General Spring Site Documentation

Contents

About this document

This document should provide you with all the needed information to participate in development of the Spring site. I presume you have a technical background and are familiar with open source development methods. I also presume that you have participated to some degree in the project and understand how the Spring project functions.
This document is written from a management perspective.

1. Background

The Spring site was started by Robin Westberg “Fnordia” in 2004 to host the then new Spring project. It consisted mainly of a phpbb2 messageboard installation and a parser to display posts as news and download entry's. There was also a screenshot gallery to show of the project. Later on a MediaWiki installation was added and a Mantis bugtracker. This set-up has been running for several years without major improvements or a roadmap.
The “new” site (developed by Tim Blokdijk) is almost ready to go into production, it's still based on the original site but features theme and language support. Also a more solid development model is applied to prevent stagnation of further site development.

2. Processes

This chapter details the processes used with the Spring site.
These are processes not “rules” they can be bent when needed. Still do try to change the processes, do not break them at will.
The processes are based on “consensus decision-making” in the sense that you should have “sufficient” support for something from the people involved but as a open source project we should always try to give as much room to people to do things differently.
I have the major disadvantages of “consensus decision-making” listed below, it's there to help us identify these problems in a early stage. Do understand that it's still our goal to use “consensus decision-making” don't use this to undermine the process when things don't go like you envisaged them.

From Wikipedia, the free encyclopaedia:

Disadvantages of “consensus decision-making”

- **Preservation of the Status quo**: In decision-making bodies that use formal consensus, the ability of individuals or small minorities to block agreement gives an enormous advantage to anyone who supports the existing state of affairs. This can mean that a specific state of affairs can continue to exist in an organization long after a majority of members would like it to change.
- **Susceptibility to disruption**: Giving the right to block proposals to all group members may result in the group becoming hostage to an inflexible minority or individual. Furthermore,"opposing such obstructive behaviour [can be] construed as an attack on freedom of speech and in turn [harden] resolve on the part of the individual to defend his or her position." As a result, consensus decision-making has the potential to reward the least accommodating group members while punishing the most accommodating.
- **Abilene paradox**: Consensus decision-making is susceptible to all forms of groupthink, the most dramatic being the Abilene paradox. In the Abilene paradox, a group can unanimously agree on a course of action that no individual member of the group desires because no one individual is willing to go against the perceived will of the decision-making body.
- **Time Consuming**: Since consensus decision-making focuses on discussion and seeks the input of all participants, it can be a time-consuming process. This is a potential liability in situations where decisions need to be made speedily or where it is not possible to canvass the opinions of all delegates in a reasonable period of time. Additionally, the time commitment required to engage in the consensus decision-making process can sometimes act as a barrier to participation for individuals unable or unwilling to make the commitment.

Now the idea is to get “sufficient” support for your idea's without running into the problems listed above. Alright? :-)
Another thing to keep in mind, “some people are more equal then others”, with that I mean you have to get support from the people involved but some of those people are more critical to the implementation then others. Use common sense to figure out whose support is critical. At any rate, nobody has a real veto on anything and

everybody can be replaced, it's open source after all.

Now that we have that covered.. the processes:

2.1 Development process

The development process is intended for the implementation of new features and improvements, it's based on Plan-Do-Check-Act. This is the primary way of development. Things developed via this process are almost guaranteed to be used by the project. The patch, via proxy and maintenance processes below are exceptions to this process and are written so that they would fit inside this process.

Plan:
Search for way's to better meet the requirements of chapter 3.
Discuss them with the people involved.
Create awareness that you like to resolve the issue.
Make a realistic plan with a limited scope on how to resolve the problem.
Get feedback on and support for your plan from the people involved.
Do:
Code the feature, update the software, write the text, pimp the graphics or do what you need to do to implement the plan.
DON'T forget to document the changes you made.
Commit the changes and documentation to svn.
Check:
See if things break in the development environment.
Check with the people involved if your work indeed fixed the thing and everybody is happy with it.
Act:
If not go back to *Plan* and improve the original plan or if your plan is fine but your work sucks go back to *Do* and do it right.
If it's good make a new site version and deploy it to the production environment.
Go back to *Plan*.

2.2 Patch process

The “plan first, code later” aspect of the development process is considered an extra barrier for new contributors, this patch process is essentially “code first, plan later” for people that just like to “code something” without having to discuss and plan the changes beforehand. The BIG drawback of this process is that there's no guaranty that anything the new contributor makes will actually end up being useful.
Also this process won't prevent developers implementing the same feature, wasting time on redundant work.

