How to Manage Business Processes Using the Internet

Rapid Application Development on the XOOPS website platform

By Freeform Solutions

July 2005

Table of Contents

Introduction

What's all this about?

The goal of this document is to explain our approach to managing business processes using the XOOPS website platform. This document is not a step-by-step guide, and it's not an instruction manual. Instead, it is written from a general point of view and attempts to give an overview of the approach and possibilities. We try to avoid jargon, but some jargon is inevitable. A glossary is included at the end.

What is a business process? It is simply a procedure carried out by one or more staff or volunteers in order to achieve a goal for an organization. The term 'business' doesn't mean we're talking about a commercial venture. Not-for-profit organizations, web-based communities, any group of people working together for a common purpose, they all have business processes and business goals, even if you don't read about them in the financial papers.

To manage these processes, we have developed a set of open source tools for building data-entry applications within XOOPS-based websites. Fundamentally, most interactions with websites can be viewed as data-entry applications. A discussion forum has an input side (the "new post" form) and an output side (the list of posts in a topic). You could use our tools to make a discussion forum, but you probably won't because specific tools already exist which do a great job at providing discussion forums.

However, in any given organization or community, there are always a wide variety of unique activities and business processes that are carried out every day, for which no specific solutions exist. The tools and techniques described in this document provide a way to build these custom interactions into a website, to improve the efficiency with which they are carried out, to provide new ways of working with the information that gets generated, and to better communicate and share results with various stakeholders within an organization, and outside it too.

How is this different from...?

Our tools are generalized to address a wide variety of information management tasks. The foundation of our approach is to find the right way to use the tools to represent and support whatever business process is at issue. With a full understanding of these tools, an application designer/developer can operate at the functional level of the processes themselves, rather than at the level of the technical implementation.

This technique fosters rapid application development and a more accurate representation of the business process. Also, because the time-to-result is reduced, when compared to building custom solutions from the ground up, there can be significant cost savings in using this approach.

To put it another way, these tools let non-programmers create advanced applications. All that is required is a clear understanding of what information management tasks are involved in the process (what data needs to be recorded, by whom, when, for what purpose, to be received by whom, when, summarized how, etc). The challenge then becomes simply mapping those requirements onto the capabilities of the tools at hand.

Some programming is necessary for advanced scenarios. But basic ones can be handled expertly "out of the box." And when programming is necessary, it is limited to only the high level interface issues, because all the low level work of storing and accessing data is handled by the tools automatically.

We know that none of these features are unique or unprecedented in the world of business IT. However to our knowledge there is no other set of tools in the open source portal/CMS world that provides this capability.

The XOOPS Platform

XOOPS is a powerful tool for building interactive websites. It is powerful because it provides many basic services that all interactive websites need, while allowing the website's creators to add in whatever customized services their particular site requires.

Our application development tools are for use within XOOPS-based websites. They take advantage of some of the basic features that XOOPS handles all by itself, such as handling user logins, and collecting users into groups to assign permissions based on group membership.

XOOPS does not provide any real "content" functionality by itself. Most information that visitors interact with in a XOOPS-based website is provided through the modules that have been installed for that website.

Beyond the Basics: Modules

To add specific functionality to a XOOPS-based website, web site creators must install modules in the XOOPS platform. Almost every page that a visitor sees in a XOOPS-based website is generated by a particular module that provides that particular kind of content.

For instance, a photo gallery in a website would be provided by a module specifically designed to store photos and present them in predefined collections, etc. Without such a module installed, you could not have a photo gallery in a XOOPS-based website. Most importantly, the specific features of the gallery, such as how the images are laid out, how website administrators must add new images to the gallery, etc, are almost all determined by the module itself, not XOOPS.

The same is true for any kind of content, whether it's forum postings, announcements, multi-page articles, files for downloading, or polls. The module that handles each type of content is responsible for the management and presentation of that content.

Formulize

Ad hoc form creation and reporting

The module-based approach to building a website runs into a problem when you start to look at the customized business processes that all organizations have. To handle all the processes, you might ideally require a particular module for each – except creating all those modules from scratch is totally impractical.

Our solution to this problem begins with Formulize, the foundation of our toolset.

Formulize is a module for XOOPS which allows you to design forms for use in your website. You have complete control over the way people interact with the forms, what questions the forms ask, and who can see the submitted information. These forms are the backbone of the succeeding layers of tools. By making customized forms, you can create all kinds of specialized applications for an organization, without having to build entire modules suited to each task.

