (mainly the coding) still can be tamed by this methodology, in practice: A challenging job for a QA team.

3.2.1.4 SUMMARY OF THE WATERFALL MODEL

Now that we have looked at the models pros and cons we cans star to draw our conclusion of the feasibility of the model. What we learned is that the waterfall model works quite well in the first criteria, as it works on defining a clear structure of the different components that the site must have and what these components should do. The Waterfall model also handles the issue of scheduling quite well in principle but with on fatal flaw, the waterfall model does not take into account change weary well, as it is a linear system a change in the late part of the development face can throw the hole schedule of course dramatically as change in the late cycle of development ripples backwards in hypothetical code of the site and renders it useless.

Again in the third basic rule we set out, the model works well the waterfall model is weary goal driven and works well in setting goals for the project. The waterfall model starts again showing its weakness in the construction part of the goals we set for our self, in principle the model could work weary well for constructing websites but only rarely can we go part by part trough a project without the need for change and this again breaks the model.

The next point of interest is if the waterfall model can incorporate testing and again the model falters, we can test the separate components of a site, but compatibility issues almost always rise, this is handled better in Wordpress but is still a again a test where the model brakes. Then in the terms of flexibility again the waterfall model breaks as it is the polar opposite of what the model is designed to do. As for post launch maintenance the model could work well if it was taken into account in the design face of the project in the beginning but the fluent post launch updates we can achieve using Wordpress are out of the reach of the waterfall model.

We have now summarized the waterfall model and have noticed that it is flawed, but it can still be implementing if we lack a better system. In the next chapter we will be discussing how it works in Wordpress.

3.2.1.5 WORDPRESS WITH THE WATERFALL MODEL IN MIND

When we have looked at the pros and cons of the model we can draw the conclusion that the waterfall model is flawed, but we can salvage some parts of it that work in Wordpress development.

The model is implemented in software design ranging from building huge networks to designing small web-pages for clients. As we are concentration our efforts on analyzing the model that most suits us that are working with Wordpress, we can side step a lot of the issues inherit with the waterfall model.

The modular design that is built into Wordpress enables us to concentrate on the designing aspect of the project more than the technical aspect that would be building an intuitive interface that was coupled with a PHP driven CMS system. If we would be designing a backend we would be using a quite unmodified version of the waterfall model, but as we get this functionality built into our platform that is Wordpress we only need to design and modify the source code in small increments to reach our designing goals.

One of the things that must be changed to suit our goals in Wordpress web design is the rigid 100 percent complete stage thinking that permeates the Waterfall model and which is time consuming and ineffective if the design goals change, with is quite normal in web design. Here Wordpress modular design works for us as it means we can test different software configurations effectively and that we do not need to lose a lot of time exploring these options.

What we can use is the first stage titled “Requirements” of trying set specific design goals for our projects as this will help to define or design goals and what the project will need in terms of time and resources. This stage will also hopefully help you map potential needs of the client and this enables us to hopefully anticipate the need of the project.

The most time consuming part utilizing this model of working with Wordpress is the “Designing” phase, but this phase is also linked to the “Implementation” phase. When designing using Wordpress we can mix these to work processes together as this is the most natural way of designing with Wordpress. The way Wordpress is built is that a set of rules control the whole design of the site and this is called a “Theme”. The functionality of Wordpress will be discussed in-depth in coming chapters. This means we start producing a highly modifiable prototype of or final page in a very early stage of the design process.

These prototypes can then be used to identify in-depth the need of the project and to do limited usability study's and even market research depending on the degree of sophistication of the prototype. This will hopefully also help the potential client narrow down his lists of requirements for his project and help identify redundancies or short comings of the project.

As we now covered the design and implementation parts of the model and applied them to a design process that is dictated by the use of Wordpress we can now move to the stages of "verification" and "maintenance".

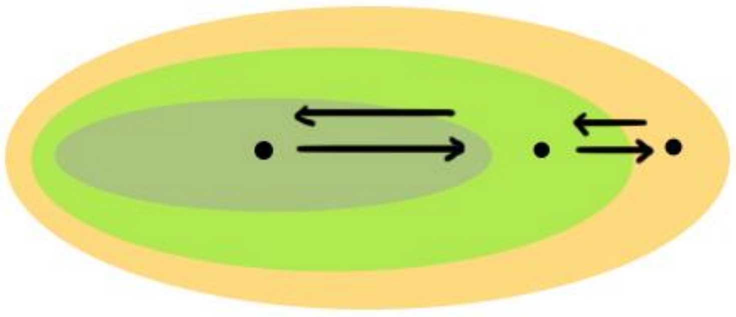
As the process of verification should be a quite straight forward with the process of using prototypes to demonstrate interface, graphics and layout. The client should have a clear understanding of what has been built and should not have any problems of approving the final product.

The final step of maintenance is the process of setting up a routine of upkeep for the product that may entail somebody actively producing material and updating the content of the site or maintaining key services it provides. When working With Wordpress this entail a web hosting service that has MySQL and PHP to support the CMS features of the platform. The Wordpress backend control panel also enables clients to go in and change content with relative ease. As with any product there can be issues with maintenance and here we can always offer updating services but the high level of user friendly features that do not require programming knowledge should help the problems of site maintenance for the client.

3.2.2 The Ripple Model as an alternative

When we now have looked at the Waterfall model and its pros and cons we may come to the conclusion that this model does not work for us. Because the Model could not adequately provide us with a good framework for designing we propose one of our own.

What we propose instead is a ripple model where we have a series of steps that, for example, go from a to d, a standard linear model, but we modify this, we work under the assumption that at every step of the process we should be able to go back and change attributes or features, in other words if we make a change in section b it creates a ripple that changes a and c. If we work using this kind of setup we can take the aspect of change into the work method and embrace it. As discussed earlier the waterfall model can create drastic setbacks in the design process if changes need to be made, we eliminate this problem by taking change as a constant, in practice this means that we jump between the different design phases, this is easily accomplished as we are suing Wordpress as our launch platform and only in the very last part of the process do we make the final design decisions and launch the site. When working with Wordpress we do not need to commit ourselves to strict design decisions and the modular design of Wordpress and the fluid design esthetics of the ripple model works well to in supporting each other. This model has all that we require when we set out the rules we wanted to adhere to, with one minor flaw, as it has no rigid structure scheduling can be hard and deadlines can be compromised, but this issue is discussed in later chapters (se Artisteer).



Ripple model figure 3

3.3 BUILDING SITES WITH WORDPRESS

In the following chapters the core concepts of Wordpress development will be covered. This will be a study of basic of what went into building two sites utilizing Wordpress. We will look at what when right and what went wrong in the individual site process, then we will summarize our findings and compare that to the goals we set for successful web design and see if we could meet those standards. Next we will go through the building of the two example pages, the Gerocompetence elder care workgroup site and the Music themed site for the band Blossom Hill.

3.3.1 Gerocompetence Webportal

The client in this case the Gerocompetence workgroup is collaboration between different countries' teaching institutions that specialize in eldercare. They are part of the Nord-Plus network that is the cover organization that they are connected to.

They needed for internal communication and marketing a web presence. The design of the page was directed at teachers and students in the field and has to be an informative entity that gives a good picture of what the organization does and also support their internal communication and publication needs, in its first draft there was also talk of it being multilingual to support directly the Scandinavian languages, but this option was dropped later in the process as it became clear that having to publish material in 5-7 languages was counterproductive and the language was changed to English.

Wordpress was chosen for this project as it has a simple backend that

As the organization was new and had no material everything had to be made for the project, this included a graphic profile, logo and a design that corresponded with Arcada's graphic profile and also with the style utilized by the NordPlus network. The text part of the project was handled by the client.

The project started at the Christmas of 2010 with a preliminary meeting of the workgroup or as we will refer from this point on the client, with a preliminary meeting to go through the major points that where specified in the work sheet that the client provided.

Already in the first meeting some features where changed to better accommodate the clients need as it was deemed that an extensive community driven feature, in this case it was the Wordpress plug-in Buddy press that was first outlined for the client, in an attempt to outline their needs. It was fast apparent that a comprehensive social media hub was not needed because the client did not like that it in some ways resemble Facebook. This idea was dropped on the spot and was substituted with a forum features that was adequate for there need but with the potential to grow if needed. The basic layout of the page was also suggested to mirror the design of Arcada polytechnics 2010 site layout. From here we started the first design face of producing a couple of demos for the preliminary design of the page. Here it must be noted that the design of Arcada's pages changed and this translated to a delay in the work process ass the sites graphic profile had to be updated.

Over the winter holidays we conducted some research into similarly themed pages and conducted studies into the organization’s purpose and collaborator sites. The first design was to incorporate an updated version of Arcada's site that fitted the graphic profile needed for the Gerocompetence site. The carrying theme was to have all the country’s that where I the project represented in the site layout in some way. This ended up being represented in the header as a graphic element represented by a map view of the country’s and stylized background image in the same theme. The first version was model on the pastel color utilized in the 2010 Arcada pages but was changed to more black and harder version to more closely mirror the design of Arcada's 2011 pages.

The first version of the logo was just a rough sketch of the letter g and c put together to have something to show and the color scheme was very pastel oriented and did not represent a Scandinavian feel at all. The second draft was better received as it had a simple layout that represented more the Scandinavian work group and fit the theme of the organization better.

As we moved forward in the design process we went through 3 drafts of the page before settling on a final design that turned out to be as explained before incompatible with the revised design of Arcada's new site.

As the client or workgroup was partly based in other countries the method of communicating was handled by a contact person and meetings about logos and design was conducted over video conference, it must be noted here that for a video conference to work the hardware must work without problems or else the meetings can’t serve their purpose.

In May of 2011 the site was ready and where deployed on servers in Arcada, the site was maintained by students and teachers and we the developer organized a teaching session for the administrative personnel to train them in the use and updating of a Wordpress site.

3.3.1.1 BUILDING THE SITE

Building the core for this site went through many steps in its development, as this was this writer's first Wordpress site the design process went through a lot of trial and error before finding a work method that suited me and grasping some of the fundamentals of how Wordpress operates.

Now we will go through the process step by step to demonstrate on way of building a Wordpress site, this is one way of accomplishing a design goal and is quite basic, feel free to intrepid this as one model of working or implement pieces freely that can help in you in a design process.

The first step of the program was receiving a list of features that that the client wanted for their site that included basic information about formatting but also more importantly two features that we could see might be considered time consuming for the project. The first was Multilanguage support for the site and the other was a community features that was loosely specified. A design for the graphic profile was not included and this needed further research..

The work proceeded with a preliminary meeting where already big changes were made, the community feature was reworked, and the initial design that was suggested was supposed to utilize the Wordpress plug-in Buddy Press that incorporates a fully realized social network feature with the options for work groups and creating a system that could be described as having shared features with Facebook, but this was canceled in the first meeting and it was decided to opt for a more streamlined approach and it was decided that the site would use a forum. A forum would enable communication between the students and teachers with the use of information chains that can be moderated or deleted, and enable the publishing of material on the forum if needed; it would also enable limited QA on the forum threads.

The other feature that was initially discussed was a Multilanguage support for the site. The first draft called for material on the site and interface to be localized, but as this was clearly unpractical for a project that incorporated so many different nationalities. As we moved on in the design of the language support the next face was to at least incorporate the Multilanguage support for the interface of the site, this could be accomplished by the use of language packs that change key word in the Wordpress construct or by a plug-in. We experimented with a plug-in that translated directly from the Google translate data base, but this to was dropped when considering it translated the whole page and not just the interface. But in the end the whole Multilanguage support was dropped as it became clear that it would not serve any purpose as producing text in 6 different languages and with a growing number of participants would be cumbersome. It was decided that the page would be English in it interface. This is a typical example where time can be lost in the design process when design goals are not considered or studied for their feasibility for a project.

The next phase was presenting a first draft of the design of the site for the workgroup; this was accomplished with a rough draft made with Photoshop that demonstrated a basic layout and color scheme. The main point of the draft and design was the banner that had all the participating countries demonstrated in the fashion of a stylized map.

The first design of the site was well received and we moved on to the coding aspect of the site. We first setup a Xampp virtual server at home to build a good test bed for experimenting with Wordpress and soon also opted to by a web hotel to have a online presence where we could demo the sites different stages to the client.

It became clear in the development process that modifying an existing theme would have an unsatisfactory result and building a theme from scratch might be too time consuming in the at this stage of the development face. The templates that where available proved to be unsatisfactory.

Here we utilized the Artsteer template building program that to enable us to work faster and to produce a workable basic template that had 90 percent of what we wanted and the rest we could code our self or modify to fit our design needs.

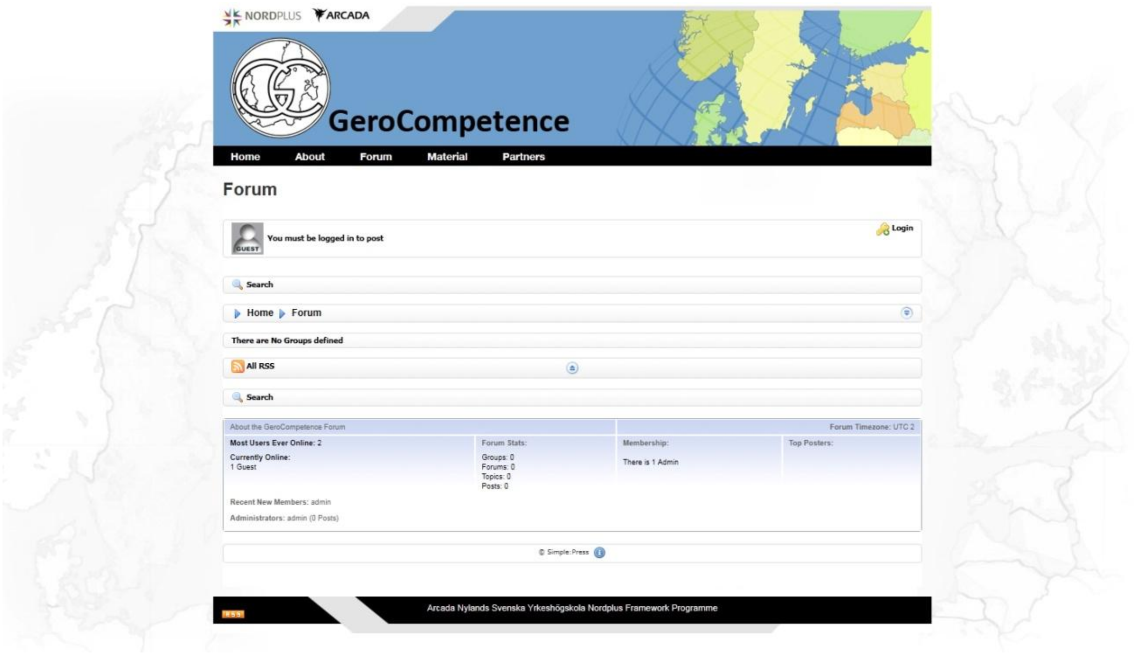
When we started using Artisteer we could produce a demo site with was presented to the workgroup in our next meeting. In this meeting we only had the contact persons from Arcada. The revised more toned down version got a more negative response but a compromise was reach and we quickly moved to other matters as the Multilanguage support and forum feature.

The Multilanguage support is a good example of editing the Artisteer theme and extending the Wordpress site functionality. The language plug-in was easily installed in the plug-in directory of Wordpress and the plug-in created a widget with different flags to enable the navigation of the different languages. As the Artsteer theme did not have a ready-made widget area under the main navigation bar that would have suited the widget but had a default widget area that corresponded with the placement it was possible to position the navigation in a Wordpress created widget are. This did however create a problem as the widget area was too thick for the widget application sit nicely into the design. This was corrected by going into the code and where the widget areas where defined and changing the height setting to suit the widgets height and this accomplished a finished look to the insert. The feature was ultimately dropped but this is a good example of what can be accomplished by combing Artisteer and an understanding of how Wordpress works.

The other feature was the forum that the site needed; the problem here was finding a suitable plug-in forum that had the necessary features, but was not overly burdened with features that would never be used or implemented and would make it harder to use for the client. The answer was Simple Press a simple page based forum but with the option to extend and accommodate more advanced use if needed. In the forum we can define roles as administrator and read and write privileges for user that register or any other configuration the client might wish for. A forum functions well when publishing study material as is the case with this site as we can lock forum treads with important information and in this way build a knowledge base in the field of the client, it is also a good medium to publish ongoing topics or other information that has the same relevancy.

As the development of the site moved to the next workgroup meeting, the basic layout of the site was ready this meaning the graphic profile, the sites functionality was finished and it was also decided in this meeting that to finally drop the Multilanguage support in favor of an English layout to accommodate the different nationalities. The feedback on the design was mixed but to its tone positive. What was left for the design was finalizing the logo design and the font for the logo text (Gerocompetence).

The logo then went through a 5 different versions that were presented in a in a meeting for the Arcada work group and it was deiced that with small alterations the design and the logo where ready for deployment online, the workgroup also committed in this phase to start providing text and material for the project. This also led to the installation phase on Arcada's servers.



Forum

You must be logged in to post

Login

Search

Home Forum

There are No Groups defined

All RSS

Search

About the GeroCompetence Forum | Forum Timezone: UTC 2

Most Users Ever Online: 2

Currently Online:
1 Guest

Forum Stats:
Groups: 0
Forums: 0
Topics: 0
Posts: 0

Membership:
There is 1 Admin

Top Posters:

Recent New Members: admin

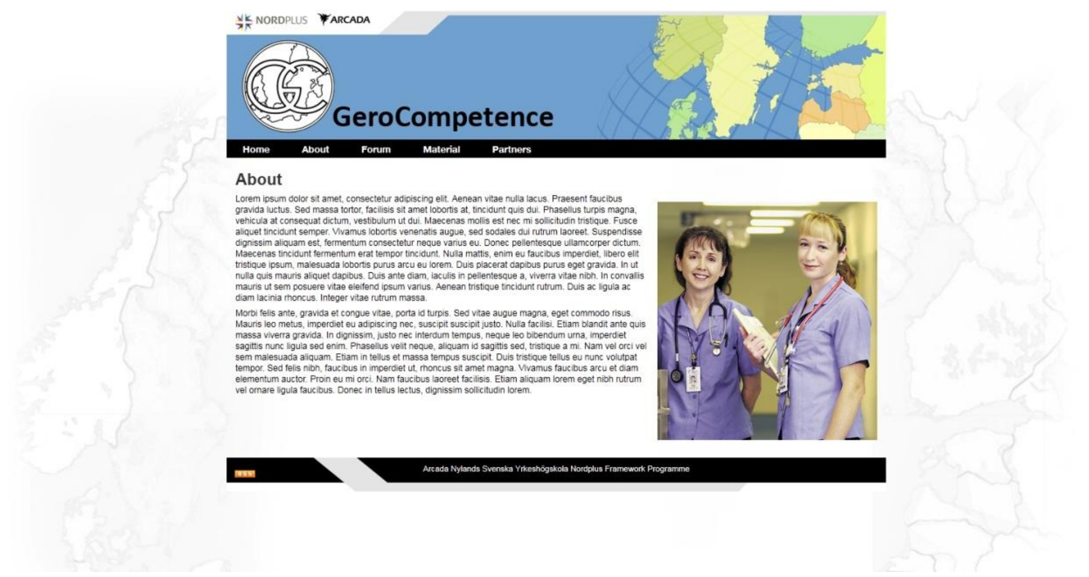
Administrators: admin (0 Posts)

© Simple:Press

Arcada Nylands Svenska Yrkeshögskola Nordplus Framework Programme

When the project was discussed with Arcada's server administrators there raised a problem with the design of the page. The graphic profile was to different to fit in with Arcada's graphic profile and this needed to be remedied. The problem was that the pages where designed to be in line with Arcada's 2010 sites and now Arcada had changed it design to a more black and hard design 2011. This was not a big problem because the design of the page could easily be modified trough the change of the graphical element in the footer part of the page, and by changing the main navigation bar to black, and finally modifying the header. This could easily be done by editing the root directory where the pictures where and changing the color code in the CSS that controlled the navigation bar in the menu.php, alternatively this could also be made directly in Artisteer but if we would have deploy modifying code into the theme that was installed we would lose it as the new theme would wipe out the old one.

When the changes to the design had be made and been inspected and agreed by the workgroup we could install the site on Arcada’s server, this was accomplished by the IT department installing the basic Wordpress package, then they provided the developer with the administrative passwords for the Wordpress package and we uploaded the theme.