Plan:
Public planning is skipped.
Do:
Code the feature, update the software, write the text, pimp the graphics or do what you like to do.
Ask the Spring developers to look at your changes.
Check:

Attention: *See the note below.*
Hopefully a developers familiar with the development process will take your changes serious and (kick-)start a discussion on how to use your work.

Act:
If this leads to a plan where your work is useful expect to be asked to make improvements to your work, its source files and/or the documentation before it's committed to svn. From here follow the development process from "*Check*" on.
If the developers decided it's not useful your work will not be implemented, maybe you can set up some sort of fork if you believe your work has potential.

2.3 Via proxy process

Not all developers like the idea that they have to keep track and involve themselves in the more formal planning processes. They just like to contribute some (small) things to an aspect of the project without having to spend time on discussions. Still they also like to have some guaranties that there work is actually used. This process would solve that by allowing those "side-developers" to use another developer as a proxy. The proxy developer would be involved with the overall planning via the development process and delegate some of the work to the other side-developer(s).

Plan:
The proxy developer and side-developer discusses what the side-developer is going to do.

Do:
The side-developer codes the feature, updates the software, writes the text, pimps the graphics or do what he/she has to do to implement the things discussed with the proxy developer.
Show the work to the proxy developer.

Check:
Attention: *See the note below.*
The proxy developer checks the work done and gives feedback.

Act:
Improve the work until the proxy developer takes it and uses it inside the overall development process.

Note - *Feedback from a developer at this stage ("check" in the "patch" and "via proxy" process) is his or her personal view on things, it's not the view of all developers involved. A finished "development process" type plan should give an good indication on what the Spring developers like to see done.*

2.4 Maintenance process

The maintenance process is intended for security updates, regression fixes and minor bug fixes, like the development process it's based on Plan-Do-Check-Act but here the *Plan* part is not dependent on more then one person and the *Do* part starts with the production environment. Allowing things to get fixed faster.

Plan:
Find a security issue, regression or minor bug.
Confirm the issue yourself if the report is not coming from a upstream project or a Spring webmaster.
Check if there's a fix (upstream patch) already available.
Create awareness that you like to resolve the issue, keep this limited to trusted people if it's a security issue.

Do:
Don't make a plan but do think things over, especially when you have to alter data (code can be fixed later, data is generally lost).
Make sure you have a data backup (especially) if you don't feel rested or are unsure about the fix.
Fix the issue in the **production** environment.
Check if the fix works in the production environment.
Create awareness that the issue is fixed.
Fix the issue in svn head.
DON'T forget to document the changes you made.
Commit the changes and documentation to svn.
Check:
See if things break in the development environment.
Check with the people involved if your work indeed fixed the thing in the development environment and if everybody is happy with it.
Act:
If not go back to *Plan* or if the plan part is fine but your fix sucks go back to *Do* and do it right.
If it's good take a beer/cola/applejuice and chill.

2.5 Translation process

The translation process helps people to make a translation of the site in there local language.

???

There is a translation page on the site that allows people to translate the strings.
Process so far is to just go for it.
Ask around if your language has to be added to the translation system.

2.6 Theme process

The site can be “themed”, there are no real guidelines for it at this moment so ask around to discuss the addition of a theme to the site.
The idea is to have a theme per Spring mod.
Technically a lot is possible with themes (you can change everything you like) but we still have to figure out what should be shown across all mods.

3. Main requirements of the site

The project develops and maintains a site as it provides a central hub where people can gather to play and work together on Spring (and related things).
Ideally the site should provided (or link) everything needed for this.

To make this rather broad goal more specific we split the community in two main groups: Spring Users and Spring Contributors and again split these up in even more specific groups and their site requirements. Every group has a short introduction to explain who the people are in the group. The groups are generalisations and people

will always be part of more then one group. So you can be a writer, a code developer and a casual user at the same time.

3.1 Spring Contributors

Spring Contributors are the people that keep the project running by providing their time and skills.

The site should provide the contributors with documentation and tools to work in the Spring project. The second thing the site should provide is a easy way for contributors to communicate with the other developers without to much noise from other people.
Spring contribution should always be as transparent as possible to prevent barriers.

This group is split in three major sub-groups, Content Developers, Code Developers and Moderators.

3.1.1 Content Developers

Mappers, Unit developers, Webmasters, everything that requires graphics, a layout or writing.

Needs to be clear what projects need help and what the goals are, work flow, etc.
Documentation on how the engine, tools and site work.
Easy way to give their work exposure on the site!

3.1.1.1 Writers

To be expanded after feedback from the people involved.