Form Types

There are two basic types of forms: single-entry forms and multi-entry forms.

Single-entry forms are, as the name implies, forms where a visitor can only make one entry. If someone returns to the form after having filled it in, they have the option of updating the information they provided, but they cannot make a second entry. Common examples include personal profile forms and conference registration forms.

Multi-entry forms do not have the restriction that single-entry forms do. Visitors can create as many entries as necessary in these forms. Common examples include status reports, activity logs, and appointment forms. Multi-entry forms are also commonly used when an organization has a large set of internal information that needs recording as a database – such as HR details or inventory lists.

Form Elements

Each form can contain a variety of different elements: text boxes, drop down boxes, list boxes, check boxes, and radio buttons. Specialized elements are available too, such as date selection boxes that have a popup calendar attached to them.

Elements are added to a form through a simple interface in the administration part of the XOOPS website. The designer can specify various features of each element, such as the size of text boxes, and the default options for radio buttons. Elements can also be reorganized into a different order, and descriptive text can be added to different parts of a form to explain how to fill in the form.

Permissions

Designers can closely control what actions visitors can perform with a form. Common actions are adding entries, updating entries, deleting entries, and adding entries on behalf of other people. Every conceivable action a visitor or administrator can perform with a form, can be individually controlled through the permission system, thereby allowing for the creation of highly customized modes of interaction with forms that match whatever internal procedures an organization might have for working with information.

Scope

There are also special permissions that control the breadth of information someone has access to. Everyone has access to information that they themselves have entered into forms. Beyond that, people need special permission to view information that other people in their group have entered.

With proper design of the group structure in the XOOPS website, and proper assignment of the scope permissions, it is possible to create situations where, for example, all staff can enter their own expense reports and review their own entries, but only senior staff can view everyone's expense reports together for purposes of aggregating the data.

There is also a permission for global scope. This setting is relevant in situations where there are a large number of groups. Perhaps all departments in an organization have their own groups in the XOOPS website. With the group-scope permission, senior staff in each department would have the ability to review all entries in a particular form that have been made by all staff *in their department*. But with the global-scope permission, it is possible to allow top-level staff to review and aggregate all the entries by all staff *in all departments*.

Summary View of Entries

Unlike many other form creation tools, Formulize also includes a powerful summary view of all the entries that have been made in a form. The summary view includes the following features:

- Sort entries by any field
- Search for partial words or phrases in any field
- Search with specific operators, including NOT, LIKE, >, =, <, and use AND and OR to combine multiple search terms into complex queries.
- View the complete details for any entry
- Change the summary view to include any fields used in the form
- Delete one or more entries in a form
- Export the data in the summary view to a spreadsheet format (.csv file)

- Perform calculations on the data in any field, or on subsets of that data, including:
    i. Sum total
    ii. Average
    iii. Minimum value
    iv. Maximum value
    v. Count the number of entries and number of blanks
    vi. Calculate the percentage breakdown of values in a field (ie: for the values in a field called *Fruit:* 55% apples, 25% oranges, 20% bananas)

Custom Views of Entries

Additionally, any particular view of entries that someone has created using the sorting, searching and calculation controls, can be saved for easy access later. For instance, in an expense report form, you could search for values greater than $1,000 and sort the resulting entries by the name of the employee. Then you could save that view so that you can easily refer to it again without having to recreate it.

There is also a permission that allows someone to publish a view to other people. This has great benefits in terms of sharing information among a group – a finance person could save the custom view of the expense reports described above, and publish that to all executive staff.

Publishing a view can be particularly valuable for making summary information available to people who do not normally have the ability to review all the data that has been entered in a form. For instance, suppose everyone in a group of volunteers fills in an activity report after they complete tasks, but each individual does not have access to the entries made by the others. A volunteer coordinator could collect together data from everyone's entries in a customized view, and then publish that view back to all the volunteers. That way, although the volunteers cannot normally view each other's entries, they would have access to whatever aggregate information the coordinator wants to make available.

Frameworks

Joining forms together for more complex data collection and presentation

The data entry forms you can create in Formulize are very useful. The forms' customizable options and permissions, together with the flexible summary view interface, give application designers a lot of range in terms of the business processes that they can model.

But that range only extends so far. Formulize has a feature called *Frameworks*, that adds several more dimensions to the capabilities of basic Formulize forms. Fundamentally, a framework is a summary of the data contained in a form, and a description of the relationship between that form and any other forms.