About

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean vitae nulla lacus. Praesent faucibus gravida luctus. Sed massa tortor, facilisis sit amet lobortis at, tincidunt quis dui. Phasellus turpis magna, vehicula at consequat dictum, vestibulum ut dui. Maecenas mollis est nec mi sollicitudin tristique. Fusce aliquet tincidunt semper. Vivamus lobortis venenatis augue, sed sodales dui rutrum laoreet. Suspendisse dignissim aliquam est, fermentum consectetur neque varius eu. Donec pellentesque ullamcorper dictum. Maecenas tincidunt fermentum erat tempor tincidunt. Nulla mattis, enim eu faucibus imperdiet, libero elit tristique ipsum, malesuada lobortis purus arcu eu lorem. Duis placerat dapibus purus eget gravida. In ut nulla quis mauris aliquet dapibus. Duis ante diam, iaculis in pellentesque a, viverra vitae nibh. In convallis mauris ut sem posuere vitae eleifend ipsum varius. Aenean tristique tincidunt rutrum. Duis ac ligula ac diam lacinia rhoncus. Integer vitae rutrum massa.

Morbi felis ante, gravida et congue vitae, porta id turpis. Sed vitae augue magna, eget commodo risus. Mauris leo metus, imperdiet eu adipiscing nec, suscipit suscipit justo. Nulla facilisi. Etiam blandit ante quis massa viverra gravida. In dignissim, justo nec interdum tempus, neque leo bibendum urna, imperdiet sagittis nunc ligula sed enim. Phasellus velit neque, aliquam id sagittis sed, tristique a mi. Nam vel orci vel sem malesuada aliquam. Etiam in tellus et massa tempus suscipit. Duis tristique tellus eu nunc volutpat tempor. Sed felis nibh, faucibus in imperdiet ut, rhoncus sit amet magna. Vivamus faucibus arcu et diam elementum auctor. Proin eu mi orci. Nam faucibus laoreet facilisis. Etiam aliquam lorem eget nibh rutrum vel ornare ligula faucibus. Donec in tellus lectus, dignissim sollicitudin lorem.

Arcada Nylands Svenska Yrkeshögskola Nordplus Framework Programme

When this was accomplished we created the necessary pages and installed the forum plug-in but left the site blank. The choice that dictated this was that we would coach the personnel that would update the site in how to ad and delete text and how to use the various functions of the page. This could easily be demonstrated to the administrators of the site if they did it theme self and in the process learn how the site worked.

When the IT department had installed Wordpress and the developer had uploaded the necessary components, we heal a class for the key personnel where updated the sites text and news feed and with this it was done.

3.3.1.2 SUMMARY

When working with Wordpress there is a learning curve when we try to understand logic that operates it but we can use tools to our advantage, as a side note this was the authors first site utilizing Wordpress and the process contained a lot of trial and error. Armed with a basic understanding of the way Wordpress operates and tool that helps us visualize and build constructs like Artsteer greatly increases the speed a witch we can deploy sites on the web. A good structured design is what is going to save us time and effort, in this project a lot of time was spent investigating the social network feature and the Multilanguage support that were never adopted in the final design, if the design goals would have been outlined more clearly this could have been avoided, this is a clear example of the Waterfall models problem when designing a website that the specifications might change in a very late stage of development. Form a technical point of view this site contains every specified feature fully implemented and the only bigger problems that where encountered where scheduling issues with material and installation.

3.3.2 Blossom Hill website

The Blossom hill web portal is a site that is built to support and work as a informative platform for the Helsinki based punk band Blossom Hill. It primary goal is to inform the public of dates and where the band has concerts and also promote their music and sell it. The site is also linked to Facebook, Twitter and MySpace to provide the audience with a full social media function where they can follow the band and what they are doing.

Preliminary work on the site started in January 2011 but was only at an idea phase ass the Blossom Hill was going to publish a new album for the summer of 2011 and wanted a fresh web presence to support the launch of the album.

The core points of the site where to be easy to keep up to date and have a graphic profile that was close to the new album and support its cover art to some degree. The sites graphic profile was to be “punk” in a conventional way but the band wished that it would not be too typical and fit well with the bands image. Also what were to be incorporated into the site were a music player and an extensive photo gallery to chronicle the bands different gigs and happenings. The music player was an afterthought and would be dropped from the final page in favor of MySpace but as an experiment in extending the capabilities of Wordpress it is worth mentioning.

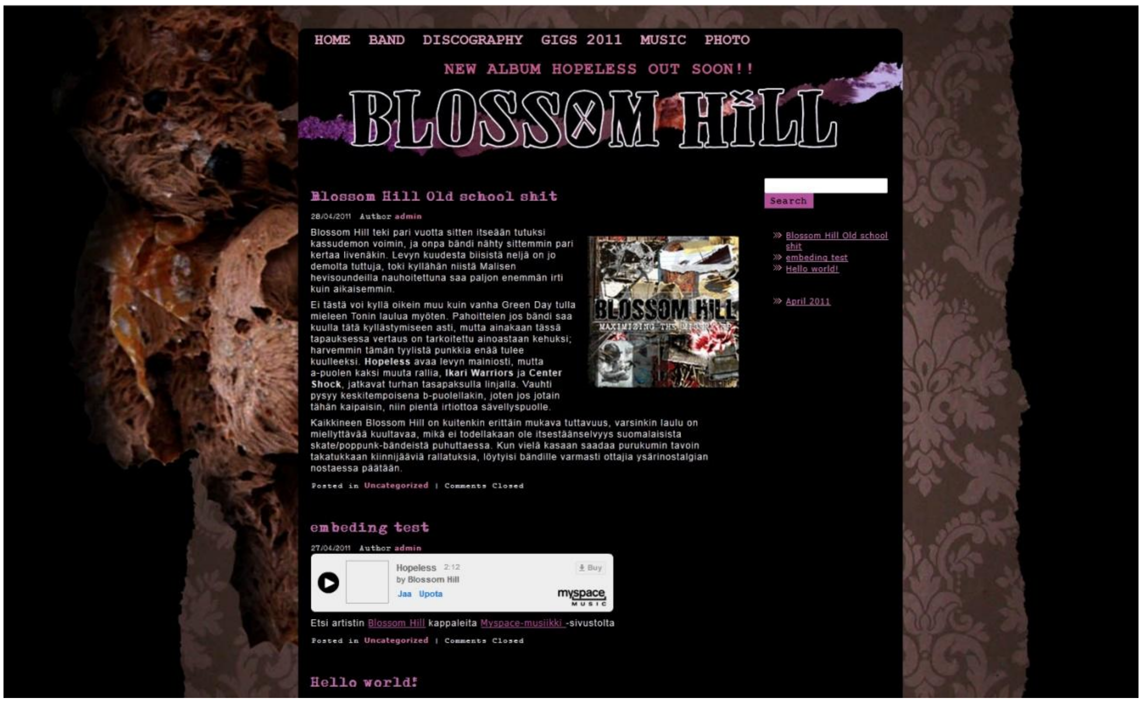
There were also issues in the launching of the site as the band owned the name blossomhillband.com and had a rudimentary website, but this was discarded in favor of a WCMS site that would accommodate their updating needs.

3.3.2.1 BUILDING THE SITE

The building of the site was handled in a slightly non standard way as the process never involved going trough phase where we would have presented a rough draft of the site to the client. As the client provided a substantial information pack that had the logos and graphic material ready at the beginning of the design process and a clear idea what they wanted we almost instantly moved to a rough working draft of the site that had been created with Artisteer.

The site was design in a primary black tone with pink and white highlights that formed a pleasing contrast in the site. The main theme of the site was based on a disheveled teddy bear that was to be used in the album cover of the album “hopeless”. This paired with the print of old wallpaper and a torn effect around the edges formed the basic graphic profile of the site. The header was design to primarily bring the logo of the band to the forefront and has a small graphic inlay that corresponds with the overall theme of the site.

The first meeting with the client was a primarily a brainstorming meeting to feel the clients need for the page and it became clear very early that the design was going to be the wary close to the album. With this in mind we could make a fully functioning demo that the client could try and also in this way we could gauge what else the client might need for the site.



HOME BAND DISCOGRAPHY GIGS 2011 MUSIC PHOTO
NEW ALBUM HOPELESS OUT SOON!!
BLOSSOM HILL
Search
Blossom Hill Old school shit
embeding test
Hello world!
April 2011

Blossom Hill Old school shit

28/04/2011 Author admin

Blossom Hill teki pari vuotta sitten itseään tutuksi kassudemon voimin, ja onpa bändi nähty sittemmin pari kertaa livenäkin. Levyn kuudesta biisistä neljä on jo demolta tuttuja, toki kyllähän niistä Malisen hevisoundeilla nauhoitettuna saa paljon enemmän irti kuin aikaisemmin.

Ei tästä voi kyllä oikein muu kuin vanha Green Day tulla mieleen Tonin laulua myöten. Pahoittelen jos bändi saa kuulla tätä kyllästymiseen asti, mutta ainakaan tässä tapauksessa vertaus on tarkoitettu ainoastaan kehuksi; harvemmin tämän tyylistä punkkia enää tulee kuulleeksi. **Hopeless** avaa levyn mainiosti, mutta a-puolen kaksi muuta rallia, **Ikari Warriors** ja **Center Shock**, jatkavat turhan tasapaksulla linjalla. Vauhti pysyy keskitempoisena b-puolellakin, joten jos jotain tähän kaipaisin, niin pientä irtiottoa sävellyspuolle.

Kaikkineen Blossom Hill on kuitenkin erittäin mukava tuttavuus, varsinkin laulu on miellyttävää kuultavaa, mikä ei todellakaan ole itsestäänselvyys suomalaisista skate/poppunk-bändeistä puhuttaessa. Kun vielä kasaan saadaa purukumin tavoin takatukkaan kiinnijääviä rallatuksia, löytyisi bändille varmasti ottajia ysärinostalgian nostaessa päätään.

Posted in Uncategorized | Comments Closed

embeding test

27/04/2011 Author admin

Hopeless 2:12 by Blossom Hill Jaa Upota Buy

Etsi artistin Blossom Hill kappaleita Myspace-musiikki -sivustolta

Posted in Uncategorized | Comments Closed

Hello world!

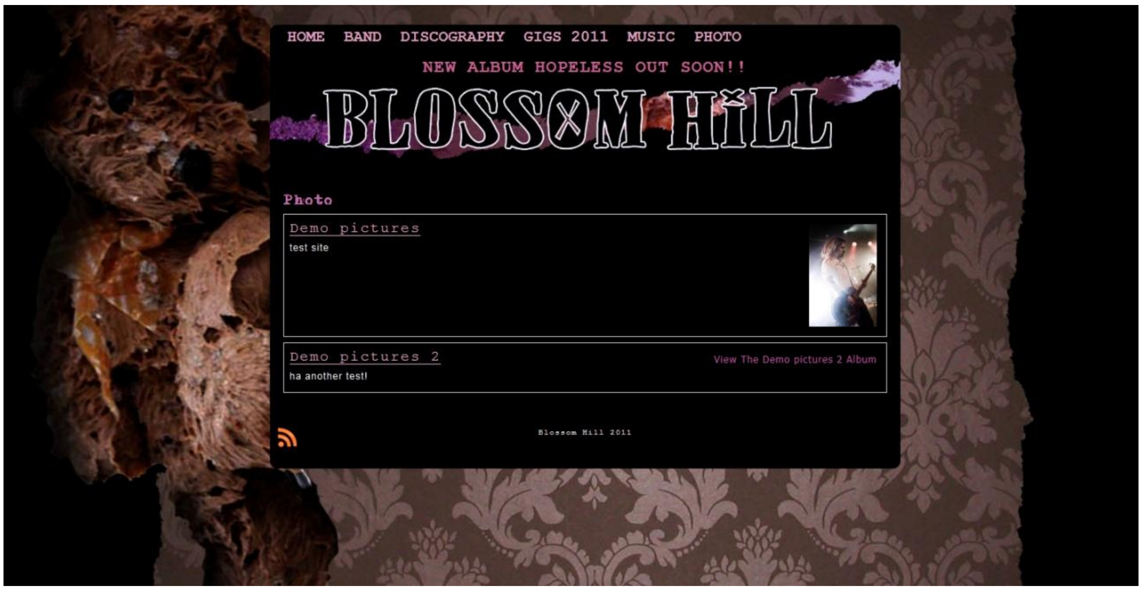
At this point of the process we had an estimated 80 percent read graphic design of a page and we only waited for the finalized version of the album art to finalize the look of the page, if the page was to be deployed before the final version was finished we could always go back to the root and change the pictures that are in the background and header.

The plug-ins that the site uses to expand its usability is a light box gallery plug-in that converts the standard gallery of Wordpress to utilize a light box style viewing experience and makes the default gallery look nicer. This is a tried and tested plug-in that is easy to deploy and has proven to be reliable. This plug-in was discarded after it became clear that the band wanted more photos on their site and this plug-in works best when used in small photo galleries. Another plug-in that that replaced the light box plug-in, was a plug-in that would make sorting pictures easier and enable the client to build files for different picture set as there would be a lot of different sets of pictures, for example "Puntala Rock 2011" picture sett from that particular happening. For the feature we chose the WP Photo Album plug-in, that was extremely easy to use and to sort pictures with and that was fully editable if needed using CSS, if we needed to create a customized gallery we only needed to change the style and layout of the photo album, copy them/edit wppa_theme.php and theme/wppa_style.css to your active theme's folder, and edit them just like we discussed in the chapter on Wordpress and CSS. This enabled us to present to the client a fully customizable and easy to use gallery for their site.

The Music plug-in was a request by the band to replace their dependence on MySpace and this was a time consuming trial and error phase, the problem was that most of the players where linked to music providers as for example Spotify or other programs with similar themes or they were so heavy or badly coded that adding or deleting music for the band member would have been very hard or in some cases probably impossible. The conclusion we reach in a second meeting was that the audience is used to MySpace and that they might not connect well with a system that was not as fluent as that in MySpace even if it would be technically similar and the feature was dropped. Here we can see that the plug-in feature in Wordpress is very hand when we want to try different features and what works and what does not, just as we discussed in the chapter Wordpress plug-in.

The last plug-in the site utilizes is an optimizing tool that tries to speed up the database and in this way make the site faster. This can be a good thing to have as this site compared for example to the Gerocompetence site will not be hosted on an extremely fast server that the school of Arcada employs but is instead hosted privately. The actual speed gains that we reach using this plug-in are hard to determine but even a small increase in speed is worth the installation.

As the graphic profile was ready and the hosting was registered there was only the issue of connecting the domain name that the band owned to the web hotel or hosting we had registered for them, this turned out to be quite a lengthy process as the member of the band that handled the name and owned it was very slow at providing the transfer data to hosting company. This is an example of that even a very efficiently moving workflow can be halted by the human factor, this has been a recurring trend in the development of these two example pages, as vital information and the clear go ahead has been missing in the steps of the work process to some extent.


HOME BAND DISCOGRAPHY GIGS 2011 MUSIC PHOTO
NEW ALBUM HOPELESS OUT SOON!!
BLOSSOM HILL
Photo
Demo pictures
test site
Demo pictures 2
ha another test!
View The Demo pictures 2 Album
Blossom Hill 2011

When the site was online and the theme and relevant plug-ins where installed and the client had approved of the design we moved to the last face of the project that is training the client to use the system. WCMS systems have their drawbacks in mater related to speed on the web, but their high customization and a well-defined graphic backend interface enable us to teach clients to maintain and upgrade their sites easily, and if the client is well versed in using blogging software or other online database programs as for example Flicker, the graphic interface should not present problems for them if we can provide them with coaching or a simplified tutorial for controlling base functions.

As a side note the client wanted the site online before the album was released so the developer made a commitment to make alterations in the graphic profile of the page to accommodate drastic design changes in the album art if needed. This is a commitment we sometimes have to make to hold deadlines but as we can provide a little bit of flexibility in our work schedule as the ground work is done and the only thing we need to make is alterations in the base graphics.

3.3.2.2 SUMMARY

As this was a site that benefited greatly from the experience we gathered doing the Gerocompetence site, the work flow was much more rapid. The design goals where met in every step of the development phase and no technical compromises where made. It can be argued that dropping the music player was a compromise but if we look at the project as a whole we can see that their presence in MySpace is much more beneficial from a marketing standpoint, and sites playing music as a default are usually ill received.

The project goal was to create a web portal for the client that could be sued for marketing of the band and for making a clear site where fans of music can readily see current news and where the band will perform next, the client was satisfied and from a professional standpoint the site work on every level.

3.4 REVIEWING THE PROCESS

As the goal of this paper was to examine the structure of basic web design and the basic structure of the Wordpress platform, that was handled by giving examples of the basic structure of Wordpress and lightly touching on design questions that are usually applied when doing web design. In the next chapters we will discuss what when right in the process and what went wrong. We will look at if we could optimize the workflow and draw our conclusions.

3.4.1 Strengths

What we noticed when working with Wordpress was that the software package that is Wordpress is very easy to work with when we first overcome the initial learning curve. The program has a solid online community that offers tutorials and help in all steps of the design process if needed. If we go through what we did step by step and evaluate what worked first.

In the beginning of the design process following the basic design model loosely help structure the building of the Gerocompetence site as the guidelines that were given for the project where very vague, and this helped narrow down the design and get a good end result that the client could approve of.

In the Blossom Hill site again we could move forward a lot faster because the client had a very clear idea of what they wanted in terms of design and this enabled us to move directly to building the site.

Building the different sites had setbacks that will be discussed in the next chapter, but some aspect of the design did move along very well. Using the template generator Artis-teer proved to be a invaluable asset in the design process as it enabled the designer to explore themes and different configurations of the Wordpress site, it enabled for exam-ple setting up typography options that would have been extremely time consuming to do by hand and as the program gave a graphic representation of what we were doing it was easier to visualize the end result better. Using the generator also made it easy to go into the code after the Theme file was installed and changing settings directly in the code to generate the final results that we want for the project.

One of the positives with Wordpress was that changing a design at the last minute was very easy, and involved minor changes to the CSS that restyled the navigation and side bar and the rest was just changing the jpg files that represented the banner and footer.

The modular design also worked well in providing a good environment to test features that where to be implemented into the site, and as this utilized plug-ins it was a relative-ly simple task to try different plug-in configurations and find the optimal solutions for the different pages, as a side note we must say that some of the design decisions regard-ing the plug-ins where very time-consuming and will be mentioned in the next chapter.

From a design point of view the Blossom Hill site was a very uncompromised final so-lution, this was mainly because the client had a very clear image of the final product and this enabled a very straight forward design style to be implemented.

The Gerocompetence sites had more design issues that will be covered in the next chapter, but as an overall design challenge, utilizing a extremely modified Waterfall, or it might be better I few say that we chose to pick selected pieces of the model for our use when doing the design work.

One of Wordpress' very strong points was the dashboard that is used to control the page, this is one of the key features that work very well when guiding the client in the use of the site, this enables us to easily teach or coach the clients to be able to update their site independently and the clear layout also enables us to if needed give technical support in later stages or after the launch of a site.

The design process utilizing Wordpress as a launch platform has a learning curve that must be taken into account when talking about the work efficiency, it can be noted that the Gerocompetence site to considerably more time to make as it was the first Wordpress driven site the Author had ever made, and the much speedier and fluent design process of the Blossom Hill page benefited greatly from the experience gained from developing the first site.

A grater familiarity with the code base after the first project helped to make the project go more smoothly and it must be noted in any case we need to step outside the Wordpress frame work we must have a working understanding of the code.

Both project led to a finished product that was released and the inherit features in Wordpress ensured that they worked on their respective servers.

3.4.2 Weaknesses

The first thing that created and obstacle was the way Wordpress is built, the structure and hierarchy is not well defined if we for example start using a blank template as our starting point and starting from the beginning building a theme is quite daunting, this problem was over come when we applied Artisteer to the problem.

Some of the biggest problems where in the design specifications for the Gerocompetence web portal as it called for community features and Multilanguage support, a lot of research and effort where spent in trying to incorporate this features that in the end both turned out to be redundant for the client, some social features where maintained but it is unclear if the client will ever use the application.

This is not a isolated case in the in the development process of these two sites, the Blossom hill site also opted for a music player that turned out to be ill fitting the needs of the client, in this case what was needed was a very basic mp3 player but as we suspect the availability of this kind of plug-in is limited by concerns of piracy and music copying. This was also a very labor intensive and time consuming part of the development that ended in a finished working player, but was so difficult to update that it would not have fitted the clients need.

It is clear that technical difficulties always rise in these kinds of project but it should also be noted that we could side step many problems if we could set realistic goals for a project and communicate with the costumer what is essential for their project and what is not, this comes down to only adding features that support the client's needs and not committing resources to length testing phases that usual led nowhere.

Another recurring problem is that even if the client wants sites deploy in a timely manner and this is stressed in the beginning of the development phase, the usual reason for setbacks in the schedule is the client's inability to produce material in a timely manner. We will concede that this is beside the point as this thesis mainly focuses on problems with designing with Wordpress and basic web design.