3.1.1.2 Mappers

To be expanded after feedback from the people involved.

3.1.1.3 Unit (mod) Developers

To be expanded after feedback from the people involved.
The site has theme support to allow the layout and graphics to match the mod.

3.1.2 Code developers

To be expanded after feedback from the people involved.

3.1.2.1 Engine coders

To be expanded after feedback from the people involved.
Bugtracker, private communication, Fisheye, Doxygen, StackTrace Translator, WebSVN.

3.1.2.2 Webmasters

To be expanded after feedback from the people involved.

Production environment & development environment.

3.1.3 Moderators

To be expanded after feedback from the people involved.

They keep everybody in line, the focus is on developing and playing a game, the Spring project is not the local support group. If someone is behaving like a asshole/bitch we won't like to have them around. The moderators are here to help people with that.

The site should enable moderators to track individuals, sensor them and if needed ban them. Rules should be easily to find and be updated by the right people. Moderators need a private place to discuss specific issues.

3.2 Spring Users

Spring Users (or Spring Player/Gamer) are the people that play Spring as entertainment and as a sport.

This group needs easy access to the latest stable Spring release and the content that is available for this release. They generally like to keep themselves informed about the major developments in and around the project. Last but not leased is the social aspect, a lot of people play Spring as they like the interaction with others.
Another big thing that we should always try to do is interest the Spring Users in the development side of the project and remove any barrier that might prevent people from becoming an active Spring Developer.
These people should be made aware that Spring is a special type of game, we are an open source project and they need to be aware that this has consequences for “the way things work around here”.

Spring Users are again split up in several groups.

3.2.1 New Users

New Users are a bit special, they are new to the Spring project/game/community but generally have played a few RTS games before.

We should show them that Spring is a cool, active project and provide them with easy access to documentation that helps them install and play Spring. Once they have concluded that Spring is indeed a cool project they should find all the information they might need to involve themselves in the project.
Understand that people with developer skills generally first try out the game but then quickly move on to the development side and don't care about actually playing the game that much any more. This New User --> Developer transition should be easy to make.

3.2.2 Casual Users

These people are playing Spring because they like it as a fun pastime, they don't involve themselves with the project. It's the thing they happen to play with friends.

They like to know about new versions of the software and that is about it.

The site should provide them with the latest stable version and try to interest them to look beyond the entry page.

3.2.3 Kids

Another special case are children (less then 16 years old), they generally don't have much (RTS) gaming background and also have little understanding of open source projects and the way we work. Still many have free time to play (and contribute to) Spring.

The site should make it easy for them to join, providing the site in there native language is a big thing for them as English can become a big barrier. Contributing to Spring more or less requires English at this point, so it would probably be rare to see non-native English speakers becoming contributors in this age group. Native English speakers still might end up contributing, (bug reporting, writing etc.) don't exclude them by having age or other requirements that can only be expected from adults. (international bank account, credit card, signing legal agreements, etc.)

Another important thing is to keep the site “family friendly”. :-)

3.2.4 Dedicated Users

These are the people that like to think of gaming as a sport.

They like to have a league, join a group/clan and learn everything there is to know about the game.
They can become very valuable contributors by balancing units so the site should try to link them up in this process.

4. Implementation

This chapter explains what the Spring project plans to do, is doing and has done to satisfy the requirements set in chapter 3.

4.1 Design philosophy

The design philosophy behind the site is to be the home of everything that is (related to) Spring that way there are no barriers between the content developers, code developers, moderators and the users. It allows people to easily participate on every level of the project and coordinate changes that affect a lot of different aspects at the same time.

Much thought need to be put into the structure and design of the site to accommodate all the different needs. To achieve this the planning, development and maintenance processes need to be transparent and all people that use the site need to be actively involved in this.

Although the site is a big “all in one” thing it still has to be split up in functional sections to allow the the specific needs of people to be met. The sections need to be decided by looking at the requirements set for the different groups in chapter 3.
The most clear two general sections are the things that are targeted at the Spring Users and those that are for the Spring contributors.

4.2 Roadmap

The site roadmap should always try to be in sync with the general Spring roadmap.

Planed for the next version is involving more people in the site development process, to get the requirements in chapter 3 clearer and to expand this roadmap.

4.3 Technical design

This section describes the current technical implementation.
For this the technical documentation documents are linked in this document if you are reading the odf (not the pdf) version of this document then those files would not show as the file path to them is hard coded.
If you like to update the text in this section you have to open the linked odf document in the “Technical Documentation” sub-directory.
If you need to add a segment to this section copy a existing document in that directory and change the text to document your topic. Then page Tim Blokdijk (the current maintainer of this document) to include it.