Specifying this information in a framework allows two main things: it allows two or more forms to be displayed together as if they were a single form, and it allows the information entered into a form or forms to be collected together for presentation and use outside the Formulize module.

Unified Presentation of Forms

Forms that have been related in a framework may be displayed together as a single form. This is an option for any two forms that are part of a framework. This allows for much more complex form design and therefore more complex business processes can be supported.

For example, using just the basic features of Formulize forms, the simplest way to setup a basic expense report would be to have each entry in the form represent a single expense. If an employee has an expense, they fill in the form to record it. Two expenses – fill in the form twice.

What if the employee goes on a business trip and has ten expenses, and you don't want to lose track of the fact those expenses are related to a single trip? Also, your employees probably don't want to fill in the form ten separate times. The basic form fails you in this case.

But you can solve this problem by creating an additional business trip form, and relating it to the basic expense report form in a framework, and then turning on the unified display option. The new unified form would have space for details about the trip itself (the new business trip form), plus space for a variable number of expenses (the original expense report form). Now, when the employee comes back from the trip, they fill in one form related to their trip, and all the expenses are recorded as part of that trip.

Unifying Data in Two Forms

Besides allowing for the unified display of multiple forms, the other major use of the *Frameworks* feature is to relate data entered into one form to data entered into another, so that all the data gets treated as a single set of information. These sets of information can be viewed using the standard summary view available for basic forms, including all the sorting, searching and calculation options available there.

For instance, for a volunteer profile form and a volunteer status report form, a framework could specify that status reports made by a given volunteer should be joined together with the profile information for that same volunteer. So if you have a volunteer named Pat who has submitted three status reports, then Pat's profile information such as name and address would be included with the information from each of Pat's status reports.

Forms can be related together in other ways besides the entries having been made by the same person. For example, an activity booking form that some staff use to book activities at schools could be related to an activity log form that other staff use to record the details of the activities after they happen. Although two different people have made the two entries, they can still be related, based on the fact that the data in both forms is about the same

specific activity. This would allow information from the booking, such as the time the activity happened, to be joined together with information about what happened during the activity, such as how many students were present.

Using and Presenting Data from a Framework

Every framework contains a summary of the data that is contained in each of its forms. This summary allows the data in the forms to be accessed and presented elsewhere in the website, besides the Formulize module. Data can also be accessed and presented on entirely different websites, even if they're not based on XOOPS.

The data summary assigns a 'handle' to every piece of data collected by a form. A handle is simply a unique word or series of characters that can be used to pick out that piece of data. For example, for an office profile form, you could create a handle called *phone* for the office phone number. You can then display the office phone number elsewhere simply by referring to *phone*. This technique allows you to create, for instance, a contact list based on data entered into basic Formulize forms that have been described in a framework. The contact list could even be displayed on a separate publicly available website as part of the 'Contact Us' section.

When someone changes the office phone number next year, all references to the phone number elsewhere are automatically updated, since the framework simply points *phone* to the part of the form where the up-to-date phone number is stored. If you are dealing with information that changes more regularly, this becomes an incredibly powerful tool for creating 'dashboards', highly customized and dynamic views of information particular to each type of visitor to your site.

For example, if you have staff that need to monitor activity reports, then when they login you can list the five most recent activity reports directly on the front page of the website for them. You can do this simply by creating a framework that includes the activity report form, and then referring to the activity report data from the front page of the website.

Pageworks

Creating custom pages and views of data inside XOOPS

In order to follow through on the promise of the *frameworks* feature in Formulize, and make it truly easy to use all that data from a framework inside XOOPS, we have created another module called Pageworks. Pageworks is a module that lets you create customized pages or blocks inside XOOPS. Each Pageworks page can refer to one or more frameworks and all the data in the referenced frameworks is available within the page, and can be formatted and presented however the application designer sees fit.

For instance, going back to the example of the office profile form above, if you want a list of the phone numbers for different offices on this page, then you would refer to the framework that contained all the office profile information, and for each office you would use the *phone* handle to display the phone number in the page.

The actual layout of the page is handled using basic HTML, while special references to the framework involve a few simple commands in the PHP programming language. With very little time and effort preparing the page, the application designer can build a highly customized interface that is based on the live data in a framework. Since Pageworks handles all the integration with the framework and with the underlying data submitted through the forms, the application designer can focus entirely on building the interface and doesn't have to concern themselves with low level issues such as where the data is, how to retrieve it and whether it is in the right format.