One of the biggest setbacks in the examples we used to illustrate the work process was a last minute change in the design of the Gerocompetence graphic profile, it was based on the old design of the cover organizations site, the question to change the design I timely manner was communicated to the client but the client insisted on keeping the old design, this led to a last minute change, that was noted in the earlier chapters as a positive because it was easy to correct but this again led to a setback in the timetable.

If we summarize the positives and the negatives we come to the conclusion that most of the problems that arise in the development process are usually features that don't serve a well-defined role in the design of the project. These lead to setbacks in the time table, these should be issues that could be corrected with a well-defined development process, here we do come to another conclusion, that is that we could utilize the Waterfall model but it serves us better in the beginning of the of the development cycle and we do not need to utilize it at the end of the cycle. This was well illustrated in defining the design goals for the Gerocompetence project.

The positives that stick out is that after an initial learning period and by deploying tools to help with the structuring of Wordpress it is an effective tool to deploy sites with, the flexibility that it offered in terms of design and the capabilities in extending it where well suited for developing sites at a good pace. Some of the features that proved most useful where the plug-in as we could experiment with different configurations but the plug-ins also in some cases caused problem, as was explained earlier. As a way of summarizing the technology was never the problem factor or a bigger deterrent it was usually human error in the design decision stat halted the workflow the most.

3.4.3 Optimizing the workflow

Under the process of studying the basics of working with Wordpress we could conclude that there are many approaches in the way we chose to develop with Wordpress, the key factor is a basic knowledge of the structure that enables us to troubleshoot problems that arise in the design process effectively. It must be noted that that this paper covers basics of the building blocks of a Wordpress site and the Theme that controls it, but I should give us an idea of the underlying functionality if we are for example modifying an existing theme or building one from scratch.

What we settled on to as our optimal way of working was using a theme generator to make the basic layout and the doing the necessary custom changes in the code by hand. This enabled us to create demo pages in a rapid pace for testing, from these demos we could directly move to shape the final sites.

Using these tools we could in the end deploy fully realized sites in a comfortable time frame and, have the necessary margins if drastic design changes were needed. In this aspect we think that we greatly optimized our workflow with Wordpress but it should be noted that people should feel to apply any of the steps to customize their own work pace and hopefully this will help in realizing that.

3.4.4 Summary

Hopefully this thesis will help setting up a good basic knowledge of how Wordpress works and provide the developer with the tools to develop efficiently on the Wordpress platform, take note that we only cover basic ground in our examples and that there are more advanced ways of doing thing in Wordpress. This is an introduction to an efficient way of setting up a development routine utilizing Wordpress and in the end we could see that usually what was holding up the development process was not lack of technical knowledge was holding us back but the human factor that is percent in the developer customer dynamic. Next we will look at the sites that where created.

4 CONCLUSION

The process of developing sites using Wordpress has a definite learning curve in understanding the underlying logic of how it works but as discussed in other chapters it a solid platform that saves time when developing and enables a developer to offer a comprehensive package to a prospective client.

Working with Wordpress is quite intuitive as it is such a broadly utilized platform there is a lot of well documented material to help us overcome design problems, usually we can create a basic theme quite easily as there are powerful tools out there to help with creating basic constructs, these can be extremely handy in eliminating time consuming steps from the design process. But we should also remember that these are only tools, take for example Artisteer that is a powerful tool in creating Wordpress themes, we can produce themes with this program without basically understanding nothing about XHTML, CSS and how Wordpress uses these, but if we want to create unique sites that must support different clients needs we can rely on a generator. This also saved a considerable amount of time in the design process; we can summarize that it at the minimum halved the time that was needed to produce the framework of the site.

One of the great strengths when working with Wordpress is the plug-ins that we can create or download to extend the capability of the construct. In the second example much of the usability of the site relies on well-placed plug-in that help order the site, for example the gallery plug-in. The main thing is that they enable us to explore different configurations and by the trial and error method examine what benefits the site and what does not.

The results of these example pages have been very positive, as we can slightly move away from coding everything by hand and when we can utilize the modular design of Wordpress, we can concentrate more on finding unique ways of doing certain things in the design phase or examine custom made components that will suit a clients specialized needs.

As we now have looked at the theory of web design and the design of two sites utilizing Wordpress and broken Wordpress down to its key components we can summarize.

We found out that a rigid model used in the web design does not work well and that it can be replaced with the more dynamic “ripple model”, that takes into account sudden changes and merging needs of the website better. This model also had all the necessary features that we set ourselves in finding the optimal web design strategy.

When we built the site as is explained in the past chapters we could utilize effectively the nonlinear way of design that was modeled in the ripple model, this reduced completion times and enabled us to do fast design changes, Wordpress proved itself to be an optimal platform to develop on tanks to its modular design.

The Wordpress development software Artisteer also worked well, its halved the time to setup pages and gave us more time to finalize the sites.

We recommend when developing with Wordpress to use Artisteer to reduce development times and to use a nonlinear “ripple model” in a configuration that is optimal to the individual developer. This paper should enable the reader to execute Wordpress sites of high standard in short developments times without sacrificing quality.

As we now have looked at the theory of web design and the design of two sites utilizing Wordpress and broken Wordpress down to its key components we can summarize.

We found out that a rigid model used in the web design does not work well and that it can be replaced with the more dynamic “ripple model”, that takes into account sudden changes and merging needs of the website better. This model also had all the necessary features that we set ourselves in finding the optimal web design strategy.

When we built the site as is explained in the past chapters we could utilize effectively the nonlinear way of design that was modeled in the ripple model, this reduced completion times and enabled us to do fast design changes, Wordpress proved itself to be an optimal platform to develop on tanks to its modular design.

The Wordpress development software Artisteer also worked well, its halved the time to setup pages and gave us more time to finalize the sites.

We recommend when developing with Wordpress to use Artisteer to reduce development times and to use a nonlinear “ripple model” in a configuration that is optimal to

the individual developer. This paper should enable the reader to execute Wordpress sites of high standard in short development times without sacrificing quality.

5 REFERENCES

McConnell, Steve 1996. Rapid Development: Taming Wild Software Schedules. Microsoft Press. ISBN 1-55615-900-5.

A rational design process: how and why to fake it: David Parnas Paul c. Clements Accessed May 2011 http://users.ece.utexas.edu/~perry/education/SE-Intro/fakeit.pdf

Sabin-Wilson, Lisa 2010. Wordpress for Dummies. Wiley Publishing. ISBN 9780470592748

Joens, Colleen 2011. Clout the art and Science of Influential web content. New Riders. ISBN 9780321733016

Richard Muscat 2009. Using WordPress to Build Small Websites:33 Step by Step Tutorial; Accessed May 2011 http://richardmuscat.wordpress.com/2009/03/25/using-wordpress-as-a-cms-step-by-step-tutorial/

Mike Johnston 2009. "What is a CMS?" Accessed May 2011. http://www.cmscritic.com/what-is-a-cms/

Php.net Accessed May 2011 http://www.php.net/

Mysql.com Accessed May 2011 http://www.mysql.com/

"History of MySQL, MySQL 5.1 Reference Manual". MySQL AB. Accessed May 2011 http://dev.mysql.com/doc/refman/5.1/en/history.html

Apachefriends.org Accessed 2011 May http://www.apachefriends.org/en/xampp.html

W3 Schools 2011 Accessed May 2011 http://www.w3schools.com/css/default.asp http://www.w3schools.com/html/default.asp

smashingmagazine.com

http://uxdesign.smashingmagazine.com/2008/11/05/strategic-design-6-steps-for-building-successful-websites/

http://www.smashingmagazine.com/2008/08/05/7-essential-guidelines-for-functional-design/

Leovolz.com 2006 Accessed May 2011 http://www.levoltz.com/2006/09/17/how-to-create-a-wordpress-template-in-5-minutes/

Wpdesigner.com 2007, Accessed May 2011
http://www.wpdesigner.com/2007/02/19/so-you-want-to-create-wordpress-themes-huh/

Wordpress.org Accessed May 2011.http://wordpress.org/about/

http://codex.wordpress.org 2011 following articles accessed;

http://wordpress.org/about/

http://codex.wordpress.org/Installing_WordPress

http://codex.wordpress.org/First_Steps_With_WordPress

http://codex.wordpress.org/Site_Architecture_1.5

http://codex.wordpress.org/Theme_Development

http://codex.wordpress.org/Templateshttp://codex.wordpress.org/The_Loophttp://codex.wordpress.org/Template_Hierarchy

http://codex.wordpress.org/Child_Themes

http://codex.wordpress.org/Plugins

http://codex.wordpress.org/Widgets_API

http://codex.wordpress.org/Hardening_WordPress

THE MANUAL

6 THE MANUAL

6.1 DEFINITIONS

The amount of information available today is almost infinite. Finding information is no longer a problem. Instead it has become more important to be able to “curate” the useful bits and discard the parts that are redundant for our work. What follows is an edited appendix of current information that includes basic structure and instructions how to use it, with some tutorials to explain some of the more advanced features of Wordpress.

This is intended to form a manual that brings explanations of the key components of Wordpress functions into one place. It has been compiled from a variety of source material but edited together to provide detailed and structured tutorials while not compromising the integrity of the original material.

6.1.1 WEB CONTENT MANAGEMENT SYSTEM (WCMS)

WCMS is the name given to database driven framework that enables publishing of webpage’s that contain a data base and usually a graphic interface in the backend of the site to enable easy updating and sorting of the information that the site contains. Wordpress is one of the most commonly used WCMS and this thesis studies it exclusively. For a more detailed explanation of the core concepts please refer to the codex.

6.1.2 PHP “HYPERTEXT PROCESSOR”

PHP can be found on 50-70 percent of all web servers and Wordpress utilizes it extensively. PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages.

For a more detailed explanation of the core concepts please refer to the codex.

6.1.3 MYSQL “STRUCTURED QUERY LANGUAGE”

MySQL is an open source management system (RDBMS) that runs as a server providing multi-user access to a number of databases. Sites that utilize databases heavily often use MySQL and these include big names as Google, Wikipedia and Facebook. The SQL phrase stands for Structured Query Language.

For a more detailed explanation of the core concepts please refer to the codex.

6.1.4 WORDPRESS

Wordpress is a CMS system that enables the creation of database driven dynamic webpage’s. It originally started as a platform mainly used for blogging but has expanded since that to be a fully fledged platform for publishing webpage’s. For a more detailed explanation of the core concepts please refer to the codex.

6.1.5 BASICS OF WORDPRESS

The core concepts of Wordpress is its WCMS powered web platform with a modular design that enables the building of simple one pages blogs to 300 page sites with streaming media and photo galleries or a web shop. This is enabled by the modular design of Wordpress as it can be expanded by the use of plug-ins, themes, and child themes to meet our design need. For a more detailed explanation of the core concepts please refer to the codex.

6.1.6 SETTING UP WORDPRESS

What needs to be known about Wordpress when setting it up is that it comes with a ready-made content management system that you can modify by using standard CSS(cascading style sheet) and html (hyper text markup language). But as Wordpress is a PHP driven system it need a MySQL database to work properly. This is also a good point to remember when registering a web hotel is that it needs to support Wordpress. This is the bare minimum of information that is needed to get Wordpress working. For a more detailed explanation of the core concepts please refer to the codex.

6.1.7 THE STRUCTURE OF WORDPRESS

To understand the structure of Wordpress its god to remember that it a composite of many interlocking pieces that work together to form the platform that is Wordpress. A Wordpress site is controlled by templates and the templates are controlled by the Theme of the page. There are also child themes. The core of the site architecture is built on XHTML tags and CSS selectors. For a more detailed explanation of the core concepts please refer to the codex.

6.1.8 INTERIOR STRUCTURE

Within the core structure there are containers that are smaller building blocks that hold the specific content within the parent containers. Wordpress Themes can feature a variety of these, but we are going to concentrate on the two Themes that come with Wordpress. Most Wordpress Themes are based on these two Themes. For a more detailed explanation of the core concepts please refer to the codex.

6.1.9 WORDPRESS TEMPLATE

Template files are the building blocks of your Wordpress site. They fit together like the pieces of a puzzle to generate the web pages on your site. Some templates (the header and footer template files for example) are used on all the web pages, while others are used only under specific conditions. For a more detailed explanation of the core concepts please refer to the codex.

6.1.10 TEMPLATE FILE HIERARCHY

As a short example of how the hierarchy works in Wordpress, here is an example. When we want Wordpress to do something, Wordpress matches every Query String to query types i.e. it decides what type of page (a search page, a category page, the home page etc.) is being requested. Templates are then chosen and web page content is generated in the order suggested by the Wordpress Template hierarchy, depending upon what templates are available in a particular Wordpress Theme. For a more detailed explanation of the core concepts please refer to the codex.

6.1.11 THE LOOP

The Loop is what Wordpress uses to display its posts and it important to have a working understanding of the basics. "The Loop" is a term that refers to the main process of Wordpress. We use The Loop in our template files to show posts to visitors. It is possible to make templates without The Loop, but it would only be able to display data from one post. For a more detailed explanation of the core concepts please refer to the codex.

6.1.12 CSS IN WORDPRESS

Wordpress relies on the presentation styles within CSS. With the use of Themes, we have an almost infinite choice of layout options. Wordpress Themes make it easy to change how the webpage is presented and how it is styled. I enables customization of almost everything that Wordpress uses to style itself. For a more detailed explanation of the core concepts please refer to the codex.

6.1.13 HTML TO XHTML AND WORDPRESS

Wordpress, as a system, is based on documents written in the XHTML scripting language. XHTML 1.0. For a more detailed explanation of the core concepts please refer to the codex.

6.1.14 WORDPRESS THEMES

Wordpress Themes are files that work together to create the design and functionality of a Wordpress site. Each Theme may be different, offering many choices for site owners to take advantage of in order to instantly change their website look. For a more detailed explanation of the core concepts please refer to the codex.

6.1.15 BUILDING THEMES

Building a Wordpress theme is a combination of all the steps in design that we have been studying in previous chapters. The basic tool-set that is primarily needed is a working understanding of CSS, XHTML and the structuring behind Wordpress. For a more detailed explanation of the core concepts please refer to the codex.

6.1.16 CHILD THEMES

Child themes are the best way to modify existing themes in Wordpress, the purpose of the parent/child WordPress Theme feature is to protect the original WordPress Theme from design changes, allowing the user to change the child Theme without impacting the original code, which makes upgrading Themes easier. For a more detailed explanation of the core concepts please refer to the codex.

6.1.17 WORDPRESS PLUGINS

One of the biggest features that Wordpress has incorporated is the modular design that enables the use of plug-in. These range from custom font options that enable other fonts than the web standard to building community websites with plug-ins enabling forums and chat boards. For a more detailed explanation of the core concepts please refer to the codex.

6.1.18 WIDGETS IN WORDPRESS

A widget is what Wordpress calls a program or function that we can move freely around the Wordpress designated widget areas. For a more detailed explanation of the core concepts please refer to the codex.

6.1.19 BASICS OF BUILDING A WORDPRESS SITE

The basics of building a Wordpress site are very complex or very easy, it can be argued as such. For a more detailed explanation of the core concepts please refer to the codex.

6.1.20 WORDPRESS SECURITY

It is good to note that as of 2011 the amount of malicious software on the internet has tripled since 2005 (f-secure study 2011). It is advisable to take this into account when designing in Wordpress and try to implement safety features. For a more detailed explanation of the core concepts please refer to the codex.

6.1.21 ARTISTEER (THEME GENERATOR)

Artisteer is a commercial Theme generator that has the capacity to publish Wordpress, Drupal and Joomla themes in ready-made packages that are install ready. The software has a setup that could be called a marriage of a simplified Microsoft Word and a very bare bones Photoshop. For a more detailed explanation of the core concepts please refer to the codex.

6.1.22 WEB CONTENT MANAGEMENT SYSTEM (WCMS)

To use Wordpress we must understand the basics of what a Web Content Management System is, this will help to grasp the underlying concepts with both the pros and the cons of using WCMS solutions to build our projects. Accessed 2011 May [http://php.opensourcecms.com].

A web content management system (WCMS) is a software system that provides website authoring, collaboration, and administration tools designed to allow users with little knowledge of web programming languages or markup languages to create and manage website content with relative ease. A robust WCMS provides the foundation for collaboration, offering users the ability to manage documents and output for multiple author editing and participation.

Most systems use a database to store page content, metadata, and other information assets that might be needed by the system.

A presentation layer displays the content to website visitors based on a set of templates. The templates are sometimes XSLT files.

Most systems use server side caching to improve performance. This works best when the WCMS is not changed often but visits happen on a regular basis. This is because the server doses not update the site as frequently as it would otherwise to increase performance as it saves the data in faster drives but in smaller pieces.

Administration is typically done through browser-based interfaces, but some systems require the use of a fat client, meaning an administrative program installed on a computer controlling the WCMS.

A WCMS allows non-technical users to make changes to a website with little training. A WCMS typically requires a systems administrator and/or a web developer to set up and add features, but it is primarily a website maintenance tool for non-technical staff.

6.1.22.1 CAPABILITIES

A WCMS is a software system used to control a dynamic collection of Web material (HTML documents, images, and other forms of media). A CMS facilitates document control, auditing, editing, and timeline management.

Automated templates

Create standard output templates (usually HTML and XML) that can be automatically applied to new and existing content, allowing the appearance of all content to be changed from one central place.

Access control

Some WCMS systems support user Groups. User Groups allow you to control how registered users interact with the site. A page on the site can be restricted to one or more groups. This means if an Anonymous User (someone not logged on) or a Logged on User who is not a member of the Group a page is restricted to, the user will be denied access to the page.

Scalable expansion

Available in most modern WCMSs is the ability to expand a single implementation (one installation on one server) across multiple domains, depending on the server's settings. WCMS sites may be able to create microsites/web portals within a main site as well.

Easily editable content

Once content is separated from the visual presentation of a site, it usually becomes much easier and quicker to edit and manipulate. Most WCMS software includes WYSIWYG editing tools allowing non-technical individuals to create and edit content.

Scalable feature sets

Most WCMS software includes plug-ins or modules that can be easily installed to extend an existing site's functionality.

Web standards upgrades

Active WCMS software usually receives regular updates that include new feature sets and keep the system up to current web standards.

Work-flow management

Work-flow is the process of creating cycles of sequential and parallel tasks that must be accomplished in the CMS. For example, one or many content creators can submit a story, but it is not published until the copy editor cleans it up and the editor-in-chief approves it.

Collaboration

CMS software may act as a Collaboration platform allowing content to be retrieved and worked on by one or many authorized users. Changes can be tracked and authorized for publication or or ignored reverting to old versions. Other advanced forms of collaboration allow multiple users to modify (or comment) a page at the same time in a collaboration session.

Delegation

Some CMS software allows for various user groups to have limited privileges over specific content on the website, spreading out the responsibility of content management.

Document management

CMS software may provide a means of collaboratively managing the life cycle of a document from initial creation time, through revisions, publication, archive, and document destruction.

Content virtualization

CMS software may provide a means of allowing each user to work within a virtual copy of the entire Web site, document set, and/or code base. This enables changes to multiple interdependent resources to be viewed and/or executed in-context prior to submission.

Content syndication

CMS software often assists in content distribution by generating RSS and Atom data feeds to other systems. They may also e-mail users when updates are available as part of the workflow process.

Multilingual

Ability to display content in multiple languages.

Versioning

Like Document Management Systems CMS software may allow the process of versioning by which pages are checked in or out of the WCMS, allowing authorized editors to retrieve previous versions and to continue work from a selected point. Versioning is useful for content that changes over time and requires updating, but it may be necessary to go back to or reference a previous copy.

These are the set of features that are commonly integrated into WCMS systems and are there to help standardize and control the page and to conform it to specified standards set by the designer.

6.1.22.2 ADVANTAGES AND DISADVANTAGES OF CMS

As with any system there are things to consider when choosing a platform for your project. Here are some of issues that can help or hinder development with CMS.

One of the advantages is the cost efficiency of CMS as many of the applications are free as for example Joomla, Drupal and Wordpress that this thesis covers.

These kinds of systems would be extremely time consuming and expensive to build but as they are available under the creative commons license they are free to be implemented into your projects.

Some of the other advantages are the user friendly update features that are commonly made with Ajax that enables drag and drop customization of web-pages and a easy interface for user without coding language skills.

The underlying themes of WCMS it that you have a constant well regulated system that controls our content, and with enables the user to publish material in a very controlled and regulated manner with full control.