4.3.1 Translation system

The functions.php file in the includes directory has a site_language class and a few functions that provide the code to load and save translations from and to the database (site_language table).
You can get a $text array from it with the language you requested for a specific section of the site, English is the fall-back language.

The transition.php file in the www-root directory facilitates the translators with a tool to make translations for the site sections.

Translations have some limited mark-up options:
[url=link_to]link text[/url] for hyperlinks.
[b]Bold[/b] for bold text.
[i]Italics[/i] for italics.
And [u]under-line[/u] to underlining things.
It's based on bbcode (used by phpbb).

We don't use/allow HTML as this makes translations easy to do for people without HTML knowledge, it prevents JavaScript, object and img injection and it moves mark-up more to the theme.

The system uses UTF-8 so all known languages should be supported.
Displaying text “right to left” is not possible at this time.

4.3.2 Site Configuration

Site configuration details are set in configuration.php one level below www-root.
This file makes a $site_configuration multidimensional array (array with array's in it) with the configuration options. Using only one array would (hopefully) prevent namespace problems with external site software (phpbb, mediawiki and the likes). It's included (require_once) with each page request. The configuration.php file will never

have any output to client.

The configuration.php file is private and should never be made public, we commit a configuration.example.php to svn with dummy values and explanation on how to use it.

All other site code (including external site software) would be configured with the values from this array.

4.3.3 PhpBB

We currently have a PhpBB 2 messageboard installation.
It has the standard SubSilver template with Springs site header integrated.

Mods applied:

http://www.phpbb.com/mods/db/index.php?i=misc&mode=display&contrib_id=1181

```
#############################################################
## MOD Title: Today At/Yesterday At
## MOD Author: netclectic < adrian@netclectic.com > (Adrian Cockburn) http://www.netclectic.com
## MOD Description: Will show Today At if the post was posted today
##                  Will show Yesterday At if the post was posted yesterday
##
## MOD Version: 1.3.1
##
## Installation Level: easy
## Installation Time: 10 Minutes
## Files To Edit: (6) page_header.php, index.php, search.php, viewforum.php, viewtopic.php, lang_main.php
## Included Files: n/a
#############################################################
```

4.3.4 MediaWiki

Changes to the MediaWiki installation code wise are in a skin SpringNew.php found in the skin sub-directory and in LocalSettings.php
SpringNew.php contains some things to show the header.
There is also a spring.css in the skins/common directory.
LocalSettings.php has the configuration of the wiki installation, database details are retrieved from the global Spring site configuration file.
An alias is configured in the Apache configuration to point /wiki to /w
The images in the images sub-directory are not stored in the development environment, make sure to back-up these images on the production environment before replacing it with the wiki installation from the development environment.

4.3.5 Mantis

At this time we have no documentation for the Mantis bug tracker installation.

4.4 Version History

This gives a general run-down on the changes made to the site over time.

4.4.1 Version 1

Version 1 is the original site as it started with the project. Documentation did not exist including version tracking.
Robin Westberg “Fnordia” made changes to the production site as needed.

External software used:

Name	Version	Website
PHPBB	v. 2.0.21	http://www.phpbb.com/
BBCode Parser	v. ?	http://www.christian-seiler.de/projekte/php/bbcode/
MediaWiki	v. 1.10.0	http://www.mediawiki.org/
Mantis	v. ?	http://www.mantisbt.org/
Galerie.php	v. ?	http://cker.name/galerie/

4.4.2 Version 2

Tim Blokdijk made a new site and integrated this with the existing site (version 1).
A development environment was set-up and the site itself was put in svn.
The main new features of the site are language and theme support. The site has a new layout and all text is rewritten. It's easier to update the site with new information. Documentation has been written to make site development with a team manageable. An new entry page has been added it should have everything a new user needs to get started. The focus of this version was on improvements for the Spring Users.

Version 2 drops the dysfunctional “upload files” function and moves away from gallerie.php in favour for a MediaWiki page.

At the same time Tom Nowell “AF” had made a site based on Joomla but decided to drop this effort in favour for collaboration.

External software used:

Name	Version	Website
PHPBB	v. 2.0.22	http://www.phpbb.com/
BBCode Parser	v. ?	http://www.christian-seiler.de/projekte/php/bbcode/
MediaWiki	v. 1.10.0	http://www.mediawiki.org/
Mantis	v. ?	http://www.mantisbt.org/

4.4.3 Version 2.1

Next version, see the roadmap for details.