Including Forms and Summary Views in Pageworks Pages

The ability to use Pageworks to customize the interface that users see extends further than just working with data from frameworks. Application designers can actually embed forms directly within a Pageworks page.

Embedding forms is very simple and requires only one special PHP command plus the name of the form or framework being embedded. When visitors go to that page of the website, the Pageworks module interprets the special PHP command and integrates with Formulize to display the form and handle the saving of any data that the visitor submits.

The same can be done for summary views of entries. A Pageworks page can contain a simple PHP command that will include a standard summary view of the entries in a form or in an entire framework.

All these options give tremendous flexibility for application designers. They can create interfaces that contain forms, and that also highlight relevant data that has been entered into the form so far. For instance, an interface for recording attendance could be made up of 12 forms, one for each month, and in addition to providing access to each month's form, the interface could also present a sum total of vacation days taken so far this year by presenting combined data from all 12 forms.

Direct Access to Data in a Framework

As mentioned above, data in a framework can be presented on completely separate websites from the XOOPS-based website where the data was entered. This is because it is a relatively simple matter to connect with a framework even from a separate website. Only a few lines of custom PHP code are required, at which point all the data in the framework is available for use in the separate website, just as if it were being accessed from within a Pageworks page.

The Pageworks module simply carries out the process of connecting with the framework automatically, removing the need for the application designer to do so. But by including a few lines of custom PHP code, the same effect can be achieved literally anywhere. At that point, whatever custom interface is necessary for working with or presenting the data can be created using PHP. The very same references to the framework and handles (ie: *phone*) as you would use within a Pageworks page can be used here too.

Conclusion

Formulize and Pageworks work together to provide total control over the collection, storage, analysis, sharing, and presentation of information. Because they operate almost entirely at the functional level of the business processes themselves, rather than at a technical programming level, and because of the options and high degree of flexibility they provide, an application designer can model virtually any conceivable activity, simply and comprehensively.

The challenge for the application designer now becomes determining how a given business process should be represented within the capabilities of these modules. Making that determination requires a high degree of understanding of the business process on the part of the designer. With that understanding, the designer can then break down the business process into its constituent parts related to who enters what information when and how, what supporting information they need to be presented with in order to carry out that task, who needs to have access to the information once it is submitted, and how the submitted information relates to any other data.

Once the business process is understood from the point of view of the information itself, developing the application is simply matter of identifying the form options and elements required, the possible framework configuration necessary, and the types of custom interfaces that might be needed to support the flow of information through the business process.

By focusing on this high-level process, Formulize and Pageworks provide a rapid application development system for creating these kinds of complex, customized data-entry applications. For more information, contact Freeform Solutions at info@freeformsolutions.ca.

Thanks for reading!

Appendix A: XOOPS – an overview

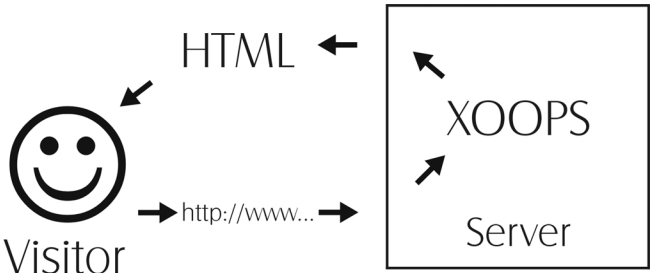
Most websites (or most old websites, from last century, at least), are built with HTML. Most people have probably heard of HTML. It's a markup language (that's the ML part), which lets you add some descriptive information to plain text. Your web browser reads that descriptive information and determines how to display all the elements on the page, based on whether each element is marked as a heading, or paragraph, or part of a table, or part of a list of items in a menu, etc.

The basic process looks like this:

The visitor requests a page, and the server sends back the HTML for that page.

HTML is a wonderfully elegant and simple system, and the internet would not have exploded in the mid '90s without it. But it's also lousy for building anything more than simple websites, especially websites with interactive features that behave differently depending on who is visiting and what the visitor has done in the website so far.