The biggest concerns with CMS systems is that the cost of maintenance can be quite high in large systems and the maintenance cost can be large, but as this is a more of a problem with large WCMS. Large scale WCMS can requite specialized hardware and staff to keep it up to date and this is of course an issue if we star developing at this scale. Specialized permits are also a norm at large-scale network building

Larger CMSs can also experience latency if the hardware infrastructure is not up to date, if databases are not being utilized correctly, and if cache files that have to be reloaded every time data is updated grow large. Load balancing issues may also impair caching files.

These are what WCMS offers as a development platform, and Wordpress contains all these features to some extent, in later chapters Wordpress and its different components will be examined and picked apart for further study.

6.1.23 PHP "HYPERTEXT PROCESSOR"

As a PHP can be found on 50-70 percent of all web servers and that Wordpress utilizes it extensively we will look at the basics of PHP. Accessed 2011 May [www.php.net].

PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. 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 as a standalone interpreter, on almost every operating system and platform free of charge. PHP is installed on more than 20 million websites and 1 million web servers.

PHP was originally created by Rasmus Lerdorf in 1995. The main implementation of PHP is now produced by The PHP Group and serves as the de facto standard for PHP as there is no formal specification. PHP is free software released under the PHP License; it 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", a recursive acronym.

6.1.24 THE USES OF PHP

PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on web sites or elsewhere. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

PHP primarily acts as a filter, taking input from a file or stream containing text and/or PHP instructions and outputs another stream of data; most commonly the output will be HTML. Since PHP 4, the PHP parser compiles input to produce bytecode for processing by the Zend Engine, giving improved performance over its interpreter predecessor.

Originally designed to create dynamic web pages, PHP now focuses mainly on server-side scripting, and it is similar to other server-side scripting languages that provide dynamic content from a web server to a client, such as Microsoft's Asp.net, Sun Microsystems' JavaServer Pages, and mod_perl. PHP has also attracted the development of many frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these include CakePHP, Symfony, CodeIgniter, and Zend Framework, offering features similar to other web application frameworks.

The LAMP architecture has become popular in the web industry as a way of deploying web applications. PHP is commonly used as the P in this bundle alongside Linux, Apache and MySQL, although the P may also refer to Python or Perl or some combination of the three. WAMP packages (Windows/ Apache/ MySQL / PHP) and MAMP packages (Macintosh / Apache / MySQL / PHP) are also available.

As of April 2007, over 20 million Internet domains had web services hosted on servers with PHP installed and mod_php was recorded as the most popular Apache HTTP Server module. PHP is used as the server-side programming language on 75% of all web servers. Web content management systems written in PHP include MediaWiki, Joomla, eZ Publish, Wordpress, Drupal and Moodle. All websites created using these tools are written in PHP, including the user-facing portion of Wikipedia, Facebook.

6.1.25 SECURITY ISSUES

The National Vulnerability Database maintains a list of vulnerabilities found in computer software. The overall proportion of PHP-related vulnerabilities on the database amounted to: 20% in 2004, 28% in 2005, 43% in 2006, 36% in 2007, 35% in 2008, and 30% in 2009. Most of these PHP-related vulnerabilities can be exploited remotely: they allow attackers to steal or destroy data from data sources linked to the web server (such as an SQL database), send spam or contribute to DoS attacks using malware, which itself can be installed on the vulnerable servers.

These vulnerabilities are caused mostly by not following best practice programming rules: technical security flaws of the language itself or of its core libraries are not frequent (23 in 2008, about 1% of the total). Recognizing that programmers cannot be trusted, some languages include taint checking to detect automatically the lack of input validation which induces many issues. Such a feature is being developed for PHP, but its inclusion in a release has been rejected several times in the past.

Hosting PHP applications on a server requires careful and constant attention to deal with these security risks

6.1.26 SUMMARY

PHP is used in many different configurations and is mainly used to process information and, or convert it to different formats, this is basically what we need to know about PHP as it runs in the background of Wordpress, but we can learn from some of the features and the inherit properties of PHP, and as a final point it’s good to consider the security issues of PHP

6.1.27 MYSQL “STRUCTURED QUERY LANGUAGE”

MySQL started development in Finland 1995 by Michael Widenius and David Axmark and the first version was released in 1996.It is named after developer Michael Widenius' daughter My. . Accessed 2011 May [HTTP://miscalculation].

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: Joomla, WordPress, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google (though not for searches) and Facebook.

6.1.28 DEPLOYMENT

MySQL can be built and installed manually from source code, but this can be tedious so it is more commonly installed from a binary package unless special customizations are required. On most Linux distributions the package management system can download and install MySQL with minimal effort, though further configuration is often required to adjust security and optimization settings.

Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well.

It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP based web application or as a standalone database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open source tools such as phpMyAdmin.

In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory.

There are however limits to how far performance can scale on a single server, so on larger scales, and multi-server MySQL deployments are required to provide improved performance and reliability. A typical high-end configuration can include a powerful master database which handles data write operations and is replicated to multiple slaves that handle all read operations. The master server synchronizes continually with its slaves so in the event of failure a slave can be promoted to become the new master, minimizing downtime. Further improvements in performance can be achieved by caching the results from database queries in memory using memcached, or breaking down a database into smaller chunks called shards which can be spread across a number of distributed server clusters.

6.1.29 KEY FEATURES AND DISTINGUISHING FEATURES IN MYSQL

As of April 2009, MySQL offered MySQL 5.1 in two different variants: the open source MySQL Community Server and the commercial Enterprise Server. MySQL 5.5 is offered under the same licenses. They have a common code base and include the following features:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures

- Triggers
- Cursors
- Updateable Views
- True Varchar support
- Information schema
- Strict mode
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using Oracle's InnoDB engine
- Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, MySQL Archive for storing historical data in little space)
- Transactions with the InnoDB, BDB and Cluster storage engines; save-points with InnoDB
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master, no automatic support for multiple masters per slave.
- Full-text indexing and searching using MyISAM engine
- Embedded database library
- Partial Unicode support (UTF-8 and UCS-2 encoded strings are limited to the BMP)
- Partial ACID compliance (full compliance only when using the non-default storage engines InnoDB, BDB and Cluster)

- Partitioned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Hot backup (via mysqlhotcopy) under certain conditions

The developers release monthly versions of the MySQL Server. The sources can be obtained from MySQL's web site or from MySQL's Bazaar repository, both under the GPL license.

MySQL implements the following features, which some other RDBMS systems may not:

- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application (in MySQL 5.0, storage engines must be compiled in; in MySQL 5.1, storage engines can be dynamically loaded at run time):

o Native storage engines (MyISAM, Falcon, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, Cluster, Berkeley DB, EXAMPLE, Maria, and InnoDB, which was made the default as of 5.5)

o Partner-developed storage engines (solidDB, NitroEDB, Infobright (formerly Brighthouse), Kickfire, XtraDB, IBM DB2). InnoDB used to be a partner-developed storage engine, but with recent acquisitions, Oracle now owns both MySQL core and InnoDB.

o Community-developed storage engines (memcache engine, httpd, PBXT, Revision Engine)

o Custom storage engines

- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

6.1.30 SUMMARY

Wordpress is built extensively on MySQL and uses it as its primary database. This enables the user to have a great control over the databases and be able to safeguard them and to backup information in a safe and flexible way.

The information that was provided here is primarily the give us a good working understanding of what MySQL provide when we look at hosting and using databases in our web design projects.

6.2 WORDPRESS

Wordpress is an open source Content Management System (CMS), often used as a blog publishing application, powered by PHP and MySQL. It has many features including a plug-in architecture and a template system. WordPress is used by over 13% of the 1,000,000 biggest websites. Accessed 2011 May [http://codex.wordpress.org].

It was first released on May 27, 2003, by Matt Mullenweg as a fork of b2/cafelog. As of February 2011, version 3.1 had been downloaded over 32.5 million times. What began as blogging software has transcended into mainstream web publishing platform.

b2/cafelog, more commonly known as simply b2 or cafelog, was the precursor to Wordpress. b2/cafelog was estimated to have been employed on approximately 2,000 blogs as of May 2003. It was written in PHP for use with MySQL by Michel Valdrighi, who is now a contributing developer to Wordpress.

Wordpress first appeared in 2003 as a joint effort between Matt Mullenweg and Mike Little to create a fork of b2. The name Wordpress was suggested by Christine Selleck, a friend of Mullenweg.

In 2004 the licensing terms for the competing Movable Type package were changed by Six Apart and many of its most influential users migrated to Wordpress. By October, 2009, the 2009 Open Source content management system Market Share Report reached the conclusion that Wordpress enjoyed the greatest brand strength of any open source content management systems.

Wordpress is still considered by some a platform only viable for blog us and that is to a degree right in the way that it the most popular platform on the internet now I we would launch a blog or blogs concerning some issue. What must be remembered is that Wordpress these days is what you design it to be. In the following chapters we will look at the key concepts and architecture that we need to grasp to design using Wordpress. We will look at structure, design and building a site, and see what can be used to make us design better pages and work in a well structured way.

6.2.1 BASICS OF WORDPRESS

At its most basic level what is needed to run Wordpress is a virtual server or web hosting service that has a (at the moment of writing) PHP version 5.2.4 or greater and MySQL version 5.0.15 or greater. These are specified as the running requirements of June 30 2011 Accessed 2011 May [http://codex.wordpress.org].

Many service providers in web-hosting offer one click installation options for Wordpress and when that is done we are presented with the default Wordpress installation and the Starting theme.

The staring theme of Wordpress 3.1.2 is called Twenty Ten as is a highly customizable theme that includes custom menu, header image, and background. Twenty Ten supports six widgetized areas (two in the sidebar, four in the footer) and featured images (thumbnails for gallery posts and custom header images for posts and pages). It includes style sheets for print and the admin Visual Editor, special styles for posts in the "Asides" and "Gallery" categories, and has an optional one-column page template that removes the sidebar.

If needed this default theme could be extended further by the use of plug-ins and widgets but as this is an introduction we will concentrate on the default settings of Wordpress to familiarizes us with basic functionality and settings.

You control the site by using the dashboard feature in Wordpress, which incorporates drag and drop features and other user friendly applications that do not require knowledge in a scripting language.

Here is a rudimentary tutorial that takes us through a very basic setup of the Twenty Ten theme. Accessed 2011 May[http://www.blogtowkay.com/setup-internet-business/how-to-customize-my-wordpress-twenty-ten-theme-basic.html]

6.2.1.1 TUTORIAL

Basic installation allows the users to setup custom headers, develop complicated menu structures, modify the sidebar and footer areas and change the background color easily. This is also a SEO-friendly theme that looks professional. Changing and installing new themes in Wordpress

If we should feel that we want to use another theme we can go to (Menu: Appearance > Themes). On the Manage Themes page, you could see all the themes which you have ever installed into your Wordpress, and which theme is currently active.

Go over to Install Themes page to search for all the latest themes. It is possible to refine your selection by searching for the featured, newest or recently updated themes. It is possible to also refine your selection by using the feature filter to search for themes of certain colors, columns etc. Once we find a theme to our liking, It is possible to either preview the theme or install it into your Wordpress. Upon successful installation, It is possible to decide to activate it for use immediately, or to do it at another time.

6.2.1.2 HOW TO CUSTOMIZE THE DEFAULT TWENTY TEN THEME IN WORDPRESS

As we are using the Twenty Ten Wordpress theme, we should familiarize ourselves with the layout and functionality of this theme. There are different areas such as the Header, Menu, and Widget, Background and Post / Page areas that can be seen in the picture blow.

6.2.1.3 TWENTY TEN THEME WIDGETS

The widget area is highly customizable, it contains two sidebar widget areas and four footer widget areas (Menu: Appearance > Widgets). On the Widgets page, the widget areas which you can customize are found on the right side of the page. The main area on the left contains all the widgets which are installed with your theme. We can drag any of the widgets from the main area to the widget areas.

Some of the useful widgets which might be worth placing in the widget areas include the Recent Posts and Recent Comments widgets.

Another very useful widget is the Text widget, which offers the more advanced users a blank area to paste customized HTML codes or Java Scripts.

For widgets which are no longer useful, they can be dragged to the Inactive Widgets area at the bottom of the Widgets page instead of deleting those widgets. This will preserve any customization or settings and in the future they can easily be recovered if we decide to reactivate these widgets subsequently.

6.2.1.4 TWENTY TEN THEME CUSTOM MENUS

The custom Menus page is a very powerful addition to this default theme (Menu: Appearance > Menus). Previously, this feature is mostly found only in premium (paid) themes or the really good free themes. Once we have created a new menu name, the Twenty Ten theme allows you to place a mix of pages, categories, or custom links to any web-page in any order you prefer.

6.2.1.5 TWENTY TEN THEME CUSTOM BACKGROUND

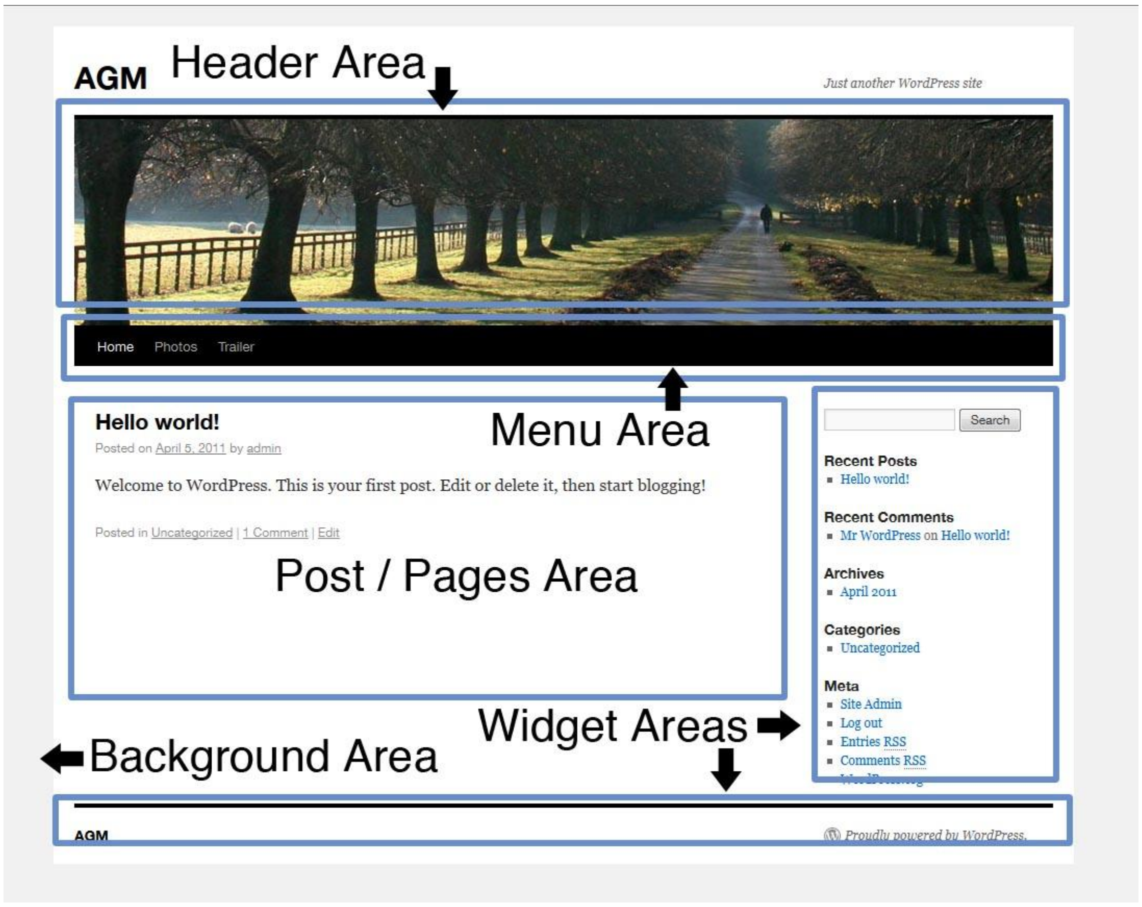
The default background of the theme is a light shade of grey. We can use the custom Background page (Menu: Appearance > Background) to change the color of this background to blend into the preferred color theme of your blog using the color wheel. Alternatively choose a really nice and relevant image to upload it as part of the background.

6.2.1.6 TWENTY TEN THEME CUSTOM HEADER

There are a total of 8 default Header pictures that can be selected for our blog on the custom Header page (Menu: Appearance > Header). These nice pictures were actually taken and contributed by Matt Mullenweg, the founding developer of Wordpress.

However it would be strongly recommended that we use images which are relevant to your sites theme. Upload your images after we have used any picture editing software to crop the images to exactly 940 × 198 pixels. Alternatively, simply upload pictures through the custom Header page and Wordpress will give you the option to crop the image online.

This will leave us with a basic a unique site that is customized for our needs and can be used if there is a need to get a web presence fast. This is the most basic of setup that we can launch using Wordpress.



Hello world!

Posted on April 5, 2011 by admin

Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!

Posted in Uncategorized | 1 Comment | Edit

Twenty ten theme layout figure 3

6.2.2 SETTING UP WORDPRESS

What needs to be known about Wordpress when setting it up is that it comes with a ready-made content management system that you can modify by using standard CSS(cascading style sheet) and html (hyper text markup language). But as Wordpress is a PHP driven system it need a MySQL database to work properly. This is also a good point to remember when registering a web hotel is that it needs to support Wordpress.

When we are setting up a good developing routine it is good to note that what feels good for some people might not be acceptable to others, with this as our starting point we will try to setup a structure for developing Wordpress efficiently and safely with implemented backup features to protect work that has been done from data lose and also present options on how to make the workflow in some ways transparent to the client and in this way improving communication.

A good setup is to have a virtual server installed on a work computer and have a separate online server to publish on. This enable that web connectivity issues are never a problem when developing and enables fast testing of different design concepts. From the virtual server it is then possible to move a sufficiently ready version of a project online that again enables clients to have a prototype to evaluate and test. Also in the occurrence of data loss we have a double back up in place. When setting up a Wordpress compatible server (usually virtual) or renting one you should take into account that we will need PHP version 4.3 or greater MySQL version 4.1.2 or greater to be included in the server setup. When setting up a virtual server we use for example when developing on a personal computer the program XAMPP. XANPP is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.Accessed 2011 May [http://www.apachefriends.org/en/xampp.html].

There is other similar programs as well that do the same ting, for example WAMP and if developing for Apple MAMP. These programs are mainly made so that it possible to develop and test different WCMS sites locally without the need of publishing them online. The virtual host also usually has a lot of the security features disabled because to work being done is made in a controlled and safe environment.

The other part is getting a Wordpress compatible web-hosting service; here what we use is dictated by what kind of bandwidth, storage space and services the hosting service provides. In the example sites that are built parallel with this thesis are hosted under the service provider Hostmonster.com, which for example offers unlimited sub domains that is good when developing multiple sites simultaneously as we can have different sites online and test there usability and demonstrate them to customers in real-time. The other is unlimited bandwidth that can be a good thing if sites start gathering a lot of hits per day, and a easy way of registering domain names.

The optimal setup is a combination of both virtual and online hosting for projects as this gives us a very flexible work environment and enables developing both online and offline. Here is a tutorial on how to setup a good work environment.

6.2.2.1 SETTING UP TUTORIAL

Here is an example of a development structure. We will investigate how we can setup a copy of your online Wordpress powered site on a local machine. This will serve as a development and test environment. Here is a tutorial for it:

6.2.2.2 PREREQUISITES

The following tools are needed:

- Wordpress Backup Plug-in (included by default as part of Wordpress)
- XAMPP (if you are using Windows) or MAMP (if you are using OS X)
- phpMyAdmin (included in XAMPP and MAMP)

6.2.2.3 BACKUP YOUR SITE

Backing up any Wordpress site is a two step process.

The first step involves backing up the database that Wordpress uses and the second step involves backing up the Wordpress core files, your themes and plug-in from the remote server.

Wordpress includes a plug-in for automating the process of backing up the MySQL database. The plug-in can be accessed from the admin section (Dashboard), under Manage > Backup. The plug-in automatically selects the core tables but since third-party plug-ins may use extra tables, a list is displayed on the right-hand side where you can select additional tables. As a rule of thumb, any table with the Wordpress prefix (default value ‘WP_’) should be selected.

Select whether you want to email the backup to yourself or download it directly to your local machine and press the Backup! button. When the operation is finished you have made a backup of the Wordpress database.

In order to backup the files, you need to use an FTP application like WinSCP if you are running Windows or CyberDuck if you are running Mac OS X.

Start your FTP application and connect to your Wordpress server. Navigate to the wp-contents directory of your remote server and download all the files and directories it contains to your local computer.

6.2.2.4 INSTALL LOCAL WEB SERVER

We will need to install a web server on your local computer. The easiest approach is to download and install an Apache web server containing MySQL, PHP. XAMPP is the best product to use if you are running Microsoft Windows. If you are running Mac OS X then MAMP is your friend. Both of them are really very easy to install and to use – just download, extract and start.