To help build more complex websites, people started creating new tools that provide real interactive features in a website. XOOPS is one such tool. (XOOPS is an acronym that stands for eXtensible Object-Oriented Portal System). XOOPS is not a replacement for HTML, it's like an intermediary between the visitor and the HTML that the server sends back:



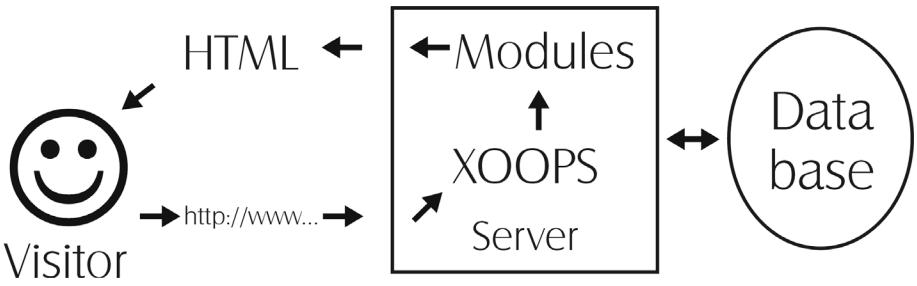
XOOPS relates information that is stored in a database to details about who the visitor is and what level of permission they have been given in the website. By taking all that into account when the visitor requests a web page, XOOPS is able to customize the HTML that gets sent to the visitor, depending on what information is appropriate for that person.

XOOPS itself is a platform for building interactive websites. But by itself, XOOPS does not provide the ability to create any useful or interesting web pages. Instead, XOOPS allows the creators of the website to add in whatever kinds of pages and features they need, by

installing particular modules in XOOPS. XOOPS stores all the information about how a website is configured, and most of the content for the site, in a database.

For instance, you can add a discussion forum to a XOOPS-based website by installing a discussion forum module. Once that is done, the webmaster then has a whole series of new options for configuring their site, all related to presenting discussion forums. Other common modules are for news and announcements, long article management, and file downloads.

So, here's a more complete picture of how a request for a web page gets turned into HTML for the visitor's web browser:



XOOPS Features

Despite the independence of the modules, the XOOPS core system provides a number of site wide features that each module can integrate with in its own way. For example, there is a site wide **Notification system** that can inform registered users when new content appears in any part of the site. The notification system itself is part of the XOOPS core, and each module simply makes use of it in its own way.

This approach ensures consistency in site wide features, because modules simply "hook in" to core functionality when necessary. This approach also streamlines the overall implementation since different parts of the site can rely on the same programming and design work, without having to duplicate effort to achieve the same results.

Besides notifications, some of the other major core features are:

- **A site wide search system.** The core system provides a search capability, which can accept simple text search terms, and allows the user to search for all or any of the terms (Boolean AND or OR). Results are sorted and displayed according to which part of the site they are found in.
- **A user profile system.** The core system stores standard information about users, such as their location, their name, e-mail address, number of postings they have made in the site, submissions they have made to the site, etc.
- **User presence and community tools.** There are several ways the system displays information to visitors about who else is online, what recent activity has happened in the site, and which people have been participating most actively. These all help to build a sense of community around the website.

- **Account creation and a login/logout system.** The core system handles the creation and tracking of user accounts, including the login and logout processes. Related to this are security features that prevent unregistered users from accessing private content.
- **A 'Groups' system for managing many users together.** Users in the site can (and must) be grouped together for administration purposes. All rights to access content and perform actions in the site are controlled through the Groups system, allowing you to clearly define what different types of users can do in the site. You can create an unlimited number of groups, and individual users can belong to more than one group, allowing for situations where an individual is part of multiple, distinct audiences with different permissions.
- **A 'comments' system.** Registered users can post comments about a piece of content in the site, such as a file available for download, and the comments appear alongside that piece of content, enriching the value of the information.
- **Private messaging system.** Registered users can send private messages to each other within the website, which can enhance the value of the site as a community development tool.

Page Layout in XOOPS

XOOPS provides the ability to use a common template for the entire website, ensuring consistency in the look and feel of common elements. The proper use of a template also allows administrators to have single changes appear across the entire site at once.

Individual page elements also have their own templates. This allows specific items in the site, such as menus, lists of users, lists of articles or files, etc, to appear on several different pages of the site while retaining the same structure and appearance.

The template system is also isolated from the programming code and database systems. This ensures that changes to the template affect only the appearance of information, and not the underlying logic that manages the information, nor the information itself.

Wide Variety of Modules

XOOPS is a good system to work with, in large part because of the variety of independent modules that are available. Modules are the parts of a XOOPS site that provide specific functionality. There are modules for virtually everything people use websites for. A few examples: discussion forums, calendars, article publishing, news and announcements, FAQs, e-commerce, file downloads, photo galleries, tests and quizzes, surveys, advertising management, real-time chatting, and classified ads.

If there are specific needs that are not quite met by the best available modules, it is entirely possible to customize an existing module. Since each module is open source, just like the core system, there are no restrictions or licensing fees on doing modifications.

Appendix B: Group Structure in a Complex XOOPS Site

All XOOPS-based websites organize users into a series of groups. Exactly which groups these are, who is in which ones, and most importantly, what overlap there is between groups, is entirely up to the people building the website. XOOPS simply provides the capability of organizing users into whatever groups you need.

In a complex site, one modeled after the internal structure of a community or organization, the number of groups can becomes very large. For example, if an organization has offices in half a dozen different cities, and there are different information needs for, say, the marketing department as opposed to the engineering department, then that organization would need at least 12 groups, probably more (a marketing group and an engineering group for each geographic location).

Furthermore, in order to ensure that certain people (managers) have the administrative control they need so that they can monitor work and information related to their subordinates, you probably need at least two groups per location: one for all staff, and one for just the managers.

The idea is that all staff, including the managers, are members of the 'lower' group intended for everyone. That way, discussion forums, data-entry forms, and other tools can be made available to everyone by simply turning on those tools for one group, the all-staff group.

However in order to give managers extra abilities that regular staff do not have, the managers would need to be members of the managers group *as well as the staff group*. The fundamental principle here is that the content for all the staff, including managers, is associated with the 'lower' group, while any special permissions intended just for managers are associated with only the 'higher' group.

This way, when managers are interacting with content that is available to everyone, their membership in the 'higher' managers group allows them to perform special actions with that content – actions that regular staff cannot perform. Freeform Solutions has developed custom extensions to the XOOPS platform to make it easier to manage the very large number of groups that can result when setting up a complex organization in this way.

Appendix C: Costs of Using Open Source Software

XOOPS is an open source product, and so we would like to address some confusion that exists regarding open source software. Formulize and Pageworks are also open source products, although any specific applications that one builds with them are not part of the underlying software. This is roughly analogous with the fact that documents you write in Microsoft Word are not themselves part of the word processing software, and the rights governing their use are not owned or controlled by Microsoft.

The most fundamental confusion about open source software is that because most of it is available free of charge, some people believe this means there is no cost to using the software. This is not true.

The term "open source software" simply refers to the method in which the software is developed. Open source software is software for which anyone using the software also has access to the underlying programming instructions – the source code – that makes the software work. The intention is that advanced users of the software will share any alterations, improvements, fixes and new features that they might add to the software themselves. This is fundamentally different from closed source software, like the kind you buy in a store, where the only people working on alterations, improvements, fixes and new features are the original creators of the software.

As you might imagine, open source software projects require a community in order to be effective. The fundamental point of participating in an open source software project is to share and share alike in the improvements to the software that are made available within the community. Usually a project manager or small group of people collect and integrate all the best changes that the community has made, and incorporate them into subsequent official versions.

This approach to development has a number of advantages for small organizations, and not-for-profit organizations. First of all, since the software is generally available free of charge, the *product cost* is infinitely lower than the cost of closed source software. Second, by leveraging the power of an open source community, a small organization can participate in and benefit from a greater development effort than it could carry on by itself.

In most large scale IT projects, no open source product is an exact match for the requirements that have been identified. Therefore some programming and development work to customize the software is usually required. But this is no different from most closed source products, for which most vendors also sell customization and implementation services. So it would be incorrect to view such costs as unique to open source software. The largest difference here between open source and closed source may be that implementing open source products probably requires more changes to the programming source code, whereas implementing closed source products may require more configuration of the software (since closed source products do not allow changes and modifications to the

source code, by definition, you cannot therefore expect that making programming changes will be part of the implementation). At the end of the day, there will be implementation costs to both approaches.

Most importantly, organizations that use open source software need to understand that they are participating in a community effort that is larger than their particular project. Any changes or customizations or new features that are added to a piece of open source software as part of their project, get shared free of charge with the rest of the community of people using that software. Since one of the fundamental benefits in using open source software is the community of people helping develop it, there is no benefit whatsoever in "hoarding" changes, unless an organization has such large development resources all its own that the efforts of the community would be trivial by comparison.