Note that both of them create a root account for MySQL with blank password or with the word “root”. It is highly recommended to change the password to something else and use that password when installing WordPress.

phpMyAdmin is also installed as part of XAMPP and MAMP. 99.99% of the times, it resides on the phpmyadmin directory (or an alias is in place for this location). Open it up, and create a database for the Wordpress development site.

6.2.2.5 INSTALL WORDPRESS

Download a copy of Wordpress from the WP site. Extract it into the httpdocs or www folder of your XAMPP installation. Navigate to that directory through your browser and complete the Wordpress installation process.

6.2.2.6 IMPORT YOUR SETTINGS

Our first step was to backup the WP database. We will now use the produced MySQL script to populate the backup WP system with our site settings.

Open the MySQL backup script with your favorite text editor, and search for the term ‘wp_options’. That’s the table that holds your settings. We need to remove the ‘siteurl’ entry since it is now different on the local server (and the correct entry for ‘siteurl’ was created during the Wordpress installation for us). So, delete the line that reads something like:

INSERT INTO `wp_options` VALUES (1, 0, 'siteurl', 'Y', 1, 'http://www.dountsis.com', 20, 8, 'Wordpress web address', 1, 'yes') ;

Save the changes and close the file. Navigate to phpMyAdmin and select the database where you have installed the development WP site. Execute the database backup script to import the dev system with your content and settings.

Finally, copy the local copy of the Wordpress Files backup (themes and plug-in) that was created during the 1st step to the equivalent location of the development site.

The development/test environment should now be an identical copy of the live site where you can test new themes and/or plug-in.

6.2.3 Setting up Hardware suited for Web Development

A good computer monitor is also advisable when developing as this will be an essential tool, we would recommend a good quality 24 inch screen with a resolution of 1920x1200 with an aspect ratio of 16:10, this enables us to work comfortably and see the design on the screen as it is without scrolling. As web developments is not as dependent on processing power the computer of choice does not have to be a very fast.

6.3 THE STRUCTURE OF WORDPRESS

To understand the structure of Wordpress its god to remember that it a composite of many interlocking pieces that work together to form the platform that is Wordpress Accessed 2011 May [http://codex.wordpress.org].

A Wordpress site is controlled by templates and the templates are controlled by the Theme of the page. There are also child themes that expand on this concept but that will be discussed in later chapters. What you see when looking Wordpress site is a collection of templates that all serve a purpose. The core of the site architecture is built on XHTML tags and CSS selectors, Wordpress encourages the developers not to change these, but it is optional.

- index.php
- header.php

- sidebar.php
- footer.php

When you view a single post page, you might be viewing the following template files:

- single.php
- header.php
- sidebar.php
- footer.php
- comments.php

On a multi-post page like categories, archives, and search, you might be viewing any combination of the following template files:

- index.php
- category.php
- 404.php
- search.php
- header.php
- sidebar.php
- footer.php

We've specified which CSS selectors belong in which template files as much as possible in the following architecture specifications.

6.3.1 Core structure of Wordpress

The core structure of a Wordpress site represents the main containers which hold the page's content. Accessed 2011 May [http://codex.wordpress.org]. These are the bare necessities that a Wordpress is constructed of. The core structure of a Wordpress site features, at a minimum, are:

- Header
- Sidebar/Menu
- Content
- Footer

These are the main containers in which the most important parts of the page are "contained". Remember, the core structure is like building blocks. They are dependent upon each other. If you change one, you have to change the others. This can lead to a domino effect where changes even small ones can cascade to create errors.

6.3.1.1 CLASSIC THEME

```
<body>

<div id="rap">

<h1 id="header"></h1>

<div id="content"></div>

<div id="menu"></div>

<p class="credit"></p>

</div>

</body>
```

6.3.1.2 DEFAULT THEME

```
<body>

<div id="page">

<div id="header"></div>

<div id="content" class="narrowcolumn"></div>
```

```
<div id="sidebar"></div>

<div id="footer"></div>

</div><!-- end page -->

</body>
```

While one calls their sidebar and the other menu, the main difference between the two Theme's core structures is the use of the header and footer. For the Classic Theme, the header is in an h1 tag and the footer is in a paragraph tag. In the Default Theme, the header is in a div called header and the footer is in the footer div.

Both Themes feature a container that encompasses or "wraps" itself around the entire page. This wrapping container allows for more definitive control over the overall structure, often used in combination with the body tag. In different Wordpress Themes, these are also found by these names:

- container
- page
- wrap
- rap

Some Themes may add a second, third, or even fourth sidebar, creating a column effect. Or they may include additional wrappers around the entire page or specific containers, but this is the core structure.

6.3.1.3 MODULAR TEMPLATE FILES

Based upon the premise of building blocks, Wordpress Themes divide up the core structure between blocks called template files. These are the template files:

- Header - header.php
- Sidebar/Menu - sidebar.php
- Content - index.php, single.php, page.php, category.php, author.php, search.php, etc.

- Footer - footer.php

As we can see, the content container can be found across many other template files. These are generated dependent upon the user's request. If they click on a category, the category template is displayed. If they choose a Page, the page template is used. And so on.

Combined with the Wordpress Loop (the loop will be cover in a later chapter) and queries, a variety of templates can be generated, and the developer can style all of these differently and independently from each other.

6.3.2 INTERIOR STRUCTURE

Within the core structure there are containers that are smaller building blocks that hold the specific content within the parent containers. Wordpress Themes can feature a variety of these, but we are going to concentrate on the two Themes that come with Wordpress. Most Wordpress Themes are based on these two Themes.

7.1 Header

The header is the structure that traditionally sits at the top of a web page. It contains the title of the website. It may also be referred to as the masthead, head, title, and banner. In all Wordpress Themes, the header is found within the header.php template file.

The Classic Theme features the simplest header code:

```
<h1 id="header"></h1>
```

The Default Theme has a more complex header code:

```
<div id="header">

  <div id="headerimg">

    <h1></h1>

      <div class="description"></div>

  </div>
```

```
</div>
```

While the styles for the Classic Theme are found within the Theme's style.css style sheet file, styles for the Default Theme are found within the style.css and the <head> of the header.php template file. Working with these styles is extensively covered in Designing Headers.

6.3.3 Content

The content container in Wordpress plays the most important role. It holds the Wordpress Loop which dictates the generation of content on the page depending upon the request by the user.

The Classic Theme has the simplest content structure:

```
<div id="content">
  <h2>Date</h2>
  <div class="post" id="post-1">
    <h3 class="storytitle">Post Title</h3>
    <div class="meta">Post Meta Data</div>
    <div class="storycontent">
      <p>Welcome to WordPress.</p>
    </div>
    <div class="feedback">Comments (2)</div>
  </div>
</div>
```

The Classic Theme hosts containers for the Date, Title, Post Meta Data, Post Content, and Feedback (number of comments). It also showcases a powerful feature. The ability to individually style a single post's look.

```
<div class="post" id="post-1">
```

The post CSS class selector applies the post styles to this container, but it also has an ID which is generated automatically by Wordpress. The code looks like this:

```
<div class="post" id="post-<?php the_ID(); ?>">
```

The use of the template tag the_ID() generates the ID number of the post. This provides a unique identifier for internal page links as well as for styles. This post could have a style for post-1, as could post-2. While it is a bit excessive to feature a style for every post, there may be a post or two you need to have looked a little different. Some plug-in may use this identifier to automatically change the look of different posts, too.

The Default Theme content container features a different look for multi-post views like the front page, categories, archives, and searches, and a different single post view for single posts. The multi-post view looks like this:

```
<div id="content" class="narrowcolumn">

  <div class="post" id="post-1">

    <h2>Post Title</h2>

    <small>Date</small>

    <div class="entry">

      <p>Post Content.</p>

    </div>

    <p class="postmetadata">Post Meta Data Section</p>

  </div>

  <div class="navigation">

    <div class="alignleft">Previous Post</div>

    <div class="alignright">Next Post</div>

  </div>
```

</div>

There is a lot going on here. Let's break it down.

<div id="content" class="narrowcolumn">

In the multi-post views, it features a class="narrowcolumn" and in the single post views, it features class="widecolumn" and the sidebar is not generated on that page, allowing the post to be viewed "wide" across the width of the content area.

<div class="post" id="post-1">

Like the Classic Theme, this division sets up the style for post and the identifier for post-X, with X representing the post's unique ID number. This allows for customizing the specific post's look.

<h2>Post Title</h2>

This encompasses the post's title code, styled by the <h2> tag.

<small>Date</small>

The date code is surrounded and styled by the small tag.

<div class="entry">

The post content is styled by a combination of the styles within the entry CSS selectors and the paragraph tag.

<p class="postmetadata">Post Meta Data Section</p>

The Post Meta Data Section contains the data details about the post such as the date, time, and categories the post belongs to.

<div class="navigation">

The Next and Previous Links are styled in the navigation. They also include classes for alignleft for the Previous Post and alignright for the Next Post in chronological order.

These elements are shifted around in the single post view content structure:

<div id="content" class="widecolumn">

```
  <div class="navigation">
    <div class="alignleft"></div>
    <div class="alignright"></div>
  </div>
  <div class="post" id="post-1">
    <h2>Post Title</h2>
    <div class="entrytext">
      <p>Post content.</p>
      <p class="postmetadata alt">
        <small>Post Meta Data</small>
      </p>
    </div>
  </div>
</div>
```

The widecolumn class is featured to stretch the content across the page to fill in the absence of the sidebar. The navigation has been moved up to the top. And the Post Meta Data is now incorporated into the entrytext parent container and styled differently with an alt style added.

These two examples from the Default Theme just give an example of the many ways to control and customize a Wordpress site and in this structure it is at its most basic. This basic structure should serve in doing design work.

6.3.4 Sidebar

As we saw with the Default Theme, the sidebar can be visible or not, depending upon the template file in use. The sidebar, in general, can be simple or complex. WordPress Themes often feature information within the sidebar in nested lists.

In general, the WordPress sidebar features titles of the various sections within a list, with the section items in a nested list below the title.

The Classic Theme sidebar looks like this, with the links removed for simplification:

```
<div id="menu">
  <ul>
    <li class="pagenav">Pages
      <ul>
        <li class="page_item">Contact</li>
        <li class="page_item">About</li>
      </ul>
    </li>
    <li id="linkcat-1"><h2>Blogroll</h2>
      <ul>
        <li>Blogroll Link 1</li>
        <li>Blogroll Link 1</li>
        <li>Blogroll Link 1</li>
      </ul>
    </li>
    <li id="categories">Categories:
```

```
   <ul>
    <li>Category Link 1</li>
    <li>Category Link 2</li>
   </ul>
 </li>
 <li id="search">
    <label for="s">Search:</label>
    <form id="searchform" method="get" action="/index.php">
<div>
   <input type="text" name="s" id="s" size="15" /><br />
   <input type="submit" value="Search" />
</div>
   </form>
 </li>
 <li id="archives">Archives:
   <ul>
       <li>Archives Month Link 1</li>
       <li>Archives Month Link 2</li>
    </ul>
 </li>
 <li id="meta">Meta:
   <ul>
```

```
        <li>RSS Feed Link</li>
        <li>RSS Comments Feed Link</li>
        <li>XHTML Validator</li>
        <li>XFN Link</li>
        <li>WordPress Link</li>
      </ul>
    </li>
  </ul>
</div>
```

Most of these are self-explanatory. Each set of links has its own CSS selector: Pages, categories, archives, search, and meta.

6.3.4.1 PAGES AND LINK CATEGORIES

The Pages and Links category, labeled "Blogroll", uses the <?php get_links_list(); ?> and <?php wp_list_pages(); ?> template tags which automatically generates a heading.

For the Links category, it generates an h2 heading for that set of links. This means you can style the menu h2 heading to look differently from the rest of the headings, or, if you want them to all look the same, make sure that the menu h2 style matches the rest of the category styles which are not automatically generated.

The Pages template tag generates pagenav as the heading and then identifies the pages in a new way. As a general list viewed on multi-post and single post views, the Page list items feature a class="page_item" to style those links. When viewing an individual Page, that Page's link will change to class="current_page_item", which can then be styled to look differently from the rest of the Page links.

6.3.4.2 CATEGORIES, ARCHIVES, AND META

The other sidebar section titles, categories, archives, meta, and others, do not use template tags which generate their own titles. These are set inside of PHP statements which "print" the text on the page. While these could be put inside of heading tags, WordPress uses the _e() function to display or "echo" the text titles while also marking the text as a possible target for language translation. If you will be developing your theme for public release, using the echo functions is highly recommended.

You can style these individually or all the same. Some Themes, like the Default Theme, put all these in <h2> headings so the list headings will all look the same. Therefore, they may or may not use style references for each section. You may add them if you need them to change the look of each section of links.

6.3.4.3 SEARCH FORM

The search form is found within the searchform.php. It may be found in different locations within the sidebar. To style the overall search form, use the search ID. Here is a list of the individual areas of the search form which may be styled by default. You may add style classes to gain more control over the look of your search form.

```
<li id="search">

  <label for="s">Search:</label>

    <form id="searchform" method="get" action="/index.php">

      <div>

         <input type="text" name="s" id="s" size="15" /><br />

         <input type="submit" value="Search" />

      </div>

     </form>
```

#search

The overall style for the search form.

#search label

Used to style the label tag, if necessary.

#searchform

Used to style the form itself.

#search div

This unlabeled div is a child container of the parent container search and maybe styled from within that selector.

#searchform input

To style the input area for the search, this selector combination will work.

#searchsubmit

Used by the Default Theme, this selector may be used to style the search or submit button.

The search form area, input, and button can be styled in many ways, or left with the default input and "button" look.

6.3.4.4 META FEED LINKS

The Meta links may be shown as text or icons representing the various links. The XHTML and CSS validation links may use the W3 icons. The various Feeds can also be represented as icons. Or left as text. It's up to you. Use of the feeds within your sidebar with text or icons is covered by the article Wordpress Feeds.

6.4 Footer

The footer is found within the footer.php template file. In both the Default and Classic Themes, the footer contains little information.

6.4.1 Classic Theme

```
<p class="credit">
  <!--15 queries. 0.152 seconds. -->
   <cite>
```

```
        Powered by <a href='http://wordpress.org'
        title='Powered by WordPress, state-of-the-art
        semantic personal publishing platform.'>
        <strong>WordPress</strong></a>
    </cite>
</p>
</div>
```

The footer's content is styled with the credit class and the paragraph and cite tags.

The tag displays the number of MySQL queries used on the page and the time it took for the page to load, in HTML commented code. It is there for the administrator's convenience and use. It is only visible within the page's source code. If you would like to display this visible on the page, remove the comments. Its look will be influenced by the credit class style of the paragraph tag. On the template file, it looks like this:

```
<!--<?php echo $wpdb->num_queries; ?> queries.
<?php timer_stop(1); ?> seconds. -->
```

6.4.2 Default Theme

```
<div id="footer">
   <p>Blogging in the WordPress World
      is proudly powered by
      <a href="http://wordpress.org/">WordPress</a><br />
      <a href="feed:http://example.com/feed/">Entries (RSS)</a>
      and
      <a href="feed:http://example.com/comments/feed/">
```

```
    Comments (RSS)</a>.
   <!-- 18 queries. 0.186 seconds. -->
  </p>
</div>
```

The Default Theme's footer is styled by the footer ID and the paragraph tag. While the footer area itself maybe styled by the footer, the paragraph tag controls the text within it. To style the paragraph tag differently within the footer than the rest of your page:

```
#footer p {styles}
```

6.4.3 WORDPRESS TEMPLATE

Template files are the building blocks of your Wordpress site. They fit together like the pieces of a puzzle to generate the web pages on your site. Some templates (the header and footer template files for example) are used on all the web pages, while others are used only under specific conditions.

A traditional web page consists of two files:

- The XHTML page to hold the structure and content of the page and
- The CSS Style Sheet which holds the presentation styles of the page.

In Wordpress, the (X)HTML structure and the CSS style sheet are present but the content is generated "behind the scenes" by various template files. The template files and the style sheet are stored together as a Wordpress Theme.

6.4.3.1 THE WORDPRESS PAGE STRUCTURE

A simple Wordpress web page structure is made up of three basic building "blocks": a header, the content, and footer. Each of these blocks is generated by a template file in your current Wordpress Theme.

Header

Content

Footer

- The header contains all the information that needs to be at the top — i.e. inside the <head> tag — of your XHTML web page, such as the <doctype>, <meta> tags and links to style sheets. It also includes the opening <body> tag and the visible header of your blog (which typically includes the title of your site, and may also include navigation menus, a logo bar, the description of your site, etc.).

- The content block contains the posts and pages of your blog, i.e. the "meat" of your site.

- The footer contains the information that goes at the bottom of your page, such as links to other Pages or categories on your site in a navigation menu, copyright and contact information, and other details.

6.4.4 Basic Template Files

To generate such a structure within a Wordpress Theme, start with an index.php template file in your Theme's directory. This file has two main functions:

- Include or "call" the other template files

- Include the Wordpress Loop to gather information from the database (posts, pages, categories, etc.)

For our simple structure, we only need to include two other template files: the header and the footer. These must be named header.php and footer.php. The Template Tags that include them look like this:

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

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

In order to display the posts and pages of your blog (and to customize how they are being displayed), your index.php file should run the Wordpress Loop between the header and footer calls.

6.4.5 More Complex Page Structures

Header

Content

Sidebar

Footer

Many Wordpress themes include one or several sidebars that contain the navigation features and more information about your website. The sidebar is generated by a template file called sidebar.php. It can be included in your index.php template file with the following template tag:

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

Notice that we have not included a template tag to "get" the content of our web page. That is because the content is generated in the Wordpress Loop, inside index.php.

Also note that the Theme's style sheet determines the look and placement of the header, footer, sidebar, and content in the user's browser screen. This is the simplified basic model of a Wordpress template and what it contains.

6.5 TEMPLATE FILE HIERARCHY

6.5.1 The General Theory

Wordpress uses the Query String information contained within each link on the web site to decide which template or set of templates will be used to display the page.

First, a Wordpress match every Query String to query types i.e. it decides what type of page (a search page, a category page, the home page etc.) is being requested.

Templates are then chosen and web page content is generated in the order suggested by the Wordpress Template hierarchy, depending upon what templates are available in a particular Wordpress Theme.

Wordpress looks for template files with specific names in the current Theme's directory and uses the first matching template file listed under the appropriate query section below.

With the exception of the basic index.php template file, Theme developers can choose whether they want to implement a particular template file or not. If Wordpress cannot find a template file with a matching name, it skips down to the next file name in the hierarchy. If Wordpress cannot find any matching template file, index.php (the Theme's home page template file) will be used.

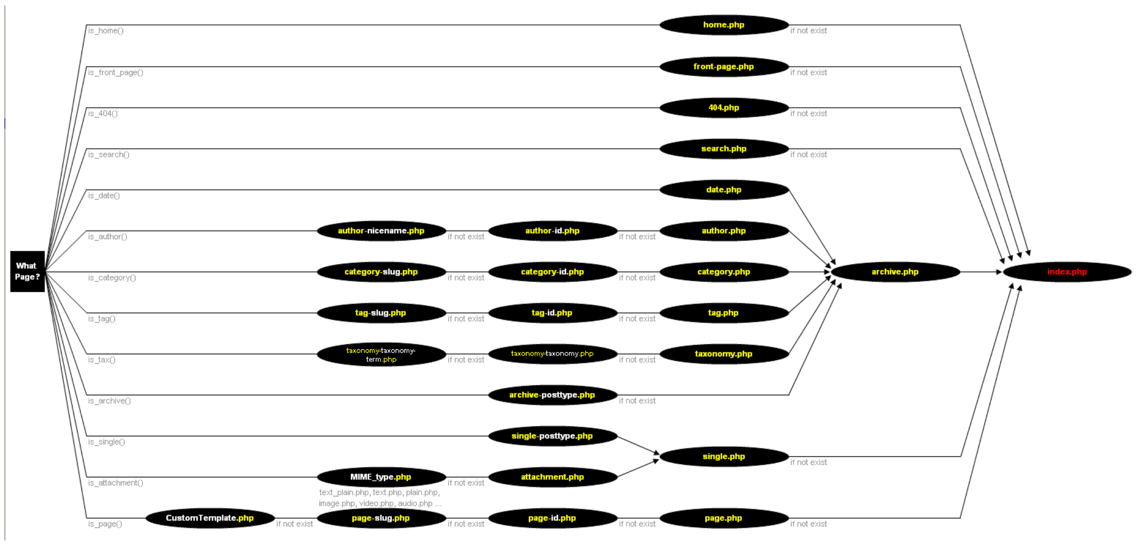
6.5.2 Examples

If your page is at http://example.com/wp/ and a visitor clicks on a link to a category page like http://example.com/wp/category/your-cat/, Wordpress looks for a template file in the current Theme's directory that matches the category's ID. If the category's ID is 4, Wordpress looks for a template file named category-4.php. If it is missing, Wordpress next looks for a generic category template file category.php. If this file does not exist either, Wordpress looks for a generic archive template, archive.php. If it is missing as well, Wordpress falls back on the main Theme template file, index.php.

If a visitor goes to the home page at http://example.com/wp/, Wordpress first determines whether it has a static front page. If a static front page has been set, then Wordpress loads that page according to the page template hierarchy. If a static front page has not been set, then Wordpress looks for a template file called home.php and uses it to generate the requested page. If home.php is missing, Wordpress looks for a file called index.php in the active theme's directory, and uses that template to generate the page.

6.5.3 Visual Overview

The following diagram shows which template files are called to generate a Wordpress page based on the Wordpress Template hierarchy.



Hierarchy picture 2

6.5.4 The Template Hierarchy in Detail

The following sections describe the order in which template files are being called by WordPress for each query type.

Home Page display

1. home.php

2. index.php

   Front Page display

1. front-page.php - Used for both Your latest posts or A static page as set in the Front page displays section of Settings -> Reading

2. Page display rules - When Front page is set in the Front page displays section of Settings -> Reading

3. Home Page display rules - When Posts page is set in the Front page displays section of Settings -> Reading

Single Post display

1. single-{post_type}.php - If the post type were product, WordPress would look for single-product.php.
2. single.php
3. index.php

Page display

1. custom template - Where custom template is the Page Template assigned to the Page.
2. page-{slug}.php - If the page slug is recent-news, WordPress will look to use page-recent-news.php
3. page-{id}.php - If the page ID is 6, WordPress will look to use page-6.php
4. page.php
5. index.php

Category display

1. category-{slug}.php - If the category's slug were news, Wordpress would look for category-news.php
2. category-{id}.php - If the category's ID were 6, Wordpress would look for category-6.php
3. category.php
4. archive.php
5. index.php

Tag display

1. tag-{slug}.php - If the tag's slug were sometag, WordPress would look for tag-sometag.php
2. tag-{id}.php - If the tag's ID were 6, WordPress would look for tag-6.php

3. tag.php
4. archive.php
5. index.php

Custom Taxonomies display

1. taxonomy-{taxonomy}-{term}.php - If the taxonomy were sometax, and taxonomy's slug were some times Wordpress would look for taxonomy-sometax-someterm.php. In the case of Post Formats, the taxonomy is 'post_format' and the terms are 'post-format-{format}. i.e. taxonomy-post_format-post-format-link.php
2. taxonomy-{taxonomy}.php - If the taxonomy were sometax, WordPress would look for taxonomy-sometax.php
3. taxonomy.php
4. archive.php
5. index.php

Custom Post Types display

1. archive-{post_type}.php - If the post type were product, WordPress would look for archive-product.php.
2. archive.php
3. index.php

(when displaying a single custom post type see the Single Post display section above.)

Author display

1. author-{nicename}.php - If the author's nice name were rami, WordPress would look for author-rami.php.

2. author-{id}.php - If the author's ID were 6, WordPress would look for author-6.php.
3. author.php
4. archive.php
5. index.php

Date display

1. date.php
2. archive.php
3. index.php

Search Result display

1. search.php
2. index.php

404 (Not Found) display

1. 404.php
2. index.php

attachment display

1. MIME_type.php - it can be any MIME type (image.php, video.php, audio.php, application.php or any other).
2. attachment.php
3. single.php
4. index.php

6.5.5 Filter Hierarchy

The WordPress templates system allows us to filter the hierarchy. The filter (located in the get_query_template() function) uses this filter name: "{$type}_template_hierarchy" where $type is the a file name in the hierarchy without the .php extension.

Like these (not a complete list):

- single_template_hierarchy
- page_template_hierarchy
- category_template_hierarchy
- tag_template_hierarchy
- taxonomy_template_hierarchy
- archive_template_hierarchy
- date_template_hierarchy
- search_template_hierarchy
- 404_template_hierarchy

6.5.6 Example

For example, let's take the default author hierarchy:

- author-{nicename}.php
- author-{id}.php
- author.php

To add author-{role}.php before author.php we can manipulate the actual hierarchy using the 'author_template_candidates' hook:

```
function author_role_template( $templates ) {
   global $role;
```

```
    $new_template = array( "author-$role.php" );

    $templates = array_merge(

        array_slice( $templates, 0, -1 ), // before

        $new_template,                    // inserted

        array_slice( $templates, -1 )     // after

    );

    return $templates;

}

add_filter( 'author_template_hierarchy', 'author_role_template' );
```

This gives us an understanding of how Wordpress functions and the better we understand the basic hierarchy we design keeping it in mind. The template hierarchy functions are located in wp-includes/theme.php.

6.6 THE LOOP

The Loop is what Wordpress uses to display its posts and it important to have a working understanding of the basics. "The Loop" is a term that refers to the main process of Wordpress. We use The Loop in our template files to show posts to visitors. It is possible to make templates without The Loop, but it would only be able to display data from one post.

The first thing Wordpress does is check that all the files it needs are present. Next, it collects the default settings, as defined by the blog administrator, from the database. This includes things like the number of posts to display per page, whether commenting is enabled, and the like. Once these defaults are established, Wordpress checks to see what the user asked for. This information is used to determine which posts to fetch from the database.

If the user didn't ask for a specific post, category, page, or date, Wordpress uses the previously collected default values to determine which posts to prepare for the user. For example, if the blog administrator has selected to display 5 posts per page in Administration > Settings > Reading, then Wordpress will fetch the five most recent posts from the database. If the user did ask for a specific post, category, page, or date, then Wordpress will use that information to specify which post(s) to fetch from the database.

Once all this is done, Wordpress connects to the database, retrieves the specified information, and stores the results in a variable. It is The Loop that accesses this variable, and uses the values for display in your templates.

By default, if the visitor did not select a specific post, page, category, or date, Wordpress uses index.php to display everything. For the first part of this discussion of The Loop we'll focus only on index.php, and the default display of your site. Later on, once you understand how things work, we'll investigate tricks with The Loop in other template files.

6.6.1 A Simple Index Page

The following is a fully functional index which will display the contents (and just the contents) of each post, according to the conditions used to prepare The Loop. The only purpose for showing you this is to demonstrate how little is actually necessary for the functioning of The Loop. The bulk of the stuff in your index.php is CSS, HTML, and PHP declarations to make The Loop look pretty.

```
<?php

get_header();

if (have_posts()) :

  while (have_posts()) :

    the_post();

    the_content();

  endwhile;
```

```
endif;

get_sidebar();

get_footer();

?>
```

Now, let's look at the bulk of the stuff that makes The Loop look pretty.

6.6.1.1 THE DEFAULT LOOP

The following is a step-by-step look at the default usage of the Loop that comes with the default and classic theme in the standard installation of WordPress v1.5. This basics setup should work in newer version of Wordpress as well.

6.6.1.2 BEGIN THE LOOP

Found at the top of the default index.php template file is the starting code for The Loop.

```
<?php if (have_posts()) : ?>

<?php while (have_posts()) : the_post(); ?>
```

1. First it checks whether any posts were collected with the have_posts() function.
2. If there are any posts, a PHP while loop is started. A while loop will continue to execute as long as the condition in the parenthesis is logically true. So as long as the function have_posts() returns a true value, The Loop will keep going.
3. The function have_posts() simply checks the next item in the collection of posts: if there's another item, return true; if there is no next item, return false.

6.6.1.3 GENERATING THE POST

The function the_post() takes the current item in the collection of posts and makes it available for use inside this iteration of The Loop. Without the_post(), many of the Template Tags used in your theme would not work.

Once the post data is made available, the template can start showing post data to the visitor.

6.6.1.4 TITLE, DATE AND AUTHOR

The following template tags get the current post's title, as well as the time it was posted and who posted it.

```
<h2 id="post-<?php the_ID(); ?>">

<a href="<?php the_permalink() ?>" rel="bookmark" title="Permanent Link to <?php the_title(); ?>">

<?php the_title(); ?></a></h2>

<small><?php the_time('F jS, Y') ?> <!-- by <?php the_author() ?> --></small>
```

6.6.1.5 POST CONTENT

The the_content() template tag displays the content of the post. This is the meat and potatoes of each pass through The Loop:

```
<div class="entry">

<?php the_content('Read the rest of this entry &raquo;'); ?>

</div>
```

If you include the Quicktag button called more, and shown as <!--more-->, in the body of your post, only the portion above that line will be displayed to viewers. So, if we only want your front page to show the first sentence or two of every post, simply insert <!--more--> after the first line into every post you make.

When viewing a single post, the <!--more--> delimiter is skipped. So putting the <!--more--> delimiter into all our posts forces the readers to click through to each individual post if they want to read the whole thing.

6.6.1.6 ADDITIONAL DETAILS

Beneath each post's content in the index.php template file is a place to display more information about the post, such as the categories, date, and comment information. Known as the post meta data section, if we are a logged in user of sufficient privilege (or the post's author), you will also see an "Edit This" link, thanks to the edit_post_link() template tag function.

```
<p class="postmetadata">

Posted in <?php the_category(', ') ?>

<strong>|</strong>

<?php edit_post_link('Edit',",'<strong>|</strong>'); ?>

<?php comments_popup_link('No Comments »', '1 Comment »', '% Comments »'); ?></p>
```

If commenting is enabled, or if the post has comments, the comments_popup_link() template tag will display a link to the comments. If we are using the comments popup window, this link will open the comments window; otherwise it will jump right to this post's comments.

If the visitor is viewing an index of posts (i.e.: more than one post in The Loop), the comments_popup_link() link will take the reader to this post's individual page.

6.6.1.7 TRACKBACK AUTO DISCOVERY

The trackback_rdf template tag's function is to output machine-readable code used for trackback auto-discovery.

```
<!—

<?php trackback_rdf(); ?>

-->
```

6.6.1.8 ENDING THE LOOP

The following ends The Loop. After this, the various post-related template tags will not work as expected (or if they do, they will use the last post from The Loop). This means, that if you need to use a template tag that works within The Loop, you need to put it in before this point.

```
<?php endwhile; ?>
```

This section, immediately after the end of The Loop, displays navigation controls to move forward and backward by each web page.

```
<div class="navigation">
```

```
<div class="alignleft"><?php posts_nav_link(",",'&laquo; Previous Entries') ?></div>
```

```
<div class="alignright"><?php posts_nav_link(",'Next Entries &raquo;',") ?></div>
```

```
</div>
```

If the site is set to display 10 posts per page, and the conditions used by The Loop collect 25 posts, there will be three pages to navigate: two pages of 10 posts each, and one page of 5 posts. The navigation links will allow the visitor to move forward and backward through the collection of posts.

The navigation controls are included outside The Loop, but inside the if condition, so that they only show up if there are any posts. The navigation functions themselves also check whether or not there is anything to which they will link, based on the current Loop, and only display links if there's something to link.

```
<?php else : ?>

 <h2 class="center">Not Found</h2>

 <p class="center">

<?php _e("Sorry, but you are looking for something that isn't here."); ?></p>
```

The else : clause determines what to do if have_posts() (from way up at the top) is false. That is to say, the stuff after the else will only be executed/displayed if The Loop had zero posts. No posts show up if, for example, the visitor requested a specific day for which no posts were made or a search was performed that produced no results.

```
  <?php endif; ?>
```

This ends the conditional test of "if there are posts do this, else if there are no posts, do that". Once the conditional test is finished, the default index.php template next includes the sidebar, and finally the footer.

This is the basic setup of the loop, there are many ways of customizing it and bending it to do some other things than in was meant for but as a basic introduction, the basic setup is what’s needed to know. If we have the need for it, we can have multiple loops working simultaneously and we can also deploy nested loops. Usually the default posting settings will be enough to satisfy the demands of a project as Wordpress offers a robust and easy way of updating pages in other ways, but is good to take into account that there are other options.

6.7 CSS IN WORDPRESS

WordPress relies heavily on the presentation styles within CSS. With the use of Themes, we have an almost infinite choice of layout options. Wordpress Themes make it easy to change your website look. I enables customization almost everything that Wordpress uses to style itself. Accessed 2011 May[http://codex.wordpress.org/]

CSS stands for Cascading Style Sheets. It allows you to store style presentation information (like colors and layout) separate from your HTML structure. This allows precision control of the website layout and makes pages faster and easier to update. Accessed 2011 May[www.w3schools.com /]

We will briefly look at a basic setup to give an idea how Wordpress functions in relation to the CSS.

6.7.1 Wordpress and CSS

Wordpress Themes use a combination of template files, template tags, and CSS files to generate your Wordpress site's look.

6.7.1.1 11.1.1 TEMPLATE FILES

Template files are the building blocks which come together to create your site. In the Wordpress Theme structure, the header, sidebar, content, and footer are all contained within individual files. They join together to create your page. This allows us to customize the building blocks. For example, in the default Wordpress Theme, the multi-post view found on the front page, category, archives, and search web pages on your site, the sidebar is present. Click on any post, we will be taken to the single post view and the sidebar will now be gone. We can choose which parts and pieces appear on your page, and customize them individually, allowing for a different header or sidebar to appear on all pages within a specific category.

6.7.1.2 TEMPLATE TAGS

Template tags are the pieces of code which provide instructions and requests for information stored within the Wordpress database. Some of these are highly configurable, allowing you to customize the date, time, lists, and other elements displayed on your website.

6.7.1.3 STYLE SHEET

The CSS what combines everything. On every template file within the site there are HTML elements wrapped around the template tags and content. In the style sheet within each Theme are rules to control the design and layout of each HTML element. Without these instructions, the page would simply look like a long typed page. With these instructions, we can move the building block structures around, making our header very long and filled with graphics or photographs, or simple and narrow. Your site can "float" in the middle of the viewer's screen with space on the left and right, or stretch across the screen, filling the whole page. The sidebar can be on the right or left, or even start midway down the page. But the instructions for styling are found in the style.css file within each Theme folder.

6.7.1.4 WORDPRESS GENERATED CLASSES

Several classes for aligning images and block elements (div, p, table etc.) were introduced in Wordpress 2.5: align center, alignleft and alignright. In addition the class align none is added to images that are not aligned, so they can be styled differently if needed.

The same classes are used to align images that have a caption (as of Wordpress 2.6). Three additional CSS classes are needed for the captions, together the alignment and caption classes are: Accessed 2011 May[www.w3schools.com /].

```
.align center, disincentive {

        display: block;

        margin-left: auto;

        margin-right: auto;

}
```

```
}

.alignleft {
	float: left;
}
.alignright {
	float: right;
}
.WP-caption {
	background-color: #f3f3f3;
	border: 1px solid #dds;
	-khtml-border-radius: 3px;
	-moz-border-radius: 3px;
	-web kit-border-radius: 3px;
	border-radius: 3px; /* optional rounded corners for browsers that support it */
	margin: 10px;
	padding-top: 4px;
	text-align: center;
}
.WP-caption mg {
	border: 0 none;
	margin: 0;
	padding: 0;
}
```

```
.WP-caption pawpaw-caption-text {

        font-size: 11px;

        line-height: 17px;

        margin: 0;

        padding: 0 4px 5px;

}
```

Each Theme should have these or similar styles in its style.css file to be able to display images and captions properly. The exact HTML elements and class and ID values will depend on the structure of the Theme you are using.

These are some of the basics of styling a Wordpress page with CSS, what can be done using the tags is almost limitless, that we do not have to utilize PHP scripting language helps us a lot in that we can use CSS and HTML (covered in the next chapter) to quickly and effectively control the look of a page.

6.8 HTML TO XHTML AND WORDPRESS

Wordpress, as a system, is based on documents written in the XHTML scripting language. XHTML 1.0 (which is currently the most widely supported version and stands for extensible Hyper Text Markup Language) became a W3C recommendation in the year 2000, and was intended to serve as an interim technology until XHTML 2.0 could be finalized. Eight years later XHTML 2.0 still isn't finished. This document therefore uses the phrase XHTML to refer to XHTML 1.0 only.

XHTML is very similar to HTML as both are descendants of a language called SGML. However, XHTML is also descended from XML, which is a scripting language with much stricter grammar rules than HTML, and XHTML has inherited some of that discipline. XHTML is mainly differentiated from HTML by its use of a new MIME TYPE and the addition of some new syntax rules which are explained below.

This is information that might not be directly relevant when developing with Wordpress as the development mainly happens by modifying the different PHP files with CSS. But as this is an integral part of the architecture of Wordpress it good to note the basics of the setup.

6.8.1 Why use XHTML

Wordpress prints XHTML from all its internal functions, all themes therefore are now in XHTML and so too are most plug-ins. This means a good working knowledge of XHTML is needed.

6.8.2 The differences between XHTML and HTML

If we are familiar with HTML, we will be glad to know that the majority of what we know about HTML is still relevant in XHTML. The main differences are that XHTML forces developers to be more consistent and to write more legible code. There are a few syntax and grammar differences and a few HTML tags have been dropped.

6.8.3 How do I write XHTML?

Here is brief example of XHTML and the differences between it and HTML. This is NOT a comprehensive XHTML language reference.

In these examples some code has been omitted for clarity

18.1.2 All tags, attributes and values must be written in lowercase.

Right:

```
<a href="buckminsterfullerene" >
```

Wrong:

```
<A HREF="buckminsterfullerene" >
```

18.1.3 All attribute values must be within quotes

Right:

```
<a href="buckminsterfullerene">
```

Wrong:

<a href=www.kilroyjames.co.uk>

6.8.3.1 ALL TAGS MUST BE PROPERLY NESTED

Right:

<em>this emphasis just keeps getting <strong>stronger and stronger</strong></em>

Wrong:

<em>this emphasis just keeps getting <strong>stronger and stronger</em></strong>

All XHTML documents must carry a DOCTYPE definition

The DOCTYPE is a mandatory piece of code that must appear at the start of every XHTML document, it tells the browser how to render the document.

6.8.3.2 RULES FOR THE DOCTYPE TAG:

- It must be the first tag in the document
- The DOCTYPE is not actually part of the XHTML document so don't add a closing slash
- It should point to a valid definition file called a DTD that tells the browser how to read the document
- You must write the DOCTYPE tag correctly otherwise your document will explode* (into little pieces of HTML called "tag soup") and be unable to be validated.

There are three types of valid XHTML 1.0 document: Strict, Transitional, and Frameset. If we can get our document to validate with "Strict", however some legacy tags and attributes aren't allowed in Strict so we can use "Transitional" state to bypass this problem.

Note: There might be a problem getting Wordpress to validate as Strict because, as of version 2.6.2, some of the internally generated <form> elements still use the "name" attribute, which is not allowed under the Strict DTD, IE. <input name="my_button" />

Note also that using a Transitional DTD takes most browsers out of "Standards" mode. It is much trickier to get a web site to look consistent across different browsers when the browsers are not in Standards mode. To get the best and most consistent result we should always strive to be able to publish “Strike”code.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-stricture">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional">
```

The HTML tag must contain an XMLNS attribute

The "XML name space" attribute must be included; it is required in all XHTML documents. Here is an example of how to write it:

```
<html XML="http://www.w3.org/1999/HTML">
```

6.8.3.3 DOCUMENTS MUST BE PROPERLY FORMED WITH HTML, HEAD, TITLE AND BODY TAGS

In HTML it is possible to write a web-page that contains none of the above tags; in XHTML it is not. The above tags must be included and they must be nested and ordered correctly, as follows (the DOCTYPE has been omitted):

```
<html xmlns="http://www.w3.org/1999/xhtml">

 <head>

  <title></title>

 </head>

 <body>

  <p>
```

See how the TITLE must be placed in the document HEAD – the TITLE is considered to be a "required child" element of the HEAD.

Notice that the HEAD must also appear before the document BODY.

Notice also how both the HEAD and the BODY must be contained

within the HTML tag. Again, HEAD and BODY are "required child"

elements of the HTML tag. Finally, notice that this text is

written within a <p>paragraph</p> tag; in XHTML you may

not write text directly in the BODY tag without using a suitable

container tag, such as <p> or <div>.

```
  </p>
 </body>
</html>
```

6.8.3.4 ALL TAGS MUST BE CLOSED, EVEN SINGLE TAGS

```
<p>Mary had a little lamb

<p>It's fleece was white as snow
```

This code is not valid XHTML as the closing </p> tags have been left out. The following example is correct.

```
<p>Mary had a little lamb</p>

<p>It's fleece was white as snow</p>
```

In XHTML even single tags have to be closed - absolutely NO tag may be left open.