The idea of sharing changes that it cost money to make can be hard to grasp when people are used to thinking about software as a commodity that is bought and sold. When money is spent on making software, there is an understandable sense of possession that comes with that. However when money is spent on contributing to an open source project, what is really being bought is a service, not a product. The buyer is paying for their needs to become the number one tasks in the open source project. The buyer gets their needs solved, having only financed the most immediate and specific features that are now part of the software. Since all the rest of the work in building the software came free of charge, this still makes participating in open source software projects an extremely cost effective approach to managing software in an organization of limited means.

Glossary

Application

By itself, computer hardware is not useful for anything except as a doorstop. Once you have an *application*, or a use, for the hardware, then the computer becomes useful.

So, an application is something that users interact with in order to carry out a task or procedure on a computer. On a desktop computer, your word processor, e-mail program and spreadsheet program are all examples of applications. On the Internet, certain websites provide you with applications to use. A banking website is one big banking application that you use to carry out banking tasks.

The tools described in this paper are designed to make it fast and easy to create applications that revolve around the submission, storage, analysis and retrieval of information.

Block

In the XOOPS website platform, a block is what you call an individual part of a web page, such as the header, or the menu, or a list of recent postings in a forum. You can control where on the page a block appears, and what it looks like in terms of colours, fonts and layout.

Business Process

Different business textbooks will give you many different answers, but for the purposes of our tools, we treat a business process as any series of specific actions carried out by one or more people to achieve a goal for an organization. These actions are well defined and repeated on a regular basis as part of the standard operating procedures for the organization.

For example, an organization that receives orders through their website will have an entire process related to fulfilling those orders. Order fulfillment is a major business process for that organization.

Business processes don't have to be so central or complex. Your organization might have a specific paper form that people fill in when they go on vacation. That form represents a simple business process that supports the smooth operation of your organization.

The tools described in this paper make it fast and easy to implement business processes in a XOOPS-based website.

Check Box

A type of form element which users click to activate. For example, a question on a form such as "What cities do you visit regularly?" might provide a list of a dozen or more cities, each with a checkbox beside their name. Users can check off their choice of one or more cities in the list.

Column

When the information that has been submitted into a form is displayed as a list, it is typically organized into columns and rows. Each column corresponds to one element in the form. Each row corresponds to one complete entry in the form. *Field* is a more technical term for column.

Content

In websites, content is often distinguished from the structure and layout of the website. In general, all text and images are content while the structure of the site (how the pages are organized in relation to each other) and the layout of each page (the appearance and organization of information on the screen) are not content.

Website platforms like XOOPS provide a systematic way to develop a strong site structure and to control the layout and appearance of pages. This makes using them an efficient way to control the costs and schedule on a project. Website platforms also provide relatively simple interfaces for managing and updating the content on each page. Often the managing of content is the responsibility of the organization that owns the website, while managing the structure and layout is the responsibility of a technical team or company that built the website.

Dashboard

Like the dashboard on a car, a dashboard in an application gives you a quick overview of the how the application is operating. For example, a dashboard for an inventory application would likely display any items where the stock was at critically low levels, and/or recently shipped items and/or recently received items. Exactly what details show up in a dashboard is up to the people building the application.

Database

A specialized computer system for storing and retrieving a large amount of information. Typically, besides the raw information in the database, there is also information about how the different pieces of information relate to each other.

Data-entry application

See *Application*.

Discussion Forum

Discussion forums are a common part of many interactive websites. They function like electronic message boards, where one person can post a message and then other people can write responses to that message. A series of messages is called a thread, or topic. Many threads can be ongoing at the same time.

The messages and threads are all visible as part of the website, which promotes more group discussion than typically found with e-mail messages that are sent to a group of people. To

help facilitate discussion, the forum area of a site is often divided into sections or categories, where related threads are gathered together.

For example, a website might have a category called General Discussion, one called New Products and one called Suggestions. Each category can contain one or more threads. This helps users find the discussions that they are most interested in, since they are not initially confronted with a huge list of threads. They can choose the relevant category first to narrow down the threads that they will see at any one time.

Dropdown Box

A type of form element which users can click on to display a large list of options. Only one option in the list can be selected at any one time. Dropdown boxes are very useful for large lists of items, since the full list is only displayed when the user clicks on the box. The rest of the time, only the selected item in the list is displayed, which saves space.

Element

See *Form Element*.

Entry

Every time someone fills in a form and saves the information they have specified, they have created one entry in the form. The information that a form collects is made up of many entries, and each entry is made up of information from all the individual elements in the form.