```
<p>
 Mary had a little lamb <br>
 Its fleece was white as snow
</p>
```

Therefore the above example is wrong because the <br> tag is not closed. To close a single tag like <br> and <hr> you simply add a forward slash before the final bracket, like so: <br /> and <hr /> (the white space is optional). To correct the above example we'd write:

```
<p>
 Mary had a little lamb <br />
 It's fleece was white as snow
</p>
```

6.8.3.5 THIS IS CORRECT XHTML.

Attribute minimization isn't allowed

In HTML, attributes can be strung together almost like they were keywords, ie. <dl compact>, this is called attribute minimization. In XHTML that is not allowed, attributes and values must be explicit, ie.

```
<dl compact="compact">
```

6.8.3.6 ID AND NAME ATTRIBUTES

In HTML it was legal to use ID and NAME attributes interchangeably. In XHTML the NAME attribute is formally deprecated and cannot be used. In all cases where you would think to use a NAME attribute you must now use ID instead.

correct HTML

```
<input type="submit" name="s" value=" Search " >
```

and now the correct XHTML version

```
<input type="submit" id="s" value=" Search " />
```

STYLE is all in your HEAD

XHTML does not allow STYLE declarations within the body of a document they must be placed in the document HEAD instead.

6.8.3.7 ENTITY REFERENCES

Write all literal ampersands as &amp; or they will be assumed to be part of an entity reference. e.g. &reg; is the entity reference for the symbol ® Therefore M&S is invalid XHTML because &S is not an entity reference, you must write it as M&amp;S.

6.8.4 Conclusion

As was previously mentioned, this is not a comprehensive reference but it contains the basics that are needed to write strict and correctly formatted XHTML.

6.9 WORDPRESS THEMES

Wordpress Themes are files that work together to create the design and functionality of a Wordpress site. Each Theme may be different, offering many choices for site owners to take advantage of in order to instantly change their website look.

Fundamentally, the Wordpress Theme system is a way to "skin" your website. Yet, it is more than just a "skin." Skinning your site implies that only the design is changed. Wordpress Themes can provide much more control over the look and presentation of the material on your website.

A Wordpress Theme is a collection of files that work together to produce a graphical interface with an underlying unifying design for a weblog. These files are called template files. A Theme modifies the way the site is displayed, without modifying the underlying software. Themes may include customized template files, image files (*.jpg, *.gif), style sheets (*.css), custom Pages, as well as any necessary code files (*.php).

An example of what a theme can do is that, in this case for simplicity we will talk about a blog, but fundamentally it’s the same with a website is that if we write a lot of post about cheese and heavy metal music. Through the use of the WordPress Loop and template files, we can customize our cheese category posts to look different from your heavy metal category posts. Shall we say the cheese post is yellow and the heavy metal posts are red. This is just one example how you can design using Wordpress.

That is a theme in basic terms, but they are good in styling your page or changing the style of an existing page. We can usually start building a theme by getting a blank theme and setting it up to serve our projects needs and specifications.

For example the “naked theme” from, accessed 2011 may [www.starkerstheme.com] is a free theme with no styling. But has basic functionality built in so we can easily find the different sections of the code and start styling it. When you open the demo of the site we can see that it clearly has no defined style but contains the basic features we need. This theme is a striped version of the “Twenty ten” default theme that ships with Wordpress. As a side note is fully GPL licensed so you can use it for commercial work.

Remember that as long as a theme is public domain property we can take it and modify it. This can be time saving as well, if we see a site that is ideal for your purposes and is under the GPL license we can take it and modify it for our purpose. It has to be noted that graphic material included in these GPL sites is copyright protected and cannot be utilized.

6.10 BUILDING THEMES

Building a Wordpress theme is a combination of all the steps in design that we have been studying in previous chapters. The basic tool-set that is primarily needed is a working understanding of CSS, XHTML and the structuring behind Wordpress. As the PHP is controlled by CSS this is sufficient. The choice of where to start building is a choice the designer must decide in what is fitting to the design. The main structure is defined by the use of the different templates that control the different elements of the Wordpress construct.

When a theme is deployed, depending on the configuration of the theme we can star expanding the sites capabilities using what Wordpress calls “Plug-ins” (Plug-ins will be covered in a later chapter). Plug-ins in Wordpress enables the Developer to expand different aspects of the sites functionality; for example we want to have a nicer photo gallery than the default version provided by Wordpress, we can go and download a plug-in that changes the photo gallery to a light box themed gallery. These are the kind of small changes we can impact using plug-ins, but we can also launch big add-ons as for example a whole forum feature or in the case of a plug-in called BuddyPress a complete social network loosely modeled on Facebook.

As discussed in the previous chapter it’s possible to star working on a blank theme that enables a developer to specify what he wants on the site and styling it by using CSS. It is also possible to expand the original design using this method.

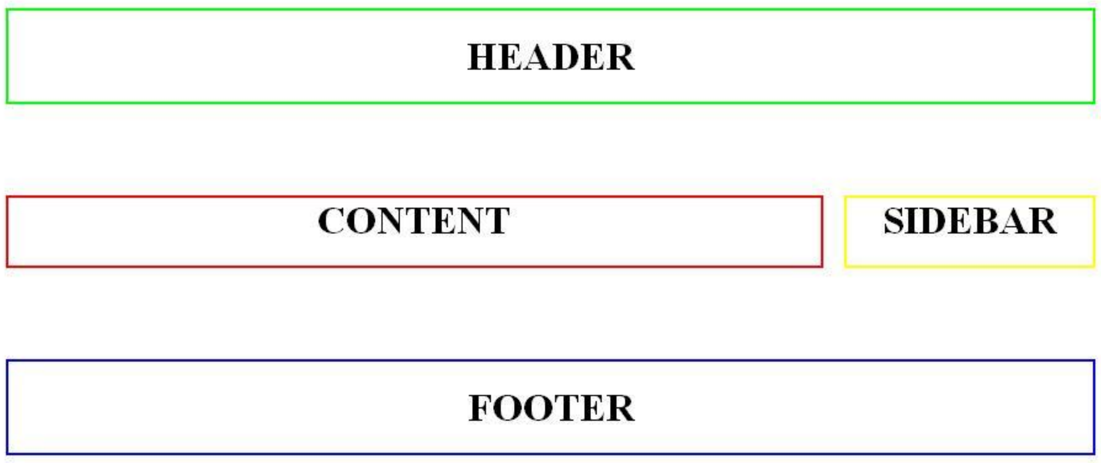
As this is a so possible to generate a theme using a online theme generator, these wary quite a lot in what they offer and the quality of code they produce, but as Wordpress is quite standardized in the way it works some of these editors and generators can be an asset.

Some ready-made themes are available online that are equipped with a backend that is so comprehensible that it enables editing of the theme to be almost anything in terms of the graphic design, but as these themes are pre-design they can be limited and because of the extensive modifying that has been done to them its highly likely that the code of the themes is so modified to suit the backend usability that we might find modifying the code impossible.

We will look at a so called 5 minute tutorial on building a simple theme for a Wordpress blog that is the basic form of a Wordpress site. Accessed 2011 May [www.levoltz.com].

6.10.1 Tutorial

The first thing is designing a web design structure as shown in the picture below.



Structure figure 3

The sections that make up are illustrated here. They are the header, content, sidebar, footer. To abstract this and give power to the template designers, Wordpress has placed the various sections into different pages.

In addition to this, you have to add proper CSS to your design. Include them in “style.css”.

Therefore, the files necessary are:

1. header.php -> Header

2. footer.php -> Footer

3. sidebar.php -> Sidebar

4. index.php -> Content glued with header, sidebar and footer.

5. style.css -> CSS

This is the basic files that are needed to create a theme.

6.10.2 Designing the Template

The HTML for our basic layout:

```
<html>

 <head>

 <title>Demo Template</title> <link rel=”stylesheet” href=”style.css” mce_href=”style.css”>

 </head>

 <body>  <div id=”wrapper”>

 <p><h1> </h1></p>

 <div id=”header”>
```

```
<h1>HEADER</h1>

</div>

<p><h1> </h1></p>

<div id=”main”>

<div id=”content”>

<h1>CONTENT</h1>

</div>

<div id=”sidebar”>

<h1>SIDEBAR</h1>

</div>

</div>

<div id=”both”></div>

<p><h1> </h1></p>

<div id=”footer”>

<h1>FOOTER</h1>

</div>

</div>

</body>

</html>
```

Have a look at the simple style sheet for the basic layout. Save this as “style.css”. It will be needed later.

```
body { text-align: center; }
```

```
#wrapper {      display: block;         /*border: 1px #000000 solid;*/         width:90%;
        margin:0px auto; }

#header {       border: 2px #00ff00 solid; }

#content {      width: 75%;    border: 2px #ff0000 solid;     float: left; }

#sidebar {      width: 23%;    border: 2px #ffff00 solid;     float: right; }

#both {         clear: both; }

#footer {       border: 2px #0000ff solid; }
```

Now that we have saved it, open the HTML file in a good text editor. We can see sections that are tagged with “header”, “footer”, “sidebar” and “content”. Let us create the header file.

1. Open notepad, and paste the code shown below. Save this file as “header.php”.

```
<br/>  <title>Demo Template</title>  <br/>”>

<div>  <p><h1> </h1></p>   <div>

<h1>HEADER</h1>

</div>          <p><h1> </h1></p> <br/>

</div>
```

2. Copy the fragment below and save it in a new file as “footer.php”.

```
<br/>   <div>

<h1>FOOTER</h1>

</div>      <br/>
```

3. Copy the sidebar code and save it in a new file as “sidebar.php”.

```
<br/>   <div>
```

<h1>SIDEBAR</h1>

</div> <br/>

4. Last let us create the “index.php”. If you notice, there are special lines such as “”, “”, “”. These are the functions that are used to glue the content with header, sidebar and footer.

<br/>

<div>   <div>

<h1>CONTENT</h1>

</div>   </div>

<div></div>   <br/>

5. Create a new folder and name it as “DarlingTemplate”. Move all the newly created files, including the “style.css” to the folder. To test the new template, upload[:1] this folder to the /wp-content/themes folder. Login to your blog and you will see the new template under the “Themes” tab.

6. Click on our template name to activate it. The template looks just like the simple HTML file we created earlier. But the post function is missing that is inherit to Wordpress.

7. Open up “index.php”, replace it with the text in the following box and save it. Make sure the modified file is uploaded to the /wp-content/themes/DarlingTemplate folder. Refresh the blog homepage. The post function should work now.

<br/><br/>

<br/>

<div><br/><br/>

<div><br/>

8. Take a screen shot of your theme. Save it as “screenshot.png” and upload it to

/wp-content/themes/DarlingTemplate.

Now, when we login to your Wordpress account, our template will have a medium sized thumbnail picture.

This is how the final version of DarlingTemplate looks in broad terms. It is a basic layout but as a guideline in developing themes it should be a good starting point.

6.11 CHILD THEMES

Child themes are the best way to modify existing themes in Wordpress, the purpose of the parent/child WordPress Theme feature is to protect the original WordPress Theme from design changes, allowing the user to change the child Theme without impacting the original code, which makes upgrading Themes easier.

A WordPress child theme is a theme that inherits the functionality of another theme, called the parent theme, and allows us to modify, or add to, the functionality of that parent theme. This article shows how to create a basic child theme and explains what you can do with it. As an example parent theme it uses Twenty Ten, the new default theme in WordPress 3.0. Accessed 2011 May [http://codex.wordpress.org]

Making a child theme is very simple. Create a directory, put a properly formatted style.css file in it, and you have a child theme! With a little understanding of HTML and CSS, you can make that very basic child theme modify the styling and layout of a parent theme to any extent without editing the files of the parent theme itself. That way, when the parent theme is updated, your modifications are preserved.

Here is basic example of the directory structure. The scheme below shows the location of a child theme along with its parent theme (Twenty Ten) in a typical WordPress directory structure:

public_html

wp-content

themes (directory where all themes are)

twentyten (directory of our example parent theme, Twenty Ten) twentyten-child (directory of our child theme; can be named anything)

style.css (required file in a child theme; must be named style.css)

This directory can contain as little as a style.css file, and as much as any full-fledged WordPress theme contains:

1. style.css (required)
2. functions.php (optional)
3. Template files (optional)
4. Other files (optional)

Now we will look at all these separately.

6.11.1 The required style.css file

style.css is the one and only required file in a child theme. It provides the information header by which WordPress recognizes the child theme, and it replaces the style.css of the parent.

As with any WordPress theme, the information header must be at the top of the file, the only difference being that in a child theme the Template: line is required, so that WordPress knows which theme is the parent of the child theme.

Here is an example information header of a child theme's style.css:

```
/*

Theme Name:     Twenty Ten Child

Theme URI:      http: //example.com/

Description:    Child theme for the Twenty Ten theme

Author:         Your name here

Author URI:     http: //example.com/about/
```

Template: twentyten

Version: 0.1.0

*/

A quick explanation of each line:

- Theme Name. (required) Child theme name.
- Theme URI. (optional) Child theme web-page
- Description. (optional) What this theme is. E.g.: My first child theme. Hurrah!
- Author URI. (optional) Author web-page
- Author. (optional) Author name.
- Template. (required) directory name of parent theme, case-sensitive.

o NOTE. You have to switch to a different theme and back to the child theme when you modify this line.

- Version. (optional) Child theme version. E.g.: 0.1, 1.0, etc.

The part after the closing */ of the header works as a regular stylesheet file. It is where you put the styling rules you want WordPress to apply.

Note that a child theme's stylesheet replaces the stylesheet of the parent completely. (The parent's stylesheet is not loaded at all by WordPress.) If we simply want to modify a few small things in the styling and layout of the parent —rather than make something new from scratch— we have to import explicitly the stylesheet of the parent, and then add your modifications. The following example shows how to use the @import rule to do that.

6.11.2 Example of a basic Child Theme

Our parent theme for this example is Twenty Ten. We like most everything about it, except the color of the site's title, which we want to change from black to green. Using a child theme, all it takes is three simple steps:

1. Make a new directory in wp-content/themes, and name it twentyten-child (or anything you like).

2. Save the code below in a file named style.css, and drop this file in the new directory.

3. Go to Dashboard › Themes and activate your new theme, the Twenty Ten Child.

```
/*

Theme Name: Twenty Ten Child

Description: Child theme for the Twenty Ten theme

Author: Your name here

Template: twentyten

*/

@import url("../twentyten/style.css");

#site-title a {

   color: #009900;

}
```

Here is what the above code, step by step:

1. /* opens the child theme's information header.

2. Theme Name: declares the child theme's name.

3. Description: describes what the theme is. (Optional; can be omitted.)

4. Author: declares the author's name. (Optional; can be omitted.)

5. Template: declares the child theme's parent; i.e., the directory of the theme whose templates the child uses.

6. */ closes the child's information header.

7. The @import rule brings in the parent’s stylesheet.

8. The #site-title a rule defines a colour (green) for the site’s name, overriding the corresponding rule of the parent.

Note on the @import rule

There must be no other CSS rules above the @import rule. If you put other rules above it, it will be invalidated and the stylesheet of the parent will not be imported.

Note on RTL support

To support RTL languages, add rtl.css file to your child theme, containing:

```
/*

Theme Name: Twenty Ten Child

Template: twentyten

*/

@import url("../twentyten/rtl.css");
```

WordPress auto-loading rtl.css file only if is_rtl(). Even if the parent theme has no rtl.css file, it's recommended to add the rtl.css file to your child theme.

6.11.3 Using functions.php

Unlike style.css, the functions.php of a child theme does not override its counterpart from the parent. Instead, it is loaded in addition to the parent’s functions.php. (Specifically, it is loaded right before the parent’s file.)

In that way, the functions.php of a child theme provides a smart, trouble-free method of modifying the functionality of a parent theme. Say that we want to add a PHP function to your theme. The fastest way would be to open its functions.php file and put the function there. But that’s not smart: The next time our theme is updated, our functions will disappear. But there is an alternative way: we can create a child theme, add a functions.php file in it, and add your function to that file. The function will do the exact same job from there too, with the advantage that it will not be affected by future updates of the parent theme.

The structure of functions.php is simple: An opening PHP tag at the top, a closing PHP tag at the bottom, and, between them, your bits of PHP. In it we can put as many or as few functions as you wish. The example below shows an elementary functions.php file that does one simple thing: Ads a favicon link to the head element of HTML pages.

```
function favicon_link() {

    echo '<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico" />' . "\n";

}

add_action('wp_head', 'favicon_link');
```

As a side note; the fact that a child theme's functions.php is loaded first means that we can make the user functions of our theme plug-able —that is, replaceable by a child theme— by declaring them conditionally. E.g.:

```
if (!function_exists('theme_special_nav')) {

    function theme_special_nav() {

        //  Do something.

    }

}
```

In that way, a child theme can replace a PHP function of the parent by simply declaring it again.

6.11.3.1REFERENCING / INCLUDING FILES IN YOUR CHILD THEME

When we need to include files that reside within our child theme's directory structure, we will use get_stylesheet_directory(). Because the parent template's style.css is re-placed by our child theme's style.css, and our style.css resides in the root of our child theme's sub directory, get_stylesheet_directory() points to our child theme's directory (not the parent theme's directory).

Here's an example, using require once, that shows how we can use get_stylesheet_directory when referencing a file stored within your child theme's directory structure.

require_once(get_stylesheet_directory(). '/my_included_filer');

6.11.3.1.1 Using Post Formats

A child theme inherits post formats as defined by the parent theme. But, when creating child themes, be aware that using add_theme_support('post-formats') will override the formats as defined by the parent theme, not add to it.

6.11.3.2 TEMPLATE FILES

Templates in a child theme behave just like style.css, in that they override their namesakes from the parent. A child theme can override any parental template by simply using a file with the same name. (NOTE. index.php can be overridden only in WordPress 3.0 and newer.)

Again, this WordPress feature lets you modify the templates of a parent theme without actually editing them, so that your modifications are preserved when the parent theme is updated.

Here are example cases for using template files in a child theme:

- To add a template that is not offered by the parent theme (e.g., a template for a site-map page, or for single-column pages, that will be available to select in the Page Edit screen).
- To add a more specific template (see Template Hierarchy) than what the parent uses (e.g., a tag.php template to use for tag archives instead of the generic archive.php of the parent).
- To replace a template of the parent (e.g., make your own home.php to override the parent’s home.php).

6.11.4 Other files

In addition to style.css, functions.php, and to template files like index.php, and home.php, a child theme can use any type of file full-fledged themes use, as long as that file is properly linked. For example, a child theme can use icons and images that are linked from its stylesheet, JavaScript files linked from the top or bottom of pages, or extra PHP files called from its templates or from its functions.php file.

This is a brief introduction to the uses of the child themes and what they can be used for in design. There are other ways of using child themes and the options this enables.

Again this is a feature that enables designers to work effectively adding functionality to an already existing site, and offers a non invasive way of designing on a other design.

6.12 WORDPRESS PLUGINS

One of the biggest features that Wordpress has incorporated is the modular design that enables the use of plug-in. These range from custom font options that enable other fonts than the web standard to building community websites with plug-ins enabling forums and chat boards.

A Wordpress Plug-in is a program, or a set of one or more functions, written in the PHP scripting language, that adds a specific set of features or services to the Wordpress web-log, which can be seamlessly integrated with the weblog using access points and me-thods provided by the Wordpress Plug-in Application Program Interface (API). As the API feature is meant for plug-in developers we will not concentrate on it in this paper as it is not strictly necessary to use when building Wordpress sites, but we will look at some theory to still clarify the subject and cover the basics.

6.12.1 Creating a Plug-in

This section goes through the steps you need to follow, and things to consider when creating a well-structured Wordpress Plug-in.

6.12.1.1PLUG-IN NAME

The first task in creating a Wordpress Plug-in is to think about what the Plug-in will do, and make a name for it. Most Plug-in developers choose to use names that somewhat describe what the Plug-in does; for instance, a weather-related Plug-in would probably have the word "weather" in the name. The name can be multiple words.

6.12.1.2PLUG-IN FILES

The next step is to create a PHP file with a name derived from your chosen Plug-in name. For instance, if your Plug-in will be called "Fabulous Functionality", you might call your PHP file fabfunc.php. Again, try to choose a unique name. People who install our Plug-in will be putting this PHP file into the Wordpress Plug-in directory in their installation, wp-content/plugins/, so no two Plugging they are using can have the same PHP file name.

Another option is to split your Plug-in into multiple files. Your Wordpress Plug-in must have at least one PHP file; it could also contain JavaScript files, CSS files, image files, language files, etc. If there are multiple files, pick a unique name for a file directory and for the main PHP file, such as fabfunc and fabfunc.php in this example, put all our plugin files into that directory, and tell your plugin users to install the whole directory under wp-content/plugins/.

In the rest of this article, "the Plugin PHP file" refers to the main Plugin PHP file, whether in wp-content/plugins/ or a sub-directory.

6.12.1.3README FILE

If we want to host your Plug-in on http://wordpress.org/extend/plugins/, we also need to create a readme.txt file in a standard format, and include it with our Plug-in.

6.12.1.4HOME PAGE

It is also very useful to create a web page to act as the home page for your Wordpress Plug-in. This page should describe how to install the Plug-in, what it does, what versions of Wordpress it is compatible with, what has changed from version to version of your Plug-in, and how to use the Plug-in.

6.12.1.5FILE HEADERS

It’s time to put some information into your main Plug-in PHP file.

6.12.2 Standard Plug-in Information

The top of your plugin's main PHP file must contain a standard plugin information header. This header lets Wordpress recognize that our Plug-in exists, add it to the Plug-in management screen so it can be activated, load it, and run its functions; without the header, our Plug-in will never be activated and will never run. Here is the header format:

```
<?php

/*

Plugin Name: Name Of The Plugin

Plugin URI: http://URI_Of_Page_Describing_Plugin_and_Updates

Description: A brief description of the Plugin.