Field

Field is the technical term for the space in the database where information from one form element is stored. Each column in a list of entries corresponds to one field in the database. *Field* and *Form Element* are somewhat interchangeable terms.

Form

A page or screen in a website, or a part of a page or screen, which contains elements that users can fill in, similar to a paper form that they might fill in with a pen or pencil. Users specify information with each element and they can save all the information as one entry in the form.

Form element

A single part of a form, which contains one piece of information. For example, a box where a user can type their name would be one element in a form. The information specified for that element is stored in a database.

Forum

See *Discussion Forum*.

Framework

A Framework is a collection of forms as specified in the Formulize module. A Framework contains information about the data contained in the forms, as well as how the forms relate to each other. Frameworks are used to provide more complex data entry systems and to support more sophisticated data presentation options.

Group

Users in a XOOPS-based website are collected together into groups, in order to control the content that users see, and in order to control what permissions people have (what actions they can carry out in the website). Users can belong to more than one group, and different groups can have access to the same content, though usually there is not that much overlap. The content and information users have access to is determined by looking at all their group memberships together. So if a user is a member of a group called All Staff and a group called Management Staff, that user would then see content that is available to everyone, as well as special content available to just the managers. See *Appendix B* for more information.

Handle

A word or series of characters that uniquely identifies a form element within a framework. A handle is used to refer to the data that is stored in the database field corresponding to that element.

HTML

Hypertext Markup Language. HTML is the basic language in which web pages are written. It allows web page designers to specify certain features of how information in a page should be laid out on the screen, such as what parts of the page are paragraphs, lists, menus, tables, images, etc.

List Box

A type of form element which provides a list of options that users can click on. List boxes can be any length. If there are more items in the list than there is space in the box, then the box will scroll to show all the options. List boxes can be configured to allow users to select only one item from the list, or multiple items.

Module

In the XOOPS website platform, a module is something you install in your website in order to provide certain features and abilities for visitors. If you want to have discussion forums for your visitors, then you need to install a discussion forum module. Formulize and Pageworks are both modules for XOOPS.

Multi-entry Forms

Formulize allows designers to specify how users should interact with a form. By nature, some forms need to be filled in more than once; each time a user comes to the form they need to create a new entry. This situation is called a multi-entry form. A typical example

would be an activity log that gets filled in after each activity with information about when, where and how the activity took place.

Permission

The right to perform an action in a XOOPS-based website. The permission system allows webmasters to control what different users can do in the website. For example, the right to access the discussion forum, or the expense report, or the calendar, can each be turned on or off for any group of users. Additionally, specific permissions within each area, or module, can be turned on or off as well. Users can have the ability to view the discussion forum, but not post in it. They can submit their own expense reports, but only managers can review everyone else's expense reports.

PHP

PHP is the programming language that XOOPS is written in. All XOOPS modules, including Formulize and Pageworks, are written in PHP as well.

Radio Button

A type of form element which presents users with a series of options, and they can only choose one. This element is named after the behaviour of buttons on a typical car radio, where you press one button to tune the radio to that particular station, and the previous station is lost. Typically, radio buttons in forms are used for choices with few options, such as Yes/No.

Row

When the information that has been submitted into a form is displayed as a list, it is typically organized into columns and rows. Each column corresponds to one element in the form. Each row corresponds to one entry in the form.

Single-entry Forms

Formulize allows designers to specify how users should interact with a form. By nature, some forms should only be filled in once by users; each time a user comes to the form they simply update the one entry they are allowed. This situation is called a single-entry form. A typical example would be a profile form, or a conference registration form. Each individual is only allowed to record one set of information, one entry, in that form.

Text Box

A type of form element in which users can type whatever they want. For example, a question on a form such as "What is your name?" would typically have a box beside it, and users would be expected to type their name in the box.

Unified Display of Forms

Using the Frameworks feature of Formulize, it is possible to display two or more forms as a single form. This way, a user interacts with one interface, but the information is actually stored separately, and the individual parts of the form can be managed separately.

User

A person who visits a website and does something there. User is a more technical term for "website visitor".

XOOPS

A platform for building interactive websites, particularly well suited to sites that need a range of different information management and sharing tools, such as forums, article lists, file repositories, photo albums, etc. Formulize and Pageworks are modules for XOOPS which add extensive capabilities to create customized applications within the XOOPS-based website. See *Appendix A* for more information.