Version: The Plugin's Version Number, e.g.: 1.0

Author: Name Of The Plugin Author

Author URI: http://URI_Of_The_Plugin_Author

License: A "Slug" license name e.g. GPL2

*/

?>
```

The minimum information Wordpress needs to recognize your Plug-in is the Plug-in Name line. The rest of the information (if present) will be used to create the table of plugins on the Plug-in management screen. The order of the lines is not important.

The License slug should be a short common identifier for the license the plug-in is under and is meant to be a simple way of being explicit about the license of the code.

6.12.3 License

It is customary to follow the standard header with information about licensing for the Plug-in; Most Plug-in use the GPL2 license used by Wordpress or a license compatible with the GPL2. To indicate a GPL2 license, include the following lines in your Plug-in:

```
<?php
/*  Copyright YEAR  PLUGIN_AUTHOR_NAME  (email : PLUGIN AUTHOR EMAIL)
  This program is free software; you can redistribute it and/or modify
   It under the terms of the GNU General Public License, version 2, as
   published by the Free Software Foundation.
   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   we should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/
?>
```

As an example of a license text that might be used.

6.12.4 Wordpress Plug-in Hooks

Many Wordpress Plug-in accomplish their goals by connecting to one or more Wordpress Plug-in "hooks". The way Plug-in hooks work is that at various times while Wordpress is running, Wordpress checks to see if any Plug-in have registered functions to run at that time, and if so, the functions are run. These functions modify the default behavior of Wordpress.

For instance, before Wordpress adds the title of a post to browser output, it first checks to see if any Plug-in has registered a function for the "filter" hook called "the_title". If so, the title text is passed in turn through each registered function, and the final result is what is printed. So, if our Plug-in needs to add some information to the printed title, it can register a "the_title" filter function.

Another example is the "action" hook called "wp_footer". Just before the end of the HTML page Wordpress is generating, it checks to see whether any Plug-in have registered functions for the "wp_footer" action hook, and runs them in turn.

6.12.5 Template Tags

Another way for a Wordpress Plug-in to add functionality to Wordpress is by creating custom Template Tags. Someone who wants to use your Plug-in can add these "tags" to their theme, in the sidebar, post content section, or wherever it is appropriate. For instance, a Plug-in that adds geographical tags to posts might define a template tag function called geotag_list_states() for the sidebar, which lists all the states posts are tagged with, with links to the state-based archive pages the Plug-in enables.

To define a custom template tag, simply write a PHP function and document it for Plug-in users on your Plugin's home page and/or in the Plugin's main PHP file. It's a good idea when documenting the function to give an example of exactly what needs to be added to the theme file to use the function, including the <?php and ?>.

6.12.6 Saving Plug-in Data to the Database

Most Wordpress Plug-in will need to get some input from the site owner or blog users and save it between sessions, for use in its filter functions, action functions, and template functions. This information has to be saved in the Wordpress database, in order to be persistent between sessions. There are two basic methods for saving Plug-in data in the database:

1. Use the Wordpress "option" mechanism (described below). This method is appropriate for storing relatively small amounts of relatively static, named pieces of data -- the type of data you'd expect the site owner to enter when first setting up the Plug-in, and rarely change thereafter.
2. Post Meta (a.k.a. Custom Fields). Appropriate for data associated with individual posts, pages, or attachments. See post_meta Function Examples, add_post_meta(), and related functions.
3. Create a new, custom database table. This method is appropriate for data associated with individual posts, pages, attachments, or comments -- the type of data that will grow as time goes on, and that doesn't have individual names. See Creating Tables with Plug-in for information on how to do this.

6.12.7 Wordpress Options Mechanism

See Creating Options Pages for info on how to create a page that will automatically save your options for you.

Wordpress has a mechanism for saving, updating, and retrieving individual, named pieces of data ("options") in the Wordpress database. Option values can be strings, arrays, or PHP objects (they will be "serialized", or converted to a string, before storage, and unserialized when retrieved). Option names are strings, and they must be unique, so that they do not conflict with either Wordpress or other Plug-ins.

Here are the main functions your Plug-in can use to access Wordpress options.

add_option($name, $value, $deprecated, $autoload);

Creates a new option; does nothing if option already exists.

$name

Required (string). Name of the option to be added.

$value

Optional (mixed), defaults to empty string. The option value to be stored.

$deprecated

Optional (string), no longer used by Wordpress, You may pass an empty string or null to this argument if you wish to use the following $autoload parameter.

$autoload

Optional, defaults to 'yes' (enum: 'yes' or 'no'). If set to 'yes' the setting is automatically retrieved by the wp_load_alloptions function.

```
get_option($option);
```

Retrieves an option value from the database.

$option

Required (string). Name of the option whose value you want returned. You can find a list of the default options that are installed with WordPress at the Option Reference.

```
update_option($option_name, $newvalue);
```

Updates or creates an option value in the database (note that add_option does not have to be called if you do not want to use the $deprecated or $autoload parameters).

$option_name

Required (string). Name of the option to update.

$newvalue

Required. (string|array|object) The new value for the option.

Administration Panels

Assuming that our Plug-in has some options stored in the Wordpress database (see section above), we will probably want it to have an administration panel that will enable your Plug-in users to view and edit option values. The methods for doing this are described in Adding Administration Menus.

6.12.8 Internationalizing Your Plug-in

Once you have the programming for your Plug-in done, another consideration (assuming you are planning on distributing your Plug-in) is internationalization. Internationalization is the process of setting up software so that it can be localized; localization is the process of translating text displayed by the software into different languages. Wordpress is used all around the world, so it has internationalization and localization built into its structure, including localization of Plug-in.

Please note that language files for Plug-in ARE NOT automatically loaded. Add this to the Plug-in code to make sure the language file(s) are loaded:

```
if(!load_plugin_textdomain('your-unique-name','/wp-content/languages/'))

        load_plugin_textdomain('your-unique-name','/wp-content/plugins/plugin-
name/location-of-mo-po-files/');
```

First line will try to load language file from /wp-content/languages/, so users can add their own files and they will be preserved when updating plugin. Second line will fall back to the location inside your plugin folder.

To fetch a string simply use __('String name','your-unique-name'); to return the translation or _e('String name','your-unique-name'); to echo the translation.

It is highly recommended that we internationalize our Plug-in, so that users from different countries can localize it. There is a comprehensive reference on internationalization, including a section describing how to internationalize your plug-in, at I18n for Wordpress Developers.

These are in broad terms considerations when designing a customized plug-in for a Wordpress site, but as there are at the time of writing 14.000 plug-in only on Wordpress' own site alone, we can assume that there are enough plug-in to satisfy basic design needs that might be faced in the design process.

6.13 WIDGETS IN WORDPRESS

A widget is what Wordpress calls a program or function that we can move freely around the Wordpress designated widget areas.

Widgets in Wordpress function like plug-ins that can be used to add elements to a site like calendars and navigation options. Widgets are by their nature more flexible as they can be moved around the site layout to add functionality and do not required to have predetermined place in the layout of a page.

A typical Wordpress site usually contains two widget areas with the most common being the side bar that typically in a blog configuration includes the widgets; search, recent post and a tag clod widget. A Wordpress site can include more widget areas and it is the good to take into account this when designing a project as widgets can be used to make the in-site navigation more intuitive for example or provide map or calendar functions easily. A good example is the use of html widgets that enable the displaying of ads linked directly from a source address.

In this chapter we will look at widgets that are included in the standard installation package that Wordpress comes in and we will also look at building a basic widget.

6.13.1 Examples of Default widget included in the installation package

Here are some default widgets that are included in the standard installation package; we will look at these as an example of what can be done utilizing them. At this point there is over 3000 widgets on the Wordpress Codex page. Accessed 2011 May [http://codex.wordpress.org] and this means a developer have many options to choose from when designing pages.

Here are the default themes that can be enough for a simple project or blog but are a good starting point for an example.

Archives - displays archive links for each month that has posts.

- Title — description that appears over the list of archive links.

- Show post counts — if checked, this box causes a count of the number of posts for each archive period.
- Display as a drop down — if checked, this box causes the archives to be displayed in a drop-down box.

Calendar - displays a calendar of the current month. Dates appear links if there are posts for that day.

- Title — description that appears over the calendar

Categories - displays a list of post categories as links to those posts.

- Title — description that appears over the list of categories.
- Show as dropdown — if checked, this box causes the categories to be displayed in a dropdown box.
- Show post counts — if checked, this box causes the count of the number of posts to display with each category.
- Show hierarchy — if checked, shows parent/child relationships in an indented manner.

Links - displays list of links (blogroll) separated by category.

- No configuration items.

Meta - displays links to meta functions such as Site Admin, Login/out, Entries RSS, Comments RSS, and WordPress.org.

- Title — description that appears over the list of meta links.

Pages - displays a link to each Page.

- Title — description that appears over the list of pages.
- Sort by — select the order to sort the list of pages. Choose Page Title, Page Order, or Page ID from pulldown box (this was added at 2.2.1)

- Exclude (Page IDs, separated by commas) — enter the Page ID(s) to exclude, separating each Page ID with a comma (this was added at 2.2.1)

Recent Comments - displays a list of the blog's most recent approved comments.

- Title — description that appears over the list of recent comments.
- Number of comments to show: (at most 15); enter the number of comments to be displayed.

Recent Posts - displays list of the blog's most recent posts.

- Title — description that appears over the list of recent posts.
- Number of posts to show: (at most 15) — enter the number of posts to display.

RSS 1 - displays an RSS Feed. Using RSS Widgets lists several feeds to use with this widget.

- Enter the RSS feed URL here — enter a complete feed URL, e.g. http://wordpress.org/development/feed/
- Give the feed a title (optional) — enter a description that appears over the list of feed items
- How many items would you like to display — enter the number of items from the feed you want displayed.

Search - displays a Search box to enter text to search your blog. A submit button is also provided.

- No configuration items.

Tag Cloud - displays list of the blog's top 45 used tags in a tag cloud.

- Title — description that appears over the tag cloud.

Text 1 - used to enter HTML, Javascript, or just plain text. Using Text Widgets details a number of possible uses for text widgets.

- Title area — a description of the text widget

- Text area — use this area to enter text, valid HTML, or even valid Javascript.

These are the default widgets and we should pay special attention to the Text 1 widgets as this enables us to deploy script, and to the search engine that should be included in every website that exceeds 100 pages of information. The combinations that can be created using different widgets are endless, but it should always be taken into consideration that deploying a lot of widgets does not make a project better and it should be noted what is used should always compliment the design and not rule it.

6.13.2 Tutorial

At the most simplistic level, a Widget now takes the form of a Class which contains 4 functions. The basic structure is shown below. Accessed 2011 May [http://codex.wordpress.org].

```
class SampleWidget extends WP_Widget

{

  function SampleWidget() { }

  function form($instance) { }

  function update($new_instance, $old_instance) { }

  function widget($args, $instance) {}

}

register_widget('SampleWidget');
```

In this example we create a widget called "SampleWidget" and tell PHP that it is going to extend (inherit from) the WP_Widget class. It is the WP_Widget class which makes your widget function, have settings and allow multiple instances so you don't have to.

The first method, SampleWidget() in the example, contains the code that is run every time the widget is loaded – either when activated, used on a page, updated and so on. This should take the same name as the class we are creating, i.e. copy the code above and change every occurrence of SampleWidget with the name of your widget (remember no spaces allowed).

The second method, form(), contains the settings page on the Widgets admin screen. This method is always called form and never changes.

The third method, update(), is called when the user click on "save" from the settings page on the widgets admin screen. This will automatically handle saving to the database of options. See the example below for basic operation, we can use this to read in and validate user input.

Finally the widget() function contains the code that will be rendered to the sidebar when the widget is added.

Here is the instruction for constructing a "Hello World" widget.

Example

```
class SampleWidget extends WP_Widget

{

  function SampleWidget()

  {

    $widget_ops = array('classname' => 'SampleWidget', 'description' => 'My Sample Widget Description' );

    $this->WP_Widget('SampleWidget', 'My Sample Widget', $widget_ops);

  }

  function form($instance)

  {

    $instance = wp_parse_args( (array) $instance, array( 'title' => '' ) );

    $title = $instance['title'];

?>
```

```
 <p><label for="<?php echo $this->get_field_id('title'); ?>">Title: <input class="widefat" id="<?php echo $this->get_field_id('title'); ?>" name="<?php echo $this->get_field_name('title'); ?>" type="text" value="<?php echo attribute_escape($title); ?>" /></label></p>

<?php

 }

 function update($new_instance, $old_instance)

 {

  $instance = $old_instance;

  $instance['title'] = $new_instance['title'];

  return $instance;

 }

 function widget($args, $instance)

 {

  extract($args, EXTR_SKIP);


  echo $before_widget;

  $title = empty($instance['title']) ? ' ' : apply_filters('widget_title', $instance['title']);

  if (!empty($title))

   echo $before_title . $title . $after_title;;

  // Do Your Widgety Stuff Here...

  echo "<h1>Hello World</h1>";

  echo $after_widget;
  }

}

register_widget('SampleWidget');
```

In the SampleWidget function (technically called the constructor) I created some configuration options which will help Wordpress (and the users of your widget) showing a friendly title and description.

The form method takes in the current settings as the $instance so you can use this to output the current settings to the user so they can update.

The update method does not do anything too complex, if you add another setting property add a line in here for it otherwise it won't get updated.

Finally the widget method will generate the output for the widget. Notice the $args parameter, this is exactly the same as the $args parameter on the old style widgets. $instance contains the settings for this particular widget. This is a basic widget setup.

These are the basic widgets demonstrated with a basic tutorial included for building one, it should be noted that there is a big base of pre-made widgets out there and that there might never be the need to build one as we can deploy ready-made widgets that fall under the creative commons license.

6.14 BASICS OF BUILDING A WORDPRESS SITE

The basics of building a Wordpress site are very complex or very easy, it can be argued as such. What is needed to setup a Wordpress is compatible hosting, this enables full control of the site and the option to theme it, and this will also enable access to the root directory with an ftp program that makes it easier to ad media in some cases and to update the code or implementing changes without utilizing the Wordpress dashboard. This also gives full administrator and creative control to the developer and it is now possible to modify the Wordpress package to what it is needed to be. It good to note that for example Wordpress offers free hosting for Wordpress sites and blogs but limits the access so it’s impossible to add customized themes and in this way limiting the freedom that developers need to build their projects and deploy them, also no root directory access.

The installation of Wordpress is a fairly automated process on most hosting services that have so called “one press installation” features enabled to install CMS systems. After the installation is complete the administrator is sent the user name and the randomly generated password for the wp-login.php that enables access to the control panel. If we chose to use a independent or otherwise different hosting model as for example running it from a server that we own we will have to install and link Wordpress to the MySQL.

When the Wordpress package has been downloaded and installed we are presented with the default theme twenty ten that has a basic layout and is to a degree customizable. At this point we could go and by implementing the different techniques that have been discussed in previous chapters to radically customize the default theme or build a completely new one but as we are covering building a basic Wordpress site we will only cover the bare basic of setting up a site. Some could argue that there is always the need to do customized graphics or other specific coding, but as we might only need to setup a blog for a project or make a online CV or joust make a site of a holiday trip a slightly modified default theme with a couple of plug-ins is usually enough.

As we by this point should have a fully working Wordpress site, our basic Wordpress tutorial is over, Setting up the basics is a straightforward and simple process, the time consuming part is design and customizing the Wordpress package to suit different project and customer needs.

A good point from which to star using and experimenting with Wordpress is to explore the dashboard in Wordpress and the different options that range from customizable menus to installing different plug-ins and exploring how much the basic twenty ten theme can extended and modified joust by using the internal settings of Wordpress, then when we have a good understanding of the basics we can set higher design goals.

6.15 WORDPRESS SECURITY

It is good to note that as of 2011 the amount of malicious software on the internet has tripled since 2005 (f-secure study 2011). It is advisable to take this into account when designing in Wordpress and try to implement safety features.

The different forms of attack out there are numerous and vary but they usually share the common characteristic of attacking only soft and unprotected targets. This means that they only go for unprotected targets and because they ping millions of IP addresses they will skip targets that are hardened for attack.

We will go through some key points of security that can be found in the article on wordpress.org codex: Hardening Wordpress. Some of the first basics are that the computer we are using should be secure, if we for example have a key logger on our computer everything that is written after this is useless and pretty much anything we will do after this is useless.

Remember to back up the data base with steady intervals to always have a secure and up to date copy handy if the data base is corrupted or destroyed, this is not only to protect a site from hackers but also other misshapes that might occur. There are some good plug-ins that do this easily for us.

The other important thing to remember is to always have the site up to date, Wordpress does not release security fixes to older version so keep the site and the plug-ins up to date to secure the sites integrity.

Good points to remember are:

1. Limiting access: Making smart choices that effectively lower the possible entry points available to a malicious person.

2. Containment: If a weak point in your installation is found by a malicious person, the system should be configured to minimize the amount of damage that can be done once inside your system.

3. Knowledge: Keeping backups, knowing the state of your Wordpress installation at regular time intervals, documenting your modifications all help you understand your Wordpress installation.

Always follow what's happening on the site and know what has been installed. One of the most important things to remember is to have a strong password protecting the site. Some vulnerability's can be avoided by good security habits or protocols. An important element of this is passwords: do not use your own name for your password, do not use a dictionary word (from any language) for your password, and do not use a 4 character string of numbers as your password. Our goal with the password is to make the search space as large as possible, so using numbers and varying capitalization all make it more difficult, statistically, to brute force a password. This is particularly important if we do not rename the administrator account. In that case half the puzzle is already solved for malicious users as they know what username will give them significant privileges to edit files and databases. Many automatic password generators can be found on the internet and used to create secure passwords. A strong admin password is necessary not just to protect the site/blog content; but also to protect against a hacker for instance uploading a script or doing other damage which could result in a compromise of the entire Wordpress installation - in other words if a hacker gains access to the admin area they can do a lot more damage than simply changing the content.

These are some of the basics that can be deploy to protect the Wordpress sites, there are a myriad of other techniques and ways of securing Wordpress, renaming files, hiding sections under different names, then there is the writing permissions that can be modified. Also one of the ways is adding a second layer of password protection to the wp-admin.php file with the server but as with all these techniques they might conflict with the server, or create error, especially the second layer password in some Ajax driven servers.

Consider these guidelines in the design process and keep the Wordpress site and plugins up to date, more importantly be consistent in the security protocols and many of the problems discussed in the article above should not percent a problem.

6.16 ARTISTEER (THEME GENERATOR)

Artisteer is a commercial Theme generator that has the capacity to publish Wordpress, Drupal and Joomla themes in ready-made packages that are install ready. The software has a setup that could be called a marriage of a simplified Microsoft Word and a very bare bones Photoshop that enables real-time editing of a theme for Wordpress where you can see in real-time all the changes that are being made on the site, Artisteer writes the code for this automatically.

These kinds of programs are often severely limited in what they can do and usually just creates blank constructs that the developer can use as a platform to build their code on and theme it to their liking but here Artisteer brakes the nor by being quite intuitive and writing code that is clear and well marked enough to be easy for the developer to go in and do changes in a very finished product.

Artisteer is a good tool to use to create prototypes and demos of sites for research and demonstration for a client; it saves time as we can produce mockups of sites in a very rapid pace. After this the prototypes can directly be further developed to a finished end product.

Some of the features are limited but can be bypassed by coding it in separately and by modifying the installation package. Other features do come in handy as they include typography tools for the sites different subcategories that can be very time consuming doing by hand.

Artisteer is a good tool and it greatly increases the pace at with we are able to try new ideas and publish; we would recommend it to every design as a good tool to augment the design and work process.